ELEMENTE DE INTELIGENTA ARTIFICIALA

1. Introducere.

1.1. Prelucrarea de imagini - principii generale.

Prelucrarea de imagini este un domeniu care îşi păstrează dinamismul în ciuda trecerii anilor. Dezvoltările tehnologice au facilitat accesul unui număr tot mai mare de oameni la această ramură fascinantă a imagisticii computerizate.

Conţinutul imagisticii computerizate a fost sintetizat (Fig.1) în lucrarea lui Pavlidis [79], lucrare fundamentală pentru începuturile prelucrării de imagini.

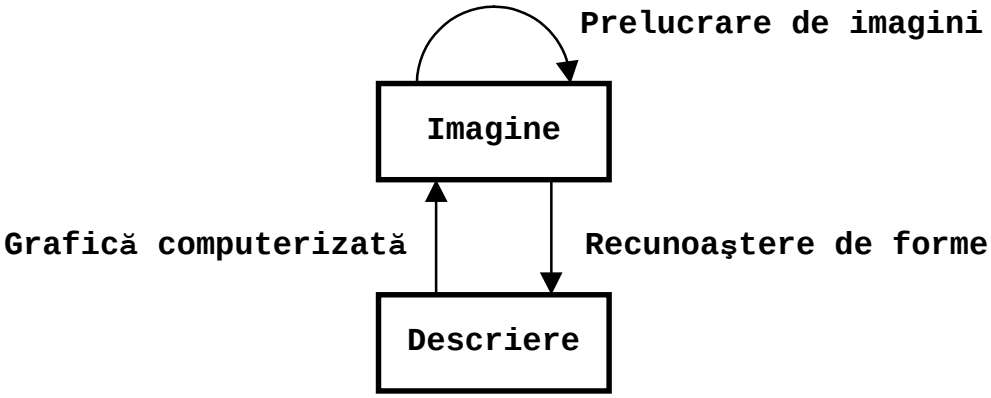


Fig. 1. Conţinutul imagisticii computerizate.

În aceeaşi lucrare, Pavlidis propune clasificarea imaginilor în patru clase, astfel:

- **Clasa (1)** de imagini include imaginile în scară de gri (sau color).
- **Clasa (2)** de imagini cuprinde imaginile binare (sau cu câteva culori).
- **Clasa (3)** de imagini include imagini formate din linii şi curbe continue.
- **Clasa (4)** de imagini cuprinde imagini formate din puncte izolate sau poligoane.

Se remarcă scăderea complexităţii imaginii odată cu numărul clasei, simultan cu reducerea semnificativă a volumului de date necesar pentru stocarea lor (Howe[51]).

Astfel imaginile de clasă **(1)** sunt cele mai complexe, un exemplu tipic de astfel de imagini fiind cele de televiziune, obținute deci cu o cameră TV. Imaginile binare conțin doar două nuanțe, de obicei alb şi negru (sau doar câteva culori în cazul imaginilor color). Clasa **(3)** cuprinde imagini şi mai simple, adică conținând doar linii şi curbe având grosimea de un pixel. Imaginile din ultima clasă sunt cele mai simple, ele fiind formate din puncte izolate sau, în cazul graficii computerizate, din poligoane care descriu corpul tridimensional a cărui reprezentare realistă se doreşte în final.

Câteva exemple de imagini din diferite clase sunt date în continuare (Fig. 2):

Fig. 2. Exemple de imagini din diferite clase.

O taxonomie a domeniului prelucrării de imagini în corelație cu grafica computerizată este propusă în continuare (Fig. 3):

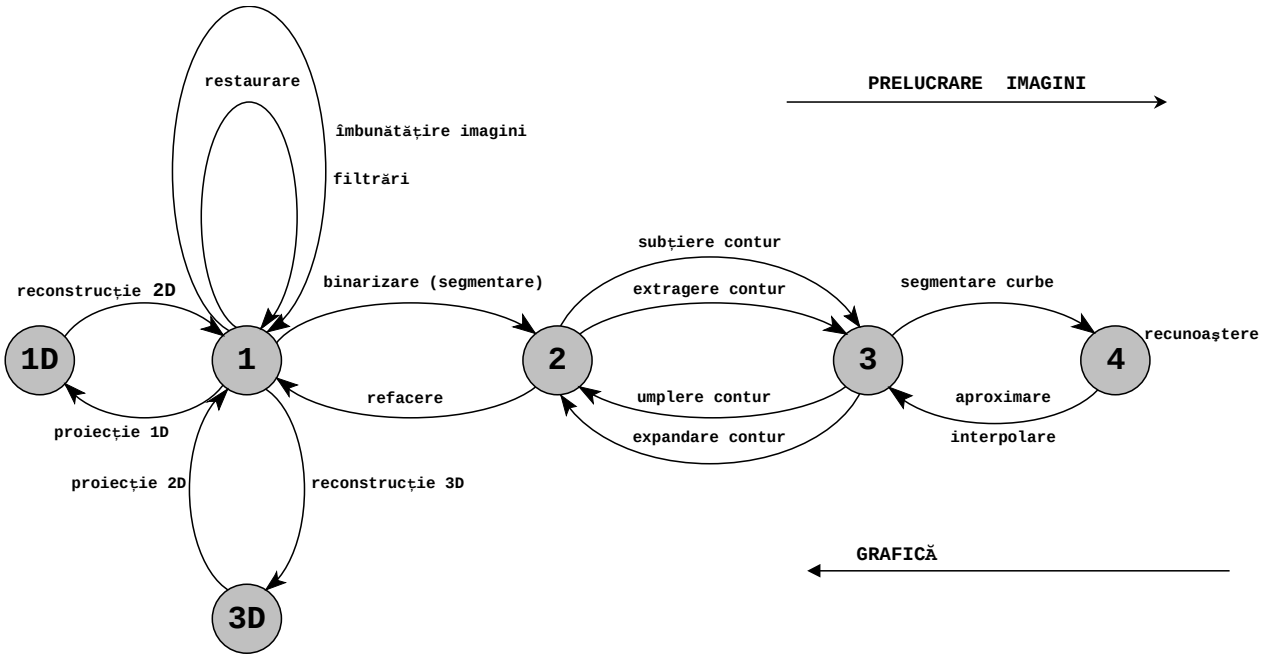


Fig. 3. O taxonomie a imagisticii computerizate.

Nu au fost reprezentați în diagrama anterioară algoritmii de codare/decodare a imaginilor, care formează un subdomeniu destul de consistent al prelucrării de imagini.

Î*mbunătățirea imaginilor* şi *filtrările* formează un domeniu care se ocupă de eliminarea zgomotelor, îmbunătățirea contrastului, accentuarea muchiilor şi detecția de muchii.

Restaurarea imaginilor îşi propune eliminarea distorsiunilor care afectează imaginea, distorsiuni datorate unor fenomene fizice cunoscute, modelate matematic sau estimate.

Segmentarea imaginilor realizează separarea regiunilor uniforme, de interes, din imagine. Uniformitatea este o noțiune generală ea nereducându-se la constanța nivelelor de gri (aceeaşi textură, aceleaşi proprietăți, etc.). *Binarizarea* furnizează la ieşire o imagine de clasă **(2)**, ea putând fi asimilată unei operații particulare de segmentare (de exemplu segmentarea cu prag).

Algoritmii de *extragere de contur* furnizează la ieşire un contur închis, deci o imagine de clasă **(3)**. Algoritmii de subțiere de contur realizează tot transformarea imaginilor de clasă **(2)** în imagini de clasă **(3)**, dar furnizează la ieşire un graf.

Imaginile de clasă **(3)** pot fi simplificate în continuare cu algoritmi de *segmentare a curbelor*, care furnizează localizarea aşa-numitele puncte critice şi, în plus, parametrii curbelor care aproximează/interpolează liniile şi curbele din imagine între două puncte critice succesive. Se poate remarca că, practic, se obține în această fază o *descriere* extrem de simplificată a imaginii inițiale.

În cazul graficii computerizate, după cum s-a menționat anterior, se pleacă de la o *descriere* a imaginii, urmărindu-se în cazul cel mai general, *sinteza unei imagini* realiste. Descrierea inițială cea mai des folosită conține o colecție de poligoane împreună cu conexiunile dintre ele, ceea ce formează aşa-numita descriere "cadru-de sârmă" (wire-frame).

Obținerea imaginii realiste (clasa **(1)**) se traduce printr-o secvență de algoritmi care "apropie" imaginea sintetizată de cea reală.

Algoritmii de *aproximare/interpolare* pleacă de la descrierea "cadru-de-sârmă" şi urmăresc obținerea unei curbe netede pentru suprafețele corpurilor tridimensionale ce se reprezintă.

Algoritmii de *umplere de contur* realizează operația complementară extragerii de contur, în timp ce *expandarea* este operația complementară subțierii.

Cunoscând sau deducând regulile de variație a nivelului de gri (culorii) de-a lungul suprafețelor care mărginesc corpul, prin operația de *refacere* se poate obține o reprezentare bidimensională realistă a corpurilor studiate. Se ține cont de caracteristicile şi poziția surselor de lumină şi caracteristicile locale ale corpurilor care se reprezintă (reflexivitate, transparență, etc.).

Alte clase de algoritmi sunt specifice *tomografiei computerizate*, care realizează investigarea nedistructivă a corpurilor.

Corpul studiat este penetrat de un fascicol de unde (raze "X", radiații electromagnetice, ultrasunete, etc) iar în prelungirea lor (mai puțin în cazul ultrasunetelor, care măsoară unda reflectată) un set de receptori măsoară energia rezultată a radiației, energie dependentă direct de proprietățile locale de absorbție ale corpului investigat. Ceea ce se obține la nivelul receptorilor este *proiecția 1D* (deci un vector) a unei secțiuni plane a corpului investigat, într-o anumită direcție.

Reconstrucția 2D reface secțiunea plană a corpului studiat dintr-un set de proiecții 1D, în direcții diferite, ale secțiunii.

Având la dispoziție mai multe asemenea secțiuni (paralele sau radiale) ale corpului studiat, se poate realiza *reconstrucția 3D* a corpului studiat. Operația inversă se numeşte *proiecție 2D*.

Trebuie de remarcat însă că această sinteză nu trebuie văzută rigid, existând în literatură o multitudine de algoritmi care nu pot fi încadrați în ea şi, în plus diferiți autori încadrează diferit un acelaşi algoritm în categorii diferite. Ca exemplu, operația de codare a datelor poate fi văzută ca o extragere de trăsături regenerative, ceea ce se obține fiind în esență o descriere a imaginii, scopurile urmărite fiind însă diferite.

1.2. Teoria recunoaşterii formelor - generalități.

Recunoaşterea formelor (Meisel[72]), (Sklansky[102]), (Vancea[108]) se ocupă de clasificarea unui set de obiecte, procese sau evenimente. Clasificarea este un proces fundamental

care caracterizează nu numai ştiinţele ci şi viaţa socială. Metodele matematice dezvoltate în cadrul recunoaşterii formelor îşi găsesc aplicaţie în cele mai diverse domenii.

Recunoaşterea formelor are ca scop determinarea clasei din care fac parte elementele unei mulţimi. Stabilirea numărului de clase este o problemă legată direct de caracteristicile aplicaţiei. Pot exista aplicaţii cu număr de clase cunoscut aprioric, dar şi aplicaţii în care numărul de clase trebuie stabilit algoritmic.

Clasificatorul este sistemul care implementează operaţia de recunoaştere a formelor, deci el realizează o procedură stabilită anterior de clasificare.

Există două abordări ale procesului de recunoaştere: recunoaşterea controlată şi recunoaşterea necontrolată.

Recunoaşterea controlată (supervizată) implică existenţa unui set de forme a căror apartenenţă la clase este cunoscută. Acest set de forme include setul de formare (învăţare, antrenare) care este folosit pentru construcţia propriu-zisă a clasificatorului şi setul de predicţie (testare) al cărui scop este testarea (evaluarea) clasificatorului respectiv. Construirea clasificatorului este asociată în acest caz algoritmului de învăţare corespunzător clasificatorului, algoritm care construieşte în mod iterativ coeficienţii acestui clasificator.

Recunoaşterea necontrolată (nesupervizată) nu necesită cunoaşterea apriorică a apartenenţei formelor din setul de formare la clase (Hartigan[46]). Această abordare dezvoltă algoritmi care realizează construcţia claselor pe măsură ce formele analizate sunt luate în considerare. Ei sunt numiţi algoritmi de grupare (clustering).

Schema bloc a unui sistem de recunoaştere a formelor este dată în continuare (Fig. 4):

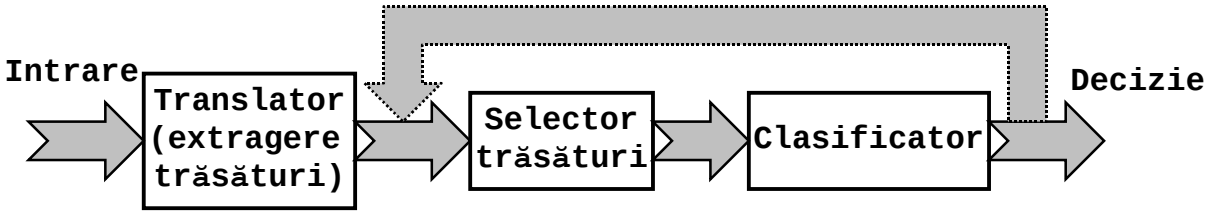


Fig. 4. Schema bloc a unui sistem de recunoaştere a formelor.

1.2.1. Principii generale.

În urma extragerii trăsăturilor se obţine un set de "*n*" trăsături notat:

$$\boldsymbol{X} = (x_1, x_2, ..., x_n)^T \tag{1}$$

care poate fi văzut ca un vector în spaţiul trăsăturilor (formelor), spaţiu notat cu $\boldsymbol{\Omega} \subset \boldsymbol{R}^n$.

Clasificarea este o partiţionare a spaţiului formelor în "*K*" regiuni (clase), notate $\{\omega_j\}_{j=1...K}$ şi care îndeplinesc condiţiile:

$$\omega_1 \cup \omega_2 \cup ... \cup \omega_k = \boldsymbol{\Omega} \tag{2}$$

$$\omega_1 \cap \omega_2 \cap ... \cap \omega_k = \boldsymbol{F} \tag{3}$$

unde $\boldsymbol{F}$ este mulţimea punctelor care alcătuiesc frontierele între aceste clase.

Funcţia discriminant ataşată unei clase este o funcţie $\boldsymbol{D}_j(\boldsymbol{X}): \boldsymbol{\Omega} \to \boldsymbol{R}$, dată de:

$$\boldsymbol{X} \in \omega_j \Leftrightarrow \boldsymbol{D}_j(\boldsymbol{X}) > \boldsymbol{D}_i(\boldsymbol{X}),\ \forall i \neq j,\ i = 1...K \tag{4}$$

Limitele de decizie între clasele ω_i şi ω_j vor fi definite prin ecuaţia:

$$\boldsymbol{D}_i(\boldsymbol{X}) - \boldsymbol{D}_j(\boldsymbol{X}) = 0 \tag{5}$$

Metodele matematice utilizate pentru rezolvarea problemelor de recunoaştere a formelor se grupează în două mari categorii:

- metode decizional-teoretice şi statistice;
- metode sintactice (lingvistice).

Între cele două clase de metode există o conexiune directă care asigură întrepătrunderea lor, după cum rezultă şi din următoarea schemă bloc care descrie modurile posibile de abordare ale unei probleme generale de clasificare:

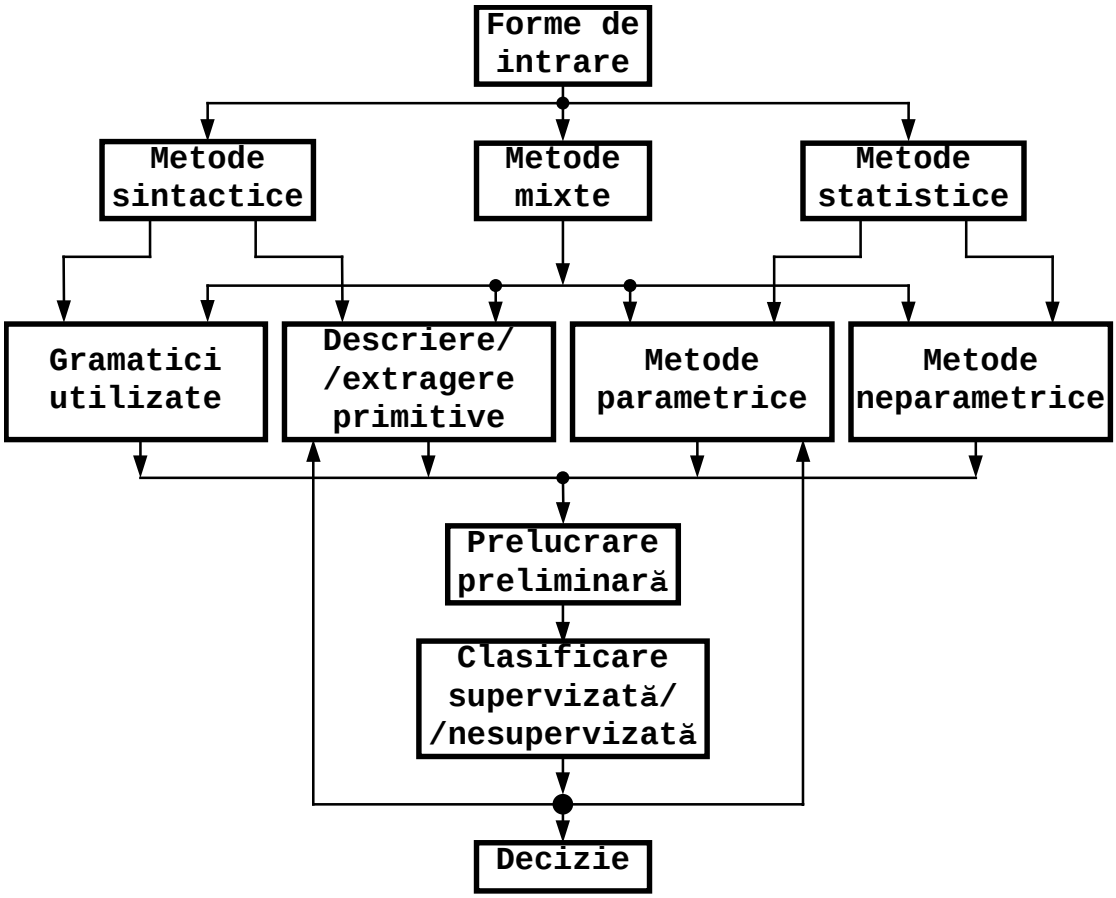


Fig. 5. Moduri posibile de abordare ale unei probleme de clasificare.

Dintre proprietățile care permit evaluarea unui clasificator, mai importante sunt următoarele:

Recunoaşterea este exprimată prin *rata de recunoaştere*, care este procentul de forme din setul de formare recunoscut corect de clasificator.

Convergența exprimă *viteza de răspuns* a unui clasificator. În cele mai multe cazuri se urmăreşte realizarea unui compromis între rata de recunoaştere şi viteza de răspuns care caracterizează un anume clasificator.

Siguranța este descrisă prin g*radul de încredere* al unui clasificator, care caracterizează capacitatea clasificatorului de a clasifica corect formele distorsionate aplicate la intrare.

Predicția exprimă capacitatea clasificatorilor de a recunoaşte corect forme care nu aparțin setului de formare. O măsură a acestei proprietăți este *abilitatea predictivă* care exprimă procentul de forme din setul de predicție (deci cu apartenență la clase necunoscută în prealabil) recunoscute corect.

1.2.2. Strategii de recunoaştere pentru N clase.

Complexitatea problemei poate fi micşorată împărțind-o în mai multe sarcini mai mici, şi anume alcătuind câteva *grupe de clase* şi apoi construind câte un clasificator pentru fiecare grupă de forme (Bulea[14][21][23]), (Pudil[88]). Trăsăturile extrase se aplică simultan la intrările tuturor clasificatorilor şi în mod normal unul singur va furniza decizia de recunoaştere. Rolul logicii de decizie este de a furniza o decizie corectă în cazurile cu o oarecare doză de

incertitudine, utilizând informațiile furnizate de toate clasificatoarele. Schemele bloc pentru două variante ale unui asemenea sistem sunt prezentate în continuare.

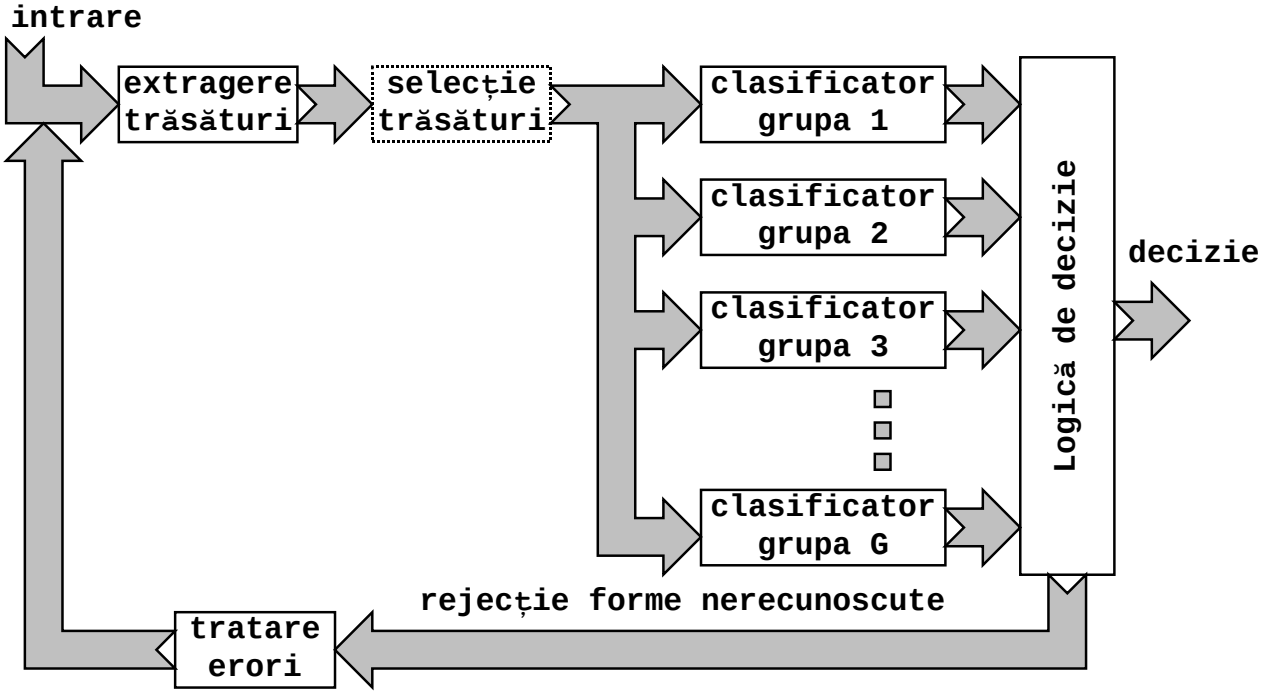


Fig. 6. Sistem de recunoaştere pentru N clase, varianta 1.

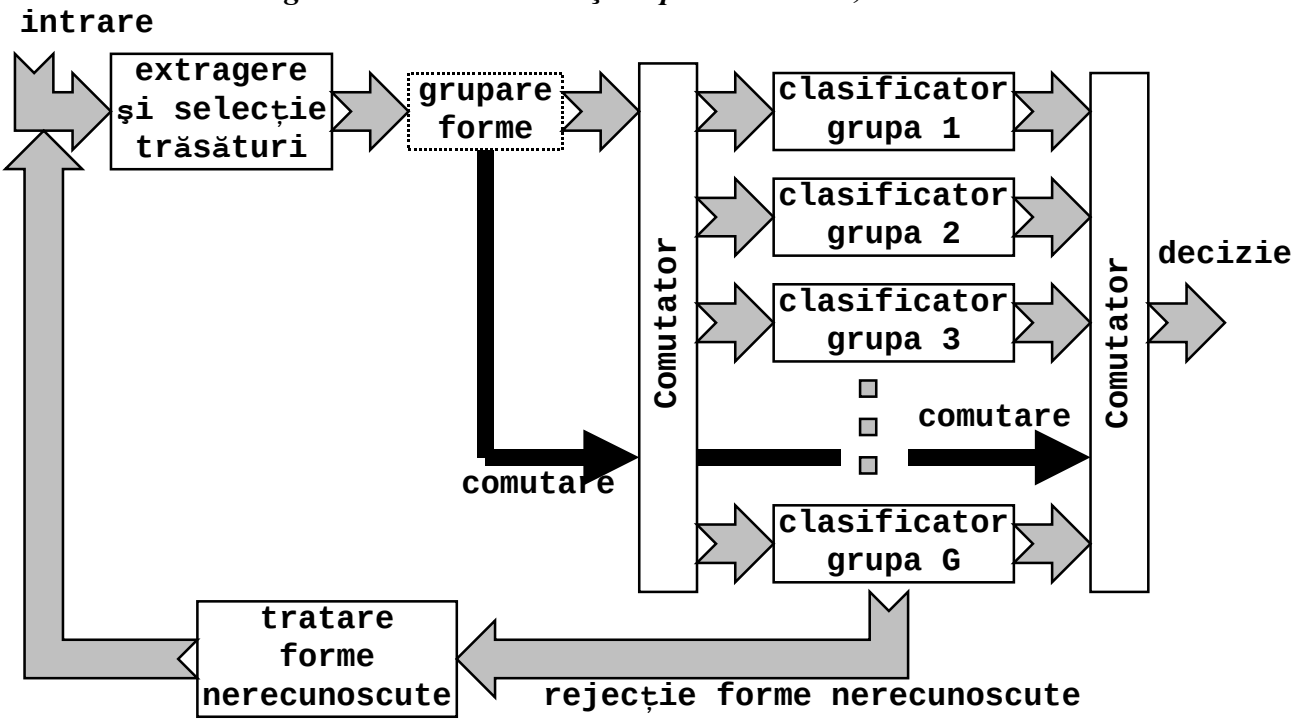


Fig. 7. Sistem de recunoaştere pentru N clase, varianta 2.

Deşi prima variantă este mai simplă, se poate utiliza, pentru optimizarea timpului de răspuns, un preclasificator care să specifice cărei grupări îi aparține fiecare formă de intrare şi în consecință, care clasificator elementar trebuie folosit pentru recunoaşterea efectivă.

De multe ori, orice sistem de recunoaştere include şi un subsistem de tratare a erorilor de recunoaştere (forme nerecunoscute), un exemplu tipic in acest sens fiind cazul caracterelor atinse sau fragmentate din sistemele OCR (Optical Character Recognition).

În ambele cazuri, pentru formele nerecunoscute se mai face o încercare în ipoteza că ele reprezintă forme încă "neînvățate"; se încearcă reantrenarea clasificatorului, apoi rezultatul este iarăşi aplicat la intrarea clasificatorului pentru o nouă încercare.

1.3. Metode conexioniste în prelucrarea de imagini şi recunoaşterea de forme.

Metodele conexioniste se bazează pe utilizarea unei rețele de "procesoare" conectate intensiv unele cu altele. Fiecare "procesor" poate executa un număr redus de operații, de obicei aceleaşi pentru toate procesoarele din rețea. Gradul lor de programabilitate este redus. Implementarea acestor rețele de procesoare impune ca o necesitate practică limitarea conexiunilor aferente fiecărui procesor, astfel încât el să fie conectat doar la un număr de procesoare vecine.

Pot fi încadrate în categoriile descrise anterior structurile sistolice, procesoarele matriceale, rețelele neurale artificiale, etc. (Conte[31]).

Structurile sistolice fac parte din categoria mai largă a maşinilor paralele complexe care sunt alcătuite dintr-un mare număr de unități procesoare, de obicei identice, fiecare fiind capabilă să execute o operație fixă de bază. Proprietățile arhitecturale ale structurilor sistolice le fac potrivite pentru o implementare VLSI. Conectarea celulelor de calcul se face conform unei topologii regulare, în care fluxul de informații este permis numai spre unitățile adiacente într-un mod "pipe-line". Necesitățile de comunicare cu exteriorul sunt limitate, conexiunile sunt foarte scurte, ceea ce permite o viteză mare de operare.

Procesoarele matriceale sunt structuri paralele care realizează aceleaşi operații, simultan, asupra unor date diferite. Ele constituie exemplul tipic de structuri SIMD (**S**ingle **I**nstruction Stream - **M**ultiple **D**ata Stream). Au un grad mai mare de programabilitate decât structurile sistolice, dar utilizarea lor este limitată la aplicații care cer un paralelism înalt, cum ar fi lucrul cu matrici de date de dimensiuni mari (cazul imaginilor).

Rețelele neurale (neuronale) artificiale (Haykin[47]), (Kohonen[62][63]), (Lipmann[66]), (Rumelhart[98]) încearcă să se apropie de modelul creierului uman. Spre deosebire de maşinile Von--Neuman care se caracterizează prin existența unei unități procesoare care execută instrucțiuni stocate în memorie, într-o secvență aflată sub controlul numărătorului de program, alte arhitecturi propuse încearcă să exploateze cât mai eficient paralelismul care este de obicei inerent. "Procesoarele" care formează rețelele neuronale, sunt denumite neuroni artificiali.

Dacă majoritatea calculatoarelor existente în momentul de față dispun de o singură unitate procesoare, extrem de puternică şi de rapidă, la cealaltă extremă din punctul de vedere al structurii interne se plasează aceste rețele neurale artificiale, caracterizate printr-o simplificare extremă a unităților componente, alături de o extindere cât mai largă a conexiunilor între aceste unități procesoare.

Orice rețea neurală este caracterizată de trei elemente: modelul neuronului, arhitectura rețelei şi algoritmul de antrenare folosit.

În ceea ce priveşte modelele de neuroni, cel mai mult folosite în momentul de față sunt cele fără memorie, deci care implementează o relație de forma:

$$y_j = \boldsymbol{f}\left(\sum_{i=1}^{N} w_{ij} x_i - \theta_j\right) \tag{6}$$

unde y_j este ieşirea neuronului "*j*", x_i este intrarea "*i*" a neuronului, w_{ij} este ponderea conexiunii de la intrarea "*i*" la neuronul "*j*", iar θ_j este pragul ataşat neuronului "*j*".

Funcția $\boldsymbol{f}$ este o funcție neliniară care poate fi de tipul limitare hardware, prag logic, sigmoidă simetrică sau nu, sau chiar funcția identică.

Modele mai sofisticate de neuroni introduc variabila timp, un astfel de model fiind descris de următoarele relații:

$$\begin{cases} \dfrac{du_j}{dt} = -u_j + \sum_{i=1}^{N} w_{ij} x_i + \theta_j \\ y_j = \boldsymbol{f}(u_j) \end{cases} \tag{7}$$

unde u_j caracterizează starea curentă a neuronului.

Între arhitecturile de rețele neurale, rețelele "feed-forward" (conexiunile sunt unidirecționale, nu există bucle de reacție) au fost cel mai mult studiate. Într-o asemenea rețea, neuronii sunt dispuşi în straturi succesive, ieşirile neuronilor de pe straturile inferioare aplicându-se la intrările neuronilor de pe stratul imediat următor.

Binecunoscutul perceptron multistrat are o asemenea structură, gama largă a aplicațiilor în care o fost folosit fiind o dovadă a capabilităților unui asemenea sistem

Teoria recunoaşterii formelor este deja o disciplina matură, care şi-a dovedit utilitatea într-o multitudine de domenii. Pe de o parte, paralelismul extins oferit de rețelele neurale deschide noi perspective teoriei recunoaşterii formelor, iar pe de alta, fundamentul matematic solid al teoriei recunoaşterii formelor direcționează şi dinamizează în permanență cercetările asupra retelelor neurale. Se poate remarca deci în acest context interdependenta strînsa între cele două domenii de studiu.

Principala calitate a rețelelor neurale este aceea de clasificare. Clasificatorii neuronali mai des folosiți, sunt prezentați în următoarea taxonomie, funcție de tipul intrărilor (binare sau continue), funcție de modul de antrenare (supervizată sau nesupervizată) şi de arhitectura rețelei:

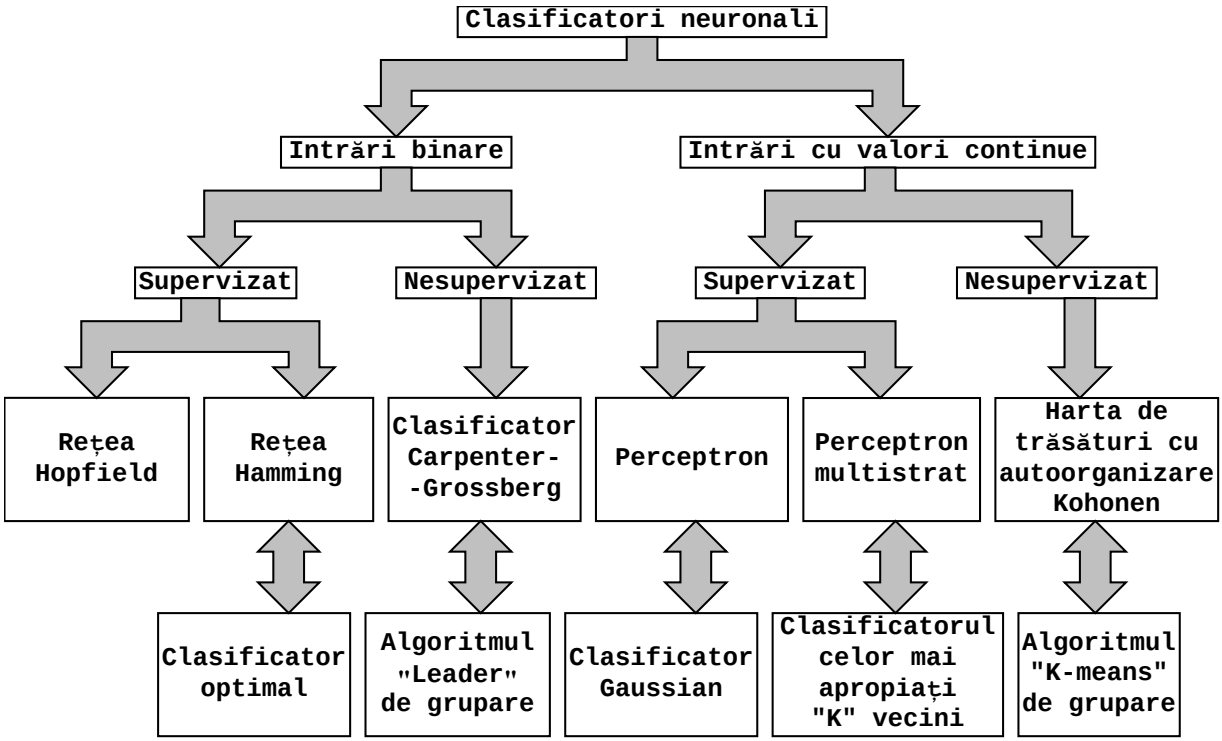


Fig. 8. O taxonomie a rețelelor neurale.

Blocurile din partea de jos a schemei anterioare indică algoritmii clasici cei mai apropiați de clasificatorul neural corespunzător.

Domeniul rețelelor neurale artificiale cunoaşte în prezent un dinamism remarcabil (Freeman[39]), (Haykin[47]) extinzându-se rapid nu numai rezultatele teoretice, ci şi aplicațiile care folosesc aceste metode.

2. Tehnici de îmbunătățire a imaginilor.

2.1. Eliminarea zgomotelor.

2.1.1. Filtre spațiale de eliminare a zgomotelor.

Tehnicile spațiale de eliminare a zgomotelor se remarcă prin simplitate, ceea de duce la posibilitatea unor implementări hardware în timp real. Ele se bazează pe folosires aşa-numitelor măşti de convoluție care, în esență, realizează înlocuirea valorii fiecărui pixel cu o combinație liniară a valorilor pixelilor dintr-o vecinătate a pixelului curent.

a) Filtre de mediere.

Pentru fiecare vecinătate de pixeli (de obicei 3x3, dar şi 5x5 sau 7x7), pixelul central se înlocuieşte cu o combinație liniară a pixelilor din vecinătate:

$$\tilde{P}(i,j)=\frac{1}{k}\sum_{m=-1}^{1}\sum_{n=-1}^{1}M(m,n)*P(i+m,j+n) \qquad (1)$$

unde masca $M(m,n)$ poate fi:

$$\boldsymbol{M}_1=\frac{1}{4}\begin{bmatrix}0&1&0\\1&0&1\\0&1&0\end{bmatrix} \quad \boldsymbol{M}_2=\frac{1}{5}\begin{bmatrix}0&1&0\\1&1&1\\0&1&0\end{bmatrix} \quad \boldsymbol{M}_3=\frac{1}{6}\begin{bmatrix}0&1&0\\1&2&1\\0&1&0\end{bmatrix}$$

$$\boldsymbol{M}_4=\frac{1}{8}\begin{bmatrix}1&1&1\\1&0&1\\1&1&1\end{bmatrix} \quad \boldsymbol{M}_5=\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix} \quad \boldsymbol{M}_6=\frac{1}{10}\begin{bmatrix}1&1&1\\1&2&1\\1&1&1\end{bmatrix} \quad \boldsymbol{M}_7=\frac{1}{16}\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix} \qquad (2)$$

Asemenea tip de filtre se pot rafina astfel încât să nu fie afectaţi de operaţia de mediere pixelii situaţi pe muchii. Un operator pentru detecţie de muchii (mască) poate indica prezenţa şi direcţia unei muchii în imagine. În absenţa muchiei, se aplică filtrul de eliminare a zgomotelor pentru toţi pixelii din vecinătate, în caz contrar doar pentru pixelii situaţi de o parte şi de alta a muchiei, cunoscând direcţia ei. De asemenea se pot utiliza măşti de mediere pentru toţi pixelii, însă având coeficienţii corelaţi cu datele furnizate de detectorul de muchii.

b) Medierea cu prag realizează compararea fiecărui pixel cu media vecinilor lui, medie ce poate fi calculată printr-o convoluţie cu masca $\boldsymbol{M}_4$. Modificarea pixelului curent se face doar dacă este îndeplinită o condiţie de prag.

$$\text{IF } [\, \boldsymbol{f}(x,y) - 1/8\sum_{i=1}^{8} v_i(x,y)] > \theta \quad \text{THEN} \quad \boldsymbol{f}(x,y) = 1/8\sum_{i=1}^{8} v_i(x,y) \tag{3}$$

c) Filtre mediane.

O categorie aparte de filtre spaţiale o constituie filtrele mediane la care pixelul central al unei vecinătăţi de forma:

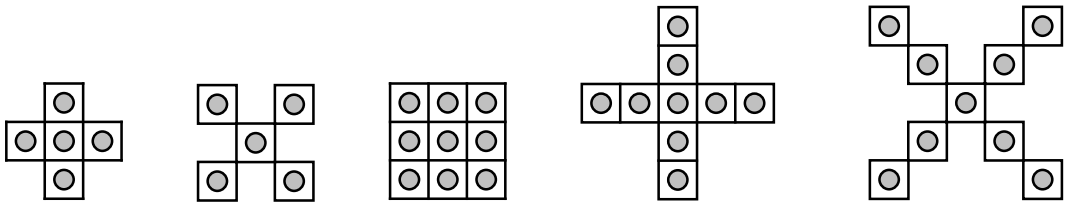
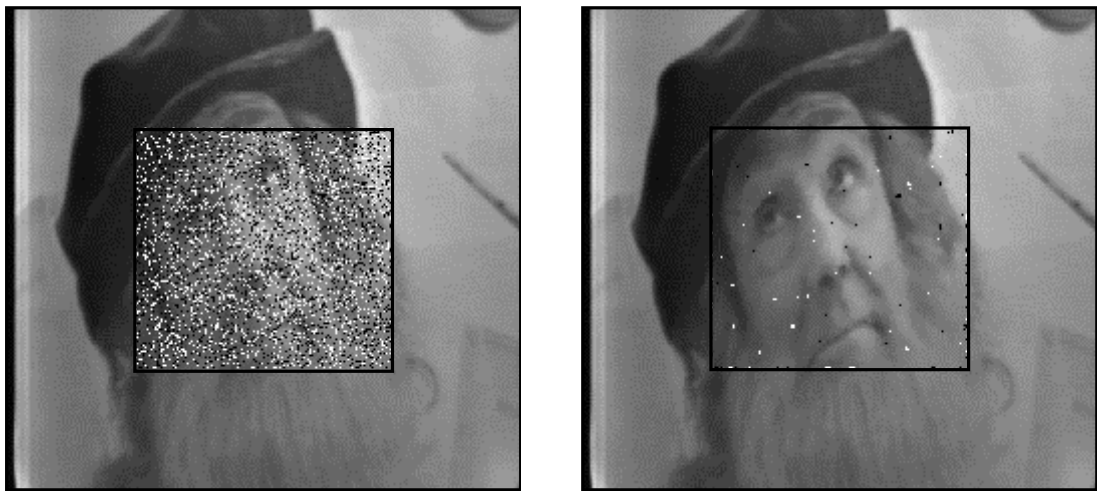


Fig. 1. Măşti pentru filtrarea mediană.

este înlocuit cu medianul mulţimii pixelilor din vecinătate. Efectul aplicării lor este că pixelii cu valoarea mult diferită de a vecinilor dispar.



Exemplu de filtrare mediană.

Mai trebuie de menţionat proprietatea filtrelor mediane de a nu altera semnificativ colţurile şi muchiile obiectelor din imagine. Filtrarea mediană necesită ordonarea pixelilor din vecinătate, ceea ce înseamnă un număr destul de mare de operaţii. În literatură (Pavlidis[79]), (Pratt[84]), (Rosenfeld[95]) sunt citate mai multe tehnici de aproximare a valorii mediane căutate (filtrul pseudomedian, mini-max, maxi-min, etc.), precum şi o serie de tehnici de minimizare a timpului de calcul.

2.1.2. Filtre în domeniul frecvenţă pentru eliminarea zgomotelor sunt mai rar folosite, mai ales datorită complexităţii calculelor. Aplicaţia tipică apare în cazul suprapunerii peste imagine a unor interferenţe periodice cu caracteristici cunoscute. Spectrul unei asemenea imagini conţine maxime semnificative corespunzătoare acestor interferenţe, care pot fi eliminate cu un filtru opreşte-bandă bidimensional, proiectat corespunzător.

2.1.3. Filtrul Wiener generalizat are schema bloc din figura următoare:

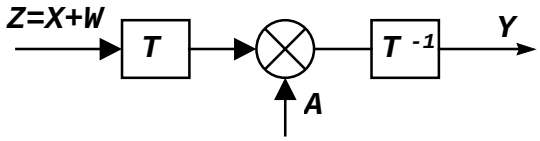


Fig. 2. Filtrul Wiener generalizat.

Unde:

$\boldsymbol{Z} = (z_0, z_1, ..., z_{N-1})$, $\boldsymbol{X} = (x_0, x_1, ..., x_{N-1})$,

$\boldsymbol{W} = (w_0, w_1, ..., w_{N-1})$, $\boldsymbol{Y} = (y_0, y_1, ..., y_{N-1})$

$\boldsymbol{A}$, $\boldsymbol{T}$ - matrici NxN ;

El este un filtru optimal care realizează minimizarea erorii medii pătratice de estimare a semnalului util $\boldsymbol{X}$, peste care se suprapune zgomotul $\boldsymbol{Z}$.

$$\varepsilon = \boldsymbol{E}\{(\boldsymbol{Y} - \boldsymbol{X})^t (\boldsymbol{Y} - \boldsymbol{X})\} \tag{4}$$

În ipotezele:

1) $\boldsymbol{E}\{\boldsymbol{X}\boldsymbol{W}^t\} = \boldsymbol{E}\{\boldsymbol{W}\boldsymbol{X}^t\} = 0$, adică semnalul util şi zgomotul sunt necorelate,

2) $\boldsymbol{T}^{-1} = \boldsymbol{T}^t$, adică transformarea $\boldsymbol{T}$ este unitară,

rezultă:

$$\boldsymbol{A}_0 = \boldsymbol{T}\boldsymbol{K}_{XX}(\boldsymbol{K}_{XX} + \boldsymbol{K}_{WW})^{-1}\boldsymbol{T}^t \tag{5}$$

$$\varepsilon = \boldsymbol{trace}\,[\boldsymbol{K}_{XX} - \boldsymbol{K}_{XX}(\boldsymbol{K}_{XX} + \boldsymbol{K}_{WW})^{-1}\boldsymbol{K}_{WW}] \tag{6}$$

unde

$$\boldsymbol{K}_{XX} = \boldsymbol{E}\{(\boldsymbol{X} - \overline{\boldsymbol{X}})(\boldsymbol{X} - \overline{\boldsymbol{X}})^t\} = \boldsymbol{E}\{\boldsymbol{X}\boldsymbol{X}^t\} \tag{7}$$

şi

$$\boldsymbol{K}_{WW} = \boldsymbol{E}\{(\boldsymbol{W} - \overline{\boldsymbol{W}})(\boldsymbol{W} - \overline{\boldsymbol{W}})^t\} = \boldsymbol{E}\{\boldsymbol{W}\boldsymbol{W}^t\} \tag{8}$$

sunt matricile de autocorelaţie pentru semnalul util şi respectiv zgomot, iar $\overline{\boldsymbol{X}} = \boldsymbol{E}\{\boldsymbol{X}\}$, $\overline{\boldsymbol{W}} = \boldsymbol{E}\{\boldsymbol{W}\}$ sunt mediile lor.

Deci cunoscând caracteristicile statistice ale semnalului util şi respectiv zgomotului se poate construi un filtru optimal, având matricea dată de $\boldsymbol{A}_0$.

Remarcând faptul că mărimea erorii nu depinde de transformarea ortogonală aleasă, putem alege matricea $\boldsymbol{T}$ din condiţia minimizării numărului de calcule.

2.1.4. Metode regionale.

Mai întâi, prin convoluţie cu un set de operatori specifici de extracţie de linii şi muchii, imaginea originală este transformată pentru a obţine o "imagine de control" (Knutsson[61]). Aceasta este o imagine complexă, fiecare punct al ei având o magnitudine $\boldsymbol{B}(x,y)$ şi o direcţie $\boldsymbol{\Theta}(x,y)$. Deci $\boldsymbol{F}(x,y)$ fiind imaginea iniţială, imaginea de control se obţine astfel:

$$\boldsymbol{S}_i(x,y) = \boldsymbol{F}(x,y) * \boldsymbol{E}_i(x,y),\ cu\ i = 1,2,3,4 \tag{9}$$

$$\boldsymbol{C}_i(x,y) = \boldsymbol{F}(x,y) * \boldsymbol{D}_i(x,y), \tag{10}$$

unde "$*$" înseamnă convoluţie, iar $\boldsymbol{E}_i(x,y)$ şi respectiv $\boldsymbol{D}_i(x,y)$ sunt filtre spaţiale pentru detecţia de muchii şi respectiv linii, având direcţia "i".

Amplitudinea imaginii de control în direcţia "i" este dată de:

$$\boldsymbol{B}_i(x,y) = \sqrt{\boldsymbol{S}_i^2(x,y) + \boldsymbol{C}_i^2(x,y)} \Big/ \boldsymbol{V}(x,y),\ i = 1,2,3,4, \tag{11}$$

unde:

$$\boldsymbol{V}(x,y) = \left[\sum_{i=1}^{4} \sqrt{\boldsymbol{S}_i^2(x,y) + \boldsymbol{C}_i^2(x,y)} \right]^{\beta},\ 0 < \beta < 1 \tag{12}$$

Amplitudinea globală este dată de:

$$\boldsymbol{B}(x,y) = \sqrt{[\boldsymbol{B}_1(x,y) - \boldsymbol{B}_3(x,y)]^2 + [\boldsymbol{B}_2(x,y) - \boldsymbol{B}_4(x,y)]^2} \tag{13}$$

iar direcţia în punctul de coordonate (x,y) este $\boldsymbol{\Theta}(x,y)$, care se deduce din

$$\boldsymbol{\sin}[2\boldsymbol{\Theta}(x,y)] = [\boldsymbol{B}_2(x,y) - \boldsymbol{B}_4(x,y)]/\boldsymbol{B}(x,y) \tag{14}$$

$$\boldsymbol{\cos}[2\boldsymbol{\Theta}(x,y)] = [\boldsymbol{B}_1(x,y) - \boldsymbol{B}_3(x,y)]/\boldsymbol{B}(x,y) \tag{15}$$

Filtrele pentru detecţia muchiilor şi a liniilor se construiesc în domeniul frecvenţă astfel:

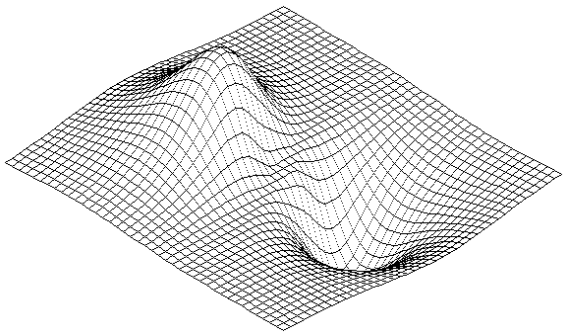
$$\boldsymbol{E}_i(\rho,\theta) = \boldsymbol{e}_i(\rho)\boldsymbol{e}_i(\theta) \tag{16}$$

$$\boldsymbol{D}_i(\rho,\theta) = \boldsymbol{d}_i(\rho)\boldsymbol{d}_i(\theta) \tag{17}$$

$$\boldsymbol{e}_i(\rho) = \boldsymbol{d}_i(\rho) = \boldsymbol{\exp}\left[-\frac{4\boldsymbol{\ln}2}{(\boldsymbol{\ln}\boldsymbol{B})^2} \left(\boldsymbol{\ln}\frac{\rho}{\rho_c}\right)^2 \right] \tag{18}$$

$$\boldsymbol{e}_i(\theta) = \boldsymbol{\cos}^2(\theta - \theta_i)\,\boldsymbol{sign}[\boldsymbol{\cos}(\theta - \theta_i)] \tag{19}$$

$$\boldsymbol{d}_i(\theta) = \boldsymbol{\cos}^2(\theta - \theta_i) \tag{20}$$



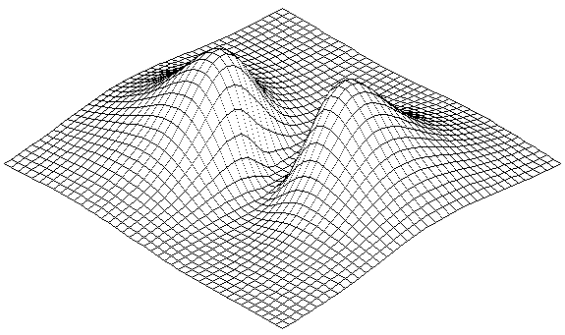


Fig.3.Filtrul pentru detecţia muchiilor $E_i(\rho,\theta)$***, Fig.4.Filtrul pentru detecţia liniilor*** $D_i(\rho,\theta)$**.**

Odată construită imaginea de control (deci estimate direcţia şi mărimea muchiei în fiecare punct), urmează construirea unui filtru anizotropic pentru operaţia efectivă de îmbunătăţire. Acest filtru este suma a două componente: un filtru izotropic trece-jos de netezire şi unul de extragere muchii, orientat în direcţia dată de imaginea de control obţinută din imagine. Proporţia celor două componente este dată de amplitudinea punctuală a imaginii de control.

Filtrul de netezire trebuie să fie izotropic, deci:

$$\boldsymbol{H}(\rho,\theta) \overset{\Delta}{=} \boldsymbol{H}(\rho) = \begin{cases} \boldsymbol{\cos}^2\left(\dfrac{\pi\rho}{1.8}\right), & \text{pentru } \rho < 0.9 \\ 0, & \text{pentru } \rho \geq 0.9 \end{cases} \tag{21}$$

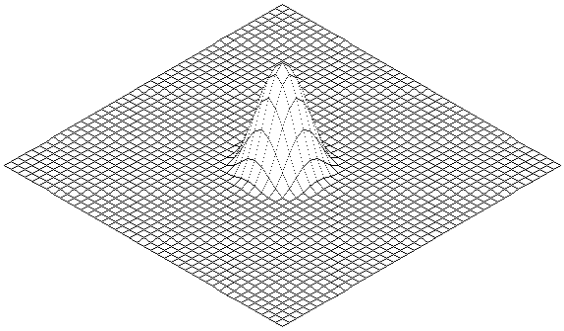


Fig. 5. Filtrul izotropic de netezire $H(\rho,\theta)$**.**

Filtrul de extragere de muchii se construieşte astfel:

$$\boldsymbol{M}(\rho,\theta) = \boldsymbol{m}(\rho)\,\boldsymbol{m}(\theta) \tag{22}$$

unde:

$$\boldsymbol{m}(\rho) = \begin{cases} 1 - \boldsymbol{H}(\rho), & \text{pentru } \rho < 0.9 \\ 1, & \text{pentru } 0.9 \le \rho < \pi - 0.9 \\ \boldsymbol{\cos}^2\left[\dfrac{\pi}{1.8}(\rho - \pi + 0.9)\right], & \text{pentru } \pi - 0.9 \le \rho < \pi \end{cases} \tag{23}$$

$$\boldsymbol{m}(\theta) = \boldsymbol{\cos}^2(\theta) \tag{24}$$

Imaginea îmbunătățită, $\boldsymbol{G}(x,y)$ poate fi exprimată astfel:

$$\boldsymbol{G}(x,y) = \alpha_s\, \boldsymbol{F}(x,y) * \boldsymbol{H}(x,y) + \alpha_e\, \boldsymbol{B}(x,y)[\,\boldsymbol{F}(x,y) * \boldsymbol{M}(x,y,\boldsymbol{\Theta}(x,y))] \tag{25}$$

Este posibilă prelucrarea iterativă a imaginilor zgomotoase, aplicând succesiv formula anterioară.

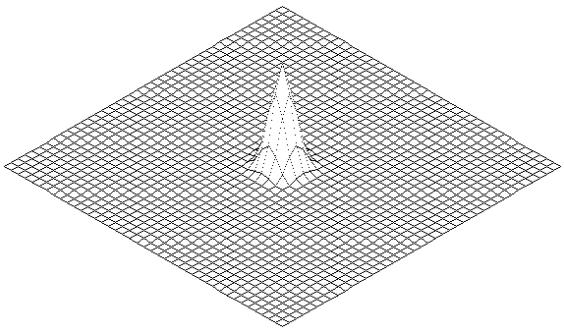


Fig. 6. Filtrul de netezire echivalent după patru iteraţii.

Deşi costisitoare din punctul de vedere al calculelor, această metodă se dovedeşte a fi deosebit de puternică.

2.2. Îmbunătățirea contrastului.

2.2.1. Îmbunătățirea contrastului prin operații asupra histogramei.

Cele mai simple metode de îmbunătățire a contrastului unei imagini au la bază operații asupra histogramei.

Din studiul histogramei unei imagini cu contrast scăzut de poate remarca utilizarea neeficientă a gamei dinamice posibile pentru nivelele de gri. Dacă într-o imagine, cel mai întunecat nivel de gri al unui pixel este K, iar W este nivelul de gri al celui mai luminos pixel, atunci transformarea punctuală liniară numită **scalare**:

$$\tilde{\boldsymbol{f}}(x,y) = \frac{\tilde{W} - \tilde{K}}{W - K}\,[\,\boldsymbol{f}(x,y) - K] \tag{26}$$

va forța noua gamă dinamică a pixelilor din imagine la intervalul $[\tilde{K},\tilde{W}]$. (În cazul general nivelele de gri K şi W se aleg funcție de rezultatul dorit).

Alegând $\tilde{K}=0$ şi $\tilde{W}=L-1$, unde L este numărul maxim posibil de nivele de gri din imagine, se obține o imagine cu contrast îmbunătățit.

Transformarea se poate generaliza prin construirea unei funcții de scalare, care să transforme, după o lege oarecare, luminozitatea pixelilor din imagine.

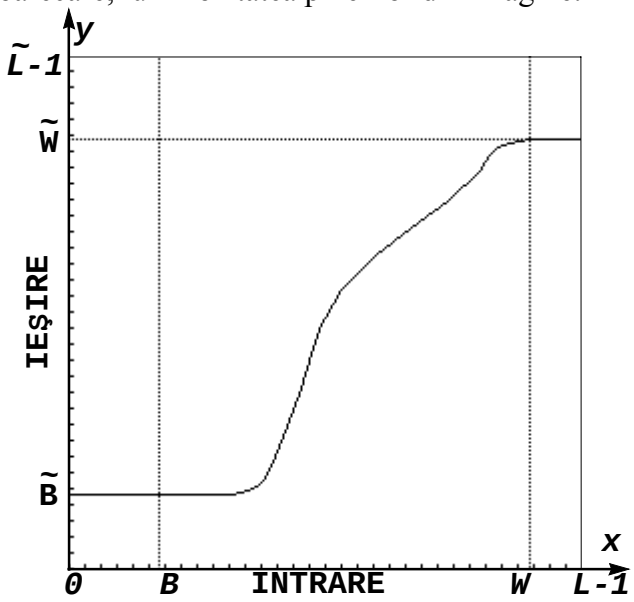


Fig. 7. Exemplu de funcție de scalare.

În literatură sunt citate o multitudine de asemenea funcții de scalare: pătratică, cubică, radical, rădăcina de ordinul trei, liniare pe porțiuni, gaussiană, inversă, logaritmică, exponențială, etc.

Egalizarea de histogramă își propune să furnizeze la ieşire o imagine având o utilizare relativ uniformă a nivelelor de gri, adică o histogramă "aproximativ uniformă":

$$He(z)=\frac{MN}{S}, \tag{27}$$

unde M, N sunt dimensiunile imaginii.

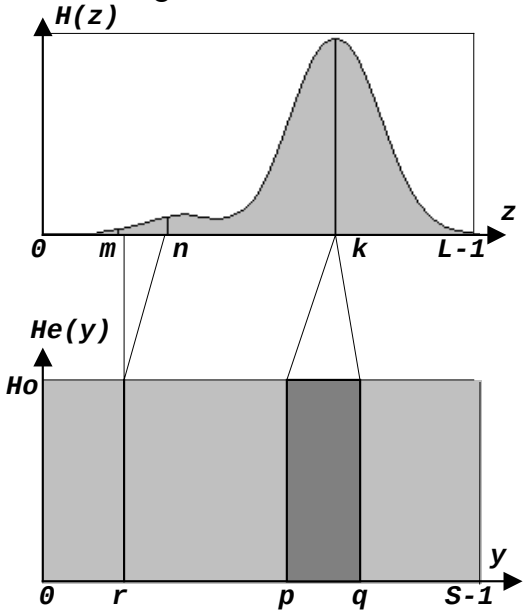


Fig. 8. Principiul egalizării de histogramă.

Se remarcă practic două situații distincte:

1) Nivelele de gri "slab reprezentate" din histograma inițială (există puțini pixeli având aceste nivele de gri), din intervalul *[m,n]* capătă prin egalizare nivelul de gri unic "*r*" în histograma egalizată;

2) Nivelul de gri "*k*", care este "bine reprezentat" în histograma inițială, va fi distribuit, prin egalizare, în intervalul *[p,q]*.

Pentru cazul 2) se poate adopta o strategie din mai multe posibile pentru a realiza distribuirea nivelului de gri "*k*" într-un interval:

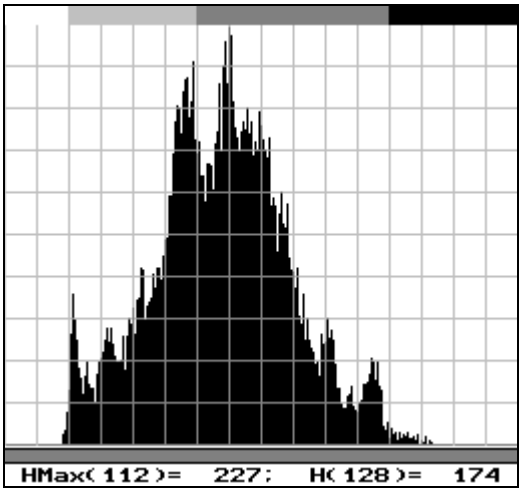
a) Pixelii din imaginea inițială având nivelul de gri "*k*" capătă după egalizare nivelul de gri *(p+q)/2*; histograma nu rezultă perfect plată, dar efortul de calcul este minim;

b) Pixelilor din imaginea inițială li se atribuie un nivel de gri aleator în intervalul *[p,q]*; efortul de calcul rămâne destul de mic, histograma rezultă aproximativ plată, dar apare un zgomot care se suprapune peste imagine;

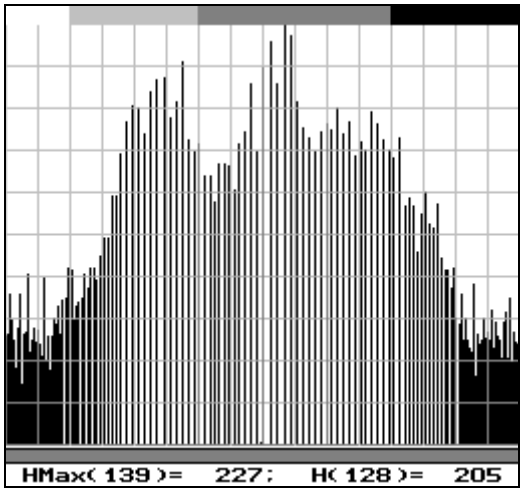
c) Nivelul de gri atribuit pixelilor din imaginea inițială se corelează cu nivelul de gri al vecinilor lui, astfel:

$$\tilde{f}(x,y) = \begin{cases} p, & \text{pentru } f(x,y) < M \\ M, & \text{pentru } p \le M \le q \\ q, & \text{pentru } f(x,y) > M \end{cases} \tag{28}$$

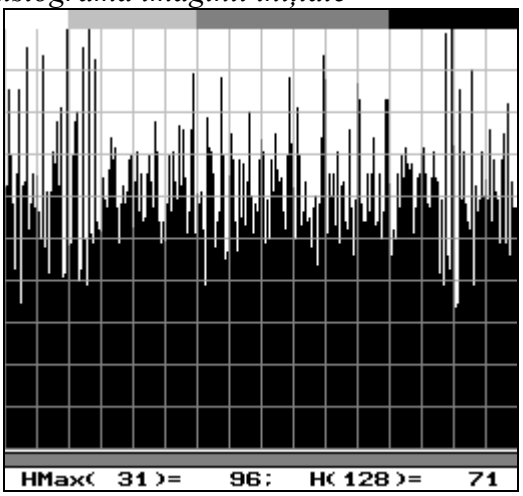
unde:
$$M = \frac{1}{9} \sum_{m=-1}^{1} \sum_{n=-1}^{1} f(x+m, y+n) \tag{29}$$



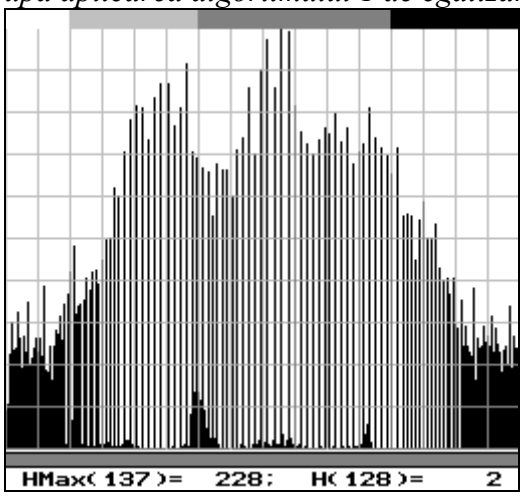
Histograma imaginii inițiale



După aplicarea algoritmului 1 de egalizare



După aplicarea algoritmului 2 de egalizare



După aplicarea algoritmului 3 de egalizare

Imaginea inițială

După aplicarea algoritmului 2 de egalizare

În lucrarea (Pizer[82]) se propune o tehnică adaptivă de egalizare a histogramei, care se bazează pe divizarea imaginii în careuri, pentru fiecare careu calculându-se histograma şi regula de atribuire a noilor nivele de gri. Pentru fiecare pixel se obține regula de atribuire printr-o combinație liniară a nivelelor de gri furnizate de regulile de atribuire corespunzătoare celor patru careuri mai apropiate:

$$M = \frac{A}{Cx}[\frac{B}{Cy}M_{00} + (1-\frac{B}{Cy})M_{10}] + (1-\frac{A}{Cx})[\frac{B}{Cy}M_{01} + (1-\frac{B}{Cy})M_{11}] \qquad (30)$$

unde $M_{00}, M_{10}, M_{01}, M_{11}$ sunt regulile de atribuire date de cele mai apropiate patru careuri, iar restul elementelor rezultă din figura următoare:

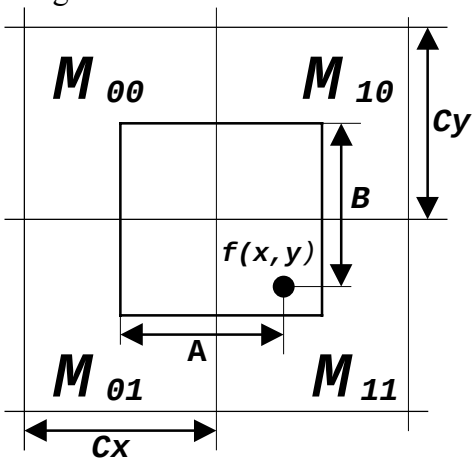


Fig. 9. Principiul egalizării adaptive de histogramă.

2.2.2. Filtrarea homomorfică.

Se acceptă drept model simplificat al imaginii:

$$f(x,y) = i(x,y)\ r(x,y) \qquad (31)$$

unde $i(x,y)$ este componenta numită "iluminare", dată de sursele de lumină aferente imaginii, iar $r(x,y)$ este "reflectanța", dată de proprietățile de reflexie locală ale corpurilor din imagine. Se remarcă proprietățile $i(x,y) > 0$, $0 < r(x,y) < 1$, precum şi faptul că iluminarea $i(x,y)$ variază lent de la o zona la alta a imaginii, pe când reflectanța, variind brusc de la o zona la alta a imaginii, este cea care furnizează frecvențele mari din spectru.

Micşorând ponderea iluminării în raport cu reflectanța se obține o îmbunătățire semnificativă a contrastului, simultan cu eliminarea efectelor iluminării neuniforme a imagini.

Separarea celor două componente se face logaritmând imaginea inițială, urmând ca, după o filtrare clasică, să se execute exponențierea imaginii filtrate. Deci paşii care se urmează sunt (Gonzales[42]):

$$1.\ \log[f(x,y)] = \log[i(x,y)] + \log[r(x,y)]; \tag{32}$$

adică $f'(x,y) = i'(x,y) + r'(x,y)$;

$$2.\ F\{f'(x,y)\} = F\{i'(x,y)\} + F\{r'(x,y)\}; \tag{33}$$

adică $F(u,v) = I(u,v) + R(u,v)$;

$$3.\ F'(u,v) = H(u,v)F(u,v) = H(u,v)I(u,v) + H(u,v)R(u,v); \tag{34}$$

adică $F'(u,v) = I'(u,v) + R'(u,v)$;

$$4.\ F^{-1}\{F'(u,v)\} = F^{-1}\{I'(u,v)\} + F^{-1}\{R'(u,v)\}; \tag{35}$$

adică $f''(x,y) = i''(x,y) + r''(x,y)$;

$$5.\ \exp[f''(x,y)] = \exp[i''(x,y)]\exp[r''(x,y)]; \tag{36}$$

adică $\tilde{f}(x,y) = \tilde{i}(x,y)\tilde{r}(x,y)$

Foarte importantă este alura filtrului bidimensional H, care va atenua frecvențele joase din spectru şi le va amplifica pe cele înalte:

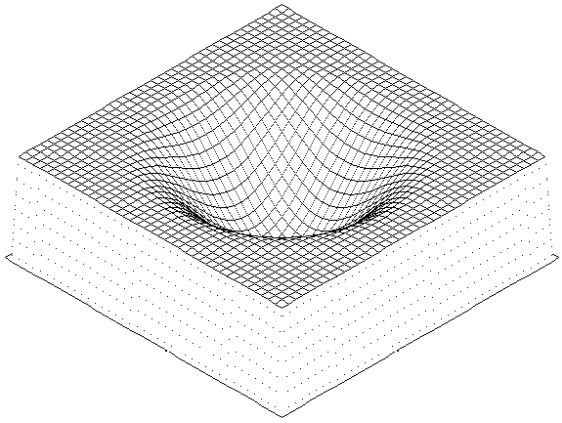


Fig. 10. Filtru trece-sus pentru filtrarea homomorfică.

2.2.3. Diferențiere statistică.

Această metodă (Wallis[110]) îşi propune mărirea contrastului pentru acele zone din imagine caracterizate prin varianță mică. Practic, pentru fiecare pixel al imaginii se calculează media şi respectiv dispersia pixelilor din vecinătatea pătratică (de obicei 9x9) ataşată pixelului curent.

$$m(x,y) = \frac{1}{(2w+1)^2}\sum_{i=-w}^{w}\sum_{j=-w}^{w} f(x+i,y+j); \tag{37}$$

$$\delta^2(x,y) = \frac{1}{(2w+1)^2}\sum_{i=-w}^{w}\sum_{j=-w}^{w}[f(x+i,y+j) - m(x+i,y+j)]^2 \tag{38}$$

cu latura vecinătății $2w+1$;

Asimilând nivelul de gri al pixelului curent cu o variabilă aleatoare, expresia

$$\tilde{f}(x,y) = \frac{f(x,y) - m(x,y)}{\delta(x,y)} \tag{39}$$

descrie o variabilă aleatoare de medie nulă şi dispersie unitară.

Expresia următoare, propusă de Wallis în [110] permite setarea mediei şi a dispersiei la valorile dorite M_d, δ_d, funcție de parametrii "A" şi "r":

$$\tilde{\boldsymbol{f}}(x,y) = \frac{A\delta_d}{A\boldsymbol{\delta}(x,y) + \delta_d}[\,\boldsymbol{f}(x,y) - \boldsymbol{m}(x,y)] + [rM_d + (1-r)\boldsymbol{m}(x,y)] \tag{40}$$

Folosind valori bine alese pentru parametrii formulei precedente, este posibilă evidenţierea unor detalii fine dintr-o imagine cu contrast scăzut.

2.3. Accentuarea muchiilor.

Principiul general al operaţiei de accentuare de muchii se deduce din schema următoare, care prezintă cazul unidimensional. Operatorul Δ este un operator oarecare de diferenţiere iar $\nabla = \Delta^2$.

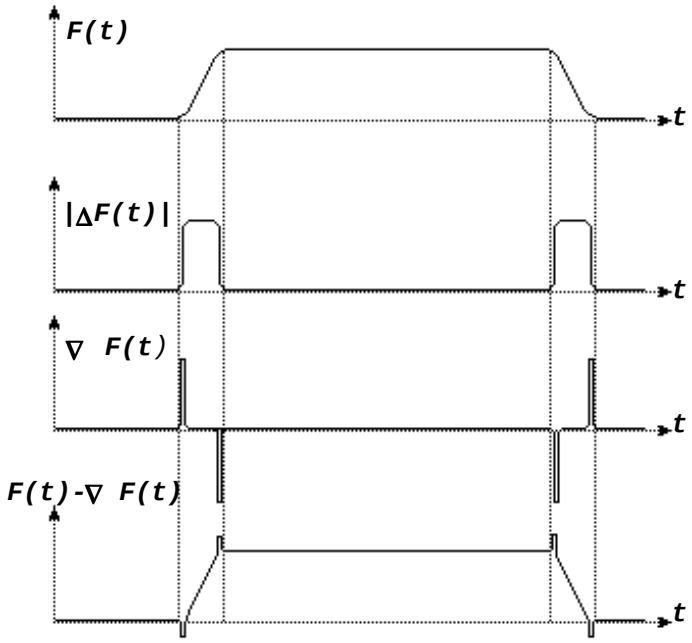


Fig. 11. Principiul accentuării de muchii.

În cazul imaginilor continue, toată informaţia privind variaţiile locale de luminozitate este concentrată în vectorul gradient:

$$\Delta \boldsymbol{f}(x,y) = \frac{\partial \boldsymbol{f}(x,y)}{\partial x^2}\vec{i} + \frac{\partial \boldsymbol{f}(x,y)}{\partial y^2}\vec{j} \tag{41}$$

cu aproximaţiile următoare pentru cazul discret:

$$\Delta_x \boldsymbol{f}(x,y) = \boldsymbol{f}(x+1,y) - \boldsymbol{f}(x,y) \tag{42}$$

$$\Delta_y \boldsymbol{f}(x,y) = \boldsymbol{f}(x,y+1) - \boldsymbol{f}(x,y) \tag{43}$$

Drept operator de diferenţiere de ordinul doi, pentru cazul continuu, se foloseşte Laplacianul:

$$\nabla \boldsymbol{f}(x,y) = \frac{\partial \boldsymbol{f}(x,y)}{\partial x^2} + \frac{\partial \boldsymbol{f}(x,y)}{\partial y^2}, \tag{44}$$

cu aproximările următoare pentru cazul discret:

$$\begin{aligned}
&\nabla_x \boldsymbol{f}(x,y) = \Delta_x \boldsymbol{f}(x,y) - \Delta_x \boldsymbol{f}(x-1,y) = \boldsymbol{f}(x+1,y) - 2\boldsymbol{f}(x,y) + \boldsymbol{f}(x-1,y)\\
&\nabla_y \boldsymbol{f}(x,y) = \Delta_y \boldsymbol{f}(x,y) - \Delta_y \boldsymbol{f}(x-1,y) = \boldsymbol{f}(x,y+1) - 2\boldsymbol{f}(x,y) + \boldsymbol{f}(x,y-1)\\
&\nabla \boldsymbol{f}(x,y) = \nabla_y \boldsymbol{f}(x,y) + \nabla_y \boldsymbol{f}(x,y) = \boldsymbol{f}(x+1,y) + \boldsymbol{f}(x-1,y) + \boldsymbol{f}(x,y+1) + \boldsymbol{f}(x,y-1) - 4\boldsymbol{f}(x,y)
\end{aligned} \tag{45}$$

ceea ce se traduce printr-o convoluţie cu masca $\boldsymbol{L}_0$ (sau $\boldsymbol{L}_1$, sau $\boldsymbol{L}_2$):

$$\boldsymbol{L}_0 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \boldsymbol{L}_1 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \boldsymbol{L}_2 = \begin{bmatrix} -1 & 2 & -1 \\ 2 & -4 & 2 \\ -1 & 2 & -1 \end{bmatrix} \tag{46}$$

Accentuarea muchiilor se face deci cu:

$$\tilde{\boldsymbol{f}}(x,y) = \boldsymbol{f}(x,y) - \nabla \boldsymbol{f}(x,y) = 5\boldsymbol{f}(x,y) - \boldsymbol{f}(x+1,y) - \boldsymbol{f}(x-1,y) - \boldsymbol{f}(x,y+1) - \boldsymbol{f}(x,y-1) \tag{47}$$

ceea ce se traduce printr-o convoluţie cu masca $\boldsymbol{A}_0$ (sau $\boldsymbol{A}_1$, sau $\boldsymbol{A}_2$):

$$\boldsymbol{A}_0 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad \boldsymbol{A}_1 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad \boldsymbol{A}_2 = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{bmatrix} \tag{48}$$

2.4. Îmbunătăţirea imaginilor binare.

2.4.1. Eliminarea zgomotelor.

Primul pas în îmbunătăţirea unei imagini deja binarizate constă în eliminarea zgomotelor. Pentru o imagine deja binarizată, zgomotul tip **"sare şi piper"** este preponderent. Trebuiesc efectuate două tipuri de operaţii de eliminare a pixelilor izolaţi: ştergerea (forţare la alb) pixelilor negri pe fond alb şi ştergerea (setare la negru) pixelilor albi pe fond negru.

2.4.2. Netezirea contururilor.

Operaţia de "netezire a contururilor" obiectelor din imagine în condiţiile păstrării aspectului lor (O'Gorman[76]), are efecte benefice nu numai asupra ratei de recunoaştere, ci şi asupra factorului de compresie posibil, dacă se are în vedere stocarea imaginilor scanate.

Se lucrează pe vecinătăţi pătratice *(kxk)* . Notând:

n = numărul de pixeli negri din vecinătatea pixelului curent;
c = numărul de submulţimi conectate de pixeli din vecinătatea pixelului curent;
r = numărul de pixeli de colţ.

Condiţia de modificare a valorii unui pixel este sintetizată de formula:

$$(c = 1) \text{ and } [(n > 3k\text{-}4) \text{ or } (n = 3k\text{-}4) \text{ and } (r = 2)] = \text{TRUE} \tag{49}$$

Obs. Condiţia $(c = 1)$ asigură păstrarea topologiei imaginii, întocmai ca la algoritmii de subţiere.

Fig. 12. Netezirea contururilor în imaginile binare.

Algoritmul se aplică succesiv pentru pixelii negri, apoi pentru cei albi, până când nu se mai constată modificări pe imagine (sau, pentru a limita durata operării, după un număr predeterminat de paşi).

Alte operaţii care pot fi folosite în îmbunătăţirea imaginilor binare sunt prezentate în capitolul 5, "**Prelucrări morfologice ale imaginilor**".

2.5. Detalii de implementare.

Operaţia de filtrare poate fi implementată în două moduri, funcţie de cerinţele aplicaţiei. Dacă imaginea sursă şi cea destinaţie sunt diferite, atunci implementarea este simplă. Exemplul de mai jos se referă la filtrul cu de masca M_7, descris anterior. Pentru pixelii marginali, pentru care masca corespunzătoare nu mai este inclusă în imaginea sursă, se pot folosi cu succes filtre unidimensionale, a căror mască coincide cu linia / coloana centrală a măştii de referinţă. Imaginile folosite sunt văzute ca matrici de dimensiune *M*N* (**ImgS**, **ImgD**), indicii lor variind în gama *[0,M-1]*, respectiv*[0,N-1]*.

```
procedure Average;
var
   i,j : byte;
begin
   (* prima linie *)
   j:=0;
   for i:=1 to M-2 do
   begin
      ImgD^[j,i]:=(ImgS^[j,i-1]+2*ImgS^[j,i]+ImgS^[j,i+1]) div 4;
   end;
   (* bucla principala *)
   for j:=2 to N-2 do
   begin
      for i:=2 to M-2 do
      begin
         ImgD^[j,i]=(ImgS^[j-1,i-1]+2*ImgS^[j,i-1]+ImgS^[j+1,i-1]+
                     2*ImgS^[j-1,i]+4*ImgS^[j,i]+2*ImgS^[j+1,i]+
                     ImgS^[j-1,i+1]+2*ImgS^[j,i+1]+ImgS^[j+1,i+1]) div 16;
      end;
      (* prima coloana *)
      i:=0;
      ImgD^[j,i]:=(ImgS^[j-1,i]+2*ImgS^[j,i]+ImgS^[j+1,i]) div 4;
      (* ultima coloana *)
      i:=M-1;
      ImgD^[j,i]:=(ImgS^[j-1,i]+2*ImgS^[j,i]+ImgS^[j+1,i]) div 4;
   end;
   (* ultima linie *)
   j:=N-1;
   for i:=1 to M-1 do
   begin
      ImgS^[j,i]:=(ImgS^[j,i-1]+2*ImgS^[j,i]+ImgS^[j,i+1]) div 4;
   end;
end;
```

Dacă operaţia de filtrare trebuie făcută "pe loc", adică sursa coincide cu destinaţia, codul se complică, deoarece este necesară folosirea unor matrici temporare în care să se salveze linia curentă şi cea precedentă din imaginii sursă. Deoarece la calculul liniei noii valori a unui pixel de pe linia *"i"* trebuie folosită informaţia de pe linia precedentă a imaginii originale (deja modificată), informaţia dorită se regăseşte în matricea temporară.

În capitolul 21 este prezentată aplicaţia PROImage, care include, printre altele, şi implementări ale multor filtre descrise în acest capitol. Pentru informaţii suplimentare se poate studia codul sursă din fişierul "**fil.pas**".

Aceeaşi aplicaţie include posibilitatea alterării unei imagini prin adăugarea de zgomot. Această operaţie este utilă în studiul comparativ al comportării diferitelor filtre de netezire în prezenţa zgomotelor. Utilizatorul poate specifica tipul şi amplitudinea zgomotului ce se adaugă imaginii. În aplicaţia PROImage sunt implementate generatoarele de zgomot gaussian, uniform şi aleator. Zgomotul aleator ("speckle noise") afectează brutal pixeli aflaţi în poziţii aleatoare în imagine.

Sunt posibile şi alte optimizări de viteză, inclusiv implementări hardware de timp real, schema bloc a unui asemenea sistem fiind dată în figura următoare:

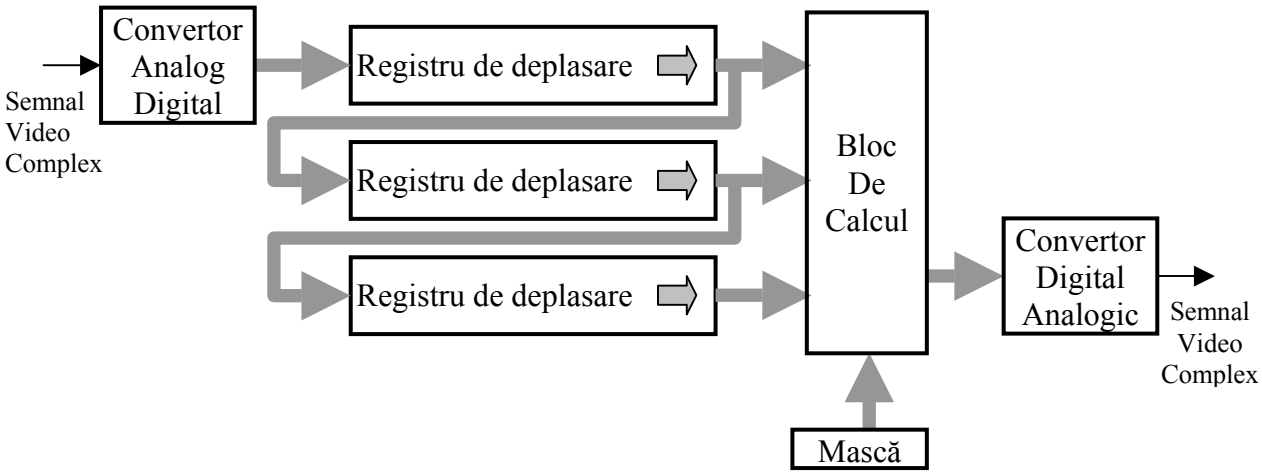


Fig. 13. Implementarea în timp real a filtrelor spațiale.

Registrele de deplasare au lungimea egală cu numărul de pixeli de pe o linie TV. Semnalul de ieşire se obține decalat în timp față de cel de intrare.

O implementare simplificată în limbajul PASCAL a algoritmului de **egalizare de histogramă** este prezentată în continuare.

```
procedure HistEqu(Levels:integer; mode:integer);
var
   i,j,k       : byte;
   HAvg        : longint;
   HInt        : longint;
   Tmp         : integer;
   Left        : array [0..255] of word;
   Right       : array [0..255] of word;
begin
   (*iniţializează vectorii limită*)
   for i:=0 to 255 do
      Left[i]:=0;
   for i:=0 to 255 do
      Right[i]:=0;
   (*calculează aria histogramei*)
   HAvg:=0;
   for i:=0 to 255 do
      HAvg:=HAvg+Hist[i];
   (*bucla de calcul a vectorilor limită*)
   j:=0;
   HInt:=0;
   for i:=0 to 255 do
   begin
      Left[i]:=j;
      HInt:=HInt+Hist[i]*Levels;
      while HInt>=HAvg do
      begin
         HInt:=HInt-HAvg;
         Inc(j);
      end;
      Right[i]:=j;
   end;

   (*prima strategie de egalizare*)
```

```
   if mode=1 then
   begin
      Randomize;
      for j:=0 to N-1 do
         for i:= 0 to M-1 do
            ImD^[j,i]:=(Left[ImS^[j,i]]+Right[ImS^[j,i]]) div 2;
   end;

   (*a doua strategie de egalizare*)
   if mode=2 then
   begin
      Randomize;
      for j:=0 to N-1 do
         for i:= 0 to M-1 do
            ImD^[j,i]:=Left[ImS^[j,i]]
                     +Random(Right[ImS^[j,i]]-Left[ImS^[j,i]]);
   end;

   (*a treia strategie de egalizare*)
   if mode=2 then
   begin
      for j:=1 to N-2 do
         for i:= 1 to M-2 do
         begin
            // calculează valoarea medie în vecinătate
            Tmp:=0;
            for n:=-1 to 1 do
               for m:= -1 to 1 do
                  Tmp:=Tmp+Ims^[j+n,i+m];
            // calculează valoarea pixelului la ieşire
            if Tmp < Left[ImS^[j,i]]*9 then
               Tmp:=Left[ImS^[j,i]]*9;
            if Tmp > Right[ImS^[j,i]]*9 then
               Tmp:=Right[ImS^[j,i]]*9;
            ImD^[j,i]:=Tmp;
         end;
   end;
end;
```

Parametrul **"Levels"** este numărul de nivele de gri al imaginii de ieşire, iar **"Mode"** specifică strategia de egalizare dorită. Imaginile de intrare/ieşire, **"ImS"** şi **"ImD"** sunt matrici *MxN* având un octet/pixel.

Diferenţierea statistică realizează "controlului automat al luminozităţii", pe baza caracteristicilor statistice ale imaginii, la nivel local, pe vecinătăţi de dimensiuni reduse. Implementarea prezentată în continuare (limbajul PASCAL) foloseşte vecinătăţi *9x9*.

```
procedure Enhn();
var
   i,j   : shortint;
   x,y,z : byte;
   Med   : real;        (*media vecinătăţii*)
   Sum   : real;        (*dispersia vecinătăţii*)
   Dif   : real;        (*valoarea calculată a ieşirii*)
begin
   (*bucla principala*)
   for y:= 4 to N-5 do
      for x:= 4 to M-5 do
      begin
         (*calcul medie aritmetică în vecinătate*)
```

```
         Med:=0.0;
         for i:=-4 to 4 do
            for j:=-4 to 4 do
               Med:=Med+ImS^[y+j,x+i];
         Med:=Med/81.0;
         (* calcul dispersie în vecinătate*)
         Sum:=0;
         for i:=-4 to 4 do
            for j:=-4 to 4 do
               Sum:=Sum+Sqr(ImS^[y+j,x+i]-Med);
         Sum:=Sum/81.0;
         (*calcul valoare pixel la ieşire*)
         (*Md este media dorită a fiecărei vecinătăţi din imagine*)
         (*Sd este dispersia dorită a fiecărei vecinătăţi din imagine*)
         (*A este factorul de amplificare*)
         (*r este factorul de ponderare, cu valori în gama [0,1] *)
         Dif:=(ImS^[y,x]-Med)*(Sd/(Sum+Sd/A))+r*Md+(1-r)*Med;
         (*corecţii*)
         if Dif>255.0 then
            Dif:=255.0;
         if Dif<0.0 then
            Dif:=0.0;
         ImD^:=Dif;
      end;
end;
```

3. Restaurarea imaginilor.

Restaurarea imaginilor are drept scop repararea imaginilor afectate de distorsiuni ale căror caracteristici se cunosc sau se deduc.

Pentru modelarea proceselor care duc la degradarea imaginii se utilizează următoarea schemă bloc generală (Jain[56]):

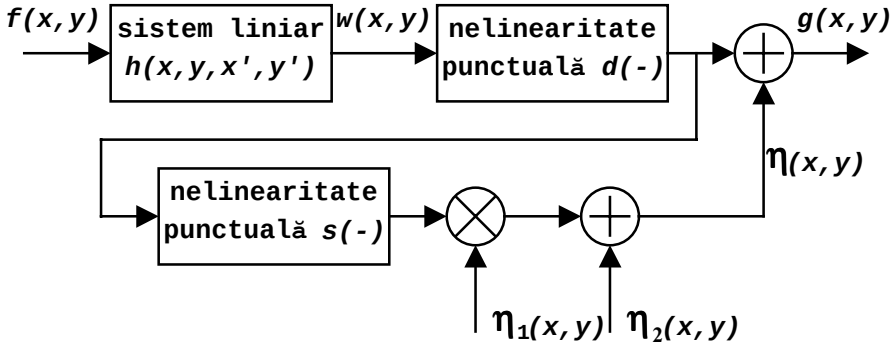


Fig. 1. Schema bloc de modelare a procesului de degradare a imaginilor.

Semnalul inițial $f(x,y)$ este mai întâi trecut printr-un filtru având răspunsul la impuls $h(x,y;x',y')$, obținându-se semnalul $w(x,y)$ (convoluție în domeniul spațial):

$$w(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h((x,y;x',y')\, f(x',y')dx'xy' \tag{1}$$

Funcția neliniară $d(-)$ modelează neliniaritățile introduse de blocul de detecție a imaginii, iar peste rezultat se suprapune un zgomot $\eta(x,y)$:

$$g(x,y) = d[w(x,y)] + \eta(x,y) \tag{2}$$

Acest zgomot conține două componente: una dependentă de semnal prin intermediul neliniarității $s(-)$ care modelează blocul de înregistrare a imaginii $\eta_1(x,y)$, şi alta independentă de semnal (aditivă), notată $\eta_2(x,y)$. Deci:

$$\eta(x,y) = s[\, g(\, w(x,y))]\, \eta_1(x,y) + \eta_2(x,y) \tag{3}$$

În concluzie, un model general al procesului de degradare a imaginii este:

$$g(x,y) = d\left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(x,y;x',y')\, f(x',y')dx'xy'\right] + s[\, g(\, w(x,y))]\, \eta_1(x,y) + \eta_2(x,y) \tag{4}$$

Modelul descris suferă adesea numeroase simplificări, care facilitează analiza unor distorsiuni tipice care apar în practică. Cel mai adesea distorsiunile sunt invariante la translații (shift invariant), deci se foloseşte

$$h(x,y;x',y') = h(x-x',y-y';0,0) \overset{\Delta}{=} h(x-x',y-y') \tag{5}$$

De asemenea, adeseori se presupune $r(-) = 0$.

Trecând în domeniul frecvență, unde analiza este mai facilă, relația anterioară se scrie:

$$G(\omega_x,\omega_y) = H(\omega_x,\omega_y)\, F(\omega_x,\omega_y) + N(\omega_x,\omega_y) \tag{6}$$

3.1. Modele matematice ale imaginilor de restaurat.

În literatură sunt descrise câteva modele matematice care aproximează răspunsurile la impuls ale câtorva sisteme reale care afectează calitatea imaginii.

Cele mai des întâlnite distorsiuni sunt cele datorate fenomenului de difracție în partea optică a sistemului de captură a imaginii. În cazul difracției pentru lumină coerentă, răspunsurile la impuls în domeniul spațial şi respectiv frecvență ale modelului sunt:

$$h(x,y) = \frac{\sin(ax)\sin(by)}{xy} \qquad H(\omega_x,\omega_y) = rect\left[\frac{\omega_x}{a},\frac{\omega_y}{b}\right] \tag{7}$$

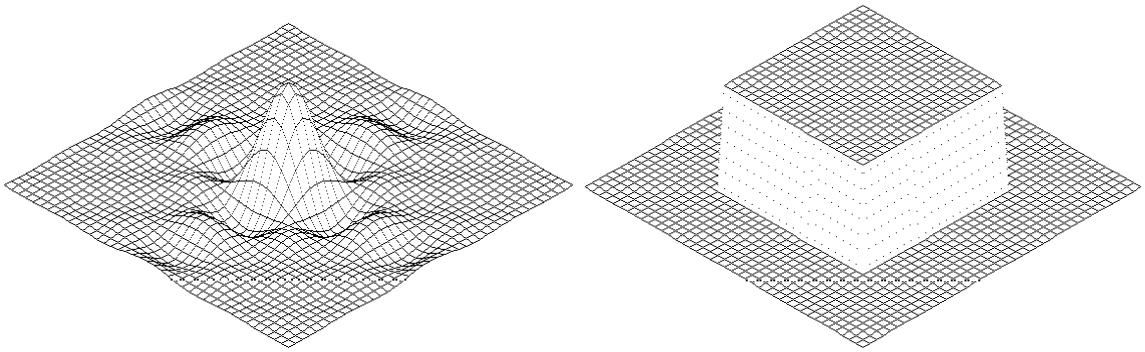


Fig. 2. Modelarea distorsiunilor datorate difracției optice (lumină coerentă).

Varianta pentru lumină incoerentă este descrisă de:

$$\boldsymbol{h}(x,y) = \frac{\boldsymbol{\sin}^2(ax)}{ax}\frac{\boldsymbol{\sin}^2(by)}{by} \qquad \boldsymbol{H}(\omega_x,\omega_y) = \boldsymbol{tri}\left[\frac{\omega_x}{a},\frac{\omega_y}{b}\right] \tag{8}$$

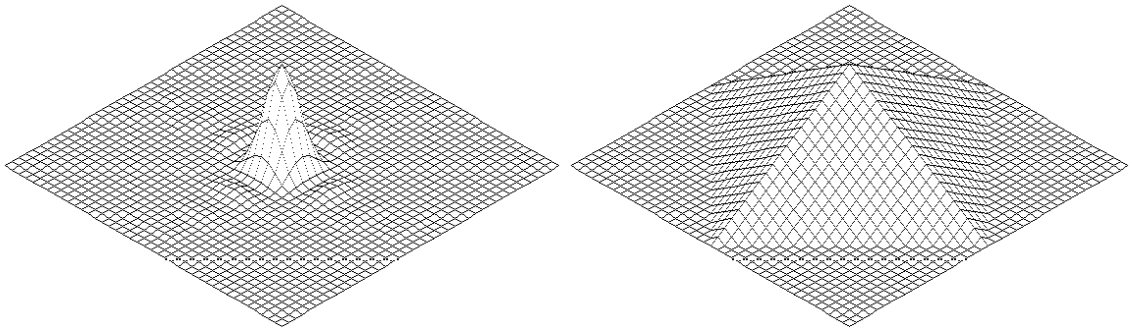


Fig. 3. Modelarea distorsiunilor datorate difracţiei optice (lumină incoerentă).

Răspunsul în frecvenţă pentru modelarea imaginilor obţinute printr-un sistem optic puternic defocalizat are aproximativ următoarea formă:

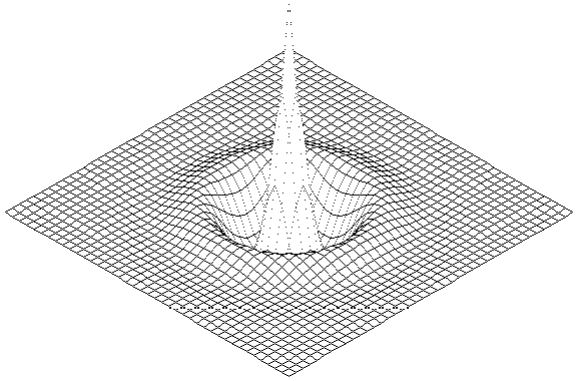


Fig. 4. Modelarea distorsiunilor datorate defocalizării puternice.

Un alt tip de distorsiune extrem de des întâlnit în practică este cazul imaginilor "mişcate", adică, pe durata achiziţiei imaginii are loc o deplasare relativă a detectorului optic faţă de scenă. Răspunsul la impuls în domeniul timp, respectiv frecvenţă sunt:

$$\boldsymbol{h}(x,y) = \frac{1}{\alpha_0}\boldsymbol{rect}\left(\frac{x}{\alpha_0}-\frac{1}{2}\right)\boldsymbol{\delta}(y) \qquad \boldsymbol{H}(\omega_x,\omega_y) = \boldsymbol{exp}\left(-j\pi\pi_x\alpha_0\right)\frac{\boldsymbol{\sin}(\alpha_0\omega_x)}{\alpha_0\omega_x} \tag{9}$$

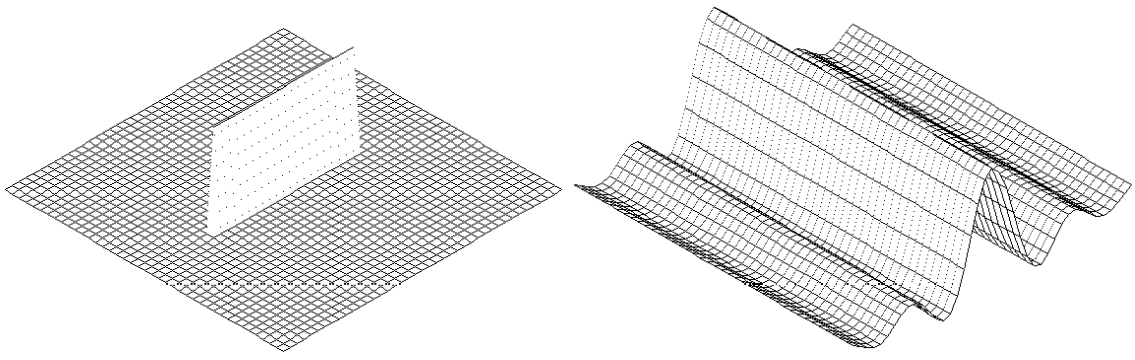


Fig. 5. Modelarea distorsiunilor de tip "imagine mişcată".

Pentru imaginile satelitare (sau luate de la mare altitudine) intervine un alt gen de distorsiune, datorat turbulenţei atmosferice, descris de:

$$\boldsymbol{h}(x,y) = \boldsymbol{exp}\left[-\pi\alpha^2(x^2+y^2)\right] \qquad \boldsymbol{H}(\omega_x,\omega_y) = \frac{1}{\alpha^2}\boldsymbol{exp}\left[\frac{-\pi(\omega_x^2+\omega_y^2)}{\alpha^2}\right] \tag{10}$$

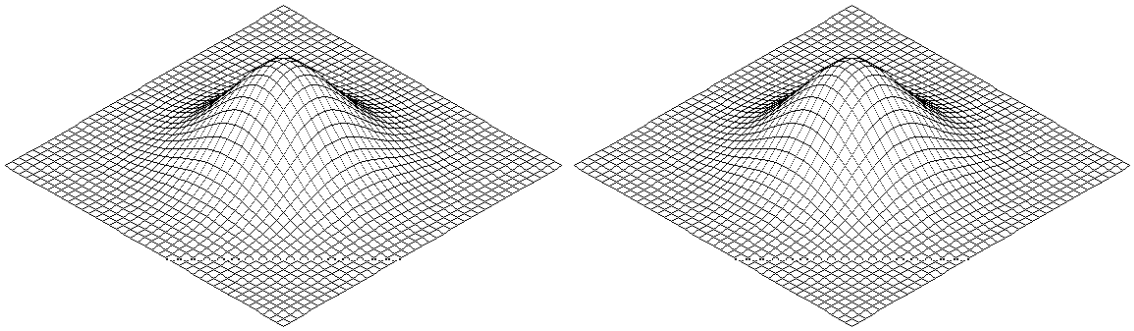


Fig. 6. Modelarea distorsiunilor datorate turbulențelor atmosferice.

La operația de scanare apare fenomenul de apertură descris de:

$$\boldsymbol{h}(x,y) = \boldsymbol{rect}\left(\frac{x}{\alpha},\frac{y}{\beta}\right) \qquad \boldsymbol{H}(\omega_x,\omega_y) = \alpha\beta\,\frac{\boldsymbol{sin}(\alpha\alpha_x)}{\alpha\omega_x}\,\frac{\boldsymbol{sin}(\beta\beta_y)}{\beta\omega_y} \tag{11}$$

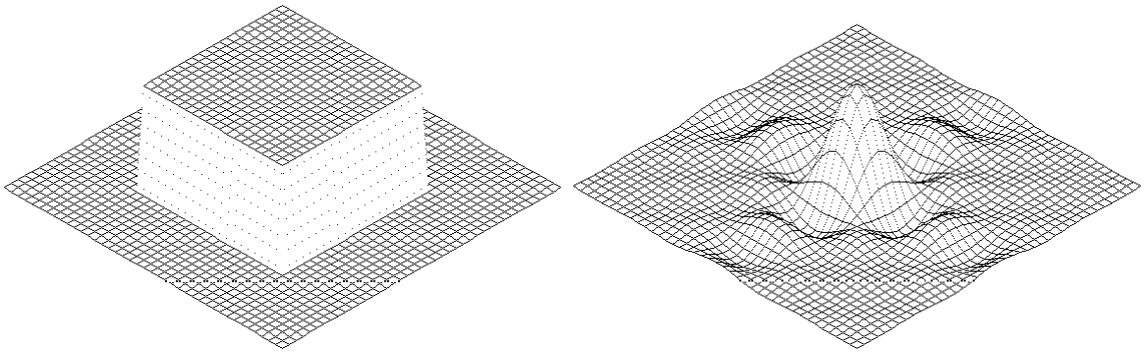


Fig. 7. Modelarea distorsiunilor fenomenului de apertură (scanare).

În cazul matricilor CCD (Charge Coupled Devices) apare un alt gen de distorsiune, datorat interacțiunii între celulele vecine ale ariei de captură. Ea este descrisă de:

$$\boldsymbol{h}(x,y) = \sum_{m=-1}^{1}\sum_{n=-1}^{1}\alpha_{m,n}\,\boldsymbol{\delta}(x-m\Delta,y-n\Delta), \quad \boldsymbol{H}(\omega_x,\omega_y) = \sum_{m=-1}^{1}\sum_{n=-1}^{1}\alpha_{m,n}\,\boldsymbol{exp}\,[-2\pi j\Delta(m\omega_x+n\omega_y)] \tag{12}$$

adică valoarea furnizată de fiecare celulă este o combinație liniară a valorilor ideale corespunzătoare unei vecinătăți 3x3 din matricea de captură a CCD.

Pentru **caracterizarea zgomotului**, cel mai adesea se utilizează modelul gaussian de medie nulă, descris deci de densitatea de repartiție:

$$\boldsymbol{p}(z) = \frac{1}{\sigma\sqrt{2\pi}}\,\boldsymbol{exp}\left(\frac{-z^2}{2\sigma^2}\right) \tag{13}$$

O primă excepție citată în literatură este cazul unor tipuri de fotodetectoare lucrând la iluminări extrem de scăzute, pentru care este mai potrivită distribuția Poisson (legea evenimentelor rare):

$$p_k = \frac{\lambda^k}{k!}\,\boldsymbol{exp}(-\lambda) \tag{14}$$

Jain descrie în [56] şi un alt tip de zgomot ("speckle noise") care apare în cazul imaginilor microscopice cu detalii având dimensiuni de acelaşi ordin de mărime cu lungimea de undă a radiației incidente. Este un zgomot multiplicativ, descris de o distribuție exponențială:

$$\boldsymbol{p}(z) = \begin{cases} \dfrac{1}{\sigma^2}\,\boldsymbol{exp}\left(\dfrac{-z}{\sigma^2}\right), & \text{pentru } z \geq 0 \\ 0, & \text{pentru } z < 0 \end{cases} \tag{15}$$

Variantele **discrete** ale ecuațiilor care descriu modelul de degradare a unei imagini sunt următoarele, foarte asemănătoare cu cazul continuu:

$$\begin{aligned} &\boldsymbol{g}(m,n) = \boldsymbol{d}[w(m,n)] + \boldsymbol{\eta}(m,n) \\ &\boldsymbol{w}(m,n) = \sum_{m'=-\infty}^{+\infty} \sum_{n'=-\infty}^{+\infty} \boldsymbol{h}(m,n;m',n')\, \boldsymbol{f}(m',n') \\ &\boldsymbol{\eta}(m,n) = \boldsymbol{s}[\, \boldsymbol{g}(\, \boldsymbol{w}(m,n))]\, \boldsymbol{\eta}_1(m,n) + \boldsymbol{\eta}_2(m,n) \end{aligned} \tag{16}$$

3.2. Tehnici convolutive de restaurare a imaginilor.

Tehnicile de restaurare a imaginilor urmăresc construcția unui filtru care, plecând de la imaginea distorsionată, să furnizeze la ieşire o imagine cât mai apropiată de cea ideală (Pratt[84]).

$$\widetilde{\boldsymbol{f}}(x,y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \boldsymbol{r}(x,y;x',y')\, \boldsymbol{g}(x',y')dx'dy' \tag{17}$$

sau în domeniul frecvență:

$$\widetilde{\boldsymbol{F}}(\omega_x,\omega_y) = \boldsymbol{R}(\omega_x,\omega_y)\boldsymbol{G}(\omega_x,\omega_y) \tag{18}$$

Ținând cont de relația de definiție a lui $\boldsymbol{G}$ rezultă:

$$\widetilde{\boldsymbol{F}}(\omega_x,\omega_y) = \boldsymbol{R}(\omega_x,\omega_y)[\, \boldsymbol{H}(\omega_x,\omega_y)\boldsymbol{F}(\omega_x,\omega_y) + \boldsymbol{N}(\omega_x,\omega_y)] \tag{19}$$

3.2.1. Metoda filtrului invers.

În absența zgomotului relația anterioară devine:

$$\widetilde{\boldsymbol{F}}(\omega_x,\omega_y) = \boldsymbol{R}(\omega_x,\omega_y)\boldsymbol{H}(\omega_x,\omega_y)\boldsymbol{F}(\omega_x,\omega_y) \tag{20}$$

Condiția ca $\widetilde{\boldsymbol{F}}(\omega_x,\omega_y)$ să fie cât mai apropiat de $\boldsymbol{F}(\omega_x,\omega_y)$ se traduce prin:

$$\boldsymbol{R}(\omega_x,\omega_y) = \frac{1}{\boldsymbol{H}(\omega_x,\omega_y)} = \frac{\boldsymbol{H}^*(\omega_x,\omega_y)}{\left|\boldsymbol{H}(\omega_x,\omega_y)\right|^2} \tag{21}$$

Filtrul invers furnizează cea mai simplă soluție de restaurare a imaginilor afectate de distorsiuni, în schimb el prezintă o serie de dezavantaje.

Realizarea practică a filtrului invers este dificilă deoarece el este adesea instabil, şi anume în apropierea zerourilor lui $\boldsymbol{H}(\omega_x,\omega_y)$, pentru care $\boldsymbol{R}(\omega_x,\omega_y)$ tinde către infinit. În plus, dacă zgomotul este prezent, estimarea obținută va fi slabă.

$$\widetilde{\boldsymbol{F}}(\omega_x,\omega_y) = \boldsymbol{F}(\omega_x,\omega_y) + \frac{\boldsymbol{H}^*(\omega_x,\omega_y)}{\left|\boldsymbol{H}(\omega_x,\omega_y)\right|^2}\boldsymbol{N}(\omega_x,\omega_y) \tag{22}$$

Filtrul **pseudoinvers** încearcă să evite instabilitatea prezentă la filtrul invers.

$$\boldsymbol{R}(\omega_x,\omega_y) = \begin{cases} \dfrac{1}{\boldsymbol{H}(\omega_x,\omega_y)}, & \text{pentru } \boldsymbol{H}(\omega_x,\omega_y) \neq 0 \\ 0, & \text{pentru } \boldsymbol{H}(\omega_x,\omega_y) = 0 \end{cases} \tag{23}$$

În practică condiția $\boldsymbol{H}(\omega_x,\omega_y) = 0$ se înlocuieşte cel mai adesea cu o condiție de tipul $\left|\boldsymbol{H}(\omega_x,\omega_y)\right| < \varepsilon$. Efectul zgomotelor asupra calității imaginii restaurate se reduce la acest tip de filtru față de precedentul.

3.2.2. Filtrul Wiener.

Acest tip de filtru, larg utilizat în prelucrarea de semnale, foloseşte drept criteriu de construcţie condiţia de minimizare a erorii medii pătratice între semnalul de intrare şi cel de ieşire (restaurat), adică:

$$\varepsilon^2 = \boldsymbol{E}\{[\, \boldsymbol{f}(x,y) - \tilde{\boldsymbol{f}}(x,y)]^2\} = \boldsymbol{min} \tag{24}$$

Filtrul Wiener încorporează cunoştinţe statistice apriorice despre semnalul util şi zgomot. Pentru $f(x,y)$, $g(x,y)$ se face ipoteza că ele sunt procese aleatoare de medie nulă.

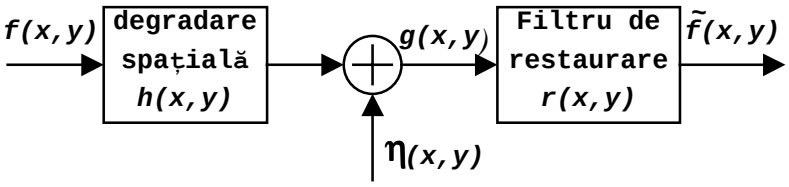


Fig. 8. Filtrul Wiener.

În ipoteza, nerestrictivă pentru început, a invarianţei la deplasări spaţiale, avem de minimizat:

$$\varepsilon^2 = \boldsymbol{E}\{[\, \boldsymbol{f}(x,y) - \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \boldsymbol{r}(x-x',y-y')\,\boldsymbol{g}(x',y')dx'dy']^2\} = \boldsymbol{min} \tag{25}$$

Această condiţie este îndeplinită când:

$$\boldsymbol{E}\left\{\left[\, \boldsymbol{f}(x,y) - \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \boldsymbol{r}(x-x',y-y')\,\boldsymbol{g}(x',y')dx'dy'\right]\boldsymbol{g}(\hat{x},\hat{y})\right\} = 0 \tag{26}$$

valabilă pentru orice pereche de puncte (x,y) şi $(\hat{x},\hat{y})$ din planul imaginii (condiţia de ortogonalitate).

Rezultă condiţia (egalitatea):

$$\boldsymbol{E}\{\, \boldsymbol{f}(x,y)\,\boldsymbol{g}(\hat{x},\hat{y})\} = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \boldsymbol{r}(x-x',y-y')\,\boldsymbol{E}\{\, \boldsymbol{g}(x',y')\,\boldsymbol{g}(\hat{x},\hat{y})\}dx'dy' \tag{27}$$

valabilă în aceleaşi condiţii. Dar deoarece:

$$\boldsymbol{E}\{\, \boldsymbol{f}(x,y)\,\boldsymbol{g}(\hat{x},\hat{y})\} = \boldsymbol{K}_{fg}(x-\hat{x},y-\hat{y}) \tag{28}$$

$$\boldsymbol{E}\{\, \boldsymbol{g}(x',y')\,\boldsymbol{g}(\hat{x},\hat{y})\} = \boldsymbol{K}_{gg}(x'-\hat{x},y'-\hat{y}) \tag{29}$$

putem exprima relaţia anterioară prin intermediul funcţiilor de corelaţie:

$$\boldsymbol{K}_{fg}(x-\hat{x},y-\hat{y}) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \boldsymbol{r}(x-x',y-y')\,\boldsymbol{K}_{gg}(x'-\hat{x},y'-\hat{y})dx'dy' \tag{30}$$

Trecând în domeniul transformatei Fourier, relaţia anterioară devine:

$$\boldsymbol{K}_{fg}(\omega_x,\omega_y) = \boldsymbol{R}(\omega_x,\omega_y)\,\boldsymbol{K}_{gg}(\omega_x,\omega_y) \tag{31}$$

cu $\boldsymbol{K}_{gg}(\omega_x,\omega_y)$ spectrul de putere al funcţiei "g", de unde:

$$\boldsymbol{R}(\omega_x,\omega_y) = \boldsymbol{K}_{fg}(\omega_x,\omega_y)\,\boldsymbol{K}_{gg}^{-1}(\omega_x,\omega_y) \tag{32}$$

Dar deoarece:

$$\boldsymbol{G}(\omega_x,\omega_y) = \boldsymbol{H}(\omega_x,\omega_y)\,\boldsymbol{F}(\omega_x,\omega_y) + \boldsymbol{N}(\omega_x,\omega_y) \tag{33}$$

Rezultă:

$$\boldsymbol{K}_{fg}(\omega_x,\omega_y) = \boldsymbol{H}^*(\omega_x,\omega_y)\,\boldsymbol{K}_{ff}(\omega_x,\omega_y) \tag{34}$$

$$\boldsymbol{K}_{gg}(\omega_x,\omega_y) = |\boldsymbol{H}(\omega_x,\omega_y)|^2\,\boldsymbol{K}_{ff}(\omega_x,\omega_y) + \boldsymbol{K}_{\eta\eta}(\omega_x,\omega_y) \tag{35}$$

Ca urmare filtrul Wiener, în cazul prezenței zgomotului aditiv, poate fi exprimat funcție de răspunsul la impuls în domeniul frecvență şi spectrele de putere ale imaginii inițiale şi, respectiv, zgomotului:

$$\boldsymbol{R}(\omega_x,\omega_y)=\frac{\boldsymbol{H}^*(\omega_x,\omega_y)\,\boldsymbol{K}_{ff}(\omega_x,\omega_y)}{|\boldsymbol{H}(\omega_x,\omega_y)|^2\,\boldsymbol{K}_{ff}(\omega_x,\omega_y)+\boldsymbol{K}_{\eta\eta}(\omega_x,\omega_y)} \tag{36}$$

Pentru cazul discret al filtrului Wiener vezi Capitolul 2, paragraful 2.1.3.

3.2.3. Metoda egalizării spectrului de putere.

Spectrul de putere al imaginii degradate este:

$\boldsymbol{K}_{\tilde{f}\tilde{f}}(\omega_x,\omega_y)=|\boldsymbol{R}(\omega_x,\omega_y)|^2\,\boldsymbol{K}_{gg}(\omega_x,\omega_y)$ sau dezvoltând $\boldsymbol{K}_{gg}(\omega_x,\omega_y)$ avem:

$$\boldsymbol{K}_{\tilde{f}\tilde{f}}(\omega_x,\omega_y)=|\boldsymbol{R}(\omega_x,\omega_y)|^2\left[|\boldsymbol{H}(\omega_x,\omega_y)|^2\,\boldsymbol{K}_{ff}(\omega_x,\omega_y)+\boldsymbol{K}_{\eta\eta}(\omega_x,\omega_y)\right] \tag{37}$$

Forțând ca $\boldsymbol{K}_{\tilde{f}\tilde{f}}(\omega_x,\omega_y)=\boldsymbol{K}_{ff}(\omega_x,\omega_y)$ rezultă expresia filtrului de restaurare:

$$\boldsymbol{R}(\omega_x,\omega_y)=\left[\frac{\boldsymbol{K}_{ff}(\omega_x,\omega_y)}{|\boldsymbol{H}(\omega_x,\omega_y)|^2\,\boldsymbol{K}_{ff}(\omega_x,\omega_y)+\boldsymbol{K}_{\eta\eta}(\omega_x,\omega_y)}\right]^{\frac{1}{2}} \tag{38}$$

Acest tip de filtru are proprietăți interesante pentru cazul când nu sunt disponibile informații apriorice privind caracteristicile statistice ale semnalului de intrare şi zgomotului. Are o comportare intermediară între filtrul invers şi filtrul Wiener.

3.2.4. Filtrul - medie geometrică.

Acest tip de filtru realizează un compromis între câştigul în rezoluție şi senzitivitatea la zgomot. Filtrul invers are o rezoluție foarte bună, dar sensibilitate mare la zgomote, pe când filtrul Wiener are imunitate ridicată la zgomot, dar introduce o pierdere a rezoluției (fiind, în esență, un filtru trece-jos). Stockham în [105] propune utilizarea următorului filtru:

$$\boldsymbol{R}(\omega_x,\omega_y)=\left[\frac{\boldsymbol{H}^*(\omega_x,\omega_y)}{|\boldsymbol{H}(\omega_x,\omega_y)|^2}\right]^{\alpha}\left[\frac{\boldsymbol{K}_{ff}(\omega_x,\omega_y)}{|\boldsymbol{H}(\omega_x,\omega_y)|^2\,\boldsymbol{K}_{ff}(\omega_x,\omega_y)+\gamma\,\boldsymbol{K}_{\eta\eta}(\omega_x,\omega_y)}\right]^{1-\alpha} \tag{39}$$

unde parametrii α şi γ se aleg funcție de problema abordată. Pentru $\alpha=0.5$ şi $\gamma=1$ se obține filtrul anterior.

3.2.5. Estimarea parametrică a filtrului de restaurare.

Fie filtrul obținut prin egalizarea spectrului de putere (Pratt[86]), care poate fi exprimat prin:

$$\boldsymbol{R}(\omega_x,\omega_y)=\left[\frac{\boldsymbol{K}_{ff}(\omega_x,\omega_y)}{\boldsymbol{K}_{gg}(\omega_x,\omega_y)}\right]^{\frac{1}{2}} \tag{40}$$

Dacă nu se cunosc aprioric nici un fel de informații privind caracteristicile statistice ale semnalului de restaurat şi zgomotului, ele se pot **estima** din analiza imaginii observate.

Cannon în [28] propune divizarea imaginii inițiale în blocuri pătratice cu dimensiunea suficient de mare în comparație cu dimensiunea spațială a răspunsului la impuls $\boldsymbol{h}(x,y)$. Pentru fiecare bloc "*j*" poate fi calculat spectrul de putere $\boldsymbol{K}_{gg}^{(j)}(\omega_x,\omega_y)$. Spectrul de putere corespunzător întregii imagini se estimează prin medierea spectrelor de putere ale tuturor blocurilor.

Pentru estimarea lui s-ar putea folosi, dacă este disponibilă, o imagine nedegradată, folosind o procedură identică cu cea descrisă anterior.

O altă posibilitate ar fi construcţia unei imagini de test care să permită determinarea lui $\boldsymbol{H}(\omega_x,\omega_y)$ prin analiza semnăturii lui. Estimând printr-o tehnică convenabilă spectrul de putere al zgomotului, $\boldsymbol{K}_{\eta\eta}(\omega_x,\omega_y)$, se poate deduce valoarea estimată a spectrului de putere al imaginii iniţiale prin:

$$\boldsymbol{K}_{ff}(\omega_x,\omega_y)=\frac{\boldsymbol{K}_{gg}(\omega_x,\omega_y)-\boldsymbol{K}_{\eta\eta}(\omega_x,\omega_y)}{\left|\boldsymbol{H}(\omega_x,\omega_y)\right|^2} \tag{41}$$

3.2.6. Concluzii.

Toate metodele prezentate până acum se referă la restaurarea imaginilor degradate cu funcţii distorsiune invariante la translaţie, prin metode convolutive. Deşi foarte utile într-o multitudine de aplicaţii, ele eşuează în unele situaţii.

Mai puţin filtrul invers, toate celelalte sunt în esenţă filtre trece-jos, atenuând frecvenţele înalte, ceea ce duce la o pierdere a rezoluţiei, care se manifestă vizual prin atenuarea muchiilor din imagine. De aici ideea de a adapta filtrul de restaurare la caracteristicile locale ale imaginii (muchie, regiune uniformă).

3.3. Tehnici adaptive de restaurare a imaginilor.

3.3.1. Filtrul Anderson-Netravali.

Cei doi autori studiază în [04] cazul unei imagini discrete degradate doar prin prezenţa unui zgomot aditiv şi propun construirea unei "funcţii de mascare":

$$\boldsymbol{M}(m,n)=\sum_{m'}\sum_{n'}c^{-\sqrt{(m-m')^2+(n-n')^2}}\left[\left|\boldsymbol{d}^{(H)}(m',n')\right|+\left|\boldsymbol{d}^{(V)}(m',n')\right|\right] \tag{42}$$

unde $\boldsymbol{d}^{(H)}(m,n)$, $\boldsymbol{d}^{(V)}(m,n)$ sunt gradienţii imaginii în punctul de coordonate (m,n), în direcţie orizontală şi respectiv verticală. În continuare este folosită o "funcţie de vizibilitate" $\varphi(M)$, construită pe baza unor măsurători psiho-fiziologice, astfel încât:

$\varphi(0)=1$, adică vizibilitate totală în absenţa muchiilor, şi

$\varphi(+\infty)=0$, adică prezenţa muchiei "ascunde" zgomotul.

Experimentele au arătat o relativă independenţă a rezultatelor de funcţia de vizibilitate, deşi au fost construite experimente pshiho-fizice precise pentru măsurarea funcţiei de vizibilitate pentru diferite categorii de imagini.

3.3.2. Filtrul Abramatic-Silverman.

Abramatic şi Silverman în [02] au extins studiul filtrului precedent, făcând practic legătura cu tehnicile convolutive descrise anterior. Situaţia studiată de ei pentru început se referă tot la cazul discret al unei imagini distorsionate doar de zgomot:

$$\boldsymbol{g}(m,n)=\boldsymbol{v}(m,n)+\boldsymbol{\eta}(m,n) \tag{43}$$

unde v(x,y) este o imagine cu zgomot, dar care, *pentru un observator uman*, arată identic cu cea ideală, iar $\boldsymbol{\eta}'(x,y)$ este zgomotul rezidual ce trebuie eliminat.

Zgomotul rezidual, evident, depinde de caracteristicile locale ale imaginii, adică:

$$\boldsymbol{\eta}'(m,n)=\gamma_{m,n}[\,\boldsymbol{f}_v(m,n),\eta m,m)] \tag{44}$$

unde indicele "v" semnifică o vecinătate a funcţiei $''\boldsymbol{f}''$ în jurul punctului (x,y), deci zgomotul rezidual depinde de valorile lui $''\boldsymbol{f}''$ din această vecinătate.

Imaginea restaurată se obţine prin:

$$\tilde{\boldsymbol{f}}(m,n)=\sum_i\sum_j\boldsymbol{r}_{m,n}(i,j)\,\boldsymbol{g}(m-i,n-j) \tag{45}$$

unde filtrul de reconstrucție $r_{m,n}(i,j)$ se construieşte din condiția minimizării erorii:

$$\varepsilon' = S_{m,n}(r_{m,n}, K_{ff}) + N_{m,n}(r_{m,n}, \sigma_\eta^2) \quad (46)$$

Această eroare conține deci două componente corespunzătoare transformării Wiener discrete clasice, una "de rezoluție":

$$S(r,K_{ff}) = \sum_m \sum_n \sum_{m'} \sum_{n'} \{[\delta(m,n) - h(m,n)]^t K_{ff}(m-m',n-n')[\delta(m',n') - h(m',n')]\} \quad (47)$$

şi una datorată zgomotului:

$$N(r,\sigma_n^2) = \sigma_n^2 \sum_x \sum_y r^2(x,y) \quad (48)$$

unde $\delta(m,n)$ este funcția delta-Kronecker bidimensională.

Autorii propun introducerea proprietății de adaptare prin intermediul lui *N*, adică:

$$N_{m,n} = \varphi[M(m,n)]\sigma_\eta^2 \sum_i \sum_j r_{m,n}(i,j) \quad (49)$$

unde φ este funcția de vizibilitate definită anterior. Rezultă filtrul de restaurare în domeniul frecvență:

$$R_{m,n}(\omega_x,\omega_y) = \frac{K_{ff}(\omega_x,\omega_y)}{K_{ff}(\omega_x,\omega_y) + \varphi[M(m,n)]\sigma_\eta^2} \quad (50)$$

A doua soluție propusă este adaptarea prin intermediul lui λ:

$$\lambda_{m,n} = \varphi[M(m,n)]\,\eta(m,n) \quad (51)$$

caz în care se obține:

$$R_{m,n}(\omega_x,\omega_y) = (1-\varphi[M(m,n)]) + \varphi[M(m,n)]\frac{K_{ff}(\omega_x,\omega_y)}{K_{ff}(\omega_x,\omega_y) + \sigma_\eta^2} \quad (52)$$

Abramatic şi Silverman extind apoi studiul lor la cazul imaginilor alterate nu numai prin zgomot, ci şi printr-o convoluție cu funcția de degradare $h(m,n)$. Filtrele corespunzătoare obținute în acest caz sunt:

$$R_{m,n}(\omega_x,\omega_y) = \frac{H^*(\omega_x,\omega_y)K_{ff}(\omega_x,\omega_y)}{|H(\omega_x,\omega_y)|^2 K_{ff}(\omega_x,\omega_y) + \varphi[M(m,n)]\sigma_n^2} \quad (53)$$

şi respectiv:

$$R_{kl}(\omega_x,\omega_y) = \varphi[M(m,n)]\frac{H^*(\omega_x,\omega_y)K_{ff}(\omega_x,\omega_y)}{|H(\omega_x,\omega_y)|^2 K_{ff}(\omega_x,\omega_y) + \sigma_n^2} + (1-\varphi[M(m,n)])\frac{H^*(\omega_x,\omega_y)}{|H(\omega_x,\omega_y)|^2} \quad (54)$$

3.4. Restaurarea unor imagini degradate cu variantă la translație.

O primă clasă de alterări ale imaginii (Wallis[110]) la care funcția punctuală de degradare este variantă la translație poate fi descompusă în trei operații distincte:

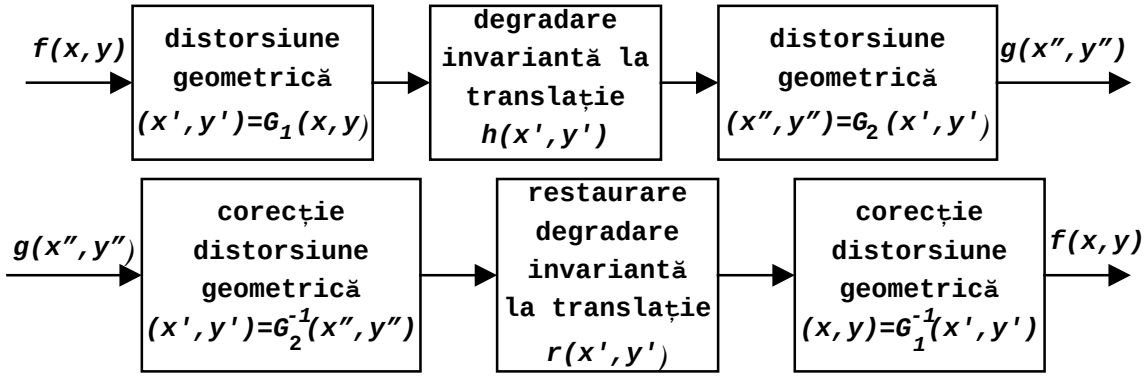


Fig. 9. Modelul degradărilor cu variantă la translație.

Primul şi ultimul pas constau în distorsiuni geometrice, în timp ce în pasul al doilea imaginea suferă o degradare invariantă la translaţie.

Ecuaţia care descrie obţinerea imaginii discrete degradate:

$$\boldsymbol{g}(m,n) = \boldsymbol{d}\left[\sum_{m'=0}^{M}\sum_{n'=0}^{N}\boldsymbol{h}(m,n;m',n')\,\boldsymbol{f}(m',n')\right] + \boldsymbol{\eta}(m,n) \tag{55}$$

poate fi scrisă sub formă matriceală dacă imaginea iniţială, cea degradată şi cea corespunzătoare zgomotului se exprimă prin vectori de lungime MxN :

$$[\boldsymbol{g}] = \boldsymbol{d}\{[\boldsymbol{H}][\boldsymbol{f}]\} + [\boldsymbol{n}] \tag{56}$$

Pentru cazul în care nelinearitarea "$\boldsymbol{d}$" se poate neglija, rezultă un sistem liniar de ecuaţii având MxN necunoscute, de obicei mult prea mare pentru a fi rezolvat prin metodele clasice:

$$[\boldsymbol{g}] = [\boldsymbol{H}][\boldsymbol{f}] + [\boldsymbol{n}] \Leftrightarrow [\boldsymbol{f}] = [\boldsymbol{H}]^{-1}([\boldsymbol{g}] - [\boldsymbol{n}]) \tag{57}$$

De aceea, de obicei, soluţia acestei probleme este una aproximativă, obţinută prin metode similare cu cazul invarianţei la translaţie.

Corespunzător **filtrului invers**, se utilizează criteriul:

$$\boldsymbol{E}\{\tilde{\boldsymbol{f}}\} = \left\|[\boldsymbol{g}] - [\boldsymbol{H}][\boldsymbol{f}]\right\|^2 = \boldsymbol{min} \tag{58}$$

ceea ce duce la inversa generalizată:

$$[\tilde{\boldsymbol{f}}] = ([\boldsymbol{H}]^{*t}[\boldsymbol{H}])^{-1}[\boldsymbol{H}]^{*t}[\boldsymbol{g}] \tag{59}$$

Dacă, corespunzător **filtrului Wiener** funcţia de minimizat se alege:

$$\boldsymbol{E}\{[\tilde{\boldsymbol{f}}]\} = \left\|\boldsymbol{K}_{ff}^{-1/2}\boldsymbol{K}_{\eta\eta}^{1/2}[\boldsymbol{f}]\right\|^2 + \left\|[\boldsymbol{g}] - [\boldsymbol{H}][\boldsymbol{f}]\right\|^2 \tag{60}$$

unde $\boldsymbol{K}_{ff}$ şi $\boldsymbol{K}_{\eta\eta}$ sunt matricile de covarianţă ale imaginii ideale şi zgomotului, se obţine drept ecuaţie de restaurare:

$$[\tilde{\boldsymbol{f}}] = \left([\boldsymbol{H}]^{*t}[\boldsymbol{H}] + \boldsymbol{K}_{ff}^{-1}\boldsymbol{K}_{\eta\eta}\right)^{-1}[\boldsymbol{H}]^{*t}[\boldsymbol{g}] \tag{61}$$

3.5. Discuţii asupra modelului discret al restaurării.

Pentru studiul cazului discret, pentru simplitate, se consideră (Gonzales[42]) mai întâi funcţia $\boldsymbol{f}(x,y)$ unidimensională, eşantionată uniform, astfel încât să se obţină A eşantioane. Modelul matematic al degradării ei presupune o convoluţie cu $\boldsymbol{h}(x)$, de asemenea o funcţie discretă, dar având B valori. Evitarea depăşirii acestor domenii prin operaţia de convoluţie se face definind extensiile lor:

$$\boldsymbol{f}_e(x) = \begin{cases} \boldsymbol{f}(x), \text{ pentru } 0 \le x \le A-1 \\ 0, \quad \text{pentru } A \le x \le M-1 \end{cases} \tag{62}$$

$$\boldsymbol{h}_e(x) = \begin{cases} \boldsymbol{h}(x), \text{ pentru } 0 \le x \le C-1 \\ 0, \quad \text{pentru } C \le x \le M-1 \end{cases} \tag{63}$$

periodice de perioadă M. Atunci operaţia de convoluţie pentru cazul discret unidimensional se scrie:

$$\boldsymbol{g}_e(x) = \sum_{n=0}^{M-1}\boldsymbol{f}_e(m)\,\boldsymbol{h}_e(x-m) \tag{64}$$

Sau matriceal:

$$\boldsymbol{g} = \boldsymbol{H}\boldsymbol{f} \tag{65}$$

adică:

$$\begin{bmatrix} g_e(0) \\ g_e(1) \\ g_e(2) \\ \dots \\ g_e(M-1) \end{bmatrix} = \begin{bmatrix} h_e(0) & h_e(M-1) & h_e(M-2) & \dots & h_e(1) \\ h_e(1) & h_e(0) & h_e(M-1) & \dots & h_e(2) \\ h_e(2) & h_e(1) & h_e(0) & \dots & h_e(3) \\ \dots & \dots & \dots & \dots & \dots \\ h_e(M-1) & h_e(M-2) & h_e(M-3) & \dots & h_e(0) \end{bmatrix} \begin{bmatrix} f_e(0) \\ f_e(1) \\ f_e(2) \\ \dots \\ f_e(M-1) \end{bmatrix} \quad (66)$$

Matricea $\boldsymbol{H}$ având forma de mai sus se numește *matrice circulantă*. Notând:

$$\lambda(k) \equiv h_e(0) + h_e(M-1)\,\boldsymbol{exp}\left[\frac{2\pi j}{M}k\right] + h_e(M-2)\,\boldsymbol{exp}\left[\frac{2\pi j}{M}2k\right] + \dots + h_e(1)\,\boldsymbol{exp}\left[\frac{2\pi j}{M}(M-1)k\right] \text{ şi}$$

$$\boldsymbol{w}(k) = \left[1 \quad \boldsymbol{exp}\left(\frac{2\pi j}{M}k\right) \quad \boldsymbol{exp}\left(\frac{2\pi j}{M}2k\right) \quad \dots \quad \boldsymbol{exp}\left(\frac{2\pi j}{M}(M-1)k\right)\right]^T \quad (67)$$

se poate demonstra relația:

$$\boldsymbol{H}\boldsymbol{w}(k) = \lambda(k)\boldsymbol{w}(k) \quad (68)$$

care exprimă faptul că $\boldsymbol{w}(k)$ sunt vectorii proprii ai matricii $\boldsymbol{H}$, iar $\lambda(k)$ sunt valorile ei proprii. Construind matricea:

$$\boldsymbol{W} = \left[\boldsymbol{w}(0) \quad \boldsymbol{w}(1) \quad \boldsymbol{w}(2) \quad \dots \quad \boldsymbol{w}(M-1)\right] \quad (69)$$

având elementele:

$$W(k,i) = \boldsymbol{exp}\left(\frac{2\pi j}{M}ki\right) \quad (70)$$

se poate arăta că inversa ei are elementele date de:

$$W^{-1}(k,i) = \frac{1}{M}\boldsymbol{exp}\left(-\frac{2\pi j}{M}ki\right) \quad (71)$$

Atunci matricea degradării punctuale se poate exprima prin:

$$\boldsymbol{H} = \boldsymbol{W}\boldsymbol{D}\boldsymbol{W}^{-1} \quad (72)$$

cu $\boldsymbol{D}$ matrice diagonală, având:

$$D(k,k) = \lambda(k) \quad (73)$$

şi care se poate calcula prin:

$$\boldsymbol{D} = \boldsymbol{W}^{-1}\boldsymbol{H}\boldsymbol{W} \quad (74)$$

Pentru *cazul bidimensional* extensiile funcției discrete $\boldsymbol{f}(x,y)$ care descrie imaginea şi respectiv $\boldsymbol{h}(x,y)$ care descrie degradarea punctuală sunt:

$$\boldsymbol{f}_e(x,y) = \begin{cases} \boldsymbol{f}(x,y), \text{ pentru } 0 \le x \le A-1 \text{ şi } 0 \le y \le B-1 \\ 0, \quad \text{pentru } A \le x \le M-1 \text{ şi } B \le y \le N-1 \end{cases} \quad (75)$$

$$\boldsymbol{h}_e(x,y) = \begin{cases} \boldsymbol{h}(x,y), \text{ pentru } 0 \le x \le C-1 \text{ şi } 0 \le y \le D-1 \\ 0, \quad \text{pentru } C \le x \le M-1 \text{ şi } D \le y \le N-1 \end{cases} \quad (76)$$

Operația de degradare a imaginii este modelată atunci de:

$$\boldsymbol{g}_e(x,y) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \boldsymbol{f}_e(m,n)\,\boldsymbol{h}_e(x-m,y-n) + \boldsymbol{n}_e(x,y) \quad (77)$$

Relația anterioară poate fi scrisă matriceal:

$$\boldsymbol{g} = \boldsymbol{H}\boldsymbol{f} + \boldsymbol{n} \quad (78)$$

unde:

$$\boldsymbol{H}=\begin{bmatrix} \boldsymbol{H_e(0)} & \boldsymbol{H_e(M-1)} & \boldsymbol{H_e(M-2)} & \boldsymbol{...} & \boldsymbol{H_e(1)} \\ \boldsymbol{H_e(1)} & \boldsymbol{H_e(0)} & \boldsymbol{H_e(M-1)} & \boldsymbol{...} & \boldsymbol{H_e(2)} \\ \boldsymbol{H_e(2)} & \boldsymbol{H_e(1)} & \boldsymbol{H_e(0)} & \boldsymbol{...} & \boldsymbol{H_e(3)} \\ \boldsymbol{...} & \boldsymbol{...} & \boldsymbol{...} & \boldsymbol{...} & \boldsymbol{...} \\ \boldsymbol{H_e(M-1)} & \boldsymbol{H_e(M-2)} & \boldsymbol{H_e(M-3)} & \boldsymbol{...} & \boldsymbol{H_e(0)} \end{bmatrix} \tag{79}$$

fiecare element al ei fiind o matrice circulantă de forma:

$$\boldsymbol{H_k}=\begin{bmatrix} h_e(j,0) & h_e(j,N-1) & h_e(j,N-2) & ... & h_e(j,1) \\ h_e(j,1) & h_e(j,0) & h_e(j,N-1) & ... & h_e(j,2) \\ h_e(j,2) & h_e(j,1) & h_e(j,0) & ... & h_e(j,3) \\ ... & ... & ... & ... & ... \\ h_e(j,N-1) & h_e(j,N-2) & h_e(j,N-3) & ... & h_e(j,0) \end{bmatrix} \tag{80}$$

Matricea $\boldsymbol{H}$ este o *matrice bloc-circulantă*. Notând:

$$w_M(i,m)=\boldsymbol{exp}\left(\frac{2\pi j}{M}im\right) \qquad \text{şi} \qquad w_N(k,n)=\boldsymbol{exp}\left(\frac{2\pi j}{N}kn\right) \tag{81}$$

se poate defini *matricea de matrici*:

$$\boldsymbol{W}=\begin{bmatrix} \boldsymbol{W(0,0)} & \boldsymbol{W(0,1)} & \boldsymbol{W(0,2)} & \boldsymbol{...} & \boldsymbol{W(0,M-1)} \\ \boldsymbol{W(1,0)} & \boldsymbol{W(1,1)} & \boldsymbol{W(1,2)} & \boldsymbol{...} & \boldsymbol{W(1,M-1)} \\ \boldsymbol{W(2,0)} & \boldsymbol{W(2,1)} & \boldsymbol{W(2,2)} & \boldsymbol{...} & \boldsymbol{W(2,M-1)} \\ \boldsymbol{...} & \boldsymbol{...} & \boldsymbol{...} & \boldsymbol{...} & \boldsymbol{...} \\ \boldsymbol{W(M-1,0)} & \boldsymbol{W(M-1,1)} & \boldsymbol{W(M-1,2)} & \boldsymbol{...} & \boldsymbol{W(M-1,M-1)} \end{bmatrix} \tag{82}$$

unde:

$$\boldsymbol{W}(i,m)=w_M(i,m)\boldsymbol{W}_N, \qquad \text{cu} \qquad \boldsymbol{W}_N(k,n)=w_N(k,n) \tag{83}$$

În continuare se poate demonstra că inversa matricii $\boldsymbol{W}$ se scrie sub forma:

$$\boldsymbol{W^{-1}}=\begin{bmatrix} \boldsymbol{W^{-1}(0,0)} & \boldsymbol{W^{-1}(0,1)} & \boldsymbol{W^{-1}(0,2)} & \boldsymbol{...} & \boldsymbol{W^{-1}(0,M-1)} \\ \boldsymbol{W^{-1}(1,0)} & \boldsymbol{W^{-1}(1,1)} & \boldsymbol{W^{-1}(1,2)} & \boldsymbol{...} & \boldsymbol{W^{-1}(1,M-1)} \\ \boldsymbol{W^{-1}(2,0)} & \boldsymbol{W^{-1}(2,1)} & \boldsymbol{W^{-1}(2,2)} & \boldsymbol{...} & \boldsymbol{W^{-1}(2,M-1)} \\ \boldsymbol{...} & \boldsymbol{...} & \boldsymbol{...} & \boldsymbol{...} & \boldsymbol{...} \\ \boldsymbol{W^{-1}(M-1,0)} & \boldsymbol{W^{-1}(M-1,1)} & \boldsymbol{W^{-1}(M-1,2)} & \boldsymbol{...} & \boldsymbol{W^{-1}(M-1,M-1)} \end{bmatrix} \tag{84}$$

unde:

$$\boldsymbol{W^{-1}}(i,m)=\frac{1}{M}w_M^{-1}(i,m)\boldsymbol{W}_N^{-1} \tag{85}$$

cu:

$$w_M^{-1}(i,m)=\boldsymbol{exp}\left(-\frac{2\pi j}{M}im\right) \tag{86}$$

$$\boldsymbol{W}_N^{-1}(k,n)=\frac{1}{N}w_N^{-1}(k,n)=\frac{1}{N}\boldsymbol{exp}\left(-\frac{2\pi j}{N}kn\right) \tag{87}$$

Evident avem:

$$\boldsymbol{W}\boldsymbol{W^{-1}}=\boldsymbol{W^{-1}}\boldsymbol{W}=\boldsymbol{I}_{MN} \tag{88}$$

Ca urmare, se poate arăta că matricea degradării care afectează imaginea digitală este:

$$\boldsymbol{H}=\boldsymbol{W}\boldsymbol{D}\boldsymbol{W^{-1}} \tag{89}$$

cu $\boldsymbol{D}$ matrice diagonală, ale cărei elemente sunt valorile proprii ale matricii $\boldsymbol{H}$. Rezultă:

$$\boldsymbol{D} = \boldsymbol{W}^{-1}\boldsymbol{H}\boldsymbol{W} \tag{90}$$

În ipoteza absenței zgomotului, degradarea care afectează imaginea este descrisă de:

$$\boldsymbol{g} = \boldsymbol{W}\boldsymbol{D}\boldsymbol{W}^{-1}\boldsymbol{f} \tag{91}$$

de unde:

$$\boldsymbol{W}^{-1}\boldsymbol{g} = \boldsymbol{D}\boldsymbol{W}^{-1}\boldsymbol{f} \tag{92}$$

Se remarcă faptul că $\boldsymbol{W}^{-1}\boldsymbol{f}$ este tocmai transformata Fourier discretă a intrării, iar $\boldsymbol{W}^{-1}\boldsymbol{g}$ este transformata Fourier a semnalului imagine degradat, adică:

$$\boldsymbol{F}(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\boldsymbol{f}_e(x,y)\,\boldsymbol{exp}\left[-2\pi j\left(\frac{ux}{M}+\frac{vy}{N}\right)\right] \tag{93}$$

$$\boldsymbol{G}(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\boldsymbol{g}_e(x,y)\,\boldsymbol{exp}\left[-2\pi j\left(\frac{ux}{M}+\frac{vy}{N}\right)\right] \tag{94}$$

Notând transformata Fourier discretă bidimensională a zgomotului cu:

$$\boldsymbol{N}(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\boldsymbol{n}_e(x,y)\,\boldsymbol{exp}\left[-2\pi j\left(\frac{ux}{M}+\frac{vy}{N}\right)\right] \tag{95}$$

iar

$$\boldsymbol{F} = \boldsymbol{W}^{-1}\boldsymbol{f} \quad \text{şi} \quad \boldsymbol{G} = \boldsymbol{W}^{-1}\boldsymbol{g} \tag{96}$$

rezultă:

$$\boldsymbol{G} = \boldsymbol{D}\boldsymbol{F} + \boldsymbol{N} \tag{97}$$

Importanța expresiei de mai sus rezidă în faptul că studiul degradărilor de "întindere" mare (descrise deci de sisteme de ecuații de dimensiuni mari) se poate reduce la calculul câtorva transformări Fourier discrete, pentru care algoritmul FFT furnizează o drastică reducere a volumului de calcul.

3.6. Modificări geometrice ale imaginilor.

Printre cele mai des întâlnite operații din imagistica computerizată se numără transformările geometrice.

3.6.1. Translația, scalarea şi rotația imaginilor.

Operația de **translație** înseamnă adăugarea unui offset constant la coordonatele fiecărui pixel din zona de interes. Ea poate fi exprimată fie sub forma unui sistem de ecuații, fie (cel mai adesea) sub formă matriceală:

$$\begin{cases}\tilde{x}_m = x_m + t_x \\ \tilde{y}_n = y_n + t_y\end{cases} \Longleftrightarrow \begin{bmatrix}\tilde{x}_m \\ \tilde{y}_n\end{bmatrix} = \begin{bmatrix}x_m \\ y_n\end{bmatrix} + \begin{bmatrix}t_x \\ t_y\end{bmatrix} \tag{98}$$

unde: - x_m, y_n sunt coordonatele punctului care se translează;

- t_x, t_y sunt offset-urile care definesc translația;
- $\tilde{x}_m, \tilde{y}_n$ sunt noile coordonate ale pixelului (m,n).

Dacă toate aceste valori sunt întregi, operația este reversibilă. Pentru t_x, t_y neîntregi este necesară interpolarea lui $f(x,y)$ pentru a obține imaginea translată.

Scalarea se realizează modificând coordonatele pixelilor după regula:

$$\begin{cases}\tilde{x}_m = s_x x_m \\ \tilde{y}_n = s_y y_j\end{cases} \Longleftrightarrow \begin{bmatrix}\tilde{x}_m \\ \tilde{y}_n\end{bmatrix} = \begin{bmatrix}s_x & 0 \\ 0 & s_y\end{bmatrix}\begin{bmatrix}x_m \\ y_n\end{bmatrix} \tag{99}$$

unde s_x, s_y sunt constantele de scalare. Dacă avem $s_x, s_y > 1$, se obține o mărire a imaginii, în caz contrar ea se micşorează. În ambele cazuri este necesară operația de interpolare.

Rotația este descrisă de:

$$\begin{cases} \tilde{x}_m = x_m \cos\theta - y_n \sin\theta \\ \tilde{y}_n = x_m \sin\theta + y_n \cos\theta \end{cases} \Longleftrightarrow \begin{bmatrix} \tilde{x}_m \\ \tilde{y}_n \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_m \\ y_n \end{bmatrix} \tag{100}$$

unde θ este unghiul cu care se face rotația în jurul originii, în sens trigonometric şi în raport cu axa *Ox*.

Operațiile descrise până acum se pot sintetiza într-o singură relație matriceală, *când este cunoscută ordinea de aplicare a acestor transformări*. Pentru cazul unei translații urmată de o rotație, avem următoarea relație liniară:

$$\begin{bmatrix} \tilde{x}_m \\ \tilde{y}_n \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x_m \\ y_n \end{bmatrix} + \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} t_x \\ t_y \end{bmatrix} \tag{101}$$

care traduce practic rotația imaginii în jurul unui "pivot" dat de operația de translație.

Trecând în spațiul extins al coordonatelor, rotația în jurul "pivotului" (x_p,y_p) este:

$$\begin{bmatrix} \tilde{x}_m \\ \tilde{y}_n \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & -x_p\cos\theta + y_p\sin\theta + x_p \\ \sin\theta & \cos\theta & -x_p\sin\theta - y_p\cos\theta - y_p \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_m \\ y_n \\ 1 \end{bmatrix} \tag{102}$$

şi analog translația şi scalarea:

$$\begin{bmatrix} \tilde{x}_m \\ \tilde{y}_n \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_m \\ y_n \\ 1 \end{bmatrix} \qquad \begin{bmatrix} \tilde{x}_m \\ \tilde{y}_n \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_m \\ y_n \\ 1 \end{bmatrix} \tag{103}$$

De remarcat că, în anumite condiții, operația de rotație a unei imagini poate fi aproximată cu o transformare separabilă, astfel:

$$\begin{cases} \tilde{x}_m = x_m \cos\theta - y_n \sin\theta \\ \tilde{y}_n = \left[\dfrac{\tilde{x}_m + y_n \sin\theta}{\cos\theta} \right] \sin\theta + y_n \cos\theta \end{cases} \tag{104}$$

În primul pas al transformării se face calculul lui $\tilde{x}_m$, rezultând o imagine independentă de x_m, după care rotația se completează cu a doua formulă. Pentru unghiuri apropiate de $\pi/2$, operația nu este recomandabilă deoarece se produce o scădere considerabilă a rezoluției.

Transformarea este utilă, de exemplu, la "îndreptarea" paginilor scanate puțin înclinate (rotite cu câteva grade), ştiut fiind faptul că, cel mai adesea, trăsăturile care se extrag din imaginile textelor tipărite nu sunt invariante la rotație.

3.6.2. Deformări geometrice polinomiale.

Deformările geometrice ale imaginilor se utilizează fie pentru obținerea unor efecte speciale, fie pentru compensarea unor distorsiuni geometrice cunoscute. Ele pot fi descrise generic de:

$$\begin{cases} \tilde{x}_m = \boldsymbol{X}(x_m,y_n) \\ \tilde{y}_n = \boldsymbol{Y}(x_m,y_n) \end{cases} \tag{105}$$

Foarte des utilizate sunt deformările geometrice polinomiale de ordinul doi, descrise de următoarele relații:

$$\begin{cases} \tilde{x}_m = a_0 + a_1 x_m + a_2 y_n + a_3 x_m^2 + a_4 x_m y_n + a_5 y_n^2 \\ \tilde{y}_n = b_0 + b_1 x_m + b_2 y_n + b_3 x_m^2 + b_4 x_m y_n + b_5 y_n^2 \end{cases} \tag{106}$$

care pot fi exprimate şi matriceal prin:

$$\begin{bmatrix}\tilde{x}_m\\ \tilde{y}_n\end{bmatrix}=\begin{bmatrix}a_0 & a_1 & a_2 & a_3 & a_4 & a_5\\ b_0 & b_1 & b_2 & b_3 & b_4 & b_5\end{bmatrix}\begin{bmatrix}1 & x_m & y_n & x_m^2 & x_m y_n & y_n^2\end{bmatrix}^t \tag{107}$$

Constantele a_0,a_1,a_2,a_3,a_4,a_5 şi respectiv b_0,b_1,b_2,b_3,b_4,b_5 se determină cunoscând coordonatele în imaginea transformată ale unui set de *M* pixeli.

Notând:

$$\boldsymbol{x}=\begin{bmatrix}x_1 & x_2 & \dots & x_M\end{bmatrix} \qquad \boldsymbol{y}=\begin{bmatrix}y_1 & y_2 & \dots & y_M\end{bmatrix} \tag{108}$$
$$\boldsymbol{a}=\begin{bmatrix}a_1 & a_2 & a_3 & a_4 & a_5\end{bmatrix} \quad \boldsymbol{b}=\begin{bmatrix}b_1 & b_2 & b_3 & b_4 & b_5\end{bmatrix}$$

eroarea medie pătratică dintre imaginile transformate reală şi ideală este:

$$\varepsilon^2=(\boldsymbol{x}-\boldsymbol{A}\boldsymbol{a})^t(\boldsymbol{x}-\boldsymbol{A}\boldsymbol{a})+(\boldsymbol{y}-\boldsymbol{A}\boldsymbol{b})^t(\boldsymbol{y}-\boldsymbol{A}\boldsymbol{b}) \tag{109}$$

unde :

$$\boldsymbol{A}=\begin{bmatrix}1 & x_1 & y_1 & x_1^2 & x_1y_1 & y_1^2\\ 1 & x_2 & y_2 & x_2^2 & x_2y_2 & y_2^2\\ \dots & \dots & \dots & \dots & \dots & \dots\\ 1 & x_M & y_M & x_M^2 & x_My_M & y_M^2\end{bmatrix} \tag{110}$$

Se demonstrează că eroarea este minimă dacă:

$$\boldsymbol{a}=(\boldsymbol{A}^t\boldsymbol{A})^{-1}\boldsymbol{A}^t\boldsymbol{x} \qquad \boldsymbol{b}=(\boldsymbol{A}^t\boldsymbol{A})^{-1}\boldsymbol{A}^t\boldsymbol{y} \tag{111}$$

unde se remarcă utilizarea inversei generalizate a matricii $\boldsymbol{A}$**:**

$$\boldsymbol{A}^*=(\boldsymbol{A}^t\boldsymbol{A})^{-1}\boldsymbol{A}^t \tag{112}$$

Câteva exemple de imagini distorsionate geometric sunt date în continuare:

3.6.3. Proiecția paralelă şi proiecția perspectivă.

Pentru ***proiecția paralelă*** caracteristic este faptul că toate razele de proiecție sunt paralele între ele.

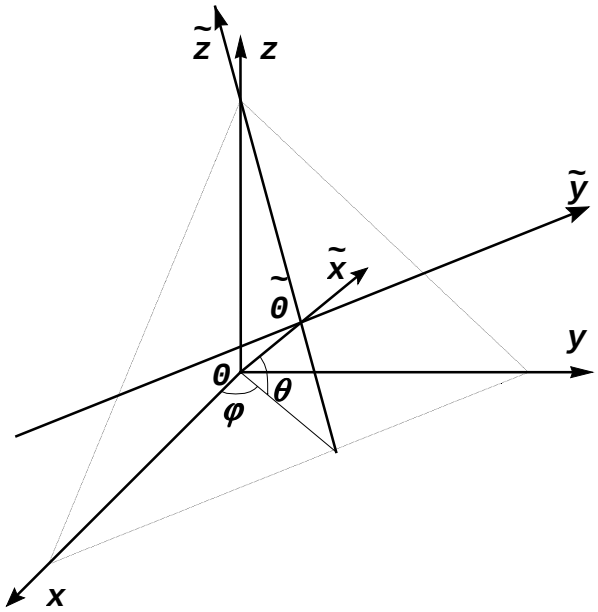
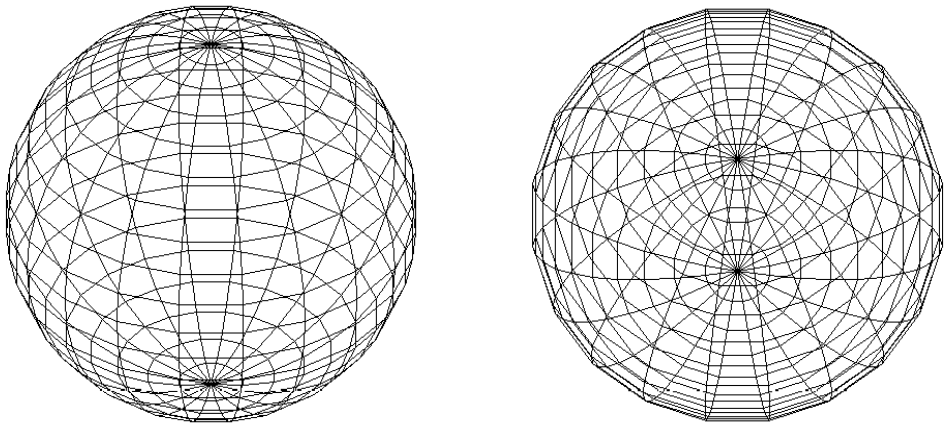


Fig. 10. Principiul proiecției paralele.

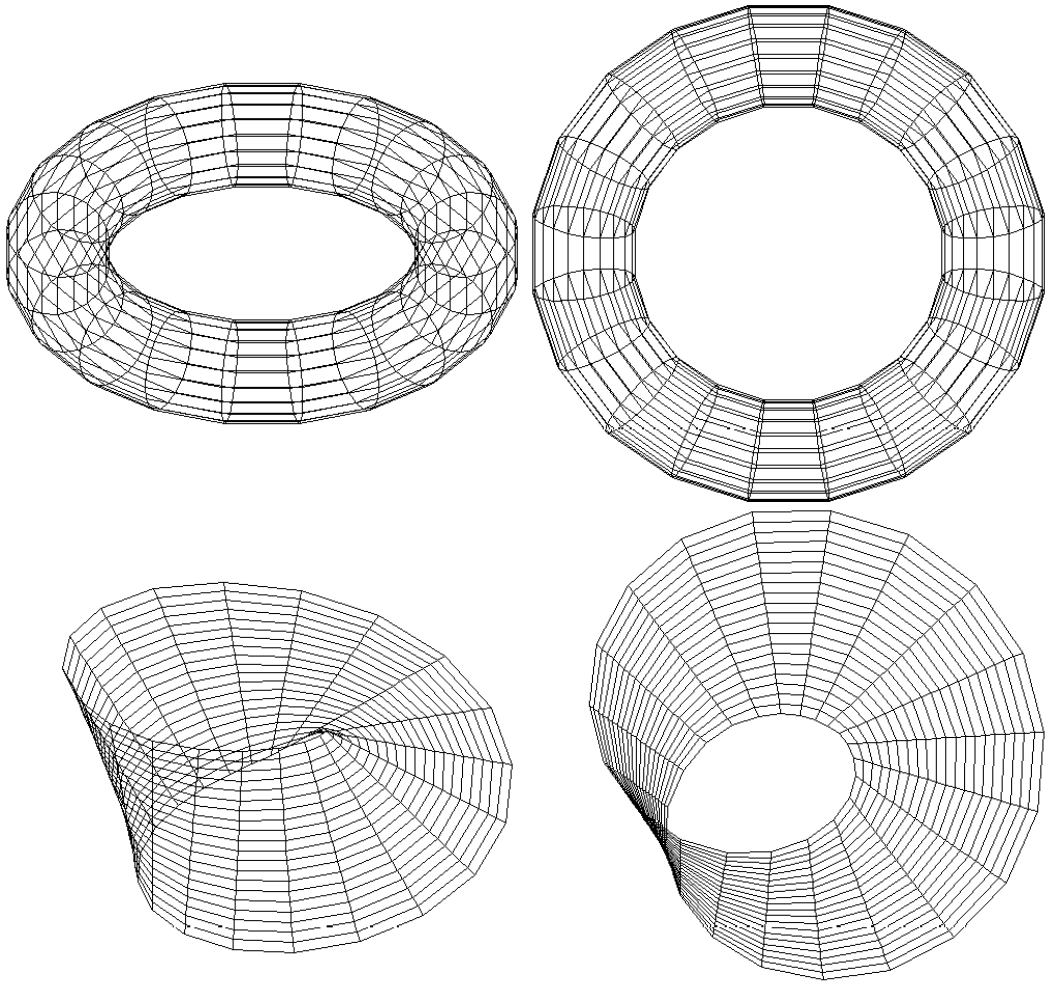
Din studiul schemei precedente se remarcă faptul că proiecția paralelă se poate reduce la o simplă schimbare de coordonate şi anume o rotație în jurul axei *Oz* cu un unghi φ, urmată de o rotație cu unghiul θ în jurul noii axe *Oy'*.

$$\begin{bmatrix}\tilde{x}\\ \tilde{y}\\ \tilde{z}\end{bmatrix}=\begin{bmatrix}\cos\theta & 0 & -\sin\theta\\ 0 & 1 & 0\\ \sin\theta & 0 & \cos\theta\end{bmatrix}\begin{bmatrix}\cos\varphi & -\sin\varphi & 0\\ \sin\varphi & \cos\varphi & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}x\\ y\\ z\end{bmatrix} \tag{113}$$

$$\begin{bmatrix}\tilde{x}\\ \tilde{y}\\ \tilde{z}\end{bmatrix}=\begin{bmatrix}\cos\varphi\cos\theta & -\sin\varphi\cos\theta & -\sin\theta\\ \sin\varphi & \cos\varphi & 0\\ \cos\varphi\sin\theta & -\sin\varphi\sin\theta & \cos\theta\end{bmatrix}\begin{bmatrix}x\\ y\\ z\end{bmatrix} \tag{114}$$



Exemple de proiecții paralele.



În figurile anterioare sunt prezentate proiecțiile paralele pentru câteva suprafețe: sferă, tor şi banda Moebius.

Trecând la coordonate ecran avem:

$$\begin{bmatrix} x_s \\ y_x \end{bmatrix} = \begin{bmatrix} \sin\varphi & \cos\varphi & 0 \\ \cos\varphi\sin\theta & -\sin\varphi\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{115}$$

Proiecția perspectivă este mult mai apropiată de cazul real al observării realității prin intermediul unui sistem optic tipic. Geometria ei este descrisă în următoarea schemă, în care (x,y,z) sunt coordonatele punctului curent din spațiu, (x_p, y_p) sunt coordonatele proiecției lui în planul imaginii, iar *f* este distanța focală a sistemului de lentile:

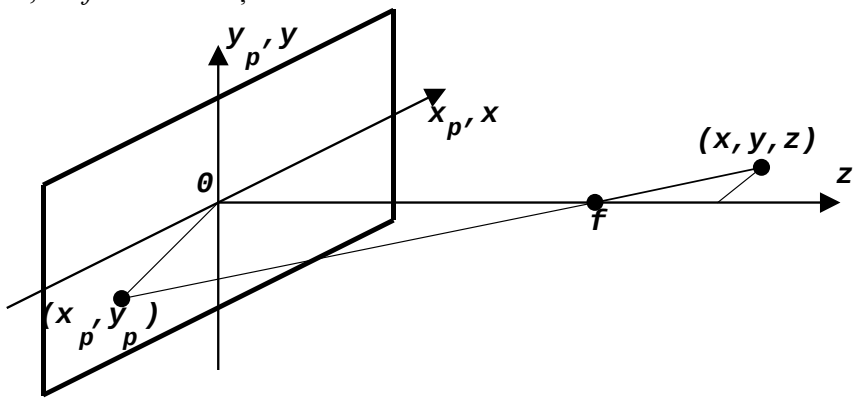


Fig. 11. Principiul proiecției perspective.

Se pot scrie imediat relațiile: $x_p = \frac{fx}{f-z}$ şi $y_p = \frac{fy}{f-z}$ (116)

Trecând la coordonate omogene Roberts în [91] deduce următoarea formulă de proiecție perspectivă:

$$\begin{bmatrix} x_p & y_p & z_p \end{bmatrix} = \begin{bmatrix} \frac{fx}{f-z} & \frac{fy}{f-z} & \frac{fz}{f-z} \end{bmatrix} \quad (117)$$

din care evident, interesează în marea majoritate a cazurilor doar (x_p, y_p).

3.6.4. Modelul capturii unei imagini cu o camera de luat vederi.

Modelul geometric clasic al utilizării unei camere de luat vederi (Fu[41]) poate fi discutat utilizând figura următoare:

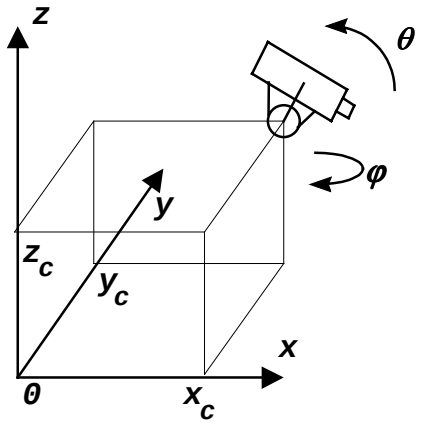


Fig. 12. Modelul geometric al camerei de luat vederi.

unde (x_c, y_c, z_c) sunt coordonatele centrului articulației cardanice pe care se sprijină camera de luat vederi. Față de aceste coordonate, centrul imaginii plane (planul de proiecție) se obține adăugând offset-ul (x_0, y_0, z_0), printr-o operație de translație. Față de (x_c, y_c, z_c), planul de proiecție poate fi rotit, datorită articulației cardanice, cu unghiul φ în plan orizontal (raportat la axa $0x$) şi cu unghiul θ în plan vertical. Adăugând la toate acestea proiecția perspectivă a obiectelor vizualizate prin cameră, se obțin relațiile:

$$x_p = \frac{[(x-x_c)\cos\varphi - (y-y_c)\sin\varphi - x_0]f}{f - (x-x_c)\sin\varphi\sin\theta - (y-y_c)\cos\varphi\sin\theta - (z-z_c)\cos\theta + z_0} \quad (118)$$

$$y_p = \frac{[(x-x_c)\sin\varphi\cos\theta + (y-y_c)\cos\varphi\cos\theta - (z-z_c)\sin\theta - y_0]f}{f - (x-x_c)\sin\varphi\sin\theta - (y-y_c)\cos\varphi\sin\theta - (z-z_c)\cos\theta + z_0} \quad (119)$$

3.6.5. Reeşantionarea geometrică a imaginilor.

După cum s-a menționat la discuțiile asupra operației de scalare, în cazul discret, de cele mai multe ori este necesară o interpolare pentru reconstrucția imaginii scalate, ceea ce este echivalentă cu reeşantionarea imaginii continue inițiale, pentru a obține noua imagine scalată. Acest proces de reeşantionare înseamnă atribuirea unor noi valori pixelilor din noua grilă de eşantionare, funcție de valorile pixelilor corespunzători din vechea grilă:

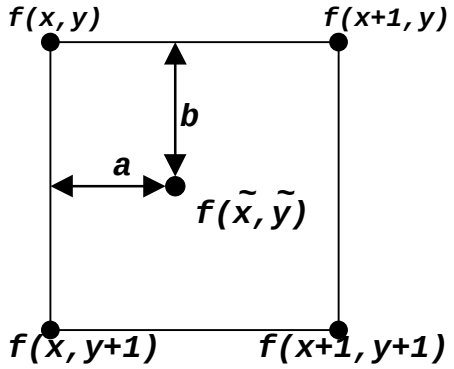


Fig. 13. Reeşantionarea geometrică a imaginilor.

Cea mai simplă soluție este folosirea unui interpolator bilinear, descris de următoarea relație:

$$\boldsymbol{f}(\tilde{x},\tilde{y}) = (1-a)[(1-b)\,\boldsymbol{f}(x,y) + b\;\boldsymbol{f}(p,q+1)] + a[(1-b)\,\boldsymbol{f}(x+1,y) + b\;\boldsymbol{f}(p+1,q+1] \tag{120}$$

Ea poate fi generalizată folosind o funcție de interpolare $R(x)$ şi pixelii dintr-o vecinătate (de obicei maximum 4x4) a imaginii inițiale.

$$\boldsymbol{f}(\tilde{x},\tilde{y}) = \sum_{m=-1}^{2}\sum_{n=-1}^{2} \boldsymbol{f}(x+m,y+n)\,\boldsymbol{R}(m-a)\,\boldsymbol{R}(n-b) \tag{121}$$

Pentru mărirea imaginilor cu un factor întreg există algoritmi care se bazează pe utilizarea operației de convoluție. Imaginea inițială este adusă mai întâi la dimensiunea $ZMxZN$, unde Z este factorul întreg de mărire, astfel încât:

$$\tilde{\boldsymbol{f}}(\tilde{x},\tilde{y}) = \begin{cases} \boldsymbol{f}(x,y), & \text{pentru } \tilde{x} = Zx, \tilde{y} = Zy \\ 0, & \text{pentru } \tilde{x} \neq Zx, \tilde{y} = Zy \end{cases} \tag{122}$$

după care este supusă unei convoluții cu o mască convenabil aleasă. Pentru $Z = 2$ se poate folosi una din măştile:

$$M_1 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \quad M_2 = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad M_3 = \frac{1}{16}\begin{bmatrix} 1 & 3 & 3 & 1 \\ 3 & 9 & 9 & 3 \\ 3 & 9 & 9 & 3 \\ 1 & 3 & 3 & 1 \end{bmatrix} \quad M_4 = \frac{1}{64}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \tag{123}$$

Pentru cazul reeşantionării unei imagini cu un factor rațional $Z = P/Q$, o transformare originală poate fi folosită (Bulea[21],[23]). Eficiența ei constă în faptul că pentru efectuarea ei sunt necesare doar operații cu numere întregi. Cu cât numerele P şi Q sunt mai mici, cu atât viteza de operare este mai mare. În continuare sunt prezentate exemple de utilizare a acestui algoritm, pentru cazul unei imagini 10x14 care se dorește adusă la dimensiunile 4x5. Deci $Z(x) = 10/4$, iar $Z(y) = 14/5$. Modul de funcționare al algoritmului rezultă din figurile următoare:

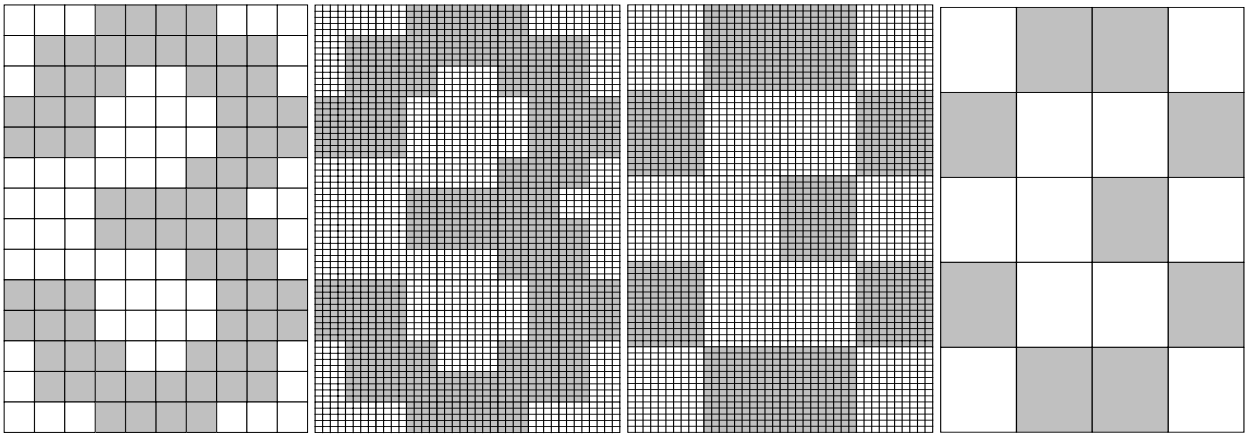
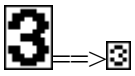


Fig. 14. Reeşantionarea geometrică cu factor raţional.

Deci se realizează transformarea: 

3.7. Compensarea distorsiunilor de nelinearitate şi a celor de spectru al luminii.

Studiul modelului general al degradării imaginii a dus la concluzia prezenţei unor nelinearităţi care nu au fost luate în consideraţie până acum. Modelul general al procesului de degradare a imaginii fiind:

$$\boldsymbol{g}(x,y) = \boldsymbol{d}\left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \boldsymbol{h}(x,y;x',y')\,\boldsymbol{f}(x',y')dx'xy'\right] + \boldsymbol{s}[\,\boldsymbol{g}(\,\boldsymbol{w}(x,y))]\,\boldsymbol{\eta}_1(x,y) + \boldsymbol{\eta}_2(x,y) \qquad (124)$$

s-au făcut observaţiile că funcţia neliniară $d(-)$ modelează nelinearităţile introduse de blocul de detecţie a imaginii, iar zgomotul conţine două componente dintre care una dependentă de semnal prin intermediul nelinearităţii $s(-)$, care modelează blocul de înregistrare a imaginii.

Condiţia esenţială pe care trebuie să o îndeplinească nelinearităţile menţionate pentru a putea fi compensate este ca ele să fie descrise de *funcţii bijective* pe domeniul de definiţie, caz în care se poate defini inversa acestor funcţii. Astfel dacă:

$$\boldsymbol{g}(x,y) = \boldsymbol{d}\,[\,\boldsymbol{f}(x,y)] \qquad (125)$$

şi funcţia $\boldsymbol{d}(-)$ este bijectivă, există inversa ei $\boldsymbol{d}^{-1}(-)$ şi avem:

$$\boldsymbol{f}(x,y) = \boldsymbol{d}^{-1}\,[\,\boldsymbol{g}(x,y)] \qquad (126)$$

În practică, adesea mai trebuie adusă o corecţie de scară, după cum rezultă din figurile următoare:

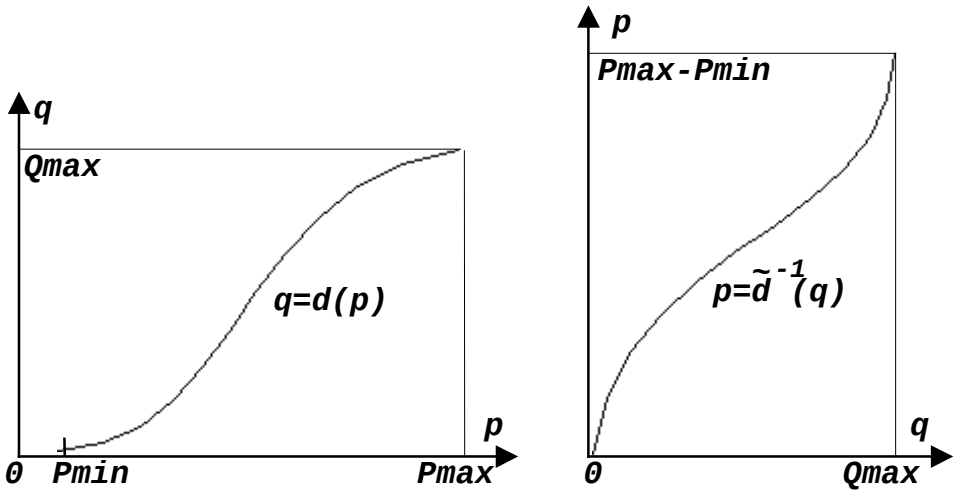


Fig. 15. Compensarea distorsiunilor de nelinearitate.

$$q = \boldsymbol{d}(p), P\,\boldsymbol{min} \le p \le P\,\boldsymbol{max} \;\Longrightarrow\; p = \frac{\tilde{\boldsymbol{d}}^{-1}(q) - P_{MIN}}{P_{MAX} - P_{MIN}} + P_{MIN} \tag{127}$$

Pentru determinarea alurii acestor nelinearități se utilizează imagini de test construite special, astfel încât, prin măsurători, să se poată deduce dependența intrare-ieşire dorită.

Implementarea practică a dispozitivului de corecție se poate face analogic (mai rar), caz în care este necesară proiectarea unui circuit nelinear cu răspunsul dat de inversa nelinearității care afectează imaginea. Implementarea digitală a circuitului de corecție se traduce printr-un dispozitiv de tip memorie LUT (Look Up Table) care să translateze codurile care i se aplică la intrare.

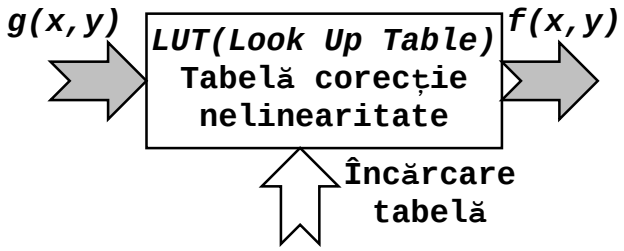


Fig. 15. Memoria LUT (Look Up Table).

Un alt tip de corecții se referă la corecțiile nelinearităților datorate răspunsului specific al senzorilor de imagine la spectrul luminii incidente (O'Handley[78]), respectiv al dispozitivelor de afişare.

Astfel, pentru cazul senzorilor, luminanța curentă furnizată de aceştia este descrisă de:

$$\boldsymbol{L} = \int_{\lambda} \boldsymbol{C}(\lambda)\,\boldsymbol{S}(\lambda)\,d\lambda \tag{128}$$

unde: λ este lungimea de undă a radiației luminoase incidente,
$\boldsymbol{C}(\lambda)$ este energia spectrală a luminii incidente, iar
$\boldsymbol{S}(\lambda)$ este răspunsul spectral al senzorului.

Dacă răspunsul dorit este:

$$\tilde{\boldsymbol{L}} = \int_{\lambda} \boldsymbol{C}(\lambda)\,\tilde{\boldsymbol{S}}(\lambda)\,d\lambda \tag{129}$$

în ipoteza minimizării erorii medii pătratice date de:

$$\boldsymbol{e} = \int_{\lambda} [\,\boldsymbol{L}(\lambda) - \tilde{\boldsymbol{L}}(\lambda)]^2\,d\lambda \tag{130}$$

se va căuta o dependență de forma:

$$\tilde{\boldsymbol{L}} = \boldsymbol{f}(\,\boldsymbol{L}\,) \tag{131}$$

Cel mai adesea, pentru simplitate, se caută parametrii unei dependențe liniare:

$$\tilde{\boldsymbol{L}}(\lambda) = a\,\boldsymbol{S}(\lambda) + b \tag{132}$$

Compensarea distorsiunilor de spectru se rezolvă la fel şi pentru cazul afişării imaginii. În cazul imaginilor color problema se complică, fiind necesare corecții pentru fiecare componentă de culoare în parte.

4. Detecţia muchiilor, liniilor şi spoturilor.

4.1. Metode spaţiale de detecţie a muchiilor.

Principiul detecţiei de muchii rezultă destul de clar studiind cazul unidimensional prezentat în figura următoare. Principial, detecţia unei muchii se traduce prin localizarea variaţiilor bruşte în semnalul de intrare.

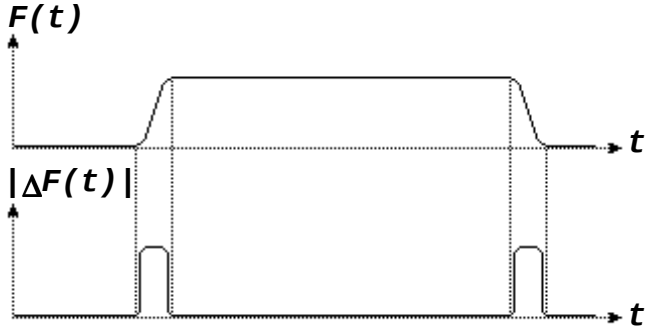


Fig. 1. Principiul detecţiei de muchii.

4.1.1. Metode bazate pe operatori de diferenţiere de ordinul unu.

În cazul imaginilor continue, după cum s-a arătat deja la accentuarea de muchii, toată informaţia privind variaţiile locale de luminozitate este concentrată în vectorul gradient:

$$\Delta \boldsymbol{f}(x,y) = \frac{\partial \boldsymbol{f}(x,y)}{\partial x^2}\vec{i} + \frac{\partial \boldsymbol{f}(x,y)}{\partial y^2}\vec{j} \tag{1}$$

Operatorul de detecţie de muchii trebuie să fie izotrop, pentru a răspunde similar în prezenţa muchiilor având orientări diferite. Se poate demonstra uşor că modulul gradientului este izotrop.

Pentru cazul discret se folosesc aproximaţiile următoare :

$$\Delta_x \boldsymbol{f}(x,y) = \boldsymbol{f}(x+1,y) - \boldsymbol{f}(x,y)$$
$$\Delta_y \boldsymbol{f}(x,y) = \boldsymbol{f}(x,y+1) - \boldsymbol{f}(x,y) \tag{2}$$

sau uneori se apelează la forme simetrice ale diferenţelor finite:

$$\Delta_x \boldsymbol{f}(x,y) = \boldsymbol{f}(x+1,y) - \boldsymbol{f}(x-1,y)$$
$$\Delta_y \boldsymbol{f}(x,y) = \boldsymbol{f}(x,y+1) - \boldsymbol{f}(x,y-1) \tag{3}$$

Deci se poate folosi drept detector de muchii modulul gradientului discret:

$$\boldsymbol{G}_1 = \sqrt{[\Delta_x \boldsymbol{f}(x,y)]^2 + [\Delta_y \boldsymbol{f}(x,y)]^2} \tag{4}$$

Se pot utiliza cu succes aproximări ale **gradientului discret** care nu mai sunt izotrope, în schimb se calculează mult mai repede:

$$\boldsymbol{G}_2 = |\Delta_x \boldsymbol{f}(x,y)| + |\Delta_y \boldsymbol{f}(x,y)| \tag{5}$$

$$\boldsymbol{G}_3 = \boldsymbol{max}(|\Delta_x \boldsymbol{f}(x,y)| + |\Delta_y \boldsymbol{f}(x,y)|) \tag{6}$$

$$\boldsymbol{G}_4 = \boldsymbol{max}(|\boldsymbol{f}(x,y) - \boldsymbol{f}(x+m,y+n)|),\ m,n \in \{-1,0,1\} \tag{7}$$

$$\boldsymbol{G}_5 = \boldsymbol{max}(|\boldsymbol{f}(x,y) - \boldsymbol{f}(x+1,y+1)|, |\boldsymbol{f}(x,y+1) - \boldsymbol{f}(x+1,y)|) \tag{8}$$

care este denumit gradientul Roberts, şi care poate fi exprimat prin intermediul convoluţiilor cu măştile:

$$R_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad R_2 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{9}$$

Modul de operare în cazul măştilor este cel cunoscut: se calculează cele două rezultate ale operaţiilor cu măştile de mai sus, se află modulul lor şi se reţine valoarea maximă.

Alţi gradienţi definiţi prin măşti 3x3 sunt Prewitt, Sobel şi Frei-Chen, daţi prin măştile corespunzătoare muchiilor verticale descrescătoare de la stânga la dreapta:

$$P_1 = \frac{1}{3}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \quad S_1 = \frac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad F_1 = \frac{1}{2+\sqrt{2}}\begin{bmatrix} 1 & 0 & -1 \\ \sqrt{2} & 0 & -\sqrt{2} \\ 1 & 0 & -1 \end{bmatrix} \tag{10}$$

Frei şi Chen în [40] propun descompunerea vecinătăţii 3x3 care conţine pixelul curent, folosind următoarea bază ortogonală:

$$H_0 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$H_1 = \begin{bmatrix} -1 & 0 & 1 \\ -\sqrt{2} & 0 & \sqrt{2} \\ -1 & 0 & 1 \end{bmatrix} \quad H_3 = \begin{bmatrix} 0 & 1 & -\sqrt{2} \\ -1 & 0 & 1 \\ \sqrt{2} & -1 & 0 \end{bmatrix} \quad H_5 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \quad H_7 = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

$$H_2=\begin{bmatrix}-1 & -\sqrt{2} & -1\\ 0 & 0 & 0\\ 1 & \sqrt{2} & 1\end{bmatrix}\ H_4=\begin{bmatrix}\sqrt{2} & -1 & 0\\ -1 & 0 & 1\\ 0 & 1 & -\sqrt{2}\end{bmatrix}\ H_6=\begin{bmatrix}-1 & 0 & 1\\ 0 & 0 & 0\\ 1 & 0 & -1\end{bmatrix}\ H_8=\begin{bmatrix}-2 & 1 & -2\\ 1 & 4 & 1\\ -2 & 1 & -2\end{bmatrix} \quad (11)$$

Descompunerea care rezultă este de forma:

$$\boldsymbol{f(X)}=\sum_{k=0}^{8}\boldsymbol{f}_k\,\boldsymbol{H}_k/\|\boldsymbol{H}_k\| \quad (12)$$

unde: $\boldsymbol{X}$ - vecinătatea 3x3 curentă;

$$\boldsymbol{f}_k=\langle \boldsymbol{f(X)},\boldsymbol{H}_k\rangle=\sum_i\sum_j \boldsymbol{f}(x_i,y_j)\boldsymbol{H}_k \quad (13)$$

$$\|\boldsymbol{H}_k\|=\langle \boldsymbol{H}_k,\boldsymbol{H}_k\rangle \quad (14)$$

Se remarcă din structura bazei faptul că, practic, informaţii privind muchiile sunt conţinute doar în coeficienţii unu şi doi ai dezvoltării, deci în:

$$E=\sum_{k=1}^{2}\boldsymbol{f}_k^2 \quad (15)$$

Normalizarea valorii obţinute se face cu valoarea:

$$S=\sum_{k=1}^{8}\boldsymbol{f}_k^2 \quad (16)$$

ceea ce are ca efect minimizarea efectelor zgomotelor. Se obţine o evaluare a "tăriei" muchiei calculând:

$$\boldsymbol{\cos\theta}=\sqrt{E/S} \quad (17)$$

Stabilind pragul T, regula de decizie este în acest caz:

$$\begin{cases}\theta > T \Rightarrow \text{muchie absentă}\\ \theta \le T \Rightarrow \text{muchie prezentă}\end{cases} \quad (18)$$

În aceeaşi clasă de operatori spaţiali pentru detecţie de muchii intră aşa-numitele **şabloane (prototipuri)** pentru detecţia de muchii. Ele se bazează pe calculul modulului gradientului discret într-un număr mare de direcţii, cu reţinerea valorii maxime:

$$\boldsymbol{G}(x,y)=\boldsymbol{max}\,[|\boldsymbol{G}_1(x,y)|,|\boldsymbol{G}_2(x,y)|,\ldots,|\boldsymbol{G}_K(x,y)|] \quad (19)$$

Astfel se construieşte un alt set de detectori de muchii, dintre care mai cunoscuţi sunt Kirsh şi Compass, daţi prin măştile de referinţă următoare:

$$K_1=\frac{1}{15}\begin{bmatrix}5 & -3 & -3\\ 5 & 0 & -3\\ 5 & -3 & -3\end{bmatrix} \qquad C_1=\frac{1}{5}\begin{bmatrix}1 & 1 & -1\\ 1 & -2 & -1\\ 1 & 1 & -1\end{bmatrix} \quad (20)$$

Restul măştilor se obţin prin rotirea elementelor măştii date în jurul elementului central. Rezultă astfel un număr de până la 8 măşti direcţionale, fiecare responsabilă de muchiile având o anume direcţie. Se pot defini analog măşti de detecţie muchii având dimensiuni mai mari: 5x5, 7x7, 9x9.

Tot măşti de dimensiuni mai mari se pot construi astfel încât ele să includă practic, într-o singură mască (**mască compusă**), o operaţie de netezire a imaginii urmată de o detecţie de muchii. Astfel, masca echivalentă unei neteziri cu masca M_5, urmată de un operator Prewitt pentru muchii verticale rezultă:

$$\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}\otimes\frac{1}{3}\begin{bmatrix}1&0&-1\\1&0&-1\\1&0&-1\end{bmatrix}\Rightarrow\frac{1}{18}\begin{bmatrix}1&1&0&-1&-1\\2&2&0&-2&-2\\3&3&0&-3&-3\\2&2&0&-2&-2\\1&1&0&-1&-1\end{bmatrix} \tag{21}$$

Evident, pot fi utilizate o multitudine de combinaţii de asemenea operatori pentru netezire şi extragere de muchii.

Argyle [05] şi Macleod [67] au propus utilizarea unor vecinătăţi mari cu măşti de netezire construite după o lege gaussiană bidimensională, compuse cu operatori de detecţie de muchii de aceeaşi dimensiune.

DroG (**Der**ivative **o**f **G**aussian) este denumirea unui alt operator compus, definit de răspunsul la impuls:

$$\boldsymbol{h}_0(x,y)=-\frac{\partial[\,\boldsymbol{g}(x,\sigma_x)\,\boldsymbol{g}(y,\sigma_y)]}{\partial x}=\frac{x}{\sigma_x^2}\boldsymbol{g}(x,\sigma_x)\,\boldsymbol{g}(y,\sigma_y) \tag{22}$$

unde:

$$\boldsymbol{g}(x,\sigma_x)=\frac{1}{\sigma_x\sqrt{2\pi}}\boldsymbol{exp}\left(-\frac{x^2}{2\sigma_x^2}\right)\qquad \boldsymbol{g}(x,\sigma_y)=\frac{1}{\sigma_y\sqrt{2\pi}}\boldsymbol{exp}\left(-\frac{y^2}{2\sigma_y^2}\right) \tag{23}$$

Muchiile se extrag cu:

$$\widetilde{\boldsymbol{f}}(x,y)=\boldsymbol{f}(x,y)*\boldsymbol{h}_0(x,y) \tag{24}$$

Canny [29] este cel care a dezvoltat o procedură sistematică de evaluare a performanţelor unui detector de muchii, luând în considerare pentru aceasta trei elemente: detecţia propriu-zisă a muchiei, precizia localizării ei şi unicitatea răspunsului (pentru a evita cazurile de dublare a muchiilor detectate).

4.1.2. Metode bazate pe operatori de diferenţiere de ordinul doi.

O altă categorie de metode are la bază folosirea operatorilor discreţi de diferenţiere, de ordinul doi, care aproximează Laplacianul:

$$\nabla\boldsymbol{f}(x,y)=\frac{\partial^2\boldsymbol{f}(x,y)}{\partial x^2}+\frac{\partial^2\boldsymbol{f}(x,y)}{\partial y^2} \tag{25}$$

Aproximările discrete ale Laplacianului se traduc prin operaţii cu una din măştile:

$$L_1=\begin{bmatrix}0&-1&0\\-1&4&-1\\0&-1&0\end{bmatrix}\quad L_2=\begin{bmatrix}-1&-1&-1\\-1&8&-1\\-1&-1&-1\end{bmatrix}\quad L_3=\begin{bmatrix}-2&1&-2\\1&4&1\\-2&1&-2\end{bmatrix} \tag{26}$$

Trecerile prin zero ale operatorilor de diferenţiere de ordinul doi pot fi utilizate pentru detecţia muchiilor:

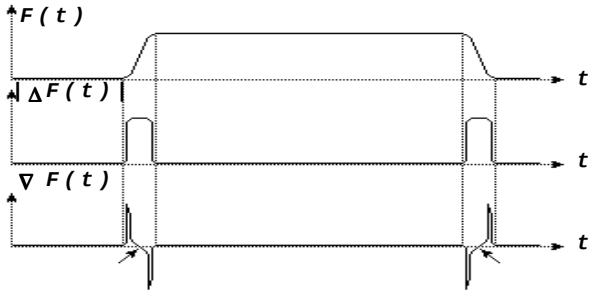


Fig. 2. Operatori de diferenţiere de ordinul doi.

Marr şi Hildrith în [68] au propus utilizarea operatorului **LoG** (**L**aplacian **o**f **G**aussian) pentru estimarea Laplacianului discret, după o procedură similară operatorului **DroG**, adică:

$$\tilde{f}(x,y) = -\nabla\{ f(x,y) * h_1(x,y)\} \tag{27}$$

$$h_1(x,y) = g(x,\sigma)\, g(y,\sigma) \tag{28}$$

Din proprietăţile produsului de convoluţie obţinem:

$$\tilde{f}(x,y) = f(x,y) * \nabla\{ g(x,\sigma_x)\, g(y,\sigma_y)\} = f(x,y) * h_L(x,y) \tag{29}$$

Se obţine în final răspunsul la impuls al filtrului de estimare a Laplacianului:

$$h_L(x,y) = \frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{30}$$

a cărui reprezentare grafică se numeşte "pălăria mexicană":

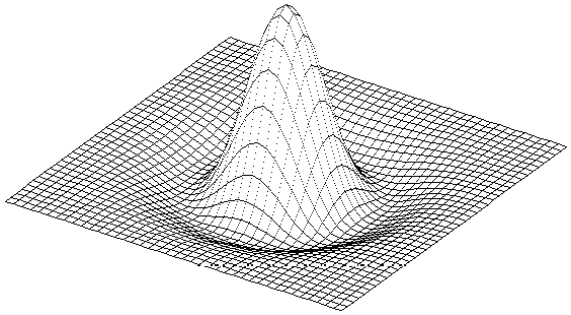


Fig. 3. Răspunsul la impuls al filtrului de estimare a Laplacianului.

Detecţia trecerilor prin zero ale Laplacianului se poate face folosind metoda sistematică dezvoltată de Huertas şi Medioni în [53]. Ei furnizează un catalog al situaţiilor care constituie "treceri prin zero" ale Laplacianului estimat anterior:

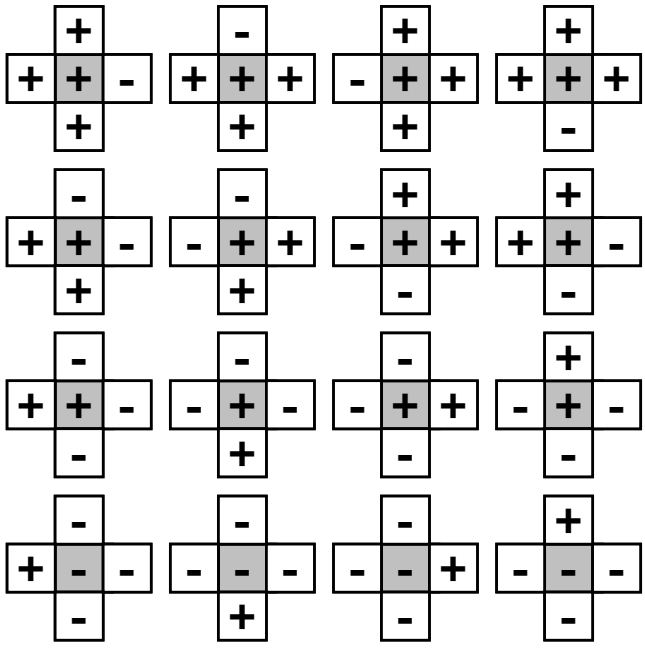


Fig. 4. Detecţia trecerilor prin zero ale Laplacianului.

unde pixelii notaţi "+" semnifică valoare ≥ 0 pentru pixel, iar cei marcaţi cu "-" înseamnă valoare ≤ 0. Algoritmul furnizează şi informaţii privind direcţia muchiilor detectate şi poate fi implementat eficient folosind operatori morfologici.

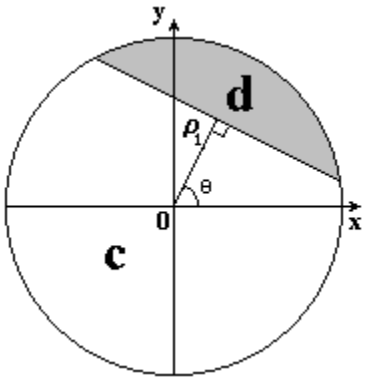
O altă soluţie (Robinson[92]) constă în estimarea iniţială a direcţiei muchiei, urmată de calculul derivatei unidimensionale de ordinul doi în lungul muchiei. Această derivată are expresia:

$$f''(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2}\cos^2\theta + \frac{\partial^2 f(x,y)}{\partial x \partial y}\sin\theta\cos\theta + \frac{\partial^2 f(x,y)}{\partial y^2}\sin^2\theta \qquad (31)$$

unde θ este unghiul între direcţia muchiei şi abscisă. Haralick în [44] utilizează pentru determinarea lui θ şi a derivatei de ordinul doi în această direcţie o metodă care are la bază aproximarea lui $f(x,y)$ cu un polinom cuadratic.

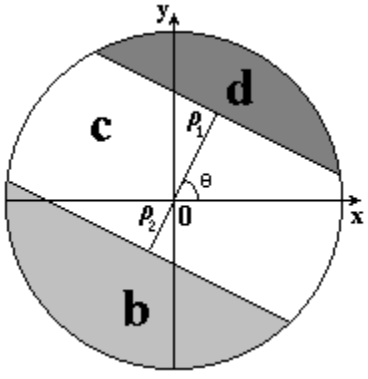
4.2. Metode regionale de detecţie a muchiilor.

O muchie ideală poate fi asimilată cu cea din figura următoare, urmând ca mai apoi să se parcurgă imaginea cu o fereastră circulară de rază *R*, încercând să se determine parametrii muchiei din condiţia de "potrivire" optimă a imaginii şi modelului.



$$f(x,y) = \begin{cases} b, & \text{pentru } (x\cos\theta + y\sin\theta) < \rho \\ b+h, & \text{pentru } (x\cos\theta + y\sin\theta) \geq \rho \end{cases} \qquad (32)$$

Modelul anterior se poate generaliza şi pentru detecţia liniilor din imagine, folosind:



$$f(x,y) = \begin{cases} b, \text{pentru } (x\cos\theta + y\sin\theta) < -\rho_1 \\ c, \text{pentru } -\rho_1 \leq (x\cos\theta + y\sin\theta) \leq \rho_2 \\ d, \text{pentru } \rho_2 < (x\cos\theta + y\sin\theta) \end{cases} \qquad (33)$$

Modelul este caracterizat de nivelele de gri "*b*" şi "*b+h*", de o parte şi de alta a muchiei, orientarea muchiei fiind dată de unghiul "θ" şi distanţa "ρ" faţă de centrul ferestrei circulare. "Potrivirea" se traduce prin minimizarea erorii medii pătratice:

$$e^2 = \iint_{x,y\in C} [f(x,y) - s(x,y,b,h,\rho,\theta)]^2 dx\, dy \qquad (34)$$

sau pentru cazul discret:

$$e^2 = \sum_{x,y\in C} [f(x,y) - s(x,y,b,h,\rho,\theta)]^2 dx\, dy \qquad (35)$$

din minimizarea căreia ar urma să se deducă elementele necunoscute b, h, ρ, θ.

Hueckel în [52] propune descompunerea lui $\boldsymbol{f}(x,y)$ şi respectiv $\boldsymbol{s}(x,y,b,h,\rho,\theta)$ într-o serie folosind drept bază funcţiile ortogonale din setul următor:

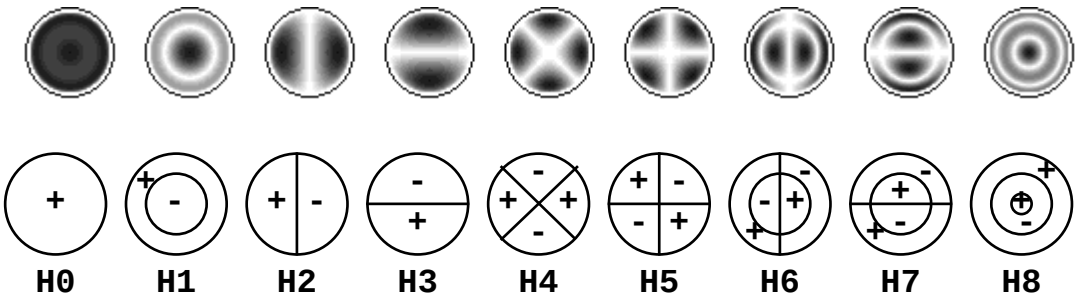


Fig. 5. Setul de funcţii ortogonale Hueckel.

Coeficienţii acestei descompuneri sunt:

$$F_i = \iint_{x,y \in C} \boldsymbol{f}(x,y)\ \boldsymbol{H}_i(x,y)\,dx\,dy \tag{36}$$

$$S_i = \iint_{x,y \in C} \boldsymbol{s}(x,y)\ \boldsymbol{H}_i(x,y)\,dx\,dy \tag{37}$$

şi ca urmare minimizarea erorii medii pătratice este echivalentă cu minimizarea lui:

$$E^2 = \sum_i (F_i - S_i)^2 \tag{38}$$

4.3. Metode statistice de detecţie a muchiilor.

4.3.1. Clasificarea statistică a pixelilor.

Fie P_M probabilitatea apriorică ca un pixel oarecare al imaginii să aparţină unei muchii, iar $P_U = 1 - P_M$ probabilitatea ca un pixel al imaginii să nu fie pixel de muchie, deci să aparţină unei zone relativ uniforme.

Vom nota $\boldsymbol{p}_M[\boldsymbol{G}(x,y)] = \boldsymbol{p}(\boldsymbol{G}(x,y)/(x,y) \in M)$ probabilitatea ca un pixel având gradientul $\boldsymbol{G}(x,y)$ să fie pixel de muchie, iar cu $\boldsymbol{p}_U[\boldsymbol{G}(x,y)] = \boldsymbol{p}(\boldsymbol{G}(x,y)/(x,y) \notin M)$ probabilitatea ca un pixel având gradientul $\boldsymbol{G}(x,y)$ să **nu** fie pixel de muchie. Ele au semnificaţia unor densităţi de probabilitate.

Atunci $P_M\ \boldsymbol{p}_M[\boldsymbol{G}(x,y)]$ este probabilitatea ca un pixel oarecare al imaginii să aparţină unei muchii, iar $P_U\ \boldsymbol{p}_U[\boldsymbol{G}(x,y)]$ este probabilitatea ca aceaşi pixel să **nu** aparţină unei muchii.

Probabilitatea globală de clasificare eronată a pixelilor din imagine este:

$$\boldsymbol{P}_E = \int_0^T P_M\ \boldsymbol{p}_M(G)dG + \int_T^{+\infty} P_U\ \boldsymbol{p}_U(G)dG \tag{39}$$

Alegând pragul astfel încât probabilitatea globală de eroare să fie minimă rezultă (Rosenfeld[94]):

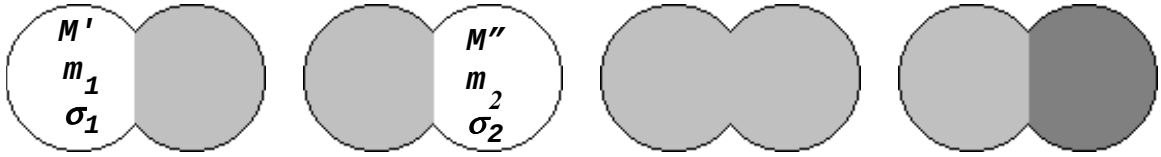
$$P_M\ \boldsymbol{p}_M(T) = P_U\ \boldsymbol{p}_U(T) \tag{40}$$

Abdou şi Pratt [01] au determinat valorile T, P_M şi P_U pentru diferiţi gradienţi şi în prezenţa zgomotelor având diferite caracteristici.

4.3.2. Metode regionale de clasificare statistică.

O primă metodă a fost propusă de Yakimovski în [113] şi constă în verificarea unor ipoteze statistice la nivel regional.

Fie două regiuni adiacente, de formă oarecare. Din studiul caracteristicilor lor statistice, se ia decizia că ele aparţin unei aceleiaşi regiuni relativ uniforme sau există o muchie la graniţa dintre ele.



Reuniunea celor două regiuni are M pixeli şi este caracterizată de o medie şi o dispersie date de:

$$m = \frac{1}{M}\sum_{i=1}^{M} f(x_i,y_i) \qquad \sigma^2 = \frac{1}{M}\sum_{i=1}^{M}[f(x_i,y_i) - m]^2 \tag{41}$$

Fiecare regiune în parte (având M' şi respectiv M'' pixeli) este caracterizată de mediile:

$$m_1 = \frac{1}{M'}\sum_{i=1}^{M'} f(x_i^{(1)},y_i^{(1)}) \qquad m_2 = \frac{1}{M''}\sum_{i=1}^{M''} f(x_i^{(2)},y_i^{(2)}) \tag{42}$$

şi de dispersiile:

$$\sigma_1^2 = \frac{1}{M'}\sum_{i=1}^{M'}[f(x_i^{(1)},y_i^{(1)}) - m_1]^2 \qquad \sigma_2^2 = \frac{1}{M''}\sum_{i=1}^{M''}[f(x_i^{(2)},y_i^{(2)}) - m_2]^2 \tag{43}$$

În ipoteza unei distribuţii normale a nivelelor de gri în interiorul regiunilor, regula de decizie care se obţine este:

$$\begin{cases} \sigma_1^{M'}\sigma_2^{M''} > \sigma^M \Rightarrow \text{o singură regiune (muchie absentă)} \\ \sigma_1^{M'}\sigma_2^{M''} \le \sigma^M \Rightarrow \text{regiuni distincte (muchie prezentă)} \end{cases} \tag{44}$$

ceea ce constituie şi regula de segmentare.

4.3.3. Tehnici relaxaţionale.

Relaxarea este o metodă iterativă care permite gruparea pe baze probabilistice a unui număr de obiecte în clase (Rogers[93]). Clasificarea este îmbunătăţită în fiecare pas prin verificarea unor ipoteze de compatibilitate relative la rezultatele obţinute în pasul anterior.

Fie $(A_1,...,A_S)$ un set de obiecte care trebuiesc clasificate probabilistic într-un număr de C clase $(\omega_1,...,\omega_C)$. Obiectele pot fi muchii, linii, curbe, regiuni, etc., în cazul prelucrării de imagini.

Fiecare obiect A_j are un set de vecini $\{A_j\}$. Vom ataşa fiecărui obiect un vector $(P_{j1},...,P_{jC})$, astfel încât P_{jk} să fie probabilitatea ca obiectul A_j să aparţină clasei ω_k. Valoarea iniţială a acestor probabilităţi poate fi obţinută prin metode convenţionale (o estimaţie a modulului gradientului pentru cazul detecţiei de muchii).

Pentru fiecare pereche de obiecte (A_i,A_j) şi pentru fiecare pereche de clase (ω_m,ω_n) trebuie definită o **măsură a compatibilităţii** afirmaţiilor (ipotezelor):

$$\begin{cases} \text{obiectul } A_i \text{ inclus în clasa } \omega_m \\ \text{obiectul } A_j \text{ inclus în clasa } \omega_n \end{cases} \tag{45}$$

Această funcție de compatibilitate, notată $\boldsymbol{r}(A_i,\omega_m;A_j,\omega_n)$, ia valori în domeniul [-1,+1] şi trebuie să respecte următoarele condiții:

$$\begin{cases} \boldsymbol{r}(A_i,\omega_m;A_j,\omega_n)=-1 \Rightarrow \text{ipoteze total incompatibile} \\ \boldsymbol{r}(A_i,\omega_m;A_j,\omega_n)=0 \Rightarrow \text{ipoteze irelevante} \\ \boldsymbol{r}(A_i,\omega_m;A_j,\omega_n)=+1 \Rightarrow \text{ipoteze perfect compatibile} \end{cases} \tag{46}$$

Funcția de compatibilitate va fi folosită, alături de probabilitățile ataşate obiectelor vecine, pentru ajustarea probabilităților corespunzătoare obiectului curent.

Folosind notațiile:

P_{im} - probabilitatea ca obiectul A_i să aparțină clasei ω_m;

P_{jn} - probabilitatea ca obiectul A_j să aparțină clasei ω_n;

regulile de ajustare a probabilităților pot fi scrise explicit astfel:

1. Dacă P_{im} este **mare** şi $\boldsymbol{r}(A_i,\omega_m;A_j,\omega_n)$ este apropiat de **"+1"**, atunci P_{jn} **trebuie mărit**.

2. Dacă P_{im} este **mic** şi $\boldsymbol{r}(A_i,\omega_m;A_j,\omega_n)$ este apropiat de **"-1"**, atunci P_{jn} **trebuie micşorat**.

3. Dacă P_{im} este **mic** şi $\boldsymbol{r}(A_i,\omega_m;A_j,\omega_n)$ este apropiat de **"0"**, atunci P_{jn} **nu trebuie modificat semnificativ**.

Aceste reguli pot fi sintetizate prin ajustarea (aditivă) a probabilităților P_{jn} proporțional cu:

$$P_{im}\,\boldsymbol{r}(A_i,\omega_m;A_j,\omega_n)$$

Dar ajustarea efectivă a probabilității P_{jn} corespunzătoare unui obiect se face ținând cont de toate ipotezele posibile pentru P_{im}, adică toți vecinii obiectului curent A_j şi toate clasele la care aceşti vecini pot aparține. Deoarece se doreşte cel mai adesea o ajustare multiplicativă ajustarea se va face prin:

$$P_{jn}(t+1)=P_{jn}(t)\left[1+\sum_{i\neq j} w_i \sum_{m=1,m\neq n}^{C} P_{im}\,\boldsymbol{r}(A_i,\omega_m;A_j,\omega_n)\right] \tag{47}$$

Mai rămâne de eliminat un singur neajuns care derivă din relația $\sum_{n=1}^{C} P_{jn}=1$, care trebuie îndeplinită necondiționat. Ca urmare este necesară o renormalizare a lui P_{jn} după fiecare ajustare. În concluzie avem:

$$P_{jn}(t+1)=P_{jn}(t)\left[1+\frac{\boldsymbol{Q}_{jn}(t)}{\sum_{n=1}^{C}\left[1+\boldsymbol{Q}_{jn}(t)\right]}\right], \tag{48}$$

unde:

$$\boldsymbol{Q}_{jn}(t)=\sum_{i\neq j} w_i \sum_{m=1,m\neq n}^{C} P_{im}\,\boldsymbol{r}(A_i,\omega_m;A_j,\omega_n) \tag{49}$$

În cazul practic al detecției de muchii, obiectele $(A_1,\ldots,A_S)$ care trebuiesc clasificate sunt pixelii dintr-o imagine şi există doar două clase:

$$\begin{cases} C_1 \Leftrightarrow \text{muchie prezentă} \\ C_2 \Leftrightarrow \text{muchie absentă} \end{cases} \tag{50}$$

Pentru început, probabilităţile P_i ataşate fiecărui pixel ($P_{i2} = 1 - P_{i1} = 1 - P_i$) se setează la o valoare proporţională cu amplitudinea furnizată de un gradient clasic:

$$P_i(0) = \frac{\boldsymbol{g}(x,y)}{\underset{x,y}{\boldsymbol{max}}\, \boldsymbol{g}(x,y)} \tag{51}$$

Prin aceleaşi metode clasice se estimează de asemenea unghiul care dă direcţia muchiei în punctul (pixelul) respectiv.

Probabilităţile P_i se ajustează funcţie de probabilităţile P_j ale vecinilor şi direcţiile θ ale gradienţilor ataşaţi lor, pe baza funcţiilor de compatibilitate, care se definesc astfel:

$$\begin{cases} \boldsymbol{r}[\,\boldsymbol{g}(x,y),\text{muchie};\boldsymbol{g}(x',y'),\text{muchie}] = \boldsymbol{cos}(\alpha-\gamma)\,\boldsymbol{cos}(\beta-\gamma)/2^D \\ \boldsymbol{r}[\,\boldsymbol{g}(x,y),\text{muchie};\boldsymbol{g}(x',y'),\overline{\text{muchie}}] = \boldsymbol{min}[\,0,-\boldsymbol{cos}(\,2\alpha-2\gamma)/2^D\,] \\ \boldsymbol{r}[\,\boldsymbol{g}(x,y),\overline{\text{muchie}};\boldsymbol{g}(x',y'),\text{muchie}] = [\,1-\boldsymbol{cos}(\,2\beta-2\gamma)]/2^{D+1} \\ \boldsymbol{r}[\,\boldsymbol{g}(x,y),\overline{\text{muchie}};\boldsymbol{g}(x',y'),\overline{\text{muchie}}] = 1/2^D \end{cases} \tag{52}$$

unde:

$D = \boldsymbol{max}(|x-x'|,|y-y'|)$;

α - panta muchiei în punctul de coordonate (x,y) ;

β - panta muchiei în punctul de coordonate (x',y')

γ - panta segmentului care uneşte punctele (x,y) şi (x',y') .

Convergenţa procesului de relaxare a fost demonstrată de Zucker (doar condiţii suficiente).

O variantă mai complexă (Canny[29]) presupune existenţa a *C+2* clase, *C* clase corespunzătoare muchiilor având "*C*" orientări, plus cele două clase definite anterior.

4.4. Detecţia muchiilor tridimensionale.

Detecţia muchiilor tridimensionale se foloseşte, de exemplu, în cazul imaginilor tridimensionale tomografice. Prin "reconstrucţie 2D" se obţin imagini bidimensionale din "proiecţiile 1D" (realizate prin metode specifice tomografiei) ale corpului studiat. Prin "reconstrucţie 3D" se obţin din mai multe "proiecţii 2D" reprezentări tridimensionale ale corpului.

Măsurători asupra reprezentărilor tridimensionale ale corpurilor pot avea drept punct de plecare extragerea muchiilor tridimensionale. Pentru corpuri, evident, se lucrează cu o generalizare a muchiilor, devenite suprafeţe, iar nivelul de gri înseamnă, de obicei, densitate locală sau, în cazul tomografiei, factor de absorbţie a razei incidente.

Toate informaţiile privind variaţiile locale de densitate sunt conţinute în vectorul gradient:

$$\vec{\boldsymbol{G}}(x,y,z) = \frac{\partial\, \boldsymbol{f}(x,y,z)}{\partial x}\vec{i} + \frac{\partial\, \boldsymbol{f}(x,y,z)}{\partial y}\vec{j} + \frac{\partial\, \boldsymbol{f}(x,y,z)}{\partial z}\vec{k} \tag{53}$$

Analog cazului bidimensional, se demonstrează că modulul gradientului, fiind izotrop, poate fi folosit drept detector de muchii tridimensionale. Pentru cazul discret se construiesc aproximaţii ale modulului acestui gradient care, fără a mai fi izotrope, au avantajul unui volum redus de calcule.

4.5. Alte metode de detecţie a muchiilor.

O multitudine de noi algoritmi de detecţie a muchiilor au fost propuşi de diferiţi autori. O mare parte din aceşti algoritmi au la bază construcţia unor modele dedicate unor aplicaţii concrete.

Astfel Azzopardi în [07] utilizează o interesantă metodă de detecţie a contururilor celulelor componente ale unor muşchi scheletali, metodă bazată pe localizarea nucleelor acestor

celule, deci, din punct de vedere geometric, a centrelor lor. Pe direcţii radiale faţă de poziţiile acestor nuclee se efectuează o detecţie de muchii unidimensionale, estimarea finală a conturului celulelor făcându-se prin interpolarea rezultatelor obţinute în pasul precedent. Metoda poate fi uşor generalizată nu numai pentru alte tipuri de celule, ci şi în metalografie, pentru analiza componentelor unor aliaje.

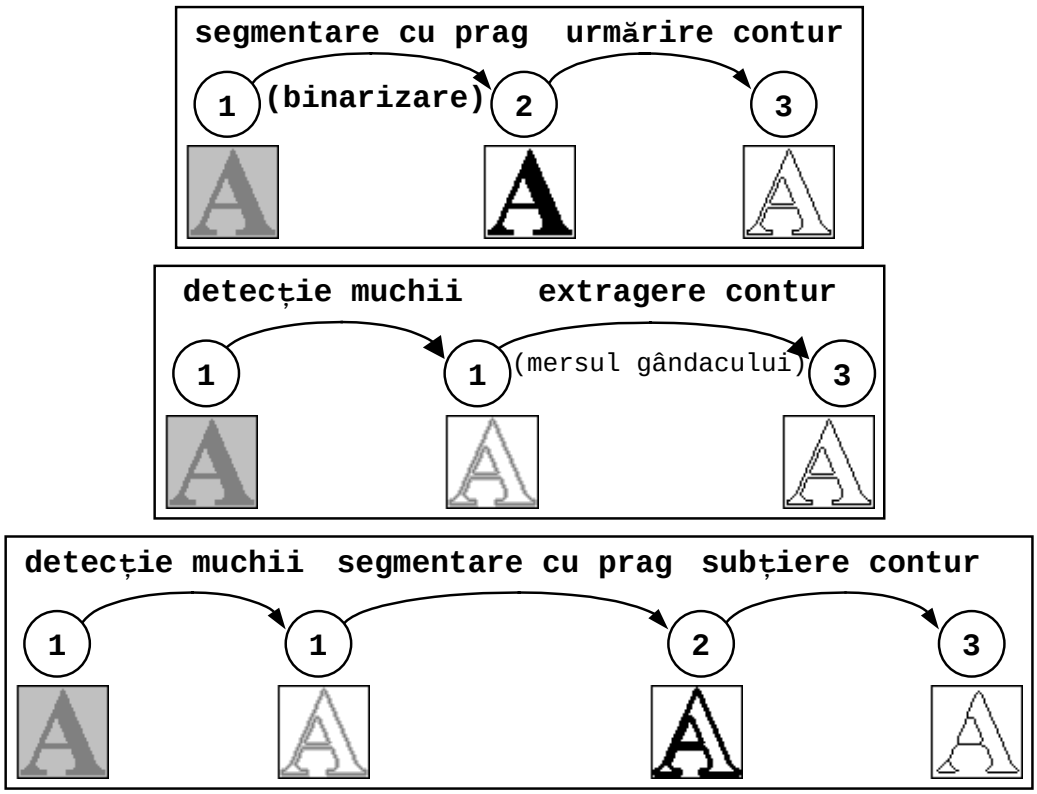
Alte metode de detecţie de muchii se referă la determinarea contururilor unor *distribuţii* de elemente de structură. Putând fi interpretată şi ca o metodă de segmentare, metoda propusă de H. Tagachi [106] generează contururile limită ale unor distribuţii (grupări, clusteri) de date, în cazul general, pe baza unor reguli adaptive.

Utilizarea unor cunoştinţe apriorice privind proprietăţile, poziţia sau orientarea muchiilor permit construirea unor detectori de muchii specializaţi, cu performanţe deosebite (Persoon[81]). Aceste cunoştinţe apriorice pot fi încorporate, de exemplu, în algoritmul de detecţie a muchiilor folosind relaxarea, rezultând un set de variante dedicate ale acestui algoritm (Canny[29]).

4.6. Algoritmi de extragere a contururilor.

Diferenţa între detecţie de muchii şi extragere de contur este, cel mai adesea, esenţială. În general, un detector de muchii furnizează la ieşire tot o imagine în scară de gri, pe când un extractor de contur furnizează la ieşire o imagine binară, conţinând linii cu grosimea de un pixel în poziţiile în care se estimează prezenţa unei muchii sau graniţa între două regiuni diferite.

Detectorii spaţiali de muchii descrişi în capitolul **(4)** realizează extragerea de contur dacă sunt urmaţi de alte operaţii care să asigure obţinerea imaginii binare conţinând linii cu grosimea de un pixel. De exemplu, s-ar putea folosi una din următoarele combinaţii de operaţii:



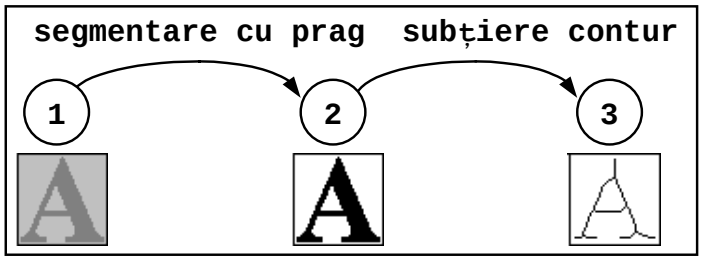


Fig. 5. Algoritmi de extragere a contururilor.

Evident, există o multitudine de alți algoritmi de extragere de muchii. Un algoritm de extragere a contururilor din imagini binare a fost deja menționat în capitolul 3., el având la bază operatorii morfologici de erodare şi/sau dilatare cu diferite şabloane.

4.6.1. Algoritmi de urmărire de contur.

Un prim algoritm ce va fi descris pe scurt în continuare (Hegron[48]) este denumit şi "mersul orbului". Plecând de la o imagine binară se urmăreşte selecția pixelilor aparținând conturului.

1. Se alege un sens de parcurgere a conturului.

2. Se localizează un prim pixel al obiectului, de obicei prin baleiaj linie cu linie sau coloană cu coloană. Acest pixel se marchează ca fiind pixelul de start. El devine pixelul curent.

3. Se baleiază vecinii pixelului curent, în sensul de parcurgere ales, plecând de la precedentul pixel curent, până la găsirea unui nou pixel obiect, care se marchează şi devine pixel curent.

4. Se repetă pasul 3 până la închiderea conturului, adică până când pixelul de start devine iar pixel curent.

5. Dacă se doreşte localizarea altor obiecte în imagine, se reia algoritmul cu pasul 2, pentru găsirea altor obiecte..

Dacă există în imagine mai multe obiecte de localizat, pixelul de start de obține căutând perechi de pixeli de forma:

 baleiaj linie cu linie, de la stânga la dreapta;

 baleiaj linie cu linie, de la dreapta la stânga;

 baleiaj coloană cu coloană, de sus în jos;

 baleiaj coloană cu coloană, de jos în sus.

unde cu gri s-au marcat pixelii obiect, iar cu alb cei de fond.

Funcționarea acestui algoritm pentru cazul a două imagini simple este ilustrată în figurile următoare:

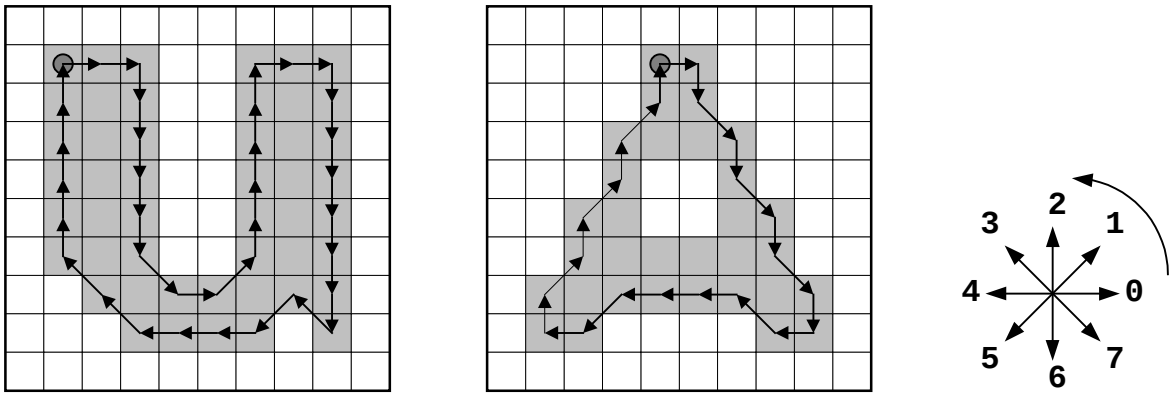


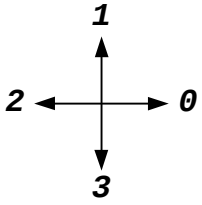
Fig. 6. Exemple de aplicare a algoritmilor de urmărire de contur.

Un alt algoritm de urmărire de contur, descris în (Pop[83]), utilizează pentru căutare o fereastră mobilă cu dimensiunile 2x2, căreia i se ataşează aşa-numitul **cod "C"**, astfel:

p	***q***
r	***s***

$$C = 8s + 4r + 2q + p$$

Se definesc direcţiile de deplasare elementară a ferestrei:



Paşii acestui algoritm sunt următorii:

1. Se alege un tip de conectivitate (contur tetra-conectat sau octo-conectat) şi un sens de parcurgere a conturului.
2. Se determină un prim punct al conturului, de obicei prin baleiaj linie cu linie sau coloană cu coloană. Acesta este marcat drept pixel de start şi se încadrează într-o fereastră pătratică 2x2.
3. Se calculează codul "C" corespunzător ferestrei curente şi se extrage din tabelele următoare, funcţie de tipul de conectivitate şi sensul de parcurs alese, direcţia de deplasare elementară a ferestrei.
4. Se repetă pasul 3 până când pixelul de start reintră în fereastră.
5. Pentru extragerea conturului altor obiecte din imagine, se reia algoritmul cu pasul 2.

Conturul tetra-conectat al obiectului conţine toţi pixelii care au intrat la un moment dat în fereastra de căutare, mai puţin pixelii marcaţi cu "o" din cazurile (tabelele) (A), (B), (C), (D).

Conturul octoconectat al obiectului conţine toţi pixelii care au intrat la un moment dat în fereastra de căutare, mai puţin pixelii marcaţi cu "X" din cazurile (7), (11), (13), (14).

Tabelul principal

Model	Cod "C"		Direcţie deplasare		Model	Cod "C"		Direcţie deplasare	
	zec	hexa	orar	trig		zec	hexa	orar	trig
	**0**	**0000**	**--**	**--**		**8**	**1000**	**0**	**3**
	**1**	**0001**	**1**	**0**		**9**	**1001**	**1**	**3**
	**2**	**0010**	**2**	**1**		**10**	**1010**	**(C)**	**(D)**
	**3**	**0011**	**2**	**0**		**11**	**1011**	**2**	**3**
	**4**	**0100**	**3**	**2**		**12**	**1100**	**0**	**2**
	**5**	**0101**	**(A)**	**(B)**		**13**	**1101**	**1**	**2**
	**6**	**0110**	**3**	**1**		**14**	**1110**	**0**	**1**
	**7**	**0111**	**3**	**0**		**15**	**1111**	**--**	**--**

Tabelul (A)

Direcţia anterioară	Noua direcţie			
	tetra-conectat		octo-conectat	
	model	direcţie	model	direcţie
0		**3**		**1**
2		**1**		**3**

Tabelul (B)

Direcţia anterioară	Noua direcţie			
	tetra-conectat		octo-conectat	
	model	direcţie	model	direcţie
1		**2**		**0**
3		**0**		**2**

Tabelul (C)

Direcţia anterioară	Noua direcţie			
	tetra-conectat		octo-conectat	
	model	direcţie	model	direcţie
1		**0**		**2**
3		**2**		**0**

În literatură sunt descrişi o multitudine de alţi algoritmi de urmărire de contur. Mulţi dintre ei se bazează pe parcurgerea secvenţială a liniilor matricii de imagine (Hegron[48]), ca în desenul următor:

Fig. 7. Parcurgerea secvențială a liniilor matricii imagine.

Problemele care trebuiesc rezolvate la implementarea unor asemenea algoritmi (care se mai folosesc şi la umplerea contururilor), țin de punctele critice ale conturului (porțiunile orizontale ale conturului, pixeli terminali, pixeli de ruptură, etc.).

4.6.2. Înlănțuirea elementelor de contur.

Cel mai adesea, contururile obținute prin detecție de muchii cu metodele descrise în capitolul 4., prezintă o serie de defecte: discontinuități, ramificații inutile de lungime mică, muchii dublate, etc. Repararea lor se face cu un set de algoritmi specializați, dintre care o parte sunt descrişi în cele ce urmează.

Metode euristice.

Roberts [91] propune o metodă euristică de înlănțuire a muchiilor (elemente de contur) care are la bază examinarea blocurilor de pixeli 4x4. În prima fază sunt reținuți drept "candidați" acei pixeli pentru care modulul gradientului depăşeşte un anumit prag. Apoi prin "potrivire" (matching) liniile cu lungimea de patru unități, având cele opt orientări de bază sunt căutate în imaginea gradient. Dacă raportul între cel mai bun scor de similitudine şi cel mai slab depăşeşte un al doilea prag, şirul de pixeli de muchie este considerat drept o linie validă.

Dacă două asemenea linii se găsesc în blocuri adiacente şi diferența direcțiilor lor este în gama $(-23^0, +23^0)$, ele se concatenează. Urmează apoi tratarea unor cazuri de tipul:

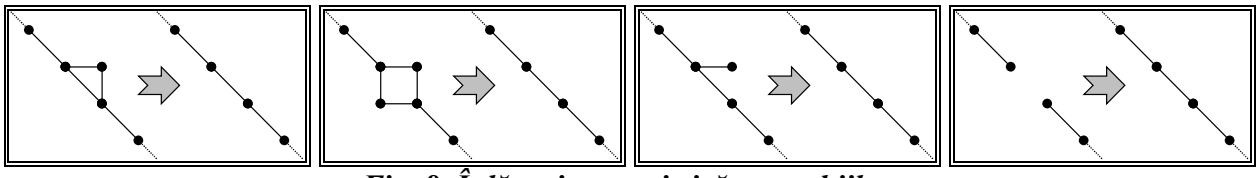
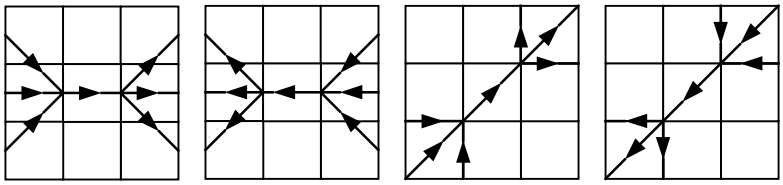


Fig. 8. Înlănțuirea euristică a muchiilor.

adică din triunghiuri se elimina laturile corespunzătoare căii mai lungi, iar dreptunghiurile se înlocuiesc cu diagonala corespunzătoare căii. Ramificațiile de lungime mică se elimină, după care urmează completarea micilor spații apărute în lungul muchiilor detectate.

Această metodă poate fi utilizată pentru o gamă largă de detectori de muchii. Nevatia propune în [75] o metodă similară pentru înlănțuirea muchiilor furnizate de detectorul Hueckel.

Muchiile furnizate de un detector tip "compass" sunt declarate valide dacă vecinii lui au o direcție conformă cu una din situațiile:



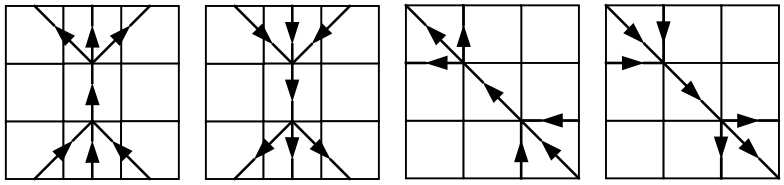


Fig. 9. Înlănțuirea euristică a muchiilor.

Metoda poate fi extinsă la vecinătăți mai mari, dar şi numărul de teste necesare se măreşte foarte mult.

Parcurgerea euristică a grafurilor.

Această metodă, datorată lui Martelli [69] pleacă de la premisa că o muchie poate fi văzută ca o cale într-un graf format de ansamblul elementelor de muchie detectate.

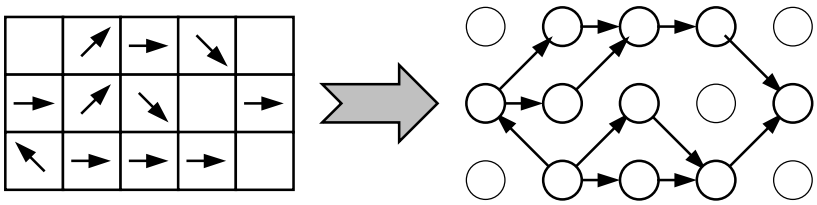


Fig. 10. Parcurgerea euristică a grafurilor.

Se asociază câte un nod fiecărui pixel care are modulul gradientului semnificativ (peste un anume prag). Între două noduri există conexiune (arc) dacă diferența orientărilor gradienților este în gama $\left(-90^0,+90^0\right)$.

O cale în graf se determină euristic, examinând succesorii fiecărui pixel şi calculând pentru fiecare cale posibilă o "*funcție de cost*". Pentru fiecare nod parcurs se alege varianta care minimizează funcția de cost. Acest algoritm nu furnizează calea optimă între două noduri ale grafului, iar viteza lui depinde de alegerea funcției de cost.

Programarea dinamică.

Folosind programarea dinamică se poate găsi calea optimă între două noduri ale unui graf orientat, adică calea care minimizează funcția de cost (sau, echivalent, maximizează o "*funcție de evaluare*"). Montanari [73] propune următoarea funcție de evaluare pentru un graf cu N nivele:

$$S(x_1,x_2,...,x_N,N)=\sum_{k=1}^{N}\left|g(x_k)\right|-\alpha\sum_{k=2}^{N}\left|\theta(x_k)-\theta(x_{k-1})\right|-\beta\sum_{k=2}^{N}d(x_k,x_{k-1}) \qquad (54)$$

unde: $x_k, k=1,..,N$ este un vector care indică nodurile de muchie de pe nivelul "k" al grafului;

$g(x_k)$ este amplitudinea gradientului pentru nodul $x_k, k=1,..,N$;

$\theta(x_k)$ este orientarea gradientului;

$d(x_k,x_{k-1})$ este distanța între nodurile x_k şi x_{k-1};

α,β sunt constante pozitive.

Utilizând *principiul optimalității al lui Bellman*, programarea dinamică furnizează o procedură care găseşte o cale în graf în condițiile minimizării funcției de evaluare dată anterior.

4.6.3. Reprezentarea contururilor.

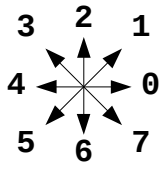
Odată determinat conturul, el poate fi reprezentat fie matriceal, ceea ce necesită însă o mare cantitate de informații, fie sub forma şirului coordonatelor pixelilor care alcătuiesc conturul, fie folosind "*codul-lanț*" (chain-code). Acesta din urmă presupune reținerea coordonatelor

pixelului de start, alături de şirul direcţiilor care indică poziţiile relative succesive ale pixelilor din contur.

Pentru obiectele din figurile anterioare conţinând imaginile caracterelor "u" şi "A", codurile lanţ corespunzătoare sunt:

U : [(x0,y0),0,0,6,6,6,6,6,7,0,1,2,2,2,2,2,0,0,6,6,6,6,6,6,6,3,5,4,4,4,3,3,2,2,2,2,2];

A : [(x0,y0),0,6,7,6,7,6,7,6,4,3,4,4,4,5,4,2,1,2,1,2,1,2].



unde direcţiile sunt date de:

Uneori se mai utilizează pentru reprezentarea contururilor "*codul-lanţ diferenţial*" (differential chain-code) care presupune păstrarea coordonatelor pixelului de start a conturului, a primei direcţii absolute şi a şirului diferenţelor relative între două direcţii succesive.

Valorile posibile pentru codul-lanţ diferenţial sunt: $(0,\pm1,\pm2,\pm3,4)$.

Avantajul acestei metode devine clar dacă se face observaţia că, cel mai adesea, între valorile posibile ale acestor diferenţe de direcţii , probabilitatea cea mai mare de apariţie o are valoarea "0":

$$P(0) \geq P(\pm1) \geq P(\pm2) \geq P(\pm3) \geq P(4) \tag{55}$$

Utilizând o codare prin coduri de lungime variabilă, se poate obţine o reducere substanţială a volumului de date necesar pentru reprezentarea conturului.

4.7. Detecţia liniilor şi a spoturilor.

Metodele cele mai simple pentru detecţia liniilor au la bază operatori spaţiali construiţi după principii analoge obţinerii măştilor de detecţie de muchii şi care se folosesc la fel, adică:

$$\tilde{\boldsymbol{f}}(x,y) = \max_{m}\left\{\left|\sum_{i=-1}^{1}\sum_{j=-1}^{1}\boldsymbol{f}(x+i,y+j)\,\boldsymbol{H}_m(i,j)\right|\right\} \tag{56}$$

unde H_m sunt măşti (matrici 3x3) pentru detecţia liniilor având diferite orientări. Se poate folosi setul de măşti:

$$\boldsymbol{H}_1=\begin{bmatrix}-1 & 2 & -1\\ -1 & 2 & -1\\ -1 & 2 & -1\end{bmatrix}\ \boldsymbol{H}_2=\begin{bmatrix}-1 & -1 & -1\\ 2 & 2 & 2\\ -1 & -1 & -1\end{bmatrix}\ \boldsymbol{H}_3=\begin{bmatrix}-1 & -1 & 2\\ -1 & 2 & -1\\ 2 & -1 & -1\end{bmatrix}\ \boldsymbol{H}_4=\begin{bmatrix}2 & -1 & -1\\ -1 & 2 & -1\\ -1 & -1 & 2\end{bmatrix} \tag{57}$$

sau măştile cu valori ponderate:

$$\boldsymbol{H}'_1=\begin{bmatrix}-1 & 2 & -1\\ -2 & 4 & -2\\ -1 & 2 & -1\end{bmatrix}\ \boldsymbol{H}'_2=\begin{bmatrix}-1 & -2 & -1\\ 2 & 4 & 2\\ -1 & -2 & -1\end{bmatrix}\ \boldsymbol{H}'_3=\begin{bmatrix}-2 & -1 & 2\\ -1 & 4 & -1\\ 2 & -1 & -2\end{bmatrix}\ \boldsymbol{H}'_4=\begin{bmatrix}2 & -1 & -2\\ -1 & 4 & -1\\ -2 & -1 & 2\end{bmatrix} \tag{58}$$

Pentru detecţia spoturilor, Prewitt [87] propune folosirea următoarei măşti de răspuns la impuls:

$$\boldsymbol{S}=\frac{1}{8}\begin{bmatrix}1 & -2 & 1\\ -2 & 4 & -2\\ 1 & -2 & 1\end{bmatrix} \tag{59}$$

Problema se mai poate aborda utilizând măşti de dimensiuni mai mari sau tehnici de "potrivire" (image/template matching).

4.8. Detalii de implementare.

Observaţiile din finalul capitolului precedent referitoare la implementarea filtrelor spaţiale de îmbunătăţire a imaginilor se aplică direct şi filtrelor spaţiale folosite pentru detecţia de muchii, linii şi spoturi.

Aplicaţia PROImage, prezentată în capitolul 21 include, printre altele, şi implementări ale câtorva filtre de detecţie a muchiilor descrise în acest capitol. Pentru informaţii suplimentare se poate studia codul sursă din fişierul "**fil.pas**".

Algoritmii de urmărire de contur sunt foarte des utilizaţi în prelucrarea de imagini. O implementare a algoritmului de urmărire de contur denumit şi "mersul orbului", este descrisă în continuare. Fie o imagine binară (matrice *MxN)* care conţine un număr de obiecte albe pe fond negru (*negru=0, alb=255*). Se baleiază pentru început imaginea, de la stânga la dreapta şi de sus în jos, în căutarea primului pixel obiect. În **xs,ys** se păstrează coordonatele pixelului de start al conturului fiecărui obiect din imagine.

```
   // caută primul obiect din imagine
   for (i=0; i<M-1; i++)
      for (j=0; j<N-1; j++)
         if (Img[j][i] == 1) {
            // pixelul de start al primului obiect
            xs = i;
            ys = j;
            break;
         }
```

În continuare, coordonatele găsite ale punctului de start al conturului sunt trecute funcţiei de urmărire a conturului unui singur obiect:

```
// direcţii relative de căutare
int nXDir[8] = {1, 1, 0,-1,-1,-1, 0, 1};
int nYDir[8] = {0, 1, 1, 1, 0,-1,-1,-1};
////////////////////////////////////
// Funcţia TrackContour() returnează
//    FALSE pentru pixeli izolaţi şi contururile interioare
//    TRUE pentru contururile exterioare
BOOL CProcessImage::TrackContour(int xs, int ys, CRect* rect)
{
   int i,j,v,n
   int dir=6;              // direcţia iniţială de cautare
   long Area = 0;          // aria inclusă în contur
   BOOL bExit = FALSE;
   rect->SetRect(M-1, N-1, 0, 0);   // setează dreptunghiul de încadrare
iniţial
   i = xs, j = ys;
   do {
      n=0;
      do {     // verifică vecinii octoconectaţi ai pixelului curent
         dir = (++dir) % 8;
         v = Img[j+nYDir[dir]][i+nXDir[dir]];
         // testul de pixel izolat
         if (++n == 8) {
            Img[j][i] = 2; // marchează pixelul izolat
            return FALSE;
         }
      } while(v == 0);
      switch (dir) {       // incrementează aria obiectului
         case 0:  Area += -j - j;      break;
         case 1:  Area += -j - j - 1;  break;
         case 2:  Area +=  0;          break;
         case 3:  Area +=  j + j + 1;  break;
         case 4:  Area +=  j + j;      break;
```

```
         case 5:  Area +=  j + j - 1;  break;
         case 6:  Area +=  0;          break;
         case 7:  Area += -j - j + 1;  break;
      }
      i += nXDir[dir];           // află noul pixel curent
      j += nYDir[dir];
      dir = (dir + 5) % 8;       // află noua direcţie de căutare
      // actualizează dreptunghiul de incadrare
      if (i < rect->left)     rect->left = i;
      if (i > rect->right)    rect->right = i;
      if (j < rect->top)      rect->top = j;
      if (j > rect->bottom)   rect->bottom = j;
      Img[j][i] = 2;             // marchează pixelul izolat
      if (i == xs && j == ys) {
         dir--;
         do {  // verifică dacă (xs, ys) este un pixel de ruptură
            dir = (++dir) % 8;
            v = Img[j + nYDir[dir]][i + nXDir[dir]];
         } while(v == 0);
         if (v == 2)
            bExit = TRUE;        // nu există alte ramificaţii
         else
            dir--;               // (xs,ys) este pixel de ruptură
      }
   } while(bExit == FALSE);
   return (Area > 0);            // Area>0 pentru contururi exterioare
}
```

Următorul pixel de start poate aparţine fie unui contur interior al aceluiaşi obiect, fie conturului exterior al unui alt obiect. El se obţine simplu, prin baleierea imaginii de la stânga la dreapta şi de sus în jos, plecând de la precedentul pixel de start:

```
// caută următorul pixel de start (alt contur)
i = xs; j = ys;
while (i < M-1) {
   while (j < N-1) {
      if (Img[j][i] == 1 && Img[j-1][i] == 0) {
         xs = i;
         ys = j;
         break;
      }
      j++;
   }
   j = 1; i++;
}
```

Odată cu operaţia de parcurgere a contururilor pot fi efectuate şi unele măsurători simple asupra imaginii, cum ar fi calculul ariei şi al perimetrului.

5. Prelucrări morfologice ale imaginilor.

5.1. Operaţiile cu şabloane fac parte din categoria operatorilor morfologici şi permit transformări ale imaginilor binare bazate pe relaţiile geometrice de conectivitate ale pixelilor din imagine. şabloanele sunt vecinătăţi ataşate pixelului curent. Ele pot avea una din următoarele forme:

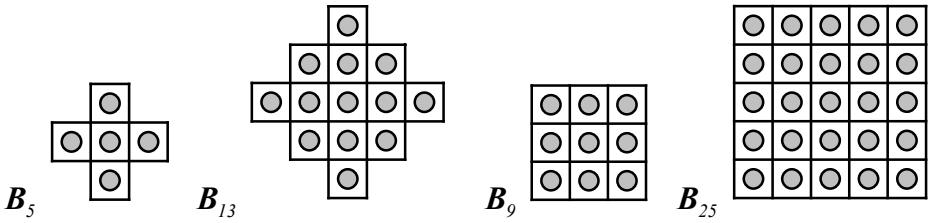


Fig. 1. Exemple de şabloane.

Principalele operaţii cu şabloane sunt **erodarea**, **dilatarea**, **deschiderea** şi **închiderea**.

Erodarea (contractarea, comprimarea) unei imagini se defineşte ca fiind mulţimea pixelilor aparţinând obiectului S care prezintă proprietatea că şablonul ataşat lor este inclus în obiectul S:

$$S \Theta B_1 = \{(x,y) \in S / B_1(x,y) \subset S\}; \quad (1)$$

Dilatarea (expandarea) unei imagini este formată din mulţimea pixelilor obiectului S la care se adaugă acei pixeli din fond al căror şablon intersectează obiectul:

$$S \oplus B = S \cup \{(x,y\} \in \overline{S} / B(x,y) \cap S \neq \varnothing\}; \quad (2)$$

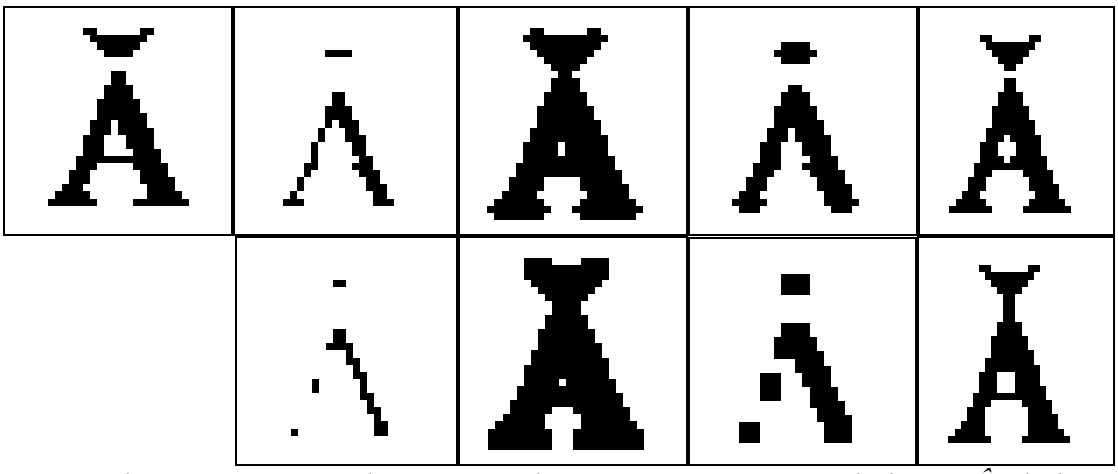
Deschiderea unei imagini este o operaţie de erodare urmată de o dilatare cu acelaşi şablon:

$$S \triangleleft B = (S \Theta B) \oplus B; \quad (3)$$

Închiderea unei imagini este obţinută printr-o dilatare urmată de o erodare folosind acelaşi şablon:

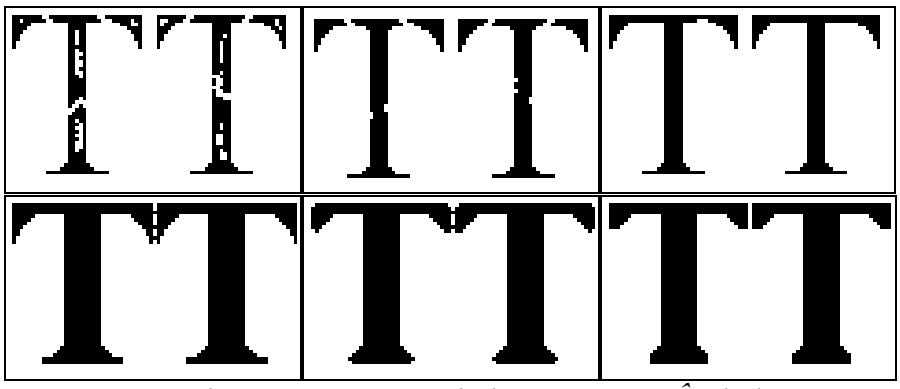
$$S \triangleright B = (S \oplus B) \Theta B \quad (4)$$

Figurile următoare prezintă rezultatele aplicării unor astfel de operatori asupra unei imagini binare, folosind şabloanele $\boldsymbol{B}_5$ şi $\boldsymbol{B}_9$.



Original *Erodare* *Dilatare* *Deschidere* *Închidere*

Operatorii închidere şi deschidere se aplică la concatenarea obiectelor fragmentate şi respectiv separarea obiectelor atinse. Fără a fi un panaceu, aceşti operatori se dovedesc a fi utili în multe situaţii.



Original *Deschidere* *Închidere*

5.2. Algoritmi de subţiere.

Subţierea furnizează la ieşire un graf care este rezultatul unor erodări succesive ale imaginii, erodări făcute în condiţia nemodificării topologiei imaginii.

Algoritmii de subţiere au la bază un set de reguli:

1. Se elimină doar pixeli aflaţi pe conturul obiectelor.
2. Pixelii terminali (care au un singur vecin octoconectat) nu se elimină.
3. Pixelii izolaţi nu se elimină (pentru că s-ar modifica topologia imaginii).
4. Pixelii de **ruptură** nu se elimină.

Uneori, regula 3 nu se aplică pentru iteraţiile iniţiale în scopul eliminării pixelilor izolaţi a căror apariţie este datorată zgomotelor.

Pixelii de ruptură sunt acei pixeli (diferiţi de pixelii izolaţi şi de cei terminali) a căror eliminare modifică topologia imaginii. Există şase cazuri de pixeli de ruptură, care rezultă din figurile următoare:

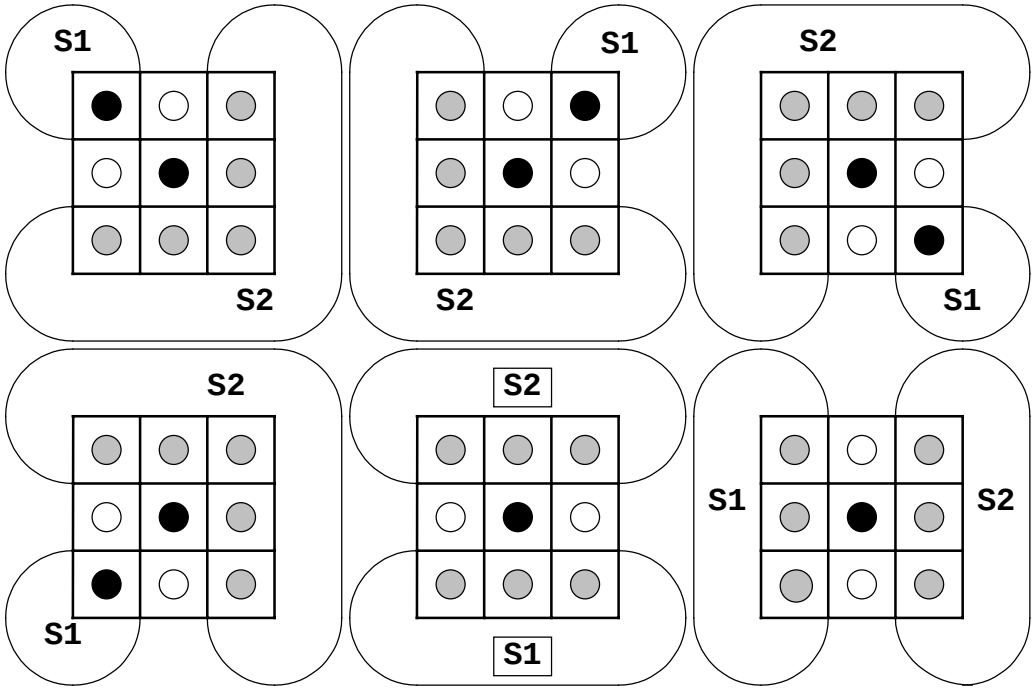


Fig. 2. Pixeli de ruptură.

unde: ● - pixel obiect (negru);
○ - pixel de fond (alb);
◍ - cel puțin un pixel din submulțimea marcată este pixel obiect (negru).

Se remarcă faptul că vecinii octoconectați ai unui pixel de ruptură formează cel puțin două submulțimi distincte.

Notând $\{v_0, v_1, ..., v_7\}$ vecinii pixelului curent, situațiile de mai sus se pot sintetiza matematic (Davies[33]), (Pop[83]) prin **numărul de treceri** dat de:

$$T(x,y) = \sum_{k=1}^{4} b_k \tag{5}$$

unde:

$$b_k = \begin{cases} 1, \text{pentru } \{v_{2k-1} \in \overline{\boldsymbol{S}}\} \bigcap \{v_{2k} \in \boldsymbol{S} \bigcup v_{2k+1} \in \boldsymbol{S}\} \\ 0, \text{în rest} \end{cases} \tag{6}$$

Există o multitudine de algoritmi de subțiere. Principala dificultate care apare la operația de subțiere este datorată situației următoare:

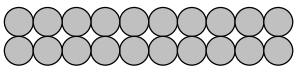
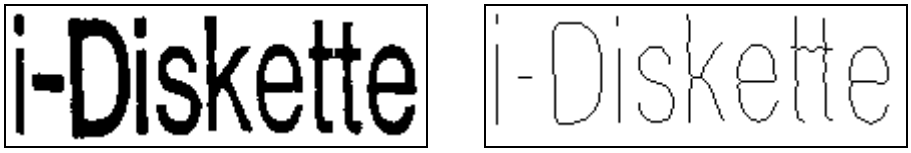


Fig. 3. Caz de dubiu la operația de subțiere.

Conform regulilor anterioare, toți pixelii care aparţin unor linii cu grosimea de doi pixeli vor fi şterşi. Se adoptă diferite soluții.

Astfel **algoritmul clasic** (Fainhurst[37]), (Pavlidis[79]) operează (verifică condițiile de ştergere şi apoi, eventual, şterge) mai întâi asupra tuturor pixelilor care au vecinul dinspre "nord" aparținând fondului, apoi pentru cei cu vecinul dinspre "vest" aparținând fondului, apoi "sud", apoi "est".



Exemplu de subțiere a unei imagini binare.

Algoritmii asincroni de subțiere încearcă să utilizeze paralelismul evident existent la operațiile de erodare. Principial, s-ar putea utiliza procesoare matriceale care să respecte următoarele restricții:

- să poată modifica doar pixelul aferent;
- să poată "citi" valorile pixelilor din vecinătatea pixelului aferent.

Se utilizează tehnica "marcării" pixelilor, ceea ce presupune lucrul pe imagini cu numărul de nivele (de gri) mai mare ca doi. Abia după parcurgerea întregii imagini, pixelii marcați vor putea fi şterşi. Problema liniilor cu grosimea de doi pixeli se păstrează şi pentru soluționarea ei se utilizează tot tehnica marcării pixelilor.

Algoritmii rapizi de subțiere încearcă să micşoreze numărul de paşi ceruți de algoritmii precedenți prin analiza grosimii locale (de-a lungul câtorva direcții) a obiectelor din imagine pentru găsirea liniei mediane folosind criterii geometrice şi nu prin erodări succesive. Parcurgerea obiectelor se face folosind teoria grafurilor.

5.3. Netezirea imaginilor binare.

Acest subiect, abordat şi în paragraful 2.4.2., deci în cadrul algoritmilor de îmbunătățire de imagini, se referă la schimbarea valorii unor pixeli dintr-o imagine binară, astfel încât să se îmbunătățească aspectul de "netezime" al contururilor obiectelor, în condițiile păstrării topologiei imaginii.

5.4. Operatori morfologici pentru imagini în scară de gri.

Operatorii morfologici studiați pentru imagini binare pot fi generalizați, în anumite condiții, pentru imagini în scară de gri. Astfel, pentru imagini în scară de gri conținând obiecte relativ distincte față de fond, la care obiectele şi fondul sunt relativ uniforme din punctul de vedere al conținutului de nivele de gri, se pretează la asemenea prelucrări morfologice care au drept rezultat îmbunătățirea calității rezultatului unei segmentări ulterioare.

Alegând B_K, unul din şabloanele definite anterior, operația de **dilatare** (expandare), pentru imagini cu nivele de gri se defineşte prin:

$$f^{(D)}(x,y) = \max_{i,j \in B_K} [\, f(x+i, y+j)] \tag{7}$$

iar cea de **comprimare** (compactare, erodare) este dată de:

$$f^{(C)}(x,y) = \min_{i,j \in B_K} [\, f(x+i, y+j)] \tag{8}$$

Operațiile de **închidere** şi respectiv **deschidere** a unei imagini în scară de gri se definesc la fel ca în cazul imaginilor binare.

5.5. Detalii de implementare.

Aplicația PROImage, prezentată în capitolul 21 include, printre altele, şi implementările a doi algoritmi de subțiere de contur, precum şi operațiile de erodare şi dilatare asupra imaginilor binare. (pentru informații suplimentare se poate studia codul sursă din fişierul "**thi.pas**").

O implementare PASCAL a unui algoritm de subțiere de contur este prezentată în continuare.

```
procedure Thinning();
type
   Conv = array[0..7] of shortint;
const
   xC : Conv = (1,1,0,-1,-1,-1,0,1);   // direcţii de căutare
   yC : Conv = (0,-1,-1,-1,0,1,1,1);
var
   i,j,x,y,r,s,t  : byte;
   k1,k2,k3       : byte;
begin
   r:=0;
   while r=0 do   // repetă până când nu mai există pixeli de eliminat
   begin
      r:=1;
      for j:=0 to 3 do  // repetă pentru cele 4 direcţii principale
      begin
         for x:=1 to M-2 do
            for y:= 1 to N-2 do  // baleiază întreaga imagine
            begin
               if (Img^[y,x]=255) and (Img^[y+yC[2*j],x+xC[2*j]]=0) then
               begin
                  t:=0;
                  for i:=1 to 4 do
                  begin             // calculează 'numărul de treceri'
                     k1:=(2*i-2) mod 8;   k1:=Img^[y+yC[k1],x+xC[k1]];
                     k2:=(2*i-1) mod 8;   k2:=Img^[y+yC[k2],x+xC[k2]];
                     k3:=(2*i) mod 8;     k3:=Img^[y+yC[k3],x+xC[k3]];
                     if (k1=0) and ((k2<>0) or (k3<>0)) then
                        Inc(t);
                  end;
                  if t=1 then             // poate fi pixel terminal
                  begin
                     s:=0;
                     for i:=0 to 7 do
                        if Img^[y+yC[i],x+xC[i]]<>0 then
                           Inc(s);        // calculează numărul de vecini
                     if s>1 then
                        s:=0;             // nu este pixel terminal
                  end
                  else
                     s:=1;                // pixel de ruptură
                  if s=1 then
                     Img^[y,x]:=254;      // marchează pixel de ruptură
                  else
                  begin
                     r:=0;                // s-au gasit pixeli de şters
                     Img^[y,x]:=100;      // marchează pixel pentru ştergere
                  end;
               end;
            end;
         for x:= 1 to M-2 do
            for y:=1 to N-2 do
               if Img^[y,x]=100 then
                  Img^[y,x]:=0;           // şterge pixelii marcaţi
      end;
   end;
end;
```

Numărul de paşi ai algoritmului depinde de grosimea obiectului a cărui subțiere se doreşte. Pixelii aflați pe conturul obiectului sunt clasificați în trei categorii: pixeli terminali, pixeli de ruptură şi pixeli regulari, care pot fi şterşi.

6. Segmentarea imaginilor.

6.1. Segmentarea cu prag.

6.1.1. Determinarea automată a pragului.

Segmentarea cu prag poate fi realizată prin stabilirea interactivă sau automată a pragului. Stabilirea automată a pragului are la bază minimizarea probabilistică a erorii de clasificare a pixelilor în pixeli de fond şi pixeli obiect.

Notând:

$P_1 =$ Probabilitatea apriori ca un pixel să fie pixel obiect;

$P_2 =$ Probabilitatea apriori ca un pixel să fie pixel de fond;

$\boldsymbol{p}_1(z) =$ densitatea de probabilitate a variabilei aleatoare ataşată pixelilor obiect;

$\boldsymbol{p}_2(z) =$ densitatea de probabilitate a variabilei aleatoare ataşată pixelilor obiect.

Presupunând o distribuție normală pentru pixelii de fond şi cei de obiect, avem:

$$\boldsymbol{p}_1(z) = \frac{1}{\delta\sqrt{2\pi}}\boldsymbol{exp}\frac{-(z-m_1)^2}{\delta^2} \tag{1}$$

$$\boldsymbol{p}_2(z) = \frac{1}{\delta\sqrt{2\pi}}\boldsymbol{exp}\frac{-(z-m_2)^2}{\delta^2} \tag{2}$$

Variabilele m_1, m_2, δ sunt mediile şi respectiv dispersia celor două variabile aleatoare, în ipoteza că, deoarece zgomotul afectează în mod egal toți pixelii din imagine, dispersiile corespunzătoare fondului şi obiectelor sunt egale.

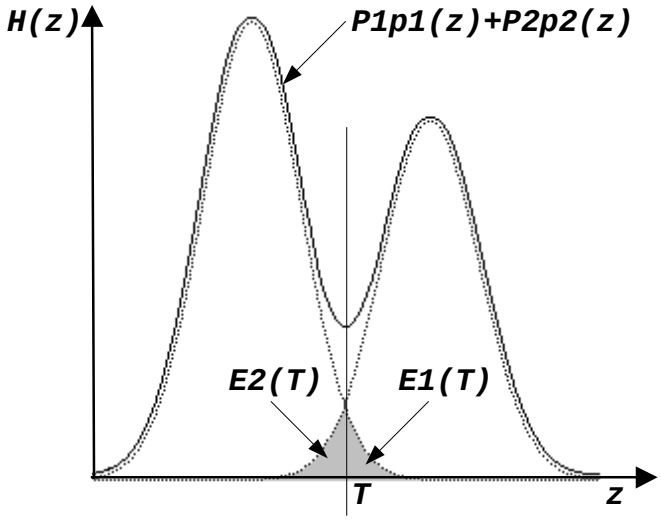


Fig. 1. Principiul determinării automate a pragului de segmentare.

În figura anterioară:

$$\boldsymbol{E}_1(T) = \int_{-\infty}^{T} P_2\, \boldsymbol{p}_2(z)\, dz \tag{3}$$

$$\boldsymbol{E}_2(T) = \int_{T}^{+\infty} P_1\, \boldsymbol{p}_1(z)\, dz \tag{4}$$

sunt respectiv eroarea de probabilitate ca un pixel fond să fie clasificat drept pixel obiect (negru), şi eroarea de probabilitate ca un pixel obiect să fie clasificat drept pixel fond (alb) prin segmentare.

Eroarea totală de clasificare a pixelilor prin segmentare este:

Minimizarea acestei erori în raport cu T duce la:

$$\frac{d\,\boldsymbol{E}}{dT} = 0 \quad \Rightarrow \quad T = \frac{m_1 + m_2}{2} + \frac{\delta^2}{m_2 - m_1}\,\boldsymbol{ln}\frac{P_2}{P_1} \tag{5}$$

Determinarea valorilor (m_1, m_2, δ, P_1 / P_2) necesare în calculul pragului T se poate face grafic, prin măsuratori pe histograma reală, folosind elementele din figura următoare:

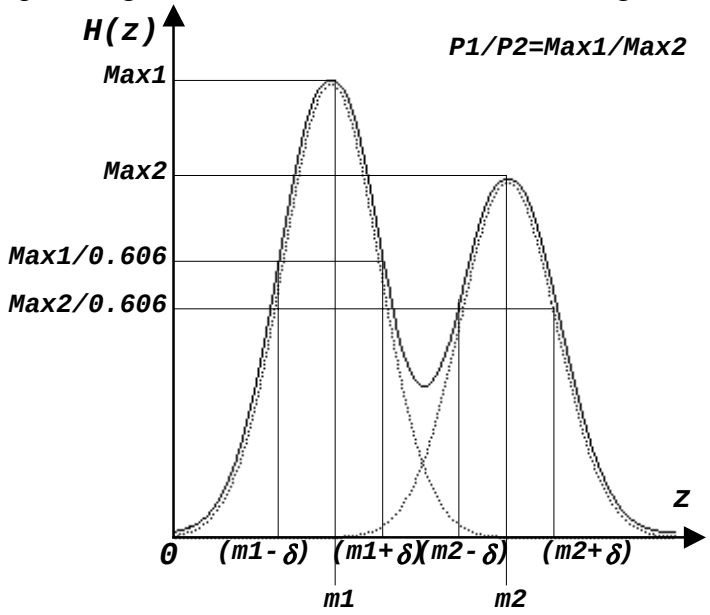


Fig. 2. Determinarea grafică a pragului de segmentare.

Dacă însă pagina scanată conține text cu o densitate mică a caracterelor sau fonturi foarte subțiri, raportul P_1 / P_2 devine foarte mic, lobul histogramei corespunzător caracterelor se diminuează foarte mult şi metoda precedentă poate da rezultate nesatisfăcătoare.

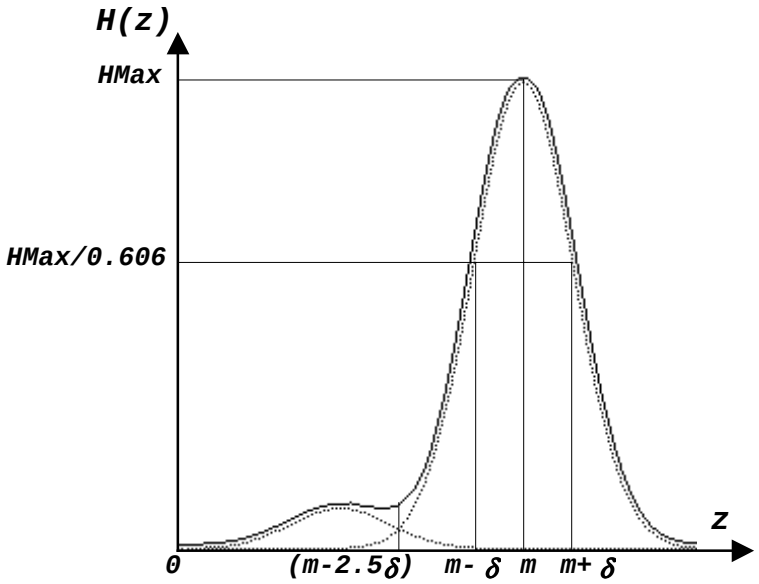


Fig. 3. Determinarea grafică a pragului de segmentare.

Criteriul alegerii pragului T la 2.5δ față de nivelul de gri pentru care se atinge valoarea maximă în histogramă poate rezolva această situație mulțumitor. Rezultate oarecum similare se obțin fixând pragul T astfel încât proporția de pixeli negri şi albi să fie cât mai apropiată de o valoare aprioric stabilită (în jur de 90%).

6.1.2. Segmentarea cu prag variabil.

Alegerea unui prag unic pentru întreaga imagine, de multe ori, nu este soluția optimă. Datorită iluminării neuniforme, diferențelor de contrast între diferite zone ale paginii, nuanțelor diferite folosite pentru pentru unele obiecte (caractere, etc.) un prag unic poate furniza rezultate nesatisfăcătoare. Soluția furnizată de Pizer [82] pentru această situație este divizarea imaginii în careuri de dimensiune 32x32 sau 64x64, suprapuse sau nu, şi determinarea pragului de segmentare t_C doar pentru careurile având histogramă bimodală, folosind una din metodele anterioare. Rezultă câte o valoare pentru fiecare careu, valoare care este folosită pentru construcția unei funcții de prag continue, $\boldsymbol{T}(x,y)$, astfel încât:

$$\boldsymbol{T}(x_C, y_C) = t_C \tag{6}$$

unde (x_C, y_C) este centrul careului "*c*".

În continuare, se execută segmentarea după formula clasică, adică:

$$\boldsymbol{f}(x,y) = \begin{cases} negru, \text{ pentru } \boldsymbol{f}(x,y) < \boldsymbol{t}(x,t) \\ alb, \text{ pentru } \boldsymbol{f}(x,y) \geq \boldsymbol{t}(x,y) \end{cases} \tag{7}$$

6.1.3. Alte strategii de segmentare cu prag.

a) În loc de a minimiza eroarea de clasificare la determinarea pragului, se poate selecta acel prag de segmentare pentru care se minimizează numărul de adiacențe între pixelii aflați sub şi deasupra pragului (Rosenfeld[95]). Evident, soluțiile care uniformizează imaginea se elimină.

b) Dacă se cunoaşte aprioric raportul între numărul de pixeli albi şi negri din imaginea binară, acesta poate fi ales drept criteriu de determinare a pragului de segmentare (Rosenfeld[95]):

$$K = \int_{-\infty}^{T} \boldsymbol{H}(z)\,dz \Big/ \int_{T}^{+\infty} \boldsymbol{H}(z)\,dz \text{ (cazul continuu) } K = \sum_{z=0}^{T-1} \boldsymbol{H}(z) \Big/ \sum_{z=T}^{L-1} \boldsymbol{H}(z) \text{ (cazul discret)} \quad (8)$$

c) Dacă anumite dimensiuni / distanţe care caracterizează obiectele din imagine sunt cunoscute, se poate alege acel prag care fixează respectiva dimensiune la valoarea dorită (Rosenfeld[95]).

6.2. Metode regionale de segmentare. Creşterea regiunilor.

Ideea de bază este următoarea: "pixelii vecini având amplitudini apropiate vor fi consideraţi ca aparţinând aceleiaşi regiuni, deci vor fi grupaţi împreună".

Brice şi Fenema [13] au folosit un algoritm de segmentare prin creşterea regiunilor bazat pe reguli stabilite euristic.

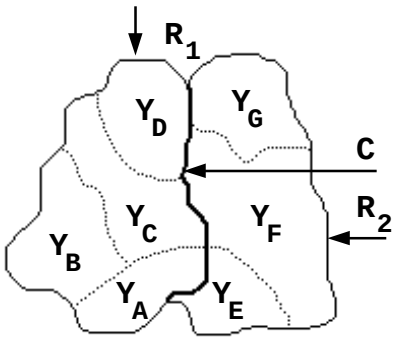


Fig. 4.Principiul creşterii regiunilor.

Se notează cu R_1 şi respectiv R_2 cele două regiuni candidate la concatenare, P_1 şi respectiv P_2 perimetrele lor, iar C frontiera comună a celor două regiuni. Fie D porţiunea din C pentru care amplitudinea diferenţei între nivelele de gri de o parte şi de alta a acestei frontiere este mai mică decât un prag ε_1.

1. În prima fază perechi de pixeli având nivele de gri apropiate sunt grupate împreună pentru a forma aşa-numitele regiuni atomice.

2. Regiunile R_1 şi R_2 se concatenează dacă:

$$\frac{D}{\boldsymbol{min}(P1,P2)} > \varepsilon_2, \quad (9)$$

ceea ce previne concatenarea regiunilor cu aproximativ aceeaşi dimensiune, dar permite înglobarea regiunilor mai mici în cele mai mari.

3. Regiunile R_1 şi R_2 se concatenează dacă: $\frac{D}{C} > \varepsilon_3$, cu $\varepsilon_3 \approx 0.75$, regulă care tratează cazul porţiunilor de frontieră rămase după aplicarea primei reguli. Aplicarea doar a acestei reguli tinde să "supra-concateneze" regiunile din imagine.

6.3. Metode de segmentare/aproximare a curbelor.

6.3.1. Aproximarea prin segmente de dreaptă.

Interpolarea curbelor prin segmente de dreaptă este cea mai simplă metodă de segmentare a curbelor. Ea se poate face (Jain[56]) prin aproximări succesive, plecând de la segmentul de dreaptă care uneşte capetele curbei ce trebuie aproximate şi care constituie practic o primă

aproximare a curbei. Se caută punctul de pe curbă cel mai depărtat de acest segment şi se obţine următoarea aproximare a curbei înlocuind segmentul de dreaptă iniţial cu cele două segmente ce unesc capetele segmentului de curbă cu punctul găsit. Operaţia descrisă anterior se repetă succesiv pentru fiecare segment de curbă obţinut, până la atingerea preciziei dorite.

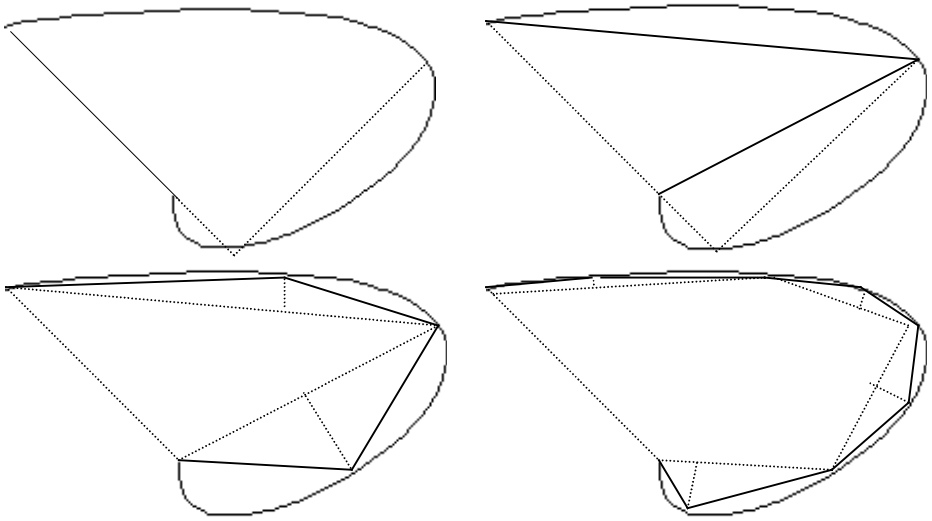


Fig. 5. Aproximarea prin segmente de dreaptă.

6.3.2. Aproximarea polinomială.

Reprezentarea polinomială, foarte des folosită, furnizează ecuaţia unui polinom care aproximează curba dată. Expresia polinomului de gradul $(n-1)$ care trece prin "*n*" puncte date, este:

$$\boldsymbol{P}_n(x) = \sum_{i=1}^{n} y_i \prod_{j \neq i} \frac{(x - x_j)}{(x_i - x_j)} \tag{10}$$

Când numărul de puncte este mare, gradul polinomului creşte şi el, ca urmare pot apare oscilaţii ale curbei aproximante. Eliminarea acestui dezavantaj se face folosind aproximările pe porţiuni, adică pe segmente de curbă. Pentru a obţine curbe netede, se impun condiţii suplimentare care se referă la continuitatea curbei şi a derivatelor sale în punctele de joncţiune. Astfel se obţin curbele spline şi curbele Bezier.

6.3.3. Aproximarea prin funcţii spline.

Funcţiile spline (Rogers[93]) sunt funcţii polinomiale pe subintervale adiacente, care se racordează împreună cu un anumit număr din derivatele sale. Fie *[x(t),y(t)]* un segment de curbă descris parametric, unde $t_1 \le t \le t_2$. El poate fi reprezentat prin funcţii spline *cubice*:

$$\boldsymbol{S}(t) = \sum_{i=1}^{4} B_i t^{i-1}, \text{unde } t_1 \le t \le t_2 \tag{11}$$

Constantele B_1, B_2, B_3, B_4 se determină specificând patru condiţii, care pot fi *poziţiile* capetelor segmentului de curbă şi orientările curbei în aceste puncte (date prin vectori tangenţi la curbă în aceleaşi puncte). Rezultă o relaţie de forma:

$$\boldsymbol{S}(t) = \boldsymbol{S}(t_1) + \boldsymbol{S}'(t_1)t + \left\{ \frac{3[\boldsymbol{S}(t_2) - \boldsymbol{S}(t_1)]}{t_2^2} - \frac{2\boldsymbol{S}'(t_1) + \boldsymbol{S}'(t_2)}{t_2} \right\} t^2 + \left\{ \frac{2[\boldsymbol{S}(t_2) - \boldsymbol{S}(t_1)]}{t_2^3} + \frac{\boldsymbol{S}'(t_1) + \boldsymbol{S}'(t_2)}{t_2^2} \right\} t^3 \tag{12}$$

Metoda anterioară se poate extinde pentru cazul a mai multe segmente de curbă. Astfel, pentru cazul a două segmente de curbă trebuiesc determinate opt constante şi aceasta se face specificând poziţiile de început şi sfârşit ale capetelor celor două segmente (patru condiţii),

orientările curbei pentru începutul primului segment şi sfârşitul celui de-al doilea (două condiții), la care se adaugă condițiile de egalitate a orientării şi curburii în punctul de joncțiune a celor două segmente.

Acest mod de operare se poate extinde uşor pentru un număr mare de segmente de curbă.

Se poate remarca influența amplitudinii vectorilor $\boldsymbol{S}'(t_1)$ şi $\boldsymbol{S}'(t_2)$ precum şi a gamei de valori a parametrilor "*t*" pentru fiecare segment asupra alurii curbei obținute. Se utilizează des alegerea gamei de valori a parametrului "*t*" *prin metoda coardei*. *Curbele spline cubice normalizate* se referă la situația în care, pentru fiecare segment de curbă se setează $0 \le t \le 1$.

În unele situații se preferă pentru segmentele terminale utilizarea *condiției de curbură nulă,* adică $\boldsymbol{S}''(t_k) = 0$, iar pentru cazul curbelor închise condițiile din punctele de joncțiune ale segmentelor sunt de acelaşi tip pentru toate segmentele.

O altă clasă de funcții spline, utilizate mai ales pentru simplitatea lor, sunt *funcțiile spline parabolice* (de ordinul doi, cuadratice), care sunt descrise de relații de forma:

$$\boldsymbol{S}(t) = (1-t)\,\boldsymbol{p}(r) + t\,\boldsymbol{q}(s) \tag{13}$$

unde t, r, s sunt parametri, iar $\boldsymbol{p}(r)$ şi $\boldsymbol{q}(s)$ sunt parabole descrise parametric. Specificarea parabolelor se face analog, prin condiții impuse punctelor de joncțiune între segmentele de curbă.

6.3.4. Aproximarea prin funcții Bezier.

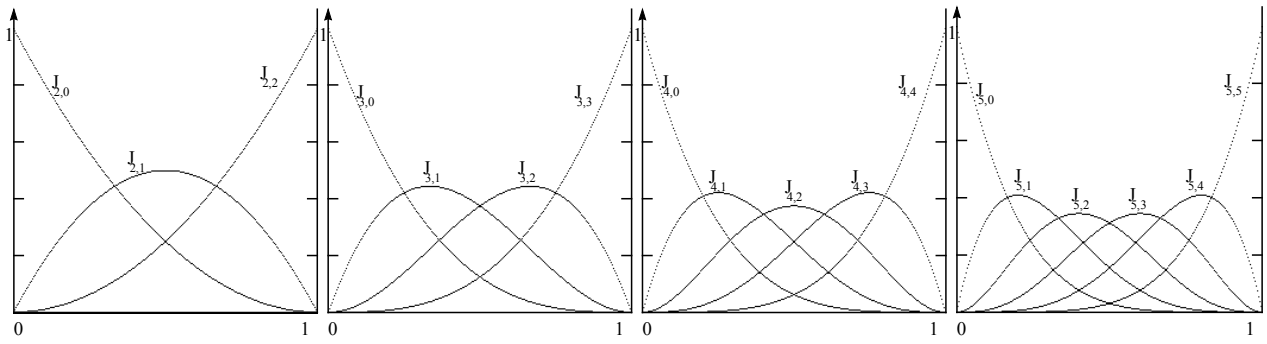
Curbele Bezier (Rogers[93]) realizează operația de *aproximare* a segmentelor de curbă. Curbele descrise anterior sunt astfel construite încât sunt constrânse să treacă prin punctele de joncțiune ale segmentelor de curbă. Ele sunt determinate definind mai întâi un poligon cu colțurile B_0, B_1, B_2, B_3. Curba Bezier parametrică asociată se scrie sub forma:

$$\boldsymbol{P}(t) = \sum_{i=0}^{n} B_i\, \boldsymbol{J}_{n,i}(t), \quad 0 \le t \le 1 \tag{14}$$

unde $\boldsymbol{J}_{n,i}(t)$ este funcția Bernstein de bază de ordinul "*n*", dată de:

$$\boldsymbol{J}_{n,i}(t) = \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i} \tag{15}$$

Pentru $n = 2,..,9$ alura acestor funcții este următoarea:



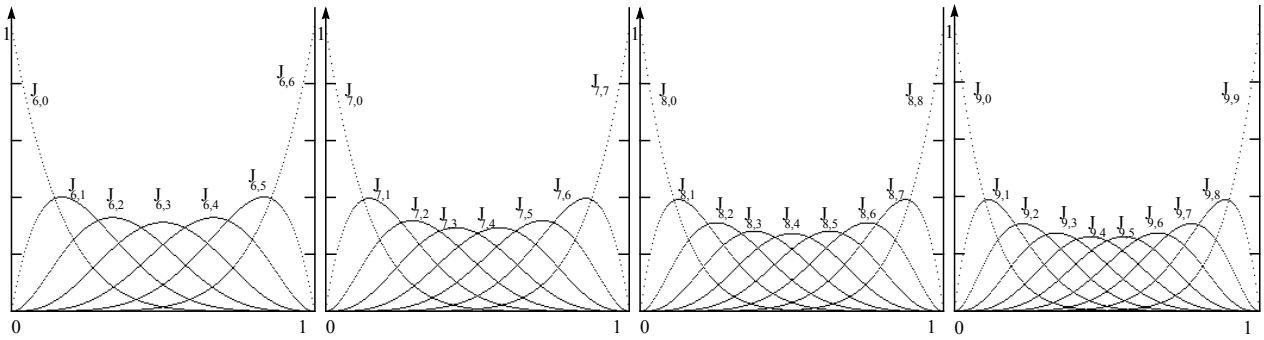
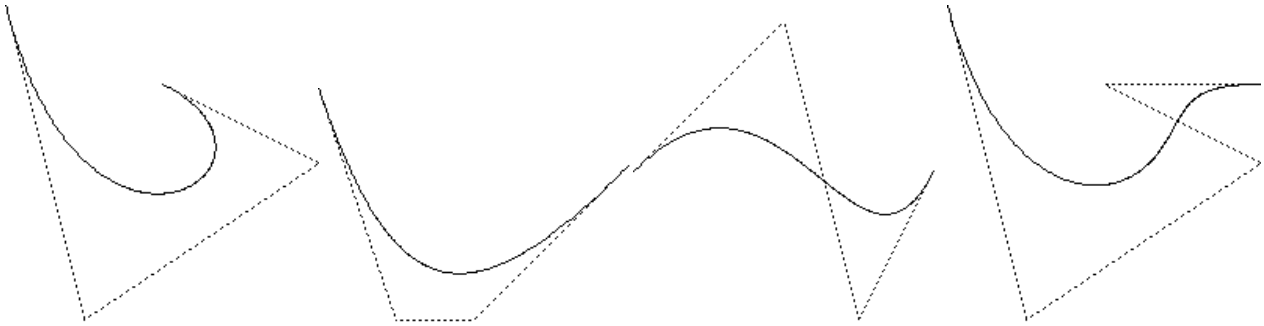


Fig. 6. Funcţiile Bernstein de ordinul 2...9.

Foarte des expresiile anterioare se pun sub formă matriceală. Pentru mărirea flexibilităţii metodei se poate mări numărul de puncte de definiţie a poligonului. Exemple pentru $n = 3$ şi $n = 4$ sunt date în continuare.



n=3 ***n=3*** ***n=3*** ***n=4***

6.3.5. Aproximarea prin funcţii B-spline.

Un alt caz de aproximare a curbelor depăşeşte limitările constatate la curbele Bezier (dependenţa directă a ordinului curbei de numărul de colţuri ale poligonului de definiţie şi faptul că $\boldsymbol{J}_{n,i}(t)$ este nenul pentru $t < 0$ şi $t > 1$). Schoenberg [100] sugerează utilizarea unei noi baze de funcţii care include funcţiile de bază ale lui Bernstein drept caz particular. De remarcat este proprietatea curbelor B-spline de a fi incluse în anvelopa convexă generată de poligonul de definiţie.

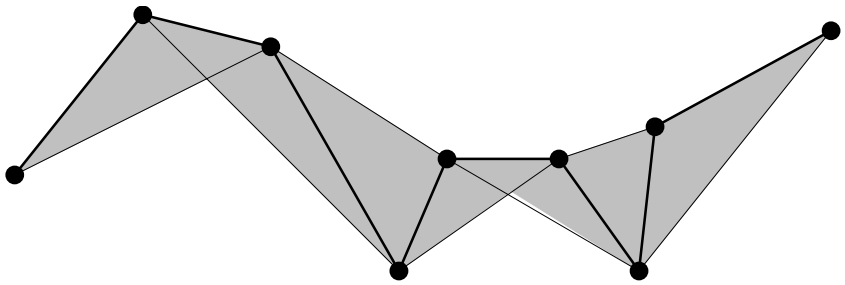


Fig. 7. Anvelopa convexă generată de poligonul de definiţie a curbelor B-spline.

Fie $[x(t),y(t)]$ un segment de curbă descris parametric, unde $t_1 \le t \le t_2$. Reprezentarea lui prin funcţii B-spline (Rogers[93]) este dată de:

$$\boldsymbol{c}(t) = \sum_{i=0}^{n} \boldsymbol{p}_i \, \boldsymbol{B}_{i,k}(t) \tag{16}$$

unde:

$\boldsymbol{c}(t) = \begin{bmatrix} \boldsymbol{x}(t) \\ \boldsymbol{y}(t) \end{bmatrix}$ este vectorul cu coordonatele curbei,

$\boldsymbol{p}_i(t) = \begin{bmatrix} \boldsymbol{p}_i^{(x)}(t) \\ \boldsymbol{p}_i^{(y)}(t) \end{bmatrix}$ sunt aşa-numitele *puncte de control*, iar

$\boldsymbol{B}_{i,k}(t), i = 0,1,...,n,\ k = 1,2,...$ sunt funcţiile B-spline generalizate de ordinul "*k*".

Parametrul "*k*" indică gradul de continuitate al funcţiei spline folosite, adică $\boldsymbol{B}_{i,k}(t)$, împreună cu primele sale $k-2$ derivate sunt continue şi în punctele de joncţiune.

Există o relaţie de recurenţă care furnizează expresiile funcţiilor B-spline de ordin superior funcţie de cele de ordin mai mic (formulele Cox-deBoor):

$$\boldsymbol{B}_{i,1}(t) = \begin{cases} 1, & t_i \le t \le t_{i+1} \\ 0, & \text{în rest} \end{cases} \tag{17}$$

$$\boldsymbol{B}_{i,k}(t) = \frac{(t - t_i)\,\boldsymbol{B}_{i,k-1}(t)}{(t_{i+k-1} - t_i)} + \frac{(t_{i+k} - t)\,\boldsymbol{B}_{i+1,k-1}(t)}{(t_{i+k} - t_{i+1})} \tag{18}$$

unde parametrii $t_i, i = 0,1,...,n$, sunt denumiţi noduri ("knots"). Funcţiile B-spline normalizate de ordinul 1,2,3,4 sunt reprezentate schematic în cele ce urmează:

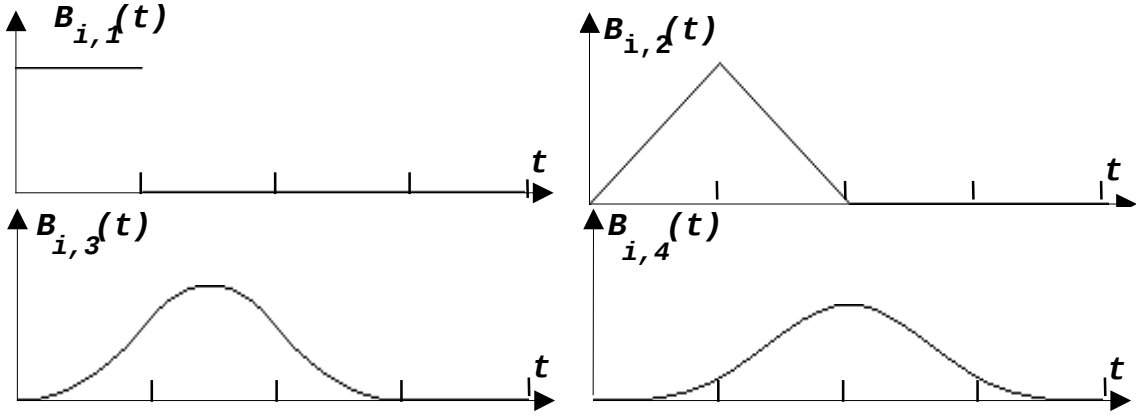


Fig. 8. Funcţiile B-spline normalizate de ordinul 1,2,3,4.

iar expresiile lor corespunzătoare sunt:

$$\boldsymbol{B}_{0,1}(t) = \begin{cases} 1, & 0 \le t < 1 \\ 0, & \text{în rest} \end{cases} \qquad \boldsymbol{B}_{0,3}(t) = \begin{cases} 0.5t^2, & 0 \le t < 1 \\ -t^2 + 3t - 1.5, & 1 \le t < 2 \\ 0.5(3-t)^2, & 2 \le t < 3 \\ 0, & \text{în rest} \end{cases}$$

$$\boldsymbol{B}_{0,2}(t) = \begin{cases} t, & 0 \le t < 1 \\ 2 - t, & 1 \le t < 2 \\ 0, & \text{în rest} \end{cases} \qquad \boldsymbol{B}_{0,4}(t) = \begin{cases} t^3/6, & 0 \le t < 1 \\ (-3t^3 + 12t^2 - 12t + 4)/6, & 1 \le t < 2 \\ (3t^3 - 24t^2 + 60t - 44)/6, & 2 \le t < 3 \\ (4-t)^2/6, & 3 \le t < 4 \\ 0, & \text{în rest} \end{cases} \tag{19}$$

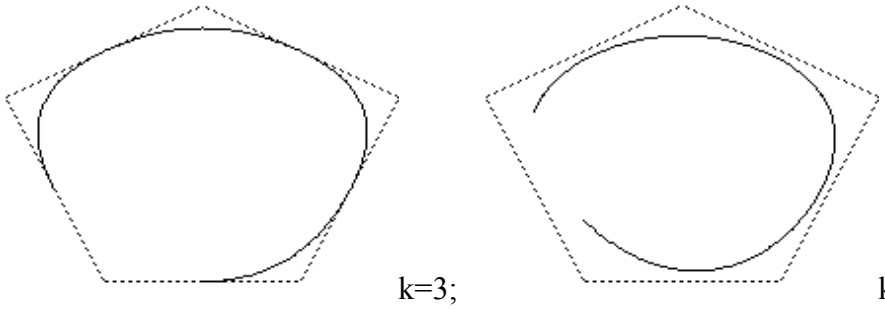
Dacă conturul de reprezentat este închis, funcţiile B-spline folosite sunt denumite închise sau periodice. Dacă punctele de control sunt uniform depărtate, adică:

$$t_{i+1} - t_i = \Delta t,\ \forall i \tag{20}$$

avem funcţii B-spline uniforme,cu proprietatea de periodicitate:

$$\boldsymbol{B}_{i,k}(t) = \boldsymbol{B}_{0,k}(t - i), i = k\text{-}1,k,...,n\text{-}k + 1, \tag{21}$$

Odată construite aceste curbe, ele pot fi translate, scalate, mărite-micşorate cu uşurinţă şi deasemenea pot fi făcute măsurători ale unor parametri ce caracterizează curba.



k=3; k=4

Dacă curba este descrisă printr-un set de puncte discrete, adică $t = s_0, s_1, ..., s_n$, avem:

$$\boldsymbol{x}(s_j) = \sum_{i=0}^{n} \boldsymbol{p}_i \, \boldsymbol{B}_{i,k}(s_j), \; j = 0,1,...,n \tag{22}$$

de unde, în formă matriceală:

$$\boldsymbol{B}_k \boldsymbol{P} = \boldsymbol{x} \Rightarrow \boldsymbol{P} = \boldsymbol{B}_k^{-1} \boldsymbol{x} \tag{23}$$

Remarcabil este faptul că, dacă se acceptă aproximarea prin curbe B-spline uniforme, matricea $\boldsymbol{B}_k$ devine o *matrice circulantă*, pentru care există algoritmi rapizi de inversare. Pentru cazul în care numărul de puncte ale curbei de aproximat este m>n, se adoptă o strategie bazată pe minimizarea erorii medii pătratice de aproximare, care duce deasemenea la inversarea unei matrici circulante.

6.4. Segmentarea texturilor.

Există opt abordări statistice în ceea ce priveşte caracterizarea texturilor, şi anume folosind *funcţia de autocorelaţie, transformările optice, transformările digitale, "cantitatea" de muchii din textură ("textural edgeness"), studiul elementelor structurale, probabilităţile de coocurenţă a nivelelor de gri, continuitatea nivelelor de gri ("gray tone run lengths") şi modelele autoregresive.*

Primele trei abordări se referă direct sau indirect la măsurători asupra frecvenţelor spaţiale. A patra metodă propune caracterizarea texturii prin cantitatea de muchii detectată în imagine. Metoda elementelor structurale este în esenţă o metodă de "potrivire" (matching) a imaginii texturii cu anumite elemente de structură definite pentru fiecare textură. Coocurenţa nivelelor de gri caracterizează interdependenţele spaţiale care există între pixelii texturii, ca şi metoda "gray tone run length". Analiza seriilor de timp este principiul care stă la baza modelului autoregresiv.

Fie $\boldsymbol{f}(x,y)$ imaginea (continuă, conţinând textura de studiat), astfel încât:

$$\boldsymbol{f}(x,y) = 0, \text{ pentru } (x < 0 \text{ şi } x > L_x) \text{ sau } (y < 0 \text{ şi } y > L_y) \tag{24}$$

Funcţia de autocorelaţie este definită prin:

$$\boldsymbol{R}(x,y) = \frac{\dfrac{1}{(L_x - |x|)(L_y - |y|)} \displaystyle\int_{-\infty-\infty}^{+\infty+\infty} \boldsymbol{f}(u,v)\, \boldsymbol{f}(u+x, y+y)\, du\, dv}{\dfrac{1}{L_x L_y} \displaystyle\int_{-\infty-\infty}^{+\infty+\infty} \boldsymbol{f}^2(u,v)\, du\, dv} \tag{25}$$

unde $|x| < L_x$ şi $|y| < L_y$. Deoarece funcţia de autocorelaţie este transformata Fourier a spectrului de putere, această metodă, indirect, caracterizează textura prin analiza frecvenţelor spaţiale. În acelaşi timp ea poate fi văzută ca o problema de "potrivire prin suprapunere" (matching).

Printre primele metode de analiză a texturilor se numără **transformările optice**, care utilizau difracţia Fraunhoffer pentru generarea unor trăsături care să caracterizeze texturile.

Transformările ortogonale discrete au fost şi ele mult utilizate pentru analiza texturilor. Gramenopolous utilizează în [43] Transformata Fourier Discretă pentru blocuri de imagine de 32x32 în scopul caracterizării unor texturi din imagini satelitare. Trecând spectrul de putere al imaginii în coordonate polare, Bajcsy şi Lieberman [09] remarcă proprietatea texturilor direcționale de a furniza un maxim al acestui spectru într-o direcție perpendiculară pe direcția principală a texturii. Pot fi utilizate şi alte tipuri de transformări ortogonale: Hadamard, Slant, etc.

Construind un filtru de decorelare, Pratt [84] foloseşte coeficienții acestui filtru împreună cu primele patru momente ale histogramei pentru a caracteriza textura.

Metoda propusă de Rosenfeld şi Troy [96] măsoară "cantitatea" de **muchii pe unitatea de suprafață**. Fiecărui pixel îi este ataşată o vecinătate pentru care se calculează media estimărilor gradientului, estimări obținute folosind una din metodele clasice.

Continuitatea nivelelor de gri ("gray tone run lengths") este o altă metodă care caracterizează textura folosind numărul de pixeli conectați având acelaşi nivel de gri. Măsurătorile care se fac vizează nu numai numărul efectiv de pixeli ci şi diametrul minim/maxim al grupării de pixeli, nivelul de gri, precum şi orientarea celor două diametre.

În esență, ultimele două metode realizează caracterizarea texturii prin calculul unei anume *măsuri* a ei. Pot fi utilizate însă şi alte măsuri, cum ar fi densitatea relativă de maxime-minime din imagine.

Matricea de coocurență a nivelelor de gri $P(i,j)$ conține în fiecare poziție (i,j) numărul de perechi de pixeli vecini având respectiv nivelele de gri "*i*" şi "*j*". De obicei ea se normalizează, împărțind fiecare element la numărul de pixeli ai imaginii, ea putând astfel fi văzută ca o matrice de probabilități de apariție, condiționate de poziția relativă.

Metodele structurale aplicate la imagini binare se referă la construcția unui element de structură ("şablon") care este apoi folosit într-o operație de filtrare, de fapt o eroziune morfologică, al cărei rezultat:

$$S \ominus B_1 = \{(x,y) \in S / B_1(x,y) \subset S\} \tag{26}$$

este tot o imagine binară în care densitatea pixelilor cu valoarea "1" poate fi folosită pentru caracterizarea imaginii. Au fost definite o multitudine de tipuri de elemente structurale, parametrizate sau nu.

Pentru cazul imaginilor în scară de gri, elementele de structură se numesc *primitive* şi sunt grupuri de pixeli conectați şi care satisfac anumite condiții particulare referitoare la nivelul de gri al pixelului curent, conjugat cu poziția lui relativă față de ceilalți. Numeroase asemenea corelații spațiale de descriere a primitivelor sunt citate în literatură.

Interdependența pixelilor apropiați din imagine poate fi caracterizată nu numai de funcția de autocorelație, dar şi utilizând **modelul autoregresiv** propus de McCormick şi Jayaramamurthy [71]. Acest model foloseşte metode care țin de analiza seriilor de timp pentru extragerea parametrilor de caracterizare a texturii. El poate fi deasemenea folosit pentru *generarea* texturilor.

Ashjari [6] propune caracterizarea texturilor prin intermediul valorilor proprii extrase din blocurile de imagine (32x32). Distribuția acestor valori proprii, ordonate descrescător permite diferențierea texturilor.

Metodele stochastice permit construcția unor modele foarte puternice pentru texturile naturale (nisip, iarbă, diferite tipuri de țesături, etc.), cu aplicații în sinteza imaginilor realiste şi în investigarea sistemului vizual uman (Pratt[85]).

Principial, sinteza texturilor pleacă de la un zgomot alb (câmp aleator), pentru a obține un model al texturii prin intermediul unui filtru:

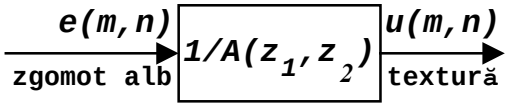


Fig. 9. Schema bloc de sinteză a texturilor.

6.5. Detalii de implementare.

O implementare C a unui algoritm de segmentare automată este prezentată în continuare.

```
int i,j,k;                    // contoare
int NMin, NMax;               // numărul de maxime şi minime din histogramă
long Max1, Max2;              // valorile primelor două maxime găsite
long Hist[256], Hist1[256];   // vectori histogramă
//calcul histogramă
for(j=0;j<DY;j++)
   for(i=0;i<DX;i++)
      Hist[Img1[i][j]]++;
do {
   // aplică filtrul nerecursiv de netezire asupra vectorului histogramă
   Hist1[0]=(Hist[0]+Hist[1])/3;
   for(i=1;i<255;i++)
      Hist1[i]=(Hist[i-1]+Hist[i]+Hist[i+1])/3;
   Hist1[255]=(Hist[254]+Hist[255])/3;
   for(i=0;i<256;i++)
      Hist[i]=Hist1[i];
   // calculează numărul şi poziţia maximelor din vectorul histogramă
   k=1; j=0; NMin=0; NMax=0; Max1=0; Max2=0;
   for(i=0;i<256;i++) {
      if(i!=0 && i!=255) {
         if(Hist[i-1]==Hist[i] && Hist[i]==Hist[i+1]) //1=palier
            ;
         if(Hist[i-1]==Hist[i] && Hist[i]< Hist[i+1]) //2=sfârşit minim
            if(k==2)
               k=1, NMin++, Min1=Min2, Min2=(i+j)/2;
         if(Hist[i-1]==Hist[i] && Hist[i]> Hist[i+1]) //3=sfârşit maxim
            if(k==1)
               k=2, NMax++, Max1=Max2, Max2=(i+j)/2;
         if(Hist[i-1]< Hist[i] && Hist[i]==Hist[i+1]) //4=început maxim
            j=i;
         if(Hist[i-1]< Hist[i] && Hist[i]< Hist[i+1]) //5=rampă crescătoare
            ;
         if(Hist[i-1]< Hist[i] && Hist[i]> Hist[i+1]) //6=maxim clar
            k=2, NMax++, Max1=Max2, Max2=i;
         if(Hist[i-1]> Hist[i] && Hist[i]==Hist[i+1]) //7=început minim
            j=i;
         if(Hist[i-1]> Hist[i] && Hist[i]< Hist[i+1]) //8=minim clar
            k=1, NMin++, Min1=Min2, Min2=i;
         if(Hist[i-1]> Hist[i] && Hist[i]> Hist[i+1]) //9=rampă căzătoare
            ;
      } else
         if(i==0) {
            if(Hist[i]==Hist[i+1])                    //1,4,7
               ;
            if(Hist[i]< Hist[i+1])                    //2,5,8
               ;
            if(Hist[i]> Hist[i+1])                    //3,6,9
               ;
         } else {
            if(Hist[i-1]==Hist[i])                    //1,2,3
               if(k==1) k=2, NMax++, Max1=Max2, Max2=(i+j)/2;
            if(Hist[i-1]< Hist[i])                    //4,5,6
               k=2, NMax++, Max1=Max2, Max2=i;
            if(Hist[i-1]> Hist[i])                    //7,8,9
               ;
```

```
            }
        }
    } while(NMax>2);  // repetă până se obțin doar două maxime
    // calcul prag de segmentare
    Th=(Max2*Hist[Max2]-Max1*Hist[Max1])/(Hist[Max2]-Hist[Max1]);
```

7. Transformata Hough.

7.1. Detecția curbelor folosind transformata Hough.

O metodă de separare a liniilor şi curbelor dintr-o imagine are la bază transformata Hough (Ballard[10]), (Duda[36]), (Haralick[45], (Hough[50]), (Pratt[84]). Fie $y = mx + n$ ecuația unei drepte în planul imaginii. Ei îi corespunde în planul descris de coordonatele (m,n) un punct, şi reciproc, fiecărei drepte din planul (m,n) îi corespunde o dreaptă în planul imaginii (fig. 1 şi 2).

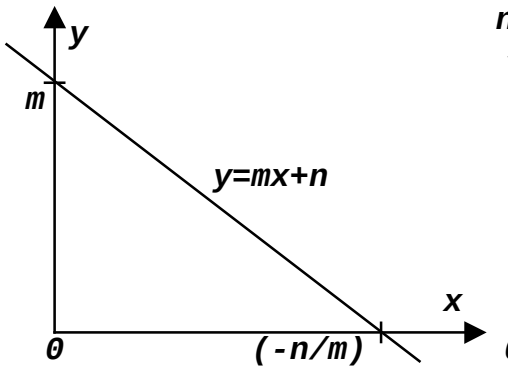


Fig. 1.

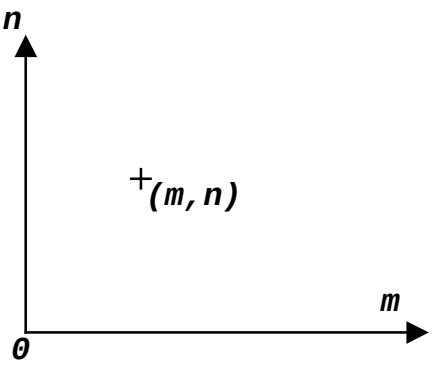


Fig. 2.

Ca urmare, unor puncte coliniare din planul (x,y) le corespunde un fascicol de drepte concurente într-un punct unic din planul transformatei Hough (fig. 3 şi 4).

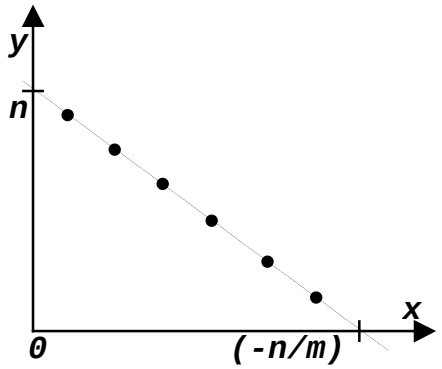


Fig. 3.

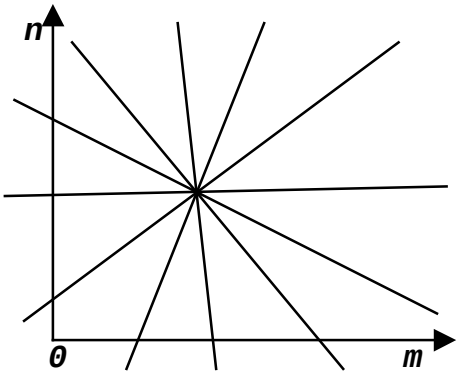


Fig. 4.

Deoarece pentru drepte verticale, valoarea parametrului "m" tinde către infinit, pentru a limita domeniul valorilor parametrilor se preferă exprimarea dreptei din planul (x,y) prin:

$$x\,\mathbf{sin}\theta + y\,\mathbf{cos}\theta = \rho \qquad (1)$$

unde "ρ" este distanța din origine şi până la dreaptă, iar "θ" este unghiul între dreaptă şi abscisă (fig. 5 şi fig. 6).

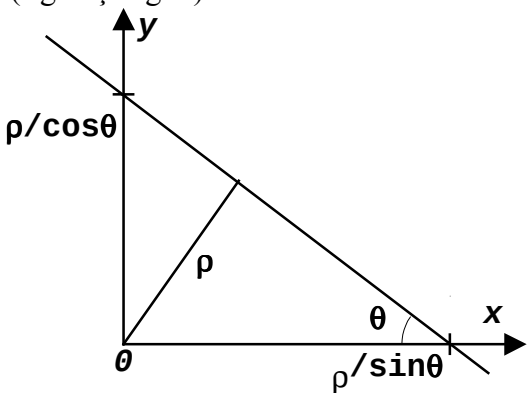


Fig. 5.

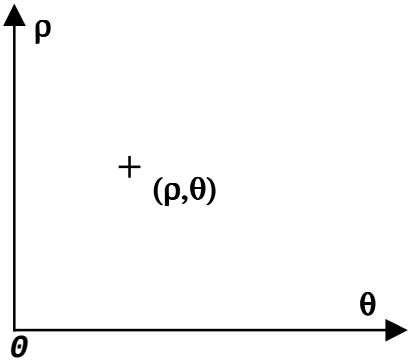


Fig. 6.

Ca urmare, unor puncte colineare din planul (x,y), le corespunde un set de sinusoide (fig. 7 şi fig. 8) care se intersectează într-un singur punct:

$$\rho_i(\theta) = x\ \mathbf{sin}\theta_i + y\ \mathbf{cos}\theta_i \qquad (2)$$

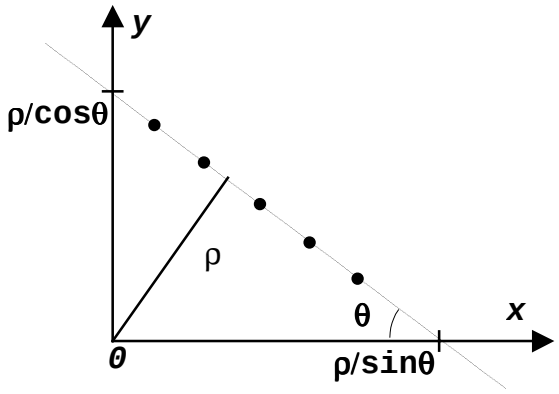


Fig. 7.

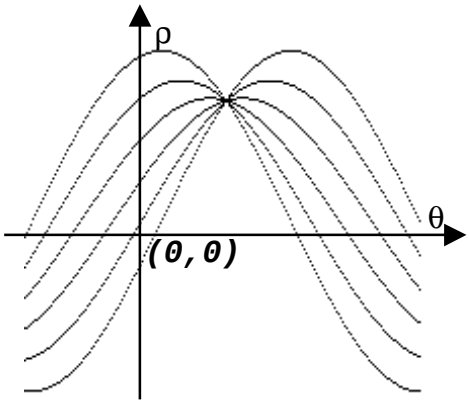


Fig. 8.

7.2. Versiuni ale transformării Hough.

Gama valorilor pe care le poate lua unghiul θ între normala la dreaptă şi abscisă, la o prima vedere, ar fi $[-\pi,\pi]$. Analizând această gamă de valori a unghiului θ, Duda şi Hart [36] ajung la concluzia că este suficient să se limiteze analiza la gama $[-\pi/2,\pi]$, deoarece gama

unghiurilor formate de vectorii de poziție ai pixelilor din imagine este $[0,\pi/2]$, iar față de aceştia, dreptele posibile din imagine formează un unghi de $[-\pi/2,\pi/2]$ (fig. 9).

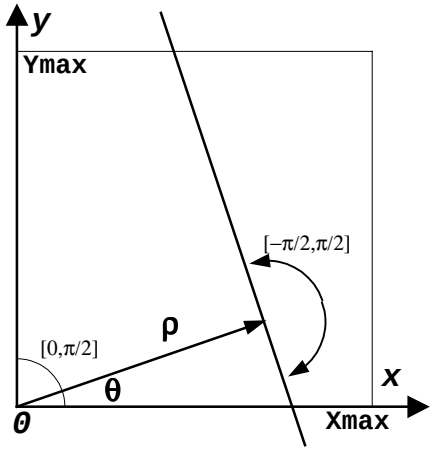


Fig. 9.

O'Gorman şi Clowes [77] propun modificarea transformării Hough, pentru înlănțuirea punctelor de muchie dintr-o imagine. Unghiul θ este obținut din direcția gradientului calculat pentru imagine, iar $\rho(\theta) = x\ \boldsymbol{sin}\theta + y\ \boldsymbol{cos}\theta$. Matricea transformatei Hough, în această versiune, se incrementează cu valoarea estimată a modulului gradientului în punctul curent, ceea ce permite mărirea influenței muchiilor puternic accentuate.

7.3. Generalizarea transformării Hough.

Dacă în imaginea inițială trebuiesc localizate curbe (contururi) cunoscute, cu dimensiuni şi orientări stabilite, care nu au o expresie analitică, poate fi utilizată metoda propusă de Ballard în [10].

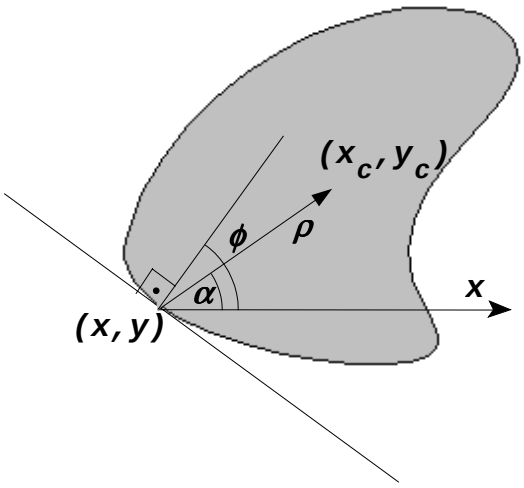


Fig. 10. Principiul generalizării transformării Hough.

Această metodă porneşte cu stabilirea unui punct de referință, notat (x_c,y_c).

Unghiul față de abscisă format de segmentul care uneşte punctul de referință cu pixelul curent, notat α în figura precedentă, poate fi măsurat pentru toți pixelii formei model.

Fiecărui pixel curent al conturului formei îi corespunde de asemenea o anumită orientare a muchiei, notată ϕ în figură, şi care se poate obține în imaginea inițială din studiul estimării discrete a vectorului gradient.

Ballard propune în continuare construirea unui tabel ("R-table") care să conțină, pentru toate orientările posibile ale vectorului gradient, informațiile necesare localizării punctului de referință. Pentru aceeaşi orientare a vectorului gradient, evident, sunt posibile mai multe variante de localizare a punctului de referință, adică mai multe perechi (ρ_j,α_j).

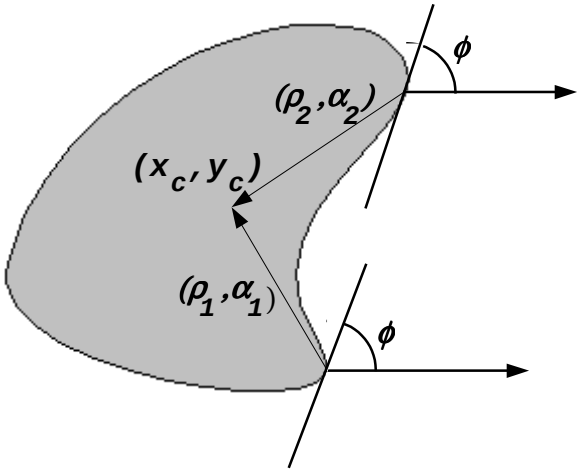


Fig. 11. Metoda Ballard de generalizare a transformării Hough.

Tabelul ce trebuie construit pentru forma de localizat are următoarea structură:

Unghiul față de abscisă al segmentului ce unește punctul curent cu punctul de referință	Localizări posibile ale punctului de referință
ϕ_1	$(\rho_1^1,\alpha_1^1),(\rho_2^1,\alpha_2^1),...,(\rho_{k_1}^1,\alpha_{k_1}^1)$
ϕ_2	$(\rho_1^2,\alpha_1^2),(\rho_2^2,\alpha_2^2),...,(\rho_{k_2}^2,\alpha_{k_2}^2)$
...	...
ϕ_m	$(\rho_1^m,\alpha_1^m),(\rho_2^m,\alpha_2^m),...,(\rho_{k_m}^m,\alpha_{k_m}^m)$

Pentru fiecare pixel de muchie se extrag din tabel localizările posibile pentru punctul de referință, iar apoi, pentru fiecare pereche (ρ_j^n,α_j^n) se calculează:

$$\begin{cases} x_c = x + \rho_j^n \cos\alpha_j^n \\ y_c = y + \rho_j^n \sin\alpha_j^n \end{cases} \tag{3}$$

În tabloul de acumulare se incrementează locația (x_c,y_c).

Metoda poate fi extinsă pentru cazul localizării obiectelor indiferent de mărimea (invarianța la scalare) şi orientarea (invarianța la rotație) lor (Illingworth[55]), dar volumul de calcul crește extrem de mult.

7.4. Optimizarea transformatei Hough.

Pentru limitatea volumului mare de operații necesare în calculul transformatei Hough, Kimme [60] propune utilizarea informațiilor privind direcția muchiei, date de vectorul gradient. Fie cazul cercului de rază fixă *R*.

$$(x-x_0)^2+(y-y_0)^2=R^2 \tag{4}$$

Metoda clasică implică pentru fiecare punct (x,y) din planul imaginii incrementarea în tabloul de acumulare a **tuturor** punctelor aflate, în spațiul parametrilor, pe cercul:

$$(x_0 - x)^2 + (y_0 - y)^2 = R^2 \tag{5}$$

Ecuația precedentă se poate scrie parametric:

$$\begin{cases} x_0 = x - R\,\boldsymbol{sin}\,\varphi \\ y_0 = y - R\,\boldsymbol{cos}\,\varphi \end{cases} \tag{6}$$

unde $\varphi = \varphi(x,y)$ poate fi direcția muchiei față de abscisă, furnizată de un operator tip gradient. În acest caz, pentru fiecare punct (x,y) din planul imaginii este incrementat în tabloul de acumulare **doar punctul** cu coordonatele (x_0,y_0) date de ecuațiile precedente.

Ideea transformării Hough poate fi generalizată pentru diferite alte curbe care pot fi descrise parametric prin:

$$\boldsymbol{f}(\boldsymbol{X},\boldsymbol{A}) = 0 \tag{7}$$

unde $\boldsymbol{X}$ este vectorul de poziție al punctului curent al curbei, iar $\boldsymbol{A}$ este vectorul de parametri, a cărui dimensiune depinde de tipul curbei.

Astfel, pentru detecția cercurilor dintr-o imagine avem o descriere cu trei parametri, (x_0,y_0,r):

$$(x - x_0)^2 + (y - y_0)^2 = r^2 \tag{8}$$

Pentru elipse sunt patru parametri, (x_0,y_0,a,b):

$$\frac{(x - x_0)^2}{a^2} + \frac{(y - y_0)^2}{b^2} = 1 \tag{9}$$

Problema principală care apare în asemenea cazuri este dimensiunea mare ocupată de "matricea de acumulare", devenită un tablou cu trei dimensiuni pentru cazul cercului şi unul cu patru dimensiuni pentru elipsă.

Metoda lui Kimme poate fi folosită şi în acest caz:

$$\frac{x - x_0}{a^2} + \frac{y - y_0}{b^2}\frac{dy}{dx} = 0 \tag{10}$$

unde:

$$dx/dy = \boldsymbol{tg}\,(\varphi) \tag{11}$$

iar

$$\varphi = \varphi(x,y) \tag{12}$$

este înclinarea muchiei dată de un operator tip gradient.

Ballard şi Brown [10] propun o metodă pentru reducerea şi mai drastică a volumului de calcul necesar construirii transformării Hough. Astfel pentru fiecare pereche de pixeli de muchie (x_1,y_1) şi (x_2,y_2) se pot obține următoarele patru ecuații (două sunt ecuațiile elipsei, iar celelalte se obțin prin derivare, conform metodei lui Kimme) care permit aflarea punctului de coordonate (x_0,y_0,a,b) din spațiul parametrilor pentru care se incrementează tabloul de acumulare:

$$\begin{cases} \dfrac{(x_1 - x_0)^2}{a^2} + \dfrac{(y_1 - y_0)^2}{b^2} = 1 \\ \dfrac{(x_2 - x_0)^2}{a^2} + \dfrac{(y_2 - y_0)^2}{b^2} = 1 \\ \dfrac{x_1 - x_0}{a^2} + \dfrac{y_1 - y_0}{b^2}\,\boldsymbol{tg}\,\varphi(x_1,y_1) = 0 \\ \dfrac{x_2 - x_0}{a^2} + \dfrac{y_2 - y_0}{b^2}\,\boldsymbol{tg}\,\varphi(x_2,y_2) = 0 \end{cases} \tag{13}$$

unde (x_1,y_1) şi (x_2,y_2) sunt pixeli de muchie. Volumul de calcul necesar în acest caz este proporțional cu pătratul numărului de pixeli de muchie.

Este interesant de studiat gama valorilor pe care le poate lua distanța ρ de la origine la dreaptă (Bulea[14]), pentru diferite valori ale lui θ. Imaginea având originea în colțul din stânga-jos, vom nota cu X_{max} şi Y_{max} dimensiunile ei, iar cu $R_{min}(\theta)$ şi $R_{max}(\theta)$ valorile limită pentru ρ, funcție de unghiul θ.

De exemplu, dacă $\theta \in [-\pi/2,0]$, se constată (fig. 12) că avem:

$$R_{max}(\theta) = X_{max}\cos\theta \tag{14}$$

Analog, studiind figurile care urmează, 13 şi 14, corespunzătoare cazurilor $\theta \in [0,\pi/2]$ şi respectiv $\theta \in [\pi/2,\pi]$, se pot sintetiza expresii pentru $R_{min}(\theta)$ şi $R_{max}(\theta)$.

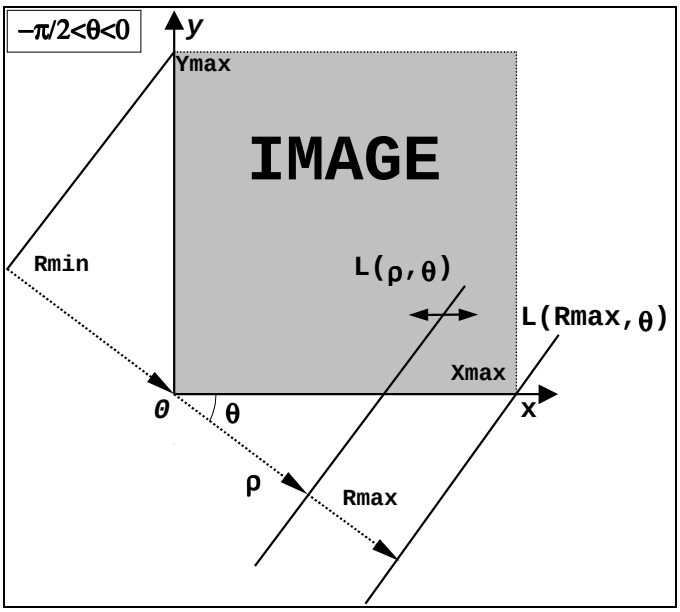


Fig. 12.

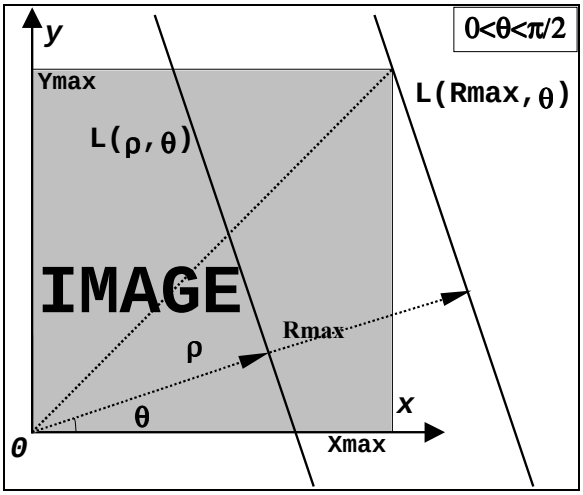


Fig. 13.

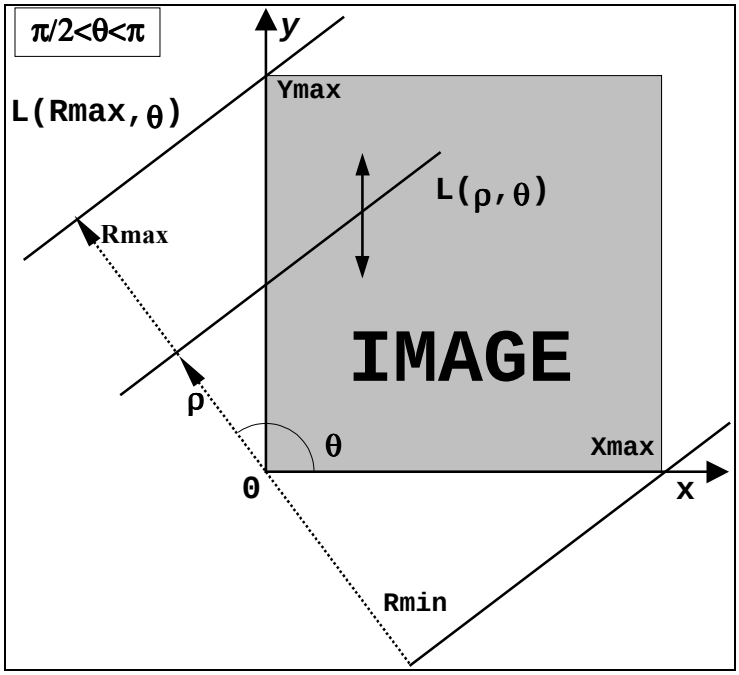


Fig. 14.

Rezultă relațiile:

$$R_{min}(\theta) = \begin{cases} Y_{max}\,\boldsymbol{sin}\,\theta, \text{pentru } \theta \in [-\pi/2,0]; \\ 0, \text{pentru } \theta \in (0,\pi/2]; \\ X_{max}\,\boldsymbol{cos}\,\theta, \text{pentru } \theta \in (\pi/2,\pi]. \end{cases} \quad (15)$$

$$R_{max}(\theta) = \begin{cases} X_{max}\,\boldsymbol{cos}\,\theta, \text{pentru } \theta \in [-\pi/2,0]; \\ \sqrt{X_{max}^2 + Y_{max}^2}\,\boldsymbol{cos}\,[arctg(Y_{max}/X_{max}) - \theta], \text{pentru } \theta \in (0,\pi/2]; \\ Y_{max}\,\boldsymbol{sin}\,\theta, \text{pentru } \theta \in (\pi/2,\pi]. \end{cases} \quad (16)$$

În figurile următoare sunt reprezentate domeniile de valori posibile pentru transformarea Hough în versiunea originală (fig. 15) şi respectiv în versiunile Duda & Hart şi O'Gorman & Clowes $(\rho > 0)$ (fig 16).

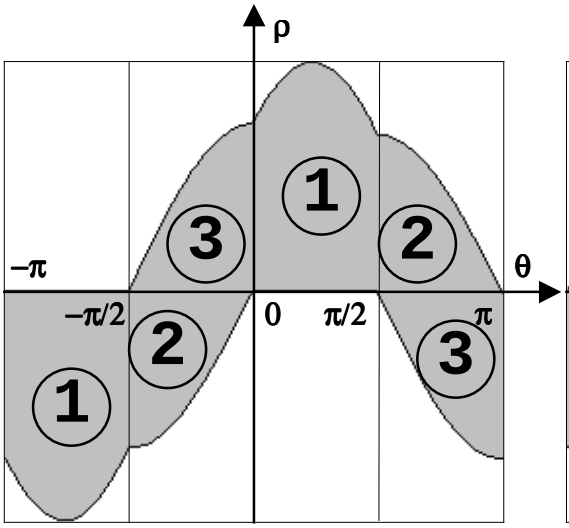


Fig. 15.

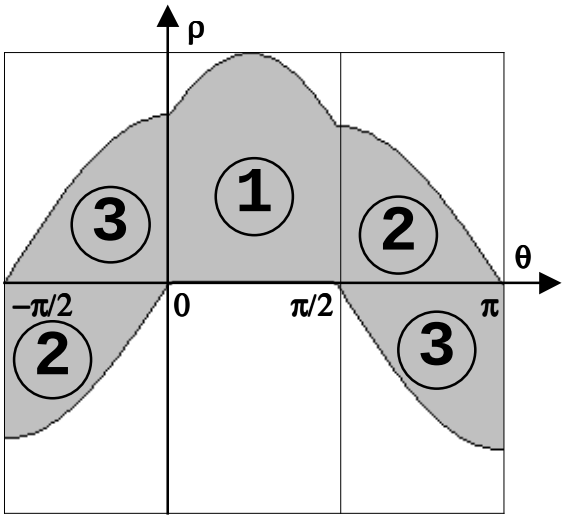


Fig. 16.

S-au marcat cu **(1)**, **(2)**, **(3)** regiunile din transformare care se repetă prin simetrie. Această se datorează faptului că linie definită de perechea (ρ,θ) este identică cu cea dată de $(-\rho,\pi+\theta)$.

Următoarea transformare (fig. 17) pemite reducerea gamei dinamice pentru unghiul θ de la $[-\pi/2,\pi]$ la intervalul $[0,\pi]$:

$$\rho'(\theta) = \begin{cases} \rho(\theta), & \text{pentru } \theta \in [0,\pi] \\ -\rho(\pi+\theta), \text{pentru } \theta \in [-\pi/2,0) \end{cases} \quad (17)$$

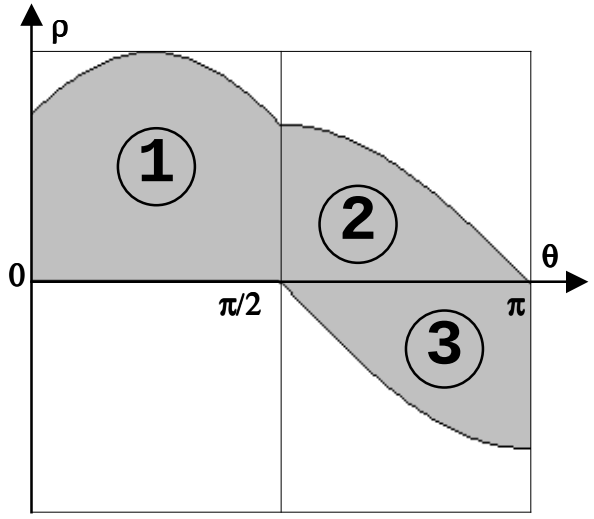


Fig. 17.

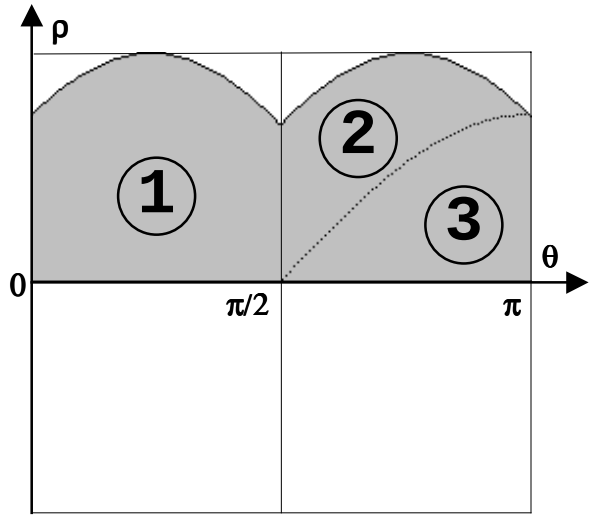


Fig. 18.

Desigur, o *transformare alternativă* poate fi definită pentru a reduce aceeaşi gamă dinamică la $[-\pi/2,\pi/2]$ şi ea este dată de:

$$\rho'(\theta)=\begin{cases}\rho(\theta), & \text{pentru } \theta\in[-\pi/2,\pi/2]\\ -\rho(\theta-\pi), & \text{pentru } \theta\in(\pi/2,\pi]\end{cases} \tag{18}$$

Mai departe se pot obține doar valori pozitive pentru $\rho'(\theta)$ prin transformarea (vezi fig. 18):

$$\rho''(\theta)=\begin{cases}\rho'(\theta), & \text{pentru } \theta\in[0,\pi/2);\\ \rho'(\theta)-R_{min}(\theta), & \text{pentru } \theta\in[\pi/2,\pi]\end{cases} \tag{19}$$

iar gama dinamica a lui $\rho''(\theta)$ devine:

$$\rho''(\theta)\in[0,R_{max}(\theta)-R_{min}(\theta)] \tag{20}$$

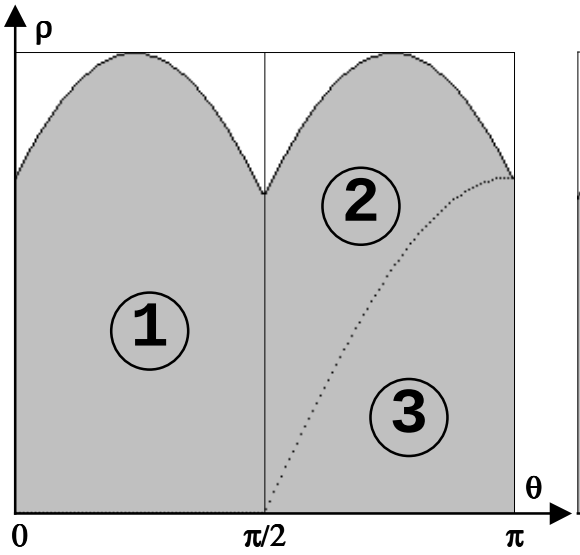


Fig. 19.

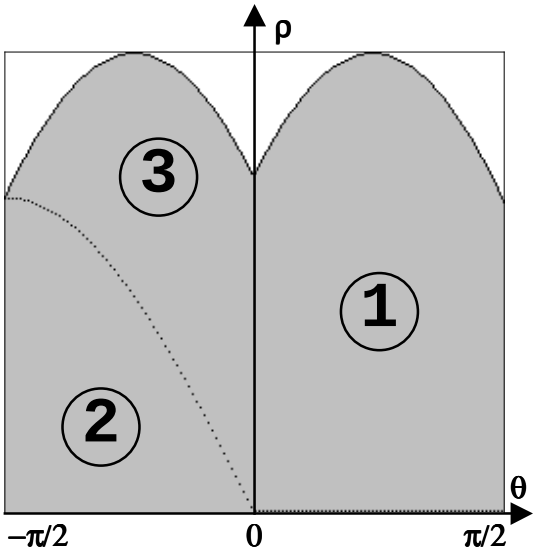


Fig. 20.

Se remarcă din figura precedentă (fig. 19) că transformarea propusă conservă toată informația existentă în imaginea transformării precedente, în timp ce gama de valori pentru unghiul θ s-a redus cu o treime. O reprezentare grafică a transformării alternate este dată în Fig. 20.

Dacă se apelează şi la normalizarea:

$$\rho'''(\theta)=\frac{\rho''(\theta)}{R_{max}(\theta)-R_{min}(\theta)} \tag{21}$$

gama de valori posibile pentru ordonată devine independentă de θ, adică:

$$\rho'''(\theta)\in[0,1],\ \forall\theta\in[0,\pi] \tag{22}$$

7.5. Transformarea inversă.

Efectuând transformarea propusă, după găsirea maximelor $\rho'''_k(\theta_k)$ în matricea transformării, liniile se localizează în imaginea inițială folosind transformarea inversă dată de:

$$\rho''_k(\theta_k)=\rho'''_k(\theta_k)[R_{max}(\theta_k)-R_{min}(\theta_k)] \tag{23}$$

$$\rho'_k(\theta_k)=\begin{cases}\rho''_k(\theta_k), & \text{pentru } \theta_k\in[0,\pi/2);\\ \rho''_k(\theta_k)+R_{min}(\theta_k), & \text{pentru } \theta_k\in[\pi/2,\pi]\end{cases} \tag{24}$$

deci:

$$\rho_k(\theta_k) = \begin{cases} \rho'_k(\theta_k), & \text{pentru } \theta_k \in [0, \pi/2] \\ \rho'_k(\theta_k), & \text{pentru } \theta_k \in (\pi/2, \pi] \text{ şi } \rho'_k(\theta_k) \geq 0 \\ -\rho'_k(\theta_k - \pi), & \text{pentru } \theta_k \in (\pi/2, \pi] \text{ şi } \rho'_k(\theta_k) < 0 \end{cases} \quad (25)$$

de unde se obține ecuația dreptei din imaginea inițială:

$$x\ \sin\theta_k + y\ \cos\theta_k = \rho_k(\theta_k); \quad (26)$$

7.6. Calculul propriu-zis al transformatei Hough.

Transformata Hough poate fi estimată prin trasarea curbelor sinusoidale în planul (θ,ρ) şi apoi calculând densitatea locală de curbe în fiecare punct. Definind în prealabil matricea de acumulare, aceste curbe sunt de obicei *trasate* prin incrementarea tuturor locațiilor corespunzătoare ale matricii.

Cea mai simplă soluție este de a calcula, pentru *fiecare* valoare discretă a unghiului:

$$\rho_k(\theta_k) = x\ \sin\theta_k + y\ \cos\theta_k \quad (27)$$

incrementând doar elementul (θ_k,ρ_k) al matricii de acumulare. În consecință, doar o singură locație este incrementată pentru fiecare valoare discretă a unghiului şi de aici rezultă principalul dezavantaj al acestei metode: pentru porțiunile de sinusoidă având panta mare, doar puține elemente ale matricii de acumulare sunt modificate, după cum se poate vedea din figura următoare (21):

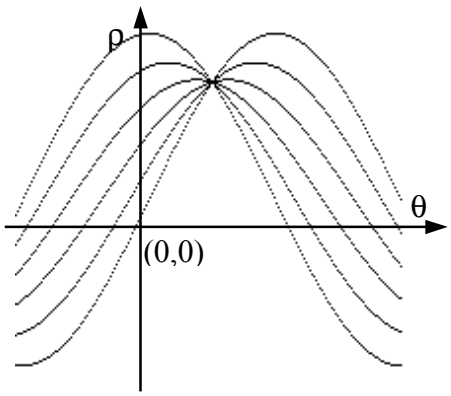


Fig. 21.

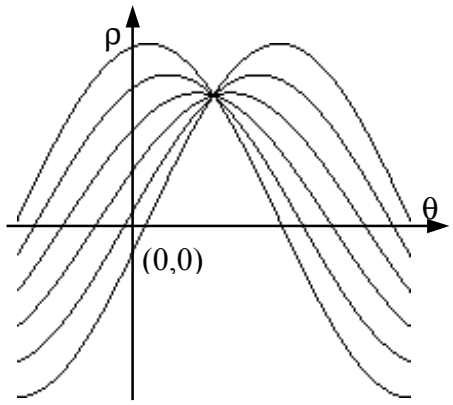


Fig. 22.

O soluție puțin mai bună este de a calcula perechile (θ_k,ρ_k) şi $(\theta_{k+1},\rho_{k+1})$, şi de a incrementa toate elementele matricii de acumulare aflate pe linia care unește aceste puncte în planul transformatei Hough, dar timpul de calcul necesar crește (Fig. 22).

O metodă mai precisa de a efectua transformata Hough se obține dacă o regiune rectangulară a planului Hough, localizată în jurul punctului (θ_k,ρ_k) este asociată fiecărei locații a acumulatorului, ca în figura 23.

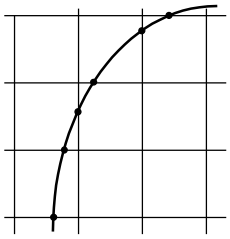


Fig. 23.

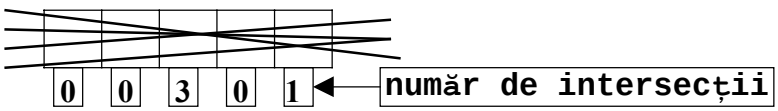


Fig. 24.

O idee foarte bună este incrementarea fiecărei locații a matricii de acumulare cu o valoare proporțională cu lungimea sinusoidei curente în careul corespunzător. Astfel se obține o reprezentare mai netedă și mai precisă a transformatei Hough, permițând localizarea mai facilă a punctelor de maxim. Calculul acestor lungimi se simplifică considerabil dacă curbele sinusoidale se aproximează prin segmente în interiorul fiecărui asemenea careu.

Deși aceasta este cea mai bună metodă prezentată până acum, ea are un dezavantaj major, prezentat în figura 24. Valorile corespunzătoare calculate pentru fiecare careu elementar sunt aproximativ egale (deoarece lungimile tuturor curbelor trasate sunt aproximativ egale), chiar dacă punctul de intersecție este foarte bine precizat. Această situație afectează precizia de localizare a maximelor în matricea de acumulare. Problema poate fi remediată prin numărarea punctelor de intersecție în fiecare careu elementar.

Pentru fiecare pereche de curbe definită de (x_k,y_k) și (x_j,y_j), punctul de intersecție este dat de:

$$\theta_{kj} = -\mathbf{arctg}\frac{x_k - x_j}{y_k - y_j} + n\pi, \quad n = \ldots -1,0,1,\ldots \quad \text{și} \tag{28}$$

$$\rho_{kj} = \frac{x_k y_j - x_j y_k}{\sqrt{(x_k - x_j)^2 + (y_k - y_j)^2}} \tag{29}$$

Alegând valoarea unghiului care îndeplinește condiția $\rho_{kj} \in [0,2\pi]$ atunci:

$$\rho'_{kj}(\theta_{kj}) = \rho_{kj}(\theta_{kj}) \tag{30}$$

și următoarele calcule trebuiesc efectuate:

$$\rho''_{kj}(\theta_{kj}) = \begin{cases} \rho'_{kj}(\theta_{kj}), & \text{pentru } \theta_{kj} \in [0,\pi/2); \\ \rho_{kj}(\theta_{kj}) - R_{\mathbf{min}}(\theta_{kj}), & \text{pentru } \theta_{kj} \in [\pi/2,\pi] \end{cases} \tag{31}$$

$$\rho'''_{kj}(\theta_{kj}) = \frac{\rho''_{kj}(\theta_{kj})}{R_{\mathbf{max}}(\theta_{kj}) - R_{\mathbf{min}}(\theta_{kj})} \tag{32}$$

Numărul necesar de operații este $N*(N-1)/2$, unde N este numărul de curbe, și el este mai mare ca numărul de operații necesar efectuării transformatei Hough clasice dacă $(N-1)/2 > \Theta$, unde Θ este numărul de valori discrete ale unghiului, pentru $\theta_{kj} \in [0,\pi)$.

Această metodă poate fi dezvoltată ținând cont de tăria muchiilor, adică setând incrementul utilizat la o valoare dependentă de valoarea estimată a modulului gradientului pentru fiecare punct. Această idee derivă din metoda lui O Gorman și Clowes.

Ca urmare, incrementul utilizat pentru fiecare intersecție este:

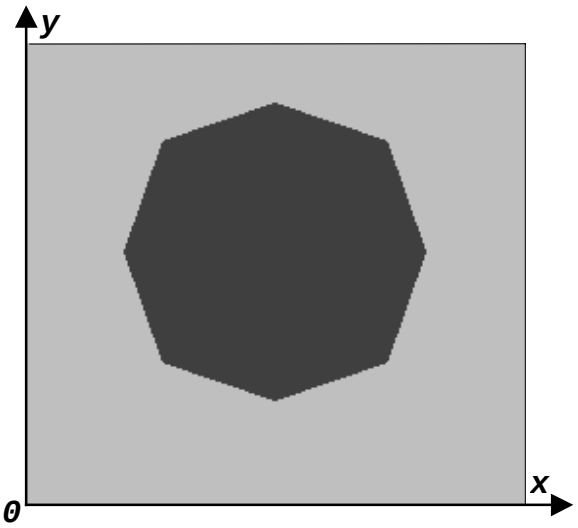
$$\Delta \mathbf{h}(\theta_{kj},\rho_{kj}) = 1 + \lambda(|\mathbf{grad}[\mathbf{f}(x_k,y_k)]| + |\mathbf{grad}[\mathbf{f}(x_j,y_j)]|), \text{ unde } \lambda > 0 \tag{33}$$

7.7. Rezultate comparative și concluzii.

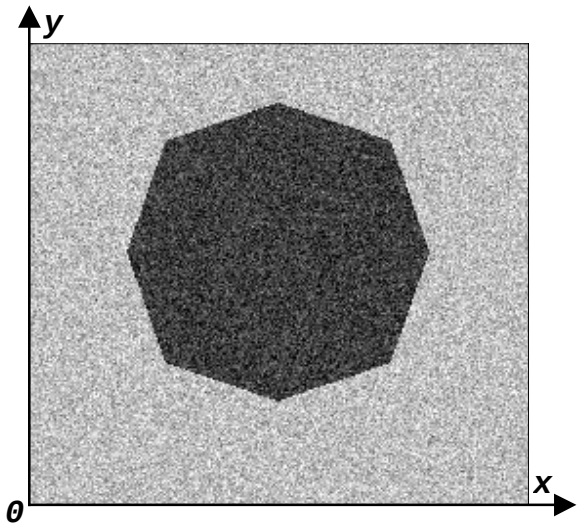
În continuare sunt prezentate rezultatele comparative obținute aplicând cele trei versiuni prezentate ale transformării Hough. S-au utilizat două imagini generate cu 256 nivele de gri (a doua conține zgomot gaussian, $\sigma^2 = 50$), și o imagine cu 256 nivele de gri, obținută de la o camera TV.

Pentru transformata Hough originală și pentru versiunea Duda & Hart a fost utilizat operatorul Sobel pentru detecția muchiilor, iar pragul de segmentare folosit a fost ales la 128.

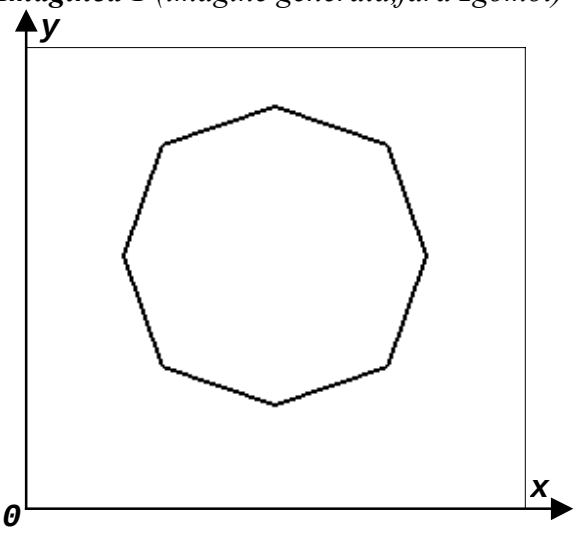
Rezultatele obținute pentru varianta propusă sunt prezentate pentru cazul transformatei calculate prin metoda trasării cu numărarea intersecțiilor $(\lambda = 1/255)$.



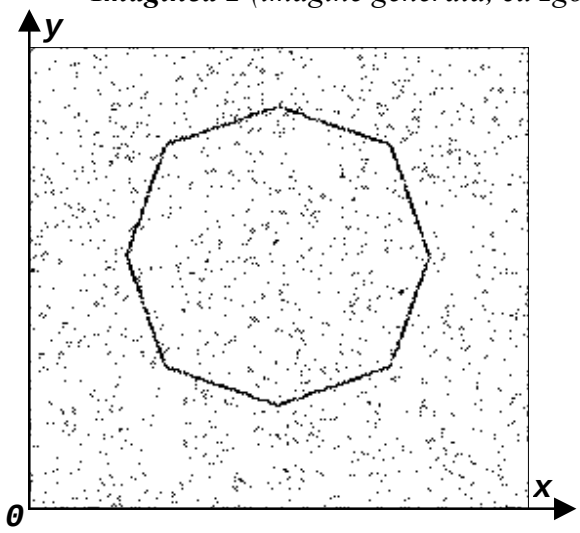
Imaginea 1 *(imagine generată,fără zgomot)*



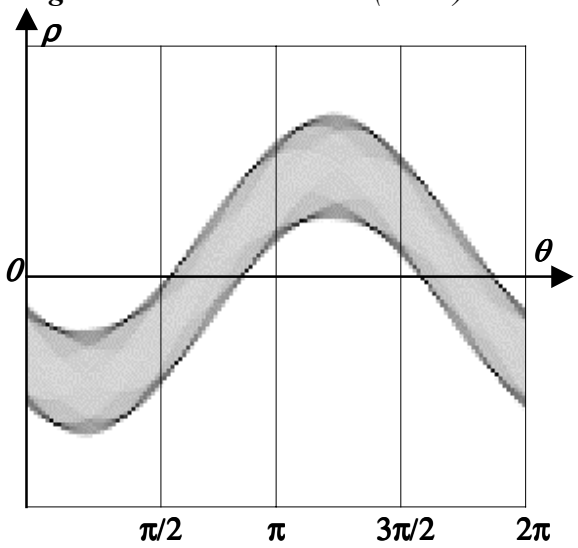
Imaginea 2 *(imagine generată, cu zgomot)*



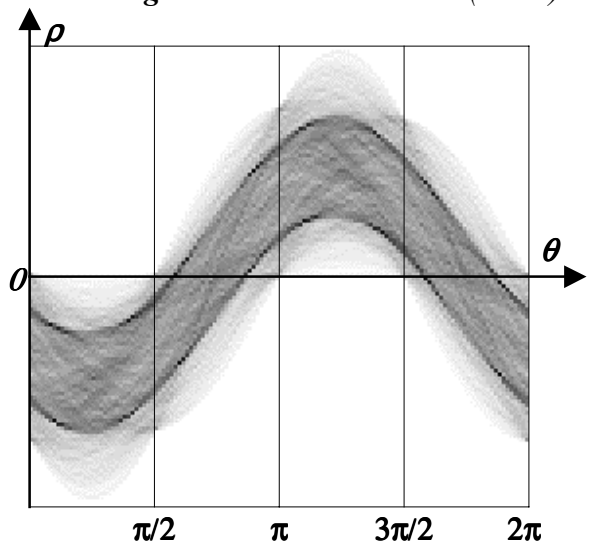
Imaginea 1 *- Muchii extrase (Sobel)*



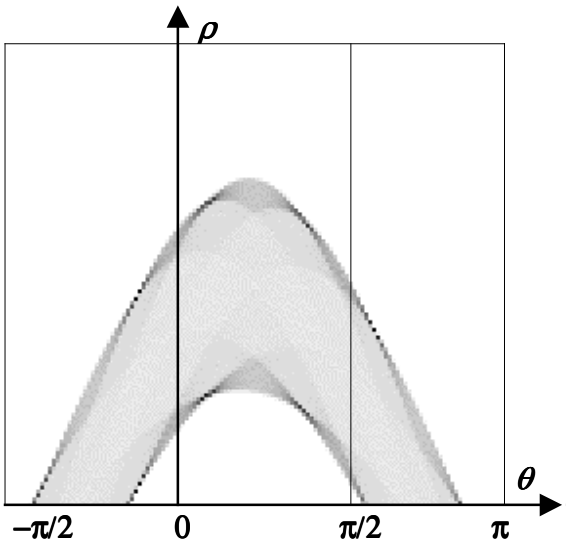
Imaginea 2 *- Muchii extrase (Sobel)*



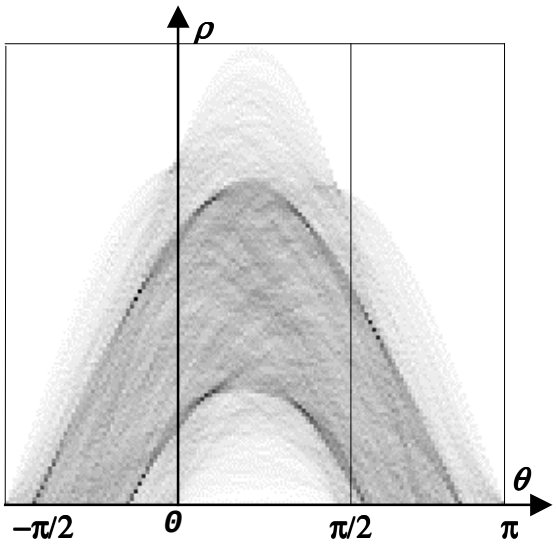
Imaginea 1 *- Transformata Hough*



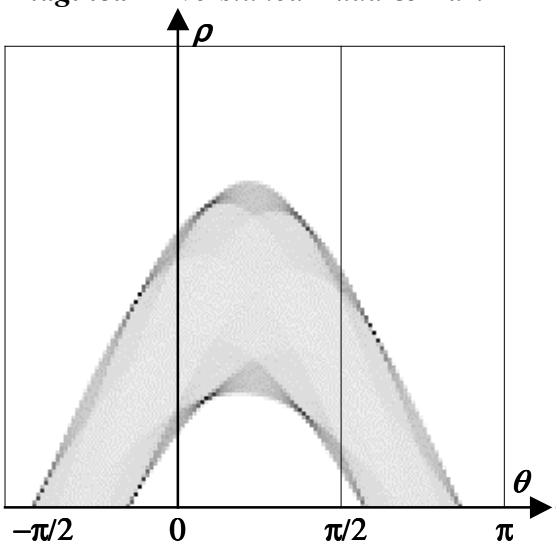
Imaginea 2 *- Transformata Hough*



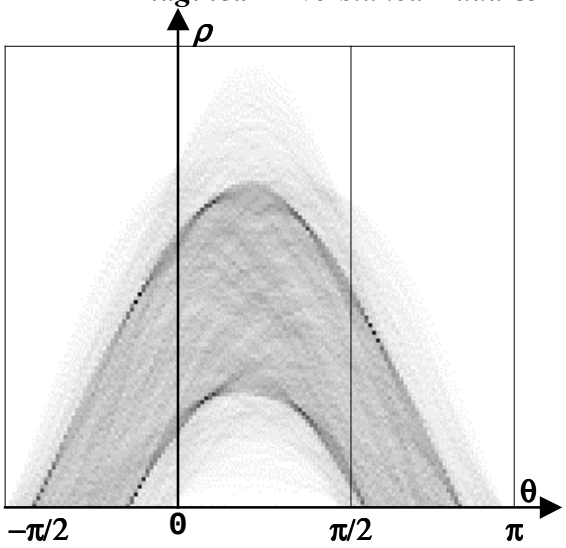
Imaginea 1 *- versiunea Duda & Hart*



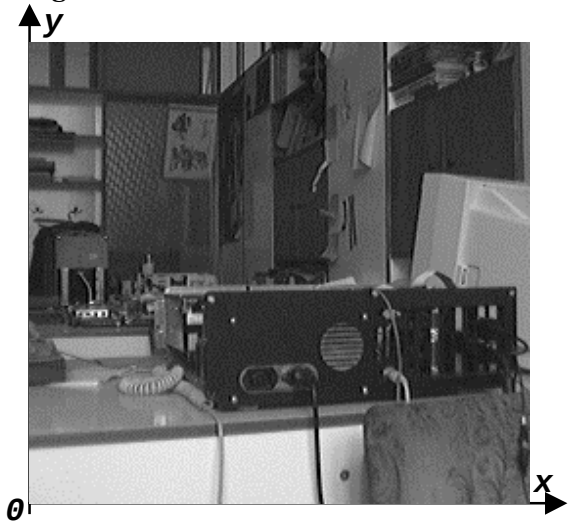
Imaginea 2 *- versiunea Duda & Hart*



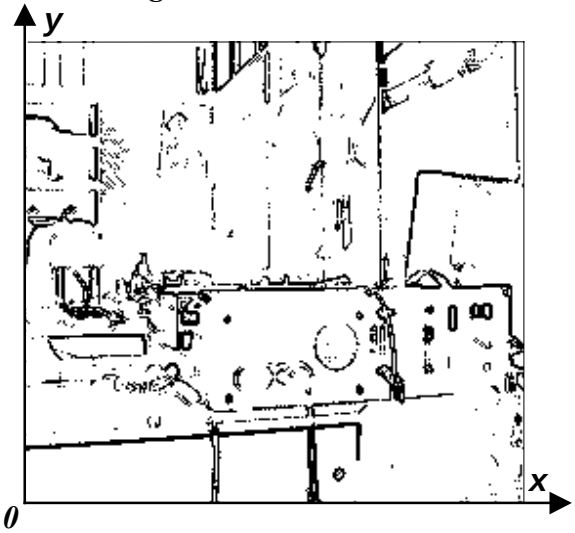
Imaginea 1*- versiunea O'Gorman & Clowes*



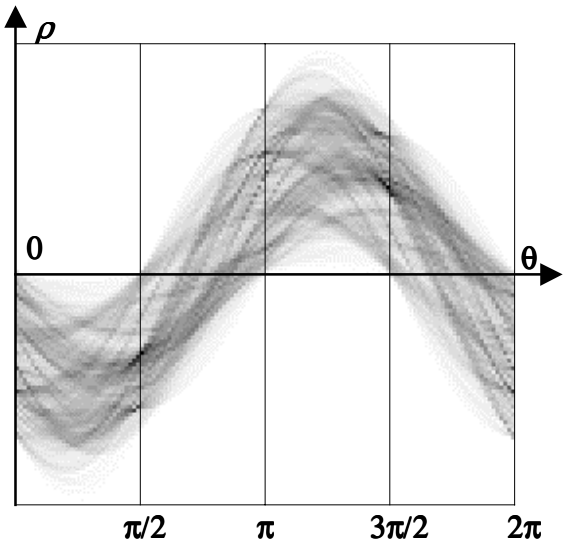
Imaginea 2*- versiunea O'Gorman & Clowes*



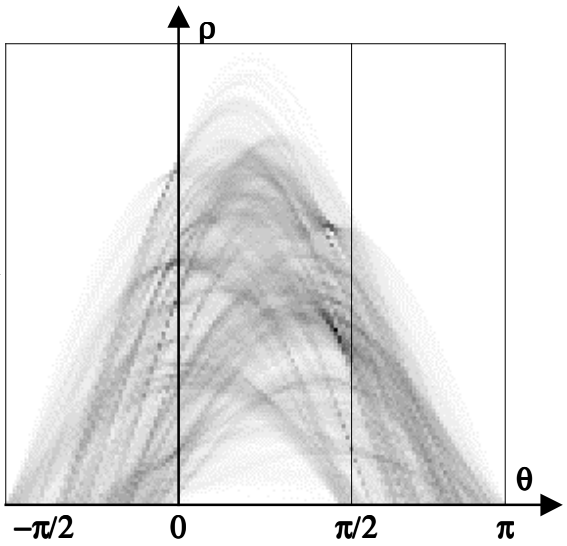
Imaginea 3 *(imagine achiziționată)*



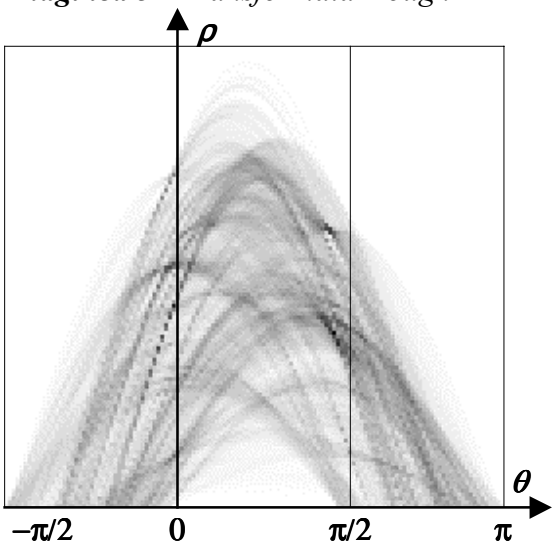
Imaginea 3 *Muchii extrase (Sobel)*



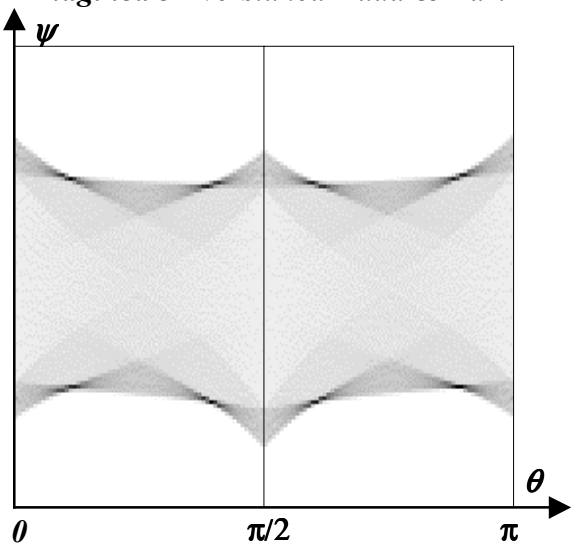
Imaginea 3 *- Transformata Hough*



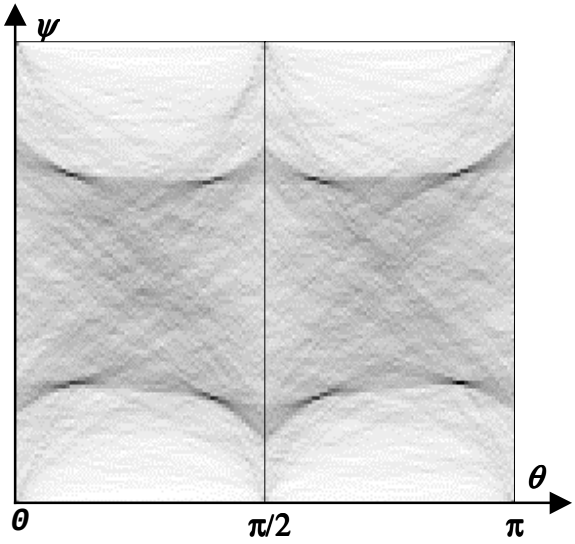
Imaginea 3 *- versiunea Duda & Hart*



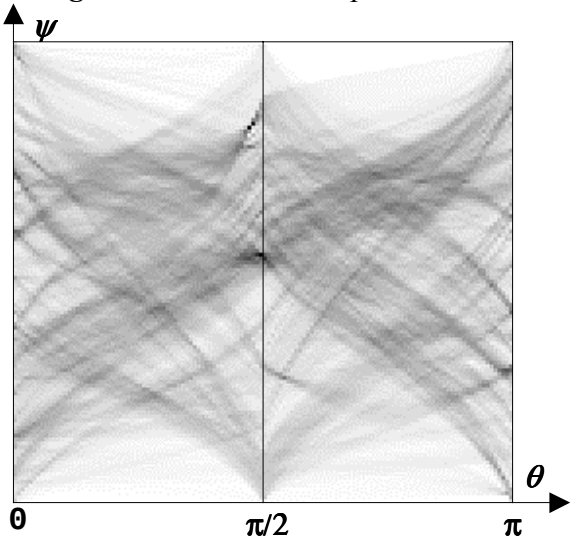
Imaginea 3 *- versiunea O'Gorman & Clowes*



Imaginea 1 *– Versiunea optimizată, trasare*



Imaginea 2 *- Versiunea optimizată, trasare*



Imaginea 3 *- Versiunea optimizată, trasare*

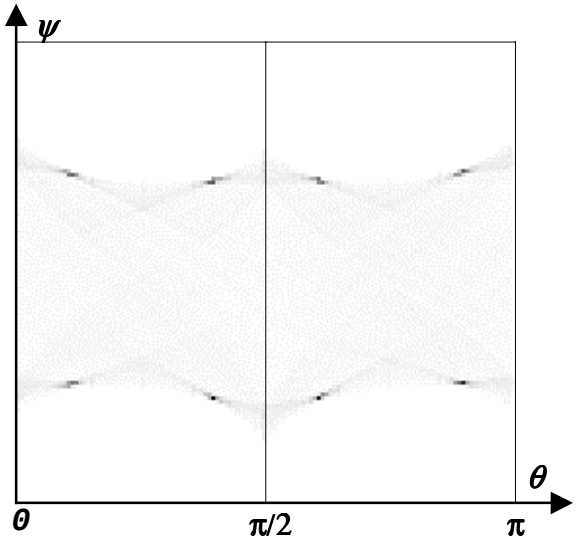


***Image 2** - Versiunea optimizată, numărare.*

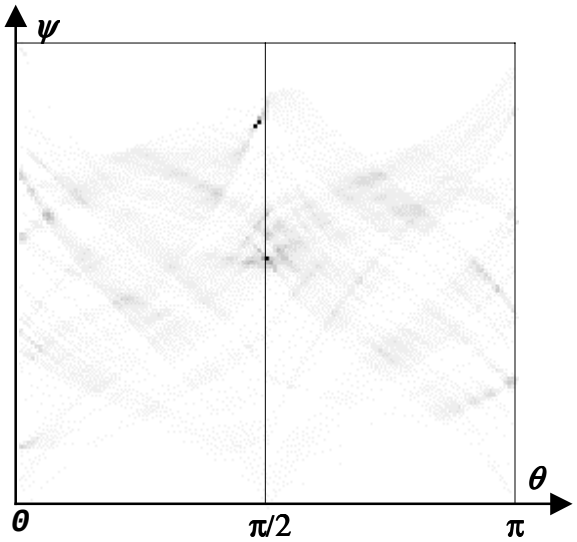
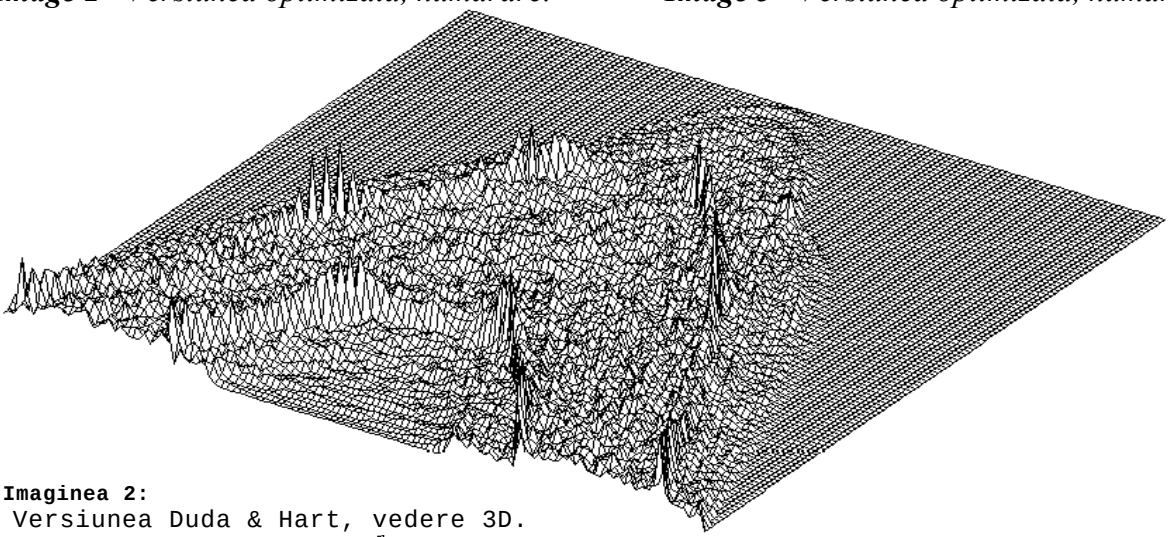
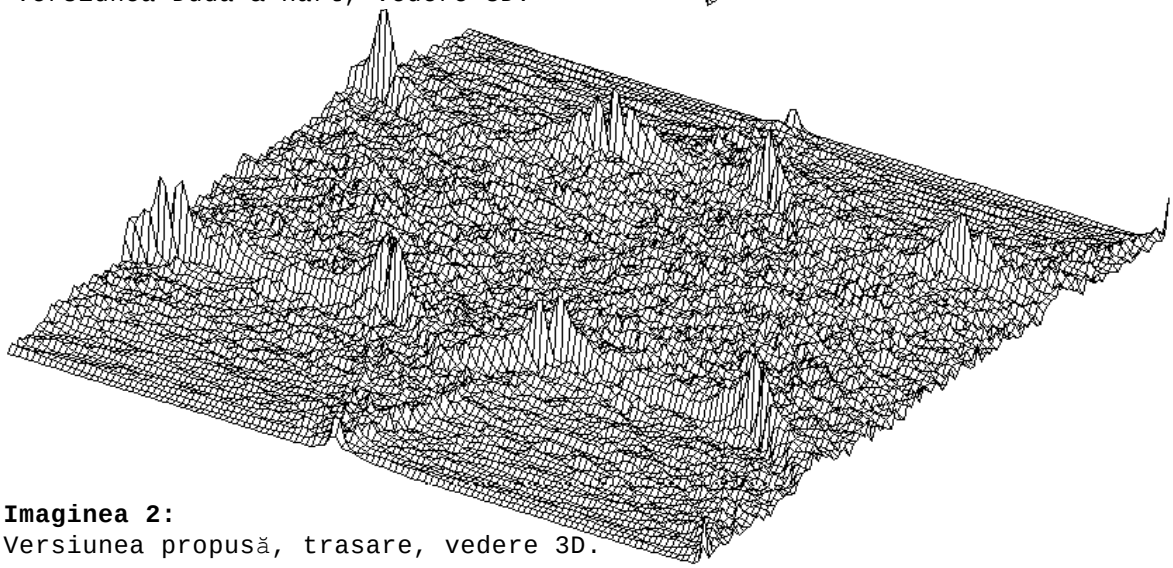


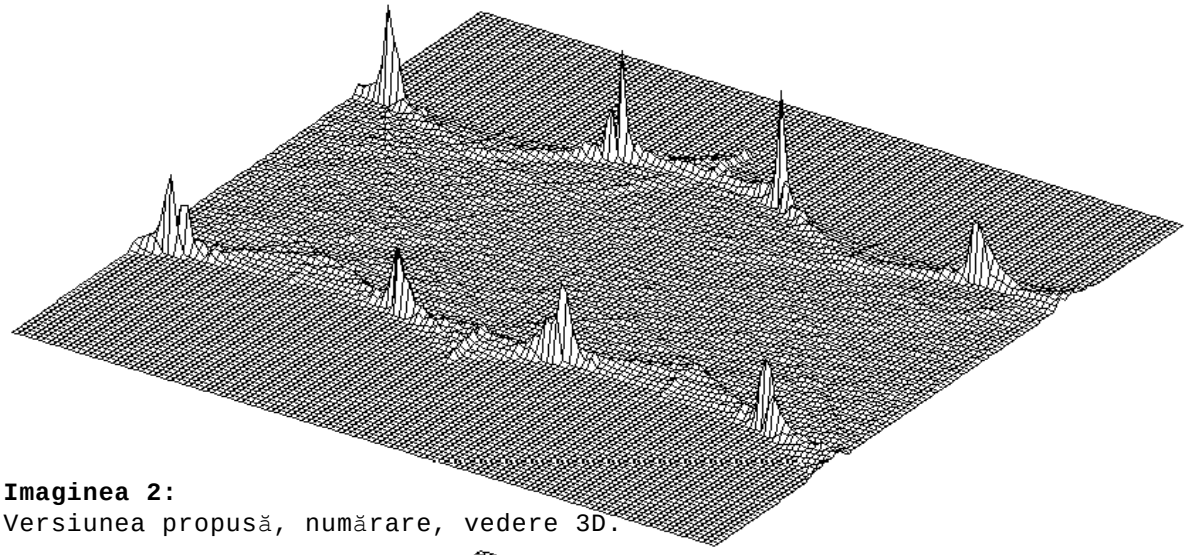
***Image 3** - Versiunea optimizată, numărare.*



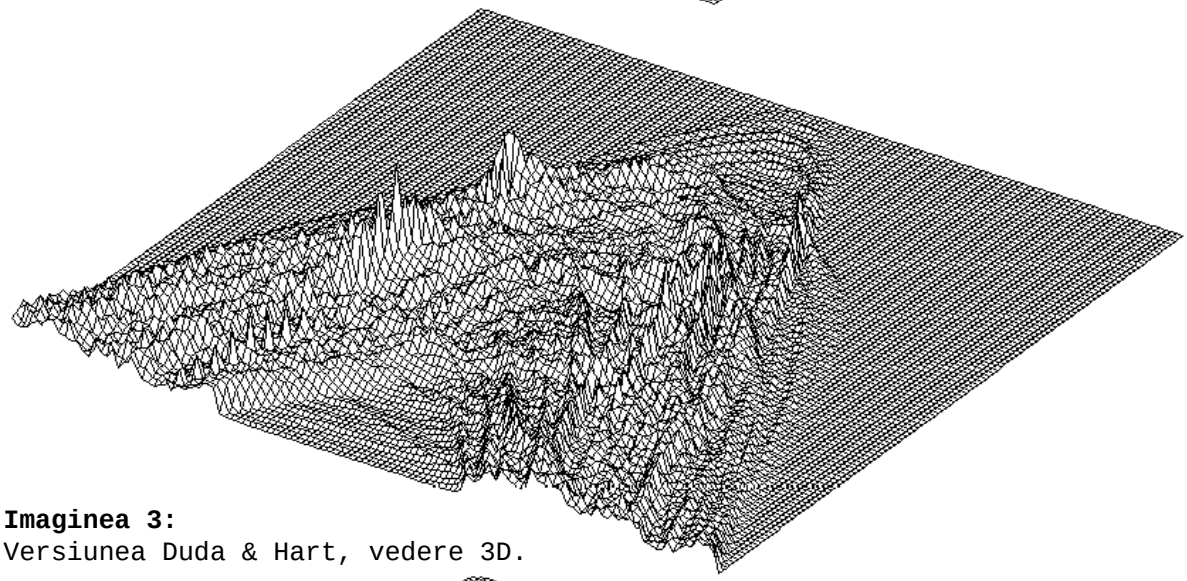
Imaginea 2:
Versiunea Duda & Hart, vedere 3D.



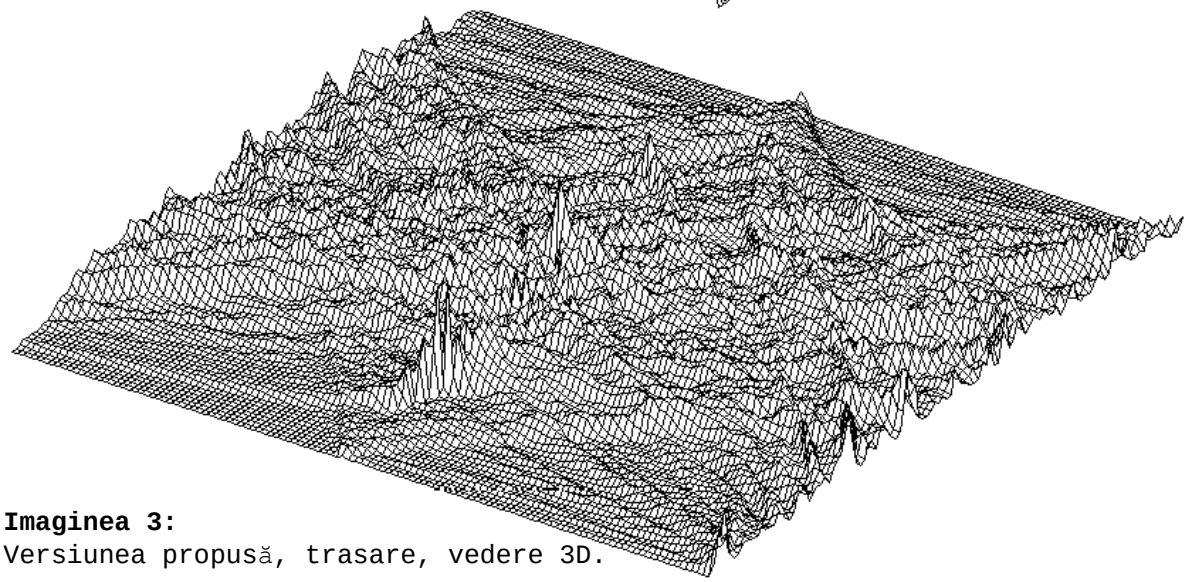
Imaginea 2:
Versiunea propusă, trasare, vedere 3D.



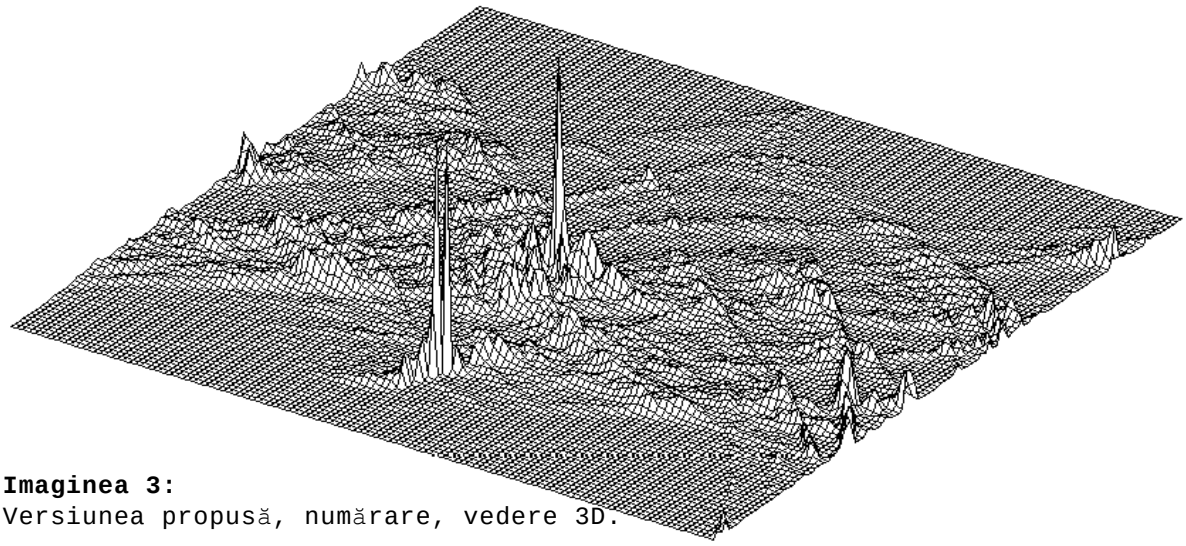
Imaginea 2:
Versiunea propusă, numărare, vedere 3D.



Imaginea 3:
Versiunea Duda & Hart, vedere 3D.



Imaginea 3:
Versiunea propusă, trasare, vedere 3D.



Imaginea 3:
Versiunea propusă, numărare, vedere 3D.

Deşi transformata Hough necesită un mare volum de operaţii şi multă memorie, ea rămâne un instrument foarte puternic în analiza imaginilor.

Versiunea optimizată prezintă o creştere semnificativă a rezoluţiei pe ambele axe. Astfel, domeniul pentru parametrul θ este redus de la $[-\pi/2,\pi]$ pentru versiunea Duda & Hart, la $[0,\pi]$ sau echivalent $[-\pi/2,\pi/2]$ pentru versiunea propusă. Aceasta transformare nu are zone neutilizate în matricea de acumulare, precum prezintă versiunile clasice ale transformării Hough (versiunea Duda & Hart foloseşte doar 60% din locaţiile matricii de acumulare). Transformata inversă descrisă în această lucrare trebuie utilizată pentru localizarea segmentelor de dreaptă din imaginea iniţială, fiecare corespunzând unui maxim local în matricea de acumulare.

Cea mai bună soluţie din cele analizate este de a calcula numărul de intersecţii de sinusoide în fiecare careu elementar corespunzător locaţiilor matricii de acumulare, dar timpul de calcul este în acest caz proporţional cu $N(N-1)/2$, unde N este numărul de puncte de muchie din imaginea iniţială. Maximele locale obţinute astfel sunt mai accentuate, iar precizia localizării lor este mai mare. Rezultatele se îmbunătăţesc dacă se îmbină această metodă cu ideea lui O'Gorman & Clowes - creşterea influenţei muchiilor accentuate prin utilizarea unui increment dependent de valoarea absolută a modulului gradientului estimat pentru fiecare pixel din imagine.

7.8. Detalii de implementare.

Aplicaţia PROImage, prezentată în capitolul 21 include mai multe implementări ale transformatei Hough şi a variantelor sale (pentru informaţii suplimentare se poate studia codul sursă din fişierul "**hou.pas**").

```
(*=======================================*)
(* Transformarea Hough, versiunea originală*)
Procedure HHough(k,m : byte);
var
   x,y,i,j,n      : integer;
   raza,teta,fi   : real;
   RMax,Hmax      : real;
   Sn,Cs          : array [0..127] of real;
begin
   for i:=0 to 127 do
   begin
      teta:=i/127.0*2*PI-PI;
      Sn[i]:=sin(teta);
      Cs[i]:=cos(teta);
```

```
   end;

   New(Hc);
   for x:=0 to 127do
      for y:=0 to 119 do
         Hc^[y,x]:=0.0;

   fi:=arctan(240/256);
   RMax:=Sqrt(Sqr(255.0)+Sqr(239.0));
   for x:=0 to 255 do
   begin
      for y:=0 to 239 do
         if Pc^[y,x]<>0 then
         begin
            for i:=0 to 127 do
            begin
               raza:=x*Cs[i]+y*Sn[i]-RMax/2.0*cos(i/127.0*2*PI-PI-fi);
               j:=Round(raza/RMax*59.0)+60;
               Hc^[j,i]:=Hc^[j,i]+1.0;
            end;
         end;
   end;

   HMax:=0.0;
   for x:=0 to 127 do
      for y:=0 to 119 do
         if Hc^[y,x]>HMax then
            HMax:=Hc^[y,x];

   if HMax<>0 then
   begin
      XOff:=((m-1)-2*((m-1) div 2))*256;
      YOff:=((m-1) div 2)*240;
      for x:=0 to 127 do
      begin
         for y:=0 to 119 do
         begin
            n:=Round(255.0*Hc^[y,x]/HMax);
            Pc^[y+y,x+x]:=n;
            Pc^[y+y,x+x+1]:=n;
            Pc^[y+y+1,x+x]:=n;
            Pc^[y+y+1,x+x+1]:=n;
         end;
      end;
   end;
   ...
end;

(*=========================================*)
(* Transformarea Hough, versiunea Duda & Hart*)
Procedure DHough(k,m : byte);
var
   x,y,i,l,n      : integer;
   raza,teta,fi   : real;
   RMax,Hmax      : real;
   Sn,Cs          : array [0..127] of real;
begin
   for i:=0 to 127 do
   begin
      teta:=i/127.0*1.5*PI-PI/2.0;
      Sn[i]:=sin(teta);
```

```
      Cs[i]:=cos(teta);
   end;

   New(Hc);
   for x:=0 to 127do
      for y:=0 to 119 do
         Hc^[y,x]:=0.0;

   RMax:=Sqrt(Sqr(255.0)+Sqr(239.0));
   for x:=0 to 255 do
   begin
      for y:=0 to 239 do
         if Pc^[y,x]<>0 then
         begin
            if x=0 then
               fi:=0.0
            else
               fi:=arctan(y/x)-PI/2.0;
            j:=Round(127.0*fi/PI/1.5+127.0/3.0);
            n:=Round(127.0*(fi+PI)/PI/1.5+127.0/3.0);
            for i:=j to n do
            begin
               raza:=x*Cs[i]+y*Sn[i];
               l:=Round(raza/RMax*119.0);
               Hc^[l,i]:=Hc^[l,i]+1.0;
            end;
         end;
   end;

   HMax:=0;
   for x:=0 to 127 do
      for y:=0 to 119 do
         if Hc^[y,x]>HMax then
            HMax:=Hc^[y,x];

   if HMax<>0 then
   begin
      XOff:=((m-1)-2*((m-1) div 2))*256;
      YOff:=((m-1) div 2)*240;
      for x:=0 to 127 do
         for y:=0 to 119 do
         begin
            n:=Round(255.0*Hc^[y,x]/HMax);
            Pc^[y+y,x+x]:=n;
            Pc^[y+y,x+x+1]:=n;
            Pc^[y+y+1,x+x]:=n;
            Pc^[y+y+1,x+x+1]:=n;
         end;
   end;
   ...
end;

(*================================================*)
(* Transformarea Hough, versiunea O'Gorman & Clove*)
Procedure GHough(k,m : byte);
var
   x,y,i,n,p,q    :  integer;
   k1,k2          : integer;
   raza,teta,fi   : real;
   RMax,HMax      : real;
   Sn,Cs          : array [0..127] of real;
```

```
begin

   New(Hc);
   New(Rc);
   New(Fc);

   for x:=0 to 255 do
      for y:=0 to 239 do
      begin
         Rc^[y,x]:=0;
         Fc^[y,x]:=0;
      end;

   for i:=0 to 127 do
   begin
      teta:=i/127.0*1.5*PI-PI/2.0;
      Sn[i]:=sin(teta);
      Cs[i]:=cos(teta);
   end;

   for x:=1 to 254 do
      for y:=1 to 238 do
      begin
         k1:=Pc^[y+1,x]-Pc^[y-1,x];
         k2:=Pc^[y,x+1]-Pc^[y,x-1];
         n:=Round(Sqrt(Sqr(k1+0.0)+Sqr(k2+0.0)));
         if n>255 then n:=255;
         Rc^[y,x]:=n;
         if k2=0 then
            fi:=pi/2
         else
            fi:=arctan(k1/k2);
         if fi<-PI/2 then
            fi:=fi+1.5*PI
         else
            if fi<PI/2 then
               fi:=fi+PI/2.0
            else
               fi:=fi-PI/2.0;
         n:=Round(fi/PI*255.0);
         Fc^[y,x]:=n;
      end;

   for x:=0 to 127 do
      for y:=0 to 119 do
         Hc^[y,x]:=0.0;

   RMax:=Sqrt(Sqr(255.0)+Sqr(239.0));
   for x:=0 to 255 do
   begin
      for y:=0 to 239 do
      if Rc^[y,x]>110 then
      begin
         if x=0 then
            fi:=PI/2
         else
            fi:=arctan(y/x);
         if Fc^[y,x]*PI/255.0<fi then
            fi:=Fc^[y,x]*PI/255.0+PI/2.0
         else
```

```
            fi:=Fc^[y,x]*PI/255.0-PI/2.0;
         for i:=0 to 127 do
         begin
            raza:=x*Cs[i]+y*Sn[i];
            n:=Round(raza/RMax*119.0);
            if n>=0 then
               Hc^[n,i]:=Hc^[n,i]+Fc^[y,x];
         end;
      end;
   end;

   HMax:=0;
   for x:=0 to 127 do
      for y:=0 to 119 do
         if Hc^[y,x]>HMax then
            HMax:=Hc^[y,x];

   if (HMax<>0) then
   begin
      XOff:=((m-1)-2*((m-1) div 2))*256;
      YOff:=((m-1) div 2)*240;
      for x:=0 to 127 do
         for y:=0 to 119 do
         begin
            n:=Round(255*Hc^[y,x]/HMax);
            Pc^[y+y,x+x]:=n;
            Pc^[y+y,x+x+1]:=n;
            Pc^[y+y+1,x+x]:=n;
            Pc^[y+y+1,x+x+1]:=n;
         end;
   end;
   ...
end;

(*=========================================*)
(* Transformarea Hough, versiunea optimizată 1*)
Procedure NHough(k,m : byte);
var
   x,y,i,j,n         : integer;
   raza,teta,fi      : real;
   HMaxim,RMaxim     : real;
   Sn,Cs,rmin,rmax   :  array [0..127] of real;
begin
   New(Hc);
   for x:=0 to 127do
      for y:=0 to 119 do
         Hc^[y,x]:=0.0;

   fi:=arctan(240/256);
   RMaxim:=Sqrt(Sqr(255.0)+Sqr(239.0));
   for i:=0 to 127 do
   begin
      teta:=i/127.0*PI;
      Sn[i]:=sin(teta);
      Cs[i]:=cos(teta);
      if teta<PI/2 then
         rmax[i]:=RMaxim*cos(fi-teta)
      else
         rmax[i]:=239.0*sin(teta);
      if teta<PI/2 then
         rmin[i]:=0.0
```

```
      else
         rmin[i]:=255.0*cos(teta);
   end;

   for x:=0 to 255 do
   begin
      for y:=0 to 239 do
         if Pc^[y,x]>0 then
         begin
            for i:=0 to 127 do
            begin
               raza:=(x*Cs[i]+y*Sn[i]-rmin[i])/(rmax[i]-rmin[i]);
               j:=119-Round(raza*119.0);
               Hc^[j,i]:=Hc^[j,i]+1.0;
            end;
         end;
   end;

   HMaxim:=0.0;
   for x:=0 to 127 do
      for y:=0 to 119 do
         if Hc^[y,x]>HMaxim then
            HMaxim:=Hc^[y,x];

   if HMaxim<>0 then
   begin
      XOff:=((m-1)-2*((m-1) div 2))*256;
      YOff:=((m-1) div 2)*240;
      for x:=0 to 127 do
      begin
         for y:=0 to 119 do
         begin
            n:=Round(255.0*Hc^[y,x]/HMaxim);
            Pc^[y+y,x+x]:=n;
            Pc^[y+y,x+x+1]:=n;
            Pc^[y+y+1,x+x]:=n;
            Pc^[y+y+1,x+x+1]:=n;
         end;
      end;
   end;
   ...
end;

(*=========================================*)
(* Transformarea Hough, versiunea optimizată 2*)
Procedure EHough(k,m : byte);
var
   x,y,i,j,n,p,n1,n2 : integer;
   raza,teta,fi      : real;
   Thr,Lmb           : real;
   HMaxim,RMaxim     : real;
   Sn,Cs,rmin,rmax   : array [0..127] of real;
   L,r,s             : longint;
begin
   New(Hc);
   New(Cc);

   Cc^[0,0]:=0;
   Cc^[0,255]:=0;
   Cc^[239,0]:=0;
   Cc^[239,255]:=0;
```

```
for x:=1 to 254 do
begin
   n:=abs(integer(Pc^[0,x-1])-Pc^[0,x+1]);
   if n>255 then n:=255;
   Cc^[0,x]:=n;
end;
for x:=1 to 254 do
begin
   n:=abs(integer(Pc^[239,x-1])-Pc^[239,x+1]);
   if n>255 then n:=255;
   Cc^[239,x]:=n;
end;
for y:=1 to 238 do
begin
   n:=abs(integer(Pc^[y-1,0])-Pc^[y+1,0]);
   if n>255 then n:=255;
   Cc^[y,0]:=n;
end;
for y:=1 to 238 do
begin
   n:=abs(integer(Pc^[y-1,255])-Pc^[y+1,255]);
   if n>255 then n:=255;
   Cc^[y,255]:=n;
end;
for x:=1 to 254 do
   for y:=1 to 238 do
   begin
      n1:=integer(Pc^[y-1,x-1])-Pc^[y-1,x+1]+Pc^[y,x-1]-
                  Pc^[y,x+1]+Pc^[y+1,x-1]-Pc^[y+1,x+1];
      n2:=integer(Pc^[y-1,x-1])-Pc^[y+1,x-1]+Pc^[y-1,x]-
                  Pc^[y+1,x]+Pc^[y-1,x+1]-Pc^[y+1,x+1];
      n1:=abs(n1);n2:=abs(n2);
      if n>n2 then n:=n1
      else n:=n2;
      if n>255 then n:=255;
      Cc^[y,x]:=n;
   end;

for i:=0 to 127 do
begin
   teta:=i/127.0*1.5*PI-PI/2.0;
   Sn[i]:=sin(teta);
   Cs[i]:=cos(teta);
end;

fi:=arctan(240/256);
RMaxim:=Sqrt(Sqr(255.0)+Sqr(239.0));
for i:=0 to 127 do
begin
   teta:=i/127.0*PI;
   Sn[i]:=sin(teta);
   Cs[i]:=cos(teta);
   if teta<PI/2 then
      rmax[i]:=RMaxim*cos(fi-teta)
   else
      rmax[i]:=239.0*sin(teta);
   if teta<PI/2 then
      rmin[i]:=0.0
   else
      rmin[i]:=255.0*cos(teta);
```

```
end;

for x:=0 to 127 do
   for y:=0 to 119 do
      Hc^[y,x]:=0.0;

New(Vc);
L:=0;
Thr:=224;
for i:=0 to 255 do
   for j:=0 to 239 do
      if Cc^[j,i]>Thr then
      begin
         Vc^[L]:=i;
         Vc^[L+1]:=j;
         L:=L+2;
      end;

Lmb:=1.0/255.0;
for r:=0 to (L div 2)-1 do
begin
   x:=Vc^[r+r];
   y:=Vc^[r+r+1];
   for s:=0 to r-1 do
   begin
      n1:=Vc^[s+s];
      n2:=Vc^[s+s+1];
      (*pentru fiecare pereche validă*)
      if y=n2 then
         fi:=PI/2
      else
         fi:=-arctan((x-n1)/(y-n2));
      if fi<0 then
         fi:=fi+PI;
      n:=round(fi/PI*127.0);
      raza:=x*Cs[n]+y*Sn[n];
      if fi>PI/2 then
         raza:=raza-rmin[n];
      raza:=raza/(rmax[n]-rmin[n]);
      j:=119-Round(raza*119.0);
      Hc^[j,n]:=Hc^[j,n]+1.0+Lmb*Cc^[y,x];
   end;
end;

HMaxim:=0;
for x:=0 to 127 do
   for y:=0 to 119 do
      if Hc^[y,x]>HMaxim then
         HMaxim:=Hc^[y,x];

if (HMaxim<>0) then
begin
   XOff:=((m-1)-2*((m-1) div 2))*256;
   YOff:=((m-1) div 2)*240;
   for x:=0 to 127 do
      for y:=0 to 119 do
         n:=Round(255*Hc^[y,x]/HMaxim);
         Pc^[y+y,x+x]:=n;
         Pc^[y+y,x+x+1]:=n;
         Pc^[y+y+1,x+x]:=n;
```

```
            Pc^[y+y+1,x+x+1]:=n;
          end;
    end;
    ...
end;
```

8. "Potrivirea" imaginilor (image matching).

8.1. Principiul metodei.

Una din cele mai importante clase de metode de localizare a obiectelor într-o imagine are la bază aşa-numita "potrivire" a conţinutului unei imagini cu un model al obiectului căutat. Aceasta este în esenţă o operaţie de căutare a minimului unei măsuri a diferenţei între imagine şi model. "Potrivirea" ideală se obţine arareori, datorită prezenţei zgomotelor, efectelor digitizării imaginii, etc.

Dacă se notează cu $\boldsymbol{f}(m,n)$ imaginea digitală, unde $-M \le m \le M$ şi $-N \le n \le N$, iar cu $\boldsymbol{T}(j,k)$ modelul căutat, o măsură a distorsiunii dintre imagine şi model poate fi scrisă:

$$\boldsymbol{d}(m,n) = \sum_j \sum_k \left[\boldsymbol{f}(j,k) - \boldsymbol{T}(j-m,k-n)\right]^2 \tag{1}$$

"Potrivirea" se obţine pentru valoarea minimă a expresiei anterioare sau când $d(m,n)$ scade sub un anumit prag. Relaţia anterioară se poate simplifica mult notând:

$$\boldsymbol{d}(m,n) = \boldsymbol{d}_1(m,n) - 2\,\boldsymbol{d}_2(m,n) + \boldsymbol{d}_3(m,n) \tag{2}$$

unde

$$\boldsymbol{d}_1(m,n) = \sum_j \sum_k [\,\boldsymbol{f}(j,k)]^2 \tag{3}$$

$$\boldsymbol{d}_2(m,n) = \sum_j \sum_k \boldsymbol{f}(j,k)\boldsymbol{T}(j-m,k-n) \tag{4}$$

$$\boldsymbol{d}_3(m,n) = \sum_j \sum_k [\,\boldsymbol{T}(j-m,k-n)]^2 \tag{5}$$

Termenul $\boldsymbol{d}_3(m,n)$ este constant şi independent de coordonatele (m,n). Energia imaginii din fereastra de "potrivire" este reprezentată de $\boldsymbol{d}_1(m,n)$, în timp ce $\boldsymbol{d}_2(m,n)$ este chiar corelaţia între imagine şi model, în fereastra curentă. Deoarece condiţia de minim poate fi atinsă pentru valori mari ale corelaţiei $\boldsymbol{d}_2(m,n)$ şi ale energiei imaginii, $\boldsymbol{d}_1(m,n)$, chiar în condiţii de "nepotrivire", se utilizează intercorelaţia normalizată drept măsură a "potrivirii", adică:

$$\tilde{\boldsymbol{R}}_{FT}(m,n) = \frac{\boldsymbol{d}_2(m,n)}{\boldsymbol{d}_1(m,n)} \tag{6}$$

Decizia de "potrivire" se ia dacă este îndeplinită condiția:

$$\widetilde{\boldsymbol{R}}_{FT}(m,n) > \boldsymbol{L}_R(m,n) \tag{7}$$

8.2. Filtrul de "potrivire".

Operația de găsire a extremului unei măsuri a similitudinii între imagine şi model poate fi privită ca o *filtrare* a imaginii. Filtrul de "potrivire" furnizează la ieşire o măsură a acestei similitudini, care poate fi chiar intercorelația normalizată.

Fie imaginea $\boldsymbol{g}(x,y)$, obținută din imaginea inițială $\boldsymbol{f}(x,y)$ afectată de zgomotul $\boldsymbol{n}(x,y)$.

$$\boldsymbol{g}(x,y) = \boldsymbol{f}(x,y) + \boldsymbol{n}(x,y) \tag{8}$$

Operația de filtrare furnizează la ieşire:

$$\boldsymbol{g}_0(x,y) = \boldsymbol{g}(x,y) * \boldsymbol{h}(x,y) \tag{9}$$

unde $\boldsymbol{h}(x,y)$ este răspunsul la impuls al filtrului de "potrivire".

Corespunzător intercorelației normalizate, în acest caz avem raportul semnal-zgomot:

$$\frac{|\boldsymbol{S}(x,y)|^2}{N} \tag{10}$$

unde $|\boldsymbol{S}(x,y)|^2 = |\boldsymbol{g}(x,y) * \boldsymbol{h}(x,y)|^2$ este energia instantanee la ieşirea filtrului, iar

N este energia zgomotului la ieşirea filtrului.

Notând cu $\boldsymbol{W}_N(\omega_x,\omega_y)$ densitatea spectrală de putere a zgomotului, rezultă:

$$\frac{|\boldsymbol{S}(x,y)|^2}{N} = \frac{\left|\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \boldsymbol{G}(\omega_x,\omega_y)\boldsymbol{H}(\omega_x,\omega_y)\boldsymbol{exp}(jx\omega_x + jy\omega_y)d\omega_x d\omega_y\right|}{\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \boldsymbol{W}_N(\omega_x,\omega_y)|\boldsymbol{H}(\omega_x,\omega_y)|^2 d\omega_x d\omega_y} \tag{11}$$

Expresia de mai sus este maximizată pentru:

$$\boldsymbol{H}(\omega_x,\omega_y) = \frac{\boldsymbol{G}^*(\omega_x,\omega_y)\boldsymbol{exp}(-jx\omega_x - jy\omega_y)}{\boldsymbol{W}_N(\omega_x,\omega_y)} \tag{12}$$

Pentru zgomot alb, numitorul expresiei de mai sus este o constantă. Pentru imagini modelate stochastic, soluția căutată este dată de:

$$\boldsymbol{H}(\omega_x,\omega_y) = \frac{\boldsymbol{G}^*(\omega_x,\omega_y)\boldsymbol{exp}(-jx\omega_x - jy\omega_y)}{\boldsymbol{W}_F(\omega_x,\omega_y) + \boldsymbol{W}_N(\omega_x,\omega_y)} \tag{13}$$

Pentru cazul discret, fie $\boldsymbol{g}$ imaginea observată, formată prin suprapunerea zgomotului $\boldsymbol{n}$ peste imaginea ideală $\boldsymbol{f}$:

$$\boldsymbol{g} = \boldsymbol{f} + \boldsymbol{n} \tag{14}$$

Dacă **m** este matricea filtrului căutat, în urma acestei filtrări se obține:

$$\widetilde{\boldsymbol{g}} = \boldsymbol{m}^T(\boldsymbol{f} + \boldsymbol{n}) \tag{15}$$

Se utilizează notațiile:

$S = (\boldsymbol{m}^T \boldsymbol{f})^2$ - energia imaginii ideale (fără zgomot), şi

$N = \boldsymbol{E}\{(\boldsymbol{m}^T\boldsymbol{n})(\boldsymbol{m}^T\boldsymbol{n})^T\} = \boldsymbol{m}^T\boldsymbol{K}_n\boldsymbol{m}$, energia zgomotului trecut prin filtrul $\boldsymbol{m}$, unde $\boldsymbol{K}_n$ este matricea de covarianță atașată zgomotului.

Raportul semnal zgomot este dat de:

$$\frac{S}{N} = \frac{(\boldsymbol{m}^T\boldsymbol{f})^2}{\boldsymbol{m}^T\boldsymbol{K}_n\boldsymbol{m}} \tag{16}$$

Proiectarea filtrului se face în condiția maximizării raportului semnal-zgomot, ceea ce duce la:

$$\boldsymbol{m} = \boldsymbol{K}_n^{-1} \boldsymbol{f} \tag{17}$$

iar pentru semnale stochastice:

$$\boldsymbol{m} = (\boldsymbol{K}_f + \boldsymbol{K}_n)^{-1} \bar{\boldsymbol{f}} \tag{18}$$

8.3. Înregistrarea imaginilor translate (image registration).

O clasă largă de aplicații ale "potrivirii" imaginilor se referă la realizarea unei corespondențe la nivel de pixel în cazul unor imagini provenite de la doi senzori diferiți, imagini ale aceleaşi scene, însă translate, rotite una față de alta sau care prezintă unele diferențe de scară. Operația se numeşte înregistrarea imaginilor. În literatură se mai citează cazul înregistrării imaginilor unor scene văzute din perspective diferite.

Cazul înregistrării imaginilor *translate* relativ una față de alta este cel mai des întâlnit. Fie două imagini, $\boldsymbol{f}_1(j,k)$ şi $\boldsymbol{f}_2(j,k)$, unde $1 \le j \le J$ şi $1 \le k \le K$. O măsură a gradului de potrivire a celor două imagini este funcția de intercorelație dată de:

$$\boldsymbol{R}(m,n) = \frac{\sum_j \sum_k \boldsymbol{f}_1(j,k)\, \boldsymbol{f}_2(j-m+(M+1)/2, k-n+(N+1)/2)}{\left[\sum_j \sum_k [\boldsymbol{f}_1(j,k)]^2\right]^{1/2} \left[\sum_j \sum_k [\boldsymbol{f}_2(j-m+(M+1)/2, k-n+(N+1)/)]^2\right]^{1/2}} \tag{19}$$

Studiind geometria operației de "potrivire", deci de calcul a funcției de intercorelație pe baza schemei următoare:

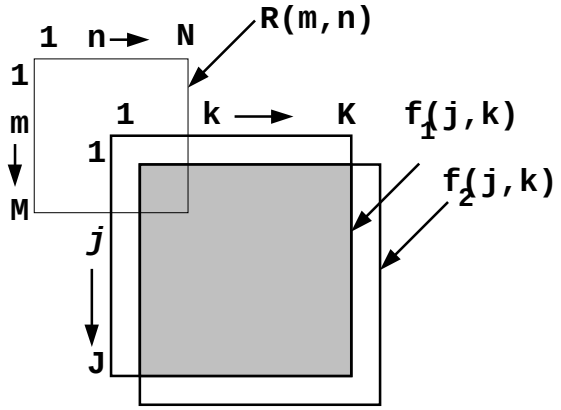


Fig. 1. Geometria operației de "potrivire".

rezultă limitele de sumare pentru expresia lui R(m,n):

$$\boldsymbol{max}\,[1, m-(M-1)/2] \le j \le MIN[J, J+m-(M+1)/2] \tag{20}$$

$$\boldsymbol{max}\,[1, n-(N-1)/2] \le k \le MIN[K, K+n-(N+1)/2] \tag{21}$$

Dezavantajele modului de operare descris principial până acum constau în alura maximumului funcției de intercorelație $\boldsymbol{R}(m,n)$, care este de obicei plat, ceea ce face problematică detecția acestui maxim, mai ales în prezența zgomotelor care afectează ambele imagini. De aceea se utilizează funcția de intercorelație statistică, având o expresie asemănătoare celei precedente:

$$\boldsymbol{R}_S(m,n)=\frac{\sum_j\sum_k \boldsymbol{g}_1(j,k)\,\boldsymbol{g}_2(j-m+(M+1)/2,k-n+(N+1)/2)}{\left[\sum_j\sum_k[\,\boldsymbol{g}_1(j,k)]^2\right]^{1/2}\left[\sum_j\sum_k[\,\boldsymbol{g}_2(j-m+(M+1)/2,k-n+(N+1)/)]^2\right]^{1/2}} \tag{22}$$

unde $\boldsymbol{g}_1(j,k)$ şi $\boldsymbol{g}_2(j,k)$ se obţin din imaginile $\boldsymbol{f}_1(j,k)$ şi $\boldsymbol{f}_2(j,k)$ prin filtrările:

$$\boldsymbol{g}_1(j,k)=[\,\boldsymbol{f}_1(j,k)-\bar{\boldsymbol{f}}_1(j,k)]*\boldsymbol{d}_1(j,k) \tag{23}$$

$$\boldsymbol{g}_2(j,k)=[\,\boldsymbol{f}_2(j,k)-\bar{\boldsymbol{f}}_2(j,k)]*\boldsymbol{d}_2(j,k) \tag{24}$$

S-au notat cu $\bar{\boldsymbol{f}}_1(j,k)$ şi $\bar{\boldsymbol{f}}_2(j,k)$ mediile spaţiale ale imaginilor în fereastra de corelaţie.

Filtrele sunt descrise de răspunsul la impuls:

$$\boldsymbol{d}_i=\frac{1}{(1+\rho^2)^2}\begin{bmatrix}\rho^2 & -\rho(1+\rho^2) & \rho^2\\ -\rho(1+\rho^2) & (1+\rho^2)^2 & -\rho(1+\rho^2)\\ \rho^2 & -\rho(1+\rho^2) & \rho^2\end{bmatrix} \tag{25}$$

unde ρ este o măsură a corelaţiei existente între pixelii adiacenţi.

Această metodă permite accentuarea semnificativă a maximelor funcţiei de intercorelaţie, pentru valori apropiate de 1 ale parametrului ρ.

Volumul extrem de mare de calcule necesare pentru obţinerea funcţiei de intercorelaţie poate fi diminuat aplicând unele strategii, descrise sumar în continuare.

Astfel Rosenfeld şi Vandenburg [97],[109] au propus reducerea într-o primă fază a rezoluţiei imaginilor ce se compară, fie prin subeşantionare, fie prin simpla selecţie a unui subset de pixeli. Determinarea, folosind aceste imagini a maximumului funcţiei de intercorelaţie, duce la o localizare grosieră a lui. Aceasta permite restrângerea semnificativă a dimensiunilor ferestrei de corelaţie pentru calculul functiei de autocorelaţie a imaginilor iniţiale, deci reducerea volumului de calcul.

Altă metodă, descrisă de Barnea şi Silverman [11] propune calculul cumulativ al funcţiei de eroare pentru cele două imagini translate, oprindu-se calculul intercorelaţiei dacă eroarea calculată, la un anumit pas, depăşeşte un anumit prag.

Metodele descrise până acum pot fi extinse pentru compararea imaginilor rotite una faţă de alta, folosind în loc de $\boldsymbol{f}_2(j,k)$ funcţia $\boldsymbol{f}_2(j,k;\theta)$ care reprezintă replicile rotite/translate ale celei de-a doua imagini. Generalizarea metodelor de mai sus constă în înlocuirea funcţiilor de imagine cu trăsături extrase din imagini, cum ar fi momentele invariante, coeficienţii unor transformări ortogonale, etc.

8.4. "Potrivirea" imaginilor şi estimatorul de mişcare MPEG.

Probabil că cea mai răspândită implementare a "potrivirii" de imagini este cea care se găseşte deja implementată (prin software sau hardware) pe toate calculatoarele personale: este vorba de codec-ul MPEG, care permite vizualizarea secvenţelor video.

Codec-ul MPEG (codor şi decodor) realizează compresia/decompresia secvenţelor de imagini în timp real. MPEG (Motion Pictures Expert Group) nu este singurul algoritm de acest tip. Există o multitudine de standarde de compresie a secvenţelor video, cum ar fi CCITT H.261, MPEG-1, MPEG-2, MPEG4.

Pentru a realiza o rată ridicată de compresie şi a menţine în acelaşi timp o calitate satisfăcătoare a secvenţei de imagini la decodare, se utilizează tehnici de codare ***intra-imagine*** şi ***inter-imagini*** (C-Cube[27]), (Jain[56])

Codarea ***intra-imagine*** este foarte apropiată principial de codarea JPEG: imaginea este împărţită în blocuri de 8x8 pixeli, fiecărui asemenea bloc i se aplică transformarea cosinus discretă. Din matricea 8x8 a transformarii se obţine un vector în urma ordonării Zig-Zag.

Urmează apoi cuantizarea vectorului ZZ şi în final o codare tip Huffman sau aritmetică. Acest tip de codate elimină doar redundanţa spaţială a secvenţei de imagini.

O secvenţă de imagini se caracterizează însă şi printr-o redundanţă temporală: conţinuturile imaginilor succesive din secvenţă sunt adesea foarte apropiate. Diferenţele care apar sunt datorate în cea mai mare parte mişcării elementelor ce compun imaginea în cadru. Plecând de la această observaţie se presupune că imaginea curentă poate fi modelată ca fiind o translaţie a imaginii precedente.

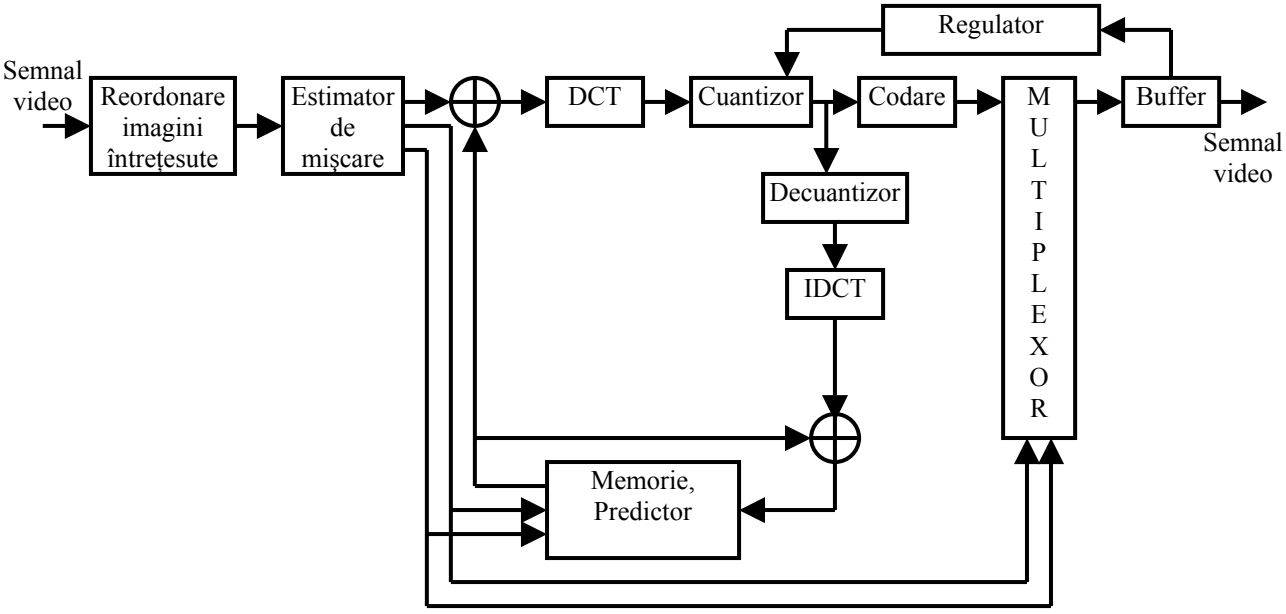


Fig.2. Schema bloc codor MPEG-1.

Practic, imaginea se descompune în blocuri 8x8(CCITT) sau 16x16(MPEG-1) şi ceea ce se codează este, pentru fiecare bloc în parte, diferenţa între blocul curent şi un bloc din imaginea precedentă (bloc de referinţă) pentru care diferenţa absolută (absolute difference - AE) este minimă:

$$\mathbf{E}(d_x, d_y) = \sum_{0}^{15}\sum_{0}^{15}\left|\mathbf{f}(i,j) - \mathbf{g}(i-d_x, j-d_y)\right| \qquad (26)$$

Localizarea acestui bloc de referinţă se face deci prin metoda care face subiectul acestui capitol, adică prin "potrivire de imagini".

Odată determinat, blocul de referinţă este reprezentat intern de un vector conţinând coordonatele carteziene.

Desi conceptual foarte simplă, metoda necesită un număr foarte mare de operaţii elementare. Practic, pentru fiecare pixel din imagine se calculează diferenţa absolută iar toate calculele trebuiesc terminate într-un interval de timp dat de perioada de cadre (intervalul de timp dintre două imagini succesive) pentru semnalul video, adică 20msec pentru PAL-SECAM.

Pentru a micşora numărul de operaţii, standardul MPEG-1 foloseşte algoritmul TSS (Three-Step-Search = căutare în trei paşi). Algoritmul evaluează mai întâi diferenţa absolută în centrul zonei de căutare şi în alte 8 locaţii din zona de căutare aflate în jurul centrului. Locaţia care furnizează diferenţa absolută minimă devine centrul zonei de căutare pentru pasul al doilea, care de această dată are dimensiunea redusă la jumătate. Căutarea se repetă de trei ori.

9. Extragerea trăsăturilor imagistice.

9.1. Tipuri de trăsături şi evaluarea lor.

O multitudine de trăsături pot fi extrase din imaginile de diferite clase. Aceste trăsături urmează a fi folosite în faza de recunoaştere. În prezent nu există încă o teorie generală asupra modului de selecţie a celor mai semnificative trăsături. Criteriile de selecţie a lor se bazează pe importanţa lor în caracterizarea formei, pe performanţele asigurate în recunoaştere şi nu în ultimul rând pe costul acestora (viteză de calcul, memorie ocupată, etc.).

Există o strânsă interdependenţă între algoritmii de segmentare şi cei de extragere a trăsăturilor. Segmentarea în sine poate fi văzută ca o extragere de trăsături, în măsura în care pixelii zonelor de interes obtinuţi prin segmentare prezintă drept trăsături proprietăţile impuse de operaţia de segmentare.

Cele mai simple trăsături care pot fi extrase dintr-o imagine sunt cele de amplitudine, obţinute din studiul histogramei şi a matricii de coocurenţă.

Utilizând o transformare ortogonală (Karhunen-Loeve discretă, Fourier Discretă, Cosinus Discretă, Hadamard, etc) şi bazându-ne pe proprietăţile de decorelare ale acestor transformări, se pot folosi drept trăsături coeficienţii acestor transformări.

Recunoaşterea imagistică se referă cel mai des la recunoaşterea obiectelor şi regiunilor care apar într-o imagine. Obiectele pot fi descrise fie prin contur, fie prin schelet, a căror reprezentare a fost abordată deja în paragraful 6.3.3. Caracterizările cele mai directe ale regiunilor

au la bază transformarea zonelor bidimensionale de imagine în vectori care se codează corespunzător.

Des folosite sunt reprezentările care au la bază arbori cuadratici, pentru care există dezvoltate o multitudine de algoritmi de parcurgere, căutare, stocare.

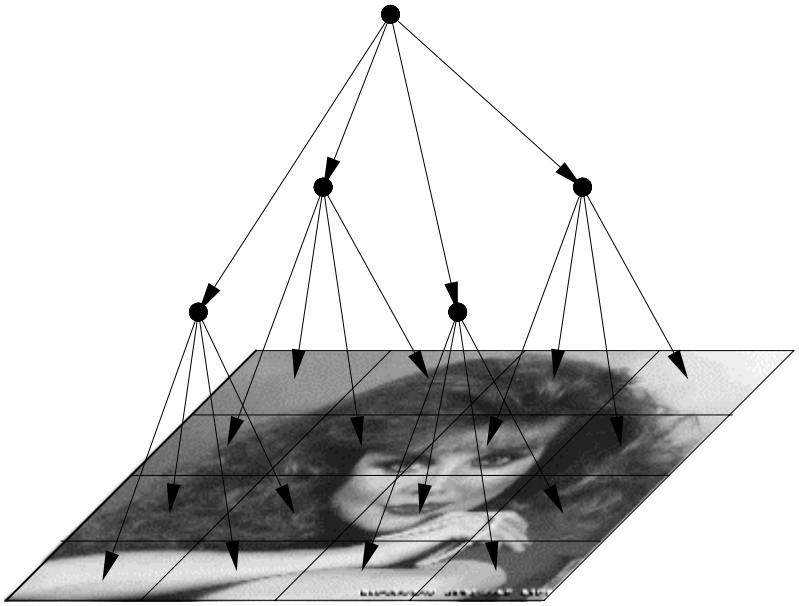


Fig. 1. Arbori cuadratici pentru reprezentarea imaginilor.

De asemenea, această reprezentare permite nu numai compresia imaginilor ci şi extragerea facilă a unor trăsături.

Trăsăturile care se extrag în vederea caracterizării obiectelor şi regiunilor sunt regenerative sau neregenerative, ultimele putând fi caracterizate drept simple măsurători asupra imaginilor.

O întreagă clasă de trăsături mult folosite la analiza obiectelor şi regiunilor are la bază momentele. Teoria matematică a momentelor permite descrierea regenerativă a elementelor de imagine, compararea prin "potrivirea" momentelor, etc. Există momente ortogonale sau nu, invariante sau nu la translație, rotație, scalare.

Foarte des folosite sunt trăsăturile geometrice (neregenerative), care se utilizează mult, mai ales datorită simplității lor.

Dacă avem de analizat o imagine de clasă **(3)**, deci formată din linii şi curbe continue formând contururi închise şi/sau grafuri, o caracterizare a lor se poate face fie prin metode ce țin de segmentarea curbelor (vezi paragrafele 6.4.), fie utilizând descriptorii Fourier.

Trăsăturile structurale/morfologice încearcă să realizeze o descriere a obiectelor pe baza unui set de elemente de structură, numite primitive. Adăugând la acestea o sintaxă care să descrie prin reguli interdependențele între primitive, se obține o descriere sintactică.

Evaluarea trăsăturilor are la bază cel mai adesea utilizarea unei măsuri între diferitele seturi de trăsături extrase pentru aceleaşi clase de obiecte (Pratt[84]). Cel mai des se utilizează în acest scop distanța Bhattacharyya (sau B-distanța) care se defineşte, pentru vectorul de trăsături $\boldsymbol{x}$ şi perechea de clase S_1 şi S_2, prin:

$$\boldsymbol{B}(S_1,S_2) = -\boldsymbol{ln}\left\{\int [\, \boldsymbol{p}(x|S_1)\, \boldsymbol{p}(x|S_2)]^{1/2} dx\right\} \qquad (1)$$

unde cu $p(x|S_1)$ s-au notat probabilitățile condiționate.

Deoarece cel mai adesea densitățile de probabilitate ale trăsăturilor extrase respectă legea normală, în acest caz B-distanța se scrie:

$$\boldsymbol{B}(S_1,S_2)=\frac{1}{8}(u_1-u_2)^T\left[\frac{\Sigma_1+\Sigma_2}{2}\right]^{-1}(u_1-u_2)+\frac{1}{2}\boldsymbol{ln}\left\{\frac{\left|\frac{1}{2}(\Sigma_1+\Sigma_2)\right|}{|\Sigma_1|^{1/2}+|\Sigma_2|^{1/2}}\right\} \tag{2}$$

unde $\boldsymbol{u}_i$ şi Σ_i reprezintă vectorul mediu de trăsături şi respectiv matricea de covarianță a trăsăturilor ce caracterizează formele din clasa "*i*".

Nu în ultimul rând trebuie menționat că, pentru fiecare trăsătură trebuiesc specificate proprietățile de invarianță pe care aceasta le prezintă, invarianțe care trebuiesc respectate de toate trăsăturile aplicate la intrarea clasificatorului, în concordanță cu cerințele aplicației.

9.2. Trăsături de amplitudine.

După cum s-a menționat deja, cele mai simple trăsături care se pot extrage dintr-o imagine sunt trăsăturile de amplitudine. Orice imagine se poate caracteriza prin media nivelelor de gri ale pixelilor aferenți:

$$\boldsymbol{M}(x,y)=\frac{1}{(2w+1)^2}\sum_{i=-w}^{w}\sum_{j=-w}^{w}\boldsymbol{f}(x+i,y+j) \tag{3}$$

precum şi prin deviația lor standard, numită şi dispersia imaginii:

$$\boldsymbol{\delta}^2(x,y)=\frac{1}{(2w+1)^2}\sum_{i=-w}^{w}\sum_{j=-w}^{w}[\,\boldsymbol{f}(x+i,y+j)-m(x+i,y+j)]^2 \tag{4}$$

Valorile $\boldsymbol{M}(x,y)$ şi $\boldsymbol{\delta}^2(x,y)$ se pot calcula fie pentru întreaga imagine, fie pentru anumite vecinătăți, eventual ataşate unor anumiți pixeli din imagine. În formulele anterioare, s-a notat latura vecinătății cu $2w+1$.

9.2.1. Trăsături ale histogramei.

Alte caracterizări extrem de simple ale imaginilor se pot obține direct din analiza histogramei imaginii, care poate fi asimilată cu o statistică de ordinul unu asociată nivelelor de gri ale pixelilor din imagine.

Alura histogramei furnizează foarte multe informații privind structura imaginii. Astfel, o histogramă îngustă indică o imagine cu contrast scăzut, o histogramă îngustă şi "deplasată" către stânga corespunde unei imagini având şi luminozitate scăzută, iar o histogramă bimodală sugerează prezența unor obiecte cu un anume nivel de gri preponderent, pe un fond de un alt nivel de gri.

În continuare sunt enumerate principalele trăsături care se pot extrage din vectorul histogramă:

Media: $$S_M=\sum_{z=0}^{L-1}z\,\boldsymbol{H}(z)\equiv\bar{z} \tag{5}$$

Dispersia: $$S_D=\sqrt{\left[\sum_{z=0}^{L-1}(z-\bar{z})^2\right]}\equiv\sigma_z \tag{6}$$

(Skewness): $$S_S=\frac{1}{\sigma_z^3}\sum_{z=0}^{L-1}(z-\bar{z})^3\,\boldsymbol{H}(z) \tag{7}$$

(Kurtosis): $$S_K=\frac{1}{\sigma_z^4}\sum_{z=0}^{L-1}(z-\bar{z})^4\,\boldsymbol{H}(z)-3 \tag{8}$$

Energia: $$S_N=\sum_{z=0}^{L-1}[\,\boldsymbol{H}(z)]^2 \tag{9}$$

Entropia: $$S_E = -\sum_{z=0}^{L-1} \boldsymbol{H}(z)\,\boldsymbol{log}_2[\,\boldsymbol{H}(z)] \tag{10}$$

Toate aceste trăsături sunt în esență estimări de ordin statistic efectuate însă nu asupra imaginii, ci doar asupra vectorului histogramă. Se folosesc destul de mult în analiza texturilor.

9.2.2. Trăsături ale matricii de coocurență.

Altă caracterizare a imaginii are la bază matricea de coocurență, care este în esență o estimare statistică de ordinul doi. Dacă se notează cu $\boldsymbol{P}(a,b)$ probabilitatea (estimată din analiza imaginii) ca pixelul cu coordonatele (j,k) să aibă nivelul de gri "a" şi pixelul cu coordonatele (m,n) să aibă valoarea "b", deci:

$$\boldsymbol{P}(a,b) = \boldsymbol{P}\{\,\boldsymbol{f}(j,k) = a;\boldsymbol{f}(m,n) = b\} \tag{11}$$

se pot extrage alte trăsături care să caracterizeze interdependența între perechile de pixeli din imagine:

Autocorelația: $$S_A = \sum_{a=0}^{L-1}\sum_{b=0}^{L-1} ab\;\boldsymbol{P}(a,b) \tag{12}$$

Covarianța: $$S_C = \sum_{a=0}^{L-1}\sum_{b=0}^{L-1} (a-\overline{a})(b-\overline{b})\,\boldsymbol{P}(a,b) \tag{13}$$

unde $\overline{a} = \sum_{a=0}^{L-1}\sum_{b=0}^{L-1} a\,\boldsymbol{P}(a,b)$ şi $\overline{b} = \sum_{a=0}^{L-1}\sum_{b=0}^{L-1} b\,\boldsymbol{P}(a,b)$ (14)

Inerția: $$S_I = \sum_{a=0}^{L-1}\sum_{b=0}^{L-1} (a-b)^2\,\boldsymbol{P}(a,b) \tag{15}$$

Modulul: $$S_V = \sum_{a=0}^{L-1}\sum_{b=0}^{L-1} |a-b|\,\boldsymbol{P}(a,b) \tag{16}$$

Diferența inversă: $$S_F = \sum_{a=0}^{L-1}\sum_{b=0}^{L-1} \frac{\boldsymbol{P}(a,b)}{1+(a-b)^2} \tag{17}$$

Energia: $$S_G = \sum_{a=0}^{L-1}\sum_{b=0}^{L-1} [\,\boldsymbol{P}(a,b)]^2 \tag{18}$$

Entropia: $$S_T = -\sum_{a=0}^{L-1}\sum_{b=0}^{L-1} \boldsymbol{P}(a,b)\,\boldsymbol{log}_2[\,\boldsymbol{P}(a,b)] \tag{19}$$

Ca şi în cazul histogramei, alura matricii de coocurență poate furniza multe informații referitoare la imagine. Astfel, într-o imagine cu o puternică corelație între pixeli, valorile semnificative ale matricii de coocurență se grupează de-a lungul diagonalei principale.

Trăsăturile enumerate mai sus se folosesc mult pentru caracterizarea texturilor.

9.3. Trăsături în domeniul transformatelor.

Transformarea Fourier a fost şi este foarte mult folosită în analiza imaginilor. Pentru cazul continuu, ea este descrisă de:

$$\boldsymbol{F}(\omega_x,\omega_y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \boldsymbol{f}(x,y)\,\boldsymbol{exp}(-jx\omega_x - jy\omega_y)dxdy \tag{20}$$

De remarcat faptul că există metode optice pentru realizarea acestei transformări, un senzor optic putând furniza la ieşire spectrul de putere sub forma:

$$\boldsymbol{M}(\omega_x,\omega_y) = \left|\boldsymbol{F}(\omega_x,\omega_y)\right|^2 \tag{21}$$

Aceste informaţii sunt invariante la translaţia originii lui $f(x,y)$, şi pot fi folosite ca atare. Dacă se trece $M(\omega_x,\omega_y)$ în coordonate polare, o integrare unghiulară în gama $[0,2\pi]$, produce un set de trăsături invariante la rotaţie:

$$N(\rho)=\int_0^{2\pi} M(\rho,\theta)d\theta \tag{22}$$

unde $\rho=\sqrt{\omega_x^2+\omega_y^2}$ şi $\theta=\mathbf{arctan}(\omega_y/\omega_x)$.

Oarecum similar, invarianţa la schimbarea scalei se obţine tot printr-o integrală, şi anume:

$$P(\theta)=\int_0^{+\infty} M(\rho,\theta)\rho d\rho \tag{23}$$

Alte trăsături pot fi obţinute din $M(\omega_x,\omega_y)$ prin intermediul unei integrale de suprafaţă pe domenii de diferite forme (bandă orizontală, bandă verticală, sector circular, sector unghiular), rezultând trăsăturile S_1,S_2,S_3,S_4, ca funcţii de parametrii regiunii de integrare.

$$S_1(m)=\int_{-\infty}^{+\infty}\int_{\omega_y(m)}^{\omega_y(m+1)} M(\omega_x,\omega_y)d\omega_x d\omega_y \tag{24}$$

$$S_2(m)=\int_{\omega_x(m)}^{\omega_x(m+1)}\int_{-\infty}^{+\infty} M(\omega_x,\omega_y)d\omega_x d\omega_y \tag{25}$$

$$S_3(m)=\int_{\rho(m)}^{\rho(m+1)}\int_0^{2\pi} M(\rho,\theta)d\rho d\theta \tag{26}$$

$$S_4(m)=\int_0^{+\infty}\int_{\theta(m)}^{\theta(m+1)} M(\rho,\theta)d\rho d\theta \tag{27}$$

Aceleaşi metode pot fi definite şi în cazul altor transformări ortogonale discrete sau continue.

9.3.1. Construcţia unui selector optimal de trăsături.

Un *selector optimal de trăsături* se construieşte astfel:

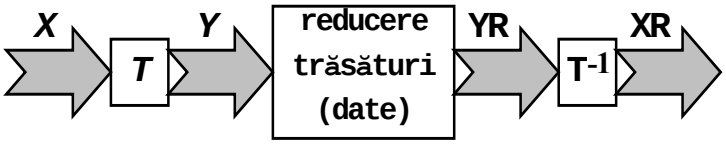


Fig. 2. Schema bloc a selectorului optimal de trăsături.

unde:

$\boldsymbol{X}=(x_1,x_2,...,x_N)$ - vectorul de intrare;

$\boldsymbol{Y}=(y_1,y_2,...,y_N)$ - vectorul transformat;

$\boldsymbol{YR}=(y_1,y_2,...,y_k,c_{k+1},c_{k+2},...,c_N)$ - vectorul de trăsături redus; el se obţine din ***YR*** înlocuind cu constante ultimele $(N-k)$ valori;

$\boldsymbol{XR}=(xr_1,xr_2,...,xr_N)$ - estimaţia vectorului de intrare.

$\boldsymbol{T} = \left\| \boldsymbol{u}_1^T, \boldsymbol{u}_2^T, ..., \boldsymbol{u}_N^T \right\|^T$, unde $\boldsymbol{u}_j^T = (u_{j0}, u_{j1}, ..., u_{jN})$ este matricea unei transformări ortogonale unitare care se va determina în condițiile minimizării erorii medii pătratice date de:

$$e = \boldsymbol{E}\{(\boldsymbol{X} - \overline{\boldsymbol{X}})^T (\boldsymbol{X} - \overline{\boldsymbol{X}})\} \tag{28}$$

Notând matricea de covarianță a intrării cu:

$$\boldsymbol{K}_{XX} = \boldsymbol{E}\{(\boldsymbol{X} - \overline{\boldsymbol{X}})(\boldsymbol{X} - \overline{\boldsymbol{X}})^T\} \tag{29}$$

unde $\overline{\boldsymbol{X}} = \boldsymbol{E}\{\boldsymbol{X}\}$, se obține transformata Karhunen-Loeve discretă (sau Hotelling) cu proprietățile:

Deoarece implementarea transformării (KL) este dificilă, modelând intrarea $\{\boldsymbol{X}\}$ cu un proces Markov de ordinul 1, cea mai bună aproximare a transformării Karhunen-Loeve discrete este DCT (Transformata Cosinus Discretă). Această modelare nu este foarte restrictivă şi se poate folosi cu succes în multe aplicații. Ahmed şi Rao [03] constată că asemenea proprietăți de decorelare a datelor prezintă multe transformări (Fourier, Hadamard, Walsh-Hadamard) dar ele sunt cel mai pregnante la Transformata Cosinus Discretă.

Pentru calculul DCT există nu numai algoritmi rapizi de calcul, ci şi circuite specializate de mare viteză. Pentru cazul bidimensional se foloseşte:

$$\boldsymbol{F}(u,v) = \frac{1}{2N} \sum_{j=0}^{N-1} \sum_{i=0}^{N-1} \boldsymbol{f}(j,k)\ \boldsymbol{cos}\left(\frac{\pi\, u\, j}{N}\right)\ \boldsymbol{cos}\left(\frac{\pi\, v\, k}{N}\right) \tag{30}$$

rezultând o matrice de aceeaşi dimensiune, cu valori reale, slab corelate. Valorile semnificative ale acestei matrici sunt plasate în colțul din stânga sus.

9.4. Momente.

Teoria generală a momentelor constituie un instrument util în analiza şi descrierea

1. $(\boldsymbol{K}_{XX} - \lambda_j \times \boldsymbol{I}_N) \times \boldsymbol{u}_j = 0$ - liniile matricii transformării sunt valorile proprii ale matricii de covarianță a domeniului de intrare;
2. $\boldsymbol{K}_{YY} = \boldsymbol{diag}(\lambda_1, \lambda_2, ..., \lambda_N)$ - transformarea furnizează la ieşire coeficienți necorelați;
3. $E = \sum_{j=k+1}^{N} \lambda_j$ - expresia erorii.

obiectelor şi regiunilor imagistice. Pentru o funcție mărginită reală $\boldsymbol{f}(x,y)$, definită pe o regiune finită *R*, momentul (regulat) de ordinul $(p+q)$ se defineşte prin:

$$m_{p,q} = \iint_R \boldsymbol{f}(x,y) x^p y^q dxdy \tag{31}$$

Funcția caracteristică a lui $\boldsymbol{f}(x,y)$ se defineşte prin conjugata transformării ei Fourier:

$$\boldsymbol{F}^*(\xi_1, \xi_2) \overset{\Delta}{=} \iint_R \boldsymbol{f}(x,y)\, \boldsymbol{exp}\{2\pi j(x\xi_1 + y\xi_2)\} dxdy \tag{32}$$

Funcția generatoare de momente pentru $\boldsymbol{f}(x,y)$ fiind definită de relația:

$$\boldsymbol{M}(\xi_1, \xi_2) \overset{\Delta}{=} \iint_R \boldsymbol{f}(x,y)\, \boldsymbol{exp}(x\xi_1 + y\xi_2) dxdy \tag{33}$$

pe baza ei se pot obține momentele de orice ordin cu expresia:

$$m_{p,q} = \left. \frac{\partial^{p+q} \boldsymbol{M}(\xi_1, \xi_2)}{\partial \xi_1^p \partial \xi_2^q} \right|_{\xi_1 = \xi_2 = 0} \tag{34}$$

Faptul că momentele sunt trăsături regenerative rezultă din *teorema de reprezentare prin momente*, care afirmă că un set infinit de momente $\{m_{p,q}; p,q = 0,1,...\}$ determină în mod unic funcția $\boldsymbol{f}(x,y)$ şi invers. Formula de reconstrucție a lui $\boldsymbol{f}(x,y)$ din momentele sale este:

$$\boldsymbol{f}(x,y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \boldsymbol{exp}[-2\pi j(x\xi_1 + y\xi_2)]\left[\sum_{p=0}^{+\infty}\sum_{q=0}^{+\infty} m_{p,q}\frac{(2\pi j)^{p+q}}{p!q!}\xi_1^p\xi_2^q\right]d\xi_1 d\xi_2 \tag{35}$$

În cazul discret momentele se calculează cu:

$$m_{pq} = \sum_x\sum_y x^p y^q\, \boldsymbol{f}(x,y) \tag{36}$$

unde $\boldsymbol{f}(x,y)$ este funcția imagine. Dacă, în plus, imaginea este binară, calculul momentelor devine separabil:

$$m_{p,q} = \sum_x x^p \sum_y y^q \tag{37}$$

9.4.1. Momente invariante.

Pe baza momentelor regulate definite anterior se construieşte *momentul centrat* de ordinul $(p+q)$:

$$\mu_{p,q} = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}(x-\bar{x})^p(y-\bar{y})^q\, \boldsymbol{f}(x,y)dxdy \tag{38}$$

iar pentru cazul discret avem:

$$\mu_{p,q} = \sum_x\sum_y (x-\bar{x})^p(y-\bar{y})^q\, \boldsymbol{f}(x,y) \tag{39}$$

Momentul centrat normalizat de ordinul $(p+q)$ se defineşte prin:

$$\eta_{p,q} = \frac{\mu_{p,q}}{\mu_{0,0}^{[(p+q)/2+1]}} \tag{40}$$

Pe baza momentelor centrate normalizate de diferite ordine au fost definite un număr de şapte momente invariante la translație, scalare şi rotație. Ele sunt date de (Gonzales[42]):

$$\begin{aligned}
\phi_1 &= \mu_{2,0} + \mu_{0,2}\\
\phi_2 &= (\mu_{2,0} + \mu_{0,2})^2 + 4\mu_{1,1}^2\\
\phi_3 &= (\mu_{3,0} - 3\mu_{1,2})^2 + (\mu_{0,3} - 3\mu_{2,1})^2\\
\phi_4 &= (\mu_{3,0} + \mu_{1,2})^2 + (\mu_{0,3} + \mu_{2,1})^2\\
\phi_5 &= (\mu_{3,0} - 3\mu_{1,2})(\mu_{3,0} + \mu_{1,2})[(\mu_{3,0} + \mu_{1,2})^2 - 3(\mu_{2,1} + \mu_{0,3})^2] +\\
&\quad + (\mu_{0,3} - 3\mu_{2,1})(\mu_{0,3} + \mu_{2,1})[(\mu_{0,3} + \mu_{2,1})^2 - 3(\mu_{1,2} + \mu_{3,0})^2]\\
\phi_6 &= (\mu_{2,0} - \mu_{0,2})[(\mu_{3,0} + \mu_{1,2})^2 - (\mu_{2,1} + \mu_{0,3})^2] + 4\mu_{1,1}(\mu_{3,0} + \mu_{1,2})(\mu_{3,0} + \mu_{2,1})\\
\phi_7 &= (3\mu_{2,1} - \mu_{0,3})(\mu_{3,0} + \mu_{1,2})[(\mu_{3,0} + \mu_{1,2})^2 - 3(\mu_{2,1} + \mu_{0,3})^2] +\\
&\quad + (\mu_{3,0} - 3\mu_{2,1})(\mu_{2,1} + \mu_{0,3})[(\mu_{0,3} + \mu_{2,1})^2 - 3(\mu_{1,2} + \mu_{3,0})^2]
\end{aligned} \tag{41}$$

De remarcat faptul că *modulul* momentului ϕ_7 prezintă invariațele sus-menționate.

9.4.2. Momente ortogonale.

Momentele regulate $m_{p,q}$ descrise anterior sunt neortogonale, deoarece x^p şi y^q sunt neortogonale (momentele $m_{p,q}$ pot fi considerate ca fiind proiecţiile lui $\boldsymbol{f}(x,y)$ pe monoamele x^p şi y^q). În locul lor pot fi folosite polinoamele Legendre care sunt ortogonale. Ele sunt definite (Jain[56]) de relaţiile:

$$\boldsymbol{P}_0(x) = 1 \tag{42}$$

$$\boldsymbol{P}_n(x) = \frac{1}{n!\,2^n}\frac{d^n}{dx^n}(x^2-1)^n \tag{43}$$

$$\int_{-1}^{+1} \boldsymbol{P}_n(x)\,\boldsymbol{P}_m(x)dx = \frac{2}{2n+1}\boldsymbol{\delta}(m-n) \tag{44}$$

Aceste polinoame ortogonale permit reprezentarea funcţiei $\boldsymbol{f}(x,y)$ prin:

$$\boldsymbol{f}(x,y) = \sum_{p=0}^{+\infty}\sum_{q=0}^{+\infty}\lambda_{p,q}\,\boldsymbol{P}_p(x)\,\boldsymbol{P}_q(y) \tag{45}$$

unde:

$$\lambda_{p,q} = \frac{(2p+1)(2q+1)}{4}\int_{-1-1}^{+1+1}\boldsymbol{f}(x,y)\,\boldsymbol{P}_p(x)\,\boldsymbol{P}_q(y)dxdy \tag{46}$$

sunt tocmai momentele ortogonale căutate. În practică, aceste momente sunt calculate prin intermediul momentelor neortogonale anterioare, folosind relaţiile:

$$\lambda_{p,q} = \frac{(2p+1)(2q+1)}{4}\sum_{j=0}^{p}\sum_{k=0}^{q}c_{p,j}c_{q,k}m_{j,k} \tag{47}$$

unde $c_{j,k}$ reprezintă coeficientul lui x^k din expresia polinomului Legendre de ordinul "*j*".

9.4.3 Momentele Zernike.

Drept variantă alternativă a momentelor descrise anterior se prezintă în continuare (Khotanzad[38],[39]) un tip special de momente ce prezintă proprietatea de invarianţă la rotaţie.

Momentele Zernike se definesc pe un domeniu egal cu interiorul cercului unitate:

$$x^2 + y^2 = 1 \tag{48}$$

Pentru construcţia lor se defineşte mai întâi un set de polinoame ortogonale, notate:

$$\{\boldsymbol{V}_{nm}(x,y)\},\ \text{cu } n-|m| \text{ par şi } |m| \le n \tag{49}$$

Expresia de definiţie a lor este:

$$\boldsymbol{V}_{nm}(x,y) = \boldsymbol{V}_{nm}(\rho,\theta) = \boldsymbol{R}_{nm}(\rho)\,\boldsymbol{exp}(jm\theta) \tag{50}$$

unde:

$$\boldsymbol{R}_{nm}(\rho) = \sum_{s=0}^{(n-|m|)/2}(-1)^s\frac{(n-s)!}{s!\left[(n+|m|)/2-s\right]!\left[(n-|m|)/2-s\right]!}\rho^{n-2s} \tag{51}$$

Se poate remarca faptul că:

$$\boldsymbol{R}_{n,-m}(\rho) = \boldsymbol{R}_{nm}(\rho) \tag{52}$$

Aceste polinoame $\{\boldsymbol{V}_{nm}(x,y)\}$ fiind ortogonale, satisfac condiţiile de ortogonalitate:

$$\iint_{x^2+y^2\le 1}\left[\boldsymbol{V}_{nm}(x,y)\right]^*\boldsymbol{V}_{pq}(x,y)\,dx\,dy = \frac{\pi}{n+1}\delta_{np}\delta_{mq} \tag{53}$$

unde:

$$\delta_{ab} = \begin{cases} 1, \text{pentru } a = b \\ 0, \text{pentru } a \neq b \end{cases}$$

Momentele Zernike sunt proiecţii ale funcţiei imagine pe această bază ortogonală. Presupunând că $\boldsymbol{f}(x,y)$ în afara cercului unitate, avem momentul Zernike de ordinul $(n+m)$:

$$\boldsymbol{A}_{nm} = \frac{n+1}{\pi} \iint\limits_{x^2+y^2 \le 1} \boldsymbol{f}(x,y) \boldsymbol{V}^*_{nm}(\rho,\theta)\, dx\, dy \tag{54}$$

Pentru cazul discret, momentul Zernike de acelaşi ordin se scrie:

$$\boldsymbol{A}_{nm} = \frac{n+1}{\pi} \sum_x \sum_y \boldsymbol{f}(x,y) \boldsymbol{V}^*_{nm}(\rho,\theta), \quad cu \quad x^2 + y^2 \le 1 \tag{55}$$

Cunoscând momentele Zernike, se poate efectua transformarea inversă, care permite reconstrucţia imaginii iniţiale cu precizie din ce în ce mai mare pe măsură ce "n" creşte:

$$\tilde{\boldsymbol{f}}(x,y) = \sum_{n=0}^{n_{max}} \sum_m \boldsymbol{A}_{nm} \boldsymbol{V}_{nm}(\rho,\theta) \tag{56}$$

Descompunând această expresie în:

$$\tilde{\boldsymbol{f}}(x,y) = \sum_{n=0}^{n_{max}} \sum_{m<0} \boldsymbol{A}_{nm} \boldsymbol{V}_{nm}(\rho,\theta) + \sum_{n=0}^{n_{max}} \sum_{m\ge 0} \boldsymbol{A}_{nm} \boldsymbol{V}_{nm}(\rho,\theta) \tag{57}$$

şi folosind o proprietate enunţată anterior, putem scrie:

$$\tilde{\boldsymbol{f}}(x,y) = \sum_{n=0}^{n_{max}} \sum_{m>0} \boldsymbol{A}_{-n,-m} \boldsymbol{V}_{-n,-m}(\rho,\theta) + \sum_{n=0}^{n_{max}} \sum_{m\ge 0} \boldsymbol{A}_{nm} \boldsymbol{V}_{nm}(\rho,\theta) \tag{58}$$

adică:

$$\tilde{\boldsymbol{f}}(x,y) = \sum_{n=0}^{n_{max}} \sum_{m>0} \boldsymbol{A}^*_{nm} \boldsymbol{V}^*_{nm}(\rho,\theta) + \sum_{n=0}^{n_{max}} \sum_{m\ge 0} \boldsymbol{A}_{nm} \boldsymbol{V}_{nm}(\rho,\theta) \tag{59}$$

Cele două sume pot fi acum unificate scriind:

$$\tilde{\boldsymbol{f}}(x,y) = \boldsymbol{A}_{n0} \boldsymbol{V}_{n0}(\rho,\theta) + \sum_{n=0}^{n_{max}} \sum_{m>0} \left[\boldsymbol{A}^*_{nm} \boldsymbol{V}^*_{nm}(\rho,\theta) + \boldsymbol{A}_{nm} \boldsymbol{V}_{nm}(\rho,\theta) \right] \tag{60}$$

Deoarece este mai uşor de operat cu numere reale vom descompune numerele complexe din formula anterioară:

$$\begin{aligned} \tilde{\boldsymbol{f}}(x,y) = {} & \left[\boldsymbol{Re}(\boldsymbol{A}_{n0}) - j\, \boldsymbol{Im}(\boldsymbol{A}_{n0}) \right] \boldsymbol{V}_{n0}(\rho,\theta) + \\ & + \sum_{n=0}^{n_{max}} \sum_{m>0} \left[\left[\boldsymbol{Re}(\boldsymbol{A}_{nm}) - j\, \boldsymbol{Im}(\boldsymbol{A}_{nm}) \right] \boldsymbol{R}_{nm}(\rho)(\boldsymbol{\cos} m\theta - j\, \boldsymbol{\sin} m\theta) + \right. \\ & \left. + \left[\boldsymbol{Re}(\boldsymbol{A}_{nm}) + j\, \boldsymbol{Im}(\boldsymbol{A}_{nm}) \right] \boldsymbol{R}_{nm}(\rho)(\boldsymbol{\cos} m\theta + j\, \boldsymbol{\sin} m\theta) \right] \end{aligned} \tag{61}$$

Efectuând calculele rezultă expresia:

$$\tilde{\boldsymbol{f}}(x,y) = \frac{1}{2} \boldsymbol{C}_{n0} \boldsymbol{R}_{n0}(x,y) + \sum_{n=0}^{n_{max}} \sum_m \left[\boldsymbol{C}_{nm} \boldsymbol{\cos} m\theta + \boldsymbol{S}_{nm} \boldsymbol{\sin} m\theta \right] \boldsymbol{R}_{nm}(x,y) \tag{62}$$

în care

$$\boldsymbol{C}_{nm} = 2\, \boldsymbol{Re}(\boldsymbol{A}_{nm}) \quad <==> \quad \boldsymbol{C}_{nm} = \frac{2n+2}{\pi} \sum_x \sum_y \boldsymbol{f}(x,y)\, \boldsymbol{R}_{nm}(x,y) \boldsymbol{\cos} m\theta \tag{63}$$

şi:

$$\boldsymbol{S}_{nm} = -2\, \boldsymbol{Im}(\boldsymbol{A}_{nm}) \quad <==> \quad \boldsymbol{S}_{nm} = \frac{-2n-2}{\pi} \sum_x \sum_y \boldsymbol{f}(x,y)\, \boldsymbol{R}_{nm}(x,y) \boldsymbol{\sin} m\theta \tag{64}$$

Se demonstrează foarte uşor că *modulul momentelor Zernike este invariant la rotaţie* şi ca urmare ele pot fi folosite drept trăsături invariante la rotaţie.

9.5. Trăsături geometrice.

Trăsăturile geometrice sunt practic măsurători geometrice care se efectuează asupra obiectelor. O parte dintre ele se calculează pe baza momentelor regulate de diferite ordine. Deşi se calculează de obicei destul de simplu, puterea lor de discriminare şi utilitatea sunt incontestabile.

Cele mai des utilizate trăsături geometrice sunt descrise în continuare.

Perimetrul unui obiect este definit de:

$$P = \int \sqrt{\boldsymbol{x}^2(t) + \boldsymbol{y}^2(t)}\,dt \tag{65}$$

El poate fi obţinut din conturul obiectului, folosind, de exemplu, codul-lanţ. Uneori în loc de perimetru se poate folosi şi lungimea liniei de contur.

Aria unui obiect este dată de:

$$A = \iint_D dxdy \tag{66}$$

Pentru cazul imaginilor discrete aria se obţine fie prin simpla numărătoare a pixelilor obiectului, fie din analiza codului-lanţ.

Razele minimă şi maximă, Rmin şi Rmax reprezintă distanţele minimă şi respectiv maximă a pixelilor obiectului faţă de centrul lui de greutate. Obţinerea lor se face limitând testele de distanţă doar la pixelii de contur exterior.

Numărul de goluri poate fi obţinut pentru fiecare obiect, după faza de extragere a contururilor, (evident, se vor extrage toate contururile, interioare şi exterioare), prin teste de incluziune. Este o trăsătură morfologică.

Numărul lui Euler este dat de:

$$E = R - G \tag{67}$$

unde R este numărul de componente conectate ale obiectului, iar *G* este numărul de goluri.

Numărul de colţuri se obţine din studiul funcţiei de curbură, date de:

$$\boldsymbol{k}(s) = \left(\frac{d^2x}{ds^2}\right)^2 + \left(\frac{d^2y}{ds^2}\right)^2 \tag{68}$$

În practică, se presupune existenţa unui colţ în acele puncte ale conturului în care curbura depăşeşte un anumit prag.

Energia curburii este definită prin:

$$E = \frac{1}{T}\int_0^S |\boldsymbol{k}(s)|^2 ds \tag{69}$$

Raportul de sveltețe (roundness, compactness) este dat de raportul între pătratul perimetrului şi arie:

$$R = \frac{P^2}{4\pi A} \tag{70}$$

Valoarea minimă a acestui raport este 1 şi se obţine pentru disc.

Următoarele trăsături geometrice se obţin pe baza momentelor regulare.

Centrul de greutate al unui obiect se obţine cu formulele:

$$\overline{x} = \frac{\sum_x \sum_y x\,\boldsymbol{f}(x,y)}{\sum_x \sum_y \boldsymbol{f}(x,y)} = \frac{m_{1,0}}{m_{0,0}} \qquad \overline{y} = \frac{\sum_x \sum_y y\,\boldsymbol{f}(x,y)}{\sum_x \sum_y \boldsymbol{f}(x,y)} = \frac{m_{0,1}}{m_{0,0}} \tag{71}$$

Orientarea indică unghiul faţă de abscisă al *momentului minim de inerţie*, moment descris de:

$$I(\theta) = \sum_x \sum_y [-(x-\bar{x})\sin\theta + (y-\bar{y})\cos\theta]^2 \tag{72}$$

Rezultă, prin minimizarea lui $I(\theta)$ în raport cu θ, orientarea:

$$\theta = \frac{1}{2}\arctan\left[\frac{2\mu_{1,1}}{\mu_{2,0}-\mu_{0,2}}\right] \tag{73}$$

Dreptunghiul de încadrare este dreptunghiul de dimensiuni minime aliniat la orientarea θ a obiectului şi care include obiectul. Laturile acestui dreptunghi pot fi descrise de ecuațiile:

$$\begin{cases} a = x\cos\theta + y\sin\theta \\ b = -x\sin\theta + y\cos\theta \end{cases} \tag{74}$$

Dreptunghiul propriu-zis se obține determinând, pentru toate punctele conturului, valorile $a_{min}, a_{max}, b_{min}, b_{max}$ care caracterizează cele patru laturi ale dreptunghiului.

Elipsa de "potrivire" (best fit ellipse) este acea elipsă ale cărei momente de inerție maxim şi respectiv minim sunt egale cu cele ale obiectului. Pentru o elipsă descrisă de:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \tag{75}$$

momentele de inerție maxim şi minim sunt date de:

$$I_{MIN} = \frac{\pi}{4}ab^3 \quad \text{şi} \qquad I_{MAX} = \frac{\pi}{4}a^3b\,, \qquad \text{pentru } a>b.$$

Calculând orientarea θ obținem momentele de inerție ale obiectului:

$$I'_{MIN} = \sum_x \sum_y [-(x-\bar{x})\sin\theta + (y-\bar{y})\cos\theta]^2 \tag{76}$$

$$I'_{MAX} = \sum_x \sum_y [(x-\bar{x})\cos\theta + (y-\bar{y})\sin\theta]^2 \tag{77}$$

Egalând corespuzător momentele de inerție rezultă elipsa descrisă de:

$$a = \left(\frac{4}{\pi}\right)^{1/4}\left[\frac{(I'_{MAX})^3}{I'_{MIN}}\right]^{1/8} \qquad a = \left(\frac{4}{\pi}\right)^{1/4}\left[\frac{(I'_{MIN})^3}{I'_{MAX}}\right]^{1/8} \tag{78}$$

Tot între trăsăturile geometrice pot fi încadrate **proiecțiile pe diferite direcții** ale obiectelor din imagine. În continuare se prezintă principial geometria unei asemenea proiecții pe o direcție dată de unghiul θ:

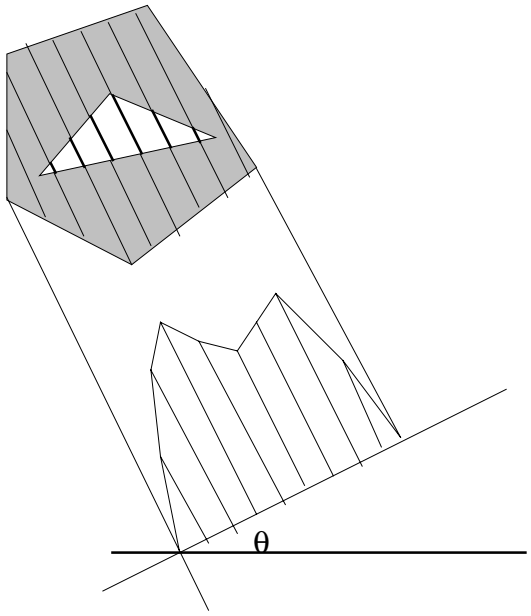


Fig. 3. Geometria proiecției pe o direcție dată.

O multitudine de alte trăsături pot fi folosite în diferite aplicații, cum ar fi ***raza medie de încadrare, orientarea axei minime şi respectiv maxime, aria dreptunghiului de încadrare, raportul între aria obiectului şi aria dreptunghiului de încadrare, raportul între perimetrul obiectului şi perimetrul dreptunghiului de încadrare, etc.***

Pentru fiecare din aceste trăsături de specifică invariațele pe care le respectă.

9.6. Trăsături ale imaginilor de clasa (3).

Imaginile de clasă **(3)** sunt formate, după cum s-a menționat deja, din linii şi curbe continue, având grosimea de un pixel. Caracterizarea lor se face fie divizând liniile/curbele în segmente pentru care se caută o descriere analitică (prin aproximare şi/sau interpolare), fie la nivel global, de exemplu prin descriptorii Fourier.

9.6.1. Segmentarea curbelor.

Operația de segmentare a curbelor poate fi asimilată unei extrageri de trăsături dacă se adoptă drept trăsături parametrii de descriere a acestor curbe. Astfel de parametri sunt constantele de definiție pentru funcțiile spline exprimate parametric, sau pozițiile colțurilor poligoanelor de definiție pentru funcțiile Bezier şi B-spline.

9.6.2. Descriptorii Fourier.

Descriptorii Fourier (Gonzales[42]), (Jain[56]), (Pratt[84]) se utilizează pentru descrierea contururilor închise ale obiectelor, contururi obținute de obicei cu algoritmi de extragere contur. Conturul este descris în această primă fază parametric, prin $\boldsymbol{x}(s)$ şi $\boldsymbol{y}(s)$, presupunând, pentru simplitate, cazul continuu.

Obținerea descriptorilor Fourier are la bază transportarea conturului descris parametric în planul complex, rezultând funcția complexă:

$$\boldsymbol{z}(s) = \boldsymbol{x}(s) + j\,\boldsymbol{y}(s) \tag{79}$$

Unghiul tangent, definit ca unghiul format de normala la vectorul de poziție al punctului curent cu tangenta la curbă este dat de:

$$\boldsymbol{\Phi}(s) = \boldsymbol{arctan}\left[\frac{d\,\boldsymbol{y}(s)/ds}{d\,\boldsymbol{x}(s)/ds}\right] \tag{80}$$

În planul complex se defineşte curbura conturului închis studiat prin:

$$\boldsymbol{k}(s) = \frac{d\,\boldsymbol{\Phi}(s)}{ds} \tag{81}$$

Cunoscând curbura şi unghiul tangent în fiecare punct al conturului, acesta se poate reconstrui cu formulele:

$$\boldsymbol{x}(s) = \boldsymbol{x}(0) + \int_0^{\pi} \boldsymbol{k}(\alpha)\,\boldsymbol{cos}\,[\,\boldsymbol{\Phi}(\alpha)]\,d\alpha \tag{82}$$

$$\boldsymbol{y}(s) = \boldsymbol{y}(0) + \int_0^{\pi} \boldsymbol{k}(\alpha)\,\boldsymbol{sin}\,[\boldsymbol{\Phi}(\alpha)]\,d\alpha \tag{83}$$

Evident, curbura $\boldsymbol{k}(s)$ este o funcție periodică de perioadă P, unde P este cel mai des lungimea perimetrului conturului. Ca urmare, $\boldsymbol{k}(s)$ poate fi dezvoltat în serii Fourier prin:

$$\boldsymbol{k}(s) = \sum_{n=-\infty}^{+\infty} c_n\,\boldsymbol{exp}\left(\frac{2\pi j}{P} n\right) \tag{84}$$

unde coeficienții c_n ai dezvoltării au expresia:

$$\boldsymbol{c}_n = \frac{1}{P}\int_0^P \boldsymbol{k}(s)\,\boldsymbol{exp}\left(\frac{-2\pi j}{P}n\right)ds \tag{85}$$

Coeficienţii $\boldsymbol{c}_n$ se numesc descriptorii Fourier ai conturului descris prin curbura $\boldsymbol{k}(s)$. Un alt set de descriptori se pot ataşa direct conturului propriu-zis $z(s)$ care este tot periodic de perioadă *P*, valorile obţinute nemaifiind însă invariante la translaţie.

În studiul curbelor închise având colţuri, funcţia curbură este însă nedefinită. Aceasta situaţie poate fi evitată înlocuind curbura cu funcţia propusă de Zahn şi Roskies [114]:

$$\boldsymbol{\theta}(s) = \int_0^s \boldsymbol{k}(\alpha)d\alpha - \frac{2\pi s}{P} \tag{86}$$

Pentru *cazul discret*, conturul apare descris de şirul coordonatelor pixelilor de contur. Similar cazului continuu, se face trecerea în planul complex prin:

$$\boldsymbol{z}(s_i) = \boldsymbol{x}(s_i) + j\,\boldsymbol{y}(s_i) \tag{87}$$

Unghiul tangent este dat de:

$$\boldsymbol{\Phi}(s_i) = \boldsymbol{arctan}\left[\frac{\boldsymbol{y}(s_i) - \boldsymbol{y}(s_{i-1})}{\boldsymbol{x}(s_i) - \boldsymbol{x}(s_{i-1})}\right] \tag{88}$$

iar curbura se obţine prin:

$$\boldsymbol{k}(s_i) = \boldsymbol{\Phi}(s_i) - \boldsymbol{\Phi}(s_{i-1}) \tag{89}$$

Cunoscând curbura pentru fiecare punct al conturului, coeficienţii transformatei Fourier discrete pot fi folosiţi drept trăsături pentru descrierea conturului. Aceste trăsături sunt invariante la translaţie.

$$\boldsymbol{z}(n) = \boldsymbol{x}(n) + j\,\boldsymbol{y}(n) \tag{90}$$

$$\boldsymbol{z}(n) = \frac{1}{N}\sum_{k=0}^{N-1}\boldsymbol{c}_k\,\boldsymbol{exp}\left(\frac{2\pi j}{N}kn\right) \tag{91}$$

$$\boldsymbol{c}_k = \sum_{n=0}^{N-1}\boldsymbol{z}(n)\,\boldsymbol{exp}\left(\frac{-2\pi j}{N}kn\right) \tag{92}$$

Se remarcă faptul că descriptorii Fourier care se obţin nu sunt invarianţi la poziţia punctului de start, rotaţie, scalare, reflexie. Descriptorii Fourier ataşaţi funcţiei de curbură sunt invarianţi la translaţia conturului.

Folosind unele proprietăţi ale descriptorilor Fourier se pot obţine din aceştia trăsături cu proprietăţile de invarianţă dorite:

Translatie:	$\tilde{z}(n) = z(n) + z_0$	==>	$\tilde{c}_k = c_k + z_0\delta_k$
Scalare:	$\tilde{z}(n) = \alpha z(n)$	==>	$\tilde{c}_k = \alpha c_k$
Punct start:	$\tilde{z}(n) = z(n - n_0)$	==>	$\tilde{c}_k = c_k\,exp(-2\pi j n_0 k/N)$
Rotatie:	$\tilde{z}(n) = z(n)\,exp(j\theta_0)$	==>	$\tilde{c}_k = c_k\,exp(j\theta_0)$
Reflexie:	$\tilde{z}(n) = z^*(n)\,exp(2j\theta) + 2\gamma$	==>	$\tilde{c}_k = c^*_{-k}\,exp(2j\theta) + 2\gamma\delta_k$

Ultima proprietate descrie cazul unui contur simetric faţă de dreapta descrisă de (ρ,θ), unde ρ este distanţa între originea axelor şi dreaptă, iar θ este unghiul între dreaptă şi abscisă.

Se poate remarca faptul că valorile $|\tilde{c}_k|$ sunt invariante la rotaţie, reflexie şi poziţia punctului de start, iar raportul $\tilde{c}_k / |\tilde{c}_k|$ este invariant la scalare, pentru orice *k*.

Având date două contururi $z_1(n)$ şi $z_2(n)$, se poate aprecia dacă ele sunt similare indiferent de poziția, mărimea şi orientarea lor, folosind distanța:

$$\boldsymbol{D}(z_0,\alpha,n_0,\theta_0)=\underset{z_0,\alpha\alpha_,0,\theta_0}{\boldsymbol{min}}\left\{\sum_{n=0}^{N-1}\left|z_1(n)-z_2(n+n_0)\,\boldsymbol{exp}(j\theta_0)-z_0\right|^2\right\} \tag{93}$$

Parametrii z_0,α,n_0,θ_0 se determină din condiția minimizării distanței $\boldsymbol{D}$ anterioare. Fie a_k şi respectiv b_k descriptorii Fourier calculați pentru cele două contururi. Se folosesc notațiile:

$$a_k b_k^* = c_k\,\boldsymbol{exp}(j\psi_k),\quad \phi=-2\pi n_0/N \tag{94}$$

Dacă $z_1(n)$ şi $z_2(n)$ se normalizează astfel încât:

$$\sum_{n=0}^{N-1} z_1(n)=0 \quad \text{şi} \quad \sum_{n=0}^{N-1} z_2(n)=0 \tag{95}$$

atunci distanța $\boldsymbol{D}(z_0,\alpha,n_0,\theta_0)$ este minimă pentru:

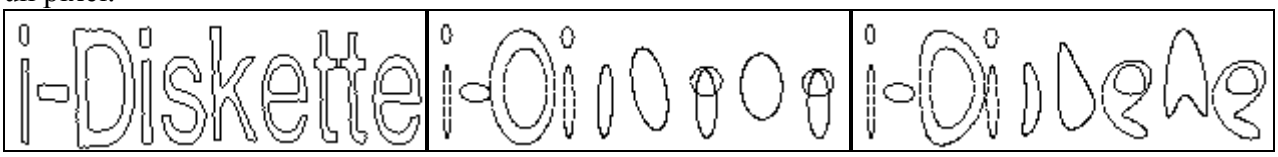
$$z_0=0 \qquad \alpha=\frac{\sum_{k=0}^{N-1}\boldsymbol{c_k}\,\boldsymbol{cos}(\psi_k+k\varphi+\theta_0)}{\sum_{k=0}^{N-1}|b_k|^2} \qquad \theta_0=\boldsymbol{arctan}\left[-\frac{\sum_{k=0}^{N-1}c_k\,\boldsymbol{sin}(\psi_k+k\varphi)}{\sum_{k=0}^{N-1}c_k\,\boldsymbol{cos}(\psi_k+k\varphi)}\right] \tag{96}$$

Înlocuind expresiile obținute rezultă:

$$\boldsymbol{D}=\underset{\varphi}{\boldsymbol{min}}\left[\sum_{k=0}^{N-1}\left|a_k-\alpha b_k\,\boldsymbol{exp}(jk\varphi+j\theta_0)\right|^2\right] \tag{97}$$

Evaluând expresia din paranteza pătrată pentru fiecare din cele *N* valori posibile pentru ***f***, se poate deduce valoarea minimă căutată pentru distanța ***D***.

Utilizarea descriptorilor Fourier poate fi extinsă la descrierea liniilor şi curbelor oarecare din orice imagine de clasă **(3)**, considerându-le contururi închise ale unor obiecte cu grosimea de un pixel.



Original *3 valori* *5 valori*

7 valori *9 valori* *11 valori*

13 valori *15 valori* *17 valori*

19 valori *21 valori* *23 valori*

9.7. Caracterizarea texturilor.

Schema bloc a unui sistem general de analiză a texturilor, propusă de Jain [56] este dată în continuare:

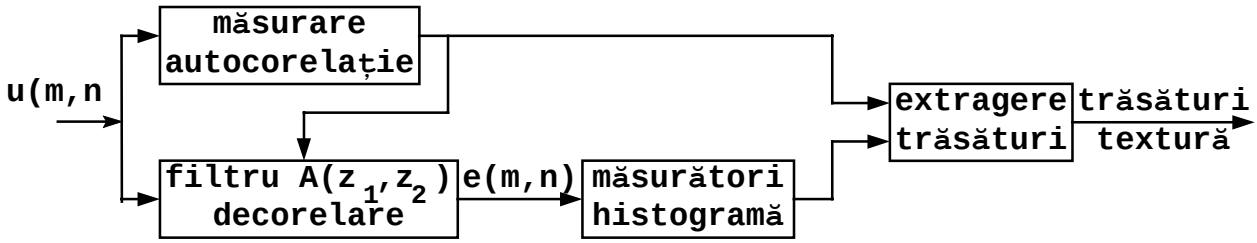


Fig. 4. Schema bloc a unui sistem general de analiză a texturilor.

Textura este mai întâi decorelată de filtrul $A(z_1,z_2)$ care poate fi proiectat cunoscând funcția de autocorelație corespunzătoare texturii. Se obține e(m,n) care poate fi descris cel mai bine de un câmp aleator necorelat, ale cărui caracteristici se deduc din studiul histogramei. Restul trăsăturilor se extrag din studiul funcției de autocorelație.

Caracterizarea texturilor se poate face printr-o multitudine de metode de analiză (Fainhurst[37]), unele deja abordate în paragraful 6.4. Bazându-ne pe dualitatea care există între segmentare şi extragere de trăsături, se pot sintetiza clasele de trăsături care pot fi extrase din imaginea unei texturi folosind următoarea diagramă (Jain[56]):

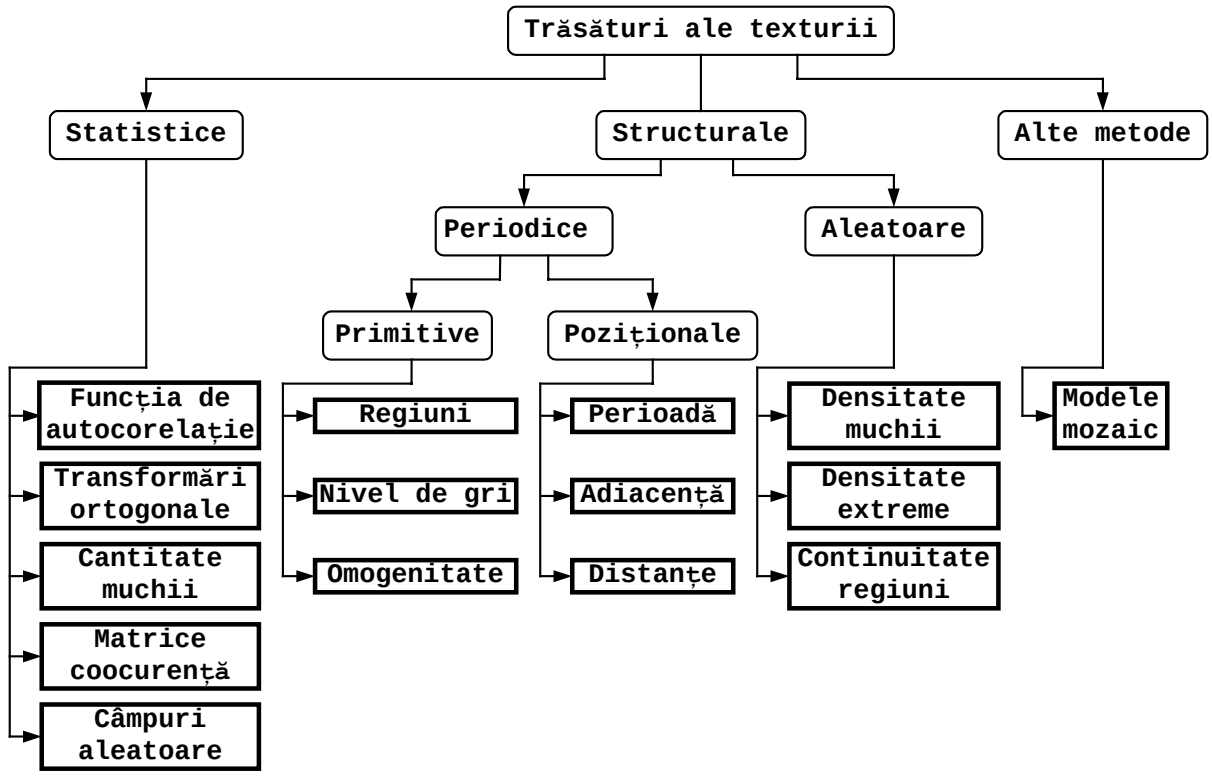


Fig. 5. Metode de analiză a texturilor.

Modelul mozaic se bazează pe câmpuri geometrice aleatoare (Cross[32]). Sunt mult utilizate pentru analiza țesăturilor, a căror imagini se apropie cel mai mult de acest model.

9.8. Trăsături structural-sintactice.

Metodele structural-sintactice de recunoaştere au la bază utilizarea primitivelor şi a relaţiilor dintre ele pentru execuţia procesului de recunoaştere (Pavlidis[80]).

Există o multitudine de primitive care pot fi extrase dintr-o imagine. Cele mai simple sunt cele folosite pentru caracterizarea imaginilor de clasă **(3)**, deci formate din linii şi curbe continue. Un asemenea set de primitive este prezentat în continuare. El permite descrierea unui contur oarecare printr-un şir de simboluri, fiecare corespunzând unei primitive.

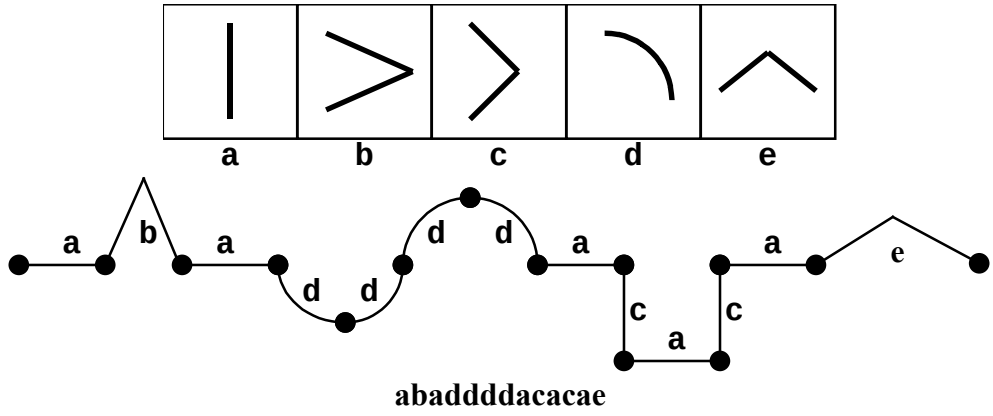


Fig. 6. Exemple de primitive generice şi utilizarea lor.

Un exemplu clasic de utilizare a descrierilor structurale în caracterizarea obiectelor este datorat lui Ledley [64] şi se referă la caracterizarea imaginilor microscopice conţinând cromozomi. Pentru această aplicaţie, primitivele folosite au fost cinci la număr, cele redate în continuare, unde marcajele romboidale indică poziţia interiorului obiectului (cromozomului):

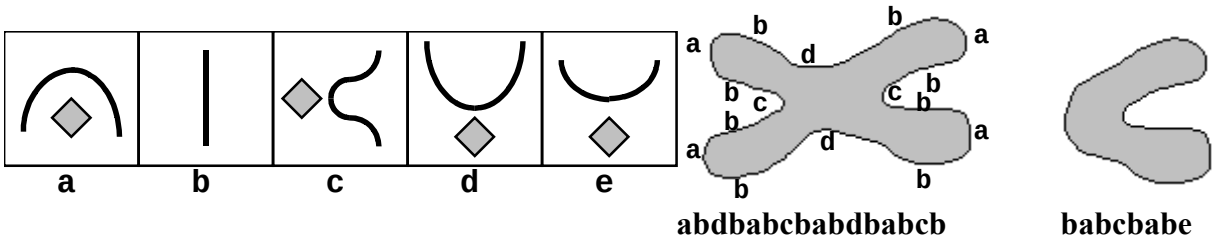


Fig. 7. Descrierea sintactică a cromozomilor.

Adăugarea la acest set de primitive a unei sintaxe care să descrie regulile de conectivitate între primitive permite obţinerea unei reprezentări sintactice unice pentru obiecte. Pentru cazul imaginilor de clasă (3) regulile de conectivitate sunt doar de tipul joncţiune între primitive, ceea ce permite descrieri simple sub formă de şiruri, ca în exemplele anterioare.

Dacă relaţiile de conectivitate sunt mai complexe ("dreapta-jos faţă de"), ca în exemplul următor al descrierii feţei umane, reprezentarea sintactică obţinută ia forma unui graf (Ballard[10]). Se remarcă şi structura mult mai complexă a primitivelor folosite("gură", "ochi", "sprâncene", etc.).

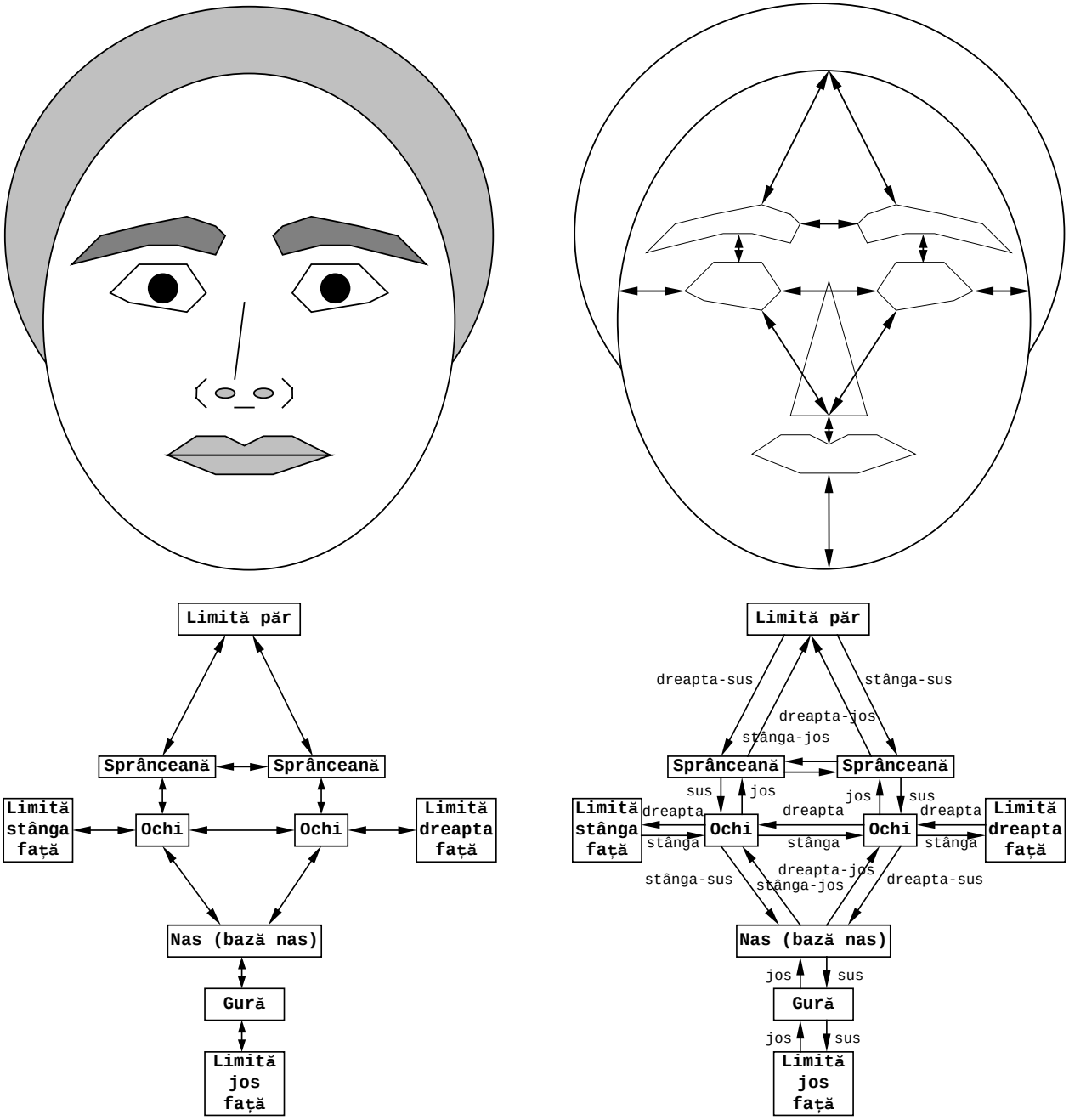


Fig. 8. Descriere sintactică pentru fața umană.

9.9. Detalii de implementare.

Aplicația PROImage, prezentată în capitolul 21 include, printre altele, şi implementarea segmentării cu prag. (pentru informații suplimentare se poate studia codul sursă din fişierul "**his.pas**").

În continuare este prezentat un program PASCAL care extrage descriptorii Fourier dintr-o imagine binară de clasă **(3)** şi apoi reconstruieşte obiectele localizate în imagine pe baza unui număr predeterminat din aceşti descriptori.

```
program Descriptori;
uses Dos,Graph,Crt;
type
   Image = array [0..239] of array [0..255] of byte;
const
```

```
var
   i,j,m,n  : integer;
   Img1,Img2: ^Image;
   Code     : byte;
   Text     : string;

(*---------------- PROCEDURA   D e s c r i p t ----------------------------*)
Procedure Descript;
type
   Conv = array[0..7] of shortint;
   vecb = array [0..0] of byte;
   vecr = array [0..0] of real;
   ivec = array [0..0] of integer;
   pvec = array [0..0] of ^vecb;
   rvec = array [0..0] of ^vecr;
const
   Cx : Conv = (-1,-1,0,1,1,1,0,-1);
   Cy : Conv = (0,-1,-1,-1,0,1,1,1);
var
   i,j,k,s,x,y,xs,ys,xc,yc,nc : integer;
   Pnc      : ^ivec;    {pointer la lista cu nr. de puncte/contur}
   Px,Py    : ^pvec;    {pointer la lista de pointeri la listele coordonate}
   Rp,Ip    : ^rvec;    {pointer la lista de pointeri la listele descriptori}
   Fi,Rt,It : real;
begin
   {inversez imagine}
   setfillstyle(SOLIDFILL,15);
   bar(256,0,511,239);
   bar(0,240,255,479);
   for x:=0 to 255 do
      for y:=0 to 239 do
         Img1^[y,x]:=255-Img1^[y,x];

   {baleiez contur, aflu nc = număr contururi}
   nc:=0;
   for x:=1 to 254 do
      for y:=1 to 238 do
      begin
         if (Img1^[y-1,x]=0) and (Img1^[y,x]=255) then
         begin
            nc:=nc+1;
            xs:=x;
            ys:=y;
            xc:=x;
            yc:=y;
            Img1^[yc,xc]:=254;
            j:=0;
            while Img1^[yc+Cy[j],xc+Cx[j]]<>0 do
               j:=(j+1) mod 8;
            repeat
               while Img1^[yc+Cy[j],xc+Cx[j]]=0 do
                  j:=(j+1) mod 8;
               xc:=xc+Cx[j];
               yc:=yc+Cy[j];
               Img1^[yc,xc]:=254;
               PutPixel(256+xc,yc,10);
               j:=(j+5) mod 8;
            until (xc=xs) and (yc=ys);
         end;
      end;
```

```
{aloc vector cu număr pixeli/contur}
GetMem(Pnc,nc*SizeOf(integer));

{completez vector Pnc}
i:=0; for x:=1 to 254 do
   for y:=1 to 238 do
   begin
      if (Img1^[y-1,x]=0) and (Img1^[y,x]>=254) then
      begin
         Pnc^[i]:=0;
         xs:=x;
         ys:=y;
         xc:=x;
         yc:=y;
         Img1^[yc,xc]:=253;
         j:=0;
         while Img1^[yc+Cy[j],xc+Cx[j]]<>0 do
            j:=(j+1) mod 8;
         repeat
            while Img1^[yc+Cy[j],xc+Cx[j]]=0 do
               j:=(j+1) mod 8;
            xc:=xc+Cx[j];
            yc:=yc+Cy[j];
            Img1^[yc,xc]:=253;
            PutPixel(256+xc,yc,5);
            Pnc^[i]:=Pnc^[i]+1;
            j:=(j+5) mod 8;
         until (xc=xs) and (yc=ys);
         i:=i+1;
      end;
   end;

GetMem(Px,nc*SizeOf(pointer));
GetMem(Py,nc*SizeOf(pointer));
GetMem(Rp,nc*SizeOf(pointer));
GetMem(Ip,nc*SizeOf(pointer));
for i:=0 to nc-1 do
begin
   GetMem(Px^[i],Pnc^[i]);
   GetMem(Py^[i],Pnc^[i]);
   GetMem(rp^[i],Pnc^[i]*SizeOf(real));
   GetMem(ip^[i],Pnc^[i]*SizeOf(real));
end;

{completez vectori Px , Py}
i:=0;
for x:=1 to 254 do
   for y:=1 to 238 do
   begin
      if (Img1^[y-1,x]=0) and (Img1^[y,x]>=253) then
      begin
         xs:=x;
         ys:=y;
         xc:=x;
         yc:=y;
         Img1^[yc,xc]:=252;
         j:=0;
         while Img1^[yc+Cy[j],xc+Cx[j]]<>0 do
            j:=(j+1) mod 8;
         k:=0;
         repeat
```

```
            while Img1^[yc+Cy[j],xc+Cx[j]]=0 do
               j:=(j+1) mod 8;
            xc:=xc+Cx[j];
            yc:=yc+Cy[j];
            Img1^[yc,xc]:=252;
            Px^[i]^[k]:=xc;
            Py^[i]^[k]:=yc;
            k:=k+1;
            j:=(j+5) mod 8;
         until (xc=xs) and (yc=ys);
         i:=i+1;
      end;
   end;

for i:=0 to nc-1 do
   for k:=0 to Pnc^[i]-1 do
      putpixel(256+Px^[i]^[k],Py^[i]^[k],0);

{calcul descriptori Fourier}
for i:=0 to nc-1 do
begin
   for j:=0 to Pnc^[i]-1 do
   begin
      Rp^[i]^[j]:=0.0;
      Ip^[i]^[j]:=0.0;
      for k:=0 to Pnc^[i]-1 do
      begin
         Fi:=-2*PI*k*j/Pnc^[i];
         Rp^[i]^[j]:=Rp^[i]^[j]+Px^[i]^[k]*cos(Fi)-Py^[i]^[k]*sin(Fi);
         Ip^[i]^[j]:=Ip^[i]^[j]+Px^[i]^[k]*sin(Fi)+Py^[i]^[k]*cos(Fi);
      end;
   end;
   ErrSound;
end;

{reducere date}
s:=11;
for i:=0 to nc-1 do
   for j:=s+1 to Pnc^[i]-1-s do
   begin
      Rp^[i]^[j]:=0.0;
      Ip^[i]^[j]:=0.0;
   end;

{transformarea inversă}
for i:=0 to nc-1 do
begin
   for j:=0 to Pnc^[i]-1 do
   begin
      Rt:=0.0;
      It:=0.0;
      for k:=0 to Pnc^[i]-1 do
      begin
         Fi:=2*PI*k*j/Pnc^[i];
         Rt:=Rt+Rp^[i]^[k]*cos(Fi)-Ip^[i]^[k]*sin(Fi);
         It:=It+Rp^[i]^[k]*sin(Fi)+Ip^[i]^[k]*cos(Fi);
      end;
      Px^[i]^[j]:=Round(Rt/Pnc^[i]);
      Py^[i]^[j]:=Round(It/Pnc^[i]);
      putpixel(Px^[i]^[j],240+Py^[i]^[j],0);
   end;
```

```
      ErrSound;
   end;

   for i:=0 to nc-1 do
   begin
      FreeMem(Px^[i],Pnc^[i]);
      FreeMem(Py^[i],Pnc^[i]);
      FreeMem(Rp^[i],Pnc^[i]*SizeOf(real));
      FreeMem(Ip^[i],Pnc^[i]*SizeOf(real));
   end;
   FreeMem(Px,nc*SizeOf(pointer));
   FreeMem(Py,nc*SizeOf(pointer));
   FreeMem(Rp,nc*SizeOf(pointer));
   FreeMem(Ip,nc*SizeOf(pointer));
end;
```

10. Clasificatori liniari.

Pentru cazul a două clase şi două trăsături avem următoarea distribuţie posibilă a formelor de intrare în spaţiul formelor:

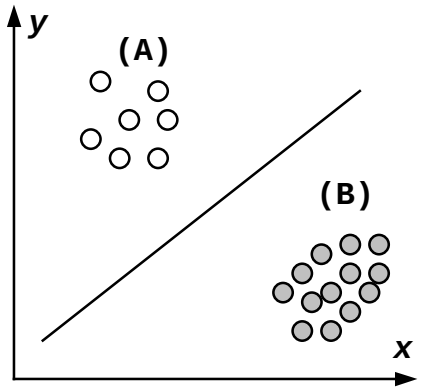


Fig. 1. Principiul clasificatorului liniar.

Limita de decizie în acest caz este de forma:

$$u = mx + n \tag{1}$$

sau echivalent:

$$a_1 x + a_2 y + a_3 = 0 \tag{2}$$

unde $a_1 = -m,\ a_2 = 1,\ a_3 = -b$

Funcţiile discriminant pentru cele două clase pot fi construite de forma:

$$\boldsymbol{D}_A(x,y) = a_1 x + a_2 y + a_3 \tag{3}$$

$$\boldsymbol{D}_B(x,y) = -a_1 x - a_2 y - a_3 \tag{4}$$

Generalizând, o funcţie discriminant liniară pentru spaţiul n-dimensional al formelor este dată de:

$$\boldsymbol{D}_i(\boldsymbol{X}) = w_{i1}x_1 + w_{i2}x_2 + \ldots + w_{in}x_n + w_{i,n+1} = \sum_{j=1}^{n} w_{ij}x_j + w_{i,n+1} \tag{5}$$

sau, cu notația $x_{n+1} = 1$, avem:

$$\boldsymbol{D}_i(\boldsymbol{X}) = \sum_{j=1}^{n+1} w_{ij}x_j \tag{6}$$

care defineşte prin $\boldsymbol{D}_i(\boldsymbol{X}) = 0$ ecuația unui hiperplan.

Matriceal, expresia de mai sus se scrie:

$$\boldsymbol{D}_i(\boldsymbol{X}) = \begin{bmatrix} w_{i1} & w_{i2} & \ldots & w_{in} & w_{i,n+1} \end{bmatrix}\boldsymbol{X}, \text{ pentru } i = 1\ldots K \tag{7}$$

Sintetizând pentru toate valorile lui "*i*", obținem:

$$\begin{bmatrix} \boldsymbol{D}_1(\boldsymbol{X}) \\ \boldsymbol{D}_2(\boldsymbol{X}) \\ \ldots \\ \boldsymbol{D}_K(\boldsymbol{X}) \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \ldots & w_{1n} & w_{1,n+1} \\ w_{21} & w_{22} & \ldots & w_{2n} & w_{2,n+1} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ w_{K1} & w_{K2} & \ldots & w_{Kn} & w_{K,n+1} \end{bmatrix}\boldsymbol{X} \Leftrightarrow \boldsymbol{D} = \boldsymbol{W}\boldsymbol{X} \tag{8}$$

Limitele de decizie dintre oricare două clase se deduc din:

$$\boldsymbol{D}_i(\boldsymbol{X}) - \boldsymbol{D}_j(\boldsymbol{X}) = 0 \Leftrightarrow \sum_{m=1}^{n+1}(w_{im} - w_{jm})x_m = 0 \tag{9}$$

Folosind notația:

$$w_m^{(ij)} = w_{im} - w_{jm} \tag{10}$$

se obține ecuația limitei de decizie între două clase de forma:

$$\sum_{m=1}^{n+1} w_m^{(ij)}x_m = 0 \tag{11}$$

care reprezintă ecuația unui hiperplan care trece prin originea spațiului extins al trăsăturilor, spațiu a cărui utilizare simplifică mult studiul clasificatorilor liniari.

Pentru cazul a doar două clase, este suficientă doar o singură funcție de decizie, $\boldsymbol{D}(\boldsymbol{X})$, astfel încât:

$$\begin{cases} \boldsymbol{D}(\boldsymbol{X}) > 0, \text{ pentru } \boldsymbol{X} \in \omega_1 \\ \boldsymbol{D}(\boldsymbol{X}) < 0, \text{ pentru } \boldsymbol{X} \in \omega_2 \end{cases} \tag{12}$$

Cazul a mai multe clase se poate reduce la clasificarea cu două clase, dacă se adoptă una din următoarele strategii de construcție a clasificatorilor multicategoriali, strategii valabile pentru orice tip de clasificator (nu numai pentru cei liniari).

Strategia (1). Se construieşte, corespunzător fiecărei clase, un clasificator care să împartă spațiul trăsăturilor în două regiuni, astfel încât una să conțină clasa respectivă, iar cealaltă restul claselor.

Decizia de clasificare se ia folosind regula:

$$\boldsymbol{X} \in \omega_j \Leftrightarrow \boldsymbol{D}_j(\boldsymbol{X}) > \boldsymbol{D}_i(\boldsymbol{X}), \forall i \neq j, i = 1\ldots K \tag{13}$$

Se remarcă apariția unei regiuni de nedeterminare (gri) pentru care nu se poate lua o decizie.

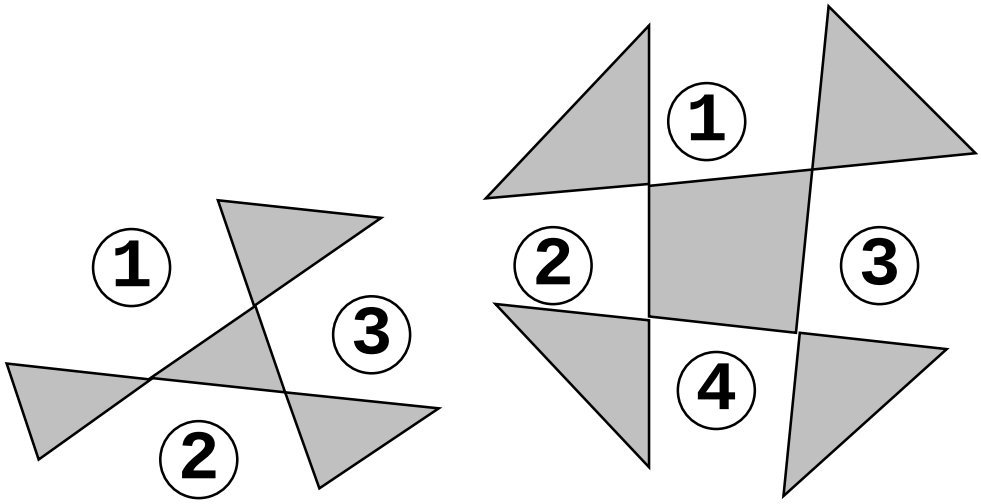


Fig. 2. Clasificator liniar multicategorial, metoda 1.

Strategia (2). Se construiește, corespunzător fiecărei perechi de clase, un clasificator care să împartă spațiul trăsăturilor în două regiuni, astfel încât una să conțină clasele respective, iar cealaltă restul claselor:

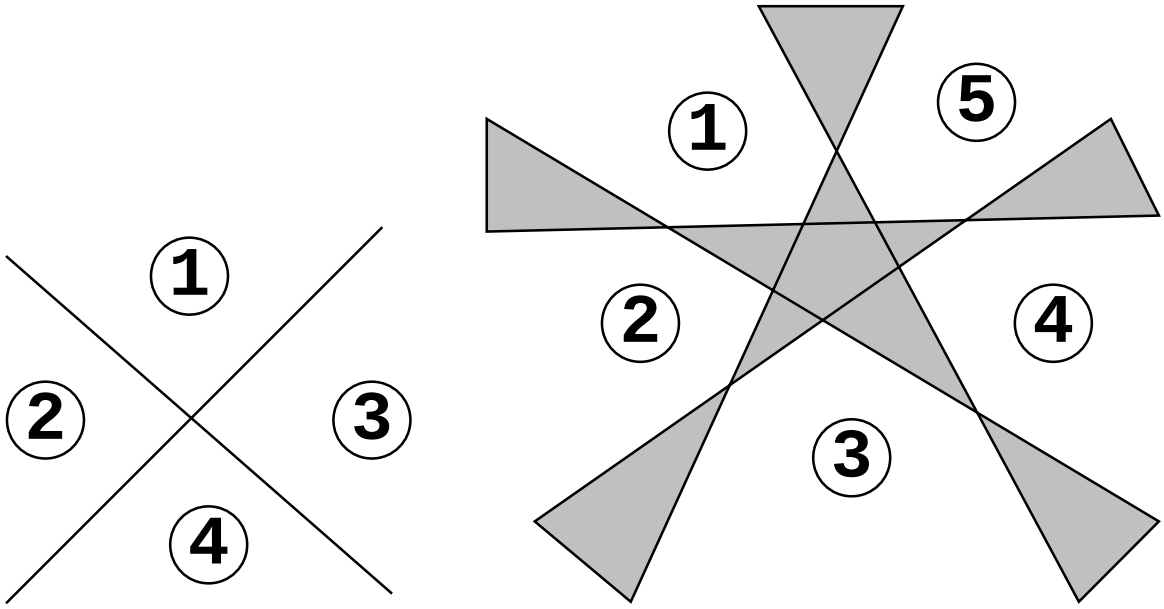


Fig. 3. Clasificator liniar multicategorial, metoda 2.

În acest caz se construiesc funcții de decizie de forma:

$$\boldsymbol{X} \in \omega_i \text{ sau } \boldsymbol{X} \in \omega_j \Leftrightarrow \boldsymbol{D}_{ij}(\boldsymbol{X}) > 0, \text{ unde } i \neq j \tag{14}$$

Se ia decizia de apartenență la una din clase folosind regula:

$$\boldsymbol{X} \in \omega_i \Leftrightarrow \boldsymbol{D}_{ij}(\boldsymbol{X}) > 0, \text{ unde } j \neq i, j = 1...K \tag{15}$$

Strategia (3). Fiecare funcție discriminant separă doar două clase între ele, caz în care zona de nedeterminare se reduce foarte mult.

În acest caz se construiesc funcții de decizie de forma:

$$\boldsymbol{D}_{ij}(\boldsymbol{X}) = \boldsymbol{D}_i(\boldsymbol{X}) - \boldsymbol{D}_j(\boldsymbol{X}) \tag{16}$$

regula de decizie folosită în acest caz fiind de forma:

$$\boldsymbol{X} \in \omega_i \Leftrightarrow \boldsymbol{D}_{ij}(\boldsymbol{X}) > 0, \text{ unde } j \neq i, j = 1...K \tag{17}$$

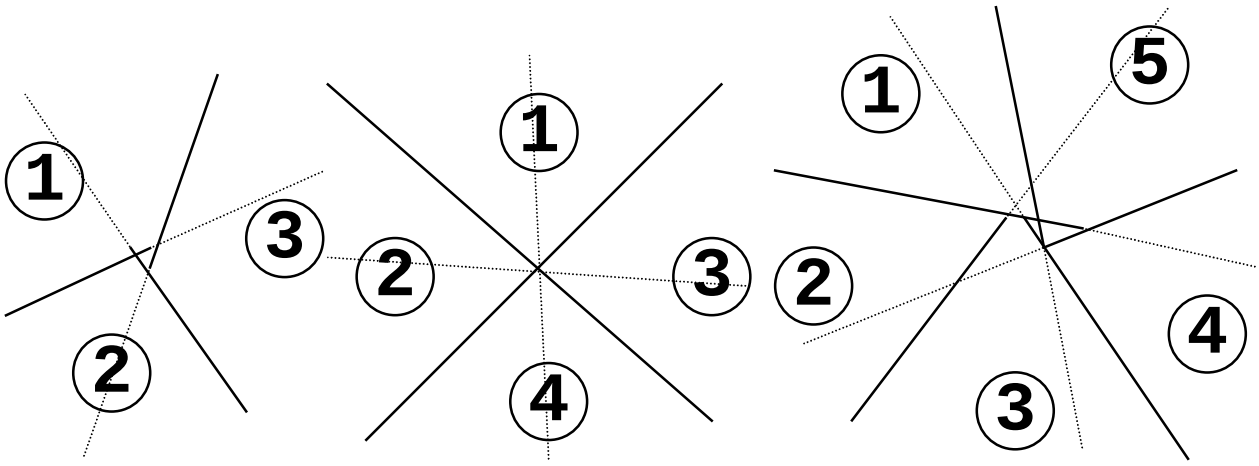


Fig. 4. Clasificator liniar multicategorial, metoda 3.

Această strategie este cea mai puternică, reuşind să separe liniar clase care nu pot fi separate folosind primele două metode:

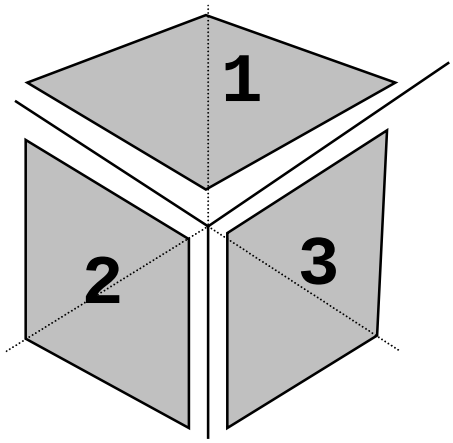


Fig. 5. Exemplu de clasificare liniară multicategorială, metoda 3.

10.1. Clasificatorul de distanţă minimă.

Pentru fiecare clasă se definesc vectorii prototip prin:

$$\boldsymbol{R}_i = \frac{1}{M_i}\sum_{j=1}^{M_i}\boldsymbol{X}_j, i = 1...K \tag{18}$$

şi care reprezintă centrul de greutate (media) formelor care alcătuiesc setul de antrenament (învăţare, formare) al clasei ω_i, iar M_i fiind numărul de asemenea vectori.

Funcţiile discriminant se construiesc în acest caz pe baza distanţei de la un vector oarecare de intrare (formă) la vectorii prototip, distanţă notată $\boldsymbol{d}(\boldsymbol{X},\boldsymbol{R}_i)$. Decizia se ia conform regulii:

$$\boldsymbol{X} \in \omega_i \Leftrightarrow \boldsymbol{d}(\boldsymbol{X},\boldsymbol{R}_i) < \boldsymbol{d}(\boldsymbol{X},\boldsymbol{R}_j),\ \forall j \neq i, j = 1...K \tag{19}$$

sau echivalent:

$$\boldsymbol{X} \in \omega_i \Leftrightarrow \boldsymbol{d}^2(\boldsymbol{X},\boldsymbol{R}_i) < \boldsymbol{d}^2(\boldsymbol{X},\boldsymbol{R}_j),\ \forall j \neq i, j = 1...K \tag{20}$$

Dacă se foloseşte distanţa euclidiană, atunci:

$$\boldsymbol{d}(\boldsymbol{X},\boldsymbol{R}_i) = \sum_{m=1}^{n}(x_m - r_{im})^2 \tag{21}$$

Pe baza acestei distanţe se construieşte funcţia discriminant, punând, mai întâi, distanţa precedentă sub forma:

$$\boldsymbol{d}(\boldsymbol{X},\boldsymbol{R}_i) = \|\boldsymbol{X} - \boldsymbol{R}_i\|^2 = \|\boldsymbol{X}\|^2 - 2\boldsymbol{R}_i^T\boldsymbol{X} + \|\boldsymbol{R}_i\|^2 \tag{22}$$

în care se remarcă faptul că primul termen este indepentent de clasă, deci poate fi eliminat, iar ultimul termen se calculează o singură dată, în faza de construcție a clasificatorului. Atunci definirea funcției discriminant este imediată, folosind:

$$\boldsymbol{X} \in \omega_i \Leftrightarrow \boldsymbol{d}^2(\boldsymbol{X},\boldsymbol{R}_i) = \text{minim} \Leftrightarrow 2\boldsymbol{R}_i^T\boldsymbol{X} - \|\boldsymbol{R}_i\|^2 = \text{maxim} \tag{23}$$

rezultă:

$$\boldsymbol{D}_i(\boldsymbol{X}) = 2\boldsymbol{R}_i^T\boldsymbol{X} - \|\boldsymbol{R}_i\|^2 \tag{24}$$

Deşi în acest caz s-a folosit distanța euclidiană, în general mai pot fi folosite şi alte distanțe, cum ar fi distanța generalizată (Minkovski), distanța Manhattan, distanța Hamming (pentru forme binare) sau distanța Tanimoto (Hamming normalizată):

- distanța Minkovski $$\boldsymbol{d}(\boldsymbol{X},\boldsymbol{Y}) = \left(\sum_{i=1}^{n}(x_i - y_i)^p\right)^{1/p} \tag{25}$$

- distanța Manhattan $$\boldsymbol{d}(\boldsymbol{X},\boldsymbol{Y}) = \sum_{i=1}^{n}|x_i - y_i| \tag{26}$$

- distanța Hamming $$\boldsymbol{d}(\boldsymbol{X},\boldsymbol{Y}) = \sum_{i=1}^{n} x_i \oplus y_i \tag{27}$$

- distanța Tanimoto $$\boldsymbol{d}(\boldsymbol{X},\boldsymbol{Y}) = \sum_{i=1}^{n} x_i y_i \Big/ \sum_{i=1}^{n}(x_i + y_i) \tag{28}$$

Funcția de decizie dată de un clasificator liniar pentru clasa ω_i, adică:

$$\boldsymbol{D}_i(\boldsymbol{X}) = \sum_{k=1}^{n+1} w_{ik}x_k \tag{29}$$

defineşte prin $\boldsymbol{D}_i(\boldsymbol{X}) = 0$ ecuația unui hiperplan în spațiul $(n+1)$ dimensional (extins) al trăsăturilor, hiperplan ce trece prin origine. Un asemenea plan este complet determinat de ***vectorul normalei*** la plan, vector definit de:

$$\boldsymbol{W}_i = \begin{bmatrix} w_{i1} & \ldots & w_{in} & w_{i,n+1} \end{bmatrix}^T \tag{30}$$

şi denumit vectorul de ponderi. Ca urmare se poate lucra direct cu aceşti vectori, modul de operare fiind:

$$\boldsymbol{X} \in \omega_i \Leftrightarrow \boldsymbol{D}_i(\boldsymbol{X}) = \boldsymbol{W}_i^T\boldsymbol{X} > 0 \Leftrightarrow \langle \boldsymbol{W}_i, \boldsymbol{X} \rangle > 0 \tag{31}$$

spațiul extins al trăsăturilor fiind echivalat în acest caz cu ***spațiul ponderilor***.

10.2. Formarea clasificatorului liniar.

Fie $\boldsymbol{X} = (x_1 \quad x_2 \quad \ldots \quad x_n)^T$ un vector de formă, $\boldsymbol{X} \in \Omega \subset R^n$, unde Ω este spațiul formelor, iar "*n*" este numărul de trăsături.

Pentru cazul clasificării binare, funcția discriminant este dată de:

$$\boldsymbol{D}(\boldsymbol{X}) = \sum_{i=1}^{n+1}(w_i^{(1)} - w_i^{(2)})x_i \overset{\Delta}{=} \sum_{i=1}^{n+1} w_i x_i = \boldsymbol{W}^T\boldsymbol{X} \tag{32}$$

care poate fi interpretată ca fiind ecuația unui hiperplan în spațiul extins al trăsăturilor, sau echivalent, ca produsul scalar dintre vectorul $\boldsymbol{W}$ şi vectorul extins $\boldsymbol{X}$. Vectorul $\boldsymbol{W}$, numit vectorul de ponderi, este normal la hiperplanul de decizie (care trece prin origine).

Ca urmare, în spațiul ponderilor, regula de decizie este:

$$\begin{aligned} &\boldsymbol{X} \in \omega_1 \Leftrightarrow s = \boldsymbol{W}^T\boldsymbol{X} > 0 \\ &\boldsymbol{X} \in \omega_2 \Leftrightarrow s = \boldsymbol{W}^T\boldsymbol{X} < 0 \end{aligned} \tag{33}$$

Presupunând că avem un set de vectori (de antrenament) a căror apartenență la clase este cunoscută, formarea este un procedeu iterativ care permite construirea hiperplanului de decizie în cazul în care cele două clase sunt liniar separabile.

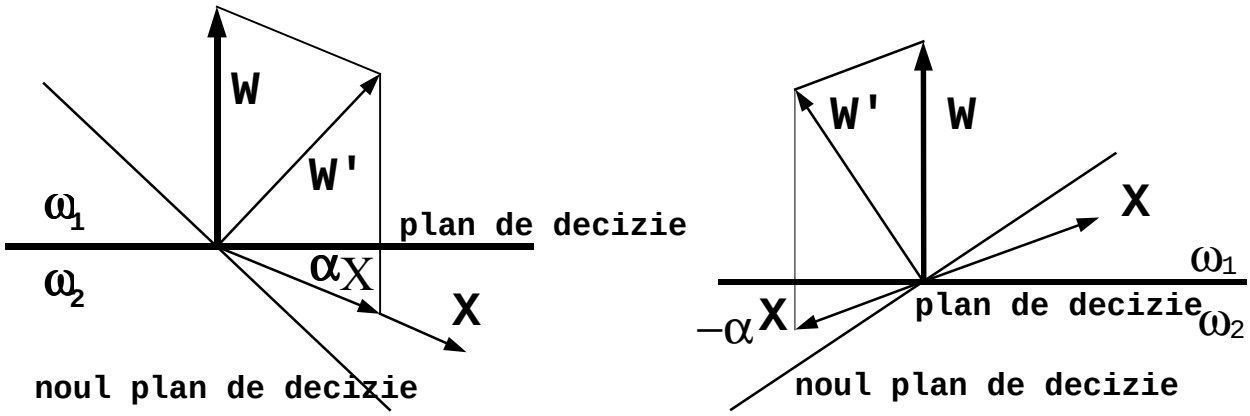


Fig. 6. Formarea clasificatorului liniar.

Dacă $\boldsymbol{X} \in \omega_1$ şi este clasificat greşit ca aparținând lui ω_2, se poate corecta această situație construind un nou vector de ponderi prin adăugarea unei fracțiuni din $\boldsymbol{X}$ la vectorul de ponderi precedent:

$$\boldsymbol{W}' = \boldsymbol{W} + \alpha \boldsymbol{X} \tag{34}$$

Dacă $\boldsymbol{X} \in \omega_2$ şi este clasificat greşit ca aparținând lui ω_1, se poate corecta această situație construind un nou vector de ponderi prin scăderea unei fracțiuni a lui $\boldsymbol{X}$ din vectorul de ponderi precedent:

$$\boldsymbol{W}' = \boldsymbol{W} - \alpha \boldsymbol{X} \tag{35}$$

Valoarea factorului de corecție "α" se poate obține din expresia produsului scalar:

$$s' = \boldsymbol{W}'^T \boldsymbol{X} = (\boldsymbol{W} + \alpha \boldsymbol{X})^T \boldsymbol{X} = \boldsymbol{W}^T \boldsymbol{X} + \alpha \left\| \boldsymbol{X} \right\|^2 \tag{36}$$

Rezultă:

$$\alpha = \frac{s' - s}{\left\| \boldsymbol{X} \right\|^2} \tag{37}$$

Dacă se impune, de exemplu, $s' = -s$, atunci rezultă:

$$\alpha = -2 \frac{\boldsymbol{W}^T \boldsymbol{X}}{\left\| \boldsymbol{X} \right\|^2} \tag{38}$$

În general însă, se pot utiliza diverse strategii pentru alegerea factorului de corecție:

(1). Metoda factorului de corecție constant:

$$\alpha = \text{const.}, \alpha > 0$$

(2). Metoda corecției absolute:

$$\alpha = \left\lfloor \text{cel mai mic întreg} > \left| \boldsymbol{W}^T \boldsymbol{X} \right| / \left\| \boldsymbol{X} \right\|^2 \right\rfloor$$

(3). Metoda corecției fracționare:

$$\alpha = \lambda \frac{\left| \boldsymbol{W}^T \boldsymbol{X} \right|}{\left\| \boldsymbol{X} \right\|^2}, \lambda \in (0,2]$$

Corecția în acest ultim caz se poate face direct funcție de "λ", adică:

$$\boldsymbol{W}' = \boldsymbol{W} + \lambda \frac{\left| \boldsymbol{W}^T \boldsymbol{X} \right|}{\left\| \boldsymbol{X} \right\|^2} \boldsymbol{X} \Leftrightarrow \boldsymbol{W}'\boldsymbol{X} = \boldsymbol{W}\boldsymbol{X} + \lambda \left| \boldsymbol{W}^T \boldsymbol{X} \right| \Rightarrow \lambda = \frac{\left| \boldsymbol{W}'^T \boldsymbol{X} - \boldsymbol{W}^T \boldsymbol{X} \right|}{\left| \boldsymbol{W}^T \boldsymbol{X} \right|} \tag{39}$$

Ultima formulă obținută dă informații despre semnificația lui "λ", adică:

(1). Dacă $\lambda < 1$, atunci planul de decizie se deplasează către "$\boldsymbol{X}$" (în jurul originii), fără a-l atinge însă niciodată.

(2). Dacă $\lambda = 1$, atunci noul plan de decizie va trece prin "$\boldsymbol{X}$".

(3). Dacă $1 < \lambda < 2$, planul de decizie trece de partea cealaltă în raport cu "$\boldsymbol{X}$", clasificându-l corect.

(4). Dacă $\lambda = 2$, vectorul de formă "$\boldsymbol{X}$" se va găsi de partea cealaltă a noului plan, la distanță egală.

Dacă clasele sunt separabile liniar, plecând de la un hiperplan de decizie inițial arbitrar ales, după un număr de iterații se reuşeşte clasificarea corectă a tuturor formelor din setul de antrenament.

Dacă clasele nu sunt separabile liniar, apar oscilații ale limitelor de decizie. S-au elaborat algoritmi care detectează aceste oscilații şi apoi adaugă (generează) noi vectori prototip subdivizând subclasele, până se realizează separabilitatea claselor.

10.3. Funcții discriminant liniare pe porțiuni.

Fiecare clasă este definită în acest caz de un set de vectori de referință:

$$\omega_k \to \boldsymbol{R}^{(k)} = \left\{ \boldsymbol{X}_m^{(k)} \right\}_{m=1...M_k} \tag{40}$$

unde M_k este numărul de vectori de referință pentru clasa ω_k.

Pentru clasificatorul liniar pe porțiuni se defineşte distanță între un vector de formă oarecare $\boldsymbol{X}$ şi clasa ω_k prin:

$$\boldsymbol{d}(\boldsymbol{X},\omega_k) = \min_{m=1...M_k} \boldsymbol{d}(\boldsymbol{X},\boldsymbol{X}_m^{(k)}) \stackrel{\Delta}{=} \boldsymbol{d}(\boldsymbol{X},\boldsymbol{R}^{(k)}) \tag{41}$$

Deci, practic, se construiesc funcții discriminant de distanță minimă pentru fiecare vector prototip ataşat unei clase.

Regula de decizie este:

$$\boldsymbol{X} \in \omega_k \Leftrightarrow \boldsymbol{d}(\boldsymbol{X},\omega_k) < \boldsymbol{d}(\boldsymbol{X},\omega_j), \forall j \neq k, j = 1...K \tag{42}$$

sau echivalent:

$$\boldsymbol{X} \in \omega_k \Leftrightarrow \boldsymbol{d}(\boldsymbol{X},\omega_k) = \min_{j=1...K} \boldsymbol{d}(\boldsymbol{X},\omega_j), \forall j \neq k, j = 1...K \tag{43}$$

unde "K" este numărul de clase.

Expresia funcției discriminant corespunzătoare se deduce în continuare prin:

$$\boldsymbol{d}(\boldsymbol{X},\boldsymbol{X}_m^{(k)}) = \left\| \boldsymbol{X} - \boldsymbol{X}_m^{(k)} \right\|^2 = \left\| \boldsymbol{X} \right\|^2 - 2\boldsymbol{X}^T \boldsymbol{X}_m^{(k)} + \left\| \boldsymbol{X}_m^{(k)} \right\|^2 \tag{44}$$

În continuare se procedează la fel ca la clasificatorul de distanță minimă, adică, folosind remarca că primul termen al expresiei anterioare este redundant, avem:

$$\boldsymbol{D}(\boldsymbol{X},\boldsymbol{X}_m^{(k)}) \stackrel{\Delta}{=} 2\boldsymbol{X}^T \boldsymbol{X}_m^{(k)} - \left\| \boldsymbol{X}_m^{(k)} \right\|^2 \tag{45}$$

Funcția discriminant pentru clasificatorul liniar pe porțiuni se scrie atunci:

$$\boldsymbol{D}(\boldsymbol{X},\omega_k) = \max_{m=1...M_k} \boldsymbol{D}(\boldsymbol{X},\boldsymbol{X}_m^{(k)}) = \max_{m=1...M_k} (\boldsymbol{X}^T \boldsymbol{X}_m^{(k)} - \left\| \boldsymbol{X}_m^{(k)} \right\|^2) \tag{46}$$

Regula de decizie pentru clasificatorul liniar pe porțiuni devine:

$$\boldsymbol{X} \in \omega_k \Leftrightarrow \boldsymbol{D}(\boldsymbol{X},\omega_k) = \max_{j=1...K} \boldsymbol{D}(\boldsymbol{X},\omega_j) \tag{47}$$

Acest clasificator poate fi văzut ca fiind obținut prin divizarea fiecărei clase în subclase, fiecare caracterizate de câte un vector prototip, şi construind câte un clasificator multicategorial de distanță minimă pentru fiecare clasă.

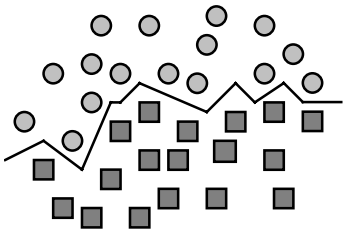


Fig. 7. Clasificator liniar pe porțiuni.

10.4. Formarea clasificatorului liniar pe porțiuni.

În acest caz se procedează oarecum analog cazului clasificatorului de distanță minimă, însă pentru fiecare vector prototip care defineşte o clasă în parte.

Dacă $\boldsymbol{X} \in \omega_k \subset \omega_1$ şi este clasificat greşit ca aparținând lui $\omega_p \subset \omega_2$, se poate corecta această situație construind un nou vector de ponderi prin adăugarea unei fracțiuni din $\boldsymbol{X}$ la vectorul de ponderi precedent:

$$\boldsymbol{W}_k' = \boldsymbol{W}_k + \alpha \boldsymbol{X}, \alpha > 0 \tag{48}$$

Dacă $\boldsymbol{X} \in \omega_p \subset \omega_2$ şi este clasificat greşit ca aparținând lui $\omega_k \subset \omega_1$, se poate corecta această situație construind un nou vector de ponderi, adaugând o fracțiune din $\boldsymbol{X}$ la vectorul de ponderi precedent:

$$\boldsymbol{W}_p' = \boldsymbol{W}_p - \alpha \boldsymbol{X}, \alpha > 0 \tag{49}$$

Metodele de alegere pentru factorul de corecție "α" sunt aceleaşi:

(1). Metoda factorului de corecție constant.

(2). Metoda corecției absolute.

(3). Metoda corecției fracționare.

Singura condiție pe care trebuie să o îndeplinească $\boldsymbol{W}$ este să fie perpendicular pe hiperplanul de decizie. Totuşi, în aceste condiții, deoarece el intră în evaluarea funcțiilor de decizie, procesul de decizie va favoriza funcțiile pentru care $|\boldsymbol{W}|$ este mai mare. De aceea se preferă lucrul cu versorul de ponderi, adică cu:

$$\boldsymbol{W}_k' = \frac{\boldsymbol{W}_k + \alpha \boldsymbol{X}}{\left\| \boldsymbol{W}_k + \alpha \boldsymbol{X} \right\|}, \text{ dacă } \boldsymbol{X} \in \omega_k \text{ e clasificat greşit în } \omega_p \text{ şi} \tag{50}$$

$$\boldsymbol{W}_p' = \frac{\boldsymbol{W}_p - \alpha \boldsymbol{X}}{\left\| \boldsymbol{W}_p - \alpha \boldsymbol{X} \right\|}, \text{ dacă } \boldsymbol{X} \in \omega_p \text{ e clasificat greşit în } \omega_k . \tag{51}$$

10.5. Clasificatorul celor mai apropiați "*k*" vecini.

Clasificatorul liniar pe porțiuni descris anterior este un clasificator de distanță minimă în raport cu vectorii prototip şi în acelaşi timp poate fi văzut ca un clasificator de tipul vecinului cel mai apropiat (NN - Nearest Neighbor).

O generalizare a metodei vecinului cel mai apropiat este clasificatorul celor mai apropiați "*k*" vecini. În acest caz apartenența unei forme la o clasă este decisă examinând un număr de "*k*" vecini (cei mai apropiați) ai formei respective, vecini a căror apartenență la clase este cunoscută.

Pentru găsirea celor mai apropiați "*k*" vecini se utilizează cel mai des distanța euclidiană. Metoda se poate aplica cu succes şi pentru forme neseparabile liniar, şi pentru clasificări multicategoriale.

Pentru clasificarea formelor binare se utilizează distanțele Hamming sau Tanimoto.

Regula de decizie cea mai simplă folosită este cea a numărului maxim de voturi:

$\boldsymbol{X} \in \omega_k$, dacă între cei "*k*" vecini, cei mai mulți aparțin clasei ω_k

Cazurile cu număr egal de voturi se elimină.

Alteori, contribuţia vecinilor la luarea deciziei se consideră şi funcţie de distanţa între "***X***" şi cei "*k*" vecini, adică:

$$\boldsymbol{V}^{(k)}(\boldsymbol{X}) = \sum_{i=1}^{N_k} \frac{1}{\boldsymbol{D}_i(\boldsymbol{X})} \qquad \text{sau} \qquad \boldsymbol{V}^{(k)}(\boldsymbol{X}) = \sum_{i=1}^{N_k} \frac{1}{\boldsymbol{D}_i^2(\boldsymbol{X})} \tag{52}$$

unde, între cei "*k*" vecini, există N_k aparţinând clasei ω_k.

Regula de decizie se scrie în acest caz:

$$\boldsymbol{X} \in \omega_j \Leftrightarrow \boldsymbol{V}^{(j)}(\boldsymbol{X}) > \boldsymbol{V}^{(i)}(\boldsymbol{X}),\ \forall i \neq j,\ i = 1...N_c \tag{53}$$

Rezultate eronate pot apare dacă clasele sunt definite prin vectori prototip în număr foarte diferit de la o clasă la alta, clasa definită prin mai puţini vectori prototip fiind dezavantajată. În acest caz se măreşte numărul de voturi pentru fiecare vector prototip al clasei dezavantajate cu raportul între numărul de vectori prototip ai clasei dezavantajate şi numărul de vectori prototip pentru cealaltă clasă.

Această metodă este laborioasă dacă numărul de forme prototip este mare. O variantă a acestei metode duce la clasificarea cu ajutorul funcţiilor de potenţial, în care potenţialul unui punct (unei forme) este estimat prin superpoziţia potenţialelor celor mai apropiaţi "*k*" vecini.

10.6. Clasificatorul celor mai mici pătrate.

Fie $\{\boldsymbol{Z}_{ij}\}$ setul de forme de antrenament, unde $\boldsymbol{Z}_{ij}$ este forma "*j*" a clasei ω_i. Pe baza lor se construieşte clasificatorul, operaţia de clasificare constând în atribuirea (maparea), pentru fiecare formă din setul de antrenament a unui număr (de exemplu, indicele clasei). Generalizând, se poate considera clasificarea ca fiind o mapare a fiecărei forme din spaţiul trăsăturilor în puncte ataşate fiecărei clase din spaţiul de decizie (având aceeaşi dimensiune):

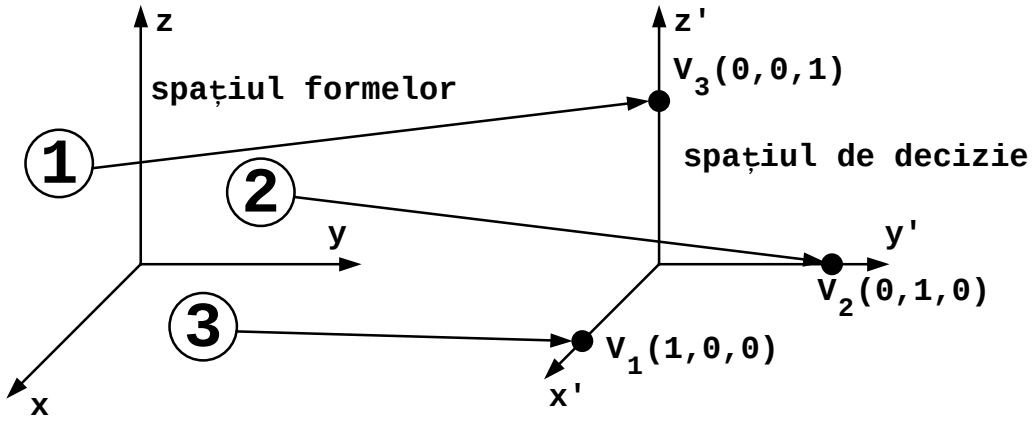


Fig. 8. Principiul clasificatorului celor mai mici pătrate.

Deci în spaţiul de decizie, fiecărei clase îi va corespunde un punct V_i. În cazul ideal, tuturor formelor dintr-o clasă ar trebui să le corespundă un singur punct în spaţiul de decizie. În realitate, această mapare se face cu o eroare care va fi minimizată. Deci corespondenţa va fi de forma:

$$\boldsymbol{Z}_{ij} \rightarrow \boldsymbol{A}\boldsymbol{Z}_{ij} \tag{54}$$

unde "***A***" este matricea transformării. Clasei ω_i îi ataşăm în spaţiul de decizie vectorul (punctul) V_i.

O măsură a erorii de clasificare este distanţa (în spaţiul de decizie) între punctul de mapare a vectorului de intrare ($\boldsymbol{A}\boldsymbol{Z}_{ij}$) şi punctul ataşat clasei ω_i (V_i).

Eroarea de clasificare poate fi estimată folosind vectorul de eroare:

$\vec{\varepsilon}_{ij} = \boldsymbol{A}\boldsymbol{Z}_{ij} - \boldsymbol{V}_i$ - pentru forma "*j*" a clasei ω_i.

Eroarea de clasificare pentru toți vectorii (formele) din clasa ω_i poate fi măsurată prin:

$$\varepsilon_i = \frac{1}{N_i}\sum_{j=1}^{N_i}\left\|\vec{\varepsilon}_{ij}\right\|^2 \tag{55}$$

unde N_i este numărul de forme de antrenament din clasa ω_i.

Eroarea globală de clasificare, pentru toate clasele se scrie atunci:

$$\varepsilon^2 = \frac{1}{N_c}\sum_{i=1}^{N_c}\varepsilon_i \tag{56}$$

unde N_c este numărul de clase. Relația anterioară trebuie corectată dacă probabilitățile de apariție ale simbolurilor din diferite clase nu sunt egale, adică:

$$\varepsilon^2 = \frac{1}{N_c}\sum_{i=1}^{N_c}P_i\varepsilon_i \tag{57}$$

Se doreşte determinarea transformării "$\boldsymbol{A}$" care să minimizeze eroarea de clasificare stabilită anterior:

$$\varepsilon = \frac{1}{N_c}\sum_{i=1}^{N_c}\frac{P_i}{N_i}\sum_{j=1}^{N_i}\left\|\vec{\varepsilon}_{ij}\right\|^2 = \frac{1}{N_c}\sum_{i=1}^{N_c}\frac{P_i}{N_i}\sum_{j=1}^{N_i}\left\|\boldsymbol{A}\boldsymbol{Z}_{ij} - \boldsymbol{V}_i\right\|^2 = \frac{1}{N_c}\sum_{i=1}^{N_c}\frac{P_i}{N_i}\sum_{j=1}^{N_i}(\boldsymbol{A}\boldsymbol{Z}_{ij} - \boldsymbol{V}_i)^T(\boldsymbol{A}\boldsymbol{Z}_{ij} - \boldsymbol{V}_i) \tag{58}$$

Rezultă:

$$\varepsilon = \frac{1}{N_c}\sum_{i=1}^{N_c}\frac{P_i}{N_i}\sum_{j=1}^{N_i}(\boldsymbol{Z}_{ij}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{Z}_{ij} - 2\boldsymbol{Z}_{ij}^T\boldsymbol{A}^T\boldsymbol{V}_i + \boldsymbol{V}_i^T\boldsymbol{V}_i) \tag{59}$$

Condiția de minimizare este:

$$\nabla_{\boldsymbol{A}}\varepsilon = 0 \Leftrightarrow \frac{1}{N_c}\sum_{i=1}^{N_c}\frac{P_i}{N_i}\sum_{j=1}^{N_i}(2\boldsymbol{A}\boldsymbol{Z}_{ij}\boldsymbol{Z}_{ij}^T - 2\boldsymbol{V}_i\boldsymbol{Z}_{ij}^T) = 0 \tag{60}$$

Rezultă în final:

$$\boldsymbol{A} = \left[\sum_{i=1}^{N_c}\frac{P_i}{N_i}\sum_{j=1}^{N_i}\boldsymbol{V}_i\boldsymbol{Z}_{ij}^T\right]\left[\sum_{i=1}^{N_c}\frac{P_i}{N_i}\sum_{j=1}^{N_i}\boldsymbol{Z}_{ij}\boldsymbol{Z}_{ij}^T\right]^{-1} \tag{61}$$

Matricea $\sum_{j=1}^{N_i}\boldsymbol{Z}_{ij}\boldsymbol{Z}_{ij}^T$ are semnificația unei matrici de autocorelație, iar $\sum_{j=1}^{N_i}\boldsymbol{V}_i\boldsymbol{Z}_{ij}^T$ are semnificația unei matrici de crosscorelație, adică:

$$\boldsymbol{S}(\boldsymbol{V}_i\boldsymbol{Z}_{ij}^T) \overset{\Delta}{=} \sum_{j=1}^{N_i}\boldsymbol{V}_i\boldsymbol{Z}_{ij}^T \qquad \text{şi} \qquad \boldsymbol{S}(\boldsymbol{Z}_{ij}\boldsymbol{Z}_{ij}^T) = \sum_{j=1}^{N_i}\boldsymbol{Z}_{ij}\boldsymbol{Z}_{ij}^T \tag{62}$$

de unde rezultă forma cel mai des utilizată pentru transformarea căutată:

$$\boldsymbol{A} = \left[\sum_{i=1}^{N_c}\frac{P_i}{N_i}\boldsymbol{S}(\boldsymbol{V}_i\boldsymbol{Z}_{ij}^T)\right]\left[\sum_{i=1}^{N_c}\frac{P_i}{N_i}\boldsymbol{S}(\boldsymbol{Z}_{ij}\boldsymbol{Z}_{ij}^T)\right]^{-1} \tag{63}$$

Matricea "$\boldsymbol{A}$" se calculează cunoscând matricile de autocorelație şi crosscorelație definite anterior sau estimându-le plecând de la setul de forme de antrenament. Calculul lui "$\boldsymbol{A}$" necesită inversarea unei matrici simetrice, pentru care există metode numerice puternice.

Dacă $\hat{\boldsymbol{Z}}$ este forma care trebuie clasificată, mai întâi se face trecerea în spațiul de decizie prin:

$$\boldsymbol{L} = \boldsymbol{A}\hat{\boldsymbol{Z}} \tag{64}$$

unde "$\boldsymbol{A}$" a fost deja calculat în faza de formare. În spațiul de decizie se foloseşte un clasificator de distanță minimă, deci ar trebui să calculăm, pentru fiecare clasă, distanța:

$$\boldsymbol{d}_i^2 = \left\|\boldsymbol{L} - \boldsymbol{V}_i\right\|^2 = \left\|\boldsymbol{L}\right\|^2 - 2\boldsymbol{L}^T\boldsymbol{V} + \left\|\boldsymbol{V}_i\right\|^2 \tag{65}$$

Primul termen al sumei este acelaşi pentru toate clasele, deci se poate renunța la calculul lui, şi ca urmare funcția discriminant poate fi definită sub forma:

$$\boldsymbol{D}_i = 2\boldsymbol{L}^T\boldsymbol{V} - \left\|\boldsymbol{V}_i\right\|^2 \tag{66}$$

Regula de decizie se scrie atunci:

$$\hat{\boldsymbol{Z}} \in \omega_i \Leftrightarrow \boldsymbol{D}_i > \boldsymbol{D}_j, \forall j \neq i, j = 1...N_c \tag{67}$$

Deoarece regula de mapare este construită, se poate alege, de exemplu, vectorul $\boldsymbol{V}_i$ de forma:

$\boldsymbol{V}_i = [0 \quad 0 \quad ... \quad 1 \quad ... \quad 0]^T$, cu "1" pe poziția "*i*" a vectorului.

Ca urmare avem $\left\|\boldsymbol{V}_i\right\|^2 = 1, \forall i = 1...N_c$ şi atunci funcția de decizie pentru clasa "*i*" este:

$$\boldsymbol{D}_i = \boldsymbol{L}^T\boldsymbol{V}_i = \boldsymbol{V}_i^T\boldsymbol{L} = \boldsymbol{V}_i^T\boldsymbol{A}\hat{\boldsymbol{Z}}, \text{ pentru } i = 1..N_c \tag{68}$$

Cele N_c relații se pot grupa într-o singură relație matriceală de forma:

$$\begin{bmatrix} D_1 \\ D_2 \\ ... \\ D_{N_c} \end{bmatrix} = \begin{bmatrix} 1 & 0 & ... & 0 \\ 0 & 1 & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & ... & 1 \end{bmatrix} \boldsymbol{A}\hat{\boldsymbol{Z}} = \boldsymbol{A}\hat{\boldsymbol{Z}} \Rightarrow \boldsymbol{D} = \boldsymbol{A}\hat{\boldsymbol{Z}} \tag{69}$$

Deci în condițiile impuse lui $\boldsymbol{V}_i$, clasificarea înseamnă calculul vectorului "$\boldsymbol{D}$" şi selecția valorii maxime din el, reținându-se indicele clasei.

11. Tehnici statistice de recunoaştere.

11.1. Principiul metodei.

Presupunerea fundamentală care se face este că există pentru fiecare clasă o funcție densitate de probabilitate care să poată fi folosită pentru determinarea probabilității ca o forma de intrare să aparțină unei clase. Aceste funcții definite pe spațiul formelor permit clasificarea prin selecția clasei de probabilitate maximă.

Vom folosi următoarele notații:

$\boldsymbol{P}(\omega_j)$ - probabilitatea apriorică a clasei ω_j;

$\boldsymbol{P}(\omega_j/\boldsymbol{X})$ - probabilitatea ca clasa ω_j să includă forma $\boldsymbol{X}$;

$\boldsymbol{P}(\boldsymbol{X}/\omega_j)$ - probabilitatea ca forma $\boldsymbol{X}$ să fie inclusă în clasa ω_j;

$\boldsymbol{F}(\omega_j,\omega_k)$ - funcția de pierdere, care descrie pierderea ce afectează procesul de recunoaştere când o formă din clasa ω_j este clasificată ca aparținând clasei ω_k.

Funcții de pierdere uzuale sunt:

a) ***Funcția de pierdere simetrică***, dată de:

$$\boldsymbol{F}(\omega_j/\omega_k) = \begin{cases} 0, \text{ pentru } j = k \\ 1, \text{ pentru } j \neq k \end{cases} \tag{1}$$

care mai poate fi scrisă:

$$\boldsymbol{F}(\omega_j/\omega_k) = 1 - \boldsymbol{\delta}(j,k) \tag{2}$$

unde $\boldsymbol{\delta}(j,k)$ este funcția delta Kronecker.

b) ***Funcția de pierdere diagonală***, descrisă de:

$$\boldsymbol{F}(\omega_j/\omega_k) = \begin{cases} -h_j, \text{ pentru } j = k \\ 0, \quad \text{pentru } j \neq k \end{cases} \tag{3}$$

Pe baza funcției de pierdere se definește pierderea medie condiționată sau riscul condiționat pentru fiecare clasă prin:

$$\boldsymbol{L}(\boldsymbol{X},\omega_k) = \sum_{j=1}^{N_C} \boldsymbol{F}(\omega_j/\omega_k)\,\boldsymbol{P}(\omega_j/\boldsymbol{X}) \tag{4}$$

Folosind formula lui Bayes (sau formula cauzei), avem:

$$\boldsymbol{P}(\omega_j/\boldsymbol{X}) = \frac{\boldsymbol{P}(\boldsymbol{X}/\omega_j)\,\boldsymbol{P}(\omega_j)}{\boldsymbol{P}(\boldsymbol{X})} \tag{5}$$

rezultă pentru riscul condiţionat expresia:

$$\boldsymbol{L}(\boldsymbol{X},\omega_k) = \frac{1}{\boldsymbol{P}(\boldsymbol{X})}\sum_{j=1}^{N_C}\boldsymbol{F}(\omega_j/\omega_k)\,\boldsymbol{P}(\boldsymbol{X}/\omega_j)\,\boldsymbol{P}(\omega_j) \tag{6}$$

Decizia de clasificare se ia pe baza riscului minim de eroare, deci conform regulii:

$$\boldsymbol{X} \in \omega_k \Leftrightarrow \boldsymbol{L}(\boldsymbol{X},\omega_k) < \boldsymbol{L}(\boldsymbol{X},\omega_j), \forall j = 1...N_C, j \neq k \tag{7}$$

Se remarcă faptul că procesul de luare a deciziei impune calculul a N_C funcţii de risc condiţionat, unde N_C este numărul de clase.

Trecerea la funcţii discriminant se face imediat definindu-le prin:

$$\boldsymbol{D}_k(\boldsymbol{X}) = -\boldsymbol{L}(\boldsymbol{X},\omega_k) \tag{8}$$

adică:

$$\boldsymbol{D}_k(X) = -\frac{1}{\boldsymbol{P}(\boldsymbol{X})}\sum_{j=1}^{N_C}\boldsymbol{F}(\omega_j/\omega_k)\,\boldsymbol{P}(\boldsymbol{X}/\omega_j)\,\boldsymbol{P}(\omega_j) \tag{9}$$

care defineşte ***funcţia de decizie pentru clasificatorul Bayes***.

Deci regula de decizie devine:

$$\boldsymbol{X} \in \omega_k \Leftrightarrow \boldsymbol{D}_k(\boldsymbol{X}) > \boldsymbol{D}_j(\boldsymbol{X}), \forall j = 1...N_C, j \neq k \tag{10}$$

Pentru funcţia de pierdere simetrică, riscul condiţionat are expresia:

$$\boldsymbol{L}(\boldsymbol{X},\omega_k) = \frac{1}{\boldsymbol{P}(\boldsymbol{X})}\sum_{j=1}^{N_C}\left[\boldsymbol{P}(\boldsymbol{X}/\omega_j)\,\boldsymbol{P}(\omega_j) - \boldsymbol{\delta}(j,k)\,\boldsymbol{P}(\boldsymbol{X}/\omega_j)\,\boldsymbol{P}(\omega_j)\right] \tag{11}$$

Dar din formula probabilităţii totale avem:

$$\boldsymbol{P}(\boldsymbol{X}) = \sum_{j=1}^{N_C}\boldsymbol{P}(\boldsymbol{X}/\omega_j)\,\boldsymbol{P}(\omega_j) \tag{12}$$

Şi ca urmare:

$$\boldsymbol{L}(\boldsymbol{X},\omega_k) = 1 + \frac{1}{\boldsymbol{P}(\boldsymbol{X})}\boldsymbol{P}(\boldsymbol{X}/\omega_k)\,\boldsymbol{P}(\omega_k) \tag{13}$$

Probabilitatea $\boldsymbol{P}(\boldsymbol{X})$ este aceeaşi pentru toate funcţiile de risc condiţionat, deci putem folosi:

$$\boldsymbol{L}(\boldsymbol{X},\omega_k) \overset{\Delta}{=} -\boldsymbol{P}(\boldsymbol{X}/\omega_k)\,\boldsymbol{P}(\omega_k) \tag{14}$$

Rezultă funcţia de decizie a clasificatorului Bayes pentru funcţia de pierdere simetrică:

$$\boldsymbol{D}_k(\boldsymbol{X}) = \boldsymbol{P}(\boldsymbol{X}/\omega_k)\,\boldsymbol{P}(\omega_k) \tag{15}$$

Pe baza ei se scrie regula de decizie:

$$\boldsymbol{X} \in \omega_k \Leftrightarrow \boldsymbol{P}(\boldsymbol{X}/\omega_k)\,\boldsymbol{P}(\omega_k) > \boldsymbol{P}(\boldsymbol{X}/\omega_j)\,\boldsymbol{P}(\omega_j) \Leftrightarrow \lambda = \frac{\boldsymbol{P}(\boldsymbol{X}/\omega_k)}{\boldsymbol{P}(\boldsymbol{X}/\omega_j)} > \frac{\boldsymbol{P}(\omega_j)}{\boldsymbol{P}(\omega_k)} \tag{16}$$

unde cu λ s-a notat ***raportul de verosimilitate***.

Pentru funcţia de pierdere diagonală, expresia riscului condiţionat este

$$\boldsymbol{L}(\boldsymbol{X},\omega_k) = \frac{1}{\boldsymbol{P}(\boldsymbol{X})}(-h_k)\,\boldsymbol{P}(\boldsymbol{X}/\omega_k)\,\boldsymbol{P}(\omega_k) \tag{17}$$

şi din aceleaşi considerente ca mai sus vom folosi:

$$\boldsymbol{L}(\boldsymbol{X},\omega_k) \overset{\Delta}{=} -h_k\,\boldsymbol{P}(\boldsymbol{X}/\omega_k)\,\boldsymbol{P}(\omega_k) \tag{18}$$

Funcţia de decizie a clasificatorului Bayes pentru funcţia de pierdere diagonală este:

$$\boldsymbol{D}_k(\boldsymbol{X}) = h_k \boldsymbol{P}(\boldsymbol{X}/\omega_k)\boldsymbol{P}(\omega_k) \tag{19}$$

iar regula de decizie corespunzătoare se scrie:

$$\boldsymbol{X} \in \omega_k \Leftrightarrow h_k \boldsymbol{P}(\boldsymbol{X}/\omega_k)\boldsymbol{P}(\omega_k) > h_j \boldsymbol{P}(\boldsymbol{X}/\omega_j)\boldsymbol{P}(\omega_j) \Leftrightarrow \lambda = \frac{\boldsymbol{P}(\boldsymbol{X}/\omega_k)}{\boldsymbol{P}(\boldsymbol{X}/\omega_j)} > \frac{h_j \boldsymbol{P}(\omega_j)}{h_k \boldsymbol{P}(\omega_k)} \tag{20}$$

Probabilitatea globală de eroare pentru clasificatorul Bayes este:

$$P_e = \sum_{i=1}^{N_C} \left[\boldsymbol{P}(\omega_i) \int_{\overline{\omega}_i} \boldsymbol{P}(\boldsymbol{X}/\omega_i) d\boldsymbol{X} \right] \tag{21}$$

11.2. Metode parametrice de clasificare statistică.

În cazul metodelor parametrice se presupun cunoscute caracteristicile statistice ale formelor de intrare, adică densitatea de probabilitate pentru fiecare clasă şi probabilităţile apriorice ale claselor.

Cel mai des, sistemele reale sunt descrise de legea normală. Pentru cazul unidimensional avem:

$$\varphi(x) = \frac{1}{\sigma\sqrt{2\pi}} \boldsymbol{exp}\left[-\frac{(x-m)^2}{2\sigma^2} \right] \tag{22}$$

unde "m" şi "σ" sunt respectiv media şi dispersia variabilei aleatoare cu distribuţie normală.

Pentru cazul multidimensional, mediei îi corespunde:

$$\boldsymbol{M} = \boldsymbol{E}\{\boldsymbol{X}\} \tag{23}$$

iar dispersiei:

$$\boldsymbol{\Phi} = \boldsymbol{E}\{(\boldsymbol{X}-\boldsymbol{M})(\boldsymbol{X}-\boldsymbol{M})^T\} \tag{24}$$

unde E este operatorul speranţă matematică, iar $\boldsymbol{\Phi}$ este matricea de covarianţă a variabilei aleatoare multidimensionale $\boldsymbol{X} = (x_1, x_2, ..., x_n)$.

Densitatea de repartiţie multidimensională este:

$$\boldsymbol{\Theta}(\boldsymbol{X}) = \frac{1}{|\boldsymbol{\Phi}|^{1/2}(2\pi)^{n/2}} \boldsymbol{exp}\left[-\frac{1}{2}(\boldsymbol{X}-\boldsymbol{M})^T \boldsymbol{\Phi}^{-1}(\boldsymbol{X}-\boldsymbol{M}) \right] \tag{25}$$

Deci:

$$\boldsymbol{P}(\boldsymbol{X}/\omega_k) = \frac{1}{|\boldsymbol{\Phi}_k|^{1/2}(2\pi)^{n/2}} \boldsymbol{exp}\left[-\frac{1}{2}(\boldsymbol{X}-\boldsymbol{M}_k)^T \boldsymbol{\Phi}_k^{-1}(\boldsymbol{X}-\boldsymbol{M}_k) \right] \tag{26}$$

Pentru funcţia discriminant se preferă folosirea logaritmului:

$$\boldsymbol{D}_k(\boldsymbol{X}) \stackrel{\Delta}{=} \boldsymbol{log}\left[\boldsymbol{P}(\boldsymbol{X}/\omega_k)\boldsymbol{P}(\omega_k) \right] \tag{27}$$

Renunţând la termenii care sunt independenţi de clasă, rezultă:

$$\boldsymbol{D}_k(\boldsymbol{X}) = -\frac{1}{2}\boldsymbol{log}|\boldsymbol{\Phi}_k| - \frac{1}{2}\boldsymbol{X}^T\boldsymbol{\Phi}_k^{-1}\boldsymbol{X} + \boldsymbol{M}_k^T\boldsymbol{\Phi}_k^{-1}\boldsymbol{X} - \frac{1}{2}\boldsymbol{M}_k^T\boldsymbol{\Phi}_k^{-1}\boldsymbol{M}_k + \boldsymbol{log}\,\boldsymbol{P}(\omega_k) \tag{28}$$

Sunt foarte utile unele propuneri simplificatoare:

1) Dacă matricile de covarianţă sunt egale pentru toate clasele se obţine binecunoscutul clasificator liniar:

$$\boldsymbol{D}_k(\boldsymbol{X}) = \boldsymbol{M}_k^T\boldsymbol{\Phi}^{-1}\boldsymbol{X} - \frac{1}{2}\boldsymbol{M}_k^T\boldsymbol{\Phi}^{-1}\boldsymbol{M}_k + \boldsymbol{log}\,\boldsymbol{P}(\omega_k) \tag{29}$$

2) Făcând presupunerea suplimentară a clasificării binare, funcţia de decizie se scrie:

$$\boldsymbol{D}(\boldsymbol{X}) = (\boldsymbol{M}_1^T - \boldsymbol{M}_2^T)\boldsymbol{\Phi}^{-1}\boldsymbol{X} - \frac{1}{2}\boldsymbol{M}_1^T\boldsymbol{\Phi}^{-1}\boldsymbol{M}_1 + \frac{1}{2}\boldsymbol{M}_2^T\boldsymbol{\Phi}^{-1}\boldsymbol{M}_2 + \boldsymbol{log}\,\boldsymbol{P}(\omega_1)/\boldsymbol{P}(\omega_2) \tag{29}$$

3) Dacă probabilităţile apriorice ale claselor sunt egale, funcţia de decizie multicategorială este:

$$\boldsymbol{D}_k(\boldsymbol{X}) = \boldsymbol{M}_k^T \boldsymbol{\Phi}^{-1} \boldsymbol{X} - \frac{1}{2} \boldsymbol{M}_k^T \boldsymbol{\Phi}^{-1} \boldsymbol{M}_k \tag{30}$$

4) Dacă trăsăturile formei de intrare sunt necorelate, atunci $\boldsymbol{\Phi}$ este o matrice diagonală şi avem:

$$\boldsymbol{D}_k(\boldsymbol{X}) = \boldsymbol{M}_k^T \boldsymbol{X} - \frac{1}{2} \boldsymbol{M}_k^T \boldsymbol{M}_k \tag{31}$$

care este tocmai clasificatorul binar de distanţă minimă. Trăsături necorelate se obţin în urma transformării Hotteling (Karhunen-Loeve discretă).

11.3. Metode neparametrice de recunoaştere statistică.

Pierderea medie condiţionată (riscul condiţionat) pentru un clasificator bayesian este dat de:

$$\boldsymbol{L}(\boldsymbol{X},\omega_k) = \frac{1}{\boldsymbol{P}(\boldsymbol{X})} \sum_{i=1}^{N_c} \boldsymbol{F}(\omega_k/\omega_i)\, \boldsymbol{P}(\boldsymbol{X}/\omega_i)\, \boldsymbol{P}(\omega_i) \tag{32}$$

Riscul global de clasificare incorectă, a cărui minimizare se doreşte, este dat de:

$$r = \int_{\Omega} \boldsymbol{L}(\boldsymbol{X},\omega_k)\, \boldsymbol{P}(\boldsymbol{X}) d\boldsymbol{X} \tag{33}$$

unde integrala se calculează peste întreg spaţiul formelor, deoarece, evident:

$$\bigcup_{k=1}^{N_c} \omega_k \subseteq \boldsymbol{\Omega} \tag{34}$$

Înlocuind prima formulă în cea precedentă, se obţine:

$$r = \int_{\Omega} \left[\sum_{i=1}^{N_c} \boldsymbol{F}(\omega_k/\omega_i)\, \boldsymbol{P}(\boldsymbol{X}/\omega_i)\, \boldsymbol{P}(\omega_i) \right] d\boldsymbol{X} = \sum_{i=1}^{N_c} \boldsymbol{P}(\omega_i) \left[\int_{\Omega} \boldsymbol{F}(\omega_k/\omega_i)\, \boldsymbol{P}(\boldsymbol{X}/\omega_i) d\boldsymbol{X} \right] \tag{35}$$

Deoarece orice clasificare multicategorială poate fi realizată folosind un set adecvat de clasificatori binari, formula anterioară devine pentru cazul $N_c = 2$:

$$\begin{aligned} r = {} & \boldsymbol{P}(\omega_1) \int_{\omega_1} \boldsymbol{F}(\omega_1/\omega_1)\, \boldsymbol{P}(\boldsymbol{X}/\omega_1) d\boldsymbol{X} + \boldsymbol{P}(\omega_1) \int_{\omega_2} \boldsymbol{F}(\omega_1/\omega_2)\, \boldsymbol{P}(\boldsymbol{X}/\omega_1) d\boldsymbol{X} + \\ & + \boldsymbol{P}(\omega_2) \int_{\omega_1} \boldsymbol{F}(\omega_2/\omega_1)\, \boldsymbol{P}(\boldsymbol{X}/\omega_2) d\boldsymbol{X} + \boldsymbol{P}(\omega_2) \int_{\omega_2} \boldsymbol{F}(\omega_2/\omega_2)\, \boldsymbol{P}(\boldsymbol{X}/\omega_2) d\boldsymbol{X} \end{aligned} \tag{36}$$

În expresia de mai sus $\boldsymbol{F}(\omega_i,\omega_i)$ este valoarea funcţiei de pierdere pentru cazul clasificării corecte, şi ca urmare, pentru simplificarea calculelor, se poate impune $\boldsymbol{F}(\omega_i,\omega_i) = 0$, de unde obţinem:

$$r = \boldsymbol{P}(\omega_1) \int_{\omega_2} \boldsymbol{F}(\omega_1/\omega_2)\, \boldsymbol{P}(\boldsymbol{X}/\omega_1) d\boldsymbol{X} + \boldsymbol{P}(\omega_2) \int_{\omega_1} \boldsymbol{F}(\omega_2/\omega_1)\, \boldsymbol{P}(\boldsymbol{X}/\omega_2) d\boldsymbol{X} \tag{37}$$

Se urmăreşte construirea unui clasificator binar în condiţiile minimizării riscului global dat de expresia anterioară. Dacă clasificatorul dorit este unul linear, el este descris de vectorul de ponderi $\boldsymbol{W}$ (care dă normala la hiperplanul de decizie din spaţiul extins al trăsăturilor), regula de decizie fiind dată în acest caz de:

$$\begin{cases} \boldsymbol{X} \in \omega_1 \Leftrightarrow \boldsymbol{W}^T \boldsymbol{X} > 0 \\ \boldsymbol{X} \in \omega_2 \Leftrightarrow \boldsymbol{W}^T \boldsymbol{X} < 0 \end{cases} \tag{38}$$

Condiţia de linearitate impusă clasificatorului nu este restrictivă. Ţinând cont de dependenţele existente, trebuie făcută înlocuirea:

$$\boldsymbol{F}(\omega_1/\omega_2) \leftrightarrow \boldsymbol{F}(\omega_1/\omega_2, \boldsymbol{X}, \boldsymbol{W}) \tag{39}$$

şi atunci avem:

$$\boldsymbol{r} = \boldsymbol{P}(\omega_1)\int_{\omega_2}\boldsymbol{F}(\omega_1/\omega_2,\boldsymbol{X},\boldsymbol{W})\,\boldsymbol{P}(\boldsymbol{X}/\omega_1)d\boldsymbol{X} + \boldsymbol{P}(\omega_2)\int_{\omega_1}\boldsymbol{F}(\omega_2/\omega_1,\boldsymbol{X},\boldsymbol{W})\,\boldsymbol{P}(\boldsymbol{X}/\omega_2)d\boldsymbol{X} \tag{40}$$

Planul de decizie, deci parametrii clasificatorului trebuiesc determinaţi din condiţia minimizării riscului global, adică:

$$\nabla \boldsymbol{r}(\boldsymbol{W}) = 0\,, \quad \text{unde} \quad \nabla = \left(\frac{\partial}{\partial w_1} \quad \frac{\partial}{\partial w_2} \quad \ldots \quad \frac{\partial}{\partial w_{n+1}}\right) \tag{41}$$

Calculând gradientul riscului global, rezultă expresia:

$$\nabla \boldsymbol{r}(\boldsymbol{W}) = \boldsymbol{P}(\omega_1)\int_{\omega_2}\nabla\boldsymbol{F}(\omega_1/\omega_2,\boldsymbol{X},\boldsymbol{W})\,\boldsymbol{P}(\boldsymbol{X}/\omega_1)d\boldsymbol{X}$$
$$+\,\boldsymbol{P}(\omega_2)\int_{\omega_1}\nabla\boldsymbol{F}(\omega_2/\omega_1,\boldsymbol{X},\boldsymbol{W})\,\boldsymbol{P}(\boldsymbol{X}/\omega_2)d\boldsymbol{X} + \boldsymbol{g}(\omega_2,\omega_1) \tag{42}$$

unde ultimul termen este datorat dependenţei gradientului de limitele de separare a claselor. Impunând ca funcţia de pierdere F să fie nulă pe aceste limite de separare, ultimul termen se anulează.

Funcţiile de pierdere care pot fi folosite în construcţia neparametrică a clasificatorilor bayesieni sunt enumerate în continuare:

(1). Funcţia de pierdere simetrică:

$$\boldsymbol{F}(\omega_k/\omega_j) = \begin{cases} 0, \text{ pentru } j = k \\ 1, \text{ pentru } j \neq k \end{cases} \tag{43}$$

(2). Funcţia de pierdere diagonală:

$$\boldsymbol{F}(\omega_k/\omega_j) = \begin{cases} -h_k, \text{ pentru } j = k \\ 0, \quad \text{pentru } j \neq k \end{cases} \tag{44}$$

(3). Funcţii de pierdere dependente de distanţa între forma clasificată incorect şi clasa corectă:

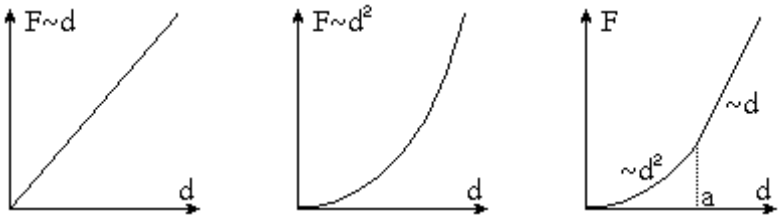


Fig. 1. Funcţii de pierdere dependente de distanţă.

Distanţa se măsoară de la planul de decizie la forma clasificată incorect folosind:

$$d = \frac{\boldsymbol{W}^T\boldsymbol{X}}{\|\boldsymbol{W}\|} = \frac{\boldsymbol{W}^T\boldsymbol{X}}{\left(\sum_{i=1}^{n+1} w_i^2\right)^{1/2}} \tag{45}$$

Dacă $d < 0$, înseamnă că forma a fost clasificată corect, deci în acest caz avem:

$$\boldsymbol{F}(d) = 0, \text{ pentru } d < 0 \tag{46}$$

Pentru o funcţie de pierdere dependentă liniar de distanţa la planul de decizie, avem:

$$\boldsymbol{F}(\omega_1/\omega_2,d) = \begin{cases} 0, & \text{pentru } \boldsymbol{W}^T\boldsymbol{X} \geq 0,\ \boldsymbol{X} \in \omega_1, \text{ clasificare corectă} \\ \dfrac{\boldsymbol{W}^T\boldsymbol{X}}{\|\boldsymbol{W}\|}, & \text{pentru } \boldsymbol{W}^T\boldsymbol{X} < 0,\ \boldsymbol{X} \in \omega_1, \text{ clasificare incorectă} \end{cases} \tag{47}$$

$$\boldsymbol{F}(\omega_2/\omega_1,d)=\begin{cases}0, & \text{pentru } \boldsymbol{W}^T\boldsymbol{X}\le 0,\ \boldsymbol{X}\in\omega_2, \text{ clasificare corectă}\\ \dfrac{\boldsymbol{W}^T\boldsymbol{X}}{\|\boldsymbol{W}\|}, & \text{pentru } \boldsymbol{W}^T\boldsymbol{X}>0,\ \boldsymbol{X}\in\omega_2, \text{ clasificare incorectă}\end{cases} \tag{48}$$

Se calculează:

$$\nabla\left(\frac{\boldsymbol{W}^T\boldsymbol{X}}{\|\boldsymbol{W}\|}\right)=\frac{\boldsymbol{X}\|\boldsymbol{W}\|-\boldsymbol{W}^T\boldsymbol{X}\nabla\|\boldsymbol{W}\|}{\|\boldsymbol{W}\|^2}=\frac{\boldsymbol{X}\|\boldsymbol{W}\|-\boldsymbol{W}^T\boldsymbol{X}\nabla\left(\boldsymbol{W}^T\boldsymbol{W}\right)^{1/2}}{\|\boldsymbol{W}\|^2}=$$

$$=\frac{\boldsymbol{X}\|\boldsymbol{W}\|-\boldsymbol{W}^T\boldsymbol{X}\left(\boldsymbol{W}^T\boldsymbol{W}\right)^{-1/2}\boldsymbol{W}}{\|\boldsymbol{W}\|^2}=\frac{\boldsymbol{X}\|\boldsymbol{W}\|-\boldsymbol{W}^T\boldsymbol{X}\|\boldsymbol{W}\|^{-1}\boldsymbol{W}}{\|\boldsymbol{W}\|^2}=\frac{\boldsymbol{X}\|\boldsymbol{W}\|^2-(\boldsymbol{W}^T\boldsymbol{X})\boldsymbol{W}}{\|\boldsymbol{W}\|^3}$$

(49)

iar rezultatul se notează cu:

$$\boldsymbol{f}(X,\boldsymbol{W})\overset{\Delta}{=}\frac{\boldsymbol{X}\|\boldsymbol{W}\|^2-(\boldsymbol{W}^T\boldsymbol{X})\boldsymbol{W}}{\|\boldsymbol{W}\|^3} \tag{50}$$

Ca urmare, gradientul riscului global poate fi exprimat cu:

$$\nabla\boldsymbol{r}(\boldsymbol{W})=\boldsymbol{P}(\omega_1)\int_{\omega_2}\boldsymbol{f}(\boldsymbol{X},\boldsymbol{W})\,\boldsymbol{P}(\boldsymbol{X}/\omega_1)d\boldsymbol{X}+\boldsymbol{P}(\omega_2)\int_{\omega_1}\boldsymbol{f}(\boldsymbol{X},\boldsymbol{W})\,\boldsymbol{P}(\boldsymbol{X}/\omega_2)d\boldsymbol{X}+\boldsymbol{g}(\omega_2,\omega_1) \tag{51}$$

Aceste metode fiind neparametrice, caracteristicile statistice ale procesului sunt necunoscute, deci $\boldsymbol{P}(\omega_i)$, $\boldsymbol{P}(\boldsymbol{X}/\omega_1)$ nu se cunosc. De aceea se adoptă o procedură iterativă de estimare a lui "$\boldsymbol{W}$", riscul global estimându-se şi el plecând de la formele din setul de antrenament:

$$\begin{cases}\boldsymbol{W}(i+1)=\boldsymbol{W}(i)+\Delta\boldsymbol{W}(i)\\ \boldsymbol{r}(\boldsymbol{W})\cong\dfrac{1}{N_1}\displaystyle\sum_{m=1}^{N_1}\boldsymbol{F}(\omega_1/\omega_2,\boldsymbol{X}_m^{(1)},\boldsymbol{W})+\frac{1}{N_2}\sum_{m=1}^{N_2}\boldsymbol{F}(\omega_2/\omega_1,\boldsymbol{X}_m^{(2)},\boldsymbol{W})\end{cases} \tag{52}$$

Metoda gradientului descrescător furnizează condiția ca $\Delta\boldsymbol{W}(i)$ să fie proporţional cu $\nabla\boldsymbol{r}(\boldsymbol{W})$, deci să avem:

$$\boldsymbol{W}(i+1)=\boldsymbol{W}(i)-\alpha\nabla\boldsymbol{r}(\boldsymbol{W}) \tag{53}$$

şi pe baza formulei precedente se poate utiliza drept estimaţie a gradientului:

$$\nabla\boldsymbol{r}(\boldsymbol{W})\cong\frac{1}{N_1}\sum_{m\in M_1}\boldsymbol{f}(\boldsymbol{X}_m^{(1)},\boldsymbol{W}(i))+\frac{1}{N_2}\sum_{m\in M_2}\boldsymbol{f}(\boldsymbol{X}_m^{(2)},\boldsymbol{W}(i)) \tag{54}$$

unde cu M_1,M_2 s-au notat mulţimile indicilor care, la iteraţia "i" produc o clasificare corectă.

Generalizarea metodei precedente pentru cazul mai multor clase este imediată.

11.4. Clasificatorul Bayes pentru forme codate binar.

Dacă forma $\boldsymbol{X}$ este codificată binar, trăsăturile $x_i, i=1...n$ pot lua deci doar două valori. Atunci $\boldsymbol{P}(\boldsymbol{X}/\omega_k)$ va fi definită de "n" probabilităţi de forma $\boldsymbol{p}(x_i/\omega_k)$, fiecare reprezentând probabilitatea ca trăsătura x_i să fie "1" pentru toate formele din clasa ω_k.

Fiind doar două valori posibile, $1-\boldsymbol{p}(x_i/\omega_k)$ reprezintă probabilitatea ca trăsătura x_i să fie "0" pentru toate formele din clasa ω_k.

Presupunând ***independența statistică*** a trăsăturilor, probabilitatea ca "$\boldsymbol{X}$" să aparţină clasei ω_k se scrie:

$$\boldsymbol{P}(\boldsymbol{X}/\omega_k) = \prod_{i=1}^{n} [\, \boldsymbol{p}(x_i/\omega_k)]^{x_i} [\, 1 - \boldsymbol{p}(x_i/\omega_k)]^{1-x_i} \tag{55}$$

Înlocuind relaţia anterioară în expresia funcţiei discriminant pentru un clasificator bayesian cu funcţie de pierdere simetrică:

$$\boldsymbol{D}_k(\boldsymbol{X}) = \boldsymbol{log}\,[\, \boldsymbol{P}(\boldsymbol{X}/\omega_k)P(\omega_k)] \tag{56}$$

rezultă succesiv:

$$\boldsymbol{D}_k(\boldsymbol{X}) = \sum_{i=1}^{n} \{\, \boldsymbol{log}\,[\, \boldsymbol{p}(x_i/\omega_k)]^{x_i} + \boldsymbol{log}\,[\, 1 - \boldsymbol{p}(x_i/\omega_k)]^{1-x_i} \} + \boldsymbol{log}\,\boldsymbol{P}(\omega_k) \tag{57}$$

$$\boldsymbol{D}_k(\boldsymbol{X}) = \sum_{i=1}^{n} x_i\, \boldsymbol{log}\, \boldsymbol{p}(x_i/\omega_k) + \sum_{i=1}^{n} (1 - x_i)\, \boldsymbol{log}\,[\, 1 - \boldsymbol{p}(x_i/\omega_k)] + \boldsymbol{log}\,\boldsymbol{P}(\omega_k) \tag{58}$$

$$\boldsymbol{D}_k(\boldsymbol{X}) = \sum_{i=1}^{n} x_i\, \boldsymbol{log}\, \frac{\boldsymbol{p}(x_i/\omega_k)}{1 - \boldsymbol{p}(x_i/\omega_k)} + \sum_{i=1}^{n} \boldsymbol{log}\,[\, 1 - \boldsymbol{p}(x_i/\omega_k)] + \boldsymbol{log}\,\boldsymbol{P}(\omega_k) \tag{59}$$

Folosind notaţiile:

$$w_i^{(k)} \overset{\Delta}{=} \boldsymbol{log}\, \frac{\boldsymbol{p}(x_i/\omega_k)}{1 - \boldsymbol{p}(x_i/\omega_k)} \qquad w_{n+1}^{(k)} \overset{\Delta}{=} \sum_{i=1}^{n} \boldsymbol{log}\,[\, 1 - \boldsymbol{p}(x_i/\omega_k)] + \boldsymbol{log}\,\boldsymbol{P}(\omega_k) \tag{60}$$

relaţia obţinută anterior se scrie:

$$\boldsymbol{D}_k(\boldsymbol{X}) = \sum_{i=1}^{n+1} w_i x_i = \boldsymbol{W}^T \boldsymbol{X} \tag{61}$$

unde $x_{n+1} = 1$. Se obţine deci, în ipoteza independenţei statistice a trăsăturilor şi pentru o funcţie de pierderi simetrică, un clasificator liniar.

Pentru cazul clasificării binare, funcţia de decizie este:

$$\boldsymbol{D}(\boldsymbol{X}) = \boldsymbol{D}_1(\boldsymbol{X}) - \boldsymbol{D}_2(\boldsymbol{X}) \tag{62}$$

şi înlocuind:

$$\boldsymbol{D}(\boldsymbol{X}) = \sum_{i=1}^{n} x_i \left[\boldsymbol{log}\, \frac{\boldsymbol{p}(x_i/\omega_1)}{1 - \boldsymbol{p}(x_i/\omega_1)} - \boldsymbol{log}\, \frac{\boldsymbol{p}(x_i/\omega_2)}{1 - \boldsymbol{p}(x_i/\omega_2)} \right] + \sum_{i=1}^{n} \boldsymbol{log}\, \frac{1 - \boldsymbol{p}(x_i/\omega_1)}{1 - \boldsymbol{p}(x_i/\omega_2)} + \boldsymbol{log}\, \frac{\boldsymbol{P}(\omega_1)}{\boldsymbol{P}(\omega_2)} \tag{63}$$

rezultă în final:

$$\boldsymbol{D}(\boldsymbol{X}) = \sum_{i=1}^{n} x_i\, \boldsymbol{log}\, \frac{\boldsymbol{p}(x_i/\omega_1)[\, 1 - \boldsymbol{p}(x_i/\omega_2)]}{\boldsymbol{p}(x_i/\omega_2)[\, 1 - \boldsymbol{p}(x_i/\omega_1)]} + \sum_{i=1}^{n} \boldsymbol{log}\, \frac{1 - \boldsymbol{p}(x_i/\omega_1)}{1 - \boldsymbol{p}(x_i/\omega_2)} + \boldsymbol{log}\, \frac{\boldsymbol{P}(\omega_1)}{\boldsymbol{P}(\omega_2)} \tag{64}$$

În cazurile în care nu se cunoaşte probabilitatea apriorică a claselor se aplică criteriul verosimilităţii maxime, care implică presupunerea că toate clasele sunt egal probabile, aceasta pentru a evita estimările eronate ale acestor probabilităţi plecând de la numărul de forme din setul de antrenare.

11.5. Aproximarea densităţilor de probabilitate.

Se disting două situaţii, cum ar fi:

(1). Se cunoaşte tipul distribuţiei respectate de formele din setul de antrenament, dar nu se cunosc parametrii acestei distribuţii.

(2). Nu se cunoaşte tipul distribuţiei.

În primul caz, cel mai adesea, parametrii care caracterizează distribuţia sunt media, care se estimează simplu prin:

$$\boldsymbol{M}^{(k)} = \frac{1}{N_k} \sum_{i=1}^{N_k} \boldsymbol{X}_i^{(k)} \tag{65}$$

şi dispersia, estimată prin:

$$\Phi^{(k)} = \sqrt{\frac{1}{N_k}\sum_{i=1}^{N_k}(X_i^{(k)} - M^{(k)})^2} \tag{66}$$

Cazul secund poate fi rezolvat prin trei modalităţi, primele două necesitând calculul histogramei de distribuţie pentru fiecare trăsătură în parte. Foarte utilă este asigurarea prealabilă a independenţei statistice a trăsăturilor. De aceea se utilizează transformarea Karhunen-Loeve care furnizează la ieşire valori necorelate sau, dacă vectorii de intrare pot fi modelaţi cu un proces Markov de ordinul unu, se poate utiliza cu foarte bune rezultate Transformarea Cosinus Discretă (DCT), care aproximează foarte bine transformarea KL.

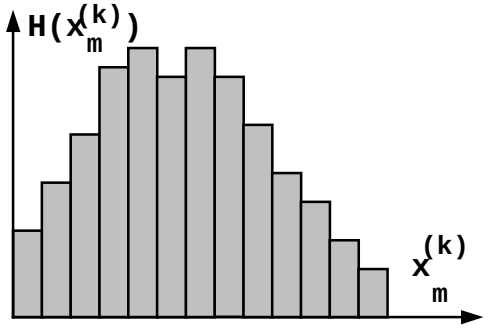


Fig. 2. Construcţia histogramei de distribuţie.

O primă soluţie este aproximarea acestei histograme de distribuţie cu o funcţie analitică, aproximare pentru care există instrumente matematice foarte bine puse la punct. Cealaltă soluţie, mai economică din punct de vedere computaţional, propune păstrarea acestor histograme de distribuţie sub forma tabelară, uzual reţinându-se 3-20 valori.

Ambele soluţii furnizează deci o estimaţie a densităţii de probabilitate $\varphi_i^{(k)}$ pentru trăsătura "*m*" şi clasa "*k*". Independenţa statistică a trăsăturilor permite calculul unei estimări a densităţii de repartiţie multidimensionale folosind relaţia:

$$\Theta^{(k)}(X) = \prod_{i=1}^{n}\varphi_i^{(k)}(x_i^{(k)}) \tag{67}$$

Ultima soluţie are la bază aproximarea densităţilor de repartiţie multidimensionale pentru fiecare clasă în parte folosind metoda funcţiilor de potenţial, metodă ce va fi discutată în cadrul capitolului următor.

12. Metoda funcţiilor de potenţial.

12.1. Construcţia clasificatorilor bazaţi pe funcţii de potenţial.

Această metodă constă în construirea unui "câmp potenţial" în jurul fiecărui punct (formă) cunoscut din spaţiul formelor. Apoi se calculează câmpul potenţial global al fiecărei clase folosind (după metoda potenţialelor electrice din electrotehnică) principiul superpoziţiei.

O formă de intrare necunoscută va fi atribuită acelei clase care exercită asupra ei câmpul cel mai puternic.

Numim funcţia de potenţial corespunzătoare unei forme cunoscute din clasa ω_k, o funcţie:

$$\boldsymbol{\Phi}(\boldsymbol{X},\boldsymbol{X}_i^{(k)}):\Omega \to R \qquad (1)$$

În general, pentru ca o funcţie oarecare să poată fi funcţie de potenţial, ea trebuie să îndeplinească următoarele condiţii:

(1). Să admită maxime pentru:

$$\boldsymbol{X} = \boldsymbol{X}_m^{(k)}, \forall m = 1...N_m, \forall k = 1...N_c \qquad (2)$$

(2). Să fie neglijabilă la distanţe mari de $\mathbf{X}_m^{(k)}$:

$$\boldsymbol{\Phi}(\boldsymbol{X},\boldsymbol{X}_i^{(k)}) \to 0, \text{ pentru } \boldsymbol{d}(\boldsymbol{X},\boldsymbol{X}_i^{(k)}) \to \infty \qquad (3)$$

(3). Să fie continuă şi să scadă monoton cu distanţa.

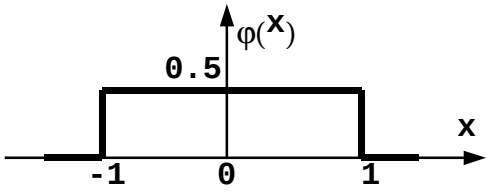
(4). Dacă $\boldsymbol{\Phi}(\boldsymbol{X}_1,\boldsymbol{X}_i^{(k)}) = \boldsymbol{\Phi}(\boldsymbol{X}_2,\boldsymbol{X}_i^{(k)})$, atunci formele $\boldsymbol{X}_1$ şi $\boldsymbol{X}_2$ trebuie să aibă aproximativ acelaşi grad de similaritate în raport cu $\boldsymbol{X}_m^{(k)}$.

Funcţia de potenţial poate fi diferită de la o formă la alta în interiorul aceleiaşi clase, funcţie de influenţa fiecărei forme asupra procesului de recunoaştere, dar şi diferită de la o clasă la alta. Scăderea funcţiei de potenţial monoton cu distanţa dintre $\boldsymbol{X}$ şi $\boldsymbol{X}_m^{(k)}$ reflectă scăderea influenţei câmpului asupra formelor noi mai depărtate.

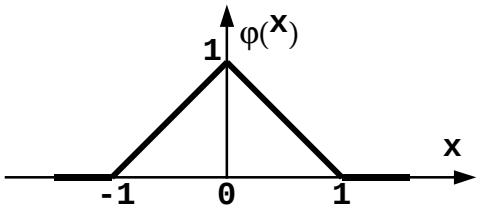
Câteva exemple de funcţii de potenţial simple sunt enumerate în continuare, pentru cazul unidimensional:

(1). funcţia de potenţial dreptunghiulară:

$$\varphi(x) = \begin{cases} 0.5, \text{ pentru } |x| \le 1 \\ 0, \quad \text{pentru } |x| > 1 \end{cases} \qquad (4)$$



(2). funcția de potențial triunghiulară:
$$\varphi(x) = \begin{cases} 1-|x|, \text{pentru } |x| \le 1 \\ 0, \quad \text{pentru } |x| > 1 \end{cases} \tag{5}$$



(3). funcția de potențial gaussiană:
$$\varphi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \tag{6}$$

(4). funcția de potențial tip descreştere exponențială:
$$\varphi(x) = \frac{1}{2} e^{-|x|}$$

(7)

(5). funcția de potențial tip Cauchy:
$$\varphi(x) = \frac{1}{\pi(1+x^2)} \tag{8}$$

Se observă similitudinile care există, ca expresie, între funcția de potențial şi densitatea de probabilitate. În practică, funcțiile de potențial se folosesc mult pentru estimarea densității de probabilitate.

Cunoscând funcțiile de potențial ataşate tuturor formelor prototip din fiecare clasă, putem construi o funcție discriminant ataşată unei clase, prin calculul câmpului potențial global:
$$\boldsymbol{D}_k(\boldsymbol{X}) = \frac{1}{M_k} \sum_{i=1}^{M_k} \boldsymbol{\Phi}(\boldsymbol{X}, \boldsymbol{X}_i^{(k)}) \tag{9}$$

Decizia se ia atunci după regula cunoscută:
$$\boldsymbol{X} \in \omega_k \Leftrightarrow \boldsymbol{D}_k(\boldsymbol{X}) > \boldsymbol{D}_j(\boldsymbol{X}),\ \forall j \ne k, j = 1...N_c \tag{10}$$

Pentru cazul clasificării binare (două clase), funcția discriminant este de forma:
$$\boldsymbol{D}(\boldsymbol{X}) = \frac{1}{M_1} \sum_{i=1}^{M_1} \boldsymbol{\Phi}(\boldsymbol{X}, \boldsymbol{X}_i^{(1)}) - \frac{1}{M_2} \sum_{i=1}^{M_2} \boldsymbol{\Phi}(\boldsymbol{X}, \boldsymbol{X}_i^{(2)}) \tag{11}$$

Planul de decizie fiind dat de punctele de potențial nul, regula de decizie în acest caz este de forma:
$$\begin{cases} \boldsymbol{X} \in \omega_1 \Leftrightarrow \boldsymbol{D}(\boldsymbol{X}) > 0 \\ \boldsymbol{X} \in \omega_2 \Leftrightarrow \boldsymbol{D}(\boldsymbol{X}) < 0 \end{cases} \tag{12}$$

Dacă funcția de potențial folosită este cea gaussiană multidimensională (foarte des utilizată), atunci:
$$\boldsymbol{\Phi}(\boldsymbol{X}, \boldsymbol{X}_i^{(k)}) = \boldsymbol{exp}\left[-(\boldsymbol{X} - \boldsymbol{X}_i^{(k)})^T (\boldsymbol{X} - \boldsymbol{X}_i^{(k)}) \right] \tag{13}$$

Dezvoltând expresia anterioară obținem:
$$\boldsymbol{\Phi}(\boldsymbol{X}, \boldsymbol{X}_i^{(k)}) = \boldsymbol{exp}\left[-\|\boldsymbol{X}\|^2 \right] \boldsymbol{exp}\left[2\boldsymbol{X}^T \boldsymbol{X}_i^{(k)} - \|\boldsymbol{X}_i^{(k)}\|^2 \right] \tag{14}$$

primul factor fiind comun tuturor funcțiilor de potențial, se foloseşte:
$$\boldsymbol{\Phi}(\boldsymbol{X}, \boldsymbol{X}_i^{(k)}) = \boldsymbol{exp}\left[2\boldsymbol{X}^T \boldsymbol{X}_i^{(k)} - \|\boldsymbol{X}_i^{(k)}\|^2 \right] \tag{15}$$

După definirea funcţiilor de potenţial pentru fiecare clasă în parte, ele pot fi apoi modificate aditiv sau multiplicativ, pentru a accentua sau diminua influenţa unui anume câmp.

O altă variantă permite evaluarea funcţiilor discriminant după metoda celor "*k*" vecini, astfel:

(1). se determină cei mai apropiaţi "*k*" vecini ai formei de intrare în clasificator.

(2). se sortează pe clase aceşti "*k*" vecini.

(3). se estimează potenţialul global al fiecărei clase doar pe baza potenţialului celor câţiva vecini.

(4). se execută clasificarea după regula descrisă anterior.

12.2. Aproximarea densităţilor de probabilitate cu funcţii de potenţial.

Estimarea densităţilor de probabilitate ale claselor poate fi făcută folosind metoda funcţiilor de potenţial, astfel:

$$\boldsymbol{p(X/\omega_k)} = \frac{1}{M_k}\sum_{m=1}^{M_k}\boldsymbol{\Phi(X,X_m^{(k)})} \tag{16}$$

Pentru ca $\boldsymbol{p(X/\omega_k)}$ să fie o densitate de probabilitate, funcţia $\boldsymbol{\Phi(X,X_m^{(k)})}$ trebuie să îndeplinească următoarele condiţii:

$$(1).\ \boldsymbol{sup}\left|\boldsymbol{\Phi(X)}\right| < +\infty \tag{17}$$

$$(2).\ \int_{-\infty}^{+\infty}\left|\boldsymbol{\Phi(X)}\right|d\boldsymbol{X} < +\infty \tag{18}$$

$$(3).\ \int_{-\infty}^{+\infty}\boldsymbol{\Phi(X)}d\boldsymbol{X} = 1 \tag{19}$$

$$(4).\ \lim_{x\to\infty} x\varphi(x) = 0 \tag{20}$$

Funcţiile de potenţial descrise mai sus îndeplinesc aceste condiţii.

Dacă se foloseşte funcţia potenţial gaussiană:

$$\boldsymbol{\Phi(X,X_m^{(k)})} = \frac{1}{(\sigma\sqrt{2\pi})^n}\boldsymbol{exp}\left[-\frac{(\boldsymbol{X}-\boldsymbol{X}_m^{(k)})^T(\boldsymbol{X}-\boldsymbol{X}_m^{(k)})}{2\sigma^2}\right] \tag{21}$$

unde parametrul "σ" este un factor de netezire, rezultă:

$$\boldsymbol{p(X/\omega_k)} = \frac{1}{M_k(\sigma\sqrt{2\pi})^n}\sum_{m=1}^{M_k}\boldsymbol{exp}\left[-\frac{(\boldsymbol{X}-\boldsymbol{X}_m^{(k)})^T(\boldsymbol{X}-\boldsymbol{X}_m^{(k)})}{2\sigma^2}\right] \tag{22}$$

$$\boldsymbol{p(X/\omega_k)} = \frac{1}{M_k(\sigma\sqrt{2\pi})^n}\boldsymbol{exp}\left[-\frac{\|\boldsymbol{X}\|^2}{2\sigma^2}\right]\sum_{m=1}^{M_k}\boldsymbol{exp}\left[\frac{\boldsymbol{X}^T\boldsymbol{X}_m^{(k)}}{\sigma^2}\right]\boldsymbol{exp}\left[-\frac{\|\boldsymbol{X}_m^{(k)}\|^2}{2\sigma^2}\right] \tag{23}$$

Dezvoltând în serii exponenţiala a doua, obţinem:

$$\boldsymbol{p(X/\omega_k)} \cong \frac{1}{M_k(\sigma\sqrt{2\pi})^n}\boldsymbol{exp}\left[-\frac{\|\boldsymbol{X}\|^2}{2\sigma^2}\right]\sum_{m=1}^{M_k}\boldsymbol{exp}\left[-\frac{\|\boldsymbol{X}_m^{(k)}\|^2}{2\sigma^2}\right]\sum_{h=1}^{p}\frac{1}{h!}\left(\frac{\boldsymbol{X}^T\boldsymbol{X}_m^{(k)}}{\sigma^2}\right)^h \tag{24}$$

Parametrul de netezire are o influenţă foarte mare asupra operaţiei de estimare a densităţii de probabilitate. Folosind formula anterioară, funcţia discriminant pentru clasa ω_k a unui clasificator bayesian cu funcţie de pierdere simetrică poate fi construită cu:

$$\boldsymbol{D_k(X)} = \frac{\boldsymbol{P}(\omega_k)}{M_k}\sum_{m=1}^{M_k}\boldsymbol{exp}\left[-\frac{\|\boldsymbol{X}_m^{(k)}\|^2}{2\sigma^2}\right]\sum_{h=1}^{p}\frac{1}{h!}\left(\frac{\boldsymbol{X}^T\boldsymbol{X}_m^{(k)}}{\sigma^2}\right)^h \tag{25}$$

unde primul raport şi prima exponenţială se calculează doar o singură dată, în faza de construcţie a clasificatorului.

13. Algoritmi de grupare (clustering).

13.1. Modalităţi de descriere a clusterilor.

Problemele de grupare a datelor intervin când se lucrează cu date neetichetate în setul de antrenament. Asemenea situaţii apar fie când, în interiorul unei clase deja definite, se urmăreşte structurarea informaţiei existente în scopul construcţiei eficiente a funcţiei discriminant corespunzătoare, fie când, necunoscând nimic despre structura internă a setului de date, se studiază grupările naturale care apar în urma interdependenţelor între aceste date.

Fiind dat setul de date de antrenament $\{\boldsymbol{x}_i\}_{i=1,...N} \subset \boldsymbol{\Omega}$, pe baza lui se urmăreşte construcţia grupărilor (clusterilor) $\{\boldsymbol{X}_i\}_{i=1,...,R}$, astfel încât:

$$\boldsymbol{X}_1 \cup \boldsymbol{X}_2 \cup ... \cup \boldsymbol{X}_R = \boldsymbol{\Omega} \quad \text{şi} \quad \boldsymbol{X}_i \cap \boldsymbol{X}_j = \varnothing, \forall i, j = 1...R, i \neq j \qquad (1)$$

Descrierea regiunilor (clusterilor) se poate face în mai multe moduri:

(1). Printr-un număr de hiperplane şi un număr de reguli logice care să localizeze clusterul în interiorul unei zone limitate de hiperplane.

(2). Prin coordonatele centrului clusterului şi o măsură a distanţei între formele aparţinând clusterului şi acest centru.

(3). Prin suprafeţe de proximitate descrise de ecuaţii de forma $\boldsymbol{f(x)} = 0$, măsura distanţei de la forma studiată la suprafaţă fiind dată de chiar $\boldsymbol{f(x)}$.

(4). Printr-un număr de forme prototip ataşate fiecărui cluster şi o măsură a distanţei între forma de intrare şi aceste forme prototip. Teoria funcţiilor de potenţial poate fi utilizată cu succes pentru deducerea măsurii acestei distanţe şi, evident, pentru construcţia unei funcţii discriminant între clase.

(5). Prin funcţii discriminant, regula de decizie rezultând din evaluarea acestora:

$$\boldsymbol{x} \in \boldsymbol{X}_j \Leftrightarrow \boldsymbol{D}_j(\boldsymbol{x}) > \boldsymbol{D}_i(\boldsymbol{x}), \forall i \neq j, i = 1...R \quad (2)$$

(6). Metode ierarhice, având de obicei la bază principiul dihotomiei, şi care realizează gruparea setului de forme de antrenament în două grupări pentru început, urmând ca apoi succesiv, să se subdividă aceste grupări în subgrupări de dimensiuni din ce în ce mai mici, până la obținerea clusterilor doriți. Decizia de apartenență se ia în urma parcurgerii arborelui binar (de obicei) care se obține astfel.

13.2. Gruparea prin descompunerea densităţlor de probabilitate.

O ***mixtură*** este o funcție densitate de probabilitate complexă care poate fi reprezentată prin superpoziția unor densități de probabilitate mai simple:

$$\boldsymbol{p}(\boldsymbol{x}) = \sum_{i=1}^{R} P_i \, \boldsymbol{p}^{(i)}(\boldsymbol{x}) \quad (3)$$

unde P_i are semnificația probabilității apriorice a clusterului "i", iar $\boldsymbol{p}^{(i)}(\boldsymbol{x})$ este densitatea de probabilitate elementară ataşată clusterului "i".

Se întâlnesc următoarele situații:

(1). Se cunoaşte legea de repartiție pentru clusterii a căror definire se doreşte, şi nici unul (sau numai o parte) din parametrii ce caracterizează aceste distribuții.

(2). Nu se cunoaşte legea de repartiție pentru aceşti clusteri.

De cele mai multe ori, fiecărui cluster îi corespunde un maxim local în mixtură. Asimilând aceste maxime locale (moduri) cu mediile variabilelor aleatoare ce generează mixtura, se obține o simplificare a problemei. Dacă, în plus, densitățile de probabilitate elementare se suprapun foarte puțin se poate scrie:

$$\boldsymbol{p}(\boldsymbol{x}) \cong P_i \, \boldsymbol{p}^{(i)}(\boldsymbol{x}), \text{ pentru } \boldsymbol{x} \in \boldsymbol{X}_i \quad (4)$$

Dacă nu poate fi făcută ultima supoziție, se foloseşte soluția determinării parametrilor necunoscuți prin minimizarea unei funcții de cost care evaluează diferențele între densitatea de probabilitate reală şi cea obținută prin superpoziția densităților elementare. Această minimizare se face de obicei iterativ. Problema poate fi abordată şi din punctul de vedere al teoriei funcțiilor de potențial, caz în care se estimează:

$$P_i \cong \frac{N_i}{N}, \qquad \text{iar} \qquad \boldsymbol{p}(\boldsymbol{x}) \cong \frac{1}{N} \sum_{i=1}^{R} N_i \, \boldsymbol{\theta}(\boldsymbol{x}, m_i) \quad (5)$$

unde $\boldsymbol{m}_i$ este modul de ordinul *"i"*, iar *"*$\boldsymbol{\theta}$*"* denotă funcția de potențial utilizată, urmărindu-se estimarea parametrilor intrinseci ai funcției de potențial.

Localizarea iterativă a clusterilor în cazul în care se cunosc modurile distribuției de probabilitate $\boldsymbol{m}_1^{(0)}, \boldsymbol{m}_2^{(0)}, ..., \boldsymbol{m}_R^{(0)}$ şi se dispune de o distanță $\boldsymbol{d}(\boldsymbol{x},\boldsymbol{y})$ în spațiul formelor, poate fi făcută folosind algoritmul de ajustare a centrelor ("center adjustment algorithm"):

(1). Atribuie succesiv fiecare formă "$\boldsymbol{y}$" din setul de antrenament clusterului al cărui centru este cel mai apropiat, folosind estimarea curentă a centrului lui, $\boldsymbol{m}_i^{(k)}$;

(2). Calculează noile centre ale fiecărui cluster, adică $\boldsymbol{m}_i^{(k+1)}$ care minimizează distanța medie între centru şi elementele care alcătuiesc clusterul:

$$\boldsymbol{Q}_i(\boldsymbol{m}) = \sum_{\boldsymbol{y} \in C_i^{(k)}} \boldsymbol{d}(\boldsymbol{y}, \boldsymbol{m}) \quad (6)$$

(3). Salt la pasul (1) pentru următoarea formă din setul de antrenament şi cu centrii ajustați în pasul (2) până când nu se mai schimbă de la o iterație la alta asignarea formelor la clase, asignare efectuată in pasul (1).

13.3. Algoritmi neparametrici de grupare cu număr predeterminat de clase.

Dacă se consideră clasificarea nesupervizată ca o metodă de grupare a claselor în scopul maximizării separabilității lor, atunci se pot construi diverse funcții de cost a căror maximizare/minimizare permite formarea clusterilor. Funcțiile de cost se construiesc de obicei pe baza următoarelor matrici:

$\boldsymbol{\Phi}_i = \boldsymbol{E}\{(\boldsymbol{x}^{(i)} - \bar{\boldsymbol{x}}^{(i)})(\boldsymbol{x}^{(i)} - \bar{\boldsymbol{x}}^{(i)})^T\}$ - matricea de covarianță a clasei "i";

$\boldsymbol{\Phi}^{(in)} = \sum_{i=1}^{M} \boldsymbol{P}(\omega_i)\boldsymbol{\Phi}_i$ - matricea de dispersie intraclasă;

$\boldsymbol{\Phi}^{(ic)} = \sum_{i=1}^{M} \boldsymbol{P}(\omega_i)(\boldsymbol{x}^{(i)} - \bar{\boldsymbol{x}}^{(i)})(\boldsymbol{x}^{(i)} - \bar{\boldsymbol{x}}^{(i)})^T$ - matricea de dispersie interclasă;

$\boldsymbol{\Phi}^{(m)} = \boldsymbol{\Phi}^{(in)} + \boldsymbol{\Phi}^{(ic)}$ - matricea de dispersie mixtă.

Funcțiile de cost utilizate cel mai des şi care se construiesc pe baza lor sunt:

$$\boldsymbol{C}_1 = \boldsymbol{tr}[(\boldsymbol{\Phi}^{(m)})^{-1}\boldsymbol{\Phi}^{(in)}] \quad \text{(minimizare)} \quad (7)$$

$$\boldsymbol{C}_2 = \boldsymbol{tr}[(\boldsymbol{\Phi}^{(in)})^{-1}\boldsymbol{\Phi}^{(m)}] \quad \text{(maximizare)} \quad (8)$$

$$\boldsymbol{C}_3 = \boldsymbol{ln}[|\boldsymbol{\Phi}^{(m)}|/|\boldsymbol{\Phi}^{(in)}|] \quad \text{(maximizare)} \quad (9)$$

Minimizarea/maximizarea lor permite construcția unei transformări optimale din familia transformărilor Karhunen-Loeve, transformări care realizează transpunerea datelor de intrare într-un spațiu în care separabilitatea claselor (clusterilor) este optimizată.

Volumul mare de calcule necesar pentru transformarea precedentă face ca, cel mai adesea, pentru gruparea claselor într-un număr predeterminat de clase să fie preferat un algoritm iterativ de construcție a clusterilor, numit algoritmul IZODATA:

(1). Se consideră o partiție inițială a formelor de intrare, $\boldsymbol{\Omega}^{(0)}$ şi pe baza ei se calculează mediile celor M clase, $\boldsymbol{m}_1^{(0)}, \boldsymbol{m}_2^{(0)}, ..., \boldsymbol{m}_M^{(0)}$, iar numărul iterației se inițializează cu $n = 0$.

(2). Folosind mediile $\boldsymbol{m}_1^{(n)}, \boldsymbol{m}_2^{(n)}, ..., \boldsymbol{m}_M^{(n)}$ rezultate în urma iterației "n", se reclasifică vectorul de intrare curent în clusterul corespunzător celei mai apropiate medii.

(3) Dacă există cel puțin o formă de intrare din setul de antrenament a cărei asignare este schimbată, se calculează noile medii $\boldsymbol{m}_1^{(n+1)}, \boldsymbol{m}_2^{(n+1)}, ..., \boldsymbol{m}_M^{(n+1)}$ şi se reia pasul (2). În caz contrar algoritmul se opreşte.

Se remarcă faptul că ajustarea mediilor se face la algoritmul IZODATA pentru fiecare eşantion, în timp ce algoritmul ajustării centrelor, descris anterior, realizează acest lucru după reclasificarea tuturor eşantioanelor. Din aceeaşi clasă de algoritmi mai face parte şi metoda minimizării sumei erorilor medii pătratice care atribuie o formă de intrare din setul de antrenament unui alt cluster, doar dacă scade eroarea medie pătratică dată de:

$$\varepsilon = \sum_{i=1}^{M} \sum_{\boldsymbol{x} \in \omega_p} \left\| \boldsymbol{x} - \boldsymbol{m}_p \right\|^2 \quad (10)$$

Algoritmul cuprinde următorii paşi:

Algoritmul IZODATA.

(1). Se alege o partiționare inițială a celor $\boldsymbol{N}$ eşantioane în clase (clusteri), calculându-se mediile corespunzătoare fiecărui cluster precum şi eroarea "$\boldsymbol{\varepsilon}$";
(2). Se inițializează iterația cu $n = 0$;
(3). $n = n + 1$
(4). Se extrage următoarea formă din setul de antrenament $\boldsymbol{x}_i \in \omega_p$. Dacă $N_p = 1$, adică $\boldsymbol{x}_i$ este singura formă asignată clasei ω_p, se continuă cu pasul (8);
(5). Se calculează:

$$\Delta\varepsilon_k = \begin{cases} \dfrac{N_q}{N_q + 1}\left\|\boldsymbol{x}_n - \boldsymbol{m}_q\right\|^2, \text{ pentru } q \neq p \\ \dfrac{N_p}{N_p - 1}\left\|\boldsymbol{x}_n - \boldsymbol{m}_p\right\|^2, \text{ pentru } q = p \end{cases} \quad (11)$$

(6). Se asignează $\boldsymbol{x}_n$ clasei ω_k dacă $\Delta\varepsilon_k \leq \Delta\varepsilon_q$, pentru toți "$q$";

(7). Se calculează "ε" şi mediile $\boldsymbol{m}_p$ şi $\boldsymbol{m}_q$.

(8). Dacă $n < N$, salt la pasul (3), în caz contrar salt la pasul (9).

(9). Stop dacă "ε" nu s-a modificat după ultima parcurgere a mulțimii de N eşantioane, în caz contrar salt la pasul (2).

13.4. Algoritmul L.B.G. (Linde-Buzo-Gray) de cuantizare vectorială.

Un algoritm intuitiv şi eficient pentru alegerea unui cuantizor vectorial (sau de bloc) definit pe baza unor măsuri de distorsiune relativ generale, a fost dezvoltat de Linde, Buzo şi Gray, el putând fi utilizat atât în descrieri de surse probabilistice cu caracteristici cunoscute, cât şi pentru studierea secvențelor lungi de date. Algoritmul este bazat pe o abordare datorată lui Lloyd, nu este o tehnică variațională şi nu implică diferențierea. Folosind tehnici variaționale comune, algoritmul produce cuantizori ce ating ***condiții necesare dar nu şi suficiente pentru optim***. Cel puțin la nivel local, optimul este asigurat.

13.4.1. Cuantizarea vectorială.

Un cuantizor pe N nivele (clase)**,** K-dimensional este o aplicație $\boldsymbol{q}$ care atribuie fiecărui vector de intrare $\boldsymbol{x}$:

$$\boldsymbol{x} = (x_0\ ,\ \ldots\ ,\ x_{K-1})\ , \quad (12)$$

un vector de refacere $\hat{\boldsymbol{x}}$:

$$\hat{\boldsymbol{x}} = \boldsymbol{q}(\boldsymbol{x})\ , \quad (13)$$

obținut dintr-un alfabet de refacere finit $\hat{\boldsymbol{A}}$:

$$\hat{\boldsymbol{A}} = \{y_i; i = 1,\ldots,N\} \quad (14)$$

Cuantizorul $\boldsymbol{q}$ este complet descris de alfabetul de refacere (cartea de coduri) $\hat{\boldsymbol{A}}$ împreună cu partiția spațiului vectorial de intrare $\boldsymbol{S}$:

$$\boldsymbol{S} = \{\boldsymbol{S}_i; i = 1,\ldots,N\}, \quad (15)$$

în seturile:

$$\boldsymbol{S}_i = \{\boldsymbol{x}; \boldsymbol{q}(\boldsymbol{x}) = y_i\} \quad (16)$$

ale aplicației vectorilor de intrare în vectorul de refacere $\boldsymbol{i}$ (cuvântul de cod $\boldsymbol{i}$). Asemenea cuantizori mai sunt denumiți şi cuantizori de bloc, cuantizori vectoriali sau coduri sursă de bloc.

13.4.2. Măsuri ale distorsiunii.

Presupunem că distorsiunea cauzată de refacerea unui vector de intrare $\boldsymbol{x}$ de către un vector de refacere $\hat{\boldsymbol{x}}$ este dată de o măsură nenegativă a distorsiunii $\boldsymbol{d}(\boldsymbol{x},\hat{\boldsymbol{x}})$. Multe asemenea măsuri ale distorsiunii au fost propuse în literatură. Cea mai uzuală, din motive de compatibilitate matematică, este eroarea medie pătratică a distorsiunii:

$$\boldsymbol{d}(\boldsymbol{x},\hat{\boldsymbol{x}}) = \sum_{i=0}^{K-1}\left|x_i - \hat{x}_i\right|^2\ , \quad (17)$$

Alte măsuri comune ale distorsiunii sunt l_ν sau norma Holder:

$$\boldsymbol{d}(\boldsymbol{x},\hat{\boldsymbol{x}}) = \left\{ \sum_{i=0}^{K-1} \left| x_i - \hat{x}_i \right|^{\nu} \right\}^{1/\nu} = \left\| \boldsymbol{x} - \hat{\boldsymbol{x}} \right\|_{\nu} \ , \tag{18}$$

şi puterea $\boldsymbol{\nu}$ a sa, distorsiunea de ordinul $\boldsymbol{\nu}$:

$$\boldsymbol{d}^{\nu}(\boldsymbol{x},\hat{\boldsymbol{x}}) = \sum_{i=0}^{K-1} \left| x_i - \hat{x}_i \right|^{\nu} = \left\| \boldsymbol{x} - \hat{\boldsymbol{x}} \right\|_{\nu}^{\nu} \ , \tag{19}$$

Cum ambele măsuri ale distorsiunii date prin relaţiile anterioare depind de puterea $\boldsymbol{\nu}$ a erorilor în coordonate separate, măsura din prima relaţie este adeseori mai folositoare, deoarece fiind o distanţă (sau metrică) şi satisfăcând inegalitatea triunghiului, permite limitarea distorsiunii totale mai uşor, într-un sistem cu mai mulţi paşi, prin sumarea distorsiunilor individuale întâlnite la fiecare treaptă, în timp ce distorsiunea de ordinul $\boldsymbol{\nu}$ din relaţia a doua nu are această proprietate.

Alte măsuri ale distorsiunii sunt l_{∞}, sau norma Minkowski:

$$\boldsymbol{d}(\boldsymbol{x},\hat{\boldsymbol{x}}) = \max_{0 \le i \le K-1} \left| x_i - \hat{x}_i \right| \ , \tag{20}$$

distorsiunea pătratică ponderată:

$$\boldsymbol{d}(\boldsymbol{x},\hat{\boldsymbol{x}}) = \sum_{i=0}^{K-1} w_i \left| x_i - \hat{x}_i \right|^2 \ , \tag{21}$$

unde $w_i \ge 0, i = 0,...,K-1$

şi o distorsiune mai generală, distorsiunea cuadratică:

$$\boldsymbol{d}(\boldsymbol{x},\hat{\boldsymbol{x}}) = (\boldsymbol{x} - \hat{\boldsymbol{x}})\boldsymbol{B}(\boldsymbol{x} - \hat{\boldsymbol{x}})^t = \sum_{i=0}^{K-1}\sum_{j=0}^{K-1} B_{i,j}(x_i - \hat{x}_i)(x_j - \hat{x}_j) \ , \tag{22}$$

unde $\boldsymbol{B} = \boldsymbol{B}\{i,j\}$ este o matrice KxK simetrică, pozitiv definită.

Toate măsurile descrise anterior pentru distorsiune au proprietatea că depind de vectorii $\boldsymbol{x}$ şi $\hat{\boldsymbol{x}}$ numai prin vectorul de eroare $\boldsymbol{x} - \hat{\boldsymbol{x}}$. Asemenea măsuri ale distorsiunii având forma:

$$\boldsymbol{d}(\boldsymbol{x},\hat{\boldsymbol{x}}) = \boldsymbol{L}(\boldsymbol{x} - \hat{\boldsymbol{x}}) \ , \tag{23}$$

sunt numite măsuri diferenţă ale distorsiunii. Măsuri ale distorsiunii care au această formă, dar depind de $\boldsymbol{x}$ şi $\hat{\boldsymbol{x}}$ într-un mod mai complicat, au fost propuse pentru sistemele de compresie de date.

Interesantă este măsura distorsiunii propusă de Itakura şi Saito şi de Chaffee care apare în sistemele de compresie vocală şi are forma:

$$\boldsymbol{d}(\boldsymbol{x},\hat{\boldsymbol{x}}) = (\boldsymbol{x} - \hat{\boldsymbol{x}})\,\boldsymbol{R}(\boldsymbol{x})(\boldsymbol{x} - \hat{\boldsymbol{x}})^t \ , \tag{24}$$

unde pentru fiecare $\boldsymbol{x}$, matricea $\boldsymbol{R}(\boldsymbol{x})$ cu dimensiunea KxK este simetrică şi pozitiv definită.

Această distorsiune seamănă cu distorsiunea pătratică anterioară, dar matricea de ponderare depinde de vectorul de intrare $\boldsymbol{x}$.

O măsură a distorsiunii $\boldsymbol{d}(\boldsymbol{x},\hat{\boldsymbol{x}})$ între $\boldsymbol{x}$ şi o refacerea $\hat{\boldsymbol{x}}$, ar putea fi văzută ca o măsură a distorsiunii între două filtre sau modele normalizate (cu câştig unitar). În acest caz o măsură a distorsiunii a fost propusă de Itakura şi Saito şi de Chaffee, ea fiind exprimată prin relaţia:

$$\boldsymbol{R}(\boldsymbol{x}) = \{ \boldsymbol{r}_x(k-j); k = 0,...,K-1; j = 0,...,K-1 \} \tag{25}$$

definită de:

$$\boldsymbol{r}_x(k) = \int_{-\pi}^{\pi} \left| \sum_{m=0}^{K} a_i e^{-i\,m\,\theta} \right|^{-2} e^{i\,k\,\theta}\, d\theta / 2\pi \ , \tag{26}$$

descrisă prin $\boldsymbol{x}$, când intrarea are un spectru plat de amplitudine, unitar (sistemele **LPC** standard îşi minimizează implicit această distorsiune, ceea ce sugerează că este o măsură potrivită a acesteia, pentru cuantizarea următoare).

Fie $\boldsymbol{X} = (X_0,...,X_{k-1})$ un vector real aleator descris de o funcție de distribuție cumulativă având forma:

$$\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{P}\{X_i \le x_i; i = 0,1,...,K-1\} \tag{27}$$

O măsură a performanței cuantizorului $\boldsymbol{q}$ aplicată vectorului aleator $\boldsymbol{X}$ este dată de distorsiunea așteptată:

$$\boldsymbol{D}(\boldsymbol{q}) = \boldsymbol{E}\{\boldsymbol{d}(\boldsymbol{X},\boldsymbol{q}(\boldsymbol{X}))\} \tag{28}$$

unde $\boldsymbol{E}$ denotă operatorul speranță matematică aplicat distribuției de mai sus $\boldsymbol{F}$.

Această măsură a performanței este semnificativă din punct de vedere fizic dacă cuantizorul $\boldsymbol{q}$ va fi folosit la cuantizarea unei secvențe de vectori:

$$\boldsymbol{X}_n = (X_{nk}, \dots, X_{nk+k-1}) \tag{29}$$

care sunt staționari şi ergodici, de vreme ce distorsiunea medie temporară:

$$n^{-1}\sum_{i=0}^{n-1}\boldsymbol{d}(X_i,\ \boldsymbol{q}(X_i)) \tag{30}$$

converge cu probabilitatea "*1*" către $\boldsymbol{D}(\boldsymbol{q})$ pentru $n \to \infty$ (din teoria ergodică), de unde rezultă că $\boldsymbol{D}(\boldsymbol{q})$ descrie distorsiunea medie (temporală) pentru rularea pe durată mare.

13.4.3. Cuantizarea optimală.

Un cuantizor pe $\boldsymbol{N}$ nivele va fi numit optimal (sau optimal global), dacă minimizează distorsiunea aşteptată, cu alte cuvinte, $\boldsymbol{q}^*$ este optimal dacă pentru toți ceilalți cuantizori $\boldsymbol{q}$ având $\boldsymbol{N}$ vectori de refacere avem:

$$\boldsymbol{D}(\boldsymbol{q}^*) \le \boldsymbol{D}(\boldsymbol{q}) \tag{31}$$

Un cuantizor va fi numit optim local dacă $\boldsymbol{D}(\boldsymbol{q})$ este doar un minim local, altfel spus, modificări uşoare ale lui $\boldsymbol{q}$ duc la o creştere a distorsiunii. Scopul proiectării cuantizorului de bloc este de a obține un cuantizor optimal, dacă este posibil, iar dacă nu, un cuantizor optim local. Câțiva asemenea algoritmi au fost propuşi în literatură pentru proiectarea asistată de calculator a cuantizorilor local optimali.

În 1957, într-o lucrare nepublicată, S. Lloyd a propus două metode de proiectare a cuantizorului pentru cazul scalar $(k = 1)$, cu un criteriu dat de distorsiunea eroare pătratică. "Metoda II" era o abordare variațională directă, în care a luat derivatele în raport cu simbolurile de refacere y_i şi cu punctele limită ce defineau S_i, făcând aceste derivate zero. Din aceasta rezultă în general doar un cuantizor "de punct staționar" (derivata zero multidimensională) care satisface condiții necesare dar nu şi suficiente pentru optimalitate. Folosind argumentele derivatei a-II-a, este uşor de stabilit că asemenea cuantizoare "de punct staționar" asigură măcar optime locale pentru măsuri folosind regula puterii $\boldsymbol{\nu}$.

Lloyd a demonstrat optimalitatea globală pentru anumite distribuții folosind tehnica căutării exhaustive a tuturor optimumurilor locale. În esență, aceeaşi tehnică a fost propusă şi utilizată în problema paralelă a analizei de grup de către Danelius în 1950, Fischer în 1953 şi Cox în 1957. Această tehnică a fost dezvoltată în mod independent de Max în 1960 iar cuantizorul rezultat este în mod uzual cunoscut sub denumirea de cuantizorul Lloyd-Max.

De remarcat că în câteva situații, abordarea variațională eşuează:

- dacă $\boldsymbol{k}$ nu este egal cu **1** sau **2**, nevoile de calcul devin prea complexe.
- dacă sunt dorite în plus măsuri ale distorsiunii mai complexe, puterea de calcul cerută de ecuațiile variaționale poate deveni exorbitantă.
- dacă distribuția de probabilități în cauză are componente discrete, atunci derivatele necesare nu există, dând naştere la probleme de calcul.

- dacă lipseşte o descriere probabilistică precisă a vectorului aleator $\boldsymbol{X}$ şi ne bazăm proiectarea pe o secvență lungă studiată, atunci nu există nici o modalitate evidentă de a aplica metoda variațională.

Pentru a descrie "Metoda 1" a lui Lloyd în cazul general, presupunem mai întâi că distribuția este cunoscută.

Fiind dat un cuantizor $\boldsymbol{q}$ descris de un alfabet de refacere $\hat{\boldsymbol{A}}$:

$$\hat{\boldsymbol{A}} = \{y_i; i=1,...,N\}\,, \tag{32}$$

şi partiția $\boldsymbol{S}$:

$$\boldsymbol{S} = \{S_i; i = 1,...,N\}, \tag{33}$$

atunci distorsiunea aşteptată de forma:

$$\boldsymbol{D}(\{\hat{\boldsymbol{A}},\boldsymbol{S}\}) = \boldsymbol{D}(\boldsymbol{q}) \tag{34}$$

poate fi scrisă ca:

$$\boldsymbol{D}([\hat{\boldsymbol{A}},\boldsymbol{S}]\} = \sum_{i=1}^{N} \boldsymbol{E}\{\boldsymbol{d}(\boldsymbol{X},y_i)|\boldsymbol{X} \in S_i\}\,\boldsymbol{P}(\boldsymbol{X} \in \boldsymbol{S}_i) \tag{35}$$

unde $\boldsymbol{E}\{\boldsymbol{d}(\boldsymbol{X},y_i)|\boldsymbol{X} \in \boldsymbol{S}_i\}$ este distorsiunea condiționată aşteptată, fiind dat $\boldsymbol{X} \in \boldsymbol{S}_i$, sau echivalent, fiind dat $\boldsymbol{q}(\boldsymbol{X}) = y_i$

Presupunem că avem dat un anume alfabet de refacere $\hat{\boldsymbol{A}}$, dar nu este specificată nici o partiție. O partiție optimă pentru $\hat{\boldsymbol{A}}$ poate fi uşor construită printr-o aplicație de la fiecare $\boldsymbol{x}$ la $\boldsymbol{y}_i \in \hat{\boldsymbol{A}}$, minimizând distorsiunea $\boldsymbol{d}(\boldsymbol{x},y_i)$, deci cu alte cuvinte, alegând distorsiunea minimă sau cel mai apropiat cuvânt de cod pentru fiecare intrare.

O regulă de departajare, cum ar fi alegerea refacerii cu cel mai mic index, este necesară în cazul în care mai mult de un cuvânt de cod minimizează distorsiunea.

Partiția notată:

$$\boldsymbol{P}(\hat{\boldsymbol{A}}) = \{\boldsymbol{P}_i; i = 1,...,N\} \tag{36}$$

construită în această manieră este astfel încât:

$$\boldsymbol{x} \in \boldsymbol{P}_i \text{ (sau echivalent } \boldsymbol{q}(\boldsymbol{x}) = y_i) \tag{37}$$

dacă şi numai dacă:

$$\boldsymbol{d}(\boldsymbol{x},y_i) \le \boldsymbol{d}(\boldsymbol{x},y_j) \tag{38}$$

oricare ar fi "j" şi de aici avem:

$$\boldsymbol{D}([\hat{\boldsymbol{A}},\boldsymbol{P}(\hat{\boldsymbol{A}})]) = \boldsymbol{E}(\min_{y \in \hat{A}}(\boldsymbol{d}(\boldsymbol{X},\boldsymbol{y}))) \tag{39}$$

care, la rândul ei, implică pentru orice partiție $\boldsymbol{S}$ ca:

$$\boldsymbol{D}([\hat{\boldsymbol{A}},\boldsymbol{S}]) \ge \boldsymbol{D}([\hat{\boldsymbol{A}},\boldsymbol{P}(\hat{\boldsymbol{A}})]) \tag{40}$$

Astfel, pentru un alfabet de refacere fixat $\hat{\boldsymbol{A}}$, cea mai bună partiție posibilă este $\boldsymbol{P}(\hat{\boldsymbol{A}})$.

Reciproc, să presupunem că avem o partiție $\boldsymbol{S} = \{\boldsymbol{S}_i; i = 1,...,N\}$ descriind cuantizorul. Pentru moment vom presupune că măsura distorsiunii şi distribuția sunt astfel încât pentru fiecare set $\boldsymbol{S}$ cu probabilitatea diferită de zero într-un spațiu euclidian $\boldsymbol{k}$-dimensional există un vector de minimă distorsiune $\hat{\boldsymbol{x}}(\boldsymbol{S})$ pentru care:

$$\boldsymbol{E}(\boldsymbol{d}(\boldsymbol{X},\hat{\boldsymbol{x}}(\boldsymbol{S}))\,\big|\,\boldsymbol{X} \in \boldsymbol{S}) = \min_{\boldsymbol{u}} \boldsymbol{E}(\boldsymbol{d}(\boldsymbol{X},\boldsymbol{u})\,\big|\,\boldsymbol{X} \in \boldsymbol{S}) \tag{41}$$

Analog cazului măsurii erorii pătratice a distorsiunii şi a distribuției de probabilitate uniformă, numim vectorul $\hat{\boldsymbol{x}}(\boldsymbol{S})$ centroidul sau centrul de greutate a setului $\boldsymbol{S}$. **D**acă asemenea puncte există, atunci în mod clar pentru o partiție fixată $\boldsymbol{S}$:

$$\boldsymbol{S} = \{\boldsymbol{S}_i; i = 1,...,N\} \tag{42}$$

nici un alfabet de refacere $\hat{\boldsymbol{A}}$:

$$\hat{\boldsymbol{A}} = \{y_i; i = 1,...,N\} \tag{43}$$

nu poate produce o distorsiune medie mai mică decât alfabetul de refacere $\hat{\boldsymbol{x}}(\boldsymbol{S})$:

$$\hat{\boldsymbol{x}}(\boldsymbol{S}) = \{\hat{\boldsymbol{x}}(\boldsymbol{S}_i); i = 1,...,N\} \tag{44}$$

conținând centroizii seturilor din $\boldsymbol{S}$, deoarece avem inegalitatea din relația:

$$\boldsymbol{D}([\hat{\boldsymbol{A}},\boldsymbol{S}]) = \sum_{i=1}^{N} \boldsymbol{E}(\boldsymbol{d}(\boldsymbol{X},y_i)\,|\,\boldsymbol{X} \in \boldsymbol{S}_i)\,\boldsymbol{Pr}(\boldsymbol{X} \in \boldsymbol{S}_i) \geq$$

$$\geq \sum_{i=1}^{N} \min_{u} \boldsymbol{E}(\boldsymbol{d}(\boldsymbol{X},y)\,|\,\boldsymbol{X} \in \boldsymbol{S}_i)\,\boldsymbol{Pr}(\boldsymbol{X} \in \boldsymbol{S}_i) = \boldsymbol{D}([\hat{\boldsymbol{X}}(\boldsymbol{S}),\boldsymbol{S}]) \tag{45}$$

Centroizii din relația anterioară există pentru toate seturile **S** cu probabilitate nenulă pentru măsuri ale distorsiunii relativ generale.

În particular, dacă $\boldsymbol{d}(\boldsymbol{x},\boldsymbol{y})$ este convex în $\boldsymbol{y}$, atunci centroizii pot fi calculați folosind tehnicile de programare convexe standard. În anumite cazuri, pot fi găsiți ușor folosind tehnici variaționale. Dacă probabilitatea unui set $\boldsymbol{S}$ este zero, atunci centroidul poate fi definit într-o manieră arbitrară, din moment ce speranța matematică dată de $\boldsymbol{S}$ nu are o definiție unică.

13.4.4. Algoritmul L.B.G. pentru distribuție cunoscută.

(0) Inițializarea : fie dat N = numărul de clase, un prag de distorsiune $\varepsilon \geq 0$, un alfabet de refacere inițial de clasă N notat $\hat{\boldsymbol{A}}_0$ şi o distribuție $\boldsymbol{F}$.
Facem $m = 0$ şi $D_{-1} = \infty$.

(1) Fiind dat $\hat{\boldsymbol{A}}_m = \{\boldsymbol{y}_i; i = 1,...,N\}$, găsim partiția de distorsiune minimă:
$\boldsymbol{P}(\hat{\boldsymbol{A}}_m) = \{\boldsymbol{S}_i; i = 1,...,N\}$ cu $\boldsymbol{x} \in \boldsymbol{S}_i$,
dacă $\boldsymbol{d}(\boldsymbol{x},y_i) \leq \boldsymbol{d}(\boldsymbol{x},y_j)$ oricare ar fi "j".
Calculăm distorsiunea medie rezultantă D_m de forma:

$$D_m = \boldsymbol{D}(\{\hat{\boldsymbol{A}}_m, \boldsymbol{P}(\hat{\boldsymbol{A}}_m)\}) = \boldsymbol{E}_{\min_{j \in \hat{A}_m}} \{\boldsymbol{d}(\boldsymbol{X},\boldsymbol{y})\} \tag{46}$$

(2) Dacă $(D_{m-1} - D_m)/D_m \leq \varepsilon$, oprire la $\hat{\boldsymbol{A}}_m$ şi $\boldsymbol{P}(\hat{\boldsymbol{A}}_m)$ ce descriu cuantizorul final.
Dacă $(D_{m-1} - D_m)/D_m > \varepsilon$, continuăm cu pasul **(3)**

(3) Căutăm alfabetul de refacere optimal de forma:
$\hat{\boldsymbol{x}}(\boldsymbol{P}(\hat{\boldsymbol{A}}_m)) = \{\hat{\boldsymbol{x}}(\boldsymbol{S}_i); i = 1,...,N\}$ pentru $\boldsymbol{P}(\hat{\boldsymbol{A}}_m)$.
Facem $\hat{\boldsymbol{A}}_{m+1} = \hat{\boldsymbol{x}}(\boldsymbol{P}(\hat{\boldsymbol{A}}_m))$.
Înlocuim m cu $m + 1$ şi reluăm de la pasul **(1)**.

Dacă într-un punct oarecare, partiția optimală $\boldsymbol{P}(\hat{\boldsymbol{A}}_m)$ are o celulă $\boldsymbol{S}_i$ astfel încât $\boldsymbol{Pr}(\boldsymbol{X} \in \boldsymbol{S}_i) = 0$, atunci algoritmul asignează ca centroid un vector arbitrar şi continuă.

O alternativă mai simplă este aceea că *dacă cuantizorul final produs de algoritm are probabilitatea zero pentru o celulă, atunci vom rula din nou algoritmul cu alte condiții inițiale.*

Deoarece avem $D_m \leq D_{m-1}$, de aici rezultă că fiecare iterație a algoritmului fie reduce distorsiunea, fie o lasă neschimbată. **C**um D_m este necrescător şi pozitiv, el trebuie să aibă o

limită inferioară, să zicem D_∞, pentru $m \to \infty$. Dacă un cuantizor limitat $\hat{\boldsymbol{A}}_\infty$ există în sensul Euclidian uzual $\hat{\boldsymbol{A}}_m \to \hat{\boldsymbol{A}}_\infty$ cu $m \to \infty$, atunci:

$$\boldsymbol{D}(\{\hat{\boldsymbol{A}}_\infty, \boldsymbol{P}(\hat{\boldsymbol{A}}_\infty)\}) = D_\infty \tag{47}$$

şi $\hat{\boldsymbol{A}}_\infty$ are proprietatea că:

$$\hat{\boldsymbol{A}}_\infty = \hat{\boldsymbol{x}}(\boldsymbol{P}(\hat{\boldsymbol{A}})), \tag{48}$$

cu alte cuvinte, $\hat{\boldsymbol{A}}_\infty$ este exact centroidul propriei sale partiții optimale (cuantizorul limită, dacă există, este numit cuantizor de punct fix).

O condiție necesară a unui cuantizor pentru a fi optimal, este aceea de a fi un cuantizor de punct fix. În cazul lui Lloyd, dacă un cuantizor de punct fix este astfel ales încât nu există nici o probabilitate pe frontiera celulelor de partiție, altfel spus, dacă:

$$\boldsymbol{Pr}[\boldsymbol{d}(\boldsymbol{X}, y_i)] = \boldsymbol{d}(\boldsymbol{X}, y_j), \text{ pentru un } i \neq j, \text{ oricare ar fi } i \neq 0, \tag{49}$$

atunci cuantizorul este optim local.

O tehnică similară a fost propusă în 1953 de Fischer în problema analizei de grup folosind deciziile lui Bayes cu privire la *costul* erorii pătratice.

În 1965, Forgy a propus algoritmul pentru analiza de grup în cazul multidimensional al erorii pătratice a distorsiunii şi o distribuție de eşantioane.

În 1977, Chen a propus în esență acelaşi algoritm pentru cazul multidimensional cu erorea pătratică drept măsură a distorsiunii şi l-a folosit la proiectarea cuantizoarelor bidimensionale pentru vectori uniform distribuiți în cerc.

Deoarece algoritmul nu pune nici un fel de condiții de diferențiabilitate, el este aplicabil pentru distribuții discrete pure. Acest fapt are o aplicabilitate importantă atunci când *nu se cunoaşte dinainte o descriere probabilistică* a sursei ce va fi comprimată, şi din acest motiv trebuie să ne bazăm în proiectare pe studiul unei secvențe lungi de date, ce urmează a fi compresată.

13.4.5. Algoritmul L.B.G. pentru distribuție necunoscută.

(0) Inițializarea: fie dat N = număr de clase, pragul distorsiunii $\varepsilon \geq 0$, un alfabet de refacere inițial de clasă N, notat cu $\hat{\boldsymbol{A}}_0$ şi o secvență studiată x_j de forma: $\{x_j; j = 0,...,n-1\}$.

Facem $m = 0$ şi $D_{-1} = \infty$. (50)

(1) Fie dat

$$\hat{\boldsymbol{A}}_m = \{\boldsymbol{y}_i; i = 1,...,N\}, \tag{51}$$

căutăm partiția de distorsiune minimă

$$P(\hat{\boldsymbol{A}}_m) = \{\boldsymbol{S}_i; i = 1,...,N\} \tag{52}$$

pentru secvența studiată $\boldsymbol{x}_j \in \boldsymbol{S}_i$ dacă:

$$d(\boldsymbol{x}_j, \boldsymbol{y}_i) \leq d(\boldsymbol{x}_j, \boldsymbol{y}_l), \tag{53}$$

oricare ar fi l.

Calculăm distorsiunea medie:

$$D_m = D(\{\hat{\boldsymbol{A}}_m, P(\hat{\boldsymbol{A}}_m)\}) = n^{-1} \sum_{j=0}^{n-1} \min_{\boldsymbol{y} \in \hat{\boldsymbol{A}}_m} d(\boldsymbol{x}_j, \boldsymbol{y}) \tag{54}$$

(2) Dacă avem $(D_{m-1} - D_m)/D_m \le \varepsilon$, (55)

ne oprim la $\hat{\boldsymbol{A}}_m$ ca alfabet de refacere final.

Dacă avem $(D_{m-1} - D_m)/D_m > \varepsilon$, (56)

continuăm cu pasul **(3)**.

(3) Căutăm alfabetul de refacere optimal:

$\hat{\boldsymbol{x}}(P(\hat{\boldsymbol{A}}_m)) = \{\hat{\boldsymbol{x}}(\boldsymbol{S}_i); i = 1,...,N\}$ pentru $P(\hat{\boldsymbol{A}}_m)$. (57)

Facem $\hat{\boldsymbol{A}}_{m+1} = \hat{\boldsymbol{x}}(P(\hat{\boldsymbol{A}}_m))$. (58)

Înlocuim m cu $m+1$ şi reluăm de la pasul **(1)**.

Dacă secvenţa de vectori aleatori este staţionară şi ergodică, atunci se respectă teorema ergodică, conform căreia cu probabilitatea unu, $\boldsymbol{G}_n$ converge spre distribuţia reală de bază, notată $\boldsymbol{F}$, pentru $n \to \infty$.

O tehnică similară folosită în analiza de grup cu funcţii de cost tip eroare medie pătratică a fost dezvoltată de McQueen în anul 1967, fiind numită metoda "***K-means***".

Altă tehnică, mult mai utilizată, folosind metoda "***K-means***" este metoda "*IZODATA*" a lui Ball şi Hall. Ideea principală, aceea de a găsi partiţiile de distorsiune minimă şi centroizii este aceeaşi, dar secvenţa de date studiată este folosită într-o manieră diferită, iar cuantizorii rezultaţi vor fi, în general, diferiţi. Tehnica lor este secvenţială, încorporează vectorii studiaţi unul câte unul şi se sfârşeşte când ultimul vector este încorporat. Aceasta este în contrast cu algoritmul anterior, care consideră toţi vectorii studiaţi pentru fiecare iteraţie.

Scopul metodei "***K-*** *means*" este de a produce o partiţie:

$$\boldsymbol{S}_0 = [\boldsymbol{S}_0, ..., \boldsymbol{S}_{N-1}] \tag{59}$$

a alfabetului studiat, cuprinzând toţi vectorii din secvenţa de studiu.

$$\boldsymbol{A} = \{\boldsymbol{x}_i; i = 0,...,n-1\} \tag{60}$$

Alfabetul de refacere corespunzător $\hat{\boldsymbol{A}}$ va fi colecţia centroizilor Euclidieni a seturilor $\boldsymbol{S}_i$; cu alte cuvinte, alfabetul de refacere final va fi optimal pentru partiţia finală, dar aceasta ar putea să nu fie optimală pentru alfabetul de refacere final, cu excepţia cazului când $n \to \infty$.

Pentru a obţine $\boldsymbol{S}$, mai întâi ne gândim la fiecare $\boldsymbol{S}_i$ ca la o colecţie de vectori în care punem vectori din secvenţa studiată până când toţi vectorii sunt distribuiţi. Iniţial vom începe prin a pune primii N vectori în seturi separate, altfel spus $\boldsymbol{x}_i \in \boldsymbol{S}_i$.

La fiecare iteraţie, un nou vector studiat $\boldsymbol{x}_m$ este observat. Găsim setul $\boldsymbol{S}_i$ pentru care distorsiunea între $\boldsymbol{x}_m$ şi centroidul $\hat{\boldsymbol{x}}(\boldsymbol{S}_i)$ este minimizată şi apoi includem $\boldsymbol{x}_m$ în acel set. Altfel la fiecare iteraţie noul vector este adăugat în setul cu cel mai apropiat centroid. La următoarea iteraţie, setul va avea un nou centroid. Operaţia continuă, până când toţi vectorii eşantion sunt încorporaţi.

Deşi similar ca filozofie, algoritmul *"**K-means**"* are câteva diferenţe esenţiale. În particular, este potrivit pentru cazul când *doar* secvenţa de studiu trebuie clasificată, deci acolo unde o secvenţă lungă de vectori trebuie grupată într-o manieră care să ofere o distorsiune scăzută.

Procedura secvenţială este eficientă din punct de vedere al calculului, dar un "cuantizor" nu este determinat decât atunci când procedura se opreşte.

Recent, Levenson şi alţii au folosit o variantă a algoritmilor *"K-means"* şi *IZODATA* cu o măsură a distorsiunii propusă de Itakura, pentru a determina modelele de recunoaştere a vorbirii indiferent de vorbitor.

13.4.6. Alegerea alfabetului de refacere inițial.

Există câteva metode de a alege alfabetul de refacere inițial $\hat{A}_0$ cerut de algoritm.

O metodă ce poate fi folosită pe distribuții de eşantioane este cea de la metoda "*K-means*", cu alte cuvinte alegând primii *N* vectori din secvența studiată.

Altă metodă este de a utiliza un cuantizor uniform pentru întregul alfabet al sursei sau pentru cea mai mare parte a sa, dacă acesta este limitat. De exemplu, dacă este utilizat pe o distribuție de eşantioane, se poate folosi un cuantizor uniform *k*-dimensional pe un cub euclidian *k*-dimensional, incluzând toate, sau cea mai mare parte a punctelor în secvența studiată.

A treia tehnică este utilă când se doreşte proiectarea de cuantizoare de clase succesive mai mari, până se atinge un nivel acceptabil al distorsiunii.

Construcția succesivă de cuantizoare.

(0) Inițializarea : facem $M = 1$ şi definim $\hat{A}_0(1) = \hat{x}(A)$ centroidul întregului alfabet, sau centroidul secvenței studiate, dacă se foloseşte o distribuție de eşantioane.

(1) Fiind dat un alfabet de refacere $\hat{A}_0(M)$ conținând *M* vectori:

$$\{y_i; i = 1,...,M\},$$

"descompunem" fiecare vector y_i în doi vectori alăturați:

$$y_i + \varepsilon \text{ şi } y_i - \varepsilon, \qquad i = 1,...,M.$$

unde ε este vectorul perturbației fixat. Colecția $\tilde{A}$ dată de:

$$\{y_i + \varepsilon, y_i - \varepsilon, i = 1,...,M\},$$

are $M + M = 2M$ vectori.

Înlocuim *M* cu *2M*.

(2) Dacă $M = N$ facem $\hat{A}_0 = \tilde{A}(M)$ şi ne oprim. $\hat{A}_0$ este deci alfabetul de refacere inițial pentru algoritmul de cuantizare de clasă *N*.

Dacă $M \neq N$ rulăm algoritmul pentru un cuantizor de clasa (nivel) *M* cu $\tilde{A}(M)$, pentru a determina un alfabet de refacere bun $\hat{A}_0(M)$ şi apoi ne întoarcem la pasul **(1)**.

Aici considerăm cuantizoare de clasa *M* cu $M = 2^R, R = 0,1,...$ şi continuăm până obținem un set de valori inițiale pentru un cuantizor de clasă *N*.

Folosind algoritmul de descompunere pe o secvență studiată se poate începe cu un cuantizor de clasa (nivel) *1*, constând în centroidul secvenței studiate. Apoi acest vector este descompus în doi vectori şi algoritmul pentru cuantizori de clasa (nivel) *2* este rulat folosind această pereche de vectori pentru a obține un cuantizor bun de punct fix de clasă sau nivel *2*. Fiecare din aceşti doi vectori este apoi descompus şi algoritmul este rulat pentru a obține un cuantizor bun de clasă *4*.

Se remarcă că putem avea cuantizoare de punct fix pentru valorile $1,2,3,8,...,N$.

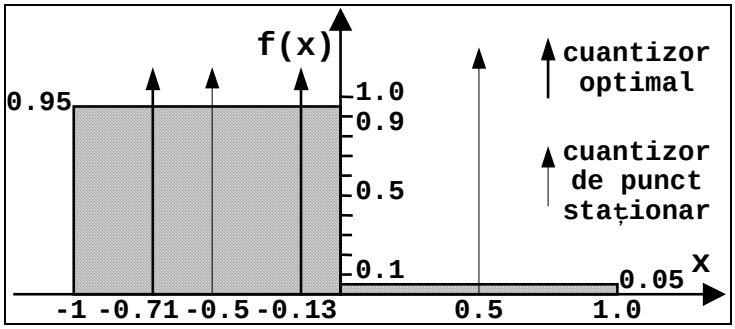


Fig. 1. Distribuția de test pentru cuantizarea vectorială.

În acest caz avem cuantizoare cu două clase şi cu două optime locale. Unul are clasele de ieşire *(0.383)*, celălalt, care este optim global, are clasele de ieşire de *(−0.71)* şi *(0.13)*, din care rezultă o eroare medie pătratică de *(0.048)*.

Algoritmul modificat este testat pe o secvență de *2000* de eşantioane alese în conformitate cu densitatea de probabilitate arătată. Zgomotul Gaussian a fost adăugat pornind de la o valoare unitară şi reducând valoarea cu aproximativ *50%* la fiecare rulare succesivă. Ipoteza inițială a fost *(+0.5)* şi *(−0.5)*, ceea ce înseamnă un optim local. Fiecare rulare a fost oprită când distorsiunea s-a modificat cu mai puțin de *0.1%* față de valoarea sa anterioară.

13.5. Detalii de implementare.

În continuare este prezentată o implementare simplă a algoritmului LBG de cuantizare vectorială. Este vorba de un program C++ care reduce numărul de culori dintr-o imagine de la 256 la 8, 16, 32 sau 64 culori. Noile culori se determină din condiția de minimizare a erorii medii pătratice de aproximare a vechilor culori. Atât imaginea inițială cât şi cea finală sunt imagini de tip ".BMP" (bitmap cu 256 culori indexate). O asemenea implementare poate fi folosită foarte simplu la generarea modelelor de goblen plecând de la orice imagine color.

Alegerea setului inițial de culori (alfabetul de refacere inițial) se face folosind metoda descrisă în paragraful 13.4.6. Inițial, se consideră că alfabetul de refacere inițial are doar o singură clasă (care coincide, evident, cu media aritmetică a setului de intrare) şi apoi se dublează numărul de clase prin divizarea fiecărei clase în două, după care se rulează algoritmul LBG. Operațiile precedente se repetă până la atingerea numărului dorit de clase (culori).

Implementarea prezentată în continuare furnizează la ieşire, pe lângă imaginea bitmap cu număr redus de culori şi un alt fişier de acelaşi tip care conține paleta de culori în format grafic si modul de etichetare a acestor culori. La acestea se adaugă un număr de fişiere text conținând "harta" goblenului.

```
// Această funcție realizează reducerea numărului de culori dintr-o
// imagine de la 256 la nColors folosind algoritmul LBG de cuantizare
// vectorială. Pentru claritate, nu au fost incluse optimizări evidente,
// cum ar fi utilizarea histogramei la calculul distorsiunii

BOOL CreateGoblenModel(CString* pstrFilePath, int nColors)
{
   // aşteaptă număr colori
   if (nColors == -1) {
      do {
         _tprintf(_T("Color Count (8,16,32 or 64): "));
         _tscanf(_T("%d"), &nColors);
      } while (nColors!=8 && nColors!=16 && nColors!=32 && nColors!=64);
   }

   // aşteaptă nume fişier
```

```
if (pstrFilePath == NULL) {
   pstrFilePath = new CString;
   TCHAR c;
   scanf(_T("%c"), &c);
   do {
      _tprintf(_T("Image File Path: "));
      _getts(pstrFilePath->GetBuffer(MAX_PATH));
      pstrFilePath->ReleaseBuffer();
   } while (pstrFilePath->GetLength() == 0);
}

// deschide fişier BITMAP (256 culori neapărat!!)
BOOL bRet = Open_BMPFile(pstrFilePath);
if (bRet == FALSE)
   return FALSE;

double dmin;
int p, q;
double* Red = (double*)malloc(nColors*sizeof(double));
double* Grn = (double*)malloc(nColors*sizeof(double));
double* Blu = (double*)malloc(nColors*sizeof(double));
Red[0]=0;
Grn[0]=0;
Blu[0]=0;
// calculează valoarea medie pentru cele 3 componente de culoare
// ele constituie coordonatele nodului iniţial
for (q=0; q<BMIH.biHeight; q++) {
   for (p=0, dmin=0.0; p<BMIH.biWidth; p++) {
      Red[0] += RGBQ[SMatrix[p][q]].rgbRed;
      Grn[0] += RGBQ[SMatrix[p][q]].rgbGreen;
      Blu[0] += RGBQ[SMatrix[p][q]].rgbBlue;
   }
}
Red[0] /= BMIH.biWidth * BMIH.biWidth;
Grn[0] /= BMIH.biWidth * BMIH.biWidth;
Blu[0] /= BMIH.biWidth * BMIH.biWidth;

int i, j, k, m, u, v;
double eps = 0.001, DC = 1000000.0, DP, d, R,G,B;
int NC = 1, s;

// Bucla principală a algoritmului LBG de cuantizare vectorială.
// Numărul initial de centroizi (culori) este 1, el se dublează la
// fiecare pas până la atingerea numărului dorit
v=0, m=0;
while((1<<v) < nColors) {
   // dublează numărul de centroizi
   for (u=0; u<NC; u++) {
      Red[u] += 0.005;
      Grn[u] += 0.005;
      Blu[u] += 0.005;
      Red[u+NC] = Red[u] - 0.01;
      Grn[u+NC] = Grn[u] - 0.01;
      Blu[u+NC] = Blu[u] - 0.01;
   }
   NC = NC * 2;

   // ajustează pozitie centroizi până distorsiunea relativă scade sub prag
   DC = 1000000.0;
   _tprintf(_T("\n\nCOLORS=%3d\n"), NC);
   do {
```

```
      DP = DC;
      DC = 0;
      // calcul distorsiune
      for (q=0; q<BMIH.biHeight; q++) {
         for (p=0, dmin=0.0; p<BMIH.biWidth; p++) {
            _tprintf(_T("\rx=%3d y=%3d"), q, p);
            dmin = 256.0 * 256 * 3;
            j = SMatrix[p][q];
            for (i=0; i<NC; i++) {
               d =(RGBQ[j].rgbRed-Red[i]) * (RGBQ[j].rgbRed-Red[i]);
               d+=(RGBQ[j].rgbGreen-Grn[i]) * (RGBQ[j].rgbGreen-Grn[i]);
               d+=(RGBQ[j].rgbBlue-Blu[i]) * (RGBQ[j].rgbBlue-Blu[i]);
               if (d < dmin) {
                  dmin = d;
                  k = i;
               }
            }
            DMatrix[p][q] = k;
            DC += dmin;
         }
      }

      k = 0;
      DC = DC / BMIH.biWidth / BMIH.biHeight;
      _tprintf(_T("\nEpoch%3d, Error=%f\n"), m, (DP-DC)/DC);
      // recalculează poziție centroizi
      if ((DP - DC) / DC > eps) {
         for (i=0; i<NC; i++) {
            _tprintf(_T("Recompute node=%3d\n"), i);
            s=0; R=0.0; G=0.0; B=0.0;
            for (q=0; q<BMIH.biHeight; q++) {
               for (p=0; p<BMIH.biWidth; p++) {
                  if (DMatrix[p][q] == i) {
                     j = SMatrix[p][q];
                     R += (float)RGBQ[j].rgbRed;
                     G += (float)RGBQ[j].rgbGreen;
                     B += (float)RGBQ[j].rgbBlue;
                     s++;
                  }
               }
            }
            if (s != 0) {
               Red[i] = R / s;
               Grn[i] = G / s;
               Blu[i] = B / s;
            }
         }
      } else
         k = 1;
      m++;
   } while (k == 0);
   v++;
}

// ordonează culorile din paletă funcție de luminanță
long Index[64];
for(i=0; i<nColors; i++)
   Index[i] = i;
for(i=0; i<nColors; i++) {
   for(j=0; j<nColors-1; j++) {
      if ((Red[j]+Grn[j]+Blu[j]) > (Red[j+1]+Grn[j+1]+Blu[j+1])) {
```

```
            R = Red[j];
            G = Grn[j];
            B = Blu[j];
            Red[j] = Red[j+1];
            Grn[j] = Grn[j+1];
            Blu[j] = Blu[j+1];
            Red[j+1] = R;
            Grn[j+1] = G;
            Blu[j+1] = B;
            k = Index[j];
            Index[j] = Index[j+1];
            Index[j+1] = k;
         }
      }
   }

   // reconstruieşte noua imagine
   for(q=0; q<BMIH.biHeight; q++) {
      for(p=0; p<BMIH.biWidth; p++) {
         for(i=0; i<nColors; i++) {
            if (Index[i] == DMatrix[p][q])
               break;
         }
         DMatrix[p][q] = i;
      }
   }

   CString strModulePath = GetOutputPath();
   _tprintf(_T("\nOutput to folder: %s\n"), strModulePath);

   // imaginea generată va fi salvată în fişierul GOBLEN.BMP
   CString strNewFile = strModulePath + _T("GOBLEN.bmp");
   bRet = fBMP.Open(strNewFile, CFile::modeCreate | CFile::modeWrite);
   if (bRet == FALSE) {
      _tprintf(_T("Cannot Open Destination File!!!"));
      return FALSE;
   }
   // construieşte noua paletă
   for(i=0; i<nColors; i++) {
      RGBQ[i].rgbRed = Red[i] + 0.5;
      RGBQ[i].rgbGreen= Grn[i] + 0.5;
      RGBQ[i].rgbBlue   = Blu[i] + 0.5;
   }
   free(Red);
   free(Grn);
   free(Blu);
   for(i=nColors; i<256; i++) {
      RGBQ[i].rgbRed = 0;
      RGBQ[i].rgbGreen= 0;
      RGBQ[i].rgbBlue   = 0;
   }
   // scrie antetul şi apoi conţinutul fişierului rezultat
   fBMP.Write(&BMFH, sizeof(BITMAPFILEHEADER));
   fBMP.Write(&BMIH, sizeof(BITMAPINFOHEADER));
   for (i=0; i<256; i++)
      fBMP.Write(&RGBQ[i], sizeof(RGBQUAD));
   for(q=BMIH.biHeight-1; q>=0; q--) {
      for(p=0; p<L; p++) {
         i = DMatrix[p][q];
         fBMP.Write(&i, 1);
      }
```

```
}
fBMP.Close();

// ataşează fiecărei colori din cele max 64 posibile un caracter
BYTE  c;
char  Cod[64] = {'0','1','2','3','4','5','6','7',
                 '8','9','A','B','C','D','E','F',
                 'G','H','J','K','L','M','N','P',
                 'R','S','T','U','V','X','Y','Z',
                 'a','b','c','d','e','f','g','h',
                 'j','k','m','n','o','p','q','r',
                 's','t','u','v','x','y','z','w',
                 '@','#','$','%','&','*','<','>'};
u = (BMIH.biWidth + PW - 1) / PW;
v = (BMIH.biHeight + PH - 1) / PH;
for(i=0; i<u; i++) {
   for(j=0; j<v; j++) {
      strNewFile.Format(_T("%sGOBL%02d%02d.TXT"), strModulePath, i, j);
      // deschide fişier text
      bRet = fTXT.Open(strNewFile, CFile::modeCreate | CFile::modeWrite);
      if (bRet == FALSE) {
         _tprintf(_T("Cannot Open Output Text File!!!"));
         return FALSE;
      }
      // scrier fişier text
      for(q=j*PH; q<j*PH+PH && q<BMIH.biHeight; q++) {
         for(p=i*PW; p<i*PW+PW && p<BMIH.biWidth; p++) {
            c = Cod[DMatrix[p][q]];
            fTXT.Write(&c, 1);
            if (p%10 == 9) {
               c=0x20;
               fTXT.Write(&c, 1);
            }
         }
         c = 0x0d;
         fTXT.Write(&c, 1);
         c = 0x0a;
         fTXT.Write(&c, 1);
         if (q%10 == 9) {
            c = 0x0d;
            fTXT.Write(&c, 1);
            c = 0x0a;
            fTXT.Write(&c, 1);
         }
         if (q%PH == PH-1) {
            c = 0x0c;
            fTXT.Write(&c, 1);
         }
      }
      fTXT.Close();
   }
}
Free_BMatrix(SMatrix, BMIH.biWidth, BMIH.biHeight);
Free_BMatrix(DMatrix, BMIH.biWidth, BMIH.biHeight);

// crează fişier paletă
strNewFile = strModulePath + _T("PALLETE.BMP");
bRet = fPAL.Open(strNewFile, CFile::modeCreate | CFile::modeWrite);
if (bRet == FALSE) {
   _tprintf(_T("Cannot Open Palette File!!!"));
   return FALSE;
```

```
   }
   BMIH.biWidth = nColors * 16;
   BMIH.biHeight = 200;
   BMIH.biSizeImage = BMIH.biWidth * BMIH.biHeight;
   BMFH.bfSize = BMIH.biSizeImage + BMFH.bfOffBits;

   fPAL.Write(&BMFH, sizeof(BITMAPFILEHEADER));
   fPAL.Write(&BMIH, sizeof(BITMAPINFOHEADER));
   for (i=0; i<256; i++)
      fPAL.Write(&RGBQ[i], sizeof(RGBQUAD));
   for(q=0; q<BMIH.biHeight-24; q++) {
      for(p=0; p<BMIH.biWidth; p++) {
         c = p / 16;
         fPAL.Write(&c, 1);
      }
   }

   HRSRC hResource = ::FindResource(NULL,
                                    (LPCTSTR)IDR_BINARY_HEADER,
                                    _T("BINARY"));
   HGLOBAL hGlobal = ::LoadResource(NULL, hResource);
   BYTE* pHeader = (BYTE*)::LockResource(hGlobal);

   for(q=0; q<24; q++) {
      for(p=0; p<BMIH.biWidth; p++) {
         if (pHeader[q*1024+p] == 0)
            c = 0;
         else
            c = nColors-1;
         fPAL.Write(&c, 1);
      }
   }
   fPAL.Close();
   return TRUE;
}
```

14. Rețele neurale folosite în recunoaşterea de forme.

Rețelele neurale (neuronale) artificiale (Haykin[47]), (Kohonen[62][63]), (Lipmann[66]), (Rumelhart[98]) încearcă să se apropie de modelul creierului uman. Spre deosebire de maşinile Von--Neuman care se caracterizează prin existența unei unități procesoare care execută instrucțiuni stocate în memorie, într-o secvență aflată sub controlul numărătorului de program, alte arhitecturi propuse încearcă să exploateze cât mai eficient paralelismul care este de obicei inerent. "Procesoarele" care formează rețelele neuronale, sunt denumite neuroni artificiali.

Dacă majoritatea calculatoarelor existente în momentul de față dispun de o singură unitate procesoare, extrem de puternică şi de rapidă, la cealaltă extremă din punctul de vedere al structurii interne se plasează aceste rețele neurale artificiale, caracterizate printr-o simplificare extremă a unităților componente, alături de o extindere cât mai largă a conexiunilor între aceste unități procesoare.

Orice rețea neurală este caracterizată de trei elemente: modelul neuronului, arhitectura rețelei şi algoritmul de antrenare folosit.

14.1. Modele de neuroni.

În ceea ce priveşte modelele de neuroni, cel mai mult folosite în momentul de față sunt cele fără memorie, deci care implementează o relație de forma:

$$y_j = \boldsymbol{f}\left(\sum_{i=1}^{N} w_{ij} x_i - \theta_j\right) \qquad (1)$$

unde y_j este ieşirea neuronului "j", x_i este intrarea "i" a neuronului, w_{ij} este ponderea conexiunii de la intrarea "i" la neuronul "j", iar θ_j este pragul ataşat neuronului "j". Deci în

esenţă, sunt aplicate un set de intrări $(x_1,x_2,...,x_n)$, fiecare reprezentând, de obicei, ieşirile altor neuroni. Fiecare intrare este multiplicată cu o pondere corespunzătoare $(w_{i1},w_{i2},...,w_{in})$, analog puterii sinapsei, şi apoi toate rezultatele sunt sumate pentru a determina nivelul de activare al neuronului.

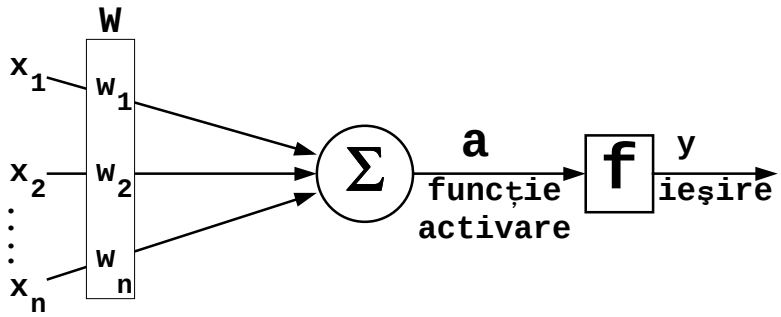
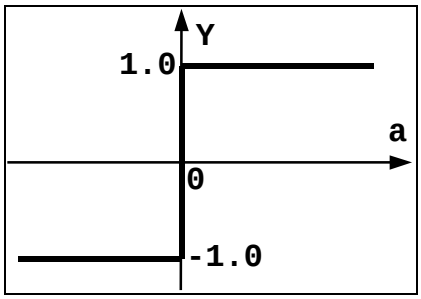


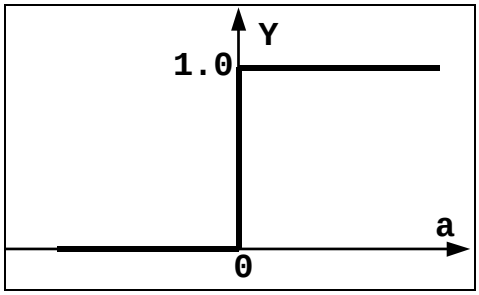
Fig. 1. Structura neuronului artificial.

Funcţia $Y = f(a)$ este o funcţie neliniară care poate fi de tipul limitare hardware, prag logic, sigmoidă simetrică sau nu, sau chiar funcţia identică.

Limitarea hardware se implementează în unul din următoarele două moduri:

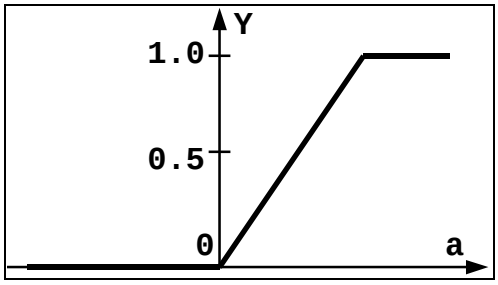


$$Y(a) = \begin{cases} -1, \text{pentru } a < 0 \\ +1, \text{pentru } a \geq 0 \end{cases}$$

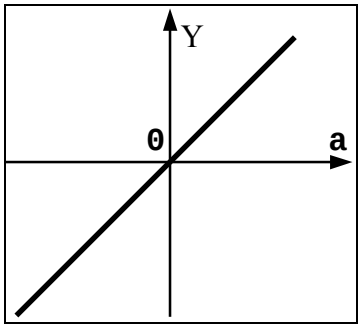


$$Y(a) = \begin{cases} 0, \text{ pentru } a < 0 \\ +1, \text{pentru } a \geq 0 \end{cases} \qquad (2)$$

Pragul logic realizează limitarea domeniului de ieşire la zero pentru valori de intrare negative (sau uneori la *-1*) şi la *+1* pentru valori ce depăşesc un anumit prag pozitiv. Pentru anumite aplicaţii se preferă utilizarea neuronului liniar.

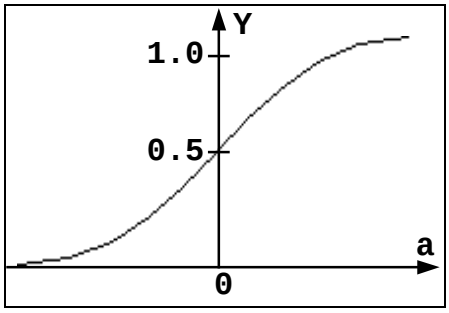


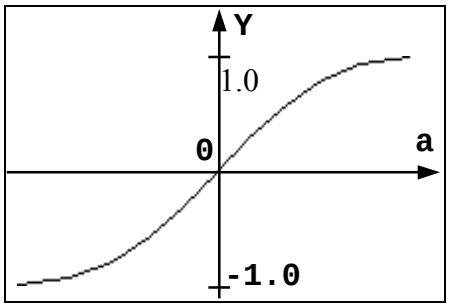
$$Y(a) = \begin{cases} 0, \text{pentru } a < 0 \\ a, \text{pentru } a \geq 0 \end{cases}$$



$$Y(a) = a \qquad (3)$$

Funcţia sigmoidă, foarte des folosită, asigură limitarea domeniului de ieşire al neuronului la gama $(0,1)$ pentru sigmoida asimetrică şi la $(-1,+1)$ pentru sigmoida simetrică.





$$\boldsymbol{Y}(a)=\frac{1}{1+\boldsymbol{exp}(-a)} \qquad \boldsymbol{Y}(a)=\frac{1-\boldsymbol{exp}(-a)}{1+\boldsymbol{exp}(-a)} \tag{4}$$

Prin această funcţie se obţine un câştig neliniar pentru neuronul artificial. Acest câştig variază de la valori mici pentru excitaţii negative mari (curba este aproape orizontală), la valori mari pentru excitaţii mari şi pozitive. Grossberg (1973) a găsit că acest câştig neliniar caracteristic rezolvă dilema saturării.

În ciuda diversităţii de modele de neuroni, majoritatea sunt bazate pe această configuraţie.

Modele mai sofisticate de neuroni introduc variabila timp, un astfel de model fiind descris de următoarele relaţii:

$$\begin{cases}\dfrac{du_j}{dt}=-u_j+\displaystyle\sum_{i=1}^{N}w_{ij}x_i+\theta_j\\ y_j=\boldsymbol{f}(u_j)\end{cases} \tag{5}$$

unde u_j caracterizează starea curentă a neuronului.

14.2. Arhitecturi de reţele neurale.

Deşi un singur neuron poate realiza funcţii simple de detecţie de modele, puterea calculului neural provine din conectarea neuronilor în cadrul reţelelor. O reţea neurală simplă este aranjarea unui grup de neuroni pe un strat ca în figura următoare:

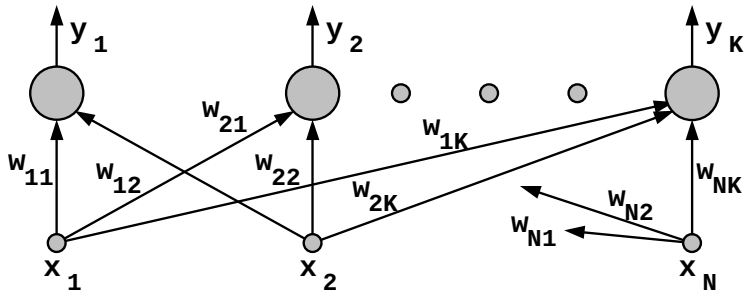


Fig. 2. Reţea neurală cu un strat.

Setul de intrare ***X*** are fiecare element conectat cu fiecare neuron, cu cîte o pondere separată. Fiecare neuron scoate la ieşire o sumă a produsului ponderi-intrări corespunzătoare. Actualele reţele artificiale şi biologice au unele din conexiuni lipsă dar conectivitatea totală a fost arătată aici din raţiuni de generalitate.

Este convenabil să reprezentăm ponderile ca fiind elementele unei matrici. Dimensiunile matricii fiind *N* linii pe *K* coloane, unde *N* este numărul de intrări, iar *K* numărul de neuroni. În acest fel putem calcula pentru setul de neuroni cele *K* funcţii de activare ca simple înmulţiri de matrici:

$$\boldsymbol{A}=\boldsymbol{W}^T\boldsymbol{X}\Rightarrow\boldsymbol{Y}=\boldsymbol{f}(\boldsymbol{A})=\boldsymbol{f}(\boldsymbol{W}^T\boldsymbol{X}) \tag{6}$$

unde ***X***, ***A***, ***Y*** sunt vectori linie, ***W*** este o matrice *NxK*.

Rețelele cu mai multe straturi, mult mai generale şi mai complexe, oferă în general, o capacitate mare de calcul. Asemenea rețele au fost construite în orice configurație imaginabilă, aranjând neuronii în straturi imitând structura straturilor dintr-o anumită zonă a creierului. Aceste rețele multistrat au capacități sporite, dovedite relativ la rețelele cu un singur strat, iar în ultimii ani algoritmi performanți s-au dezvoltat pentru antrenarea lor.

Rețelele multistrat pot fi formate prin simpla cascadare de straturi, ieşirile unui strat devenind intrări pentru stratul următor.

Rețelele multistrat nu furnizează nici o creştere a puterii de calcul în raport cu rețelele cu un singur strat dacă nu este prezentă o funcție de activare neliniară între straturi. Calculul ieşirii unui strat liniar constă în înmulțirea vectorului de intrare cu prima matrice de ponderi, apoi (dacă nu este prezentă o funcție de activare neliniară) multiplicând vectorul rezultat cu a doua matrice de ponderi. Aceasta ar putea fi exprimată prin :

$$\boldsymbol{Y} = \boldsymbol{W}_2^T(\boldsymbol{W}_1^T \boldsymbol{X}) \tag{7}$$

Având în vedere că înmulțirea de matrici este asociativă, relația poate fi rescrisă :

$$\boldsymbol{Y} = (\boldsymbol{W}_2^T \boldsymbol{W}_1^T)\boldsymbol{X} = (\boldsymbol{W}_1 \boldsymbol{W}_2)^T \boldsymbol{X} \tag{8}$$

Aceasta arată că o rețea liniară cu două straturi este exact echivalentă cu o rețea cu un singur strat ce are matricea ponderilor egală cu produsul celor două matrici. De aici, se poate trage concluzia că funcția de activare neliniară este vitală pentru extinderea capacităților rețelelor față de cele cu un singur strat.

Rețelele considerate până acum nu au conexiuni cu răspunsuri spre înapoi ("feedback"), conexiuni care să aplice valorile de la ieşirea unui strat spre intrarea aceluiaşi strat sau a altui strat. Această clasă specială, numită **rețele nerecurente** sau cu răspuns (propagare a fluxului de date) spre înainte ("feed foreward") prezintă un interes considerabil şi sunt larg aplicate.

În general, rețelele care conțin conexiuni feedback sunt denumite ***rețele recurente***. Rețelele nerecurente nu au memorie, ieşirile fiind determinate doar pe baza intrărilor curente şi a valorilor ponderilor.

În cîteva configurații, rețelele recurente recirculează ieşirile anterioare înapoi spre intrare; de aici ieşirile lor sunt determinate pe baza intrărilor curente şi ieşirilor anterioare. Din acest motiv rețelele recurente pot etala proprietăți similare cu memoria pe termen scurt a omului.

14.3. Algoritmi de antrenare a rețelelor neurale.

Dintre toate caracteristicile interesante ale rețelelor neurale, nici una nu captează atâta imaginație ca abilitatea lor de a învăța.

Antrenarea rețelelor artificiale este însă limitată, multe probleme dificile rămânând a fi rezolvate. Oricum, impresionante demonstrații au fost deja făcute, cum ar fi **NetTalk** a lui Sejnowski.

Algoritmii de antrenare sunt categorizați în ***supervizați*** şi ***nesupervizați***. O rețea este antrenată dacă aplicarea unui set de intrări produce ieşirea dorită sau una apropiată . Antrenarea este realizată prin aplicarea secvențială de vectori de intrare, în timpul fiecărei asemenea operații ajustând ponderile din rețea în acord cu o procedură predeterminată. În timpul antrenării, ponderile din rețea converg gradual spre valori pentru care fiecare vector de intrare produce vectorul de ieşire dorit.

Antrenarea supervizată cere pentru fiecare vector de antrenare perechea vector țintă care este ieşirea dorită. Acestea sunt denumite pereche de antrenare. Uzual, o rețea este antrenată după un număr de astfel de perechi de antrenare.

Un vector de intrare este aplicat, ieşirea rețelei este calculată şi comparată cu vectorul țintă, iar diferența (eroarea) este trimisă înapoi, iar ponderile sunt schimbate în acord cu un anumit algoritm care tinde să minimizeze eroarea la un nivel acceptabil.

Învățarea nesupervizată nu cere cunoştințe apriorice despre datele aplicate la intrarea rețelei, rețelele se auto-organizează pentru a produce ieşirile dorite. Antrenarea supervizată a fost criticată ca fiind biologic neplauzibilă. Antrenarea nesupervizată este un model mai plauzibil de învățare în sistemele biologice.

Dezvoltată de Kohonen (1984) şi încă de mulți alții, antrenarea nesupervizată nu cere vector țintă pentru ieşiri, de aceea nu au loc comparații cu răspunsul ideal. Setul de antrenare nu consă decât în vectori de antrenare. Algoritmul de antrenare modifică ponderile rețelei pentru a produce un vector de ieşire consistent. Procesul de antrenare este deci extragerea de proprietăți statistice ale setului de antrenare şi a vectorilor, dintr-un grup similar de vectori, în clase. Aplicând la intrare un vector, dintr-o clasă dată, va produce la ieşire un vector specific, dar aici nu este nici o cale de a determina care model specific va fi găsit la ieşire de un vector de intrare dintr-o clasă dată.

De aici, ieşirea unei astfel de rețele trebuie în general transformată într-o formă ce să poată fi înțeleasă, ulterior procesului de antrenare.

Marea majoritate a algoritmilor utilizați în prezent au evoluat de la concepțiile lui Hebb (1961). El propunea un model pentru învățare nesupervizată unde "tăriile" sinapselor (ponderile) sunt mărite dacă atît neuronul sursă cît şi neuronul destinație sunt active. În acest fel modelele utilizate în rețele sunt puternice, iar fenomenul de folosire şi învățare prin repetiție este explicat.

O rețea neurală artificială utilizând algoritmul de învățare al lui Hebb va modifica ponderile în acord cu produsul nivelelor de excitație dintre neuronii sursă şi destinație. Legea de învățare este descrisă de ecuația:

$$w_{i,j}(n+1) = w_{i,j}(n) + \alpha Y_i Y_j \tag{9}$$

unde: $w_{i,j}(n)$ - valoarea ponderii de la neuronul "i" la neuronul "j" înaintea ajustării;

$w_{i,j}(n+1)$ - valoarea ponderii de la neuronul "i" la neuronul "j" după ajustare;

α – coeficientul de învățare;

Y_i - ieşirea neuronului "i" şi intrarea neuronului "j";

Y_j - ieşirea neuronului "j".

Au fost construite rețele pentru a folosi algoritmul de învățare al lui Hebb. Oricum, algoritmi de antrenare mai buni au fost descoperiți de Rosenblatt (1962), Widrow (1959), Widrow, Hoff (1960), realizând rețele capabile să învețe o arie largă de modele de intrare, la o rată de învățare ridicată, algoritmi care au perfecționat algoritmul de învățare al lui Hebb.

14.4. Perceptronul cu un singur strat.

Căutând să descopere modele hardware/software ale neuronului biologic şi a rețelei sale de interconectare McCulloch şi Pitts (1943) au publicat primul studiu sistematic al rețelelor neurale artificiale. Într-o lucrare ulterioară Pitts şi McCulloch (1947) au explorat configurații de rețele pentru recunoaştere de modele cu invarianță la rotație şi translație.

S-a folosit cel mai mult modelul neuronului descris anterior, însă folosind nelinearitatea prag logic:

$$f(a) = \begin{cases} 1, \text{pentru } a > 0 \\ 0, \text{pentru } a \le 0 \end{cases} \tag{10}$$

Aceste sisteme poartă numele de perceptron. În general, acestea constau într-un singur strat de neuroni artificiali conectați prin ponderi la un set de intrări.

Rosenblatt (1962) a demonstrat teorema învățării perceptronilor, Widrow (Widrow 1961,1963, Widrow şi Angell 1962, Widrow şi Hoff 1960) au făcut un număr de demonstrații convingătoare asupra perceptronilor ca sisteme. Minsky (Minsky şi Papert 1969) au analizat

aceste probleme cu o mare rigurozitate, dovedind că există severe restricţii în ceea ce poate reprezenta perceptronul cu un singur strat.

În ciuda limitărilor pe care le prezintă, studiul perceptronilor a fost extins (chiar dacă nu sunt folosiți pe scară largă). Teoria lor reprezintă o fundație pentru multe alte forme de rețele neurale artificiale şi au demonstrat că reprezintă principii importante în acest domeniu. Pentru aceste motive ei reprezintă punctul logic de început în studiul teoriei rețelelor neurale artificiale.

Valabilitatea teoremei învățării perceptronilor a lui Rosenblatt (1962) demonstrează că perceptronii pot învăța orice poate fi reprezentat. Este important de făcut distincția între reprezentare şi învățare. Reprezentarea se referă la abilitatea perceptronului (sau a altor rețele) de a simula o funcție specifică. Învățarea reclamă existența unei proceduri sistematice de ajustare a ponderilor din rețea pentru a produce acea funcție.

14.5. Perceptronul multistrat.

În figura următoare este arătat modelul neuronului folosit ca bloc de construcție fundamental pentru rețelele perceptron multistrat:

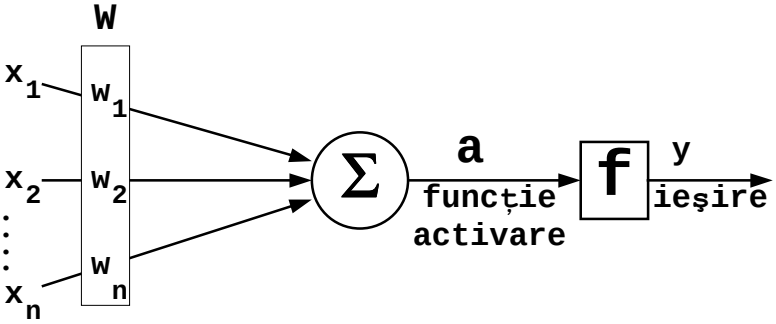


Fig. 3. Structura unui neuron.

unde -"a" reprezintă suma intrărilor ponderate pentru neuronul respectiv;

- $f(a)$ funcția neliniară de ieşire sigmoidă simetrică sau nu, sau chiar funcție liniară.

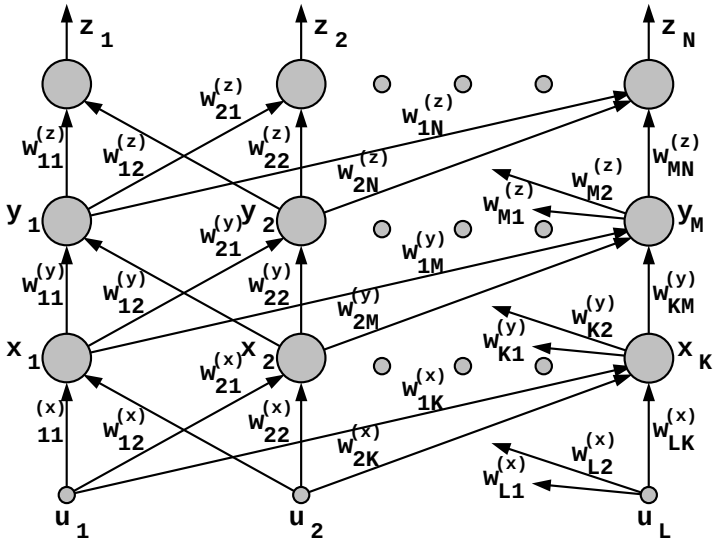


Fig. 4. Arhitectura tipică a perceptronului cu trei straturi.

O arhitectură tipică pentru un perceptron cu trei straturi este prezentată în figura 4.

Capabilitatea perceptronului multistrat rezidă în neliniaritatea folosită în nodurile sale. Dacă nodurile sunt elemente liniare, atunci un perceptron cu un singur strat având ponderi apropiate poate realiza foarte bine aceleaşi calcule ca şi rețeaua multistrat.

Algoritmul backpropagation este o metodă sistematică de antrenare a rețelelor neurale artificiale. El are un fundament matematic puternic, dar care nu este întotdeauna practic.

Pentru a antrena rețeaua, un vector ***X*** este aplicat la intrare, iar ieşirea ***Y*** este calculată. Ponderile care conectează intrările sunt modificate la o valoare care să minimizeze eroarea dintre ieşirea obținută şi ieşirea dorită. După un anumit număr de paşi, rețeaua va învăța să diferențieze vectorii cu care a fost antrenată de alți vectori de intare.

Fiecare neuron are intrările conectate cu ieşirile stratului precedent. Relațiile care descriu funcționarea perceptronului multistrat sunt:

$$\begin{cases} z_{pj} = \boldsymbol{f}\left(\sum_{k=1}^{N} w_{kj}^{(z)} y_{pk}\right) \\ y_{pj} = \boldsymbol{f}\left(\sum_{k=1}^{M} w_{kj}^{(y)} x_{pk}\right) \\ x_{pj} = \boldsymbol{f}\left(\sum_{k=1}^{L} w_{kj}^{(x)} u_{pk}\right) \end{cases} \tag{11}$$

unde: N - numărul de neuroni de pe stratul de ieşire;

M - numărul de neuroni de pe stratul intermediar (ascuns);

K - numărul de neuroni de pe stratul de intrare (ascuns);

L - numărul de intrări în rețea

z_{pj} - ieşirea neuronului "j" de pe stratul de ieşire pentru forma "p" de intrare;

y_{pj} - ieşirea neuronului "j" de pe stratul intermediar pentru forma "p" de intrare;

x_{pj} - ieşirea neuronului "j" de pe stratul de intrare pentru forma "p" de intrare;

u_{pj} - valoarea aplicată intrării "j" în rețea pentru forma "p" de intrare;

$w_{ij}^{(z)}$ - ponderea conexiunii neuronului "i" de pe stratul intermediar cu neuronul "j" de pe stratul de ieşire;

$w_{ij}^{(y)}$ - ponderea conexiunii neuronului "i" de pe stratul de intrare cu neuronul "j" de pe stratul de intermediar;

$w_{ij}^{(z)}$ - ponderea conexiunii intrării "i" în rețea cu neuronul "j" de pe stratul de intrare.

Funcția de eroare corespunzătoare formei de intrare "p" este dată de:

$$E_p = \frac{1}{2}\sum_{j=1}^{M}(d_{pj} - z_{pj})^2 \tag{12}$$

Toate ponderile rețelei neurale se vor modifica în sensul minimizării acestei erori, folosind, de exemplu**,** metoda gradientului descrescător, adică:

$$\begin{cases} \Delta w_{ij}^{(z)} = -\eta \dfrac{\partial E_p}{\partial w_{ij}^{(z)}} \\ \Delta w_{ij}^{(y)} = -\eta \dfrac{\partial E_p}{\partial w_{ij}^{(y)}} \\ \Delta w_{ij}^{(x)} = -\eta \dfrac{\partial E_p}{\partial w_{ij}^{(x)}} \end{cases} \tag{13}$$

Pentru fiecare strat în parte, derivata parțială a erorii medii pătratice în raport cu ponderea curentă poate fi exprimată prin:

$$\begin{cases} \dfrac{\partial E_p}{\partial w_{ij}^{(z)}} = \dfrac{\partial E_p}{\partial a_{pj}^{(z)}} \dfrac{\partial a_{pj}^{(z)}}{\partial w_{ij}^{(z)}} \\ \dfrac{\partial E_p}{\partial w_{ij}^{(y)}} = \dfrac{\partial E_p}{\partial a_{pj}^{(y)}} \dfrac{\partial a_{pj}^{(y)}}{\partial w_{ij}^{(y)}} \\ \dfrac{\partial E_p}{\partial w_{ij}^{(x)}} = \dfrac{\partial E_p}{\partial a_{pj}^{(x)}} \dfrac{\partial a_{pj}^{(x)}}{\partial w_{ij}^{(x)}} \end{cases} \tag{14}$$

unde funcția de activare corespunzătoare nodului *"j"* din fiecare strat este dată de:

$$\begin{cases} a_{pj}^{(z)} = \sum_{i=1}^{M} w_{ij}^{(z)} y_{pi} \\ a_{pj}^{(y)} = \sum_{i=1}^{M} w_{ij}^{(y)} x_{pi} \\ a_{pj}^{(x)} = \sum_{i=1}^{M} w_{ij}^{(x)} u_{pi} \end{cases} \tag{15}$$

Se poate evalua derivata parțială a funcției de activare în raport cu ponderea curentă:

$$\begin{cases} \dfrac{\partial a_{pj}^{(z)}}{\partial w_{ij}^{(z)}} = \sum_{k=1}^{M} \dfrac{\partial w_{kj}^{(z)}}{\partial w_{ij}^{(z)}} y_{pi} = y_{pi} \\ \dfrac{\partial a_{pj}^{(y)}}{\partial w_{ij}^{(y)}} = \sum_{k=1}^{M} \dfrac{\partial w_{kj}^{(y)}}{\partial w_{ij}^{(y)}} x_{pi} = x_{pi} \\ \dfrac{\partial a_{pj}^{(x)}}{\partial w_{ij}^{(x)}} = \sum_{k=1}^{M} \dfrac{\partial w_{kj}^{(x)}}{\partial w_{ij}^{(x)}} u_{pi} = u_{pi} \end{cases} \quad \text{deoarece, evident } \frac{\partial w_{kj}^{(.)}}{\partial w_{ij}^{(.)}} = 0, \text{ pentru } k \neq i \tag{16}$$

Rezultă:

$$\begin{cases} \dfrac{\partial E_p}{\partial w_{ij}^{(z)}} = \dfrac{\partial E_p}{\partial a_{pj}^{(z)}} y_{pi} \\ \dfrac{\partial E_p}{\partial w_{ij}^{(y)}} = \dfrac{\partial E_p}{\partial a_{pj}^{(y)}} x_{pi} \\ \dfrac{\partial E_p}{\partial w_{ij}^{(x)}} = \dfrac{\partial E_p}{\partial a_{pj}^{(x)}} u_{pi} \end{cases} \tag{17}$$

Se notează funcția de eroare pentru nodul "*j*" din fiecare strat şi forma "*p*", cu:

$$\begin{cases} \delta_{pj}^{(z)} \overset{\Delta}{=} -\dfrac{\partial E_p}{\partial a_{pj}^{(z)}} \\ \delta_{pj}^{(y)} \overset{\Delta}{=} -\dfrac{\partial E_p}{\partial a_{pj}^{(y)}} \\ \delta_{pj}^{(x)} \overset{\Delta}{=} -\dfrac{\partial E_p}{\partial a_{pj}^{(x)}} \end{cases} \tag{18}$$

şi atunci rezultă:

$$\begin{cases} \Delta w_{ij}^{(z)} = \eta \delta_{pj}^{(z)} y_{pi} \\ \Delta w_{ij}^{(y)} = \eta \delta_{pj}^{(y)} x_{pi} \\ \Delta w_{ij}^{(x)} = \eta \delta_{pj}^{(x)} u_{pi} \end{cases} \tag{19}$$

de unde se obține regula de ajustare a ponderilor:

$$\begin{cases} w_{ij}^{(z)}(t+1) = w_{ij}^{(z)}(t) + \eta\delta_{pj}^{(z)}(t) y_{pi}(t) \\ w_{ij}^{(y)}(t+1) = w_{ij}^{(y)}(t) + \eta\delta_{pj}^{(y)}(t) x_{pi}(t) \\ w_{ij}^{(x)}(t+1) = w_{ij}^{(x)}(t) + \eta\delta_{pj}^{(x)}(t) u_{pi}(t) \end{cases} \tag{20}$$

Funcția de eroare poate fi evaluată folosind:

$$\begin{cases} \delta_{pj}^{(z)} = -\dfrac{\partial E_p}{\partial a_{pj}^{(z)}} = -\dfrac{\partial E_p}{\partial z_{pj}} \dfrac{\partial z_{pj}}{\partial a_{pj}^{(z)}} \\ \delta_{pj}^{(y)} = -\dfrac{\partial E_p}{\partial a_{pj}^{(y)}} = -\dfrac{\partial E_p}{\partial y_{pj}} \dfrac{\partial y_{pj}}{\partial a_{pj}^{(y)}} \\ \delta_{pj}^{(x)} = -\dfrac{\partial E_p}{\partial a_{pj}^{(x)}} = -\dfrac{\partial E_p}{\partial x_{pj}} \dfrac{\partial x_{pj}}{\partial a_{pj}^{(x)}} \end{cases} \tag{21}$$

unde:

$$\begin{cases} \dfrac{\partial z_{pj}}{\partial a_{pj}^{(z)}} = \boldsymbol{f}'(a_{pj}^{(z)}) \\ \dfrac{\partial y_{pj}}{\partial a_{pj}^{(y)}} = \boldsymbol{f}'(a_{pj}^{(y)}) \\ \dfrac{\partial x_{pj}}{\partial a_{pj}^{(x)}} = \boldsymbol{f}'(a_{pj}^{(x)}) \end{cases} \quad \text{şi} \quad \begin{cases} \dfrac{\partial E_p}{\partial z_{pj}} = \dfrac{\partial}{\partial z_{pj}} \left[\dfrac{1}{2} \sum_{k=1}^{N} (d_{pk} - z_{pk})^2 \right] = -(d_{pj} - z_{pj}) \\ \dfrac{\partial E_p}{\partial y_{pj}} = \sum_{k=1}^{N} \dfrac{\partial E_p}{\partial a_{pk}^{(z)}} \dfrac{\partial a_{pk}^{(z)}}{\partial y_{pj}} = -\sum_{k=1}^{N} \delta_{pk}^{(z)} w_{jk}^{(z)} \\ \dfrac{\partial E_p}{\partial x_{pj}} = \sum_{k=1}^{M} \dfrac{\partial E_p}{\partial a_{pk}^{(y)}} \dfrac{\partial a_{pk}^{(y)}}{\partial x_{pj}} = -\sum_{k=1}^{N} \delta_{pk}^{(y)} w_{jk}^{(y)} \end{cases} \tag{22}$$

Rezultă:

$$\begin{cases} \delta_{pj}^{(z)} = (d_{pj} - z_{pj}) \boldsymbol{f}'(a_{pj}^{(z)}) \\ \delta_{pj}^{(y)} = \boldsymbol{f}'(a_{pj}^{(y)}) \sum_{k=1}^{N} \delta_{pk}^{(z)} w_{jk}^{(z)} \\ \delta_{pj}^{(x)} = \boldsymbol{f}'(a_{pj}^{(x)}) \sum_{k=1}^{M} \delta_{pk}^{(y)} w_{jk}^{(y)} \end{cases} \tag{23}$$

Derivata funcției sigmoidale *asimetrice* în raport cu funcția de activare este dată de:

$$\begin{cases} \boldsymbol{f}'(a_{pj}^{(z)}) = \dfrac{\partial}{\partial a_{pj}^{(z)}} \left[\dfrac{1}{1 + \boldsymbol{exp}(-a_{pj}^{(z)})} \right] = \dfrac{\boldsymbol{exp}(-a_{pj}^{(z)})}{\left[1 + \boldsymbol{exp}(-a_{pj}^{(z)})\right]^2} = z_{pj}(1 - z_{pj}) \\ \boldsymbol{f}'(a_{pj}^{(y)}) = \dfrac{\partial}{\partial a_{pj}^{(y)}} \left[\dfrac{1}{1 + \boldsymbol{exp}(-a_{pj}^{(y)})} \right] = \dfrac{\boldsymbol{exp}(-a_{pj}^{(y)})}{\left[1 + \boldsymbol{exp}(-a_{pj}^{(y)})\right]^2} = y_{pj}(1 - y_{pj}) \\ \boldsymbol{f}'(a_{pj}^{(x)}) = \dfrac{\partial}{\partial a_{pj}^{(x)}} \left[\dfrac{1}{1 + \boldsymbol{exp}(-a_{pj}^{(x)})} \right] = \dfrac{\boldsymbol{exp}(-a_{pj}^{(x)})}{\left[1 + \boldsymbol{exp}(-a_{pj}^{(x)})\right]^2} = x_{pj}(1 - x_{pj}) \end{cases} \tag{24}$$

În cazul în care se foloseşte sigmoida *simetrică* drept nelinearitate de ieşire, avem:

$$\begin{cases} f'(a_{pj}^{(z)}) = \dfrac{\partial}{\partial a_{pj}^{(z)}}\left[\dfrac{\exp(-a_{pj}^{(z)})}{1+\exp(-a_{pj}^{(z)})}\right] = \dfrac{2\exp(-a_{pj}^{(z)})}{\left[1+\exp(-a_{pj}^{(z)})\right]^2} = (1-z_{pj})^2 \\ f'(a_{pj}^{(y)}) = \dfrac{\partial}{\partial a_{pj}^{(y)}}\left[\dfrac{\exp(-a_{pj}^{(z)})}{1+\exp(-a_{pj}^{(y)})}\right] = \dfrac{2\exp(-a_{pj}^{(y)})}{\left[1+\exp(-a_{pj}^{(y)})\right]^2} = (1-y_{pj})^2 \\ f'(a_{pj}^{(x)}) = \dfrac{\partial}{\partial a_{pj}^{(x)}}\left[\dfrac{\exp(-a_{pj}^{(z)})}{1+\exp(-a_{pj}^{(x)})}\right] = \dfrac{2\exp(-a_{pj}^{(x)})}{\left[1+\exp(-a_{pj}^{(x)})\right]^2} = (1-x_{pj})^2 \end{cases} \quad (25)$$

În anumite aplicaţii, pentru stratul de ieşire se pot folosi neuroni lineari, caz în care aceste derivate sunt date de:

$$\begin{cases} f'(a_{pj}^{(z)}) = 1 \\ f'(a_{pj}^{(y)}) = y_{pj}(1-y_{pj}) \\ f'(a_{pj}^{(y)}) = x_{pj}(1-x_{pj}) \end{cases} \quad \text{sau} \quad \begin{cases} f'(a_{pj}^{(z)}) = 1 \\ f'(a_{pj}^{(y)}) = (1-y_{pj})^2 \\ f'(a_{pj}^{(y)}) = (1-x_{pj})^2 \end{cases} \quad (26)$$

Obiectivul antrenării reţelei neurale este de a ajusta ponderile astfel încît aplicarea unui set de intrări să producă ieşirea dorită. Antrenarea asigneaza fiecărui vector de intrare un vector pereche denumit vector ţintă care reprezintă ieşirea dorită.

Algoritmul "backpropagation" include deci următorii paşi:

Algoritmul "backpropagation"

Pasul 1: Iniţializarea ponderilor şi a offseturilor.
Se setează toate ponderile şi offseturile nodurilor la valori aleatoare mici.
Pasul 2: Aplică intrarea şi ieşirea dorită.
Se prezintă un vector continuu cu valorile de intrare $x_0, x_1, ..., x_{N-1}$ şi se specifică ieşirile dorite $d_0, d_1, ..., d_{M-1}$. Dacă reţeaua este folosită drept clasificator, atunci, de obicei, toate ieşirile dorite sunt setate la zero, exceptând cele care corespund clasei valorilor de intrare, care sunt setate la "*1*". Intrarea poate fi alta la fiecare încercare sau eşantioanele dintr-un set de antrenare pot fi prezente ciclic până cînd ponderile se stabilizează.
Pasul 3: Calcularea ieşirilor actuale.
Folosind neliniaritatea sigmoidală anterioară, se calculează ieşirile $y_0, y_1, ..., y_{M-1}$.
Pasul 4: Modificarea ponderilor
Pasul 5: Salt la **Pasul 2**.

Pentru **Pasul 4** se foloseşte algoritmul recursiv descris anterior pornind de la nodurile de ieşire către înapoi, spre primul strat ascuns. Ponderile se modifică conform formulelor descrise anterior. Funcţia de eroare are expresii diferite pentru stratul de ieşire:

$$\delta_j = z_j \cdot (1-z_j) \cdot (d_j - z_j) \quad (27)$$

şi straturile ascunse:

$$\delta_j = x_j \cdot (1-x_j) \cdot \sum_k (\delta_k \cdot w_{jk}) \quad (28)$$

unde d_j este ieşirea dorită a nodului "*j*", y_j este ieşirea actuală, iar "*k*" parcurge toate nodurile din straturile de deasupra nodului *j*. Pragurile nodurilor interne sunt adaptate într-o manieră asemănătoare, presupunînd că ele sunt ponderi conectate la intrări cu valori constante.

Rumelhart, Hinton, Williams (1966) descriu o metodă de îmbunătăţire a timpului de antrenare a reţelelor backpropagation, atâta timp cât este respectată stabilitatea sistemului. Se

numeşte "momentum" şi aduce un termen în plus la ajustarea ponderilor, proporţional cu cantitatea schimbării anterioare. Această ajustare este făcută "cu memorare" şi serveşte la modificarea ponderilor din subsecvenţele următoare:

$$w_{ij}(t+1) = w_{ij}(t) + \eta \cdot \delta_j \cdot x_i' + \alpha \cdot (w_{ij}(t) - w_{ij}(t-1)) \qquad (29)$$

unde α este coeficientul "momentum", setat de obicei în jurul valorii de *0.9* $(0 < \alpha < 1)$.

După un număr suficient de repetări a acestor pasi, eroarea dintre ieşirea actuală şi vectorul ţintă se va reduce ca valoare, iar reţeaua se spune că este antrenată. Din acest punct, reţeaua este folosită pentru recunoaştere şi ponderile nu se mai schimbă.

Condiţia de stop poate fi una din următoarele:

1. Număr prestabilit de paşi;
2. Eroarea medie pătratică globală (pentru întreg setul de antrenare) scade sub un anumit prag;
3. Numărul de erori de clasificare pentru setul de antrenare este zero.

O teoremă interesantă ce pune în evidenţă capabilităţile perceptronului multinivel a fost enunţată de Kolmogorov. Această teoremă spune că orice funcţie continuă de *N* variabile poate fi calculată folosind numai sumări liniare şi neliniare, dar variind crescător continuu numai funcţii de o singură variabilă. Aceasta implică că un perceptron cu *3* nivele cu $N(2N+1)$ noduri folosind neliniarităţi variind crescător continuu, poate calcula orice funcţie continuă de o singură variabilă. Din păcate, teorema nu indică cum pot fi selectate ponderile şi neliniarităţile în reţea sau cît de sensibilă este funcţia de ieşire la variaţiile ponderilor şi a funcţiilor interne.

14.6. Alte tipuri de reţele neurale.

În continuare sunt prezentate foarte pe scurt câteva tipuri importante de reţele neurale. Pentru fiecare reţea neurală în parte se prezintă tipul de neuron, arhitectura reţelei şi algoritmul de antrenare. În ultimul paragraf al capitolului curent pot fi găsite implementări simple în limbajul C ale acestor reţele.

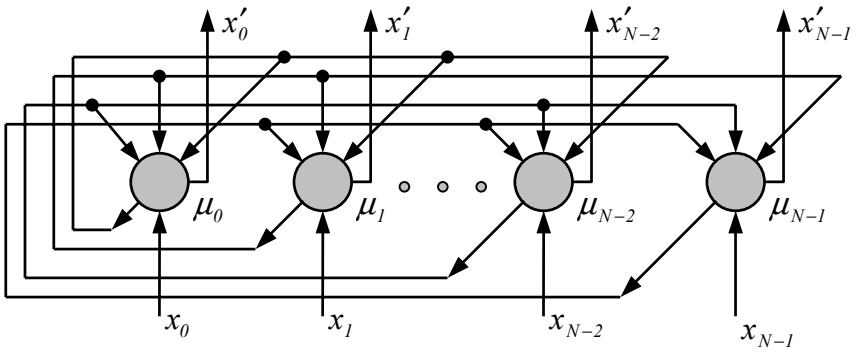
14.6.1. Reţeaua Hopfield.

Acest tip de reţea poate fi folosit ca memorie asociativă sau pentru rezolvarea unor probleme de optimizare. Foloseşte neuroni cu intrări binare, a căror ieşire conţine nelinearităţi de tipul limitare hardware (vezi primul paragraf al acestui capitol).

Arhitectura unei reţele Hopfield cu *N* neuroni este prezentată în figura 5. Ieşirea fiecărui neuron este aplicată tuturor celorlalte noduri prin intermediul unor ponderi t_{ij}. Hopfield a demonstrat că această reţea converge dacă ponderile sunt simetrice ($t_{ij} = t_{ji}$)

Deşi simplitatea ei o face atractivă, această reţea prezintă două limitări majore când este utilizată ca memorie adresabilă prin conţinut. În primul rând, numărul de forme prototip care pot fi stocate şi apoi regăsite corect în reţea este limitat la aproximativ 15% din numărul de neuroni. Dacă se încearcă stocarea prea multor forme prototip, reţeaua ar putea converge către un prototip fictiv. A doua limitare prezentă la reţeaua Hopfield apare dacă prototipurile memorate în reţea sunt foarte asemănătoare (prezintă un mare număr de biţi identici), caz în care reţeaua devine instabilă.

Fig. 5. Arhitectura rețelei Hopfield.



Dacă rețeaua Hopfield este folosită drept clasificator, ieşirea rețelei (după convergență) trebuie comparată cu fiecare formă prototip, după care se poate lua decizia de apartenență la una din clasele ataşate formelor prototip.

Algoritmul de antrenare a rețelei Hopfield.

Pasul 1. Inițializează ponderile conexiunilor:

$$t_{ij} = \begin{cases} \sum_{s=0}^{M-1} x_i^s x_j^s, i \neq j \\ 0, \qquad i = j, 0 \leq i \leq M-1, 0 \leq j \leq M-1 \end{cases} \tag{30}$$

unde t_{ij} este ponderea conexiunii de la nodul *"i"* la nodul *"j"*, iar x_i^s (unde $x_i^s \in \{-1,1\}$) este valoarea intrării *"i"* pentru un vector din clasa *"s"*.

Pasul 2. Aplică un vector de intrare necunoscut.

$$\mu_i(0) = x_i, 0 \leq i \leq N \tag{31}$$

unde este ieşirea nodului "i" la momentul "t".

Pasul 3. Iterează până la convergență.

$$\mu_j(t+1) = f_h[\sum_{i=0}^{N-1} t_{ij}\mu_i(t)], 0 \leq j \leq M-1 \tag{32}$$

unde funcția neliniară f_h este de tipul limitare hardware, descrisă deja în primul paragraf al acestui capitol. Operația se repetă până când toate ieşirile neuronilor rămân neschimbate. În acest caz, configurația ieşirilor coincide cu forma prototip cea mai apropiată de forma necunoscută de intrare.

Pasul 4. Repetă paşii 2-3 pentru orice altă formă necunoscută de intrare.

O implementare simplă a acestei rețele este prezentată în finalul capitolului.

14.6.2. Rețeaua Hamming.

Arhitectura rețelei Hamming este prezentată în figura 6.

Acest tip de rețea poate fi folosit drept clasificator pentru forme binare afectate de zgomot. Ea conține neuroni de tip prag logic. Se remarcă în arhitectura rețelei prezența a două straturi, primul (cel inferior) realizează o valoare proporțională cu distanța Hamming la un număr de *M* forme prototip. Stratul superior selectează nodul cu ieşirea maximă.

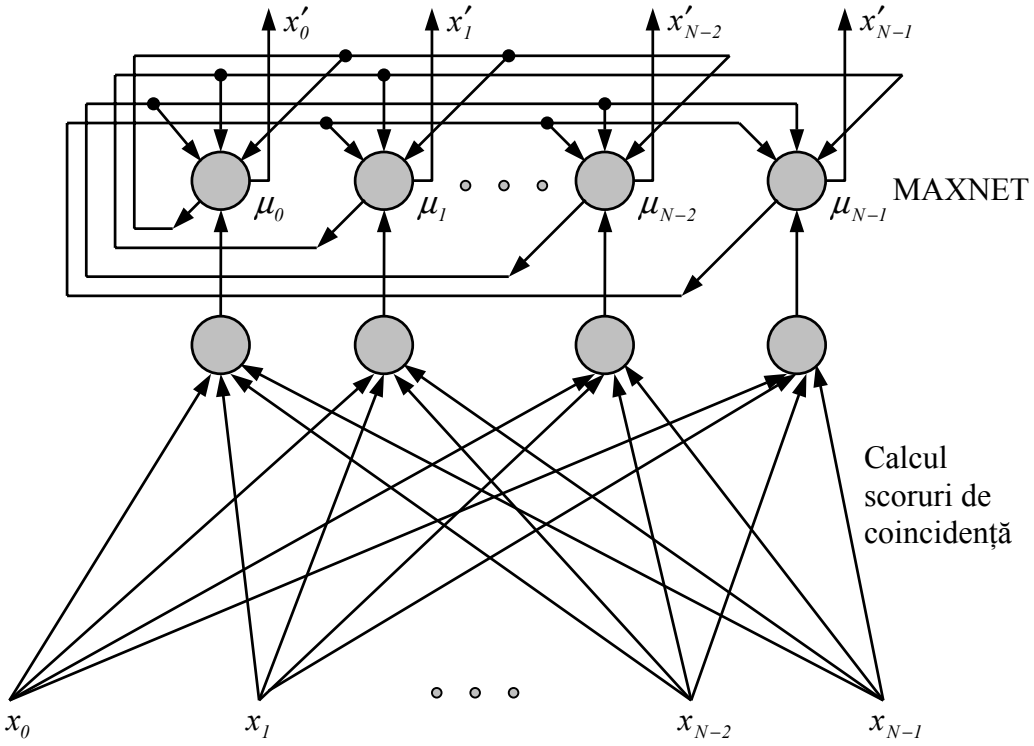


Fig. 6. Arhitectura rețelei Hamming.

Algoritmul de antrenare a rețelei Hamming.

Pasul 1. Inițializează ponderile conexiunilor şi offseturile:

În rețeaua inferioară

$$t_{ij} = \begin{cases} \sum_{s=0}^{M-1} x_i^s x_j^s, i \neq j \\ 0, \qquad i = j, 0 \leq i \leq M-1, 0 \leq j \leq M-1 \end{cases} \tag{33}$$

unde t_{ij} este ponderea conexiunii de la nodul *"i"* la nodul *"j"*, iar x_i^s (unde $x_i^s \in \{-1,1\}$) este valoarea intrării *"i"* pentru un vector din clasa *"s"*.

Pasul 2. Aplică un vector de intrare necunoscut.

$$\mu_i(0) = x_i, 0 \leq i \leq N \tag{34}$$

unde este ieşirea nodului "i" la momentul "t".

Pasul 3. Iterează până la convergență.

$$\mu_j(t+1) = \boldsymbol{f}_h [\sum_{i=0}^{N-1} t_{ij} \mu_i(t)], 0 \leq j \leq M-1 \tag{35}$$

unde funcția neliniară $\boldsymbol{f}_h$ este de tipul limitare hardware, descrisă deja în primul paragraf al acestui capitol. Operația se repetă până când toate ieşirile neuronilor rămân neschimbate. În acest caz, configurația ieşirilor coincide cu forma prototip cea mai apropiată de forma necunoscută de intrare.

Pasul 4. Repetă paşii 2-3 pentru orice altă formă necunoscută de intrare.

O implementare simplă a acestei rețele este prezentată în finalul capitolului.

14.6.3. Clasificatorul Carpenter-Grossberg.

Arhitectura unei rețele Carpenter-Grossberg cu trei neuroni pe stratul de intrare şi doi neuroni pe cel de ieşire este prezentată în figura următoare:

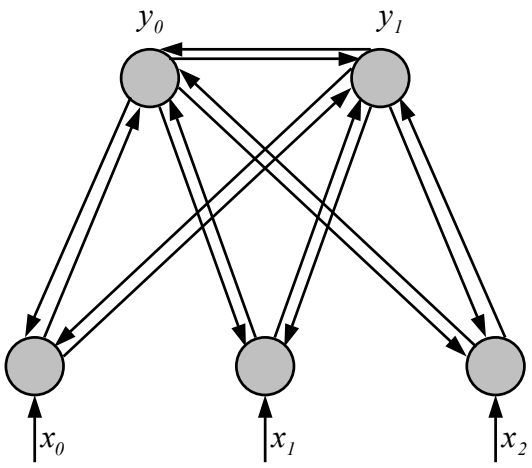


Fig. 7. Arhitectura rețelei Carpenter-Grossberg.

Această rețea implementează un clasificator nesupervizat, fără număr prestabilit de clase. Principial, prima formă de intrare din secvență este selectată ca fiind prima formă prototip. Apoi următoarea intrare este comparată cu vectorii prototip şi, funcție de distanța dintre ei, fie este atribuit primei clase, fie este selectată ca vector prototip al unei noi clase. Operația de atribuire a unei forme de intrare unei clase implică recalcularea prototipului asociat.

Algoritmul de antrenare a rețelei Carpenter-Grossberg.

Pasul 1. Inițializări:

$$t_{ij}(0) = 1,\ b_{ij}(0) = 1/(N+1),\ 0 \le i \le N-1,\ 0 \le j \le M-1 \quad (36)$$

Setează $\rho \in [0,1]$

unde b_{ij} este ponderea conexiunii dintre nodul *"i"* de intrare şi nodul *"j"* de ieşire, iar t_{ij} este ponderea conexiunii de la nodul *"j"* de ieşire la nodul *"i"* de intrare. Ele definesc prototipul corespunzător nodului *"j"* de ieşire. Parametrul ρ specifică cât de apropiat trebuie să fie o intrare față de un prototip pentru a lua decizia de apartenență la aceeaşi clasă.

Pasul 2. Aplică un vector de intrare necunoscut.

Pasul 3. Calculează scorurile de coincidență.

$$\mu_j = \sum_{i=0}^{N-1} b_{ij}(t)x_i, 0 \le j \le M-1 \quad (37)$$

unde μ_j este ieşirea nodului *"j"* iar x_i este elementul *"i"* al intrării (*0* sau *1*).

Pasul 4. Selectează cel mai apropiat prototip:

$$\mu_{j^*} = \max_j \{\mu_j\} \quad (38)$$

Aceasta se realizează folosind conexiunile inhibitorii laterale dintre neuronii de ieşire.

Pasul 5. Testul de vigilență:

$$\|X\| = \sum_{i=0}^{N-1} x_i \,, \quad \|T \cdot X\| = \sum_{i=0}^{N-1} t_{ij^*} x_i \tag{39}$$

Dacă $\|T \cdot X\| / \|X\| > \rho$ salt la **Pasul 7**, în caz contrar salt la **Pasul 6**.

Pasul 6. Inactivează cel mai apropiat prototip găsit în **Pasul 4** prin forțarea la zero a nodului corespunzător. Ca urmare acest prototip nu mai este luat în considerare la calculul maximului din **Pasul 4**.

Pasul 7. Adaptează cel mai apropiat prototip:

$$t_{ij^*}(t+1) = t_{ij^*}(t) x_i \tag{40}$$

$$b_{ij^*}(t+1) = [t_{ij^*}(t) x_i] / [0.5 + \sum_{i=0}^{N-1} t_{ij^*} x_i] \tag{41}$$

Pasul 8. Salt la **Pasul 2** pentru o nouă intrare.

O implementare simplă a acestei rețele este prezentată în finalul capitolului.

14.6.4. Harta de trăsături Kohonen.

Kohonen, plecând de la presupunerea că, deşi o parte a structurării creierului uman este predeterminată genetic, există organizare determinată prin învățare, pe baza unor algoritmi fiziologici care au la bază auto-organizarea în structurile "superioare" ale cortexului.

Folosind acest model el propune un tip de rețea neurală numită "harta de trăsături cu auto-organizare" (Kohonen self-organizing feature map). Împreună cu algoritmul de antrenare corespunzător acestei rețele se realizează practic un cuantizor vectorial, în mod iterativ prin ajustarea ponderilor conexiunilor între nodurile rețelei şi intrare.

Vectorii de intrare N-dimensionali (valori continue) sunt aplicați la intrare succesiv, ajustarea ponderilor corespunzătoare făcându-se folosind interconexiunile extensive care unesc nodurile rețelei. Fiecare neuron este conectat cu fiecare ieşire dar şi cu neuronii vecini prin conexiuni excitatorii/inhibitorii laterale. O remarcă importantă este faptul că fiecare nod este conectat cu neuronii din vecinătatea pătratică aferentă lui, dar dimensiunea acestei vecinătăți scade în timp.

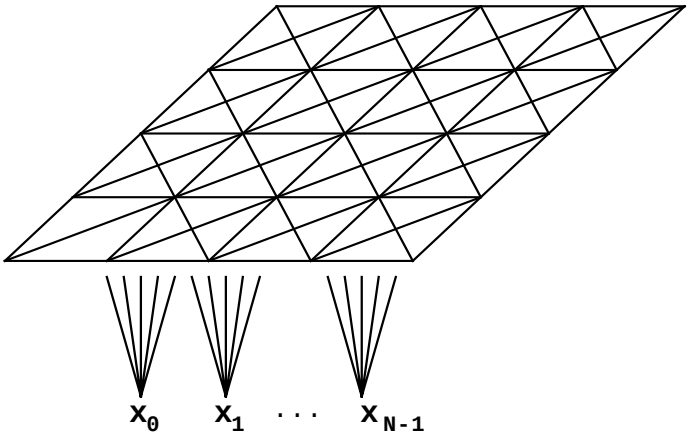


Fig. 8. Arhitectura rețelei Kohonen.

Algoritmul de antrenare a rețelei Kohonen este descris în caseta următoare:

Algoritmul de antrenare a rețelei Kohonen

(1). Inițializează ponderile.

Ponderile conexiunilor de la cele *N* intrări la cele *M* noduri se setează la valori aleatoare mici. Raza inițială a vecinătăților ataşate fiecărui nod se setează la valoarea maximă.

(2). Aplică o nouă intrare.

(3). Calculează distanță dintre vectorul de intrare şi vectorul de ponderi pentru fiecare nod.

$$D_j = \sum_{i=0}^{N-1} [\, \boldsymbol{x}_i(t) - \boldsymbol{w}_{ij}(t)]^2 \qquad (42)$$

unde: - $\boldsymbol{x}_i(t)$ este valoarea aplicată intrării "*i*" la momentul "*t*";

- $\boldsymbol{w}_{ij}(t)$ este ponderea conexiunii dintre intrarea "*i*" şi nodul "*j*" la momentul "*t*".

(4). Selectează nodul "k" de distanță minimă:

$$D_k = \min_{j=0,\ldots,M-1} D_j \qquad (43)$$

(5). Actualizează ponderile pentru nodul "k" şi toate nodurile din vecinătatea aferentă nodului "k" la momentul "t". Regula de ajustare a ponderilor este:

$$\boldsymbol{w}_{ij}(t+1) = \boldsymbol{w}_{ij}(t) + \boldsymbol{\eta}(t)[\, \boldsymbol{x}_i(t) - \boldsymbol{w}_{ij}(t)],\ i = 0,\ldots,N-1 \qquad (44)$$

Coeficientul $\boldsymbol{\eta}(t) \in (0,1)$ este factorul de câştig, care descreşte şi el în timp.

(6). Salt la pasul (2).

Calculul distanţei din pasul (3) poate fi redus la un simplu produs scalar, tipic pentru neuronul liniar dacă poate fi asigurată condiția normalizării vectorilor de intrare şi a vectorilor de ponderi aferenți după fiecare ajustare. Selectarea nodului care furnizează distanța minimă poate fi efectuată iterativ, după modelul rețelei *MAXNET*, care converge (furnizează la ieşire) valoarea maximă aplicată la intrare.

După antrenare, ponderile conexiunilor intrare-ieşire tind să specifice clusterii detectați în spațiul formelor de intrare prin centrele lor. Ca urmare, densitatea locală a acestor clusteri tinde să aproximeze densitatea de probabilitate calculată sau estimată pentru vectorii de formă care constituie setul de antrenament.

O implementare simplă a acestei rețele este prezentată în finalul capitolului.

14.6.5. Rețele RBF (Radial Basis Functions).

Rețelele RBF sunt rețele hibride cu două straturi, care conțin pe primul strat *M* neuroni ce implementează funcția Gaussiană normalizată de activare:

$$z_j = \exp(-|\boldsymbol{x} - \boldsymbol{m}_j|^2 / 2\sigma_j^2) \,/\, \sum_k \exp(-|\,\boldsymbol{x} - \boldsymbol{m}_j\,|^2 / 2\sigma_j^2) \qquad (45)$$

în timp ce cei *N* neuroni de pe stratul al doilea realizează suma ponderată a intrărilor, trecută printr-o nelinearitate sigmoidală.

$$y_k = \boldsymbol{f}\left(\sum_{j=1}^{M} w_{kj} z_j + \theta_k\right) \qquad (46)$$

În formulele anterioare $\boldsymbol{x}$ este vectorul de intrare, $\boldsymbol{m}_j$ (media) şi σ_j (dispersia) descriu formele prototip ataşate grupărilor create în domeniul de intrare, iar θ_k este valoarea pragului de activare.

Primul strat al rețelei realizează o partiționare a spațiului de intrare în grupări descrise prin medie şi dispersie, în timp ce stratul al doilea ia decizia de apartenență a formei de intrare la une din clase (o clasă poate include una sau mai multe grupări), funcție de configurația de activare a neuronilor de pe primul strat.

Arhitectura tipică a unei rețele RBF este prezentată în figura următoare:

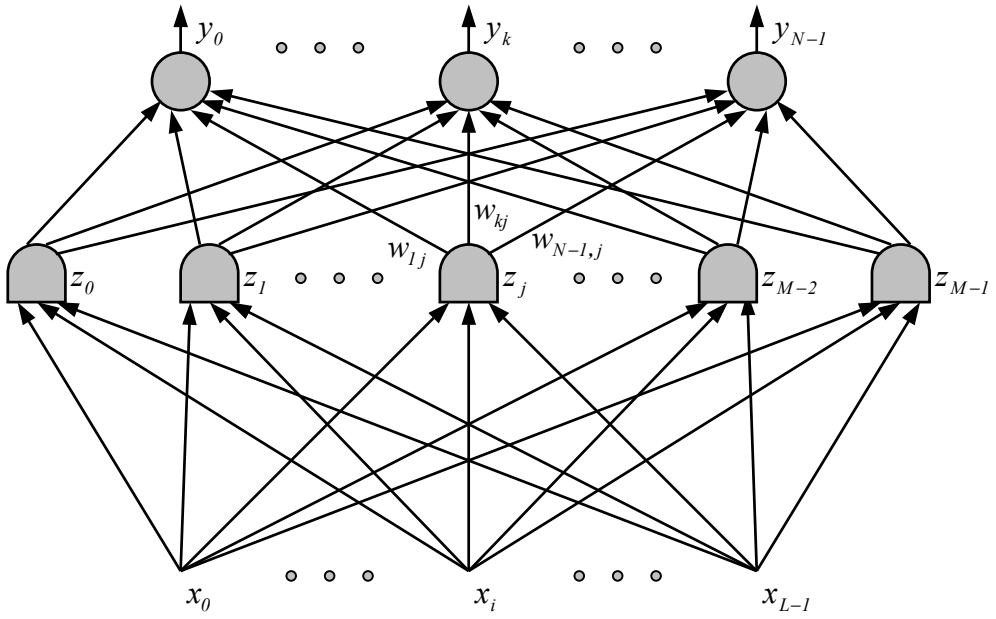


Fig. 9. Arhitectura rețelei RBF.

Antrenarea acestei rețele se face de obicei nesupervizat pentru stratul de intrare şi supervizat pentru stratul de ieşire, dar poate fi folosită şi metoda "backpropagation" de antrenare supervizată pentru ambele straturi.

Funcția de eroare corespunzătoare formei de intrare "p" este dată de:

$$E^{(p)} = \frac{1}{2}\sum_{k=1}^{N}(d_k^{(p)} - y_k^{(p)})^2 \tag{47}$$

Toți parametrii rețelei neurale se vor modifica în sensul minimizării acestei erori, folosind, de exemplu**,** metoda gradientului descrescător, adică:

$$\Delta w_{jk}^{(p)} = -\eta\frac{\partial E^{(p)}}{\partial w_{jk}^{(p)}} \qquad \Delta m_{ij}^{(p)} = -\eta\frac{\partial E^{(p)}}{\partial m_{ij}^{(p)}} \qquad \Delta\sigma_j^{(p)} = -\eta\frac{\partial E^{(p)}}{\partial\sigma_j^{(p)}} \tag{48}$$

Dacă se notează funcția de activare a nodului "k" de ieşire cu $a_k^{(p)}$, şi se defineşte $\delta_k^{(y)}$ ca fiind funcția de eroare corespunzătoare aceluiaşi nod, atunci derivata parțială a erorii medii pătratice în raport cu ponderea curentă se calculează cu:

$$\frac{\partial E^{(p)}}{\partial w_{jk}^{(p)}} = \frac{\partial E^{(p)}}{\partial a_k^{(p)}}\frac{\partial a_k^{(p)}}{\partial w_{jk}^{(p)}} = \frac{\partial E^{(p)}}{\partial a_k^{(p)}}\frac{\partial}{\partial w_{jk}^{(p)}}\left[\sum_{k*=1}^{N} w_{jk*}^{(p)} z_j^{(p)}\right] = \frac{\partial E^{(p)}}{\partial a_k^{(p)}} z_j^{(p)} \equiv \delta_k^{(y)} z_j^{(p)} \tag{49}$$

şi atunci se obține regula de ajustare a ponderilor:

$$w_{jk}^{(p)}(t+1) = w_{jk}^{(p)}(t) + \eta\delta_k^{(y)}(t) z_j^{(p)}(t) \tag{50}$$

Funcția de eroare poate fi evaluată folosind:

$$\delta_k^{(y)} = -\frac{\partial E^{(p)}}{\partial a_k^{(p)}} = -\frac{\partial E^{(p)}}{\partial y_k^{(p)}}\frac{\partial y_k^{(p)}}{\partial a_k^{(p)}} = -\frac{\partial E^{(p)}}{\partial y_k^{(p)}} f'(a_k^{(p)}) = (d_k^{(p)} - z_k^{(p)}) f'(a_k^{(p)}) \tag{51}$$

unde derivata funcției sigmoidale *asimetrice* în raport cu funcția de activare este dată de:

$$f'(a_k^{(p)}) = z_j^{(p)}(1 - z_j^{(p)}) \tag{52}$$

Oarecum similar se face adaptarea celorlalți parametri ai rețelei, media şi dispersia. Dacă se defineşte $\delta_j^{(z)}$ ca fiind funcția de eroare corespunzătoare nodului "j" de pe primul strat, atunci

derivata parțială a erorii medii pătratice în raport cu media/dispersia ataşată nodului curent se calculează cu:

$$\frac{\partial E^{(p)}}{\partial m_{ij}^{(p)}} = \frac{\partial E^{(p)}}{\partial z_j^{(p)}} \frac{\partial z_j^{(p)}}{\partial m_{ij}^{(p)}} = -\frac{\partial E^{(p)}}{\partial z_j^{(p)}} z_j^{(p)}(1 - z_j^{(p)}) \equiv \delta_j^{(z)} z_j^{(p)}(1 - z_j^{(p)}) \tag{53}$$

$$\frac{\partial E^{(p)}}{\partial \sigma_j^{(p)}} = \frac{\partial E^{(p)}}{\partial z_j^{(p)}} \frac{\partial z_j^{(p)}}{\partial \sigma_j^{(p)}} = \frac{\partial E^{(p)}}{\partial z_j^{(p)}} z_j^{(p)}(1 - z_j^{(p)}) \equiv \delta_j^{(z)} z_j^{(p)}(1 - z_j^{(p)}) \tag{54}$$

şi atunci se obține regula de ajustare a mediilor/dispersiilor:

$$m_{ij}^{(p)}(t+1) = m_{ij}^{(p)}(t) + \eta \delta_j^{(z)}(t) z_j^{(p)}(t)(1 - z_j^{(p)}(t)) \tag{55}$$

$$m_{ij}^{(p)}(t+1) = m_{ij}^{(p)}(t) + \eta \delta_j^{(z)}(t) z_j^{(p)}(t)(1 - z_j^{(p)}(t)) \tag{56}$$

Funcția de eroare poate fi evaluată folosind:

$$\delta_j^{(z)} = -\frac{\partial E^{(p)}}{\partial z_j^{(p)}} = \sum_{k=1}^{N} \delta_k^{(y)} w_{jk}^{(p)} \tag{57}$$

În afara unor formule de calcul diferite, algoritmul "backpropagation" pentru rețeaua RBF nu prezintă diferențe esențiale față de cel pentru perceptronul multistrat.

14.7. Detalii de implementare.

Acest paragraf include implementări simple, în limbajul C, ale rețelelor neurale de tip Hopfield, Hamming, Carpenter-Grossberg, Kohonen şi RBF, precum şi a perceptronului multistrat cu algoritmul backpropagation de antrenare. Dar pentru început iată un set de rutine (de alocare şi de generare numere aleatoare) utilizate în aceste implementări:

```
/*--------------------------------------------------------------------------*/
int *IVector(int N)
{
   int* v;
   v=(int *)malloc((unsigned)N*sizeof(int));
   if (!v) {
      printf("eroare de alocare 1");
      exit(1);
   }
   return v;
}
/*--------------------------------------------------------------------------*/
void Free_IVector(int *v, int N)
{
   free(v);
}
/*--------------------------------------------------------------------------*/
double *DVector(int N)
{
   double *v;
   v = (double *)malloc((unsigned)N*sizeof(double));
   if (!v) {
      printf("eroare de alocare 2");
      exit(1);
   }
   return v;
}
/*--------------------------------------------------------------------------*/
void Free_DVector(double *v,int N)
{
   free(v);
}
/*--------------------------------------------------------------------------*/
```

```
byte **BMatrix(int M, int N)
{
   byte** m;
   m = (byte **)malloc((unsigned) M*sizeof(byte*));
   if (!m) {
      printf("eroare de alocare 3");
      exit(1);
   }
   for (int i=0; i<M; i++) {
      m[i] = (byte *)malloc((unsigned) N*sizeof(byte));
      if (!m[i]) {
         printf("eroare de alocare 4");
         exit(1);
      }
   }
   return m;
}
/*--------------------------------------------------------------------------*/
void Free_BMatrix(byte **m, int M, int N)
{
   for (int i=0; i<M; i++)
      free(m[i]);
   free(m);
}
/*--------------------------------------------------------------------------*/
int **IMatrix(int M, int N)
{
   int** m;
   m = (int **)malloc((unsigned) M*sizeof(int*));
   if (!m) {
      printf("eroare de alocare 5");
      exit(1);
   }
   for (int i=0; i<M; i++) {
      m[i] = (int *)malloc((unsigned) N*sizeof(int));
      if (!m[i]) {
         printf("eroare de alocare 6");
         exit(1);
      }
   }
   return m;
}
/*--------------------------------------------------------------------------*/
void Free_IMatrix(int **m, int M, int N)
{
   for (int i=0; i<M; i++)
      free(m[i]);
   free(m);
}
/*--------------------------------------------------------------------------*/
double **DMatrix(int M, int N)
{
   double** m;
   m = (double **)malloc((unsigned) M*sizeof(double *));
   if (!m) {
      printf("eroare de alocare 7");
      exit(1);
   }
   for (int i=0; i<M; i++) {
      m[i] = (double *)malloc((unsigned) N*sizeof(double));
      if (!m[i]) {
```

```
         printf("eroare de alocare 8");
         exit(1);
      }
   }
   return m;
}
/*-------------------------------------------------------------------------*/
void Free_DMatrix(double **m, int M, int N)
{
   for (int i=0; i<M; i++)
      free(m[i]);
   free(m);
}
/*-------------------------------------------------------------------------*/
double ***TMatrix(int L,int M,int N)
{
   double*** m;
   int i,j;
   m=(double***)malloc((unsigned) L*sizeof(double**));
   if (!m) {
      printf("eroare de alocare 9");
      exit(1);
   }
   for(i=0;i<L;i++) {
      m[i]=(double**)malloc((unsigned) M*sizeof(double*));
      if (!m[i]) {
         printf("eroare de alocare 10");
         exit(1);
      }
   }
   for(i=0;i<L;i++) {
      for(j=0;j<M;j++) {
         m[i][j]=(double*)malloc((unsigned) N*sizeof(double));
         if (!m[i][j]) {
            printf("eroare de alocare 11");
            exit(2);
         }
      }
   }
   return m;
}
/*-------------------------------------------------------------------------*/
void Free_TMatrix(double ***m,int L,int M,int N)
{
   int i,j;
   for(i=0;i<L;i++)
      for(j=0;j<M;j++)
         free(m[i][j]);
   for(i=0;i<L;i++)
      free(m[i]);
   free(m);
}
```

Reţeaua Hopfield

```
#include <alloc.h>
#include <conio.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
```

```
#define NN 80
#define NC 4

int **Class;
struct Hopfield
{
   int *X;
   int **W;
   int *Y;
   int *Z;
} H;

int **IMatrix(int M,int N);
int *IVector(int N);
void Free_IMatrix(int **m,int M,int N);
void Free_IVector(int *v,int N);

void Define_Net(void);
void Train_Net(void);
int Use_Net(void);
/*-------------------------------------------------------------------------*/
void Define_Net(void)
{
   //alocări
   H.X=IVector(NN);
   H.W=IMatrix(NN,NN);
   H.Y=IVector(NN);
   H.Z=IVector(NN);
   Class=IMatrix(NC,NN);
}
/*-------------------------------------------------------------------------*/
void Train_Net()
{
   int i,j,k;

   for(i=0;i<NN;i++)
      for(j=0;j<NN;j++)
         if(i!=j)
            for(k=0,H.W[i][j]=0;k<NC;k++)
               H.W[i][j]+=Class[k][i]*Class[k][j];
         else
            H.W[i][j]=0;
}
/*-------------------------------------------------------------------------*/
int Use_Net(void)
{
   int i,j,k,m;
   m=0;
   do {
      for(i=0;i<NN;i++)       //salvează ieşiri
         H.Z[i]=H.Y[i];
      for(j=0;j<NN;j++) {     //actualizează ieşiri
         for(i=0,k=0;i<NN;i++)
            k+=H.W[i][j]*H.Z[i];
         if(k<0)
            H.Y[j]=-1;
         else
            H.Y[j]=1;
      }
      for(i=0,k=0;i<NN;i++)   //test convergenţă
```

```
         if(H.Z[i]!=H.Y[i])
            k++;
      printf("%4d",m++);
   }
   while(k!=0 && m<100);

   for(j=0;j<NC;j++) {
      for(i=0,k=0;i<NN;i++)
         if(H.Y[i]!=Class[j][i])
            k++;
      if(k==0)
         break;
   }
   return(j);
}


/*****************************************************************************/
main()
{
   int i,j;

   clrscr();
   Define_Net();

   //iniţializare clase
   printf("classes\n");
   for(i=0;i<NC;i++)
      for(j=0;j<NN;j++)
         if(sin((i+1)*(j+1))>=0.0)
            Class[i][j]=1,printf("X");
         else
            Class[i][j]=-1,printf("_");

   Train_Net();

   //aplică intrare
   for(i=0;i<NN;i++)
      H.Y[i]=Class[2][i];
   H.Y[0]=-1;
   H.Y[3]=1;
   H.Y[33]=-1;
   H.Y[66]=1;
   printf("input\n");
   for(i=0;i<NN;i++)
      if(H.Y[i]==1)
         printf("X");
      else
         printf("_");
   printf("steps ");

   //reiterează până la convergenţă
   j=Use_Net();
   printf("\nclass %d",j);

   //dealocări
   Free_IMatrix(Class,NC,NN);
   Free_IVector(H.Z,NN);
   Free_IVector(H.Y,NN);
   Free_IMatrix(H.W,NN,NN);
   Free_IVector(H.X,NN);
```

```
   getch();
   return(0);
}
```

Reţeaua Hamming

```
#include <alloc.h>
#include <conio.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

#define NN 80
#define NC 4

int **Class;
struct Hamming
{
   int *XI;       //intrare *2 în subreţeaua inferioară
   int **WI;      //ponderi *2 în subreţeaua inferioară
   int *TI;       //praguri *2 în subreţeaua inferioară
   double **WS;   //ponderi în subreţeaua superioară
   double *YS;    //ieşire curentă în subreţeaua superioară
   double *ZS;    //ieşire precedentă în subreţeaua superioară
} H;

int **IMatrix(int M,int N);
int *IVector(int N);
void Free_IMatrix(int **m,int M,int N);
void Free_IVector(int *v,int N);
double **DMatrix(int M,int N);
double *DVector(int N);
void Free_DMatrix(double **m,int M,int N);
void Free_DVector(double *v,int N);

/*************************************************************************/

main()
{
   int i,j,k,m;
   double eps,sum;

   clrscr();

   //alocări
   H.XI=IVector(NN);
   H.WI=IMatrix(NC,NN);
   H.TI=IVector(NC);
   H.WS=DMatrix(NC,NC);
   H.YS=DVector(NC);
   H.ZS=DVector(NC);
   Class=IMatrix(NC,NN);

   //iniţializare clase
   printf("classes\n");
   for(i=0;i<NC;i++)
      for(j=0;j<NN;j++)
         if(sin((i+1)*(j+1))>=0.0)
            Class[i][j]=1,printf("X");
         else
```

```
            Class[i][j]=-1,printf("_");

   eps=0.8/NC;
   //asignare ponderi şi praguri
   for(i=0;i<NC;i++)
      for(j=0;j<NN;j++)
         H.WI[i][j]=Class[i][j];
   for(i=0;i<NC;i++)
      H.TI[i]=NN;
   for(i=0;i<NC;i++)
      for(j=0;j<NC;j++)
         if(i==j)
            H.WS[i][j]=1.0;
         else
            H.WS[i][j]=-eps;

   //aplică intrare
   for(i=0;i<NN;i++)
      H.XI[i]=Class[3][i];
   H.XI[0]=-1, H.XI[3]=1, H.XI[33]=-1, H.XI[66]=1;
   for(i=0;i<NC;i++) {
      for(j=0,k=0;j<NN;j++)
         k+=H.WI[i][j]*H.XI[j];
      H.YS[i]=k-H.TI[i]/2.0;
      if(H.YS[i]<0.0)
         H.YS[i]=0.0;
   }
   printf("input\n");
   for(i=0;i<NN;i++)
      if(H.XI[i]==1)
         printf("X");
      else
         printf("_");
   printf("steps ");

   //reiterează până la convergenţă
   m=0;
   do {
      for(i=0;i<NC;i++)    //salvează ieşiri
         H.ZS[i]=H.YS[i];
      for(j=0;j<NC;j++) {  //actualizează ieşiri
         for(i=0,sum=0.0;i<NC;i++)
            if(i!=j)
               sum+=H.ZS[i];
         H.YS[j]=H.ZS[j]-eps*sum;
         if(H.YS[j]<0)
            H.YS[j]=0;
      }
      //test convergenţă
      for(i=0,k=0;i<NC;i++)
         if(H.YS[i]>0.0)
            k++;
      printf("%4d",m++);
   }
   while(k!=1 && m<100);

   //test apartenenţă
   for(i=0;i<NC;i++)
      if(H.YS[i]>0.0)
            break;
   printf("\nclass %d",i);
```

```
   //dealocări
   Free_IMatrix(Class,NC,NN);
   Free_DVector(H.ZS,NC);
   Free_DVector(H.YS,NC);
   Free_DMatrix(H.WS,NC,NC);
   Free_IVector(H.TI,NC);
   Free_IMatrix(H.WI,NC,NN);
   Free_IVector(H.XI,NN);
   getch();
   return(0);
}
```

Reţeaua Carpenter Grossberg

```
#include <alloc.h>
#include <conio.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

#define NN 80
#define NC 4

int **Input;

struct CarpGros
{
   int      *X;      /* intrare                    */
   double   **WB;    /* ponderi conexiuni directe  */
   int      **WT;    /* ponderi conexiuni inverse  */
   double   *Y;      /* ieşire                     */
   double   *Z;      /* inhibări laterale          */
} C;

int **IMatrix(int M,int N);
int *IVector(int N);
void Free_IMatrix(int **m,int M,int N);
void Free_IVector(int *v,int N);
double **DMatrix(int M,int N);
double *DVector(int N);
void Free_DMatrix(double **m,int M,int N);
void Free_DVector(double *v,int N);

/*************************************************************************/

main()
{
   int i,j,k,m,X;
   double p,Max,T;

   clrscr();

   //alocări
   C.X=IVector(NN);
   C.WB=DMatrix(NC,NN);
   C.WT=IMatrix(NC,NN);
   C.Y=DVector(NC);
   C.Z=DVector(NC);
   Input=IMatrix(NC+1,NN);
```

```
   //iniţializare Input
   printf("input string\n");
   for(i=0;i<NC;i++)
      for(j=0;j<NN;j++)
         if(sin((i+1)*(j+1))>=0.0)
            Input[i][j]=1,printf("X");
         else
            Input[i][j]=0,printf("_");
   for(i=0;i<NN;i++)
      Input[NC][i]=Input[1][i];
   Input[NC][0]=0, Input[NC][3]=1, Input[NC][33]=0, Input[NC][66]=1;
   for(i=0;i<NN;i++)
      if(Input[NC][i]==1)
         printf("X");
      else
         printf("_");

   //asignare ponderi
   p=0.9;
   for(i=0;i<NC;i++)
      for(j=0;j<NN;j++)
         C.WT[i][j]=1;
   for(i=0;i<NC;i++)
      for(j=0;j<NN;j++)
         C.WB[i][j]=1.0/(1.0+NN);
   for(i=0;i<NC;i++)
      C.Z[i]=1.0;

   //aplică succesiv intrări
   for(m=0;m<=NC;m++)
   {
      for(i=0;i<NN;i++)       //aplică intrare
         C.X[i]=Input[m][i];
      for(j=0;j<NC;j++)       //actualizează ieşiri
         for(i=0,C.Y[j]=0.0;i<NN;i++)
            C.Y[j]+=C.WB[j][i]*C.X[i];
P3:   Max=0.0;
      for(j=0;j<NC;j++)       //selectează forma tipică apropiată
         if(C.Y[j]*C.Z[j]>Max)
            k=j, Max=C.Y[j];
      for(i=0,X=0;i<NN;i++)   //test vigilenţă
         X+=C.X[i];
      for(i=0,T=0.0;i<NN;i++)
         T+=C.WT[k][i]*C.X[i];
      if(T/X > p) {
         for(i=0;i<NN;i++) {
            for(j=0,C.WB[k][i]=0.0;j<NN;j++)
               C.WB[k][i]+=C.WT[k][j]*C.X[j];
            C.WB[k][i]=C.WT[k][i]*C.X[i]/(0.5+C.WB[k][i]);
            C.WT[k][i]*=C.X[i];
         }
         for(i=0;i<NC;i++)
            C.Z[i]=1.0;
         printf("%d",k);
      }
      else {
         C.Z[k]=0.0;
         goto P3;
      }
   }
```

```
   //dealocări
   Free_IMatrix(Input,NC+1,NN);
   Free_DVector(C.Z,NC);
   Free_DVector(C.Y,NC);
   Free_IMatrix(C.WT,NC,NN);
   Free_DMatrix(C.WB,NC,NN);
   Free_IVector(C.X,NN);
   return(0);
}
```

Harta de trăsături Kohonen

```
#include <alloc.h>
#include <conio.h>
#include <graphics.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

#define NC 4
#define NN 2
#define MM 10
#define STEPS 5000

struct KohonenMap
{
   int      Rad;     /* raza inițială a vecinătății   */
   double   Eta;     /* raza inițială a vecinătății   */
   double   *XI;     /* intrare în rețea              */
   double   ***WC;   /* ponderi intrare-ieşire        */
   double   ***WP;   /* ponderi intrare-ieşire        */
} K;

double ***TMatrix(int L,int M,int N);
double *DVector(int N);
void Free_TMatrix(double ***m,int L,int M,int N);
void Free_DVector(double *v,int N);

/*************************************************************************/

main()
{
   int i,j,k,m,n,s;
   int GrDriver,GrMode;
   char Text[10];
   double d,Min;

   clrscr();

   //alocări
   K.XI=DVector(NN);
   K.WC=TMatrix(NN,MM,MM);
   K.WP=TMatrix(NN,MM,MM);

   //asignare ponderi inițiale
   K.Rad=3;
   K.Eta=0.7;
   srand(1);
   for(i=0;i<NN;i++)
```

```
   for(j=0;j<MM;j++)
      for(k=0;k<MM;k++)
         K.WC[i][j][k]=K.WP[i][j][k]=rand()/32767.0-0.5;

detectgraph(&GrDriver,&GrMode);
initgraph(&GrDriver,&GrMode,"C:\\bc31\\bgi");
settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
settextjustify(LEFT_TEXT,BOTTOM_TEXT);
setfillstyle(SOLID_FILL,0);
setcolor(15);
cleardevice();

//aplică succesiv intrări
for(s=0;s<STEPS;s++)
{
   sprintf(Text,"%4d",s);
   outtextxy(0,10,Text);
   for(i=0;i<NN;i++)
      K.XI[i]=rand()/327.67-50.0;

   //calculează nodul de distanță minimă
   Min=1000000.0;
   for(j=0;j<MM;j++)
      for(k=0;k<MM;k++) {
         for(i=0,d=0.0;i<NN;i++)
            d+=(K.XI[i]-K.WC[i][j][k])*(K.XI[i]-K.WC[i][j][k]);
         if(d<Min)
            Min=d, m=j, n=k;
      }

//actualizează ponderi
for(i=0;i<NN;i++)
   for(j=0;j<MM;j++)
      for(k=0;k<MM;k++)
         K.WP[i][j][k]=K.WC[i][j][k];
for(j=-K.Rad;j<=K.Rad;j++)
   if(m+j>=0 && m+j<MM)
      for(k=-K.Rad;k<=K.Rad;k++)
         if(n+k>=0 && n+k<MM)
            for(i=0;i<NN;i++)
               K.WC[i][m+j][n+k]=K.WP[i][m+j][n+k]+K.Eta
                                 *(K.XI[i]-K.WP[i][m+j][n+k]);

   //afișează poziție noduri
   cleardevice();
   for(j=0;j<MM;j++)
      for(k=0;k<MM;k++)
         putpixel(K.WC[0][j][k]+360,K.WC[1][j][k]+174,15);

   //actualizează parametri
   K.Rad=3.0*(STEPS-s)/STEPS+0.5;
   K.Eta=0.1*(STEPS-s)/STEPS;
   if(s==STEPS-1)
      getch();
}
closegraph();

//dealocări
Free_TMatrix(K.WP,NN,MM,MM);
Free_TMatrix(K.WC,NN,MM,MM);
Free_DVector(K.XI,NN);
```

```
   return(0);
}
```

Perceptron multistrat antrenat cu algoritmul "backpropagation"

```
#include <alloc.h>
#include <conio.h>
#include <dos.h>
#include <graphics.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define NN 80
#define N1 40
#define N2 20
#define NC 8
#define STEPS 200
#define ETA 0.35
#define ALPH 0.7

double **Input;
struct Perceptron
{
   double *XI;    /* intrare în reţea                                  */
   double **WI1p; /* ponderi precedente intrare -> primul strat        */
   double **WI1c; /* ponderi curente intrare -> primul strat           */
   double *T1p;   /* praguri precedente pentru primul strat            */
   double *T1c;   /* praguri curente pentru primul strat               */
   double *X1;    /* ieşiri din primul strat                           */
   double *S1;    /* erori pentru primul strat                         */
   double **W12p; /* ponderi precedente primul strat -> al 2-lea strat */
   double **W12c; /* ponderi curente primul strat -> al 2-lea strat    */
   double *T2p;   /* praguri precedente pentru al 2-lea strat          */
   double *T2c;   /* praguri curente pentru al 2-lea strat             */
   double *X2;    /* ieşiri din al 2-lea strat                         */
   double *S2;    /* erori pentru stratul al 2-lea                     */
   double **W2Yp; /* ponderi precedente al 2-lea strat -> ieşire       */
   double **W2Yc; /* ponderi curente al 2-lea strat -> ieşire          */
   double *TYp;   /* praguri precedente pentru stratul de ieşire       */
   double *TYc;   /* praguri curente pentru stratul de ieşire          */
   double *Y;     /* ieşire din reţea                                  */
   double *SY;    /* erori pentru stratul de ieşire                    */
   double *D;     /* ieşire dorită din reţea                           */
} P;

double   **DMatrix(int M,int N);
double   *DVector(int N);
void     Free_DMatrix(double **m,int M,int N);
void     Free_DVector(double *v,int N);
void     Alloc_Net(void);
void     DeAll_Net(void);
/*-------------------------------------------------------------------------*/
void Alloc_Net(void)
{
   P.XI=DVector(NN);
   P.WI1p=DMatrix(NN,N1);
   P.WI1c=DMatrix(NN,N1);
   P.T1p=DVector(N1);
```

```
   P.T1c=DVector(N1);
   P.X1=DVector(N1);
   P.S1=DVector(N1);
   P.W12p=DMatrix(N1,N2);
   P.W12c=DMatrix(N1,N2);
   P.T2p=DVector(N2);
   P.T2c=DVector(N2);
   P.X2=DVector(N2);
   P.S2=DVector(N2);
   P.W2Yp=DMatrix(N2,NC);
   P.W2Yc=DMatrix(N2,NC);
   P.TYp=DVector(NC);
   P.TYc=DVector(NC);
   P.Y=DVector(NC);
   P.SY=DVector(NC);
   P.D=DVector(NC);
}
/*-------------------------------------------------------------------------*/
void DeAll_Net(void)
{
   Free_DVector(P.D,NC);
   Free_DVector(P.SY,NC);
   Free_DVector(P.Y,NC);
   Free_DVector(P.TYc,NC);
   Free_DVector(P.TYp,NC);
   Free_DMatrix(P.W2Yc,N2,NC);
   Free_DMatrix(P.W2Yp,N2,NC);
   Free_DVector(P.S2,N2);
   Free_DVector(P.X2,N2);
   Free_DVector(P.T2c,N2);
   Free_DVector(P.T2p,N2);
   Free_DMatrix(P.W12c,N1,N2);
   Free_DMatrix(P.W12p,N1,N2);
   Free_DVector(P.S1,N1);
   Free_DVector(P.X1,N1);
   Free_DVector(P.T1c,N1);
   Free_DVector(P.T1p,N1);
   Free_DMatrix(P.WI1c,NN,N1);
   Free_DMatrix(P.WI1p,NN,N1);
   Free_DVector(P.XI,NN);
}

/*************************************************************************/

main()
{
   int i,j,k,m;
   int GrDriver,GrMode;
   char Text[10];
   double Save;

   clrscr();

   //alocări
   Alloc_Net();
   Input=DMatrix(4*NC+1,NN);

   // initializare Input
   printf("input string\n");
   for(i=0;i<4*NC;i++)
      for(j=0;j<NN;j++)
```

```
      if(sin((i+1)*(j+1))>=0.0)
         Input[i][j]=1.0,printf("X");
      else
         Input[i][j]=-1.0,printf("_");
for(i=0;i<NN;i++)
   Input[4*NC][i]=Input[2][i];
Input[4*NC][0]=-1.0;
Input[4*NC][3]=1.0;
Input[4*NC][31]=-1.0;
for(i=0;i<NN;i++)
   if(Input[4*NC][i]==1.0)
      printf("X");
   else
      printf("_");

//asignare ponderi şi praguri iniţiale
srand(1);
for(i=0;i<NN;i++)
   for(j=0;j<N1;j++)
      P.WI1p[i][j]=P.WI1c[i][j]=rand()/32767.0-0.5;
for(i=0;i<N1;i++)
   P.T1p[i]=P.T1c[i]=rand()/32767.0-0.5;
for(i=0;i<N1;i++)
   for(j=0;j<N2;j++)
      P.W12p[i][j]=P.W12c[i][j]=rand()/32767.0-0.5;
for(i=0;i<N2;i++)
   P.T2p[i]=P.T2c[i]=rand()/32767.0-0.5;
for(i=0;i<N2;i++)
   for(j=0;j<NC;j++)
      P.W2Yp[i][j]=P.W2Yc[i][j]=rand()/32767.0-0.5;
for(i=0;i<NC;i++)
   P.TYp[i]=P.TYc[i]=rand()/32767.0-0.5;

detectgraph(&GrDriver,&GrMode);
initgraph(&GrDriver,&GrMode,"C:\\bc31\\bgi");
settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
settextjustify(LEFT_TEXT,BOTTOM_TEXT);
setfillstyle(SOLID_FILL,0);
setcolor(1);
cleardevice();

//aplică succesiv intrări şi ieşirile dorite
for(k=0;k<STEPS;k++)
   for(m=0;m<4*NC;m++) {
      sprintf(Text,"%4d %3d",k,m);
      outtextxy(0,10,Text);

      for(i=0;i<NN;i++)       //aplică intrare
         P.XI[i]=Input[m][i];
      for(i=0;i<NC;i++)       //aplică ieşire dorită
         P.D[i]=0.0;
      P.D[m%8]=1.0;

      //actualizează ieşiri
      for(j=0;j<N1;j++) {
         for(i=0,P.X1[j]=0.0;i<NN;i++)
            P.X1[j]+=P.WI1c[i][j]*P.XI[i];
         P.X1[j]=1.0/(1.0+exp(-P.T1c[j]-P.X1[j]));
         //line(5*j,300,5*j,300.5-50.0*P.X1[j]);
      }
      for(j=0;j<N2;j++) {
```

```
   for(i=0,P.X2[j]=0.0;i<N1;i++)
      P.X2[j]+=P.W12c[i][j]*P.X1[i];
   P.X2[j]=1.0/(1.0+exp(-P.T2c[j]-P.X2[j]));
   //line(5*j,200,5*j,200.5-50.0*P.X2[j]);
}
for(j=0;j<NC;j++) {
   for(i=0,P.Y[j]=0.0;i<N2;i++)
      P.Y[j]+=P.W2Yc[i][j]*P.X2[i];
   P.Y[j]=1.0/(1.0+exp(-P.TYc[j]-P.Y[j]));
   //line(5*j,100,5*j,100.5-50.0*P.Y[j]);
}

//adaptează ponderi şi praguri
for(j=0;j<NC;j++) {
   P.SY[j]=P.Y[j]*(1.0-P.Y[j])*(P.D[j]-P.Y[j]);
   //line(5*j+400,100,5*j+400,100.5-50.0*P.SY[j]);
}
for(j=0;j<NC;j++) {
   for(i=0;i<N2;i++) {
      Save=P.W2Yc[i][j];
      P.W2Yc[i][j]+=ETA*P.SY[j]*P.X2[i]+ALPH
                    *(P.W2Yc[i][j]-P.W2Yp[i][j]);
      P.W2Yp[i][j]=Save;
   }
   Save=P.TYc[j];
   P.TYc[j]+=ETA*P.SY[j]+ALPH*(P.TYc[j]-P.TYp[j]);
   P.TYp[j]=Save;
}

for(i=0;i<N2;i++) {
   for(j=0,P.S2[i]=0.0;j<NC;j++)
      P.S2[i]+=P.W2Yp[i][j]*P.SY[j];
   P.S2[i]*=P.X2[i]*(1.0-P.X2[i]);
   //line(5*i+400,200,5*i+400,200.5-50.0*P.S2[i]);
}
for(j=0;j<N2;j++) {
   for(i=0;i<N1;i++) {
      Save=P.W12c[i][j];
      P.W12c[i][j]+=ETA*P.S2[j]*P.X1[i]+ALPH
                    *(P.W12c[i][j]-P.W12p[i][j]);
      P.W12p[i][j]=Save;
   }
   Save=P.T2c[j];
   P.T2c[j]+=ETA*P.S2[j]+ALPH*(P.T2c[j]-P.T2p[j]);
   P.T2p[j]=Save;
}

for(i=0;i<N1;i++) {
   for(j=0,P.S1[i]=0.0;j<N2;j++)
      P.S1[i]+=P.W12p[i][j]*P.S2[j];
   P.S1[i]*=P.X1[i]*(1.0-P.X1[i]);
   //line(5*i+400,300,5*i+400,300.5-50.0*P.S1[i]);
}
for(j=0;j<N1;j++) {
   for(i=0;i<NN;i++) {
      Save=P.WI1c[i][j];
      P.WI1c[i][j]+=ETA*P.S1[j]*P.XI[i]+ALPH
                    *(P.WI1c[i][j]-P.WI1p[i][j]);
      P.WI1p[i][j]=Save;
   }
   Save=P.T1c[j];
```

```
            P.T1c[j]+=ETA*P.S1[j]+ALPH*(P.T1c[j]-P.T1p[j]);
            P.T1p[j]=Save;
         }
         if(k==STEPS-1)
            getch();
         cleardevice();
      }
   closegraph();
   printf("\n antrenare terminată");

   for(i=0;i<NN;i++)             //aplică intrare
      P.XI[i]=Input[4*NC][i];
   for(j=0;j<N1;j++) {           //calculează ieşiri
      for(i=0,P.X1[j]=0.0;i<NN;i++)
         P.X1[j]+=P.WI1c[i][j]*P.XI[i];
      P.X1[j]=1.0/(1.0+exp(-P.T1c[j]-P.X1[j]));
   }
   for(j=0;j<N2;j++) {
      for(i=0,P.X2[j]=0.0;i<N1;i++)
         P.X2[j]+=P.W12c[i][j]*P.X1[i];
      P.X2[j]=1.0/(1.0+exp(-P.T2c[j]-P.X2[j]));
   }
   for(j=0;j<NC;j++) {
      for(i=0,P.Y[j]=0.0;i<N2;i++)
         P.Y[j]+=P.W2Yc[i][j]*P.X2[i];
      P.Y[j]=1.0/(1.0+exp(-P.TYc[j]-P.Y[j]));
   }
   for(j=0;j<NC;j++)
      printf("\n%12.6f ",P.Y[j]);

   //dealocări
   Free_DMatrix(Input,4*NC+1,NN);
   DeAll_Net();
   return(0);
}
```

Reţea RBF cu antrenare hibridă.

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

#define DBL_MAX   1.7976931348623158e+308
#define PI        3.1415926535
#define byte      unsigned char

#define OFF 20       // ofset geometric pentru construcţia claselor

#define NC  3        // numărul de clase
#define NC1 4        // numărul total de clase (inclusiv clasa de rejecţie)
#define NZ  256      // numărul total de clusteri
#define NI  2        // numărul de intrări în reţea (x,y)
#define NF  600      // numărul de forme per clasă
#define NN  (NC1*NF) // numărul total de forme de intrare

#define EPOCHS 200   // parametri antrenare
#define ETA    0.35
#define ALPH   0.7

double   **Input; // matrice date de intrare
```

```
byte     **Image; // matrice imagine bitmap
int      *Histo;  // histograma distribuţiei datelor de intrare

typedef struct tagBITMAPFILEHEADER {   // antet fişier bitmap
   unsigned short bfType;
   unsigned long  bfSize;
   unsigned short bfReserved1;
   unsigned short bfReserved2;
   unsigned long  bfOffBits;
} BITMAPFILEHEADER;

typedef struct tagBITMAPINFOHEADER {   // antet bitmap
   unsigned long  biSize;
   long           biWidth;
   long           biHeight;
   unsigned short biPlanes;
   unsigned short biBitCount;
   unsigned long  biCompression;
   unsigned long  biSizeImage;
   long           biXPelsPerMeter;
   long           biYPelsPerMeter;
   unsigned long  biClrUsed;
   unsigned long  biClrImportant;
} BITMAPINFOHEADER;

typedef struct tagRGBQUAD {            // paleta de culori
   byte  rgbBlue;
   byte  rgbGreen;
   byte  rgbRed;
   byte  rgbReserved;
} RGBQUAD;

struct RBFNet     // reţea RBF
{
   double *XI;    // intrare în reţea
   double **Mp;   // medii precedente
   double **Dp;   // dispersii precedente
   double **Mc;   // medii curente
   double **Dc;   // dispersii curente
   double *Z;     // ieşiri din primul strat

   double **Wp;   // ponderi precedente strat de ieşire
   double **Wc;   // ponderi curente strat de ieşire
   double *Tp;    // praguri precedente strat de ieşire
   double *Tc;    // praguri curente strat de ieşire
   double *Ey;    // funcţia de eroare
   double *Y;     // ieşire din reţea
   double *D;     // ieşire dorită din reţea
} R;

double   **DMatrix(int M,int N);
byte     **BMatrix(int M,int N);
double   *DVector(int N);
int      *IVector(int N);
void  Free_DMatrix(double **m,int M,int N);
void  Free_BMatrix(byte **m,int M,int N);
void  Free_DVector(double *v,int N);
void  Free_IVector(int *v,int N);
void  Alloc_Net(void);
void  Free_Net(void);
```

```
/*-------------------------------------------------------------------------*/
void Alloc_Net(void)
{
   //alocă primul strat
   R.XI=DVector(NI);
   R.Mp=DMatrix(NI,NZ);
   R.Dp=DMatrix(NI,NZ);
   R.Mc=DMatrix(NI,NZ);
   R.Dc=DMatrix(NI,NZ);
   R.Z=DVector(NZ);
   //alocă stratul secund
   R.Wp=DMatrix(NZ,NC);
   R.Wc=DMatrix(NZ,NC);
   R.Tp=DVector(NC);
   R.Tc=DVector(NC);
   R.Ey=DVector(NC);
   R.Y=DVector(NC);
   R.D=DVector(NC);
}
/*-------------------------------------------------------------------------*/
void Free_Net(void)
{
   //dealocă stratul secund
   Free_DVector(R.D,NC);
   Free_DVector(R.Y,NC);
   Free_DVector(R.Ey,NC);
   Free_DVector(R.Tc,NC);
   Free_DVector(R.Tp,NC);
   Free_DMatrix(R.Wc,NZ,NC);
   Free_DMatrix(R.Wp,NZ,NC);
   //dealocă primul strat
   Free_DVector(R.Z,NZ);
   Free_DMatrix(R.Dc,NI,NZ);
   Free_DMatrix(R.Mc,NI,NZ);
   Free_DMatrix(R.Dp,NI,NZ);
   Free_DMatrix(R.Mp,NI,NZ);
   Free_DVector(R.XI,NI);
}

/*-------------------------------------------------------------------------*/

main()
{
   int      idum, i, j, k, n, nEr;
   long     r, x1, y1, x2, y2, x3, y3;
   double   x, y, Sum, Save, Sgn, Err;

   // defineşte intrare
   r = (long)((480.0-OFF-OFF)/(2+sqrt(3.0)));
   x1 = r+OFF;
   y1 = r+OFF;
   x2 = 2*r+OFF;
   y2 = 479-r-OFF;
   x3 = 3*r+OFF;
   y3 = r+OFF;

   // creaza reţea
   Alloc_Net();
   // creează matrice date de intrare
   Input = DMatrix(NN, NI);
   // creează bitmap
```

```
Image = BMatrix(480, 640);

idum = -1;
ran1(&idum);
i = 0;
// generează formele din prima clasă
do {
   x = (ran1(&idum)*2-1)*r;
   y = (ran1(&idum)*2-1)*r;
   if (x*x+y*y <= r*r) {
      x += x1;
      y += y1;
      Input[i][0] = x/400.0;
      Input[i][1] = y/400.0;
      i++;
   }
} while(i < NF);
// generează formele din clasa a doua
do {
   x = (ran1(&idum)*2-1)*r;
   y = (ran1(&idum)*2-1)*r;
   if (x*x+y*y <= r*r) {
      x += x2;
      y += y2;
      Input[i][0] = x/400.0;
      Input[i][1] = y/400.0;
      i++;
   }
} while(i < 2*NF);
// generează formele din clasa a treia
do {
   x = (ran1(&idum)*2-1)*r;
   y = (ran1(&idum)*2-1)*r;
   if (x*x+y*y <= r*r) {
      x += x3;
      y += y3;
      Input[i][0] = x/400.0;
      Input[i][1] = y/400.0;
      i++;
   }
} while(i < 3*NF);
// generează formele din clasa de rejecţie
do {
   x = ran1(&idum)*(4*r+2*OFF-2)+1;
   y = ran1(&idum)*478+1;
   if(((x-x1)*(x-x1)+(y-y1)*(y-y1) > r*r)
   && ((x-x2)*(x-x2)+(y-y2)*(y-y2) > r*r)
   && ((x-x3)*(x-x3)+(y-y3)*(y-y3) > r*r)) {
      Input[i][0] = x/400.0;
      Input[i][1] = y/400.0;
      i++;
   }
} while(i < NN);

// iniţializează bitmap (fond alb)
for (j=0; j<480; j++) {
   for (i=0; i<640; i++)
      Image[j][i] = 0xff;
}
// trasează limitele claselor (trei cercuri tangente)
for (k=0; k<360; k++) {
```

```
   i = (int)(x1 + r *sin(k*PI/180));
   j = (int)(y1 + r *cos(k*PI/180));
   Image[j][i] = 0x00;
   i = (int)(x2 + r *sin(k*PI/180));
   j = (int)(y2 + r *cos(k*PI/180));
   Image[j][i] = 0x00;
   i = (int)(x3 + r *sin(k*PI/180));
   j = (int)(y3 + r *cos(k*PI/180));
   Image[j][i] = 0x00;
}
// trasează formele de intrare
for (k=0; k<NF; k++) {        // prima clasă: simbol x
   i = (int)(400.0*Input[k][0]);
   j = (int)(400.0*Input[k][1]);
   Image[j][i] = 0xaa;
   Image[j-1][i-1] = 0xaa; Image[j-1][i+1] = 0xaa;
   Image[j+1][i-1] = 0xaa; Image[j+1][i+1] = 0xaa;
}
for (k=NF; k<NF*2; k++) {     // a doua clasă: simbol +
   i = (int)(400.0*Input[k][0]);
   j = (int)(400.0*Input[k][1]);
   Image[j][i] = 0xaa;
   Image[j][i-1] = 0xaa; Image[j][i+1] = 0xaa;
   Image[j-1][i] = 0xaa; Image[j+1][i] = 0xaa;
}
for (k=NF*2; k<NF*3; k++) {   // a treia clasă: simbol o
   i = (int)(400.0*Input[k][0]);
   j = (int)(400.0*Input[k][1]);
   Image[j-1][i-1] = 0xaa; Image[j-1][i] = 0xaa; Image[j-1][i+1] = 0xaa;
   Image[j][i-1] = 0xaa; Image[j][i+1] = 0xaa;
   Image[j+1][i-1] = 0xaa; Image[j+1][i] = 0xaa; Image[j+1][i+1] = 0xaa;
}
for (k=NF*3; k<NN; k++) {     // clasa de rejecţie: simbol I
   i = (int)(400.0*Input[k][0]);
   j = (int)(400.0*Input[k][1]);
   Image[j-1][i-1] = 0xaa; Image[j-1][i] = 0xaa; Image[j-1][i+1] = 0xaa;
   Image[j][i] = 0xaa;
   Image[j+1][i-1] = 0xaa; Image[j+1][i] = 0xaa; Image[j+1][i+1] = 0xaa;
}

////////////////////////////
// antrenare strat intrare //
////////////////////////////
//construieşte nesupervizat prototipuri pentru primul strat
int z = 1;  // contor clase (una singură la început, clasa 0)
for (i=0; i<NI; i++) {
   for (n=0, R.Mc[i][0]=0.0; n<NN; n++)
      R.Mc[i][0] += Input[n][i];
   R.Mc[i][0] /= NN;    // medie
   for (n=0, R.Dc[i][0]=0.0; n<NN; n++)
      R.Dc[i][0] += (Input[n][i]-R.Mc[i][0])*(Input[n][i]-R.Mc[i][0]);
   R.Dc[i][0] /= NN;    // dispersie
}

int s = 1;  // offset poziţie neuron curent în stratul de intrare
do {
   if(s*2 > NZ) break;
   s = s*2;
} while(1);

double DP, DC, eps=0.001;
```

```
do { //while z<NZ
   srand(1);
   if(z == s) {
      // completează numărul de clase până la NZ
      for (j=0; j<NZ-z; j++) {
         for (i=0; i<NI; i++) {
            Sgn=(rand()/32767000.0-0.0005);
            R.Mp[i][j*2] = R.Mc[i][j]+Sgn*R.Dc[i][j];
            Sgn=(rand()/32767000.0-0.0005);
            R.Mp[i][j*2+1] = R.Mc[i][j]+Sgn*R.Dc[i][j];
         }
      }
      // păstrează restul claselor
      for (i=0; i<NI; i++) {
         for (j=2*(NZ-z); j<NZ; j++)
            R.Mp[i][j] = R.Mc[i][j];
      }
      z = NZ;
   } else {
      // dublează numărul de clase
      for (j=0; j<z; j++) {
         for (i=0; i<NI; i++) {
            Sgn=(rand()/32767000.0-0.0005);
            R.Mp[i][j*2] = R.Mc[i][j] + Sgn*R.Dc[i][j];
            Sgn=(rand()/32767000.0-0.0005);
            R.Mp[i][j*2+1] = R.Mc[i][j] - Sgn*R.Dc[i][j];
         }
      }
      z = z*2;
   }

   // LBG: bucla de minimizare a erorii de aproximare pentru 'z' clase
   DC=DBL_MAX;
   do {
      DP=DC;
      DC=0;
      // Iniţializări
      Histo = IVector(z);
      for (j=0; j<z; j++) {
         for (i=0; i<NI; i++) {
            R.Mc[i][j] = 0.0;
            R.Dc[i][j] = 0.0;
         }
         Histo[j] = 0;
      }
      // află pentru fiecare vector de intrare partiţia cea mai apropiată
      for (n=0; n<NN; n++) {
         double   dMinDist2, dDist2;
         int      nMinClass=-1;
         for (j=0, dMinDist2=DBL_MAX; j<z; j++) {
            for (i=0, dDist2=0.0; i<NI; i++)
               dDist2 += (Input[n][i]-R.Mp[i][j])*(Input[n][i]-R.Mp[i][j]);
            if(dDist2 < dMinDist2) {
               dMinDist2 = dDist2;
               nMinClass = j;
            }
         }
         if (nMinClass == -1)
            exit(1);
         Histo[nMinClass]++;
         // actualizează eroarea medie pătratică de aproximare
```

```
            DC += dMinDist2;
            // valoarea contribuţiei la noua medie & dispersie
            for (i=0; i<NI; i++) {
               R.Mc[i][nMinClass] += Input[n][i];
               R.Dc[i][nMinClass] += Input[n][i]*Input[n][i];
            }
         }
         for (j=0; j<z; j++) {
            for (i=0; i<NI; i++) {
               R.Mc[i][j] = R.Mc[i][j]/Histo[j];   // calcul final medie
               R.Dc[i][j] = R.Dc[i][j]/Histo[j];
               R.Dc[i][j] -= R.Mc[i][j]*R.Mc[i][j];// calcul final dispersie
            }
         }
         for (j=0; j<z; j++)
            printf("%2d ", Histo[j]);
         printf("\neroare absoluta = %f\n",DC);
         Free_IVector(Histo, z);
      } while ((DP-DC)/DC > eps);
      printf("eroare relativa = %f\n",(DP-DC)/DC);
      printf("partitionare %d clase terminata\n\n",z);
   } while (z < NZ);
   // marchează în imagine poziţiile centrelor grupărilor
   for (z=0; z<NZ; z++) {
      i = (int)(R.Mc[0][z]*400.0);
      j = (int)(R.Mc[1][z]*400.0);
      Image[j][i] = 0x00;
   }

   ////////////////////////////
   // antrenare strat ieşire //
   ////////////////////////////
   // asignare ponderi şi praguri iniţiale (stratul de ieşire)
   srand(1);
   for (k=0; k<NC; k++) {
      for (j=0; j<NZ; j++)
         R.Wp[j][k] = R.Wc[j][k] = rand()/32767.0 - 0.5;
      R.Tp[k] = R.Tc[k] = rand()/32767.0 - 0.5;
   }

   // aplică succesiv intrări şi ieşirile dorite, adaptează parametri reţea
   for (s=0; s<EPOCHS; s++) {
      Err = 0.0; nEr = 0;
      for (n=0; n<NN; n++) {
         // aplică intrare în ordinea 0,1,2,rej,0,1,2,rej,...
         for (i=0; i<NI; i++)
            R.XI[i] = Input[(n%NC1)*NF+n/NC1][i];
         // aplică ieşire dorită
         for (i=0; i<NC; i++)
            R.D[i] = 0.0;
         if (n%NC1 < 3)    // ordinea 0,1,2,rej,0,1,2,rej,...
            R.D[n%NC1] = 1.0;
         // calculează ieşiri din primul strat
         for (j=0; j<NZ; j++) {
            for(i=0, R.Z[j]=0.0; i<NI; i++) {
               if (R.Dc[i][j] > 0)
                  R.Z[j] += (R.XI[i]-R.Mc[i][j])
                            *(R.XI[i]-R.Mc[i][j])/R.Dc[i][j];
            }
            R.Z[j] = exp(-R.Z[j]/2.0);
         }
```

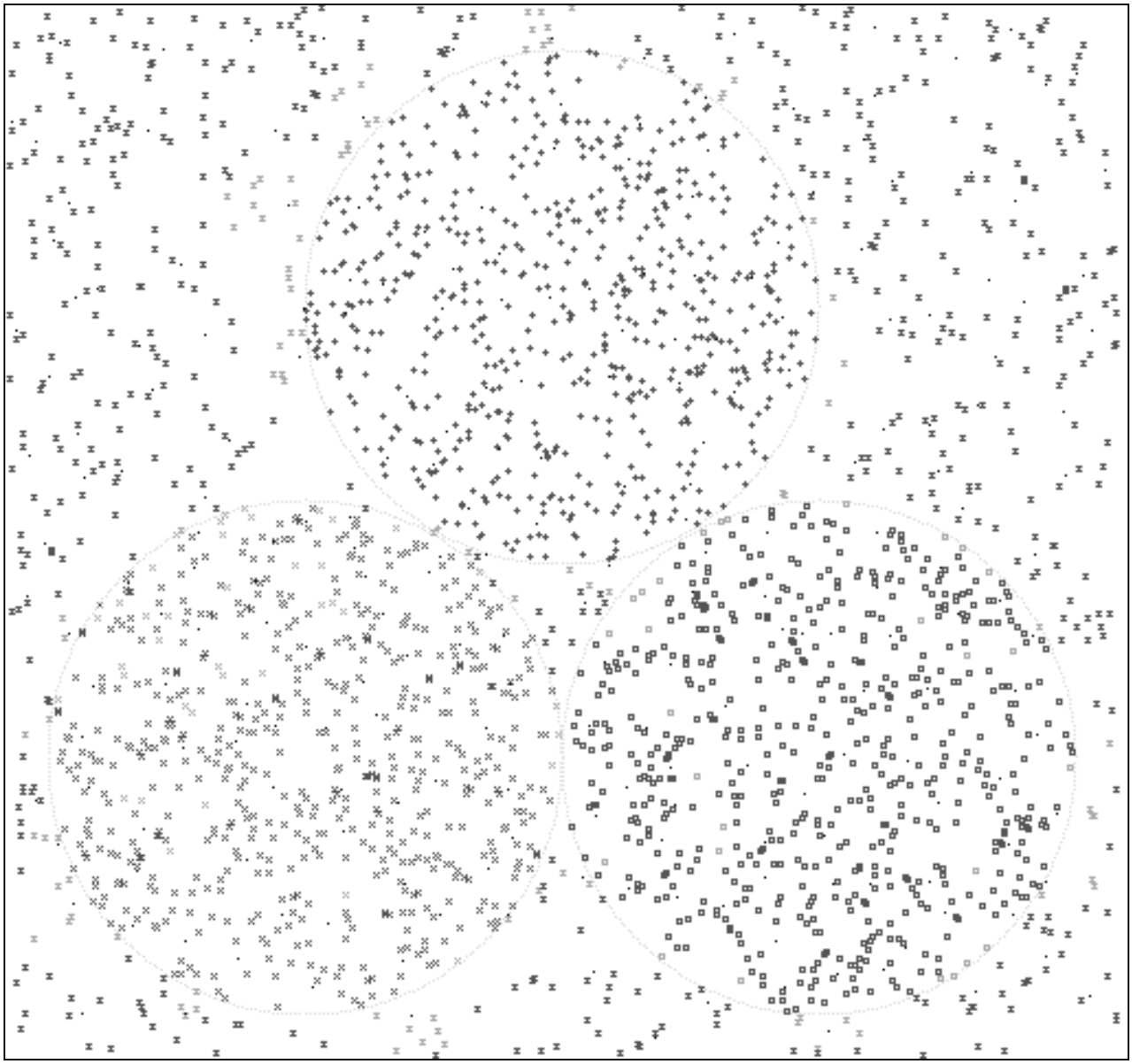
```
      // normalizare
      for (j=0, Sum=0.0; j<NZ; j++)
         Sum += R.Z[j];
      for (j=0; j<NZ; j++)
         R.Z[j] /= Sum;
      // calculează ieşiri strat de ieşire
      for (k=0; k<NC; k++) {
         for (j=0, R.Y[k]=0.0; j<NZ; j++)
            R.Y[k]+=R.Wc[j][k]*R.Z[j];
         R.Y[k] = 1.0 / (1.0 + exp(-R.Tc[k]-R.Y[k]));
      }
      // calculează eroare
      for (k=0; k<NC; k++)
         Err += (R.Y[k]-R.D[k])*(R.Y[k]-R.D[k]);
      // test clasificare corectă
      for (k=0; k<NC; k++) {
         if (k == (n%NC1)) {
            if (R.Y[k] < 0.5)
               break;
         } else {
            if (R.Y[k] > 0.5)
               break;
         }
      }
      if (k != NC)
         nEr++;   // incrementeaza numărul de erori
      // adaptează ponderi şi praguri (stratul de ieşire)
      for (k=0; k<NC; k++)
         R.Ey[k] = R.Y[k] * (1.0-R.Y[k]) * (R.D[k]-R.Y[k]);
      for (k=0; k<NC; k++) {
         for (j=0; j<NZ; j++) {
            Save = R.Wc[j][k];
            R.Wc[j][k] += ETA*R.Ey[k]*R.Z[j]
                        + ALPH*(R.Wc[j][k]-R.Wp[j][k]);
            R.Wp[j][k] = Save;
         }
         Save = R.Tc[k];
         R.Tc[k] += ETA*R.Ey[k] + ALPH*(R.Tc[k]-R.Tp[k]);
         R.Tp[k] = Save;
      }
      printf("step=%4d\r", n);
   }
   printf("\repoch=%4d err=%12.6f ner=%d\n", s, Err, nEr);
}
printf("\n antrenare terminata\n");

// test final
nEr = 0;
for (n=0; n<NN; n++) {
   // aplică intrare
   for (i=0; i<NI; i++)
      R.XI[i] = Input[n][i];
   // calculează ieşiri din primul strat
   for (j=0; j<NZ; j++) {
      for(i=0, R.Z[j]=0.0; i<NI; i++) {
         if (R.Dc[i][j] > 0)
            R.Z[j] += (R.XI[i]-R.Mc[i][j])*(R.XI[i]-R.Mc[i][j])/R.Dc[i][j];
      }
      R.Z[j] = exp(-R.Z[j]/2.0);
   }
   // normalizare
```

```
   for (j=0, Sum=0.0; j<NZ; j++)
      Sum += R.Z[j];
   for (j=0; j<NZ; j++)
      R.Z[j] /= Sum;
   // calculează ieşiri strat de ieşire
   for (k=0; k<NC; k++) {
      for (j=0, R.Y[k]=0.0; j<NZ; j++)
         R.Y[k] += R.Wc[j][k] * R.Z[j];
      R.Y[k] = 1.0/(1.0+exp(-R.Tc[k]-R.Y[k]));
   }
   // test clasificare corectă
   for (k=0; k<NC; k++) {
      if (k == n/NF) {
         if (R.Y[k] < 0.5)
            break;
      } else {
         if (R.Y[k] > 0.5)
            break;
      }
   }
   if (k != NC)
      nEr++;   // incrementează numărul de erori
   else {
      // clasificare corectă, marchează intrările recunoscute corect
      switch (n/NF) {
         case 0:  // prima clasă
            i = (int)(400.0*Input[n][0]);
            j = (int)(400.0*Input[n][1]);
            Image[j][i] = 0x55;
            Image[j-1][i-1] = 0x55; Image[j-1][i+1] = 0x55;
            Image[j+1][i-1] = 0x55; Image[j+1][i+1] = 0x55;
            break;
         case 1:  // a doua clasă
            i = (int)(400.0*Input[n][0]);
            j = (int)(400.0*Input[n][1]);
            Image[j][i] = 0x55;
            Image[j][i-1] = 0x55; Image[j][i+1] = 0x55;
            Image[j-1][i] = 0x55; Image[j+1][i] = 0x55;
            break;
         case 2:  // a treia clasă
            i = (int)(400.0*Input[n][0]);
            j = (int)(400.0*Input[n][1]);
            Image[j-1][i-1] = 0x55; Image[j-1][i] = 0x55;
            Image[j-1][i+1] = 0x55; Image[j][i-1] = 0x55;
            Image[j][i+1] = 0x55; Image[j+1][i-1] = 0x55;
            Image[j+1][i] = 0x55; Image[j+1][i+1] = 0x55;
            break;
         case 3:  // clasa de rejectie
            i = (int)(400.0*Input[n][0]);
            j = (int)(400.0*Input[n][1]);
            Image[j-1][i-1] = 0x55; Image[j-1][i] = 0x55;
            Image[j-1][i+1] = 0x55; Image[j][i] = 0x55;
            Image[j+1][i-1] = 0x55; Image[j+1][i] = 0x55;
            Image[j+1][i+1] = 0x55;
            break;
      }
   }
}
printf("\nTotal %d erori din %d\n", nEr, NN);
// construcţie fişier bitmap
// antet fişier
```

```
   BITMAPFILEHEADER bmfh;
   bmfh.bfType = 0x4d42;
   bmfh.bfSize = sizeof(BITMAPFILEHEADER)+sizeof(BITMAPINFOHEADER)
               + 2*sizeof(RGBQUAD)+640*480;
   bmfh.bfReserved1 = 0;
   bmfh.bfReserved2 = 0;
   bmfh.bfOffBits = sizeof(BITMAPFILEHEADER)+sizeof(BITMAPINFOHEADER)
               + 256*sizeof(RGBQUAD);
   // antet bitmap
   BITMAPINFOHEADER bmih;
   bmih.biSize = sizeof(BITMAPINFOHEADER);
   bmih.biWidth = 640;
   bmih.biHeight = 480;
   bmih.biPlanes = 1;
   bmih.biBitCount = 8;
   bmih.biCompression = 0;
   bmih.biSizeImage = 640*480/8;
   bmih.biXPelsPerMeter = 0;
   bmih.biYPelsPerMeter = 0;
   bmih.biClrUsed = 0;
   bmih.biClrImportant = 0;
   // paleta de culori
   RGBQUAD rgbq[256];
   for (i=0; i<256; i++) {
      rgbq[i].rgbBlue = rgbq[i].rgbGreen = rgbq[i].rgbRed = i;
      rgbq[i].rgbReserved = 0;
   }
   // scrie fişier bitmap
   FILE* pFile = fopen("c:\\rbf.bmp", "wb");
   fwrite(&bmfh, sizeof(BITMAPFILEHEADER), 1, pFile);
   fwrite(&bmih, sizeof(BITMAPINFOHEADER), 1, pFile);
   for (i=0; i<256; i++)
      fwrite(&rgbq[i], sizeof(RGBQUAD), 1, pFile);
   for (j=0; j<480; j++) {
      for (i=0; i<640; i++)
         fwrite(&Image[j][i], 1, 1, pFile);
   }
   fclose(pFile);

   // dealocări
   Free_BMatrix(Image, 480, 640);
   Free_DMatrix(Input, NN, NI);
   Free_Net();
   return(0);
}
```



Fişier imagine (bitmap) generat de programul te test pentru rețeaua RBF cu antrenare hibridă.

15. Selecția trăsăturilor.

Dintr-un proces oarecare se pot extrage un număr foarte mare de trăsături. Odată cu creşterea numărului de trăsături folosite, creşte însă şi complexitatea clasificatorului. De aceea se pune problema selecției acelor trăsături care sunt esențiale în procesul de recunoaştere.

15.1. Metode bazate pe estimarea caracteristicilor statistice ale trăsăturilor.

Fie $\{\boldsymbol{X}_j\}_{1\le j\le N}$ setul de N vectori de date folosiți pentru antrenarea clasificatorului. Notând cu x_{ij} trăsătura "j" a vectorului $\boldsymbol{X}_i$ şi cu N_k numărul de forme prototip pentru clasa "k", avem:

$$N = \sum_{k=1}^{N_C} N_k \tag{1}$$

şi putem defini în continuare:

Media trăsăturii "j" pentru vectorii din setul de date:

$$m_j = \frac{1}{N}\sum_{i=1}^{N} x_{ij} \Leftrightarrow m_j = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{N_C} x_{ij}^{(k)} \tag{2}$$

Varianța trăsăturii "j" pentru setul de vectori de date:

$$v_j = \frac{1}{N}\sum_{i=1}^{N}(x_{ij} - m_j)^2 \Leftrightarrow v_j = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{N_C}(x_{ij}^{(k)} - m_j)^2 \tag{3}$$

Dacă v_j are valori foarte mici (nesemnificative), trăsătura "j" poate fi eliminată din setul de trăsături folosite pentru definirea formelor, ea nereuşind să separe bine clasele.

Variabilitatea intraclasă pentru trăsătura "j" se defineşte prin:

$$V_j^{(in)} = \frac{1}{p}\sum_{(k,l)}(x_{kj} - x_{lj})^2 \tag{4}$$

unde suma se efecuează pentru toate perechile de forme din setul de date aparținând aceleiaşi clase, "p" fiind numărul de astfel de perechi de forme.

Variabilitatea interclasă pentru trăsătura "*j*" este dată de:

$$V_j^{(ic)} = \frac{1}{r}\sum_{(k,l)}(x_{kj} - x_{lj})^2 \tag{5}$$

unde suma se calculează pentru toate perechile posibile de forme aparţinând unor clase diferite, "*r*" fiind numărul de astfel de perechi.

Cu cât $V_j^{(in)}$ este mai mic, cu atât gruparea valorilor trăsăturii în jurul mediei este mai bună, deci clasificarea va fi mai facilă, şi cu cât $V_j^{(ic)}$ este mai mare, cu atât clasele sunt mai depărtate, deci mai bine separate. Atunci, pentru fiecare trăsătura în parte se calculează raportul:

$$g_j = \frac{V_j^{(in)}}{V_j^{(ic)}} \tag{6}$$

reţinându-se acele trăsături pentru care valoarea g_j este cât mai mică.

Se mai utilizează pentru problemele de clasificare binară raportul (ponderea) Fisher:

$$F_j = \frac{(m_{j1} - m_{j2})^2}{V_{j1} - V_{j2}} \tag{7}$$

sau raportul Coomans:

$$C_j = \frac{m_{j1} - m_{j2}}{\sqrt{V_{j1}} + \sqrt{V_{j2}}} \tag{8}$$

15.2. Metode bazate pe densitatea de probabilitate.

Pentru un clasificator bayesian cu funcţie de pierdere simetrică, riscul condiţionat este dat de:

$$\boldsymbol{L(X,\omega_k) = p(X/\omega_k)\, p(\omega_k)} \tag{9}$$

Eroarea globală de clasificare este dată de:

$$\varepsilon = \sum_{k=1}^{Nc} \int_{\overline{\omega}_k} \boldsymbol{L(X,\omega_k)dX} \tag{10}$$

de unde se poate scrie:

$$\varepsilon = \sum_{k=1}^{Nc} \boldsymbol{p(\omega_k)} \int_{\overline{\omega}_k} \boldsymbol{p(\,X/\omega_k)d\,X} \tag{11}$$

O expresie asemănătoare se poate scrie şi pentru fiecare trăsătura în parte:

$$\varepsilon_j = \sum_{k=1}^{Nc} \boldsymbol{p(\omega_k)} \int_{\overline{\omega}_{k(j)}} \boldsymbol{p(x_j/\omega_k)dx_j} \tag{12}$$

Pentru fiecare trăsătură în parte se calculează ε_j, cunoscând aprioric sau estimând $\boldsymbol{p(\omega_k)}$ şi $\boldsymbol{p(x_j/\omega_k)}$, iar apoi se reţin acele trăsături pentru care ε_j este minim şi eliminându-le pe celelalte ca fiind nesemnificative.

15.3. Transformări diagonale pentru selecţia trăsăturilor.

Aceste metode permit mărirea gradului de similaritate a formelor dintr-o clasă prin îmbunătăţirea gradului de grupare a formelor din aceeaşi clasă şi prin accentuarea trăsăturilor comune formelor din aceeaşi clasă.

Fie $\boldsymbol{X}$ un vector de formă oarecare:

$$\boldsymbol{X} = (x_1, x_2, ..., x_n) \tag{13}$$

Considerăm transformarea diagonală:

$$\boldsymbol{X} \to (\ \boldsymbol{diag W})\boldsymbol{X}, \text{ unde } \boldsymbol{diag W} = \begin{bmatrix} w_{11} & \dots & 0 \\ \dots & \dots & \dots \\ 0 & \dots & w_{nn} \end{bmatrix} \tag{14}$$

Ca măsură a varianţei vectorilor de forma din clasa ω_k se poate folosi suma pătratelor distanţelor între perechile posibile de vectori de forma din clasa ω_k, adică:

$$D_k^2 = \frac{1}{N_k(N_k-1)} \sum_{i=1}^{N_k} \sum_{j=1}^{N_k} \boldsymbol{d}^2(\boldsymbol{X}_i^{(k)}, \boldsymbol{X}_j^{(k)}) \tag{15}$$

Aceeaşi relaţie în spaţiul transformatei devine:

$$D_k^2 = \frac{1}{N_k(N_k-1)} \sum_{i=1}^{N_k} \sum_{j=1}^{N_k} \boldsymbol{d}^2((\ \boldsymbol{diag W})\boldsymbol{X}_i^{(k)}, (\ \boldsymbol{diag W})\boldsymbol{X}_j^{(k)}) \tag{16}$$

Dar:

$$\boldsymbol{d}^2(\boldsymbol{X}, \boldsymbol{Y}) = \sum_{c=1}^{n} (x_c - y_c)^2 \tag{17}$$

unde "*n*" este numărul de trăsături folosite.

Înlocuind, se obţine:

$$D_k^2 = \frac{1}{N_k(N_k-1)} \sum_{i=1}^{N_k} \sum_{j=1}^{N_k} \sum_{c=1}^{n} w_c^2 (x_{ic}^{(k)} - x_{jc}^{(k)})^2 \tag{18}$$

Transformarea diagonală optimă pentru clasa "*k*" se poate obţine din condiţia de minimizare a lui D_k^2 şi folosind o condiţie suplimentară de forma:

$$\sum_{c=1}^{n} w_c = 1 \qquad \text{sau} \qquad \prod_{c=1}^{n} w_c = 1 \tag{19}$$

condiţie necesară pentru a limita valorile ponderilor printr-o scalare aditivă sau multiplicativă.

Deci vom deduce valorile ponderilor w_c din condiţia:

$$\frac{\partial D_k^2}{\partial w_m} = 0 \tag{20}$$

plus una din condiţiile suplimentare precedente, folosind metoda multiplicatorilor lui Lagrange.

Astfel, pentru prima variantă (scalare aditivă a parametrilor), trebuie de minimizat funcţia:

$$\boldsymbol{f} = \frac{1}{N_k(N_k-1)} \sum_{i=1}^{N_k} \sum_{j=1}^{N_k} \sum_{c=1}^{n} w_c^2 (x_{ic}^{(k)} - x_{jc}^{(k)})^2 - \lambda(\sum_{c=1}^{n} w_c - 1) \tag{21}$$

Se obţine:

$$\frac{\partial \boldsymbol{f}}{\partial w_m} = 0 \Rightarrow \frac{1}{N_k(N_k-1)} \sum_{i=1}^{N_k} \sum_{j=1}^{N_k} 2w_m (x_{ic}^{(k)} - x_{jc}^{(k)})^2 - \lambda = 0 \tag{22}$$

$$2w_m \frac{1}{N_k(N_k-1)} \sum_{i=1}^{N_k} \sum_{j=1}^{N_k} (x_{ic}^{(k)} - x_{jc}^{(k)})^2 = \lambda \tag{23}$$

Remarcând faptul că:

$$\sigma_m^2 \overset{\Delta}{=} \frac{1}{N_k(N_k-1)} \sum_{i=1}^{N_k} \sum_{j=1}^{N_k} (x_{ic}^{(k)} - x_{jc}^{(k)})^2 \tag{24}$$

este tocmai varianţa trăsăturii "*m*" pentru clasa "*k*", rezultă:

$$2w_m\sigma_m^2 = \lambda \Rightarrow w_m = \frac{\lambda}{2\sigma_m^2} \tag{25}$$

şi folosind condiţia suplimentară de scalare aditivă, se obţine prin înlocuire:

$$1 = \frac{\lambda}{2}\sum_{m-1}^{n}\frac{1}{\sigma_m^2} \Rightarrow \lambda = \frac{2}{\sum_{m-1}^{n}\frac{1}{\sigma_m^2}} \tag{26}$$

Ca urmare rezultă în final:

$$w_c^{(k)} = \frac{1}{\sigma_c^2\sum_{m-1}^{n}\frac{1}{\sigma_m^2}} \tag{27}$$

Deoarece ponderile w_c sunt proporţionale cu $1/\sigma_c^2$, în urma acestei transformări, trăsăturile cu varianţă mare vor fi ponderate cu valori mici, deci importanţa lor va scade.

Dacă se utilizează condiţia suplimentară de scalare multiplicativă a coeficienţilor transformării diagonale, va trebui să minimizăm funcţia:

$$f = \frac{1}{N_k(N_k-1)}\sum_{i=1}^{N_k}\sum_{j=1}^{N_k}\sum_{c=1}^{n} w_c^2(x_{ic}^{(k)} - x_{jc}^{(k)})^2 - \lambda(\prod_{c=1}^{n} w_c - 1) \tag{28}$$

Efectuând acelaşi şir de operaţii se obţine succesiv:

$$\frac{\partial f}{\partial w_m} = 0 \Rightarrow \frac{1}{N_k(N_k-1)}\sum_{i=1}^{N_k}\sum_{j=1}^{N_k} 2w_m(x_{ic}^{(k)} - x_{jc}^{(k)})^2 - \frac{\lambda}{w_m} = 0 \tag{29}$$

$$\frac{2w_m^2}{N_k(N_k-1)}\sum_{i=1}^{N_k}\sum_{j=1}^{N_k}(x_{ic}^{(k)} - x_{jc}^{(k)})^2 = \lambda \Leftrightarrow 2w_m^2\sigma_m^2 = \lambda \Leftrightarrow w_m^2 = \frac{\lambda}{2\sigma_m^2} \tag{30}$$

Înlocuind în condiţia de scalare multiplicativă:

$$\frac{\lambda^n}{2^n}\frac{1}{\prod_{m=1}^{n}\sigma_m^2} = 1 \Leftrightarrow \lambda = 2(\prod_{m=1}^{n}\sigma_m^2)^{1/n} \tag{31}$$

Rezultă în final:

$$w_c^{(k)} = \frac{(\prod_{m=1}^{n}\sigma_m)^{1/n}}{\sigma_c} \tag{32}$$

Cu formulele obţinute se poate construi o transformare diagonală care să separe foarte bine o clasă de celelalte. Aceeaşi transformare permite, de asemenea, selectarea trăsăturilor semnificative prin reţinerea acelora pentru care ponderea $w_c^{(k)}$ are valoare mare.

Pentru a selecta trăsăturile semnificative pentru toate clasele, se pot utiliza funcţii de cost de forma $f(w_c^{(k)})$, dependente de ponderile transformării diagonale obţinute pentru fiecare clasă. Există "n" asemenea funcţii, corespunzătoare fiecărei trăsături. Se pot reţine acele trăsături pentru care "f" ia valoarea cea mai mare pentru toate clasele.

15.4. Selecţia optimală a trăsăturilor.

Selecţia optimală a trăsăturilor se realizează construind o transformare ortogonală (unitară) în următorul sistem de selecţie a trăsăturilor:

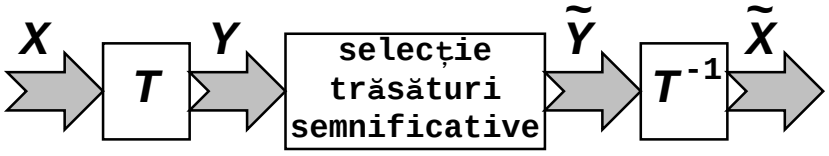


Fig. 1. Selecția optimală a trăsăturilor.

unde:

$\boldsymbol{X}$ - vectorul de formă de intrare, $\boldsymbol{X} = (x_1, x_2, ..., x_n)^T$

$\boldsymbol{Y}$ - vectorul de formă transformat, $\boldsymbol{Y} = (y_1, y_2, ..., y_n)^T$

$\tilde{\boldsymbol{Y}}$ - vectorul de formă redus, $\tilde{\boldsymbol{Y}} = (y_1, y_2, ..., y_k, c_{k+1}, ..., c_n)$

$\tilde{\boldsymbol{X}}$ - estimaţia vectorului de formă de intrare, $\tilde{\boldsymbol{X}} = (\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_n)^T$

Se remarcă modul de efectuare a reducerii datelor prin forţarea ultimelor $"n-k"$ valori din vectorul transformat la constante.

Se urmăreşte determinarea matricii transformării ortogonale unitare din condiţia ca eroarea medie pătratică de estimare a formei iniţiale să fie minimă. Se folosesc notaţiile:

$$\boldsymbol{T} = \begin{bmatrix} u_{11} & u_{12} & ... & u_{1n} \\ u_{21} & u_{22} & ... & u_{2n} \\ ... & ... & ... & ... \\ u_{n1} & u_{n2} & ... & u_{nn} \end{bmatrix} = \begin{bmatrix} \boldsymbol{u}_1^T \\ \boldsymbol{u}_2^T \\ ... \\ \boldsymbol{u}_n^T \end{bmatrix} \tag{33}$$

Deoarece transformarea $\boldsymbol{T}$ este unitară, avem:

$$\boldsymbol{T}^{-1} = \boldsymbol{T}^T \tag{34}$$

şi atunci:

$$\boldsymbol{T}^T = \begin{bmatrix} u_{11} & u_{21} & ... & u_{n1} \\ u_{12} & u_{22} & ... & u_{n2} \\ ... & ... & ... & ... \\ u_{1n} & u_{2n} & ... & u_{nn} \end{bmatrix} = \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & ... & \boldsymbol{u}_n \end{bmatrix} \tag{35}$$

Din structura schemei bloc rezultă:

$$\boldsymbol{Y} = \boldsymbol{TX} \Longrightarrow \boldsymbol{X} = \boldsymbol{T}^T\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & ... & \boldsymbol{u}_n \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ ... \\ y_n \end{bmatrix} \Longrightarrow \boldsymbol{X} = \sum_{j=1}^{n} \boldsymbol{u}_j y_j \tag{36}$$

Eroarea medie pătratică de estimare a lui $\boldsymbol{X}$ este dată de:

$$\varepsilon = \boldsymbol{E}\{\left\|\boldsymbol{X} - \tilde{\boldsymbol{X}}\right\|^2\} = \boldsymbol{E}\{(\boldsymbol{X} - \tilde{\boldsymbol{X}})^T(\boldsymbol{X} - \tilde{\boldsymbol{X}})\} \tag{37}$$

Dar, deoarece:

$$\boldsymbol{X} - \tilde{\boldsymbol{X}} = \sum_{j=1}^{n} \boldsymbol{u}_j y_j - \sum_{j=1}^{n} \boldsymbol{u}_j \tilde{y}_j = \sum_{j=k+1}^{n} \boldsymbol{u}_j (y_j - c_j) \tag{38}$$

Rezultă pentru eroarea medie patratică expresia:

$$\varepsilon = \boldsymbol{E}\left\{ \sum_{j=k+1}^{n} (y_j - c_j)\boldsymbol{u}_j^T \sum_{j=k+1}^{n} \boldsymbol{u}_j (y_j - c_j) \right\} = \boldsymbol{E}\left\{ \sum_{i=k+1}^{n} \sum_{j=k+1}^{n} (y_j - c_j)\boldsymbol{u}_j^T \boldsymbol{u}_i (y_i - c_i) \right\} \tag{39}$$

şi deoarece din condiţia de ortogonalitate avem:

$$\boldsymbol{u}_j^T \boldsymbol{u}_i = \begin{cases} 1, \text{ pentru } i = j \\ 0, \text{ pentru } i \neq j \end{cases} \tag{40}$$

se obţine în final:

$$\varepsilon = \boldsymbol{E}\left\{ \sum_{j=k+1}^{n} (y_j - c_j)^2 \right\} \tag{41}$$

Primul pas al optimizării constă în determinarea constantelor "*c*" din condiţia de minimizare a erorii medii pătratice:

$$\frac{\partial \varepsilon}{\partial c_j} = 0 \Leftrightarrow \boldsymbol{E}\{-2(y_j - c_j)\} = 0 \Leftrightarrow c_j = \boldsymbol{E}\{y_j\} = \bar{y}_j \tag{42}$$

Atunci eroarea medie pătratică de estimare a intrării devine:

$$\varepsilon = \boldsymbol{E}\left\{ \sum_{j=k+1}^{n} (y_j - \bar{y}_j)^2 \right\} = \boldsymbol{E}\left\{ \sum_{j=k+1}^{n} (y_j - \bar{y}_j)(y_j - \bar{y}_j)^T \right\} \tag{43}$$

Deoarece:

$$y_j = \boldsymbol{u}_j^T \boldsymbol{X} \qquad \text{şi} \qquad \bar{y}_j = \boldsymbol{u}_j^T \bar{\boldsymbol{X}} \tag{44}$$

rezultă după înlocuiri:

$$\varepsilon = \boldsymbol{E}\left\{ \sum_{j=k+1}^{n} \boldsymbol{u}_j^T (x_j - \bar{x}_j)(x_j - \bar{x}_j)^T \boldsymbol{u}_j \right\} = \sum_{j=k+1}^{n} \boldsymbol{u}_j^T \, \boldsymbol{E}\{(x_j - \bar{x}_j)(x_j - \bar{x}_j)^T\} \boldsymbol{u}_j \tag{45}$$

Deci:

$$\varepsilon = \sum_{j=k+1}^{n} \boldsymbol{u}_j^T \boldsymbol{K}_{XX} \boldsymbol{u}_j \tag{46}$$

unde $\boldsymbol{K}_{XX}$ este matricea de covarianţă asociată procesului aleator care descrie vectorii formelor de intrare.

Pentru a determina $\boldsymbol{u}_j$ din condiţia de minimizare a erorii medii pătratice de estimare, adică:

$$\nabla_{\boldsymbol{u}_j} \varepsilon = 0 \tag{47}$$

cu condiţia suplimentară $\boldsymbol{u}_j^T \boldsymbol{u}_j = 1$ care exprimă proprietatea de ortogonalitate a matricii $\boldsymbol{T}$, se foloseşte metoda multiplicatorilor lui Lagrange, adică se minimizează funcţia:

$$\nabla_{\boldsymbol{u}_j} \boldsymbol{f} = 0 \tag{48}$$

unde:

$$f = \sum_{j=k+1}^{n} \boldsymbol{u}_j^T \boldsymbol{K}_{XX} \boldsymbol{u}_j - \lambda_j (\boldsymbol{u}_j^T \boldsymbol{u}_j - 1) \tag{49}$$

Rezultă în final:

$$(\boldsymbol{K}_{XX} - \lambda_j \boldsymbol{I}_n)\boldsymbol{u}_j = 0 \tag{50}$$

care exprima faptul că $\lambda_j, j = 1..n$ sunt valorile proprii ale matricii de covarianţă a intrării, iar $\boldsymbol{u}_j, j = 1..n$ sunt vectorii proprii ai aceleiaşi matrici $\boldsymbol{K}_{XX}$.

Deci matricea $\boldsymbol{T}$ a transformării ortogonale căutate are drept linii, vectorii proprii ai matricii de covarianţă a domeniului de intrare.

Mărimea erorii medii pătratice de estimare a intrării este dată atunci de:

$$\varepsilon = \sum_{j=k+1}^{n} u_j^T \boldsymbol{K}_{XX} u_j = \sum_{j=k+1}^{n} u_j^T \lambda_j u_j = \sum_{j=k+1}^{n} \lambda_j u_j^T u_j \tag{51}$$

şi folosind condiţia de ortogonalitate obţinem:

$$\varepsilon = \sum_{j=k+1}^{n} \lambda_j \tag{52}$$

Această condiţie permite alcătuirea următorului algoritm de selecţie optimală a trăsăturilor:

(1). Se calculează sau se estimează matricea de covarianţă a formelor de intrare $\boldsymbol{K}_{XX}$.
(2). Se calculează valorile proprii ale matricii $\boldsymbol{K}_{XX}$.
(3). Se ordonează descrescător aceste valori proprii.
(4). Se construieşte matricea $\boldsymbol{T}$ a transformării unitare căutate din vectorii proprii ai matricii $\boldsymbol{K}_{XX}$, conform ordinii găsite pentru valorile proprii.
(5). Numărul "k" de trăsături care se reţin după transformarea unitară se obţine din valoarea maximă admisibilă a erorii medii pătratice de estimare.

Matricea de covarianţă a domeniului transformatei este dată de:

$$\boldsymbol{K}_{YY} = \boldsymbol{E}\{(\boldsymbol{Y}-\overline{\boldsymbol{Y}})(\boldsymbol{Y}-\overline{\boldsymbol{Y}})^T\} = \boldsymbol{E}\{\boldsymbol{T}(\boldsymbol{X}-\overline{\boldsymbol{X}})(\boldsymbol{X}-\overline{\boldsymbol{X}})^T\boldsymbol{T}^T\} = \boldsymbol{T}\boldsymbol{K}_{XX}\boldsymbol{T}^T \tag{53}$$

şi deoarece:

$$\boldsymbol{K}_{XX}\boldsymbol{u}_j = \lambda_j \boldsymbol{u}_j, \text{ pentru } j = 1...n \tag{54}$$

rezultă:

$$\boldsymbol{K}_{XX}[\boldsymbol{u}_1 \quad \boldsymbol{u}_2 \quad ... \quad \boldsymbol{u}_n] = \boldsymbol{diag}(\lambda_1 \quad \lambda_2 \quad ... \quad \lambda_n)[\boldsymbol{u}_1 \quad \boldsymbol{u}_2 \quad ... \quad \boldsymbol{u}_n] \tag{55}$$

$$\boldsymbol{K}_{XX}\boldsymbol{T}^T = \boldsymbol{\Lambda}\boldsymbol{T}^T$$

Atunci matricea de covarianţă a domeniului transformatei se scrie:

$$\boldsymbol{K}_{YY} = \boldsymbol{T}\boldsymbol{\Lambda}\boldsymbol{\Lambda}^T = \boldsymbol{\Lambda}\boldsymbol{T}\boldsymbol{T}^T = \boldsymbol{\Lambda} \tag{56}$$

Deci prin aplicarea transformării optimale se obţin trăsături necorelate.

15.5. Transformări rotaţionale pentru selecţia trăsăturilor.

Considerând formele din spaţiul formelor $\boldsymbol{\Omega} \subset R^n$ dezvoltate folosind o bază ortonormată completă de vectori:

$$\{\tilde{\boldsymbol{b}}\} = \{\boldsymbol{b}_p, p = 1...m\} \tag{57}$$

unde $m < n$, atunci orice formă poate fi exprimată sub forma unei combinaţii liniare a vectorilor din această bază:

$$\boldsymbol{X}_j^{(k)} = a_{j1}\boldsymbol{b}_1 + a_{j2}\boldsymbol{b}_2 + ... + a_{jm}\boldsymbol{b}_m \tag{58}$$

Eroarea indusă de această dezvoltare are expresia:

$$e_j^{(k)} = \boldsymbol{E}\{\,|\,\boldsymbol{X}_j^{(k)} - \sum_{p=1}^{m} a_{jp}\boldsymbol{b}_p\,|^2\,\} \tag{59}$$

Dacă se urmăreşte în continuare determinarea bazei ortonormate $\{\tilde{\boldsymbol{b}}\}$ în condiţiile minimizării erorii medii pătratice anterioare, se obţine, aşa cum s-a arătat în paragraful precedent, transformarea Karhunen-Loeve.

Volumul de calcule necesar implementării acestei transformări este foarte mare şi de aceea, pentru anumite aplicaţii care permit modelarea formelor de intrare cu un proces Markov de ordinul unu, se foloseşte Transformata Cosinus Discretă (DCT) care aproximează cel mai bine transformarea optimală descrisă anterior, mai ales pentru valori mici ale numărului de trăsături $n \le 64$. În plus, algoritmul transformarii DCT rapide (derivat din FFT) permite reducerea substanţială a numărului de calcule, de la n^2 la $2n\log_2 n$ operaţii (o operaţie = o înmulţire şi o adunare).

Proprietăţi de decorelare a vectorilor aplicaţi la intrare prezintă însă şi alte transformări, cum ar fi:

(1). Transformata Fourier, pentru calculul căreia există algoritmul FFT (Fast Fourier Transform) care permite micşorarea substanţială a numărului de calcule, de la n^2 la $n \log_2 n$ operaţii.

(2). Transformata Walsh-Hadamard care prezintă avantajul că pentru calculul ei sunt necesare doar $n \log_2 n$ adunări/scăderi şi nici o înmulţire.

(3). Transformata Haar, care furnizează la ieşire o combinaţie liniară a tuturor valorilor din vectorul de intrare doar prin intermediul primilor doi coeficienţi, în timp ce restul coeficienţilor conţin câte o combinaţie liniară doar a valorilor de intrare adiacente. Pentru calculul ei sunt necesare doar $2(n-1)$ adunări/scăderi.

15.6. Măsuri ale calităţii trăsăturilor.

Pe lângă parametrii descrişi în primul paragraf al acestui capitol pentru caracterizarea calităţii trăsăturilor, adică varianţa trăsăturilor, variabilitatea intraclasă, variabilitatea interclasă, (ponderea) Fisher şi raportul Coomans, există un set întreg de măsuri de separabilitate a claselor, cum ar fi:

(1) Divergenţa, care trebuie maximizată pentru o bună separabilitate a claselor:

$$\boldsymbol{Q}_1 = \int_{\Omega} [\, \boldsymbol{p}(\omega_i)\, \boldsymbol{p}(\boldsymbol{X}/\omega_i) - \boldsymbol{p}(\omega_j)\, \boldsymbol{p}(\boldsymbol{X}/\omega_j)] \log \left[\frac{\boldsymbol{p}(\omega_i)\, \boldsymbol{p}(\boldsymbol{X}/\omega_i)}{\boldsymbol{p}(\omega_j)\, \boldsymbol{p}(\boldsymbol{X}/\omega_j)} \right] d\boldsymbol{X} \tag{60}$$

(2) Distanţa Bhattacharyya (B-distanţa), al cărui minim asigură o bună separabilitate a claselor:

$$\boldsymbol{Q}_2 = -\log \int_{\Omega} [\, \boldsymbol{p}(\boldsymbol{X}/\omega_i)\, \boldsymbol{p}(\boldsymbol{X}/\omega_j)]^{1/2} d\boldsymbol{X} \tag{61}$$

(3). Patrick şi Fisher sugerează folosirea coeficientului:

$$\boldsymbol{Q}_3^2 = -\log \int_{\Omega} [\, \boldsymbol{p}(\boldsymbol{X}/\omega_i)\, \boldsymbol{p}(\boldsymbol{X}/\omega_j)]^2 d\boldsymbol{X} \tag{62}$$

(4). Meisel propune varianta discretă următoare, care este utilă mai ales în cazul formelor cu număr mare de trăsături:

$$\boldsymbol{Q}_4^2 = \frac{p_i}{M_i} \sum_{k=1}^{M_1} [\, \boldsymbol{p}(\omega_i)\, \boldsymbol{p}(\boldsymbol{X}/\omega_i) - \boldsymbol{p}(\omega_j)\, \boldsymbol{p}(\boldsymbol{X}/\omega_j)]^2 + \frac{p_j}{M_j} \sum_{k=1}^{M_2} [\, \boldsymbol{p}(\omega_i)\, \boldsymbol{p}(\boldsymbol{X}/\omega_i) - \boldsymbol{p}(\omega_j)\, \boldsymbol{p}(\boldsymbol{X}/\omega_j)]^2 \tag{63}$$

(5). Măsura următoare ţine cont de eroarea de clasificare pentru un clasificator Bayes:

$$\boldsymbol{Q}_5^2 = \frac{p_i}{M_i} \sum_{k=1}^{M_1} \boldsymbol{\mu}[\, \boldsymbol{p}(\omega_i)\, \boldsymbol{p}(\boldsymbol{X}/\omega_i) - \boldsymbol{p}(\omega_j)\, \boldsymbol{p}(\boldsymbol{X}/\omega_j)]^2 + \frac{p_j}{M_j} \sum_{k=1}^{M_2} \boldsymbol{\mu}[\, \boldsymbol{p}(\omega_i)\, \boldsymbol{p}(\boldsymbol{X}/\omega_i) - \boldsymbol{p}(\omega_j)\, \boldsymbol{p}(\boldsymbol{X}/\omega_j)]^2 \tag{64}$$

unde:

$$\boldsymbol{\mu}(x) = \begin{cases} 1, \text{pentru } x \geq 0 \\ 0, \text{pentru } x < 0 \end{cases}$$

Dacă măsurile precedente se referă la separabilitatea între clasele ω_i şi ω_j, următoarele măsuri urmăresc separabilitatea globală, deci pentru toate cele N_c clase.

(6). O variantă continuă pentru estimarea separabilităţii globale a claselor este:

$$\boldsymbol{Q}_6 = \int_{\Omega} \max_{i=1\ldots N_c} \boldsymbol{p}(\omega_i)\, \boldsymbol{p}(\boldsymbol{X}/\omega_i) d\boldsymbol{X} \tag{65}$$

(7). Altă modalitate de apreciere are la bază riscul condiţionat de clasificare eronată:

$$\boldsymbol{Q}_7 = \int_{\Omega}\left[\sum_{i=1}^{N_c} \boldsymbol{p}(\omega_i)\,\boldsymbol{p}(\boldsymbol{X}/\omega_i)\right]^2 d\boldsymbol{X} - \int_{\Omega}\sum_{i=1}^{N_c}[\,\boldsymbol{p}(\omega_i)\,\boldsymbol{p}(\boldsymbol{X}/\omega_i)]^2\, d\boldsymbol{X} \tag{66}$$

(8). Varianta discretă a coeficientului precedent este dată de:

$$\boldsymbol{Q}_8 = \frac{1}{M}\sum_{j=1}^{M}\left\{\left[\sum_{i=1}^{N_c} \boldsymbol{p}(\omega_i)\,\boldsymbol{p}(\boldsymbol{X}/\omega_i)\right]^2 - \sum_{i=1}^{N_c} \boldsymbol{p}(\omega_i)\,\boldsymbol{p}(\,\boldsymbol{X}/\omega_i)\right\} \tag{67}$$

Alegerea criteriului optim de selecţie a acelor trăsături care asigură cea mai bună separabilitate a claselor este încă o problemă deschisă, ea depinzând de legea de distribuţie care guvernează formele de intrare (pentru distribuţia gaussiană se recomandă $\boldsymbol{Q}_1$ şi $\boldsymbol{Q}_2$) şi de volumul de calcule necesar pentru determinarea lor.

15.7. Detalii de implementare.

Concret se urmăreşte proiectarea unui sistem bazat pe circuitul ***INMOS A100*** - filtru digital transversal cu *32* de nivele - care să rezolve problema calculului Transformatei Cosinus Discretă (DCT) şi a ieşirilor unei reţele neuronale de tipul perceptron multistrat, ambele fiind mari consumatoare de timp şi ambele având la bază calculul unor sume ponderate.

$$\boldsymbol{DCT}(u,v) = \frac{1}{2N}\sum_{j=0}^{N-1}\sum_{i=0}^{N-1} \boldsymbol{f}(j,k)\ \boldsymbol{cos}\left(\frac{\pi\, u\, j}{N}\right)\ \boldsymbol{cos}\left(\frac{\pi\, v\, k}{N}\right) \tag{68}$$

iar pentru reţea neuronală:

$$z_{pj} = \boldsymbol{f}\left(\sum_{k=1}^{N} w_{kj}^{(z)} y_{pk}\right) \quad y_{pj} = \boldsymbol{f}\left(\sum_{k=1}^{M} w_{kj}^{(y)} x_{pk}\right) \quad x_{pj} = \boldsymbol{f}\left(\sum_{k=1}^{L} w_{kj}^{(x)} u_{pk}\right) \tag{69}$$

IMS A100 este un filtru transversal cu 32 de etaje, de precizie şi viteză înaltă. Arhitectura sa flexibilă face posibilă utilizarea acestui circuit ca un element constitutiv într-o gamă largă de aplicaţii pentru prelucrare de semnal (DSP = 'digital signal processing'). Cuvântul de date la intrare este în lungime de *16* biţi, iar mărimea coeficientului este programabilă la *4*, *8*, *12*, *16* biţi. Pentru ambii se foloseşte formatul în complement faţă de doi. Coeficienţii pot fi reactualizaţi asincron faţă de ceasul sistemului, permiţând ca circuitul să fie folosit în sisteme adaptive. ***IMS A100*** poate fi cascadat pentru a se construi filtre transversale de lungime mai mare, fără a folosi circuite suplimentare. Şi în acest caz, se păstrează precizia de lucru şi domeniul dinamic.

Circuitul ***IMS A100*** implementează o structură de filtru transversal modificată (Fig. 2).

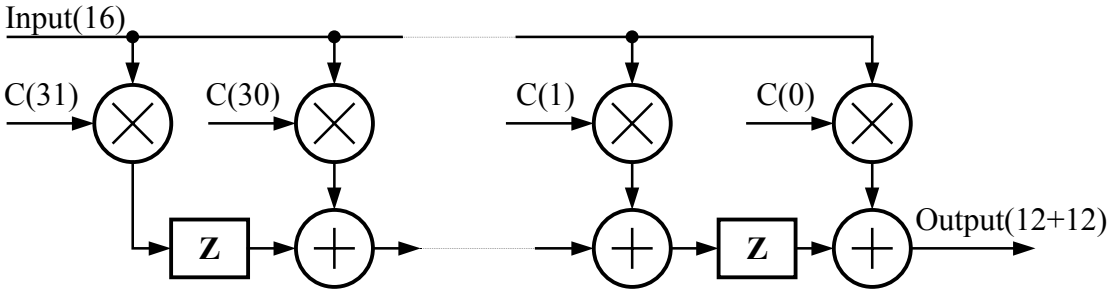


Fig. 2. Schema bloc a circuitului IMS A100.

Ieşirea se calculează cu relaţia:

$$y(kT)=C(0)*x(kT)+C(1)*x((k-1)T)+...+C(N-1)*x((k-N+1)T) \tag{70}$$

unde $x(kT)$ reprezintă eşantionul de ordinul k, iar $C(0),..,C(N-1)$ sunt coeficienţii pentru cele N etaje.

IMS A100 este prevăzut cu două tipuri de interfeţe: una asincronă ce se conectează la microprocesorul "gazdă", (pentru furnizarea datelor "brute" şi pentru preluarea rezultatelor) şi o

interfață sincronă (pentru cascadarea circuitelor şi pentru realizarea de structuri de filtrare "on-line", terminate cu convertoare analog/digitale şi digital/analogice). Rezultatul se calculează pe *36* de biți. Sub controlul programului, în funcție de aplicație, se poate alege din aceştia orice câmp de *24* de biți. Puterea de calcul, pentru lungimea coeficientului *'Lc'* şi frecvența de ceas a circuitului *'F'*, rezultă din formula *2*F/Lc*.

Deoarece scopul principal al demersului nostru este de a calcula rapid sume ponderate de forma:

$$Y = \sum_{i=0}^{N+1} w_i x_i \tag{71}$$

iar coeficienții (ponderile) w_i sunt deja cunoscute, ele pot fi încărcate într-o memorie de tip ***PROM/EPROM*** de capacitate convenabil aleasă.

Astfel deoarece $N = 32$ pentru calculul Transformatei Cosinus Discrete (DCT), sunt necesari *1024* de coeficienți a *16* biți fiecare pentru aplicații de tipul celor descrise anterior (DCT bidimensională pentru o matrice *32x32*).

Proiectarea s-a axat pe o variantă constructivă a cărei schemă bloc este prezentată în continuare şi a cărei utilizare poate fi extinsă la o gamă întreagă de aplicații de aceeaşi natură.

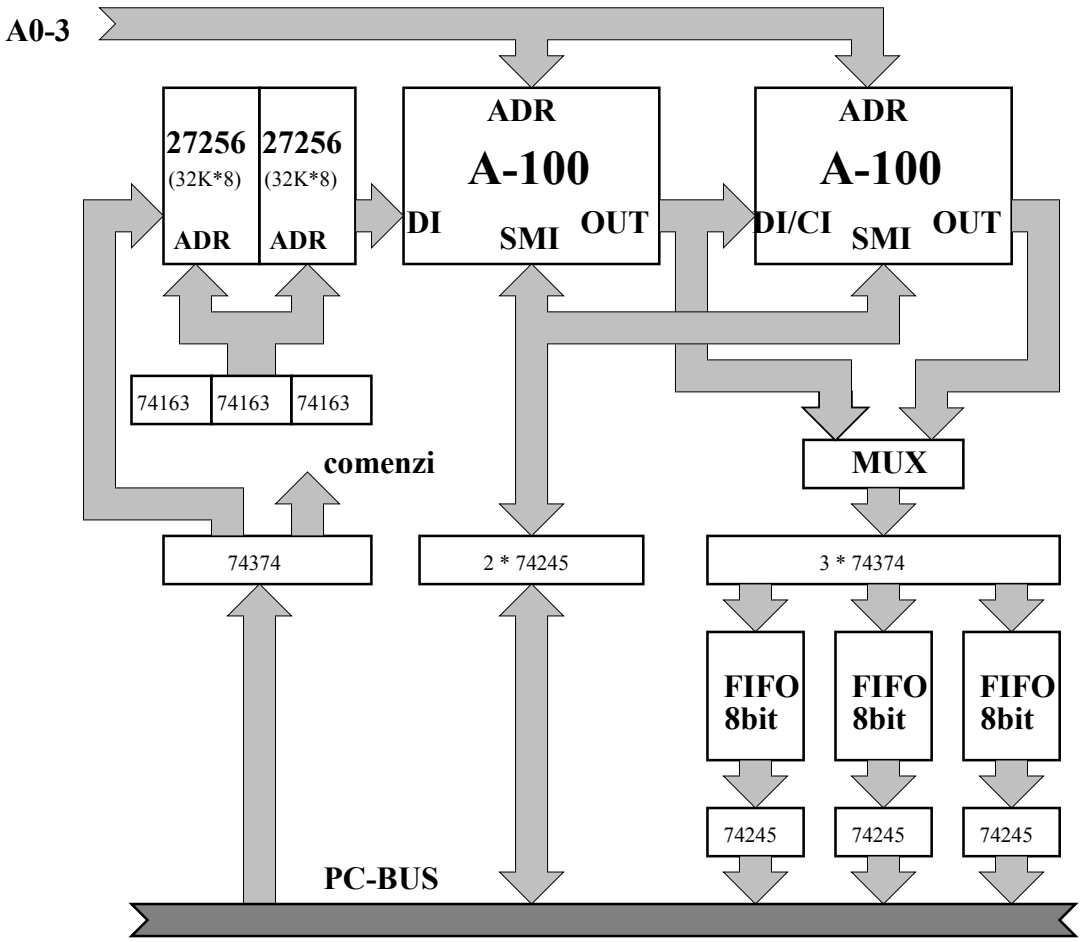


Fig. 3. Schema bloc a acceleratorului.

Ţinând cont de complexitatea aplicațiilor studiate până acum, s-a optat pentru o rețea neuronală având maximum *64* intrări, *64* neuroni pe primul strat, maximum *32* neuroni pe stratul al doilea şi maximum *32* neuroni pe stratul de ieşire (ultimul strat). Pentru varianta maximală de rețea neuronală, numărul de ponderi ce trebuiesc stocate este:

$$64*64 + 64*32 + 32*32 = 4Kword + 2Kword + 1Kword = 7Kword \tag{72}$$

Se poate opta şi pentru o variantă mai complexă, care necesită însă încă un circuit ***IMS A100***, variantă care ar permite o rețea neuronală având maximum *96* intrări, *96* neuroni pe primul strat, maximum *64* neuroni pe stratul al doilea şi maximum *32* neuroni pe stratul de ieşire (ultimul strat), capacitatea de stocare necesară fiind:

$$96*96 + 96*64 + 64*32 = 9Kword + 6Kword + 2Kword = 17Kword \quad (73)$$

În toate cazurile capacitatea de stocare nu depăşeşte *32* Kword şi de aceea se vor utiliza pentru stocarea acestor coeficienți două circuite ***EPROM 32KO***. Adresele lor sunt furnizate în parte de procesorul central (pentru selecția bancului de coeficienți/ponderi) iar restul de un numărător pe *12/14* biți.

Deoarece în unele cazuri este necesară calcularea sumei ponderate a *32* termeni (DCT, stratul de ieşire al rețelei) iar în altele apare o sumă de *64* termeni, ieşirile celor două circuite *IMS A100* sunt multiplexate (pentru selecția ieşirii dorite) şi apoi despachetate într-un buffer.

Circuitele FIFO permit stocarea temporară a rezultatelor (furnizate la rată foarte mare de ***IMS A100***), urmând ca apoi, printr-un transfer DMA ele să fie mutate în memoria calculatorului gazdă.

O problemă delicată este calculul funcțiilor sigmoidale caracteristice neuronilor. Ele ar putea fi implementate tabelar în memorii ***PROM/EPROM*** având capacitatea de *128KO (64Kword)* intercalate în lanțul de ieşire după multiplexor şi buffere, dar s-a renunțat în final la ele pentru a nu complica excesiv schema, dar mai ales pentru că soluția software echivalentă este suficient de rapidă.

Studiile şi simulările efectuate relevă o remarcabilă mărire a vitezei de calcul, în timp ce precizia mai scăzută a calculelor nu afectează semnificativ performanțele globale ale sistemului, iar prețul de cost al circuitelor folosite rămâne destul de modest. Gama de aplicații a sistemului poate fi extinsă şi în multe alte domenii (ex. filtre digitale).

16. Algoritmi de gestiune a seturilor de forme de antrenament.

Structura setului de antrenare influențează direct performanțele unui clasificator antrenat cu respectivul set de forme.

Cu cât setul de antrenament este mai mare, cu atât limitele de decizie sunt mai precis pozitionate şi eroarea de clasificare este mai mică. În schimb, timpul de antrenare este oarecum proporțional cu numărul de forme din setul de antrenament şi de aceea este mai avantajos din punctul de vedere al timpului consumat un set de antrenament de dimensiuni mici.

Există multe aplicații la care performanțele clasificatorului se pot ameliora în timp prin luarea în considerare a noi forme nerecunoscute sau recunoscute incorect, dar a căror apartenență corectă la clase este clară. Aceasta se întâlneşte, de exemplu, în situația când, după o primă antrenare a clasificatorului, verificarea comportării lui pe setul de forme de test furnizează un număr de simboluri nerecunoscute; la acestea se pot adăuga simbolurile nerecunoscute rezultate din exploatarea sistemului.

Practic, s-ar putea adăuga aceste noi forme la setul de antrenament şi apoi se reia antrenarea clasificatorului, pentru includerea lor în clasele corespunzatoare. În schimb, în acest fel dimensiunea setului de forme de antrenament poate creşte nelimitat, depăşind capacitățile de stocare (memorie, disc) curent folosite.

Pentru a evita această situație şi a menține limitată dimensiunea setului de antrenament, ideea studiată şi apoi folosită este de a înlocui anumite forme din setul de antrenament, forme a căror contribuție la construcția efectivă a clasificatorului este nesemnificativă sau mai puțin importantă cu forme noi, nerecunoscute, şi a căror includere în setul de antrenare poate ameliora performanțele clasificatorului.

Problema care apare este care anume forme din setul de antrenament să se elimine în scopul înlocuirii lor cu noile forme.

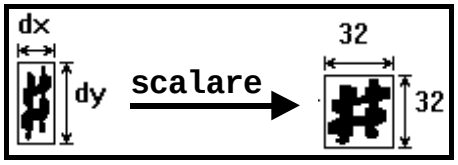
16.1. Calculul scorurilor de concidență.

Prima operație care se impune este construirea ***matricii scorurilor de coincidența (similitudine).*** Pot exista mai multe soluții, funcție de modul de extragere şi selecție a trăsăturilor. Deoarece în aplicațiile descrise în următoarele trei capitole se foloseşte o metodă specifică de

extragere a trăsăturilor, ea va fi descrisă sumar în continuare, urmând ca pe baza lor să se discute modalitățile de construcție a matricii scorurilor de coincidență.

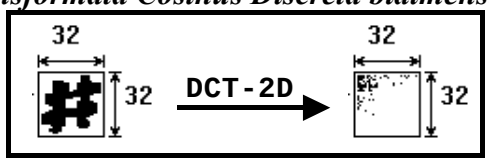
Pentru cazul recunoaşterii unor simboluri tipărite (caractere, simboluri muzicale), faza de extragere a trăsăturilor presupune parcurgerea următorilor paşi:

a) Scalarea simbolurilor:



Se obține o matrice binară de dimensiune constantă (32x32), conținând imaginea scalată.

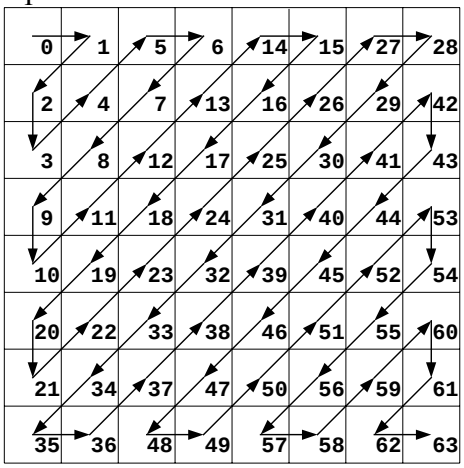
b) Transformata Cosinus Discretă bidimensională:

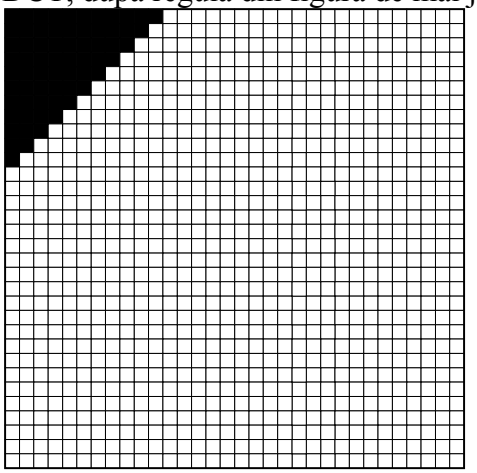


Această transformare se foloseşte pentru decorelarea informației existente în matricea scalată, după transformarea DCT bidimensională rezultând o matrice de aceeaşi dimensiune, cu valori reale, slab corelate. Valorile semnificative ale acestei matrici sunt plasate în colțul din stânga sus.

c) Ordonarea Zig - Zag.

Este o procedură de vectorizare a matricii DCT, după regula din figura de mai jos:





Matrice 8x8 ===> Vector[64]; Matrice 32x32 ===> Vector[1024];

Deoarece DCT bidimensională concentrează informația din imaginea inițială în colțul din stînga sus al matricii rezultate, după ordonarea Zig - Zag aceeaşi informație va fi regăsită în primele elemente ale vectorului obținut.

d) Selecția trăsăturilor.

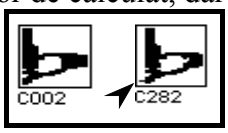
Se reține primele valori din vectorul obținut după pasul precedent, aproximativ cele marcate cu negru în figura de mai sus. Conform afirmațiilor anterioare, aceste elemente concentrează cea mai mare parte din informația din imaginea inițială şi ele vor fi folosite în faza de recunoaştere.

Dacă N este mărimea setului de antrenament, scorurile de coincidență S_{KL} reprezintă o matrice NxN simetrică şi cu elementele de pe diagonală nule, deci calculul ei înseamnă calculul a $N(N/2-1)$ valori semnificative. Pentru $N = 120$ rezultă 7080 valori. Ele se calculează practic o singură dată şi doar se actualizează pe parcursul rulării algoritmului (în pasul 5).

Pentru estimarea scorurilor de coincidență au fost testate mai multe măsuri:

$$\boldsymbol{S}^{(1)}{}_{KL} = \sum_{I=1}^{32}\sum_{J=1}^{32}\left|\boldsymbol{MS}^{(K)}[I,J] - \boldsymbol{MS}^{(L)}[I,J]\right| \tag{1}$$

care reprezintă distanța Hamming între obiectele "K" şi "L" (imagini binare scalate - *32x32*). Este foarte uşor de calculat, dar prezintă un dezavantaj care rezultă din următoarea figură:



Deşi obiectele sunt foarte asemănătoare, un mic zgomot poate afecta mult încadrarea în fereastră, deci şi valoarea calculată pentru scorul de coincidență. O expresie mai corectă este:

$$\boldsymbol{S}^{(2)} = \min_{M,N}\left(\sum_{I=1}^{32}\sum_{J=1}^{32}(\boldsymbol{MS}^{(K)}[I,J] - \boldsymbol{ME}^{(L)}[I+M,J+N])^2\right) \tag{2}$$

care calculează cel mai bun scor de coincidență între simbolul *(K)* şi simbolul *(L)* translat pe orizontală şi verticală. Deoarece prelucrările ulterioare asupra imaginii (DCT) nu sunt invariante la translații, (2) ar putea fi folosită în cazul utilizării unui neocognitron în faza de recunoaştere.

$$\boldsymbol{S}^{(3)}{}_{KL} = \sum_{I=1}^{32}\sum_{J=1}^{32}(\boldsymbol{DCT}^{(K)}[I,J] - \boldsymbol{DCT}^{(L)}[I,J])^2 \tag{3}$$

reprezintă distanța euclidiană între obiectele "K" şi "L" descrise în spațiul transformatei DCT. Este mai greu de calculat, deoarece setul de antrenament este memorat pe calculator sub forma imaginilor binare scalate.

$$\boldsymbol{S}^{(4)}{}_{KL} = \sum_{I=1}^{66}(\boldsymbol{Z}^{(K)}[I] - \boldsymbol{Z}^{(L))}[I])^2 \tag{4}$$

reprezintă distanța euclidiană între vectorii *"K"* şi *"L"* rezultați în urma ordonării Zig-Zag. Implică de asemenea un volum mare de calcule, dar se referă exact la valorile care se aplică intrării în clasificator. Pe de altă parte pare mai corectă o ponderare a valorilor $\boldsymbol{Z}^{[K]}[I]$, funcție de contribuția lor la "forma" simbolului, dar nu se pot face decât presupuneri privind valorile acestor ponderi.

16.2. Strategii de adaptare a setului de forme de antrenament

Pentru eliminarea din setul de antrenament a unor forme mai puțin semnificative în scopul înlocuirii lor, pot fi adoptate diferite strategii:

(1). Pentru fiecare clasă ω_i se caută perechea de forme $(\boldsymbol{X}_K^{(i)},\boldsymbol{X}_L^{(i)})$ pentru care scorul de coincidență $\boldsymbol{S}_{KL}^{(i)}$ este minim. Regula de eliminare a simbolurilor este:

$$\begin{aligned} &IF\ \boldsymbol{S}_{KM}^{(i)} < \boldsymbol{S}_{LM}^{(i)},\ \forall M \neq K,\ M \neq L,\ THEN \\ &\qquad \text{elimină forma } \boldsymbol{X}_K^{(i)} \\ &ELSE \\ &\qquad \text{elimină forma } \boldsymbol{X}_L^{(i)} \end{aligned} \tag{5}$$

În acest caz rezultă o distribuție cvasiuniformă a formelor din fiecare clasă. Sunt eliminate succesiv formele mai apropiate intre ele.

(2). Pentru fiecare clasă ω_i se caută perechea de forme $(\boldsymbol{X}_K^{(i)},\boldsymbol{X}_L^{(i)})$ pentru care scorul de coincidență $\boldsymbol{S}_{KL}^{(i)}$ este minim. Se calculează apoi distanța $\boldsymbol{d}(\boldsymbol{X}_K^{(i)},\boldsymbol{X}_M^{(j)}),\forall j \neq i$ de la fiecare din formele $\boldsymbol{X}_K^{(i)}$ şi $\boldsymbol{X}_L^{(i)}$ la fiecare din formele aparținând celorlalte clase. Regula de eliminare a simbolurilor este:

$$IF\ \min_{X_M \notin \omega_i} \boldsymbol{d}(\boldsymbol{X}_K^{(i)},\boldsymbol{X}_M) < \min_{X_M \notin \omega_i} \boldsymbol{d}(\boldsymbol{X}_L^{(i)},\boldsymbol{X}_M),\ THEN$$

elimină forma $\boldsymbol{X}_K^{(i)}$ (6)

ELSE

elimină forma $\boldsymbol{X}_L^{(i)}$

adică se păstrează acele forme aflate mai aproape de vecinii din alte clase.

Formele din setul de antrenament, fără a fi eliminate din zona centrală a clasei curente, capătă o densitate locală mai mare în apropierea limitelor de decizie.

(3). Pentru fiecare clasă ω_i şi fiecare formă $\boldsymbol{X}_K^{(i)}$ se calculează distanța până la vecinul cel mai apropiat aparținând altei clase, adică cel pentru care avem:

$$\min_{X_M \notin \omega_i} \boldsymbol{d}(\boldsymbol{X}_K^{(i)}, \boldsymbol{X}_M) \tag{7}$$

Dintre toate formele clasei ω_i se elimină acea formă pentru care această distanță este maximă:

$$\max_K \min_{X_M \notin \omega_i} \boldsymbol{d}(\boldsymbol{X}_K^{(i)}, \boldsymbol{X}_M) \tag{8}$$

adică se elimină forma cea mai depărtată de vecinii cei mai apropiați din alte clase.

Ca urmare dispar din setul de antrenament acele forme dispuse către zona centrală a clasei curente, rămânând în setul de antrenament doar acele forme apropiate de limita de decizie între clase.

16.3. Descrierea algoritmilor.

Algoritmul de actualizare a setului de antrenament utilizat începe după o primă antrenare a clasificatorului şi include următorii paşi:

Algoritmul de optimizare a setului de antrenament

1) Marcarea celor "n" simboluri nerecunoscute din setul de test;
2) Calculul scorurilor de coincidență $\boldsymbol{S}_{KL}$ pentru perechile *(KL)* de simboluri din setul de antrenament (metoda 1) sau a distanțelor $\boldsymbol{d}(\boldsymbol{X}_K^{(i)}, \boldsymbol{X}_M)$ menționate mai sus (metodele 2,3).
3) Elimină din setul de antrenare "*m*" simboluri, maximum câte un simbol din fiecare clasă, cele corespunzătoare regulilor de eliminare aferente strategiei alese;
4) Înlocuieşte simbolurile eliminate cu "*m*" din simbolurile nerecunoscute; setul de antrenament include acum o parte din simbolurile nerecunoscute detectate;
5) Actualizează scorurile de coincidență $\boldsymbol{S}_{KL}$ corespunzătoare celor "*n*" simboluri nou introduse în setul de antrenament.
6) Antrenează clasificatorul (ex. rețea neurală), plecând eventual de la ponderile deja calculate.
7) Testare clasificator: dacă numărul "*n*" de erori detectate devine acceptabil de mic, algoritmul se termină; în caz contrar salt la pasul 3).

Dacă, folosind o rețea neurală, după un număr predeterminat de paşi numărul de erori de recunoaştere detectate nu devine suficient de mic, rezultă că rețeaua neurală folosită nu poate "învăța" un număr atât de mare de simboluri, deci ar trebui adăugați neuroni pe primele straturi ale rețelei.

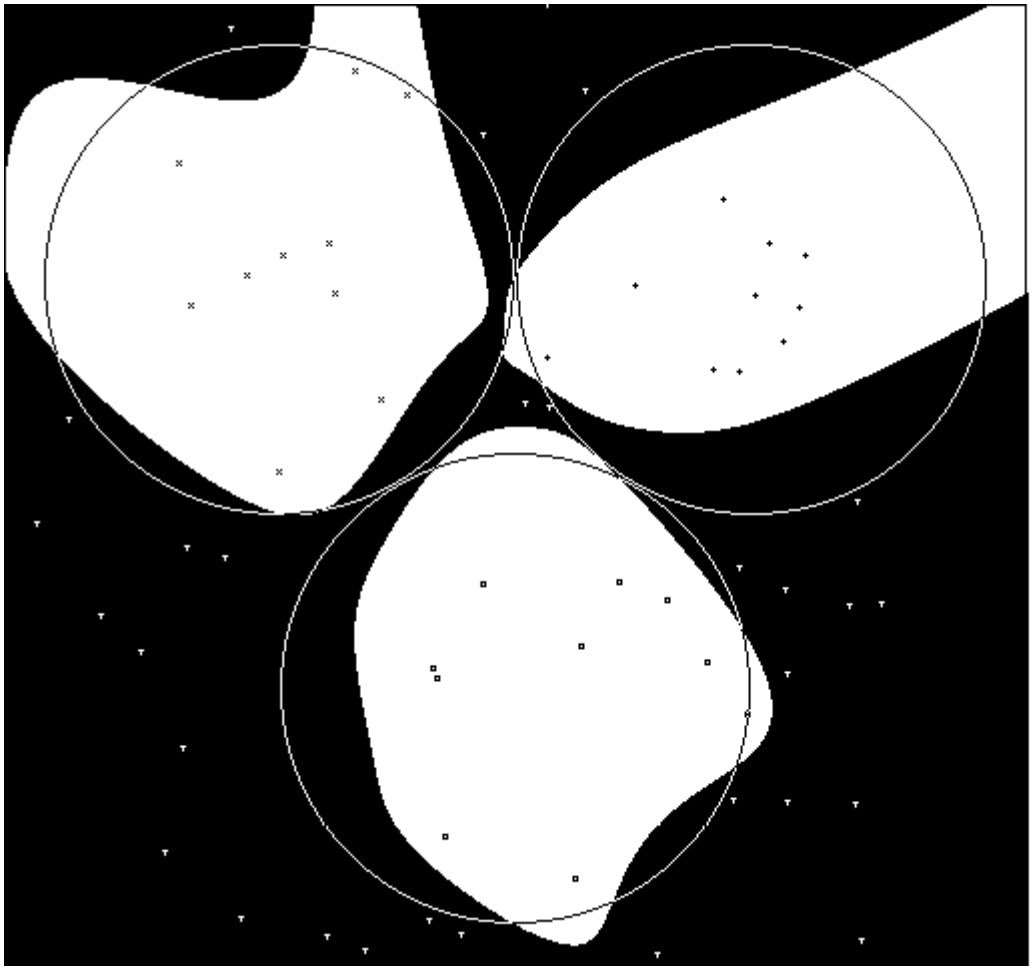
16.4. Detalii de implementare.

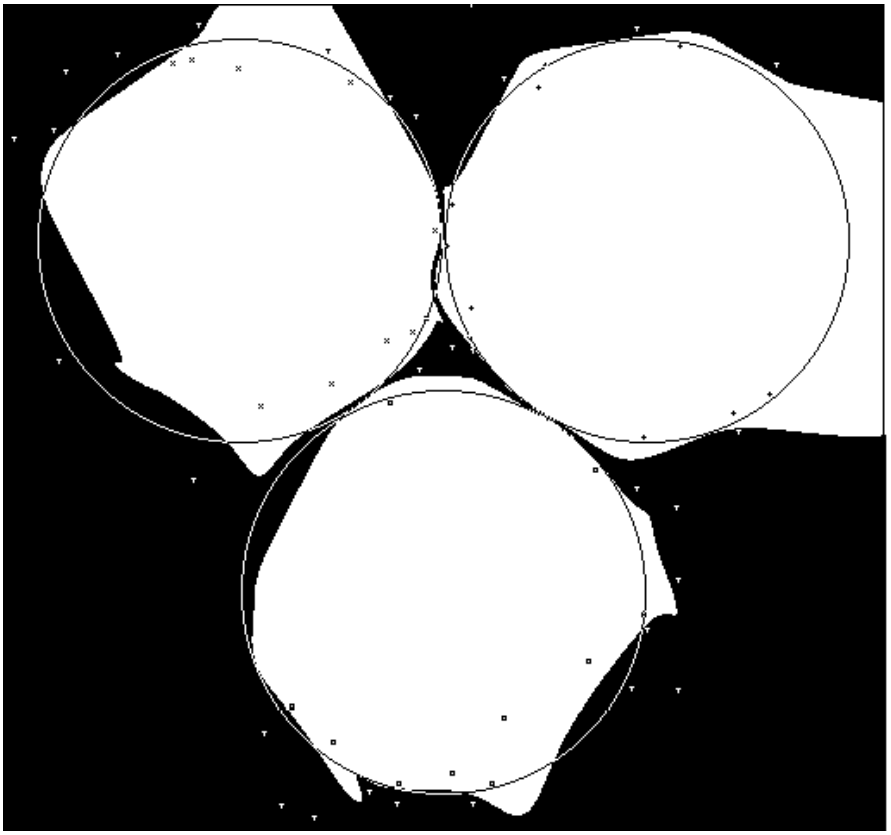
Algoritmii descrişi au fost testați pentru cazul existenței a trei clase plus o clasă de rejecție, folosindu-se pentru aceasta o rețea neurală (perceptron cu trei straturi) având doar *18* neuroni (*10+5+3*). Formele de intrare, câte *10* pentru fiecare clasă şi *30* pentru clasa de rejecție

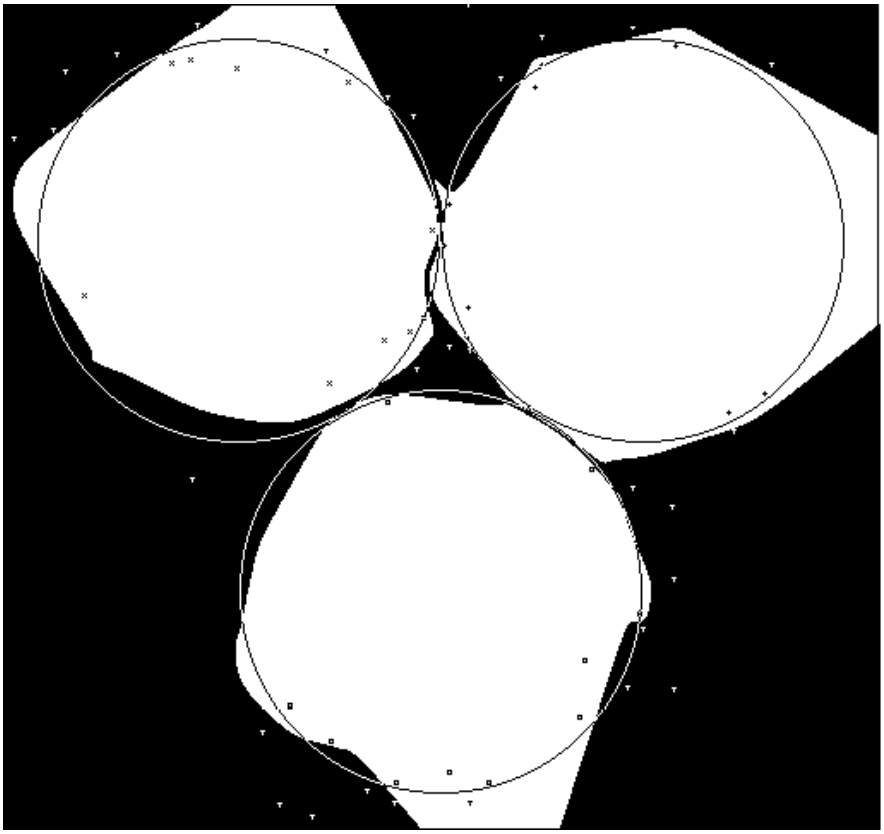
au fost amplasate aleator în interiorul a trei cercuri tangente, cu raze egale, marcate în figurile următoare. Forme de intrare generate la fel şi nerecunoscute inițial au fost incluse în setul de antrenare, după eliminarea unor forme deja existente în set. Formele sunt generate cu distribuție normală şi dispersie *0.4* la o rază a claselor de *0.5*.

Se remarcă în figura următoare faptul că după prima antrenare a rețelei, limitele de decizie obținute sunt total nesatisfăcătoare, datorită numărului mic de forme din setul de antrenament.

După şase şi respectiv şapte iterații, structura obținută a claselor s-a ameliorat mult şi se îmbunătățeşte continuu.




NORMAL
SET= 10
VAR=0.40
err= 3
ETA=0.35
ALP=0.70
STP=10000
ITR= 6


NORMAL
SET= 10
VAR=0.40
err= 4
ETA=0.35
ALP=0.70
STP=10000
ITR= 7

17. Recunoaşterea obiectelor imagistice cu invarianţe specificate.

Studiul de faţă are drept scop final realizarea unor studii comparative privind unele metode de recunoaştere a obiectelor imagistice. O problemă care apare destul de des este recunoaşterea obiectelor dintr-o imagine indiferent de mărimea (distanţa faţă de observator) şi orientarea lor.

Imaginile luate cu o cameră de luat vederi sunt prelucrate de un sistem software construit în acest scop, urmărindu-se într-o primă fază extragerea unor trăsături invariante la translaţie, rotaţie şi scalare pentru fiecare obiect care a fost depistat în imagine. Se propun două metode, una bazată pe transformarea imaginii în coordonate polare, cealaltă pe calculul momentelor Zernike.

Sunt luate în considerare şi unele metode de realizare a invarianţei la condiţiile de iluminare, utilizându-se o transformare care furnizează o imagine cu luminozitate şi contrast aproximativ constante.

Pentru faza de recunoaştere este utilizat un clasificator neural implementat cu un perceptron multistrat. Construirea acestui clasificator se face prin program, definindu-se în prealabil o bază de date de trăsături extrase din obiecte având diferite orientări, drept set de antrenament. Pentru testarea clasificatorului se foloseşte un alt set de obiecte din care s-au extras trăsăturile cu exact aceeaşi metodă.

Metodele folosite pot fi folosite într-o gamă largă de aplicaţii şi de aceea se are în vedere continuarea studiilor pentru:

-perfecţionarea algoritmilor de segmentare folosiţi;

-implementarea unor metode mai eficiente de localizare a obiectelor de formă cunoscută dintr-o imagine, folosind transformata Hough;

-optimizări de viteză;

-studiul numărului necesar de trăsături de extras raportat la complexitatea problemei.

] Sunt prezentate în acest studiu etapele parcurse pentru construcţia unui sistem-test de recunoaştere a obiectelor imagistice în condiţii de invarianţă la translaţie, scalare şi rotaţie. Paşii de parcurs sunt explicaţi în detaliu, iar în finalul raportului este dată o prezentare a programului de test care implementează algoritmii descrişi.

Exemplele prezentate sunt extrase dintr-o aplicaţie de recunoaştere a numărului de serie înscris pe anumite ceasuri de mână. Exemple de imagini analizate de această aplicaţie sunt prezentate în continuare:

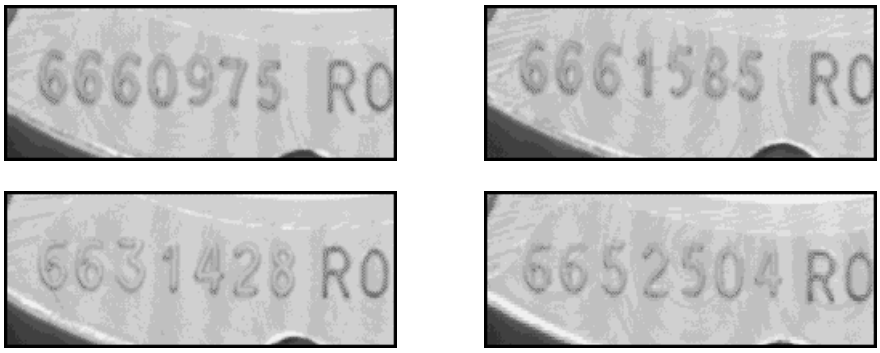


Fig. 1. Imagini ale numărului serial al ceasurilor.

17.1. Segmentarea

Segmentarea este un proces de partiţionare a imaginii digitizate în submulţimi, prin atribuirea pixelilor individuali la aceste submulţimi (denumite şi clase), rezultând obiectele distincte din scenă.

Algoritmii de segmentare se bazează, în general, pe două principii de bază:

- discontinuitate, având ca principală metodă detecţia conturului;
- similitudine, cu metoda pragului şi metoda regiunilor, cele mai utilizate.

1) Detecţia conturului.

Un operator de contur este un operator matematic, cu o extindere spaţială mică, pentru a putea determina existenţa unui contur local în funcţia imagine. Există diferite tipuri de operatori de contur, în funcţie de aplicaţie, dar proprietatea care-i uneşte este faptul că determină direcţia schimbării intensităţii maxime a nivelului de gri şi modulul, care dă severitatea acestei schimbări.

Utilizarea măştilor de convoluţie spaţială care baleiază întreaga imagine pixel cu pixel, calculând o mărime proporţională cu discontinuitatea existentă în porţiunea de imagine aflată sub mască constituie cei mai simpli şi rapizi asemenea operatori de detecţie a contururilor.

2) Metoda pragului.

Pentru cele mai multe imagini sunt puţine locuri unde modulul gradientului este zero, datorită prezenţei zgomotului în imagine. Din acest motiv se utilizează o metodă mai expeditivă, cea a pragului, care considera un element de contur în punctul de coordonate *(i,j)* dacă funcţia *f(i,j)* este mai mare decât un prag. Obţinerea unor bune rezultate cu această metodă depinde de alegerea pragului.

3) Metoda regiunilor.

Segmentarea iterativă sau segmentarea prin tehnici de relaxare este o metodă care foloseşte procedee probabilistice de clasificare în fiecare punct, în paralel la fiecare iteraţie. Procesul de relaxare este conceput pentru a aduce nivelele de gri spre capetele opuse ale scării de gri astfel încât pragul *T* să devină o problemă banală. Acest tip de procesare este fundamental în aria analizei automate de scene sau a recunoaşterii de modele, unde una dintre dorinţe este de a folosi maşina pentru a extrage date detaliate despre conţinutul imaginii la nivel de obiect. Obiectivul tehnicii de segmentare este de a partiţiona o imagine dată în regiuni sau componente.

De exemplu, pentru o imagine derivată dintr-o scena tridimensională, obiectivul segmentării poate fi identificarea regiunilor corespunzătoare obiectelor din scenă.

Putem considera segmentarea ca o problemă punctuală sau regională, ambele legate de procesul în cauză. În prima categorie intră metodele care se bazeaza pe examinarea imaginii pixel cu pixel. În a doua categorie, aşa cum implică şi denumirea, intră metodele care se bazează pe utilizarea informaţiilor din imagine în prescrierea vecinătăţilor. În ambele cazuri putem vedea problema ca o luare de decizie în procesul de recunoaştere a modelelor, ale cărui obiective sunt stabilite în limitele unor regiuni.

Segmentarea scenelor diferă de celelalte probleme de recunoaştere prin câteva proprietăţi. Una din cele mai importante diferenţe este că după ce am obţinut soluţia, regiunile dintr-o imagine pot fi vizualizate, iar apoi verificate la un mare nivel de acurateţe. În mod uzual este oricând posibil să suprapunem rezultatele segmentării cu originalul pentru a putea determina eficacitatea metodei. Prima operaţie este de a localiza graniţele sau marginile regiunilor. A doua operaţie este de a grupa punctele în regiuni similare, cu alte cuvinte chiar determinarea hotarelor. Cele două metode sunt similare şi le-am putea defini ca exprimate în puncte sau ca intersectie a două suprafeţe.

În situaţiile în care este cerută setarea automată a pragului, această problemă revine la a caracteriza o histogramă dată într-un mod invariant.

Presupunem că este ştiut aprioric că imaginea conţine doua regiuni principale. Dacă forma densităţilor de probabilitate este cunoscută atunci este posibil să determinăm un prag optim (în direcţia unei erori minime) pentru segmentarea imaginii în două regiuni diferite de strălucire.

Presupunem că imaginea conţine două valori combinate cu un zgomot aditiv Gaussian. Funcţia de densitate probabilă a mixturii este dată de:

$$\boldsymbol{p}(x) = P_1 \ \boldsymbol{p}_1(x) + P_2 \ \boldsymbol{p}_2(x) \tag{1}$$

şi pentru cazul Gaussian :

$$\boldsymbol{p}(x) = \frac{P_1}{\sqrt{2\pi}\sigma_1}\boldsymbol{exp}\frac{(x-\mu_1)^2}{2\sigma_1^2} + \frac{P_2}{\sqrt{2\pi}\sigma_2}\boldsymbol{exp}\frac{(x-\mu_2)^2}{2\sigma_2^2} \tag{2}$$

unde: μ_1 şi μ_2 sunt valorile medii ale celor două nivele de strălucire,

σ_1 şi σ_2 sunt deviaţiile standard de la medie,

P_1 şi P_2 sunt probabilităţile nivelelor.

Având în vedere faptul că relaţia următoare este satisfăcută automat:

$$P_1 + P_2 = 1 \tag{3}$$

în formulă avem cinci parametri necunoscuţi. Deci dacă toţi parametrii ar fi cunoscuţi atunci pragul optim ar fi uşor de determinat.

Presupunem că regiunea întunecată corespunde fondului iar regiunea mai stralucitoare corespunde obiectelor. În acest caz $\mu_1 < \mu_2$ şi vom defini pragul T astfel încât să considerăm că toti pixelii cu nivelul de gri aflat sub pragul T fac parte din punctele fondului, iar toţi pixelii care au nivelul de gri peste pragul T fac parte din obiecte.

Probabilitatea clasificării eronate a punctelor din cadrul unui obiect ca fiind puncte din fond este:

$$\boldsymbol{E}_1(T) = \int_{-\infty}^{T} \boldsymbol{p}_2(x)dx \tag{4}$$

Similar, probabilitatea clasificării unui punct de fond ca fiind punct-obiect este:

$$\boldsymbol{E}_2(T) = \int_{T}^{\infty} \boldsymbol{p}_1(x)dx \tag{5}$$

De aici eroarea totală probabilă este:

$$\boldsymbol{E}(T) = P_2\,\boldsymbol{E}_1(T) + P_1\,\boldsymbol{E}_2(T) \tag{6}$$

Pentru a găsi acea valoare a pragului pentru care această eroare este minimă, va trebui să diferenţiem $\boldsymbol{E}(T)$ relativ la T (folosind regula lui Leibnitz) şi să egalăm rezultatul cu zero:

$$P_1\ \boldsymbol{p}_1(T) = P_2\ \boldsymbol{p}_2(T) \tag{7}$$

Deoarece zgomotul afectează în mod egal atât fondul cât şi obiectele, avem $\sigma^2 = \sigma_1^2 = \sigma_2^2$ şi atunci:

$$T = \frac{\mu_1 + \mu_2}{2} + \frac{\sigma^2}{\mu_2 - \mu_2}\boldsymbol{ln}\frac{P_1}{P_2} \tag{8}$$

Dacă probabilităţile sunt egale $P_1 = P_2$, atunci pragul optim este chiar media. În capitolul 6 este prezentată o metodă grafică de determinare a parametrilor formulei anterioare şi depistare automată a pragului optim.

În capitolul 18 este prezentată o implementare foarte robustă, bazată pe prelucrarea vectorului histogramă folosind filtre nerecursive. Filtrul este aplicat succesiv de mai multe ori, până când histograma devine clar bimodală, după care pragul se obţine imediat folosind o formulă deviată din cea precedentă.

În practică, dacă aplicaţia studiată permite obţinerea unor informaţii suplimentare, se pot utiliza metode mai puţin sofisticate. Astfel, dacă se cunoaşte sau poate fi estimat raportul R între aria obiectelor din imagine şi aria fondului, atunci pragul de segmentare dorit este dat de primul nivel de gri (ordine crescătoare) de la 0[*negru*] la N-1[*alb*, de obicei 256] pentru care este îndeplinită următoarea inegalitate:

$$\sum_{j=1}^{T-1} H[j] \Big/ \sum_{j=T}^{N} H[j] > R \tag{9}$$

O asemenea metodă, foarte rapidă, a fost folosită în implementarea aplicaţiei de recunoaştere a numărului de serie a ceasurilor de mână.

17.2. Localizarea obiectelor într-o imagine.

Scopul acestei operaţii este de a localiza obiectele dintr-o scenă pentru a putea fi apoi supuse ori altor prelucrări ori unor transformări de imagine (de exemplu scalări, invarianţe la rotaţii, translaţii, etc.).

În principiu această metodă se bazează pe găsirea zonelor de contact între fondul imaginii şi obiectul-imagine. În studiu s-a folosit această tehnică după realizarea segmentării, obţinând o separare precisă a fondului de obiecte.

Presupunem că asociem fondului codul *1*, iar obiectului- imagine codul *0*. Problema ar reveni la a determina perechile de pixeli *(1,0)* şi a marca pixelii de graniţă ai obiectului.

Algoritmul ce se poate folosi este următorul:

Algoritm de localizare obiecte.

(1). Se baleiază imaginea linie cu linie până se găseşte perechea de pixeli *(1,0)*, în care pixelul obiect este denumit pixel de start: *(xs,ys)* ;

(2). Se baleiază pixelii vecini până când se determină un alt pixel obiect, folosind pentru aceasta următoarea matrice de căutare :

7	**0**	**1**
6	**C**	**2**
5	**4**	**3**

unde: *C* = este pixelul de căutare vechi;
1, ... , 7 = direcţiile corespunzătoare după care fac căutarea.

Acestor direcţii li se asociază doi vectori de direcţie care reprezintă incrementii pentru abscisa şi ordonata pixelului curent.
Vectorul incrementţilor coordonatei x pentru pixelul de căutare este:

$$V_X = [0,1,1,1,0,-1,-1,-1]$$

Vectorul incrementilor coordonatei **y** pentru pixelul de căutare este:

$$V_Y = [-1,-1,0,1,1,1,0,-1]$$

Noul pixel găsit devine pixel actual şi este marcat. În acelaşi timp se reţin valorile actualizate pentru minimele şi respectiv maximele lui x, y.

(3). Stop când pixelul curent coincide cu pixelul de start, adică:

$$xc = xs, yc = ys$$

În cazul existenţei mai multor obiecte în scena respectivă, se caută un alt obiect, aceasta revenind la găsirea unei alte perechi de pixeli de tipul $(1,0)$.

În program s-a folosit şi testul de pixel izolat, care se bazează tot pe căutarea după matricea de direcţii. În plus, tehnica standard de conturare este de a determina toate zonele de graniţă *(1,0)*, apoi se elimină acele zone din interiorul obiectelor mari. În studiu s-a folosit o variantă a acestei metode fără a mai căuta conturul interior al obiectelor deja localizate.

În multe cazuri, în lista obiectelor localizate apar şi obiecte a căror analiză nu se doreşte. De aceea trebuie proiectat, cel mei adesea pe baze euristice, un selector de obiecte utile – numit adesea şi ***filtru logic*** sau ***filtru euristic***. Folosind cel mai adesea criterii dimensionale (lungime, lăţime, arie, perimetru, raport de svelteţe, etc.) se selectează din mulţimea obiectelor localizate pe acelea care vor face obiectul operaţiei propriu-zise de recunoaştere.

Un asemenea filtru logic va fi descris în capitolul 18, criterii asemănătoare de selecţie fiind aplicate şi în cazul aplicaţiei de recunoaştere a numărului serial marcat pe ceasuri de mână.

17.3. Invarianţa la condiţiile de iluminare.

Urmând modelul reglajului automat al amplificării existent în orice camera TV (şi nu numai), se poate construi o transformare punctuală care, aplicată tuturor pixelilor din zona de interes (de obicei numai pixelilor obiectului de recunoscut, determinarea cărora s-a făcut prin segmentare şi apoi parcurgerea contururilor lor) astfel încât să se asigure o luminozitate globală constantă.

Problema se poate traduce matematic prin setarea mediei şi dispersiei variabilei aleatoare ataşate nivelului de gri al pixelilor obiectului la valori prestabilite. Media şi respectiv dispersia sunt date de:

$$m_\Theta = \frac{1}{M_\Theta} \sum_{(x,y)\in\Theta} f(x,y) \tag{10}$$

$$\sigma_\Theta^2 = \frac{1}{M_\Theta} \sum_{(x,y)\in\Theta} [f(x,y) - m_\Theta]^2 \tag{11}$$

unde: - Θ este mulţimea pixelilor obiectului a cărui invarianţă la iluminare se urmăreşte;
- M_Θ este numărul pixelilor corespunzători acestui obiect Θ.

Atunci variabila aleatoare:

$$\bar{f}(x,y) = f(x,y) - m_\Theta,\ (x,y)\in\Theta \tag{12}$$

are media nulă şi dispersia neschimbată,

$$\tilde{f}(x,y) = \frac{f(x,y) - m_\Theta}{\sigma_\Theta^2},\ (x,y)\in\Theta \tag{13}$$

este o variabilă aleatoare cu media nulă şi dispersia unitară, iar:

$$\tilde{f}(x,y) = \frac{f(x,y) - m_\Theta}{\sigma_\Theta^2} + q,\ (x,y) \in \Theta \tag{14}$$

reprezintă o variabilă aleatoare cu media "q" şi dispersie unitară. Constanta "q" se alege de obicei la mijlocul gamei dinamice a nivelelor de gri cu care se lucrează (ex. $q=128$).

Fig. 2. Invarianţa la condiţiile de iluminare.

O altă modalitate de asigurare a invarianţei la condiţiile de iluminare este egalizarea histogramei nivelelor de gri corespunzătoare pixelilor obiectului. Fiind mai complexă decât precedenta soluţie, se foloseşte destul de rar.

17.4. Transformarea din coordonate carteziene în coordonate polare.

După ce s-a realizat localizarea într-un dreptunghi de arie minimă a obiectelor din imagine folosind imaginea binară, se localizează centrul de greutate al acestor obiecte şi se determină raza minimă a cercului de încadrare (circumscris), având centrul în centrul de greutate al obiectelor. Coordonatele centrului de greutate sunt date de următoarele formule:

$$X_G = \frac{\sum_{i=x_{min}}^{x_{max}} \sum_{j=y_{min}}^{y_{max}} x_i * f(x_i,y_j)}{\sum_{i=x_{min}}^{x_{max}} \sum_{j=y_{min}}^{y_{max}} f(x_i,y_j)}, \qquad Y_G = \frac{\sum_{i=x_{min}}^{x_{max}} \sum_{j=y_{min}}^{y_{max}} y_i * f(x_i,y_j)}{\sum_{i=x_{min}}^{x_{max}} \sum_{j=y_{min}}^{y_{max}} f(x_i,y_j)} \tag{15}$$

Raza de încadrare se obţine parcurgând iarăşi conturul şi determinând distanţa maximă faţă de centrul de greutate.

Realizarea adiţională a acestei operaţii asigură invarianţa la translaţie, prin raportarea ulterioară a tuturor coordonatelor la centrul de greutate al fiecărui obiect din imagine.

Aproape orice transformare geometrică a unei imagini se realizează aplicând în final o interpolare liniară în două dimensiuni. Prin aceasta dorim să găsim o estimare a lui $G(x_1, x_2, \ldots, x_n)$, pentru fiecare din varibilele independente $x_1, x_2, \ldots, x_n$.

Pentru cazul particular al trecerii în coordonate polare avem:

$$\begin{cases} r = \sqrt{x^2 + y^2} \\ \theta = \mathbf{arctg}\,(y/x) \end{cases} \qquad \begin{cases} x = r\cos\theta \\ y = r\sin\theta \end{cases} \tag{16}$$

Având o matrice $f(j,k)$, unde $j = 1...m; k = 1...n$ şi funcţiile r şi s, relaţia dintre aceste valori de intrare şi funcţia $f(r,s)$ este:

$$G(j,k) = f[r(j), s(k)] \tag{17}$$

Scopul este să estimăm prin interpolare valoarea funcţiei în câteva puncte (r,s). Grila pătratică de reprezentare a imaginii în coordonate carteziene face ca punctul (r,s) din matricea polară să aibă corespondent un punct localizat prin indecşi neîntregi, deci în afara grilei carteziene. Totuşi, el este înconjurat de patru puncte ale acestei grile cunoscute. Deci mai precis, dacă:

$$r(j) \le r \le r(j+1) \qquad \text{şi} \qquad s(k) \le s \le s(k+1) \tag{18}$$

şi având j, k :

$$g_1 \equiv \mathbf{G}(j,k); \qquad g_2 \equiv \mathbf{G}(j+1,k); \tag{19}$$
$$g_3 \equiv \mathbf{G}(j,k+1); \qquad g_4 \equiv \mathbf{G}(j+1,k+1);$$

Interpolarea liniară se obţine cu:

$$t \equiv [\mathbf{r} - \mathbf{r}(j)]/[\mathbf{r}(j+1) - \mathbf{r}(j)] \tag{20}$$
$$u \equiv [\mathbf{s} - \mathbf{s}(k)]/[\mathbf{s}(k+1) - \mathbf{s}(k)]$$

cu u, t între *0* şi *1*:

$$\mathbf{G}(x_1,x_2) = (1-t)(1-u)g_1 + t(1-u)g_2 + (1-t)ug_3 + tug_4 \tag{21}$$

În prelucrarea de imagini aceasta are ca efect regăsirea fiecărui sector din cercul de încadrare al imaginii-obiect în linia corespunzătoare matricii transformatei polare.

17.5.Transformarea inversă: coordonate polare-coordonate carteziene

Se procedează astfel: se pleacă de la punctele din imaginea originală şi în funcţie de poziţionarea acestora faţă de centrul de greutate al obiectului-imagine, se găseşte poziţia valorii corespondente în matricea transformării polare. Ca şi în cazul precedent, se poate aplica o interpolare, de obicei liniară.

17.6. Scalarea şi invarianţa la rotaţii.

Scalarea este realizată dacă în transformarea din coordonate carteziene în coordonate polare este respectată condiţia ca raza maximă (devenită ordonată maximă) a matricii transformării în coordonate polare să fie constantă pentru toate obiectele. În programul realizat se consideră raza polară ca fiind egală cu *32*, regăsită în numărul de linii ale matricii polare.

Această matrice reprezintă spaţiul coordonatelor polare în care originea este în $(0,0)$, axa unghiurilor este abscisa, iar axa razelor este ordonata. Având în vedere că raza este tot timpul *32* avem de-a face într-adevăr cu o scalare.

Se folosesc valorile estimate pentru valorile discrete:

$$f_n \equiv \frac{n}{N\Delta}, n = -\frac{N}{2}, \ldots, \frac{N}{2} \tag{22}$$

Următorul pas este de a calcula transformata Fourier discretă a liniilor matricii transformării polare:

$$H_n \equiv \sum_{k=0}^{N-1} h_k e^{2\pi \mathrm{j\,k\,n/N}} \tag{23}$$

Ultima sumare este transformata Fourier discretă a N puncte h_k.Transformata Fourier operează asupra a N numere complexe (h_k -urile) şi furnizează N numere complexe (H_n -urile).

Pentru a putea face transformata pentru un set de numere reale se împărte setul de date în două subseturi luând numerele cu index ***par*** f_j pentru un set, iar numerele cu index ***impar*** $\mathbf{f_j}$ pentru celălalt set. *Puterea acestui artificiu consta în faptul că putem vedea vectorul real de intrare ca o arie de numere complexe cu dimensiunea la jumătate*. Cu alte cuvinte putem considera:

$$h_j = f_{2j} + i f_{2j+1} \text{ , pentru } j = 0, \ldots, N/2\text{-}1 \tag{24}$$

Vom aplica asupra acestora algoritmul FFT, care returnează un vector complex:

$$H_n = F_n^e + jF_n^0, \; n = 0, \ldots, N/2\text{-}1 \tag{25}$$

cu:

$$F_n^e = \sum_{k=0}^{N/2-1} f_{2k} e^{2\pi j\,k\,n/(N/2)} \qquad \text{şi} \qquad F_n^0 = \sum_{k=0}^{N/2-1} f_{2k+1} e^{2\pi j\,k\,n/(N/2)} \tag{26}$$

Pentru a obţine invarianţa la rotaţie calculăm ***modulul transformatei Fourier*** deoarece o rotaţie în spaţiul cartezian $f(x,y)$ se regăseşte în spaţiul (r,f_i), al transformării polare ca o translaţie şi atunci transformata Fourier a acestei mişcări este:

$$\tilde{F}\{ f(x-x_0)\} = \tilde{F}\{ f(x)\} * e^{-jx_0} \qquad (27)$$

Având în vedere că $|e^{-jx_0}| = 1$ şi aplicând modulul obţinem egalitate între modulul translaţiei în (r,f_i) şi funcţia netranslată.

17.7. Extragerea trăsăturilor din imagine.

În urma scalării şi a prelucrărilor pentru a realiza invarianţa la rotaţie se obţine o matrice de dimensiune constantă $(32*32)$, conţinând informaţiile procesate din imaginea primară.

Paşii ce urmează a fi făcuţi sunt următorii:

a) **Transformata Cosinus Discretă**.

Deoarece implementarea transformării optimale Karhunen-Loeve **(KL)** este dificilă, modelând intrarea $\{X\}$ cu un proces Markov de ordinul unu, cea mai bună aproximare a transformării **KL** discretă este **DCT** (Transformata Cosinus Discretă). Această modelare nu este foarte restrictivă şi se poate folosi cu succes în multe aplicaţii.

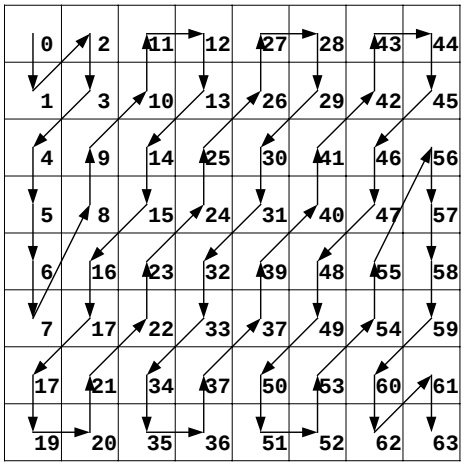
Pentru calculul **DCT** există nu numai algoritmi rapizi de calcul, ci şi circuite specializate de mare viteză. Pentru cazul bidimensional se foloseşte:

$$F(u,v) = \frac{1}{2N} \sum_{j=0}^{N-1} \sum_{i=0}^{N-1} f(j,k) \cos\left(\frac{\pi u j}{N}\right) \cos\left(\frac{\pi v k}{N}\right) \qquad (28)$$

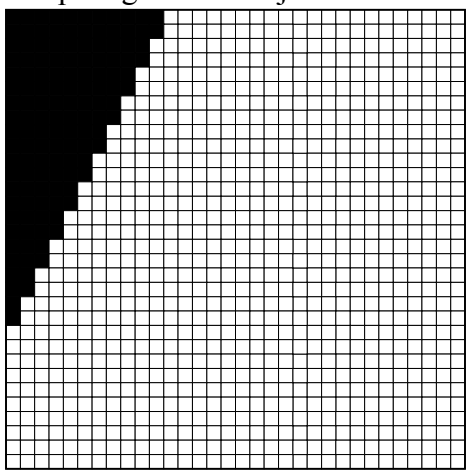
rezultând o matrice de aceeaşi dimensiune, cu valori reale slab corelate. Valorile semnificative ale acestei matrici sunt plasate în colţul din stânga sus.

b) **Ordonarea Zig - Zag modificată.**

Este o procedură de vectorizare a matricii DCT după regula de mai jos:



Mat [8][8] ===> Z[64]



Mat [D][D] ===>Z[D*D]

Deoarece **DCT** bidimensională concentrează informaţia din imaginea iniţială în colţul din stânga sus al matricii rezultate, după ordonarea în Zig-Zag aceeaşi informaţie va fi regasită în primele elemente ale vectorului obţinut. Forma specifică a "colţului" în care se concentrează informaţiile semnificative se datorează transformării Fourier (FFT) efectuate anterior asupra liniilor matricii, transformare care prezintă şi ea proprietăţi de decorelare, însă mai slabe decât DCT.

c)Se reţin primele *F* valori din vectorul ***Z[D*D]*** , adică doar o fracţiune din valori, cele marcate cu negru în figura de mai sus. Conform afirmaţiilor anterioare, aceste elemente concentrează cea mai mare parte din informaţia din imaginea iniţială şi ele vor fi folosite în faza de recunoaştere.

Acest bloc funcţional elimină trăsăturile care sunt neimportante în faza de recunoaştere. Ieşirile lui sunt aplicate la intrarea ***clasificatorului neural***, descris mai amănunţit în capitolul precedent.

Obiectivul antrenării reţelei neurale este de a ajusta ponderile astfel încât aplicarea unui set de intrări să producă ieşirea dorită. Vom denumi seturile de intrări-ieşiri vectori. Antrenarea atribuie fiecărui vector de intrare un vector pereche denumit vector ţintă şi care reprezintă ieşirea dorită. Acestea amândouă sunt denumite "perechea de antrenare".

Uzual o reţea este antrenată după un anumit număr de perechi de antrenare, iar grupul perechilor de antrenare este denumit "set de antrenare". Înainte de a începe procesul de antrenare, toate ponderile trebuie să fie iniţializate la valori aleatoare mici. Aceasta ne asigură că reţeaua nu este saturată prin valori mari ale ponderilor, şi ne asigură că nu mai apar şi alte defecte de antrenare (de exemplu dacă toate ponderile de start au valori egale iar performanţele dorite cer valori neegale, atunci reţeaua nu va învaţa).

17.8. Momentele Zernike.

Drept variantă alternativă a algoritmului descris anterior se prezintă în continuare un tip special de momente ce prezinta proprietatea de invarianţă la rotaţie. Momentele au fost utilizate înca de la inceput drept trăsături într-un mare număr de aplicaţii. Momentele regulate au fost cele mai populare, ele fiind definite prin:

$$m_{pq} = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} x^p\, y^q \;\; \boldsymbol{f}(x,y)\, dx\, dy \tag{29}$$

unde "*p*" şi "*q*" indică ordinul momentului respectiv. Pentru imagini digitale, momentul se calculează prin:

$$m_{pq} = \sum_x \sum_y x^p\, y^q \;\; \boldsymbol{f}(x,y) \tag{30}$$

Momentele Zernike se definesc pe un domeniu egal cu interiorul cercului unitate:

$$x^2 + y^2 = 1 \tag{31}$$

Pentru construcţia lor se defineşte mai întâi un set de polinoame ortogonale, notate:

$$\boldsymbol{V}_{nm}(x,y), \text{ cu } n\text{-}|m| \text{ par şi } |m| < n \;\; , \tag{32}$$

Expresia de definiţie a lor este:

$$\boldsymbol{V}_{nm}(x,y) = \boldsymbol{V}_{nm}(\rho,\theta) = \boldsymbol{R}_{nm}(\rho)\,\boldsymbol{exp}(jm\theta) \tag{33}$$

unde:

$$\boldsymbol{R}_{nm}(\rho) = \sum_{s=0}^{(n-|m|)/2} (-1)^s \frac{(n-s)!}{s!\left[(n+|m|)/2-s\right]!\left[(n-|m|)/2-s\right]!}\rho^{n-2s} \tag{34}$$

Se poate remarca faptul că:

$$\boldsymbol{R}_{n,-m}(\rho) = \boldsymbol{R}_{nm}(\rho) \tag{35}$$

Aceste polinoame $\{\boldsymbol{V}_{nm}(x,y)\}$ fiind ortogonale, satisfac condiţiile de ortogonalitate:

$$\iint_{x^2+y^2\le 1} \left[\boldsymbol{V}_{nm}(x,y)\right]^* \boldsymbol{V}_{pq}(x,y)\, dx\, dy = \frac{\pi}{n+1}\delta_{np}\delta_{mq} \tag{36}$$

unde:

$$\delta_{ab} = \begin{cases} 1, \text{ pentru } a = b \\ 0, \text{ pentru } a \neq b \end{cases} \tag{37}$$

Momentele Zernike sunt proiecţii ale funcţiei imagine pe această bază ortogonală. Presupunând că $f(x,y) = 0$ în afara cercului unitate, avem momentul Zernike de ordinul $n + m$:

$$A_{nm} = \frac{n+1}{\pi} \iint\limits_{x^2+y^2 \le 1} f(x,y) V^*_{nm}(\rho,\theta)\, dx\, dy \tag{38}$$

Pentru cazul discret, momentul Zernike de acelaşi ordin se scrie:

$$A_{nm} = \frac{n+1}{\pi} \sum_x \sum_y f(x,y) V^*_{nm}(\rho,\theta), \quad cu \quad x^2 + y^2 \le 1 \tag{39}$$

Cunoscând momentele Zernike, se poate efectua transformarea inversă, care permite reconstrucţia imaginii iniţiale cu precizie din ce în ce mai mare pe masură ce "n" creşte:

$$\tilde{f}(x,y) = \sum_{n=0}^{n_{max}} \sum_m A_{nm} V_{nm}(\rho,\theta) \tag{40}$$

Descompunând această expresie în:

$$\tilde{f}(x,y) = \sum_{n=0}^{n_{max}} \sum_{m<0} A_{nm} V_{nm}(\rho,\theta) + \sum_{n=0}^{n_{max}} \sum_{m\ge 0} A_{nm} V_{nm}(\rho,\theta) \tag{41}$$

şi folosind o proprietate enunţată anterior, putem scrie:

$$\tilde{f}(x,y) = \sum_{n=0}^{n_{max}} \sum_{m>0} A_{-n,-m} V_{-n,-m}(\rho,\theta) + \sum_{n=0}^{n_{max}} \sum_{m\ge 0} A_{nm} V_{nm}(\rho,\theta) \tag{42}$$

adică:

$$\tilde{f}(x,y) = \sum_{n=0}^{n_{max}} \sum_{m>0} A^*_{nm} V^*_{nm}(\rho,\theta) + \sum_{n=0}^{n_{max}} \sum_{m\ge 0} A_{nm} V_{nm}(\rho,\theta) \tag{43}$$

Cele două sume pot fi acum unificate scriind:

$$\tilde{f}(x,y) = A_{n0} V_{n0}(\rho,\theta) + \sum_{n=0}^{n_{max}} \sum_{m>0} \left[A^*_{nm} V^*_{nm}(\rho,\theta) + A_{nm} V_{nm}(\rho,\theta) \right] \tag{44}$$

Deoarece este mai uşor de operat cu numere reale vom descompune numerele complexe din formula anterioară:

$$\begin{aligned} \tilde{f}(x,y) = {} & \left[\mathbf{Re}(A_{n0}) - j\, \mathbf{Im}(A_{n0}) \right] V_{n0}(\rho,\theta) + \\ & + \sum_{n=0}^{n_{max}} \sum_{m>0} \left[\left[\mathbf{Re}(A_{nm}) - j\, \mathbf{Im}(A_{nm}) \right] R_{nm}(\rho) (\cos m\theta - j \sin m\theta) + \right. \\ & \left. + \left[\mathbf{Re}(A_{nm}) + j\, \mathbf{Im}(A_{nm}) \right] R_{nm}(\rho) (\cos m\theta + j \sin m\theta) \right] \end{aligned} \tag{45}$$

Efectuând calculele rezultă expresia:

$$\tilde{f}(x,y) = \frac{1}{2} C_{n0} R_{n0}(x,y) + \sum_{n=0}^{n_{max}} \sum_m \left[C_{nm} \cos m\theta + S_{nm} \sin m\theta \right] R_{nm}(x,y) \tag{46}$$

în care

$$C_{nm} = 2\, \mathbf{Re}(A_{nm}) \quad <==> \quad C_{nm} = \frac{2n+2}{\pi} \sum_x \sum_y f(x,y) R_{nm}(x,y) \cos m\theta \tag{47}$$

şi:

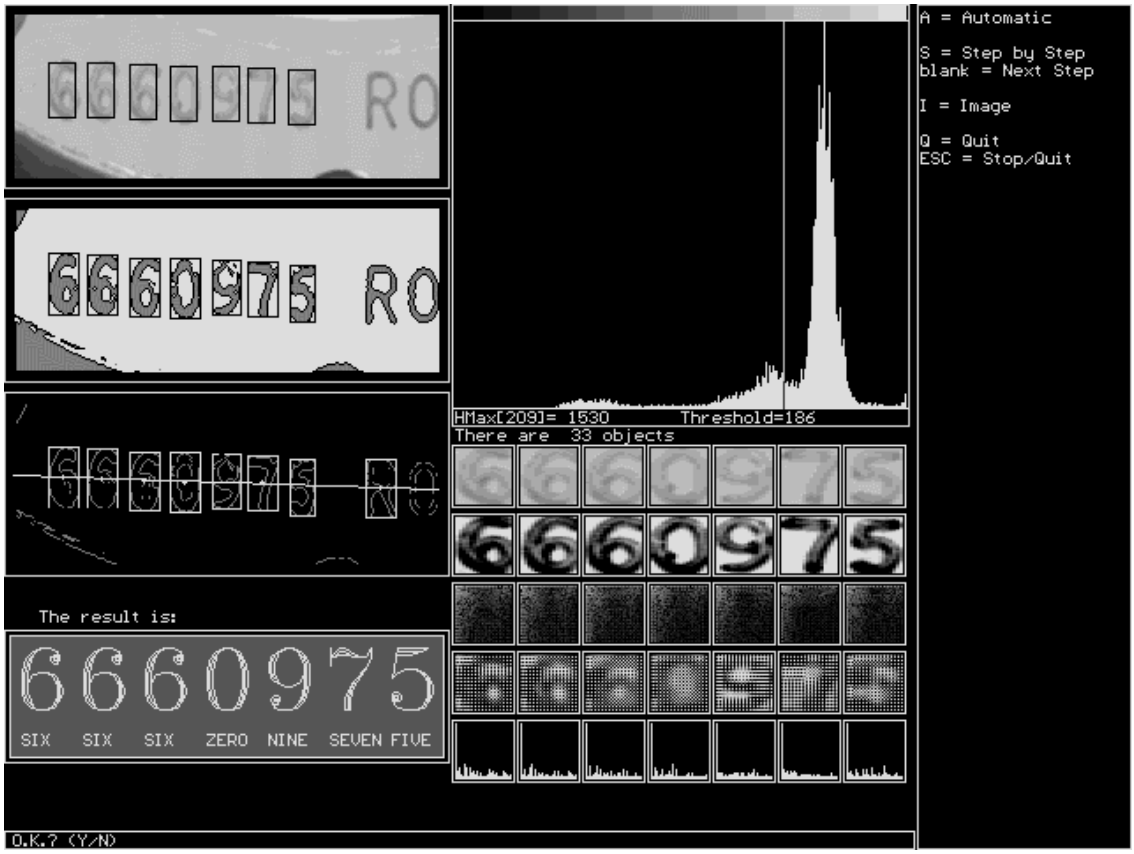
$$S_{nm} = -2\, \mathbf{Im}(A_{nm}) \quad <==> \quad S_{nm} = \frac{-2n-2}{\pi} \sum_x \sum_y f(x,y) R_{nm}(x,y) \sin m\theta \tag{48}$$

Se demonstrează foarte uşor că modulul momentelor Zernike este invariant la rotaţie şi ca urmare ele pot fi folosite drept trăsături invariante la rotaţie.

17.9. Rezultate obţinute, detalii de implementare.

Programul elaborat recunoaşte cu precizie foarte bună toate obiectele din imagine pentru care a fost antrenat. Erorile de recunoaştere care mai apar se datorează reducerii masive de date realizate în selectorul de trăsături (de ordinul a *60-70* ori), aceasta însemnând o limitare considerabilă a cantităţii de informaţie existente în imaginea iniţială. Testele au fost făcute pe imagini realizate în condiţii de variaţie a condiţiilor de iluminare, a poziţiei (rotaţie) şi a mărimii (scalare). Adăugând la toate acestea dezvoltarea unor algoritmi mai performanţi de segmentare, se poate face o idee asupra direcţiilor ulterioare de perfecţionare a programului.

În ce priveşte sistemul de recunoaştere automată a numărului serial al ceasurilor de mână , exte prezentat în figurile următoare ecranul principal al aplicaţiei pentru două imagini de test. Imaginile au fost furnizate sub formă de fişiere, urmând ca într-o etapă ulterioară să se conecteze direct aplicaţia la un sistem de achiziţie de imagini.



După cum se poate deduce şi din analiza imaginilor precedente, principalele operaţii efectuate de sistemul de recunoaştere a numărului serial sunt:

- Determinarea automată a pragului de segmentare prin analiza histogramei;
- Segmentarea cu prag;
- Localizarea obiectelor din imagine;
- Filtrul euristic de selecţie a cifrelor de recunoscut;
- Scalarea obiectelor de interes;
- Realizarea invarianţei la condiţiile de iluminare;
- Aplicarea transformatei DCT bidimensionale;
- Ordonarea Zig-Zag şi reţinerea trăsăturilor semnificative pentru fiecare obiect;
- Recunoaşterea propriu-zisă folosind o reţea neurală tip perceptron cu trei straturi.

Aplicația a fost scrisă în limbajul C (Borland C 3.1), iar codul sursă şi programul executabil sunt incluse în CD-ul ataşat cărții de față. Programul (numit DigitRecogn.exe) poate fi rulat după instalare.

18. Măsurarea automată a sitelor textile.

18.1. Introducere.

Standarde înalte in controlul calității textilelor pot fi obținute doar prin dezvoltarea şi implementarea unor noi metode de testare, caracterizate prin acuratețe înaltă şi viteză de răspuns mare. Măsurarea parametrilor produselor textile trebuie făcută cel mai adesea on-line, fără nici un fel de pierderi în productivitate.

Pentru măsurarea sitelor textile pot fi utilizate nu numai metode simple, cum ar fi cele bazate pe măsurători regionale ale transparenței (folosind de obicei o pereche –emițător-receptor- de dispozitive opto-electronice), ci şi metode mai sofisticate, cum ar fi cele bazate pe difracția laser (Sodomka[103]), metodele bazate pe analiza Fourier (Ribolzi[89]), sau metodele bazate pe analiza digitală a imaginilor (Wolfrum[112]), (Baetens[08]).

Există o multitudine de metode de analiză a produselor textile, multe dintre ele fiind destinate unui anumit tip de material textil. În particular, sitele textile au o structură vizuală foarte simplă, ceea ce le face potrivite pentru analiza computerizată a imaginilor lor. În consecință, am încercat să îmbunătățim precizia evaluării lor printr-o metodă bazată pe analiza digitală a imaginilor. Ea urmăreşte în primul rând îmbunătățirea preciziei măsurătorilor. Se foloseşte un nou algoritm de aproximare poligonală pentru localizarea ochiurilor sitei, după care o evaluare statistică completă a parametrilor eşantionului de sită textilă studiat este furnizată.

Toate etapele metodei de măsurare automată sunt descrise în amănunt. Metoda a fost implementată şi testată, şi este dată o descriere amănunțită a rezultatelor experimentale obținute.

18.2. Etapele metodei de măsurare automată.

Lanțul prelucrărilor folosite pentru a obține măsurătorile dorite asupra unor eşantioane de site textile este descris în următoarea schemă bloc (Fig. 1).

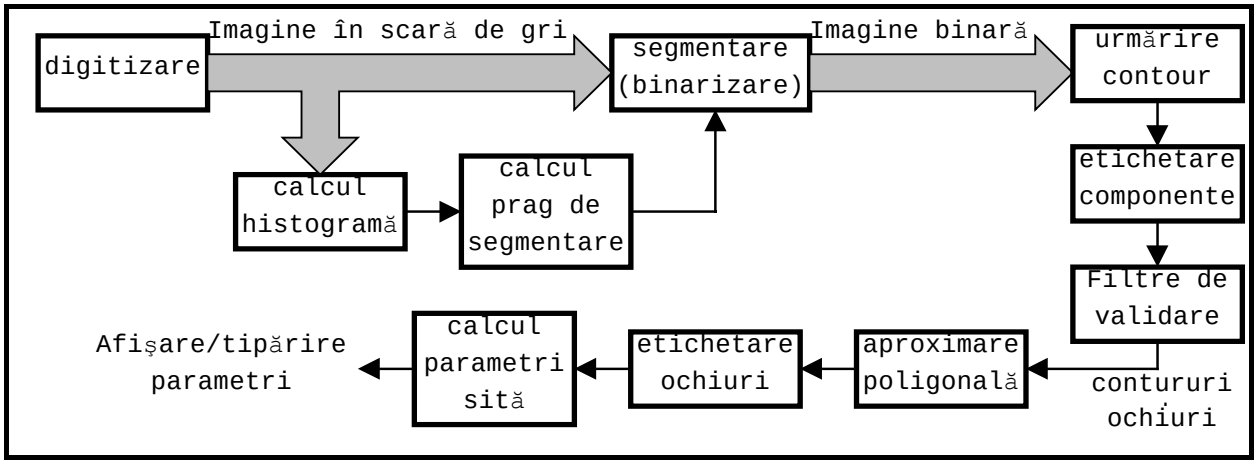


Figura 1. Schema bloc a sistemului.

Primul pas către scopul propus este obținerea unor imagini de bună calitate (Ballard[10]) ale sitelor textile (iluminare uniformă, claritate, contrast bun, fără zgomot). Folosind o camera TV conectată la un microscop şi o interfață specializată de calculator, aceste imagini sunt digitizate în scară de gri (256 de nivele) şi, dacă este necesar, salvate ca fişiere pe unitatea de disc dur a calculatorului (fig. 2).

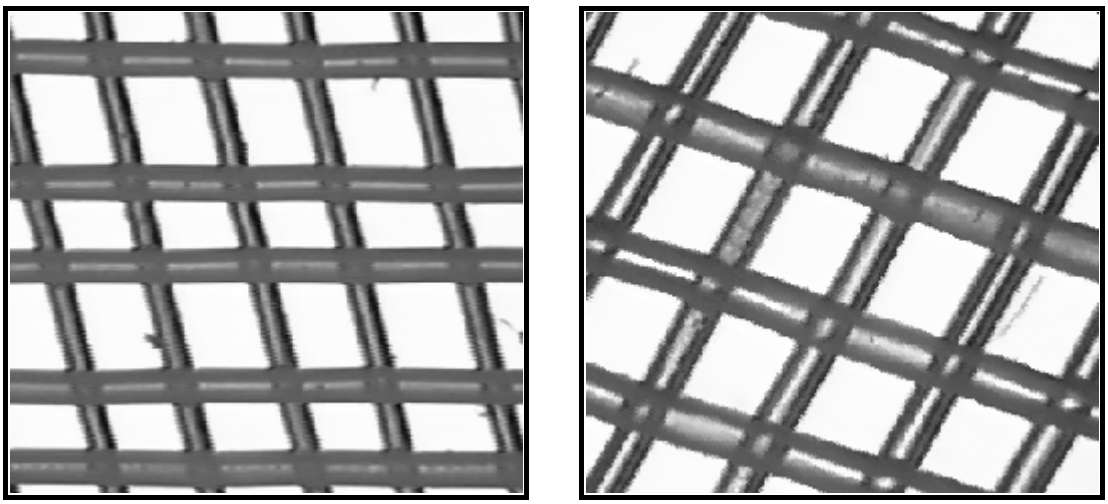


Fig. 2. Imagini ale eşantioanelor de sită textilă.

18.3. Segmentarea.

Pentru fiecare imagine se calculează histograma nivelelor de gri (fig. 3) în scopul obținerii unui prag (nivel de gri) de segmentare cât mai bun (Pavlidis[79]), (Haralick[45]), (Pratt[84]). Două strategii au fost testate pentru a realiza acest lucru:

1. Presupunând că următorul raport este cunoscut:

$$R="arie\ fire"/"arie\ ochiuri" \tag{1}$$

sau poate fi estimat pentru un anume eşantion de sită textilă, pragul de segmentare dorit este dat de *primul nivel de gri (ordine crescătoare)* de la 0*[negru]* la N-1*[alb, de obicei 256]* pentru care este îndeplinită următoarea inegalitate:

$$\sum_{j=1}^{T-1} \boldsymbol{H}[j] \Big/ \sum_{j=T}^{N} \boldsymbol{H}[j] > R \tag{2}$$

Această strategie este foarte rapidă, dar valoarea raportului R trebuie cunoscută. Acest raport depinde de densitatea sitei textile, grosimea firelor folosite, transparența lor, şi deci el poate varia foarte mult de la un eşantion la altul.

2. Folosind un filtru digital liniar nerecursiv, vectorul histogramă este filtrat până când mai rămân doar două maxime locale (fig. 4). Două soluții au fost încercate pentru a construi acest "filtru de netezire a histogramei ". Prima dintre ele este un filtru de ordinul trei:

$$\boldsymbol{y}[j] = \frac{1}{3}\left(\boldsymbol{x}[j-1] + \boldsymbol{x}[j] + \boldsymbol{x}[j+1]\right) \tag{3}$$

care trebuie aplicat succesiv vectorului histogramă până acestuia îi mai rămân doar două maxime locale. A doua soluție implică folosirea unui filtru de ordin superior:

$$\boldsymbol{y}[j] = \frac{1}{2k+1}\sum_{i=j-k}^{i=j+k}\boldsymbol{x}[i] \tag{4}$$

unde *k* creşte de la *1* la acea valoare pentru care histogramei filtrate îi mai rămân doar două maxime locale. Ordinul filtrului este $2k+1$.

Principalul dezavantaj al acestui din urmă filtru este că, pentru imagini cu contrast scăzut (cele două maxime locale sunt foarte apropiate unul de altul), şi când ordinul filtrului de apropie ca valoare de distanţa între aceste maxime, atunci precizia localizării lor nu mai este aşa de bună comparativ cu primul filtru descris.

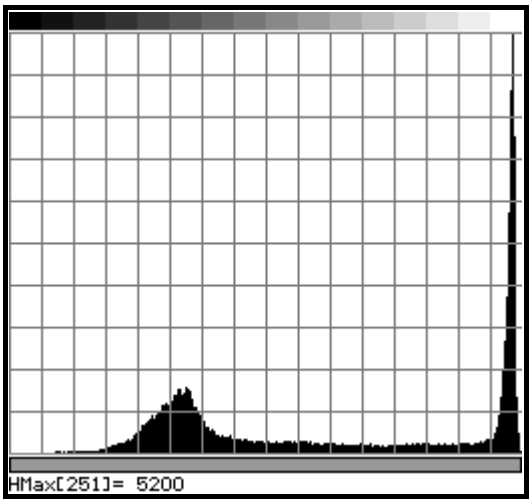


Fig. 3. Histograma inițială.

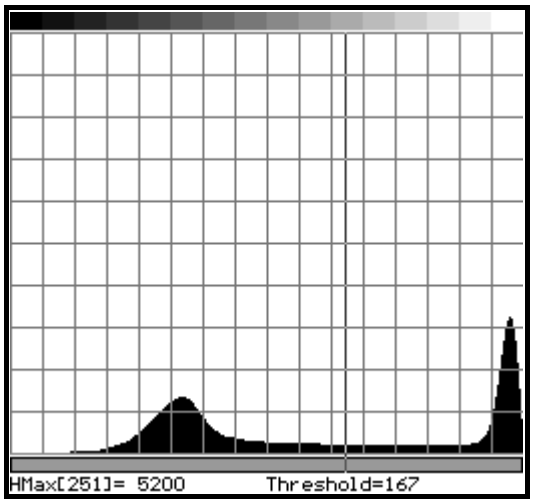


Fig. 4. Histograma filtrată.

Pragul de segmentare poate fi stabilit ca fiind nivelul de gri corespunzător *minimului local* aflat între cele două maxime localizate anterior, dar dacă cele două maxime principale sunt foarte bine separate, aşa cum apar ele în figura 4, atunci precizia localizării acestui minim local nu este aşa de bună. O variantă mai bună calculează acest prag ca o funcție de pozițiile maximelor găsite, adică, dacă m_1, m_2 sunt nivelele de gri pentru care vectorul histogramă atinge cele două maxime, iar $\boldsymbol{H}[m_1], \boldsymbol{H}[m_2]$ sunt aceste valori maxime, atunci pragul de segmentare este dat de:

$$T = \frac{m_1 + m_2}{2} \qquad \text{sau} \qquad T = \frac{m_2\boldsymbol{H}[m_2] - m_1\boldsymbol{H}[m_1]}{\boldsymbol{H}[m_2] - \boldsymbol{H}[m_1]} \tag{5}$$

Această din urmă strategie este mai lentă, dar dă rezultate mai bune pentru marea majoritate a eşantioanelor de site folosite. Ea poate localiza cu precizie cele două maxime principale din histogramă şi pragul de segmentare.

Dacă timpul de răspuns al sistemului nu este critic, atunci este mai simplu de implementat o procedură interactivă de segmentare.

Folosind pragul de segmentare găsit anterior, segmentarea prin binarizare a imaginii originale $\boldsymbol{f}(x,y)$ este realizată (fig. 5) prin:

$$\boldsymbol{f}_S(x,y) = \begin{cases} 0, & \text{pentru} \quad f(x,y) < T \\ 1, & \text{pentru} \quad f(x,y) \geq T \end{cases} \tag{6}$$

Aceasta etapă de prelucrare poate eşua dacă eşantioanele de sită nu sunt iluminate uniform, cel mai adesea ca urmare a plasării incorecte a surselor de lumină. Această situaţie poate fo eliminată prin software, adică prin includerea următorilor paşi de prelucrare:

- Achiziţionarea unei imagini $\boldsymbol{b}(x,y)$ a fondului folosit (fără eşantion de sită pe el).
- Eliminarea zgomotelor din imaginea $\boldsymbol{b}(x,y)$ prin aplicarea unui filtru (ex. median) clasic.
- Aplicarea următoarei transformări punctuale pentru fiecare pixel al imaginilor de analiză:

$$\boldsymbol{f}_C(x,y) = \boldsymbol{f}(x,y) - \boldsymbol{b}(x,y) + \max_{x,y} |\boldsymbol{f}(x,y) - \boldsymbol{b}(x,y)| \tag{7}$$

18.4. Etichetarea componentelor.

Imaginile binare obţinute în urma prelucrărilor anterioare, $\boldsymbol{f}_S(x,y)$ sunt folosite pentru a localiza cu precizie ochiurile sitei. Folosind un algoritm de urmărire/extragere de contur (Pratt[84]), (Jain[55]) (fig. 6) toate componentele)obiectele din imagine sunt localizate şi pentru fiecare dintre ele se salvează într-o listă area ocupată (A_i), coordonatele punctului de start (x_S, y_S) ale conturului, şi coordonatele dreptunghiului de încadrare, $(X^{(i)}_{min}, Y^{(i)}_{min}, X^{(i)}_{max}, Y^{(i)}_{max})$. Toate aceste informaţii vor fi utilizate în următoarele etape de prelucrare.

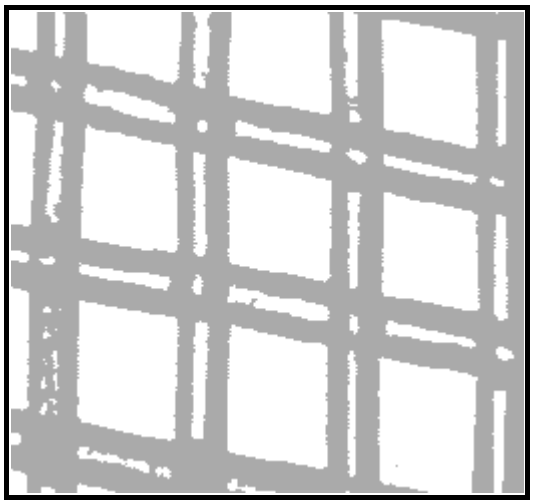


Figura 5. Imaginea segmentată.

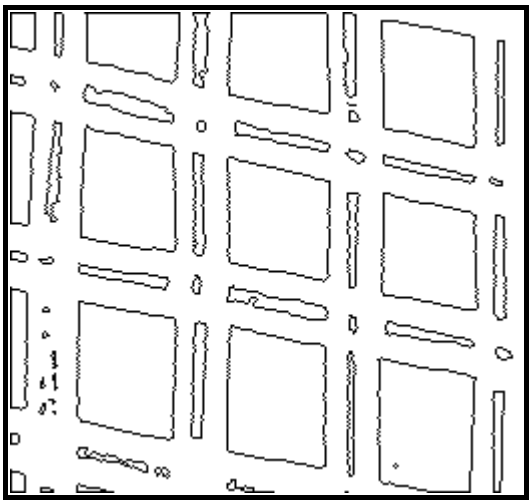


Figura 6. Contururi extrase.

Conform figurilor precedente, există multe obiecte localizate care nu sunt ochiuri ale sitei, şi deci ele trebuiesc eliminate din lista de obiecte (Duda[36]). Pentru efectuarea acestor operaţii se defineşte un "filtru logic euristic " care verifică succesiv următoarele reguli pentru fiecare obiect:

1. Conturul unui ochi de sită *nu este inclus* în nici un alt contur.

2. Conturul (complet) al unui ochi de sită *nu atinge* limitele imaginii (el ar putea fi incomplet, deci trebuie ignorat).

3. Raportul laturilor orizontală şi verticală ale dreptunghiului de încadrare pentru fiecare ochi de sită localizat *trebuie să îndeplinească* următoarea condiţie:

$$1/K_1 < \Delta x/\Delta y < K_1 \tag{8}$$

4. Aria unui ochi de sită trebuie să fie peste un nivel minim (zgomotul tip "sare şi piper" într-o imagine poate da asemenea obiecte mici):

$$A_i > A_{min} \tag{9}$$

5. Raportul dintre aria unui ochi de sită şi aria celui mai mare obiect din imagine trebuie să fie peste un anumit nivel minim:

$$A_i / \max_j (A_j) > K_2 \tag{10}$$

Acele obiecte care nu îndeplinesc *toate* regulile anterioare vor fi şterse din lista ochiurilor sitei (fig. 7).

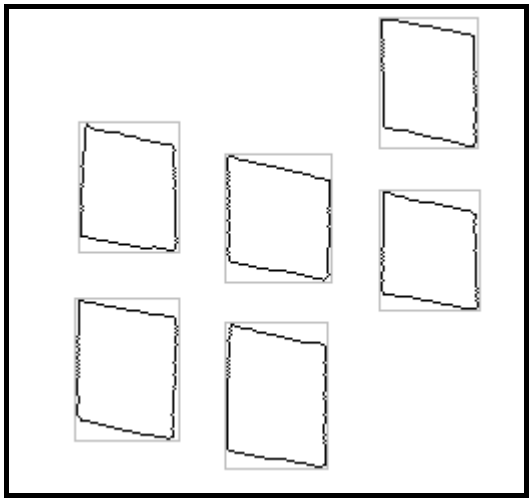


Fig. 7. Contururi de ochiuri selectate.

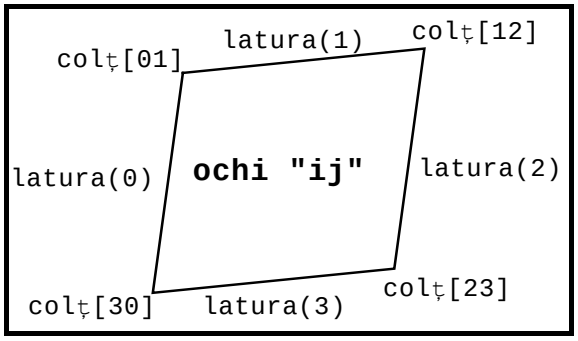


Fig. 8. Notaţii.

18.5 Aproximarea poligonală a ochiurilor sitei.

Pentru a efectua măsurătorile dorite asupra eşantionului de sită, trebuie să localizăm cât mai precis posibil *cele patru colţuri* ale fiecărui ochi de sită. Laturile fiecărui ochi sunt apoximate prin segmente de dreaptă, aşa încât avem de-a face cu o problemă de aproximare poligonală (fitting) (Jain[56]), (Pavlidis[80]) care trebuie rezolvată (fig. 8).

Soluţia dezvoltată se bazează pe minimizarea iterativă a erorii de aproximare (eroarea medie pătratică). Pentru că numărul laturilor poligonului de aproximare este stabilit la patru, aproximarea iniţială a ochiului sitei poate fi setată la dreptunghiul său de încadrare.

Punctele de contur corespunzătoare fiecărei laturi a poligonului (patrulaterului) se stabilesc folosind un criteriu de distanţă minimă, adică lista punctelor de contur este partiţionată in patru grupe, fiecare corespunzător unei laturi, fiecare punct al conturului fiind ataşat celui mai apropiat segment aproximant.

Dacă exisă în total N puncte de contur pentru un obiect (fig. 9) şi partiţiile punctelor de contur conţin respectiv N_0, N_1, N_2 şi N_3 elemente, atunci, desigur:

$$N = N_0 + N_1 + N_2 + N_3 \tag{11}$$

Ecuaţia dreptei care trece prin punctul (x_0,y_0) şi are panta dată de φ_0 este:

$$(x - x_0)\sin\varphi_0 - (y - y_0)\cos\varphi_0 = 0 \tag{12}$$

Ecuaţia dreptei care trece prin punctul (x_k,y_k) şi este perpendiculară pe dreapta (12) este:

$$(x - x_k)\cos\varphi_0 + (y - y_k)\sin\varphi_0 = 0 \tag{13}$$

Atunci punctul de intersecţie a acestor două drepte are coordonatele următoare:

$$x = x_0 \sin^2\varphi_0 + x_k \cos^2\varphi_0 - (y_0 - y_k)\sin\varphi_0\cos\varphi_0 \tag{14}$$

$$y = y_0 \cos^2\varphi_0 + y_k \sin^2\varphi_0 - (x_0 - x_k)\sin\varphi_0\cos\varphi_0 \tag{15}$$

Distanţa de la punctul (x_k,y_k) la linia care trece prin punctul (x_0,y_0) şi are panta φ_0 este dată de:

$$d_k^2 = (x - x_k)^2 + (y - y_k)^2 \tag{16}$$

adică:

$$d_k^2 = [(x_0 - x_k)\sin^2\varphi_0 - (y_0 - y_k)\sin\varphi_0\cos\varphi_0]^2 + [(y_0 - y_k)\cos^2\varphi_0 - (x_0 - x_k)\sin\varphi_0\cos\varphi_0]^2$$

(17)

Atunci eroarea medie pătratică corespunzătoare fiecărei partiții a punctelor de contur este:

$$E_0 = \frac{1}{N_0}\sum_{k_0=1}^{N_0} d_{k_0}^2 \qquad E_1 = \frac{1}{N_1}\sum_{k_1=1}^{N_1} d_{k_1}^2 \qquad E_2 = \frac{1}{N_2}\sum_{k_2=1}^{N_2} d_{k_2}^2 \qquad E_3 = \frac{1}{N_3}\sum_{k_3=1}^{N_3} d_{k_3}^2 \tag{18}$$

Pentru prima partiție, eroarea medie pătratică poate fi exprimată prin:

$$E_0 = \frac{1}{N_0}\sum_{k_0=1}^{N_0}\{[(x_0 - x_{k_0})\sin^2\varphi_0 - (y_0 - y_{k_0})\sin\varphi_0\cos\varphi_0]^2 + [(y_0 - y_{k_0})\cos^2\varphi_0 - (x_0 - x_{k_0})\sin\varphi_0\cos\varphi_0]^2\} \tag{19}$$

și dacă definim următoarele cantități (tot pentru prima partiție):

$$\overline{x_{(0)}} = \frac{1}{N_0}\sum_{k_0=1}^{N_0} x_{k_0}\ ;\ \overline{y_{(0)}} = \frac{1}{N}\sum_{k_0=1}^{N_0} y_{k_0}\ ;\ \overline{xy_{(0)}} = \frac{1}{N_0}\sum_{k_0=1}^{N_0} x_{k_0}y_{k_0}\ ;\ \overline{x_{(0)}^2} = \frac{1}{N_0}\sum_{k_0=1}^{N_0} x_{k_0}^2\ ;\ \overline{y_{(0)}^2} = \frac{1}{N_0}\sum_{k_0=1}^{N_0} y_{(0)}^2$$

(20)

se obține următoarea expresie pentru eroarea medie pătratică E_0:

$$E_0 = (x_0^2 - 2x_0\overline{x_{(0)}} + \overline{x_{(0)}^2})\sin^2\varphi_0 - 2(x_0y_0 - x_0\overline{y_{(0)}} - \overline{x_{(0)}}y_0 + \overline{xy_{(0)}})\sin\varphi_0\cos\varphi_0 + (y_0^2 - 2y_0\overline{y_{(0)}} + \overline{y_{(0)}^2})\cos^2\varphi_0 \tag{21}$$

Prin minimizarea acestei erori valorile necunoscute x_0, y_0 și φ_0 pot fi aflate:

$$\frac{\partial E}{\partial x_0} = 0 \Leftrightarrow (x_0 - \overline{x_{(0)}})\sin\varphi_0 - (y_0 - \overline{y_{(0)}})\cos\varphi_0 = 0 \Leftrightarrow x_0 = \overline{x_{(0)}} \tag{22}$$

$$\frac{\partial E}{\partial y_0} = 0 \Leftrightarrow -(x_0 - \overline{x_{(0)}})\sin\varphi_0 + (y_0 - \overline{y_{(0)}})\cos\varphi_0 = 0 \Leftrightarrow y_0 = \overline{y_{(0)}} \tag{23}$$

$$\frac{\partial E}{\partial \varphi_0} = 0 \Leftrightarrow \varphi_0 = \frac{1}{2}\arctan\frac{2(\overline{xy_{(0)}} - \overline{x_{(0)}}\,\overline{y_{(0)}})}{\overline{x_{(0)}^2} - (\overline{x_{(0)}})^2 - \overline{y_{(0)}^2} + (\overline{y_{(0)}})^2} \tag{24}$$

Formule similare pot fi scrise pentru toate partițiile atașate conturului ochiului sitei, astfel încât toate seturile $[(x_0,y_0),\varphi_0]$, $[(x_1,y_1),\varphi_1]$, $[(x_2,y_2),\varphi_2]$, și $[(x_3,y_3),\varphi_3]$ pot fi obținute.

Rezolvând sistemul de ecuații:

$$\begin{cases}(x - x_0)\sin\varphi_0 - (y - y_0)\cos\varphi_0 = 0\\ (x - x_1)\sin\varphi_1 - (y - y_1)\cos\varphi_1 = 0\end{cases} \tag{25}$$

coordonatele primului colț al ochiului se calculează cu:

$$x_{01} = \frac{(x_0\sin\varphi_0 - y_0\cos\varphi_0)\cos\varphi_1 - (x_1\sin\varphi_1 - y_1\cos\varphi_1)\cos\varphi_0}{\sin(\varphi_0 - \varphi_1)} \tag{26}$$

$$y_{01} = \frac{(x_0\sin\varphi_0 - y_0\cos\varphi_0)\sin\varphi_1 - (x_1\sin\varphi_1 - y_1\cos\varphi_1)\sin\varphi_0}{\sin(\varphi_0 - \varphi_1)} \tag{27}$$

și expresii similare pot fi găsite pentru celelalte trei colțuri ale poligonului aproximant, adică pentru (x_{12},y_{12}), (x_{23},y_{23}) și (x_{30},y_{30}).

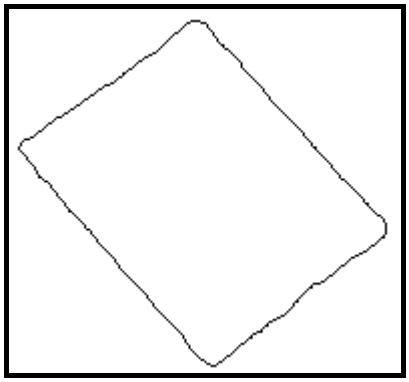


Fig. 9. Conturul inițial.

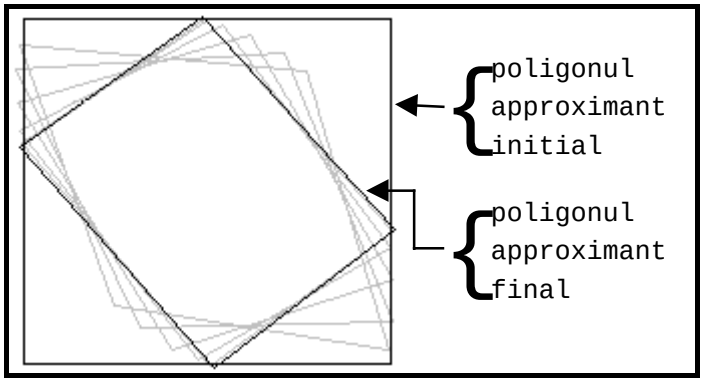


Fig. 10. Procesul de aproximare.

Acest nou poligon este folosit mai departe pentru a construi alte patru partiții ale punctelor de contur ale ochiului sitei, pe baza cărora se va obține o mai bună aproximare a ochiului. Acest proces continuă (fig. 10) până când eroarea globală relativă de aproximare îndeplineşte condiția:

$$\varepsilon = \frac{E^{(t)} - E^{(t+1)}}{E^{(t)}} < \varepsilon_0 \tag{28}$$

unde $E^{(t)} = E_0^{(t)} + E_1^{(t)} + E_2^{(t)} + E_3^{(t)}$ este eroarea globală pentru iterația "t".

18.6. Măsurători asupra sitei.

Dacă orientările corespunzătoare firelor de urzeală şi respectiv bătătură se calculează cu:

$$\overline{\varphi}_{02} = \frac{1}{2M}\sum_i\sum_j(\varphi_0^{(ij)} + \varphi_2^{(ij)}) \qquad \overline{\varphi}_{13} = \frac{1}{2M}\sum_i\sum_j(\varphi_1^{(ij)} + \varphi_3^{(ij)}) \tag{29}$$

atunci unghiul mediu între firele de urzeală şi cele de bătătură este:

$$\overline{\varphi} = \frac{1}{M}\sum_i\sum_j\varphi^{(ij)} \tag{30}$$

unde

$$\varphi^{(ij)} = \frac{1}{2}(\varphi_0^{(ij)} + \varphi_2^{(ij)} - \varphi_1^{(ij)} - \varphi_3^{(ij)}) \tag{31}$$

Unghiurile $\overline{\varphi}_{02}$ şi $\overline{\varphi}_{12}$ sunt apoi folosite pentru etichetarea matriceală a ochiurilor sitei, folosind teste de poziție relativă în lungul acestor două direcții (fig. 11).

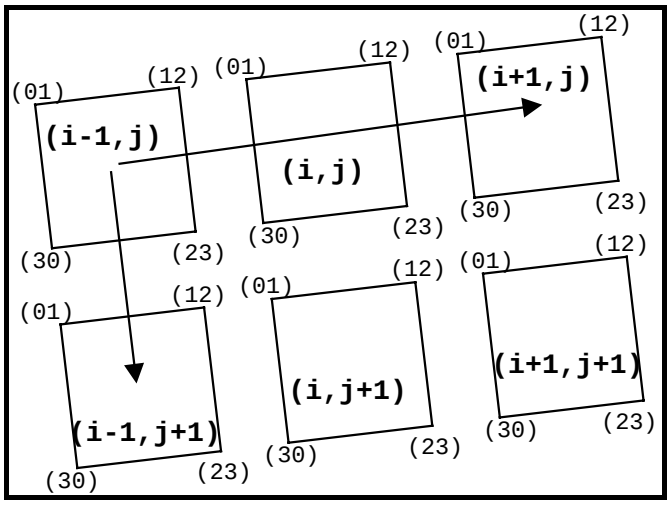


Fig. 11. Direcțiile de căutare pentru etichetare.

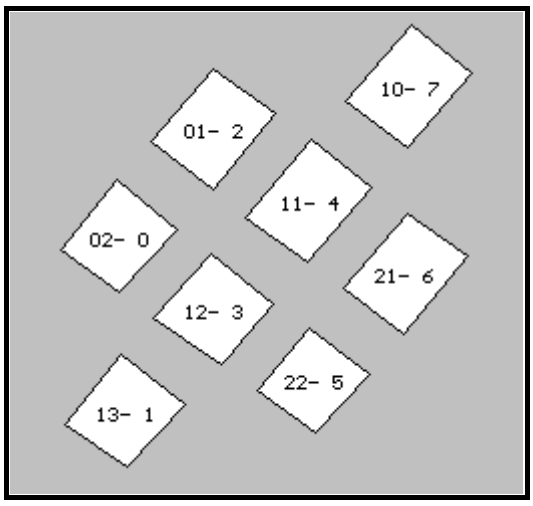


Fig. 12. Ochiuri etichetate.

Această etichetare (fig. 12) este utilizată mai departe pentru a stabili perechile valide de ochiuri vecine în lungul ambelor direcții.

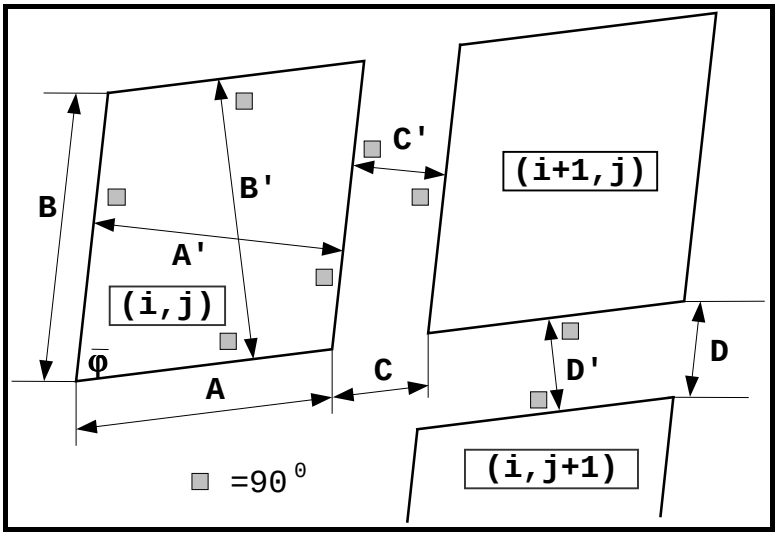


Fig. 13. Principalele măsurători asupra eşantionului de sită.

Valorile distanțelor marcate în figura (13) se calculează folosind formulele următoare:

$$A_{(ij)} = \frac{1}{2}\left[\sqrt{(x_{01}^{(ij)} - x_{12}^{(ij)})^2 + (y_{01}^{(ij)} - y_{12}^{(ij)})^2} + \sqrt{(x_{23}^{(ij)} - x_{30}^{(ij)})^2 + (y_{23}^{(ij)} - y_{30}^{(ij)})^2}\right] \quad (32)$$

$$B_{(ij)} = \frac{1}{2}\left[\sqrt{(x_{01}^{(ij)} - x_{30}^{(ij)})^2 + (y_{01}^{(ij)} - y_{30}^{(ij)})^2} + \sqrt{(x_{12}^{(ij)} - x_{23}^{(ij)})^2 + (y_{12}^{(ij)} - y_{23}^{(ij)})^2}\right] \quad (33)$$

$$C_{(ij)} = \frac{1}{2}\left[\sqrt{(x_{12}^{(ij)} - x_{01}^{(i+1,j)})^2 + (y_{12}^{(ij)} - y_{01}^{(i+1,j)})^2} + \sqrt{(x_{23}^{(ij)} - x_{30}^{(i+1,j)})^2 + (y_{23}^{(ij)} - y_{30}^{(i+1,j)})^2}\right] \quad (34)$$

$$D_{(ij)} = \frac{1}{2}\left[\sqrt{(x_{30}^{(ij)} - x_{01}^{(i,j+1)})^2 + (y_{30}^{(ij)} - y_{01}^{(i,j+1)})^2} + \sqrt{(x_{23}^{(ij)} - x_{12}^{(i,j+1)})^2 + (y_{23}^{(ij)} - y_{12}^{(i,j+1)})^2}\right] \quad (35)$$

$$A'_{(i,j)} = A_{(ij)} \sin\varphi^{(ij)} \qquad B'_{(i,j)} = B_{(ij)} \sin\varphi^{(ij)} \qquad C'_{(i,j)} = C_{(ij)} \sin\varphi^{(ij)} \qquad D'_{(i,j)} = D_{(ij)} \sin\varphi^{(ij)} \quad (36)$$

unde:

A = lungimea laturii ochiului corespunzătoare firelor de urzeală;
B = lungimea laturii ochiului corespunzătoare firelor de bătătură;
A' = distanța între firele de urzeală;
B' = distanța între firele de bătătură;
C' = diametrul firelor de urzeală;
D' = diametrul firelor de bătătură.

Valorile lor medii, corespunzătoare tuturor ochiurilor de sită din imagine sau mai bine, conform regulilor impuse de controlul de calitate, *pentru un set de imagini ale aceluiaşi sortiment de sită textilă* sunt:

$$\overline{A} = \frac{1}{M}\sum_i\sum_j A_{(i,j)} \qquad \overline{B} = \frac{1}{M}\sum_i\sum_j B_{(i,j)} \qquad \overline{C} = \frac{1}{M_H}\sum_i\sum_j C_{(i,j)} \qquad \overline{D} = \frac{1}{M_V}\sum_i\sum_j D_{(i,j)}$$

$$\overline{A}' = \frac{1}{M}\sum_i\sum_j A'_{(i,j)} \qquad \overline{B}' = \frac{1}{M}\sum_i\sum_j B'_{(i,j)} \qquad \overline{C}' = \frac{1}{M_H}\sum_i\sum_j C'_{(i,j)} \qquad \overline{D}' = \frac{1}{M_V}\sum_i\sum_j D'_{(i,j)} \quad (37)$$

unde
- M este numărul total de ochiuri;
- M_H este numărul de ochiuri vecine în lungul direcției $\overline{\varphi}_{13}$.
- M_V este numărul de ochiuri vecine în lungul direcției $\overline{\varphi}_{02}$.

O bună aproximare a ultimelor patru valori din (37) este dată de:

$$\overline{A'} = \overline{A}\,\sin\overline{\varphi}\,; \qquad \overline{B'} = B_{(ij)}\,\sin\overline{\varphi}\,; \qquad \overline{C'} = C\,\sin\overline{\varphi}\,; \qquad \overline{D'} = D\,\sin\overline{\varphi} \qquad (38)$$

O descriere statistică completă a acestor măsurători implică cel puțin calculul dispersiilor lor:

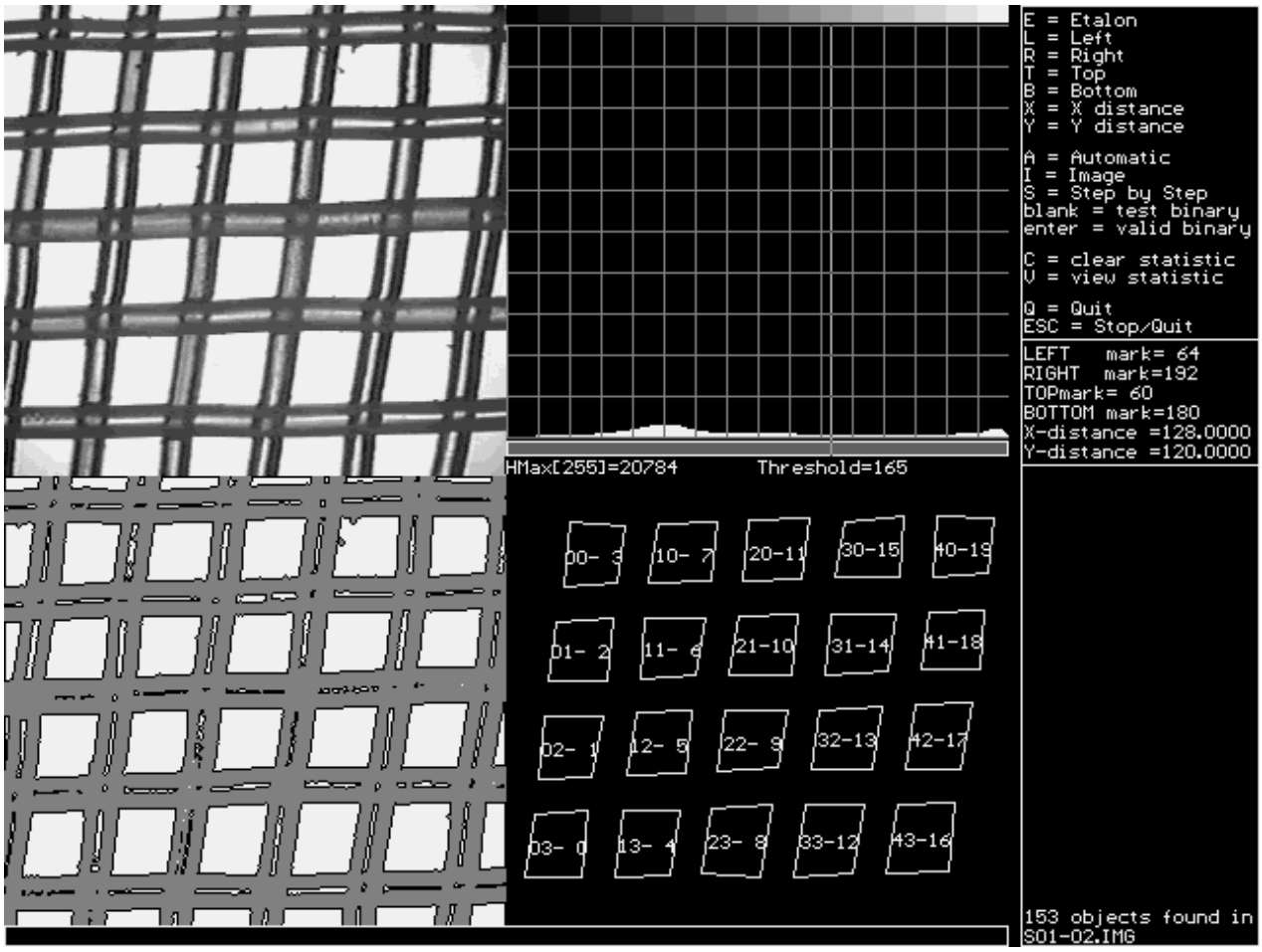
$$\delta_A^2 = \frac{1}{M}\sum_i\sum_j (A_{(ij)} - \overline{A})^2 \text{ , etc.} \qquad (39)$$

18.7. Rezultate experimentale. Concluzii.

Un mediu software integrat a fost dezvoltat pentru a realiza o analiză completă a eşantioanelor de site textile. El include şi opțiuni software de calibrare, care pot fi folosite pentru a converti toate distanțele în milimetri, cu ajutorul imaginilor unor rigle de calibrare.

Toate valorile calculate (buletine de analiză) sunt afişate pe ecran şi/sau pot fi salvate pe unitățile de disc ale calculatorului.

S-a folosit pentru această aplicație un calculator cu o interfață de achiziție a imaginilor şi o cameră TV standard conectată la un microscop. Toate programele au fost scrise în limbajul C++.



Acuratețea măsurătorilor este foarte bună şi este afectată doar de erorile de digitizare şi de cele datorate aproximării poligonale a ochiurilor sitei.

Pentru un prag de eroare de $\varepsilon_0 = 0.01$, numărul maxim de iterații necesar pentru aproximarea poligonală este de *5-7* şi depinde de orientarea inițială a firelor. Dacă laturile ochiurilor sunt aproape orizontale/verticale, numărul necesar de iterații scade la *2-3*. Am folosit în experimente următoarele valori pentru constantele descrise anterior:

$$K_1 = 4\,,\ A_{min} = 10 \text{ şi } K_2 = 0.2\,. \qquad (40)$$

Deşi unele optimizări de viteză încă mai pot fi făcute, timpul de prelucrare obținut pentru o imagine 256x240 în scară de gri este de 0.4sec pe un calculator Pentium 400Mhz şi creşte pentru imagini mai mari (1sec pentru o imagine 640x480 în scară de gri).

Rezultatele obținute demonstrează acuratețea acestei metode de măsurare, ca şi posibilitatea de a detecta multe din defectele de țesere posibile. Echipamentul necesar pentru a implementa această metodă nu este foarte costisitor şi poate fi adaptat cu uşurință la o instalație de producție pentru testarea on-line a sitelor textile.

Aceeaşi metodă a fost testată cu succes şi pentru site metalice şi poate fi extinsă la alte sortimente de materiale țesute.

Programul executabil (SitaApp.exe) se găseşte împreună cu codul sursă pe CD-ul ataşat şi poate fi rulat după instalare.

19. Recunoaşterea simbolurilor muzicale tipărite.

Recunoaşterea automată a simbolurilor creşte spactaculos eficiența interfețelor om-maşină. Dar nu întotdeauna "simbol" înseamnă caracter, el poate însemna de asemenea simbol muzical. Se constată un interes în creştere pentru recunoaşterea simbolurilor muzicale, datorat în mare măsură multiplelor aplicații posibile (Sicard[101]).

În continuare este prezentată o soluție robustă adoptată pentru recunoaşterea unor simboluri muzicale tipărite, folosită în cadrul unei aplicații de recunoaştere a partiturilor muzicale. Este descris modul de construcție a setului de simboluri de antrenament/test, algoritmul de pregătire a datelor pentru recunoaştere (extragerea trăsăturilor) şi o procedură nouă de adaptare a setului de date de antrenament.

19.1. Descrierea aplicației generale.

S-a urmărit construcția unui sistem complet de recunoaştere a partiturilor muzicale.

Partitura tipărită este digitizată folosind un scanner, după care procedura generală de recunoaştere parcurge următorii paşi:

- *separare semiautomată portative*, necesară deoarece există simboluri mai apropiate de un portativ, dar aparținând logic altuia;
- pentru fiecare portativ, *extragere linii portativ*, cu distorsionarea minima a simbolurilor muzicale;
- *localizare obiecte* prin încadrare în fereastra dreptunghiulară; algoritmi dedicați trebuiesc utilizați pentru simbolurile fragmentate sau atinse.

Informația din dreptunghiul de încadrare este utilizată pentru recunoaştere, într-un sistem cu schema bloc din fig.1 :

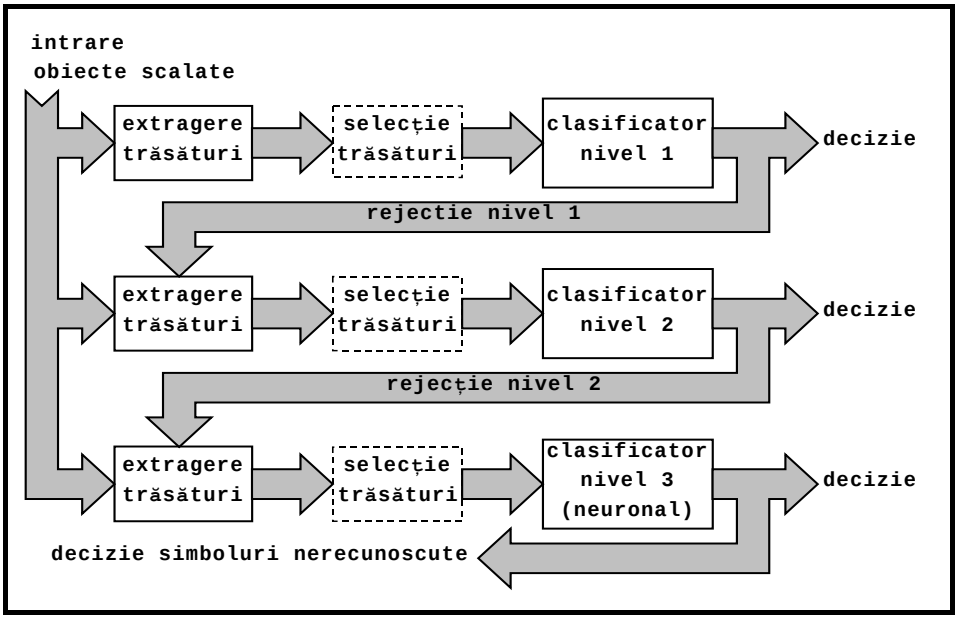


Fig. 1: Schema bloc a sistemului de recunoaştere a partiturilor.

Preclasificarea împarte obiectele în clase. Pentru fiecare clasă se utilizează metode specifice de clasificare, urmărindu-se, evident, mărirea ratei de recunoaştere la un timp de clasificare cât mai mic. A utiliza aceeaşi metodă în toate nivelurile de clasificare de mai sus este o soluţie greşită: sau obiectele pot fi separate prin metode specifice foarte simple, fie numai metode sintactice duc la performanţele dorite.

Deoarece aplicaţia impune invarianţa la scalare se lucrează în general cu mărimea normalizată a obiectelor, adică (fig.2):

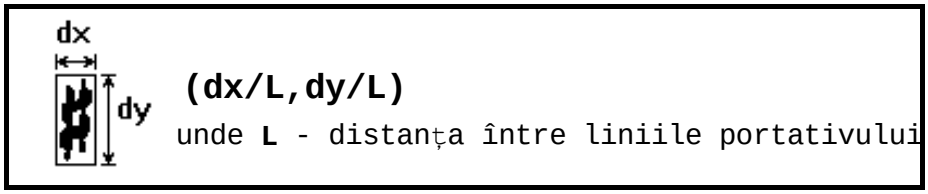


Fig. 2. Normalizarea obiectelor.

Capitolul curent prezintă modul de implementare a unui clasificator (nivelul 3, fig.1) pentru recunoaşterea a şase simboluri muzicale: *diez*, *bemol*, *becar*, *pauza 1/4*, *pauza 1/16* şi *pauza 1/8*, pentru care alte metode testate au dat rezultate mult mai slabe.

Aspectul obiectelor care urmează a fi recunoscute depinde de calitatea partiturii originale, performanţele scanner-ului utilizat în digitizare şi de efectele algoritmului de eliminare a liniilor portativului. Oricum, rezultă o mare variabilitate a formei simbolurilor din aceeaşi clasă (fig 3):

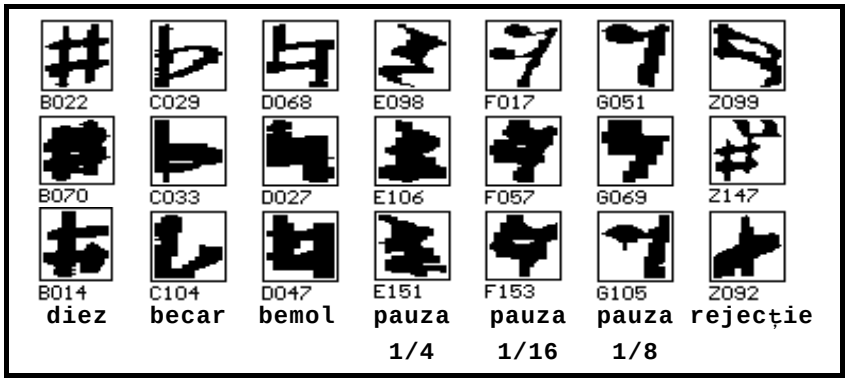


Fig. 3. Clasele de simboluri ce trebuiesc recunoscute.

Se remarcă utilizarea unei a 7-a clase de obiecte, "clasa de rejecţie" (Pudil[88]), care permite obţinerea unor rezultate net superioare. Din punct de vedere teoretic, introducerea ei este justificată de faptul că în lipsa clasei de rejecţie, spaţiul trăsăturilor este partiţionat în şase clase, astfel încât obiectele care nu aparţin la nici una din cele şase clase pot fi uşor atribuite lor.

Construind clasa de rejecţie, care include cifrele, literele, fragmente de simboluri şi obiecte concatenate, limita de decizie între clasa de rejecţie şi cele 6 clase devine bine precizată.

19.2. Crearea setului de date de antrenament / test.

Pentru crearea seturilor de date de antrenament şi test se folosesc partituri tipice din punctul de vedere al fonturilor muzicale folosite. Simbolurile extrase sunt salvate sub formă de fişiere al căror nume indică apartenenţa la clasa corespunzătoare.

Simbolurile care trec de preclasificator şi sunt diferite de cele 6 simboluri studiate se atribuie clasei de rejecţie. Ele reprezintă cifre, litere, alte simboluri muzicale, fragmente de simboluri, simboluri concatenate.

Astfel s-au creat:
- *337* forme de tipul "*diez*";
- *171* forme de tipul "*bemol*";
- *223* forme de tipul "*becar*";
- *360* forme de tipul "*pauză 1/4*";
- *239* forme de tipul "*pauză 1/16*";
- *351* forme de tipul "*pauză 1/8*";
- *191* forme de tipul "*rejecţie*";

Deci s-au folosit *1872* simboluri: setul de antrenament conţine primele *120* simboluri din fiecare clasă (plus clasa de rejecţie), deci în total *840* simboluri de antrenament şi *1032* simboluri în setul de test. De menţionat faptul că, deoarece simbolurile din setul de antrenament au fost alese aleator, există multe simboluri similare. Pentru selectarea în setul de antrenament a celor mai semnificative simboluri (fără a creşte dimensiunea acestui set) s-a dezvoltat un algoritm de gestiune a seturilor de antrenament, descris în paragraful 7.

19.3. Sistemul de recunoaştere.

Recunoaşterea celor 6 simboluri muzicale a fost implementată (Duda[35]), (Meisel[72]) pe baza următoarei scheme bloc generale (fig.4):

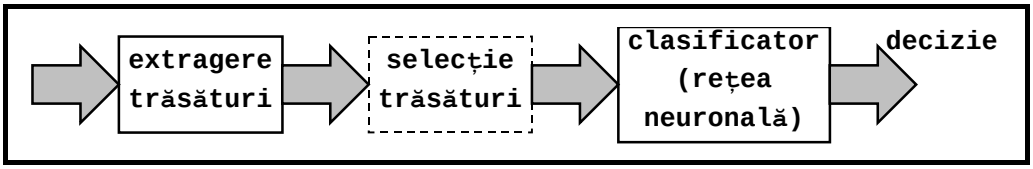


Fig. 4. Schema bloc a clasificatorului.

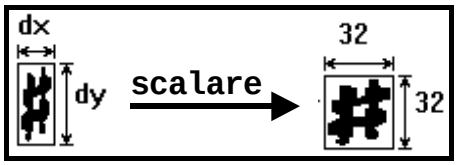
În proiectarea acestui clasificator s-a urmărit, pe lângă optimizarea timpului de răspuns (cât mai mic) şi a ratei de recunoaştere (cât mai mare), şi posibilitatea unei implementări hardware, într-o structură dedicată de mare viteză, care să poată rezolva probleme asemănătoare (dar nu identice).

Selectorul de trăsături poate reţine cele mai semnificative caracteristici furnizate de extractorul de trăsături. Decizia clasificatorului este asimilată cu activarea specifică a ieşirilor reţelei neurale.

19.4. Extragerea trăsăturilor.

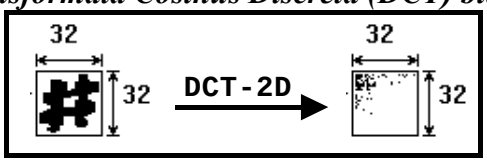
Faza de extragere a trăsăturilor presupune parcurgerea următorilor paşi:

a) Scalarea simbolurilor:



Se obţine o matrice binară de dimensiune constantă (32x32), conţinând imaginea scalată. Astfel este rezolvată invarianţa la scalare pe orizontală şi verticală, impusă de aplicaţie.

b) Transformata Cosinus Discretă (DCT) bidimensională:



Se obţine o matrice de valori reale de aceeaşi dimensiune ca cea iniţială. Se foloseşte DCT deoarece proprietăţile ei o fac utilă în selecţia trăsăturilor (Rosenfeld[95]), (Ahmed[03]).

Un selector optimal de trăsături se construieşte astfel (fig.5):

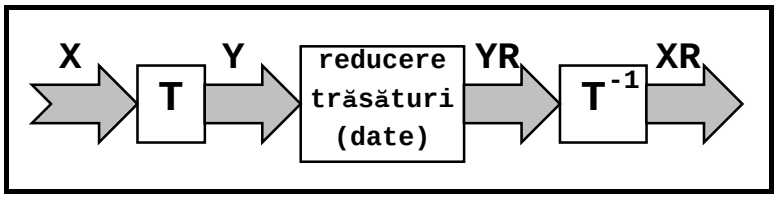


Fig. 5. Construcţia selectorului optimal de trăsături.

unde:

$\boldsymbol{X} = (x_1, x_2, ..., x_N)$ - vectorul de intrare;

$\boldsymbol{Y} = (y_1, y_2, ..., y_N)$ - vectorul transformat;

$\boldsymbol{YR} = (y_1, y_2, ..., y_k, c_{k+1}, c_{k+2}, ..., c_N)$ - vectorul de trăsături redus; el se obţine din $\boldsymbol{Y}$ înlocuind cu constante ultimele (N-K) valori;

$\boldsymbol{XR} = (xr_1, xr_2, ..., xr_N)$ - estimaţia vectorului de intrare.

$\boldsymbol{T} = \left\| \boldsymbol{u}_1^T, \boldsymbol{u}_2^T, ..., \boldsymbol{u}_N^T \right\|^T$, cu $\boldsymbol{u}_j^T = (u_{j0}, u_{j1}, ..., u_{jN})$ este matricea unei transformări ortogonale unitare care se va determina în condiţiile minimizării erorii medii pătratice date de:

$$e = E\left\{ (\boldsymbol{X} - \overline{\boldsymbol{X}})^T \times (\boldsymbol{X} - \overline{\boldsymbol{X}}) \right\}$$

Notând:

$\boldsymbol{K}_{XX} = E\{(\boldsymbol{X} - \overline{\boldsymbol{X}})(\boldsymbol{X} - \overline{\boldsymbol{X}})^T\}$, matricea de covarianţă a intrării, unde $\overline{\boldsymbol{X}} = E\{\boldsymbol{X}\}$, se obţine transformata Karhunen-Loeve discretă (sau Hotelling) cu proprietăţile:

1. $(\boldsymbol{K}_{XX} - \lambda_j \times \boldsymbol{I}_N) \times \boldsymbol{u}_j = 0$ - liniile matricii transformării sunt valorile proprii ale matricii de covarianţă a domeniului de intrare;
2. $\boldsymbol{K}_{YY} = \text{diag}(\lambda_1, \lambda_2, ..., \lambda_N)$ - transformarea furnizează la ieşire coeficienţi necorelaţi;
3. $E = \sum_{j=k+1}^{N} \lambda_j$ - expresia erorii.

Deoarece implementarea transformării (KL) este dificilă şi în plus, modelând intrarea {X} cu un proces Markov de ordinul 1, cea mai bună aproximare a transformării Karhunen-Loeve

discrete este DCT (***Transformata Cosinus Discretă***). Această modelare nu este foarte restrictiva şi se poate folosi cu succes în multe aplicaţii (Ahmed[03]).

Pentru calculul DCT există nu numai algoritmi rapizi de calcul, ci şi circuite specializate de mare viteză..

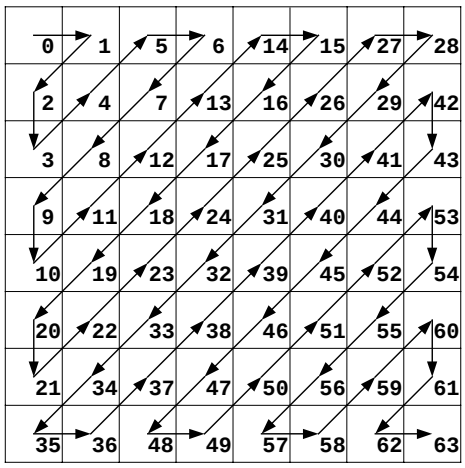
Pentru cazul bidimensional se foloseşte:

$$F(u,v) = \frac{1}{2N} \sum_{j=0}^{N-1} \sum_{i=0}^{N-1} f(j,k)\ cos(\frac{\pi u j}{N})\ cos(\frac{\pi v k}{N})$$

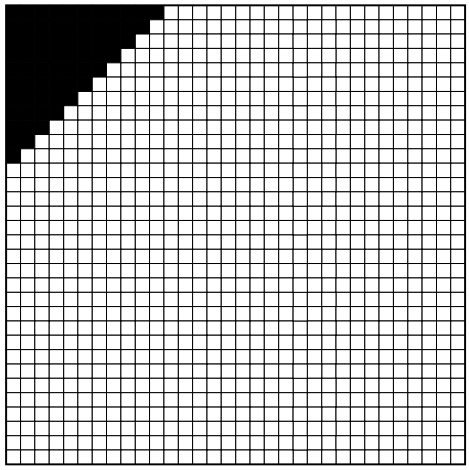
rezultând o matrice de aceeaşi dimensiune, cu valori reale, slab corelate. Valorile semnificative ale acestei matrici sunt plasate în colţul din stânga sus.

c) Ordonarea Zig - Zag.

Este o procedură de vectorizare a matricii DCT, după regula din figura de mai jos:



Matrice[8][8] ==> Vector[64];



Matrice[32][32] ==> Vector[1024];

Deoarece DCT bidimensională concentrează informaţia din imaginea iniţială în colţul din stînga sus al matricii rezultate, după ordonarea Zig - Zag aceeaşi informaţie va fi regăsită în primele elemente ale vectorului obţinut.

19.5. Selecţia trăsăturilor.

Pentru a minimiza timpul de clasificare/recunoaştere, cea mai bună soluţie este de a utiliza doar cele mai semnificative dintre trăsăturile extrase. În cazul nostru, trăsăturile cele mai semnificative se găsesc la începutul vectorului de trăsături. Ca urmare se reţin doar primele *66* valori din vectorul ***Z****[1024]*, adică *6.5%* din valori, cele marcate cu negru în figura anterioară.

Conform afirmaţiilor anterioare, aceste elemente concentrează cea mai mare parte din informaţia din imaginea iniţială şi ele vor fi folosite în faza de recunoaştere. În plus, deoarece restul vectorului Zig-Zag nu este folosit, valorile corespunzătoare din matricea DCT nici nu mai trebuiesc calculate, ceea ce reduce semnificativ timpul de calcul al DCT (doar 18.8% din multiplicări sunt necesare).

19.6. Reţeaua neurală.

Primele 66 elemente ale vectorului ***Z*** se aplică la intrarea unei reţele neurale de tipul perceptron cu trei straturi şi nelinearitate sigmoidală simetrică *(+/-1)* la toţi neuronii (Lipmann[66]), (Rumelhart[98]).

Număr de neuroni folosiţi este de *80 neuroni* pentru stratul de intrare, *20 neuroni* pentru stratul intermediar şi *6 neuroni* pentru stratul de ieşire.

Pentru antrenare s-a folosit evident algoritmul "backpropagation" cu factorul de câştig $\eta = 0.3$, factorul de inerţie $\alpha = 0.7$. Antrenarea s-a făcut prin aplicarea ciclică a trăsăturilor obiectelor din fiecare clasă la intrările reţelei neurale.

Fiecare neuron din stratul de ieşire este asignat unei clase, ca urmare ieşirea dorită pentru fiecare clasă înseamnă activarea neuronului corespunzător clasei respective şi inhibarea celorlalţi neuroni de ieşire. Pentru obiectele din clasa de rejecţie reţeaua neurală trebuie să răspundă cu toate ieşirile inactive.

Decizia de apartenenţă a unui obiect X la una din clase este de forma:

IF [y1>thr] and [y2<(-thr)] and [y3<(-thr)] and [y4<(-thr)] and [y5<(-thr] and [y6<(-thr)] THEN
Obiectul X aparţine clasei (1)

unde pragul este 0<thr<1. Pentru cazul aplicaţiei de faţă se pot scrie 6 asemenea condiţii, câte una pentru fiecare clasă. Pentru clasa de rejecţie condiţia este:

IF [y1<(-thr)] and [y2<(-thr)] and [y3<(-thr)] and [y4<(-thr)] and [y5<(-thr] and [y6<(-thr)] THEN
Obiectul X aparţine clasei de rejecţie

Dacă nici una din cele 6+1 condiţii nu este adevărată, decizia este de "obiect nerecunoscut". Un asemenea tip de condiţie va furniza, evident şi funcţie de prag, mai multe obiecte nerecunoscute decât greşit recunoscute, în concordanţă cu cerinţele aplicaţiei. De notat faptul că ***este mai avantajos cazul în care clasificatorul furnizează obiecte nerecunoscute decât obiecte recunoscute greşit deoarece corectarea interactivă a obiectelor recunoscute greşit implică o operaţie în plus (ştergerea simbolului greşit recunoscut) pe lângă înlocuirea lui.***

19.7. Adaptarea setului de date de antrenament.

În exploatarea sistemului pot apare noi tipuri de fonturi ce trebuiesc recunoscute. De aceea este necesară actualizarea setului de date de antrenament şi reantrenarea periodică a reţelei neurale folosite. Am impus condiţia de limitare a numărului de simboluri din setul de antrenament pentru limitarea spaţiului de memorie folosit şi a timpului necesar pentru reantrenarea reţelei.

Descrierea mai amănunţită a acestui gen de algoritmi se poate găsi în capitolul 16.

Pentru estimarea scorurilor de coincidenţă pentru perechi de obiecte din setul de antrenament, au fost testate mai multe măsuri:

$$\boldsymbol{S}^{(1)}{}_{KL} = \sum_{I=1}^{32}\sum_{J=1}^{32}\left|\boldsymbol{MS}^{(K)}[I,J] - \boldsymbol{MS}^{(L)}[I,J]\right| \qquad (1)$$

care reprezintă distanţa Hamming între obiectele "K" şi "L" (imagini binare scalate - *32x32*). Este foarte uşor de calculat, dar pentru obiecte foarte asemănătoare, un mic zgomot poate afecta mult încadrarea în fereastră, deci şi valoarea calculată pentru scorul de coincidenţă. O expresie mai corectă este:

$$\boldsymbol{S}^{(2)} = \min_{M,N}\left(\sum_{I=1}^{32}\sum_{J=1}^{32}(\boldsymbol{MS}^{(K)}[I,J] - \boldsymbol{ME}^{(L)}[I+M,J+N])^2\right) \qquad (2)$$

care calculează cel mai bun scor de coincidenţă între simbolul *(K)* şi simbolul *(L)* translat pe orizontală şi verticală. Deoarece prelucrările ulterioare asupra imaginii (DCT) nu sunt invariante la translaţii, (2) ar putea fi folosită în cazul utilizării unui neocognitron în faza de recunoaştere.

$$S^{(3)}_{KL} = \sum_{I=1}^{32}\sum_{J=1}^{32}(DCT^{(K)}[I,J] - DCT^{(L)}[I,J])^2 \quad (3)$$

reprezintă distanța euclidiană între obiectele "K" şi "L" descrise în spațiul transformatei DCT. Este mai greu de calculat, deoarece setul de antrenament este memorat pe calculator sub forma imaginilor binare scalate.

$$S^{(4)}_{KL} = \sum_{I=1}^{66}(Z^{(K)}[I] - Z^{(L))}[I])^2 \quad (4)$$

reprezintă distanța euclidiană între vectorii *"K"* şi *"L"* rezultați în urma ordonării Zig-Zag. Implică de asemenea un volum mare de calcule, dar se referă exact la valorile care se aplică intrării în clasificator. Pe de altă parte pare mai corectă o ponderare a valorilor $Z^{[K]}[I]$, funcție de contribuția lor la "forma" simbolului, dar nu se pot face decât presupuneri privind valorile acestor ponderi.

Testele efectuate au arătat că distanțele (1), (3), (4) dau rezultate similare, aşa că distanța (1) a fost aleasă datorită complexității reduse a calculelor

După o primă antrenare a rețelei, verificarea comportării ei pe setul de test furnizează un număr de simboluri nerecunoscute. La acestea se adaugă simbolurile nerecunoscute rezultate din exploatarea sistemului. Pe baza lor se schimbă succesiv structura bazei de date, reantrenând de fiecare dată rețeaua neurală folosită drept clasificator. Se îmbunătățeşte astfel rata de recunoaştere în condițiile menținerii dimensiunii setului de forme de antrenament.

Dacă după un număr predeterminat de paşi numărul de erori de recunoaştere detectate nu devine suficient de mic, rezultă că rețeaua neurală folosită nu poate "învăța" un număr atât de mare de simboluri, deci ar trebui adaugați neuroni pe primele două straturi ale rețelei neurale.

19.8. Rezultate obținute şi concluzii.

Timpul de recunoaştere obținut (localizare, scalare, DCT, rețea neurală) pentru un calculator Pentium 200MHz este sub 100msec/simbol.

Rata de recunoaştere obținută inițial pentru setul de antrenament/setul de test (*840* / *1032* simboluri) rezultă din tabelul următor.

Înainte de actualizare set antrenament:

Erori/Prag	0.1	0.2	0.3	0.4
Diez	0 / 1	0 / 0	0 / 0	0 / 0
Becar	0 / 0	0 / 0	0 / 0	0 / 0
Bemol	0 / 1	0 / 1	0 / 0	0 / 0
Pauză ¼	2 / 6	1 / 4	0 / 3	0 / 1
Pauză 1/16	0 / 0	0 / 0	0 / 0	0 / 0
Pauză 1/8	1 / 4	1 / 3	0 / 2	0 / 1
Rejecție	0 / 2	0 / 1	0 / 1	0 / 0
Total	3/ 14	2 / 9	0 / 6	0 / 2
Antr(%)	0.36%	0.24%	0%	0%
Test(%)	1.36%	0.87%	0.58%	0.19%

După adaptare set antrenament (3iterații):

Erori/Prag	0.1	0.2	0.3	0.4
Diez	0 / 0	0 / 0	0 / 0	0 / 0
Becar	0 / 0	0 / 0	0 / 0	0 / 0
Bemol	0 / 0	0 / 0	0 / 0	0 / 0
Pauză ¼	0 / 3	0 / 2	0 / 1	0 / 0
Pauză 1/16	0 / 0	0 / 0	0 / 0	0 / 0
Pauză 1/8	0 / 1	0 / 1	0 / 1	0 / 0
Rejecție	0 / 1	0 / 0	0 / 0	0 / 0
Total	0 / 5	0 / 3	0 / 2	0 / 0
Antr(%)	0%	0%	0%	0%
Test(%)	0.47%	0.28%	0.19%	0%

Rularea aplicației a furnizat într-o primă fază un număr de *42* simboluri nerecunoscute provenite de la un stil nou de partituri. Ele au fost adăugate setului de test (*1072* simboluri) şi a fost rulată procedura descrisă anterior.

Câteva luni de utilizare a sistemului de recunoaştere au confirmat robustețea lui şi eficacitatea algoritmului de adaptare a setului de antrenament. De asemenea a fost confirmată

ideea supradimensionării inițiale a rețelei neurale, tocmai în scopul de a putea învăța noi simboluri pe durata exploatării, folosind algoritmul de adaptare descris.

20. Recunoaşterea automată a textelor tipărite.

Studiile efectuate s-au concentrat în direcţia recunoaşterii caracterelor tipărite şi a caracterelor izolate scrise de mâna, deoarece pentru ambele cazuri pot fi folosite metode oarecum similare. Complexitatea mare a problemei e datorată nu atât numărului de caractere, cât variabilităţii mari a fonturilor utilizate în tipărirea documentelor, respectiv variabilităţii scrisului de mâna.

Scopul urmărit în această primă etapă este recunoaşterea textelor tipărite în condiţiile unor pagini de document suficient de bune calitativ, astfel încât să se evite, pe cât posibil, apariţia în text a caracterelor concatenate ("atinse") sau fragmentate ("sparte"). Deşi există soluţii şi pentru astfel de cazuri, ele nu formează obiectul studiului de faţă.

Fig. 1. Caractere concatenate şi caractere fragmentate

Un sistem complet de recunoaştere a documentelor tipărite ar trebui să execute următoarele operaţii:

1. ***Scanarea*** (achiziția) imaginii documentului de recunoscut. De regulă, se obține la ieşire un fişier, într-un format standard (**PCX**, **BMP**, **TIFF**, etc. în cazul conectării scanner-ului la un calculator compatibil IBM), care conține imaginea scanată.

2. ***Separarea zonelor de text*** de cele grafice din imagine. Paşii următori se referă doar la zonele de text astfel localizate. Unele implementări de sisteme de recunoaştere descrise în literatură propun strategii diferite de abordare, ceea ce se traduce prin realizarea separării zonelor de text după sau chiar în cadrul operației de localizare a obiectelor (pasul 3).

3. ***Localizarea obiectelor*** din imagine (sau din zona de text determinată) prin încadrarea lor într-o fereastră dreptunghiulară.

4. ***Ordonarea obiectelor***, funcție de poziția lor relativă, de la stânga la dreapta şi de sus în jos, pentru a le furniza clasificatorului în ordinea naturală a scrisului.

5. Se tratează ***cazurile particulare*** ale punctului precedent, adică simbolurile compuse, precum "**i**", "**j**", "**!**"(semnul exclamării), "**?**"(semnul întrebării), "**;**"(punct şi virgulă), "**:**"(două puncte), "**'**"(ghilimele), "**%**"(procent), la care se adaugă, în cazul limbii române, caracterele "**ă**", "**â**", "**î**", "**ş**", "**ț**", plus spațiile(caracterele "blank" şi "tab). Funcție de poziția relativă a acestor obiecte ele se concatenează pentru a forma, în dreptunghiul de încadrare, simbolul căutat.

Observație. Se poate adopta şi o altă strategie, mai puțin eficientă însă. Se renunță în această fază la punctul 5, încercându-se recunoaşterea ***obiectelor*** izolate: "**i fără punct**", "**j fără punct**", "**.**"(punct), "**,**"(virgulă), "/"(de la procent **%**), "**o**" (de la procent **%**), urmând ca apoi să se ia decizia definitivă după reguli simple de forma:

- "**.**"(punct) deasupra unui "**i fără punct**" ==> caracterul "**i**";
- "**o**" în stânga-sus față de obiectul "/" şi un alt "**o**" în dreapta-jos față de acelaşi obiect "/" ==> caracterul "**%**", etc.

Ca urmare, se poate utiliza o tehnică de *preclasificare* a obiectelor în "obiecte mici", incluzând "**.**"(punctul), "**,**"(virgula), etc. şi restul obiectelor. Cu reguli de genul celor de mai sus se face apoi asamblarea obiectelor în caractere. Dezavantajul apare în cazul caracterelor compuse, la care obiectele componente se ating (caz destul de des întâlnit), pentru care această a doua metodă nu mai funcționează.

6. ***Extragerea şi selecția trăsăturilor*** are drept scop obținerea celor mai semnificative trăsături ale fiecărui caracter. Ele trebuie să fie suficient de puține la număr pentru a scurta timpul de clasificare, dar şi suficient de multe pentru a caracteriza cât mai bine fiecare caracter.

7. ***Clasificarea*** propriu-zisă furnizează la ieşire decizia de apartenență a fiecărui obiect la una din clasele posibile.

8. Pasul care urmează în mod natural este un ***analizor lexical*** care să valideze informațiile furnizate de sistemul de recunoaştere.

20.1. Scanarea.

Scanarea unei pagini de text (document) se face la o rezoluție care să permită separarea corectă a obiectelor din pagină (dimensiunea unui pixel să fie mai mică decît grosimea oricărui obiect din pagină), fără să apară obiecte concatenate sau sparte din această cauză. Este de preferat o rezoluție mai mare decât necesară pentru început (sunt posibile scalări ulterioare ale imaginii).

Deşi majoritatea scanner-elor existente pot furniza la ieşire imagini binare (deci gata segmentate), pentru calități mai slabe ale tipăriturii, ele dau rezultate nesatisfăcătoare, fapt pentru care este de preferat achiziția imaginii în scară de gri, urmată de o segmentare folosind algoritmi mai performanți.

Diferitele tipuri de scannere existente sunt livrate cu un software dedicat. Indiferent de principiul lor de funcționare, pentru imaginile tip document, ele furnizează la ieşire un fişier, într-un format standard, conținând o imagine binară (la 1 bit/pixel) sau cu mai multe nuanțe (de gri sau culori).

Formatele utilizate în mod curent sunt **BMP**, **PCX** şi **TIFF**. Deoarece există deja programe de conversie dintr-un format într-altul (Picture Eze, Graphics Workshop, etc.), am optat pentru lucrul cu formatul cel mai simplu, adică **BMP**.

Formatul **PCX** prezintă avantajul unui spațiu-disc ocupat mai mic, datorită compresiei **RLL** utilizate. Formatul **TIFF**, probabil cel mai complet format de descriere a fişierelor imagistice, are un antet destul de complex, iar informația propriu-zisă poate fi necompresată sau compresată (cel mai ades cu algoritmul **LZW** care asigură o rată de compresie deosebit de bună).

De notat faptul că informația propriu-zisă este scrisă în fişier linie cu linie; un octet include deci 8 pixeli succesivi dintr-o linie, iar numărul de octeți alocat unei linii este întotdeauna rotunjit la multiplu de 4 octeți (32 biți). Prima linie din imagine corespunde ultimei linii din fişier.

Selectarea unui fişier de imagine pentru recunoaştere corespunde deschiderii fişierului **.BMP** respectiv şi verificării câmpurilor antetului corespunzător acestui format, astfel încât să fie respectată condiția ca fişierul să conțină o imagine (binară sau nu). Lungimea acestui fişier este:

$$L_{total} = L_{antet} + \Delta x \, \Delta y \, N_{bit/pixel} / 8 \qquad (1)$$

20.2. Localizarea obiectelor din imagine.

Algoritmul de localizare a obiectelor din imagine este, în principiu, un algoritm de urmărire de contur, adaptat la necesitățile aplicației.

Paşii urmați de acest algoritm sunt următorii:

(a). Prima şi ultima linie, prima şi ultima coloană din imagine sunt şterse (forțate la *alb=1*), ceea ce asigură funcționarea rapidă şi corectă a algoritmului. Nu se mai execută testele de "depăşire limite imagine" pentru fiecare pixel, ceea ce măreşte viteza de execuție a urmăririi de contur. De asemenea, această ştergere nu poate produce nici un fel de pierdere de informație, deoarece nu pot exista informații esențiale în zonele menționate.

(b). Se baleiază imaginea linie cu linie, de sus în jos şi de la stânga la dreapta, până la depistarea unui pixel obiect. (Prin obiect, în cele ce urmează, vom desemna caracterele încă nerecunoscute, părți sau fragmente de caracter, caractere concatenate, alte "pete" negre pe fond alb).

Se reține coordonatele X_S şi Y_S ale acestui prim pixel aparținând unui obiect din imagine. El devine "pixelul curent", adică $P(X_C, Y_C)$.

(c). Plecând de la coordonatele $(X_S, Y_S - 1)$ se baleiază vecinii pixelului curent, (în sens orar, de exemplu), în căutarea unui alt pixel obiect.

Dacă nu se găseşte aşa ceva, avem de-a face cu un pixel izolat, care se şterge, adică $P(X_C, Y_C) = P(X_S, Y_S) = 1(alb)$. În caz contrar, pixelul nou găsit devine pixel curent, cu coordonatele (X_C, Y_C). Astfel se elimină o parcurgere suplimentară a imaginii în scopul ştergerii pixelilor izolați.

Plecând de la precedentul pixel curent, se baleiază în acelaşi sens vecinii pixelului curent, până la depistarea unui nou pixel obiect, care va deveni pixel curent. Procedeul continuă până la închiderea conturului, adică până când pixel curent devine $P(X_S, Y_S)$.

(d). Pacurgerea conturului unui obiect este însoțită de determinarea limitelor dreptunghiului de încadrare, adică vor fi determinate valorile $X_{MIN}, X_{MAX}, Y_{MIN}, Y_{MAX}$, astfel:

$X_{MIN} = X_{MAX} = X_S; Y_{MIN} = Y_{MAX} = X_S$; (inițializare)

apoi, pentru fiecare pas al algoritmului (pixel curent) avem:

$\text{IF}\,(X_C < X_{MIN})\,\text{THEN}\; X_{MIN} = X_C;$ (2)

$\text{IF}\,(X_C > X_{MAX})\,\text{THEN}\; X_{MAX} = X_C;$ (3)

$\text{IF}\,(Y_C < Y_{MIN})\,\text{THEN}\; Y_{MIN} = Y_C;$ (4)

$$\text{IF}\,(Y_C < X_{MAX})\,\text{THEN}\ Y_{MIN} = Y_C; \qquad (5)$$

(e). Pacurgerea conturului unui obiect este însoţită de determinarea ariei mărginite de conturul urmărit. Dacă notăm cu A variabila-acumulator folosită pentru calculul ariei, iniţial avem $A = 0$ şi pentru fiecare nou pixel de contur ea este incrementată cu o valoare dependentă de poziţia relativă a pixelului curent faţă de cel precedent, conform tabelelor următor:

Sens orar parcurgere contur

Cod	Δx	Δy	ΔA
0	+1	0	+y
1	+1	+1	+y+0.5
2	0	+1	0
3	-1	+1	-y+0.5
4	-1	0	-y
5	-1	-1	-y-0.5
6	0	-1	0
7	+1	-1	+y-0.5

Sens trigonometric parcurgere contur

Cod	Δx	Δy	ΔA
0	+1	0	-y
1	+1	+1	-y-0.5
2	0	+1	0
3	-1	+1	+y+0.5
4	-1	0	+y
5	-1	-1	+y-0.5
6	0	-1	0
7	+1	-1	-y+0.5

Dacă valoarea finală obţinută pentru arie este negativă s-a parcurs un contur interior, deci nu avem de-a face cu un obiect autentic. Aria calculată pote fi folosită încă din această etapă pentru selectarea obiectelor foarte mici în scopul eliminării lor.

Odată localizat, obiectul ar putea fi salvat şi şters din imagine. De notat că este mai corectă ştergerea obiectelor din interiorul conturului închis depistat decât ştergerea tuturor pixelilor din dreptunghiul de încadrare, deoarece astfel se evită ştergerea unor informaţii utile ca în exemplul următor:

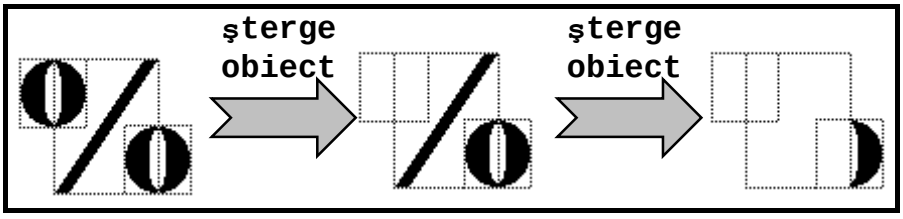


Fig. 2. Ştergerea eronată a obiectelor localizate.

Apoi se reia baleiajul de la exact aceeaşi linie (Ys), în căutarea altor obiecte. Obiectele prea mici sau prea mari se vor elimina (filtrul logic). Condiţiile de eliminare folosite sunt următoarele:

$$(X_{MAX}\text{-}X_{MIN}) < Th_1 \text{ sau } (Y_{MAX} - Y_{MIN}) < Th_2 \text{ sau } \sum_{i=X_{MIN}}^{X_{MAX}} \sum_{j=Y_{MIN}}^{Y_{MAX}} [1 - P(i,j)] < Th_3 \qquad (6)$$

$$(X_{MAX}\text{-}X_{MIN}) > Th_4 \text{ sau } (Y_{MAX} - Y_{MIN}) > Th_5 \text{ sau } \sum_{i=X_{MIN}}^{X_{MAX}} \sum_{j=Y_{MIN}}^{Y_{MAX}} [1 - P(i,j)] < Th_6 \qquad (7)$$

20.3. Localizarea zonelor de text din imagine.

O componentă importantă a procesului de pregătire a unui document pentru recunoaştere este separarea zonelor de text din documentul scanat. O pagină document obişnuită poate conţine nu numai text ci şi imagini, desene, grafice, etc.

Separarea zonelor de text dintr-o imagine scurtează semnificativ timpul de operare, ştiut fiind faptul că operaţiile de netezire obiecte, localizare obiecte, ordonare pe rânduri şi coloane, etc. durează destul de mult şi aplicarea unor algoritmi specifici care să limiteze aria de interes doar la zonele de text este în măsură să micşoreze timpul de răspuns al sistemului de recunoaştere.

Algoritmul RLSA (Run Length Smoothing Algorithm) conectează împreună în aceeaşi zonă caractere care sunt separate de mai puţin de un număr dat de pixeli. Acest algoritm se aplică pe verticală şi orizontală iar rezultatul final se obţine printr-o intersecţie logică între cele două imagini binare astfel obţinute. Rezultă astfel blocuri pentru fiecare rând de text. Apoi blocurile (deci rândurile de text) se concatenează pentru a forma paragrafe.

Iniţial obiectele din imagine pot fi clasificate în 6 tipuri (clase):

- **Tip 1** : separator vertical.
- **Tip 2** : text sau zgomot;
- **Tip 3** : separator orizontal;
- **Tip 4** : text;
- **Tip 5** : tabel, diagramă sau grafic;
- **Tip 6** : imagine "halftone";

Testele de mărime se referă la înălţimea obiectelor şi la raportul lăţime/înălţime, iar testul de "halftone" verifică conectivitatea între liniile adiacente ale aceluiaşi bloc, prin studierea **LAG** (Line Adjacency Graph) ataşat blocului curent. Obiectele cu lăţime mare pot fi încadrate într-unul din tipurile (2), (3), sau (4); un obiect cu înălţime mare dar cu lăţime foarte mică va fi un separator vertical (6), etc.

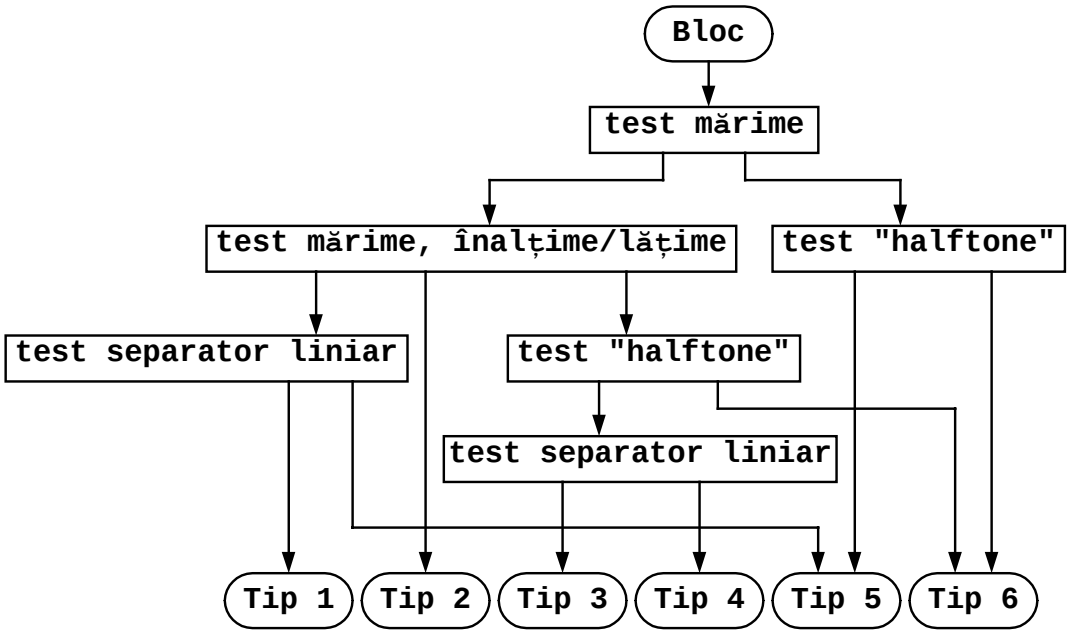


Fig. 3. Clasificarea obiectelor din imagine funcţie de dimensiuni.

Estimarea înclinării liniei de bază a rîndurilor de text se poate face de asemenea prin mai multe metode. Se foloseşte curent o tehnică LMS de potrivire a unei linii care trece prin centrul blocului ataşat unui rînd de text. Pot apare erori de estimare din cauza înălţimii diferite a caracterelor în limbile ce folosesc caractere latine (ex.: "**h**" faţă de "**e**"). Deoarece frecvenţa cea mai mare de apariţie o au caracterele cu înălţime mică ("**m**","**n**","**e**","**a**","**s**", etc.), eliminându-le pe celelalte, se poate micşora eroarea de estimare a acestui unghi.

20.4.Definirea relaţiilor între obiecte.

Două obiecte "i" şi "j" se pot considera ca aparţinând aceluiaşi rând din document dacă:

$$max[Y_{MIN}(i), Y_{MIN}(j)] < min[Y_{MAX}(i), Y_{MAX}(j)]$$

Dacă însă avem de-a face cu o scanare relativ incorectă, adică rândurile tipărite apar înclinate în imaginea scanată, atunci regula anterioară nu mai este valabilă.

Evident, pot fi utilizate tehnici mai performante de separare a rândurilor într-o imagine scanată, cum ar fi algoritmul următor:

a) ordonează obiectele în ordinea crescătoare a lui X_{MIN} ;

b) pentru primul obiect din lista ordonată se selectează vecinul lui din dreapta, adică acel obiect "*i*" pentru care:

$$max[Y_{MIN}(1), Y_{MIN}(i)] < min[Y_{MAX}(1), Y_{MAX}(i)] \text{ şi}$$

$$X_{MIN}(i) = \min_{k=2,\ldots}[X_{MIN}(K)]$$

c) Se selectează succesiv obiectele din acelaşi rând, folosind procedura de căutare anterioară. Căutarea se opreşte când nu mai există obiecte care să îndeplinească condiţiile de mai sus. Se obţine o secvenţă de obiecte aflată, în ordinea selectării, pe acelaşi rând, de la stânga la dreapta.

d) Se elimină din şirul ordonat obiectele marcate anterior ca aparţinând unui rând. Se repetă paşii b), c), d) ai algoritmului, plecând de fiecare dată de la primul obiect din şirul ordonat.

Obiectele din acelaşi rând pentru care este îndeplinită condiţia:

$$max[X_{MIN}(i), X_{MIN}(j)] < min[X_{MAX}(i), X_{MAX}(j)]$$

aparţin aceluiaşi caracter, deci ele vor fi concatenate prin plasarea lor într-o fereastră cu coordonatele colţurilor:

$$(min[X_{MIN}(i), X_{MIN}(j)], min[Y_{MIN}(i), Y_{MIN}(j)]) \text{ şi}$$

$$(max[X_{MAX}(i), X_{MAX}(j)], max[Y_{MAX}(i), Y_{MAX}(j)])$$

O altă procedură care ar putea fi folosită implică utilizarea unor ferestre mobile, şi este descrisă în continuare:

a) ordonează obiectele în ordinea crescătoare a lui X_{MIN} ;

b) pentru primul obiect din lista ordonată, care implicit este plasat pe prima poziţie din stânga a unui rând oarecare se construieşte o fereastră având ordonatele date de:

$$[Y_{MIN} - k*(Y_{MAX} - Y_{MIN}), Y_{MIN} + k*(Y_{MAX} - Y_{MIN})]$$

căutarea făcându-se în această fereastră, după deplasarea ei pe orizontală, către dreapta.

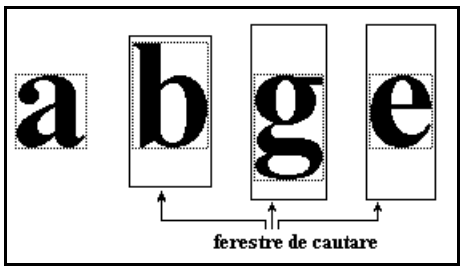
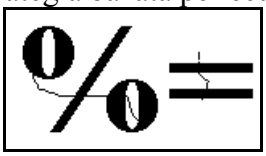


Fig. 4. Fereastră mobilă de căutare a caracterelor.

Avantajul acestei metode este evident în cazul în care apar caractere concatenate formate din obiecte care se ating, caz în care strategia bazată pe recunoaşterea obiectelor eşuează.



O altă variantă determină, după localizarea primului obiect, valoarea $(Y_{MIN} + Y_{MAX})/2$, căutarea următoarelor obiecte facându-se în stânga şi în dreapta obiectului găsit, de-a lungul liniei cu ecuaţia:

$$y = (Y_{MIN} + Y_{MAX})/2 = const.$$

Se remarcă posibilitatea funcţionării incorecte a acestei variante pentru anumite caractere speciale, precum "**:**", "**;**", "**.**", "**,**", etc. Amplasarea corectă în text a acestor simboluri se poate face ulterior.

Dacă ***nu se execută în această fază operația de asamblare*** a caracterelor compuse din mai multe simboluri, asamblarea poate fi făcută mai tîrziu, pe baze logice. Către clasificator vor fi trimise în acest caz nu ***caractere*** ci ***obiecte***. Lista obiectelor ce pot rezulta în urma operației de localizare (fără asamblare) este:

Cifrele(10): **01234567890**;
Litere mici(23): **abcdefghklmnpqrstuvwxyz;**
Litere mari(25): **ABCDEFGHIJKLMNPQRSTUWVXYZ;**
Caractere speciale(18): **( ) [ ] { } @ # $ & * + / \ < > ^ ~**
Alte caractere şi fragmente de caractere(8):
- punct, virgulă, bară orizontală, bară verticală;
- "i fără punct", "j fără punct";
- "semnul întrebării fără punct";
- "₀ de la procent %".

Rezultă un total de 84 simboluri la care, pentru cazul limbii române, ar putea fi adăugate căciulița de la "**î**" şi "**â**". Pot fi folosite în acest scop însă ultimele două simboluri de la caractere speciale (^ şi ~), deoarece diferențierea între cele două perechi de simboluri se face relativ uşor din studiul pozițiilor relative ale obiectelor aflate pe acelaşi rând şi la aceeaşi abscisă: "^" şi "~" apar izolate, pe când căciulițele apar deasupra unui "**i fără punct**", respectiv "**a**".

Caracterele "**0**"(zero), "**o**"(o mic) şi "**O**"(O mare), datorită formei lor mult prea asemănătoare vor fi considerate ca reprezentând un singur caracter. Diferențierea lor se va face din analiza contextului, eventual din estimarea proporțiilor.

Odată recunoscute aceste obiecte, recunoaşterea caracterelor compuse se face după următorul şir de reguli:

- *"."(punct) deasupra unui "i fără punct" ==> caracterul "i";*
- *"."(punct) deasupra unui "j fără punct" ==> caracterul "j";*
- *"₀" în stânga-sus față de obiectul "/" şi un alt "₀" în dreapta-jos față de acelaşi obiect "/" ==> caracterul "%";*
- *"."(punct) deasupra unui "."(punct) ==> caracterul ":";*
- *"."(punct) deasupra unui ","(virgula) ==> caracterul ";";*
- *"." (punct) sub o bară verticală ==> caracterul "!";*
- *"." (punct) sub "semnul întrebării fara punct" ==> caracterul "?";*

În plus, pentru limba română, apar regulile:

- *"~" deasupra unui obiect "a" ==> caracterul "a";*
- *"^" deasupra unui obiect "a" ==> caracterul "a";*
- *"^" deasupra unui "i fără punct" ==> caracterul "i";*
- *"," (virgula) sub un obiect "s" ==> caracterul "s";*
- *"," (virgula) sub un obiect "t" ==> caracterul "t";*
- *"," (virgula) sub un obiect "S" ==> caracterul "S";*
- *"," (virgula) sub un obiect "T" ==> caracterul "T";*
- *"-" (minus) sub sau deasupra unui alt "-" ==> caracterul "=".*

Un alt set de reguli suplimentare pot fi folosite pentru a elimină anumite tipuri de erori, destul de frecvente, precum confuzia între "."(punct), ","(virgula),"^" şi "~":

- *","(virgula) sub un "! fără punct" se transformă în "." pentru a forma caracterul "!";*
- *","(virgula) sub un "? fără punct" se transformă în "." pentru a forma caracterul "?";*
- *","(virgula) deasupra unui "i fără punct" se transformă în "." pentru a forma caracterul "i";*

- *","(virgula) deasupra unui "j fără punct" se transformă în "." pentru a forma caracterul "j";*
- *"."(punct) sub un caracter "s", "S", "t", "T" se transformă în obiectul "virgulă" pentru a forma caracterele din limba română "s", "S", "t", "T";*

20.5. Algoritm de segmentare a textului prin analiză morfologică.

În acest paragraf se descrie un algoritm original de segmentare a textului prin analiza morfologică a obiectelor din imagine. El se bazează pe folosirea unor informaţii apriorice referitoare la structura oricărui text tipărit.

Orice text tipărit poate fi definit ca o mulţime de obiecte (care pot fi caractere sau părţi din caractere – cum ar fi punctul literei "i") a căror poziţii relative permit gruparea lor în caractere, a caracterelor în cuvinte, a cuvintelor în rânduri şi, eventual, a rândurilor în paragrafe şi a paragrafelor în coloane.

Această definiţie constituie cheia algoritmului care va fi descris în continuare.

Doar câteva caractere sunt compuse din mai multe obiecte. Pe lângă "**i**", "**j**", "**!**"(semnul exclamării), "**?**"(semnul întrebării), "**:**"(două puncte), "**;**"(punct şi virgulă), "**”**"(ghilimele) şi "**%**"(procent) care se folosesc în mai toate limbile de origine europeană, s-au mai introdus în studiu caracterele specifice limbii române: "**ă**", "**î**", "**ş**", "**ţ**", "**â**".

Propoziţiile următoare definesc poziţiile relative ale obiectelor din cadrul caracterelor compuse:

- ***Toate aceste caractere compuse sunt alcătuite din câte două obiecte, cu excepţia caracterului "%"(procent).***
- ***Toate caracterelor compuse se caracterizează prin amplasarea relativă a celor două obiecte componente "unul deasupra celuilalt", cu excepţia caracterelor "”"(ghilimele) şi "%"(procent).***
- ***Caracterul "”"(ghilimele) este format din două obiecte de dimensiuni aproximativ egale, mai mici decât caracterele necompuse şi amplasate pe aceeaşi linie***
- ***Caracterul "%"(procent) este compus din trei obiecte, un "o" în stânga-sus faţă de obiectul "/" şi un alt "o" în dreapta-jos faţă de acelaşi obiect "/".***

Analizând, pe baza propoziţiilor precedente, obiectele localizate într-o imagine se realizează *gruparea obiectelor în caractere*.

Un alt grup de reguli poate fi definit pentru *gruparea caracterelor în cuvinte* pentru limbile de origine europeană.

- ***În cadrul unui cuvânt, caracterele se poziţionează pe aceeaşi linie, unul în stânga celuilalt.***
- ***Distanţele dintre caractere sunt sensibil mai mici decât distanţele dintre cuvinte.***

Următorul grup de reguli poate fi definit pentru *gruparea cuvintelor în rânduri* pentru limbile de origine europeană.

- ***În cadrul unui rând, cuvintele se poziţionează pe aceeaşi linie, unul în stânga celuilalt.***
- ***Distanţele dintre coloane sunt sensibil mai mari decât distanţele dintre cuvinte.***

Pentru gruparea rândurilor în paragrafe, funcţie de stilul de tehnoredactare ales, poate fi folosită una din regulile următoare (de obicei doar una din ele este adevărată).

- ***Distanţa dintre două paragrafe este mai mare ca distanţa dintre două rânduri.***
- ***Rândul de început al unui paragraf are limita stângâ sensibil mai mare decât limita stânga a următorului rând din paragraf.***

Pentru gruparea paragrafelor în coloane, poate fi folosită una din regulile următoare (de obicei doar una din ele este adevărată).

- ***Paragrafele din aceeaşi coloană au aceeaşi limită stângă şi (aproximativ) aceeaşi limită dreaptă.***
- ***Distanţele pe orizontală dintre coloane sunt sensibil mai mari decât distanţele dintre cuvinte.***
- ***Distanţele pe verticală dintre coloane sunt sensibil mai mari decât distanţele dintre paragrafe.***

Adesea segmente de linie suplimentare se folosesc ca separatori de coloane sau pentru separarea articolelor între ele.

Relaţiile dimensionale definite până acum urmează a fi traduse în relaţii matematice mai departe. Folosind regulile descrise aici s-a dezvoltat un algoritm original de segmentare a textului din imaginile digitale.

20.5.1. Conectarea obiectelor.

În această fază se urmăreşte construcţia unui graf orientat care să descrie cât mai fidel relaţiile dintre obiectele localizate în imagine. Fiecare din nodurile acestui graf este aferent unuia din obiectele detectate în imagine. Conexiunile dintre ele (arcele) se construiesc pe baza poziţiilor relative are obiectelor corespunzătoare, folosind un set de reguli deduse folosind propoziţiile definite în paragraful următor.

În faza iniţială se face o conectare preliminară a obiectelor, după care se încearcă tratarea conexiunilor multiple (conexiuni de la sau către un nod comun).

Regula folosită în această fază iniţială este dată de:

- ***Orice obiect se conectează cu cel mai apropiat obiect aflat în dreapta sa, pe aceeaşi linie***.

Implementarea acestei reguli se realizează principial în doi paşi:

- ***Se conectează mai întâi orice obiect cu toate obiectele aflate în stânga sa pe acelaşi rând.***
- ***Pentru orice tripletă de obiecte {A, B, C} aflate în această ordine pe acelaşi rând, în perechea de conexiuni [A->B] şi [A->C] se înlocuieşte ultima conexiune cu [B->C].***

Graful orientat care se obţine în această fază din imaginea unei zone de text va conţine conexiuni multiple doar pentru caracterele compuse. Prezenţa în imagine a altor zone diferite de cele de text (linii, grafice, imagini) generează conexiuni multiple suplimentare. După tratarea acelor conexiuni multiple datorate caracterelor multiple, se poate presupune cu un grad extrem de mare de siguranţă, că toate conexiunile multiple rămase sunt aferente obiectelor de tip linie, grafic, imagine, dar nu obiectelor de tip text.

- Pentru fiecare obiect din imagine (nod în graf) se reţin următoarele informaţii:
- Coordonatele punctului de start pentru parcurgerea conturului; ele sunt necesare în acele cazuri în care dreptunghiurile de încadrare ale caracterelor alăturate se suprapun, caz destul de des întâlnit dacă caracterele din text sunt de tip "***italic***". Coordonatele acestui punct de start sunt necesare pentru selecţia rapidă a obiectului vizat din dreptunghiul de încadrare;
- Coordonatele dreptunghiului de încadrare;
- Lista referinţelor către obiectele ţintă cu care acest obiect este conectat;
- Lista referinţelor către obiectele rădăcină conectate cu acest obiect;
- Indicator pentru cazul in care obiectul curent este ataşat unui alt obiect împreună cu care formează un obiect compus;

Pentru accelerarea manipulării informaţiilor referitoare la obiectele din imagine şi conexiunile între ele se poate completa lista precedentă cu:

- Numărul obiectelor ţintă cu care este conectat obiectul curent;
- Numărul obiectelor rădăcină conectate cu acest obiect;
- Indicator pentru obiectele aflate la început de rând de text.

20.5.2. Tratarea conexiunilor convergente.

Plecând de la ipoteza că unele din conexiunile multiple rămase sunt datorate caracterelor compuse din mai multe obiecte, se verifică poziţiile relative ale acestor obiecte şi dacă ele satisfac ipotezele corespunzătoare descrise în paragraful 20.4, vor fi etichetate în mod corespunzător.

În figurile următoare este prezentat un caz tipic de evoluţie a unei conexiuni convergente datorate unui caracter compus (litera "**i**" în acest caz). Ambele obiecte componente ale acestui caracter se află pe acelaşi rând cu caracterul "**b**", ceea ce duce la apariţia conexiunii convergente.

Deoarece obiectele "**punct**" şi "**m**" nu se află în aceeaşi coloană, conexiunea de la obiectul "**punct**" la obiectul "**b**" se înlocuieşte cu conexiunea mai scurtă de la obiectul "**punct**" la obiectul "**m**". Se obţine o conexiune multiplă similară, care se tratează folosind aceleaşi reguli. Situaţia obţinută acum este diferită, deoarece obiectele rădăcină ale conexiunii multiple se află acum în aceeaşi coloană, deci se poate presupune că ele aparţin aceluiaşi caracter. Ca urmare, se va eticheta într-un mod special obiectul rădăcină mai mic ("**punct**") şi va fi ataşat de obiectul rădăcină mai mare "**i fără punct**".

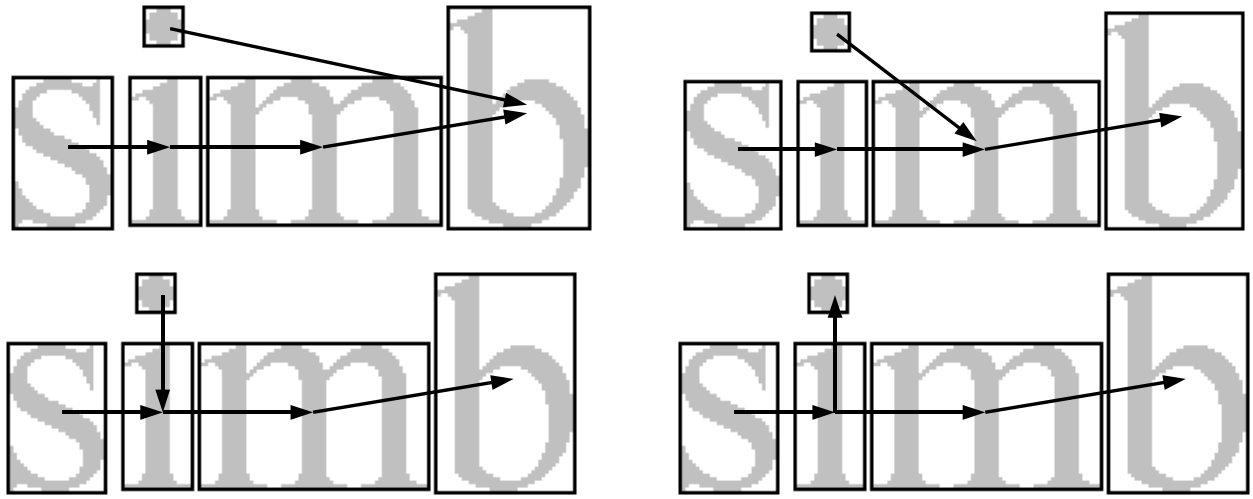


Fig. 5. Tratarea conexiunilor convergente.

Generalizând, o conexiune convergentă se tratează folosind următoarele două reguli:

- ***Dacă obiectele rădăcină ale unei conexiuni convergente se află în coloane diferite, conexiunea de la obiectul rădăcină aflat mai la stânga capătă ca destinaţie celălalt obiect rădăcină.***
- ***Dacă obiectele rădăcină ale unei conexiuni convergente se află în aceeaşi coloană, obiectul rădăcină mai mic se conectează la obiectul rădăcină mai mare, conexiunea sa către obiectul destinaţie este distrusă, el fiind etichetat ca obiect* "ataşat".**

20.5.3. Tratarea conexiunilor divergente.

Se va folosi şi în acest caz ipoteza că unele din conexiunile multiple rămase sunt datorate caracterelor compuse din mai multe obiecte. Verificând poziţiile relative ale acestor obiecte pe baza ipotezelor corespunzătoare descrise în paragraful 20.4, ele pot fi tratate în mod asemănător conexiunilor convergente.

În figurile următoare este prezentat un caz tipic de evoluţie a unei conexiuni divergente datorate unui caracter compus (litera "**i**"). Ambele obiecte componente ale acestui caracter se află pe acelaşi rând cu caracterul "**l**", ceea ce duce la apariţia conexiunii divergente.

Deoarece obiectele "**punct**" şi "**a**" nu se află în aceeaşi coloană, conexiunea de la obiectul "**l**" la obiectul "**punct**" se înlocuieşte cu conexiunea mai scurtă de la obiectul "**a**" la obiectul "**punct**". Se obţine o conexiune divergentă similară, care se tratează folosind aceleaşi reguli.

Situaţia obţinută după trei paşi este diferită, deoarece obiectele ţintă ale conexiunii multiple se află acum în aceeaşi coloană, deci se poate presupune că ele aparţin aceluiaşi caracter. Ca urmare, se va eticheta într-un mod special obiectul rădăcină mai mic ("**punct**") şi va fi ataşat de obiectul rădăcină mai mare "**i fără punct**".

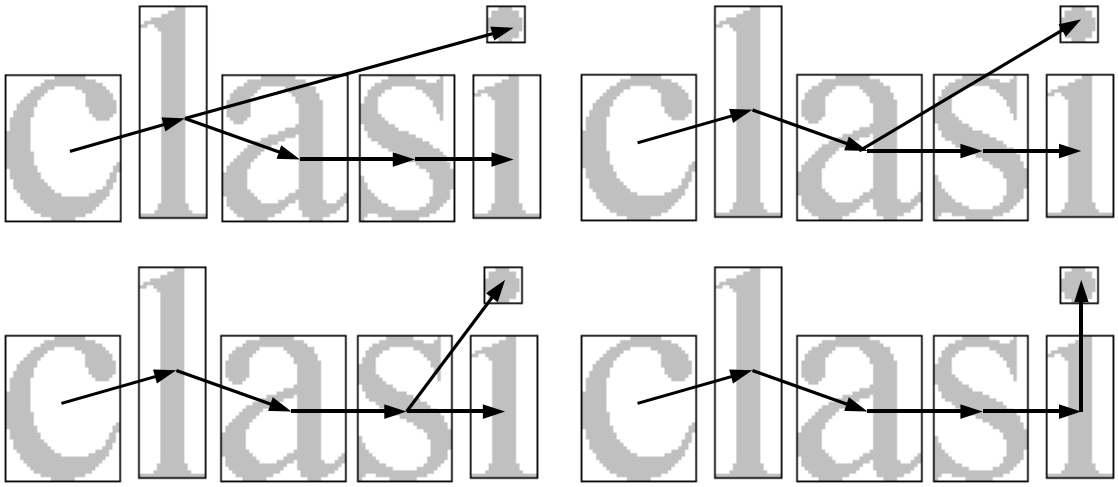


Fig. 6. Tratarea conexiunilor divergente.

Generalizând, o conexiune divergentă se tratează folosind următoarele două reguli:

- ***Dacă obiectele ţintă ale unei conexiuni convergente se află în coloane diferite, conexiunea către la obiectul ţintă aflat mai la dreapta capătă ca rădăcină celălalt obiect ţintă.***
- ***Dacă obiectele ţintă ale unei conexiuni divergente se află în aceeaşi coloană, obiectul ţintă mai mic se conectează la obiectul ţintă mai mare, conexiunea sa de la obiectul rădăcină este distrusă, el fiind etichetat ca obiect* "ataşat".**

Se poate remarca cu uşurinţă simetria existentă în modul de tratare a conexiunilor convergente şi a celor divergente.

20.5.4. Tratarea conexiunilor multiple reziduale.

Conexiunile multiple rămase în această fază a algoritmului pot fi datorate unor cauze diferite:

- ***Erori de scanare, de obicei sub forma caracterelor fragmentate, ceea ce duce la mai mult de două obiecte pentru un caracter. De asemenea prezenţa zgomotului poate avea acelaşi efect.***
- ***Prezenţa în imaginea scanată a unor obiecte diferite de text: linii, desene, imagini binarizate (halftone images).***
- ***Prezenţa în imaginea scanată, în zonele de text, a formulelor matematice.***

Obiectele implicate în conexiunile multiple rămase se tratează separat, încercându-se fie concatenarea lor, fie eliminarea unor conexiuni inutile, fie chiar eliminarea unora din obiecte din faza de recunoaştere. Se apelează pentru aceasta la criterii dimensionale şi la analiza poziţiei lor relative.

Apariţia în text a caracterelor de tip "***italic***" duce adesea la suprapunerea drepunghiurilor de încadrare a caracterelor alăturate.

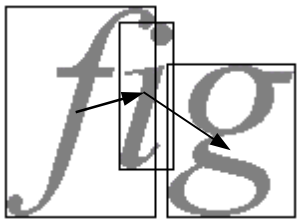


Fig. 7. Suprapunerea dreptunghiurilor de încadrare.

Aceasta nu afectează cu nimic comportarea algoritmului de segmentare, atâta timp cât caracterele nu se ating între ele. În faza de recunoaştere trebuie însă avut în vedere faptul că nu trebuie analizat întreg conținutul dreptunghiului de încadrare, ci doar caracterul corespunzător.

20.5.5. Ordonarea obiectelor din text.

Folosind metoda descrisă până acum, este foarte simplu de efectuat ordonarea obiectelor din text, pentru a furniza la intrarea clasificatorului caracterele în ordinea naturală a scrisului şi a stabili în acelaşi timp localizarea finală a celorlalte componente ale documentului: cuvinte, rânduri, paragrafe, eventual coloane, linii, desene, imagini.

Folosind informația din structura de date care descrie graful orientat obținut până acum se realizează mai întâi gruparea obiectelor în caractere, grupând obiectele etichetate ca "ataşate" astfel încât ele vor fi considerate mai departe ca formând împreună un caracter.

Gruparea caracterelor în cuvinte este echivalentă cu detectarea separatorilor (spațiilor) dintre cuvinte. Este o operație simplă care are ca efect etichetarea conexiunilor între acele caractere care sunt suficient de depărtate între ele. Pragul peste care distanța între două caractere se consideră a fi un separator de cuvânt se determină foarte simplu, ca o fracțiune din înălțimea caracterelor respective. După faza de recunoştere între caracterele recunoscute corespunzătoare se vor însera unul sau mai multe caractere spațiu, funcție de distanța măsurată.

O distanță foarte mare între două caractere aflate pe acelaşi rând poate însemna faptul că ele aparțin unor coloane de text diferite.

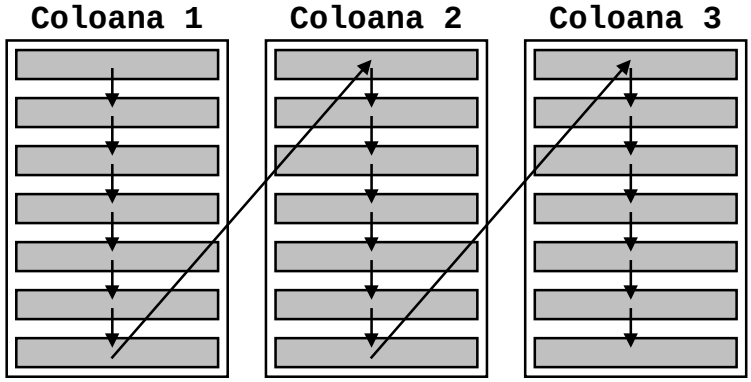


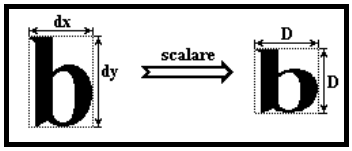
Fig. 8. Ordonarea rândurilor şi coloanelor de text.

În acest caz conexiunea între caracterele corespunzătoare trebuie distrusă, iar în aplicarea algoritmului de ordonare a rândurilor de text trebuie analizată poziția relativă a rândurilor depistate, nu doar ordonata începutului de rând.

20.6. Extragerea trăsăturilor.

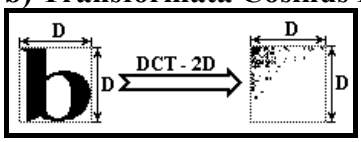
Faza de extragere a trăsăturilor presupune parcurgerea următorilor paşi:

a) Scalarea simbolurilor:



Se obţine o matrice binară de dimensiune constantă, conţinând imaginea scalată. Astfel este rezolvată invarianţa la scalare pe orizontală şi verticală, impusă de aplicaţie.

b) Transformata Cosinus Discretă bidimensională:



Deoarece implementarea transformării optimale (Karhunen-Loeve) este dificilă, modelând intrarea **{X}** cu un proces Markov de ordinul unu, cea mai bună aproximare a transformării Karhunen-Loeve discretă este **DCT** (Transformata Cosinus Discretă). Această modelare nu este foarte restrictivă şi se poate folosi cu succes în multe aplicaţii.

Pentru calculul **DCT** există nu numai algoritmi rapizi de calcul, ci şi circuite specializate de mare viteză.

Pentru cazul bidimensional se foloseşte:

$$F(u,v) = \frac{1}{2N} \sum_{j=0}^{N-1} \sum_{i=0}^{N-1} f(j,k)\ \cos\left(\frac{\pi u j}{N}\right)\ \cos\left(\frac{\pi v k}{N}\right)$$

rezultând o matrice de aceeaşi dimensiune, cu valori reale, slab corelate. Valorile semnificative ale acestei matrici sunt plasate în colţul din stânga sus.

c) Ordonarea Zig - Zag.

Este o procedură de vectorizare a matricii **DCT**, după regula din figura de mai jos:

Mat[8][8] ===> Z[64]; **Mat[D][D] ===> Z[D*D];**

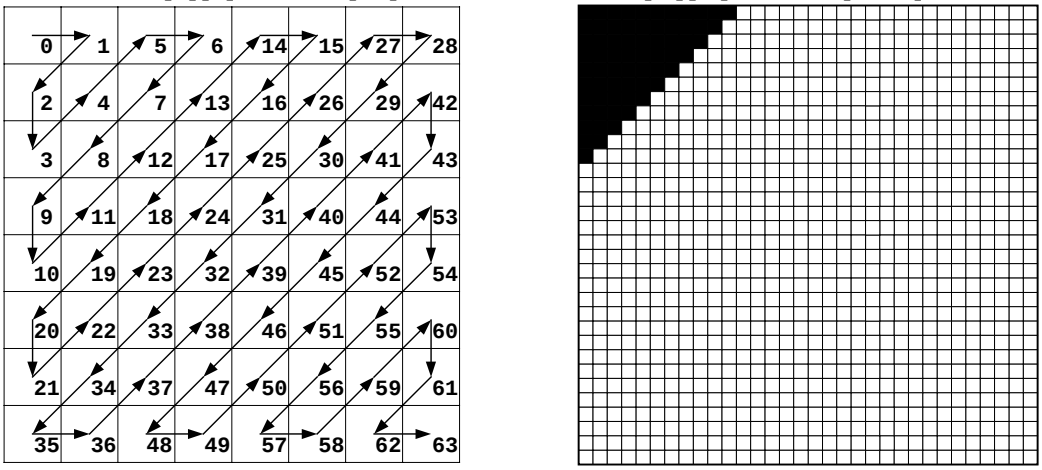


Fig. 9. Ordonarea Zig-Zag.

Deoarece **DCT** bidimensională concentrează informaţia din imaginea iniţială în colţul din stânga sus al matricii rezultate, după ordonarea Zig - Zag aceeaşi informaţie va fi regăsită în primele elemente ale vectorului obţinut.

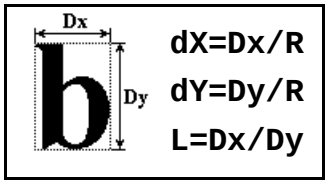
d) Selecţia trăsăturilor.

Se reţin primele **F** valori din vectorul **Z[D*D]**, adică doar o fracţiune din valori, cele marcate cu negru în figura de mai sus. Conform afirmaţiilor anterioare, aceste elemente concentrează cea mai mare parte din informaţia din imaginea iniţială şi ele vor fi folosite în faza de recunoaştere.

Acest bloc funcțional elimină trăsăturile care sunt neimportante în faza de recunoaştere. Ieşirile lui sunt aplicate la intrarea clasificatorului neural. Soluția prezentată de obținere a trăsăturilor poate fi implementată pe un suport hardware dedicat de mare viteză, care poate fi utilizat pentru a rezolva probleme similare.

20.7. Clasificarea propriu-zisă.

La intrarea în sistemul de clasificare se aplică, pentru fiecare caracter, vectorul de trăsături extrase în faza precedentă, la care se adaugă dimensiunile normalizate ale fiecărui caracter (normalizarea se face prin raportare la înalțimea estimată a rândului). De asemenea se utilizează şi raportul dimensiunilor dreptunghiului de încadrare.



Se remarcă faptul că aceste noi trăsături sunt şi ele invariante la scalare, aşa cum aplicația o cere.

Caracterele care trebuiesc recunoscute sunt 89 la număr:

Cifrele(10): **0123456789**;

Litere mici(25): **abcdefghijklmnpqrstuvwxyz;**

Litere mari(25): **ABCDEFGHIJKLMNPQRSTUWVXYZ;**

Caractere speciale(29): **( ) [ ] { } @ # $ & * + / \ < > ^ ~**

. ,(virgula, apostroafe) **-**(minus, underline) **| ? ! % " ; : =**

Pentru limba română mai apar în plus şi caracterele "**ă**", "**â**", "**î**", "**ş**", "**ț**".

Numărul lor poate fi redus la 78 dacă se realizează recunoaşterea unor caractere simple încă din faza de localizare, utilizând pentru aceasta doar măsurătorile dimensionale (trăsături) menționate mai sus: dimensiunile normalizate la înălțimea estimată a rândului pentru fiecare caracter şi raportul dimensiunilor dreptunghiului de încadrare. Pot fi astfel recunoscute următoarele caractere: **-**(minus), **_**(underline), **=**(egal), **!**(semnul exclamării), **|**(bara verticală), **.**(punct), **,**(virgula), **;**(punct şi virgulă, **:**(două puncte), **"**(ghilimele), ' şi `(apostroafe).

Complexitatea problemei poate fi micşorată împărțind-o în mai multe sarcini mai mici, şi anume alcătuind câteva grupe de caractere (fiecare caracter = o clasă) şi apoi construind câte un clasificator pentru fiecare grupă de caractere. Trăsăturile extrase se aplică simultan la intrările tuturor clasificatorilor şi în mod normal unul singur va furniza decizia de recunoaştere. Rolul logicii de decizie este de a furniza o decizie corectă în cazurile cu o oarecare doză de incertitudine, utilizând informațiile furnizate de toate clasificatoarele. Schema bloc a unui asemenea sistem este prezentată în figura 10.

Pentru simbolurile nerecunoscute se mai face o încercare în ipoteza că ele reprezintă caractere concatenate: se încearcă separarea lor cu metode specifice, apoi rezultatul este iarăşi aplicat la intrarea clasificatorului pentru o nouă încercare.

Avantajul unei asemenea scheme apare evident în cazul unei structuri de calcul paralel care să implementeze rețelele neurale componente ale clasificatorului. Deşi această variantă este optimă, se poate utiliza, pentru simplificarea problemei, un preclasificator care să specifice cărei grupări îi aparține fiecare caracter de intrare şi în consecință, care rețea neurală trebuie folosită pentru recunoaşterea efectivă.

Această variantă este descrisă în schema bloc din figura 11:

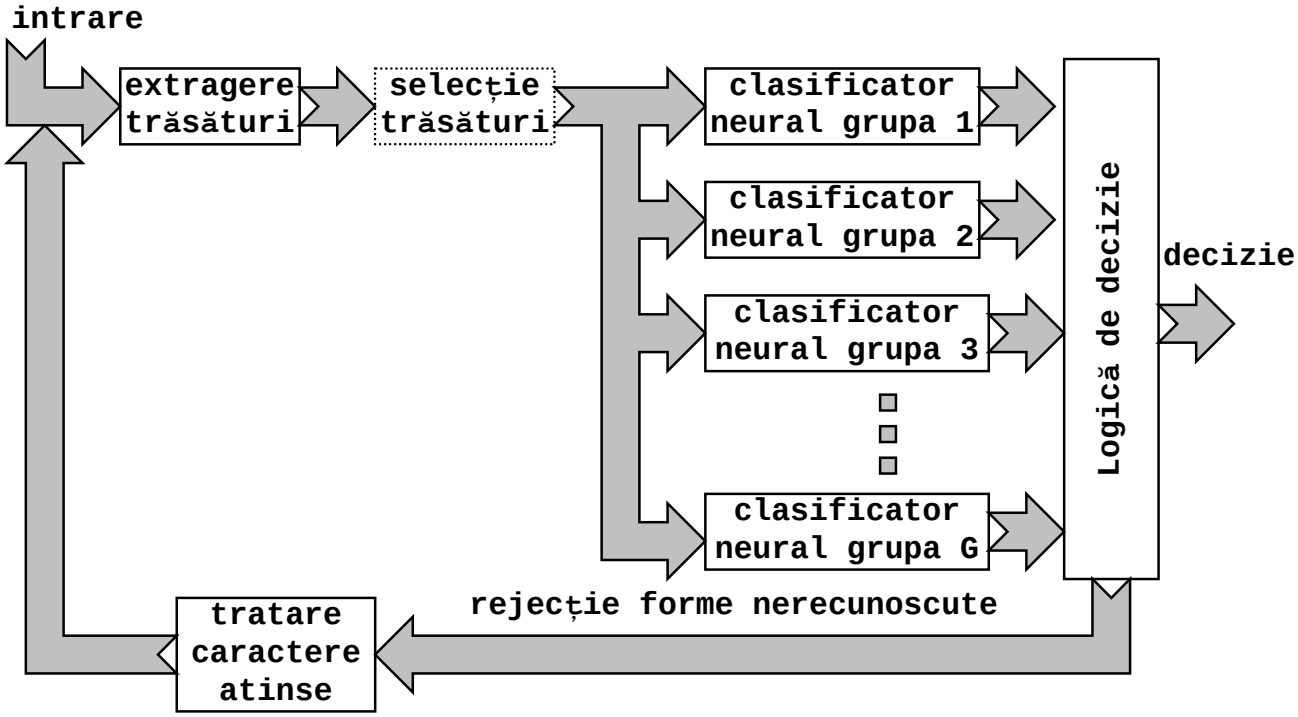


Fig. 10. Sistem de recunoaştere caractere, varianta 1.

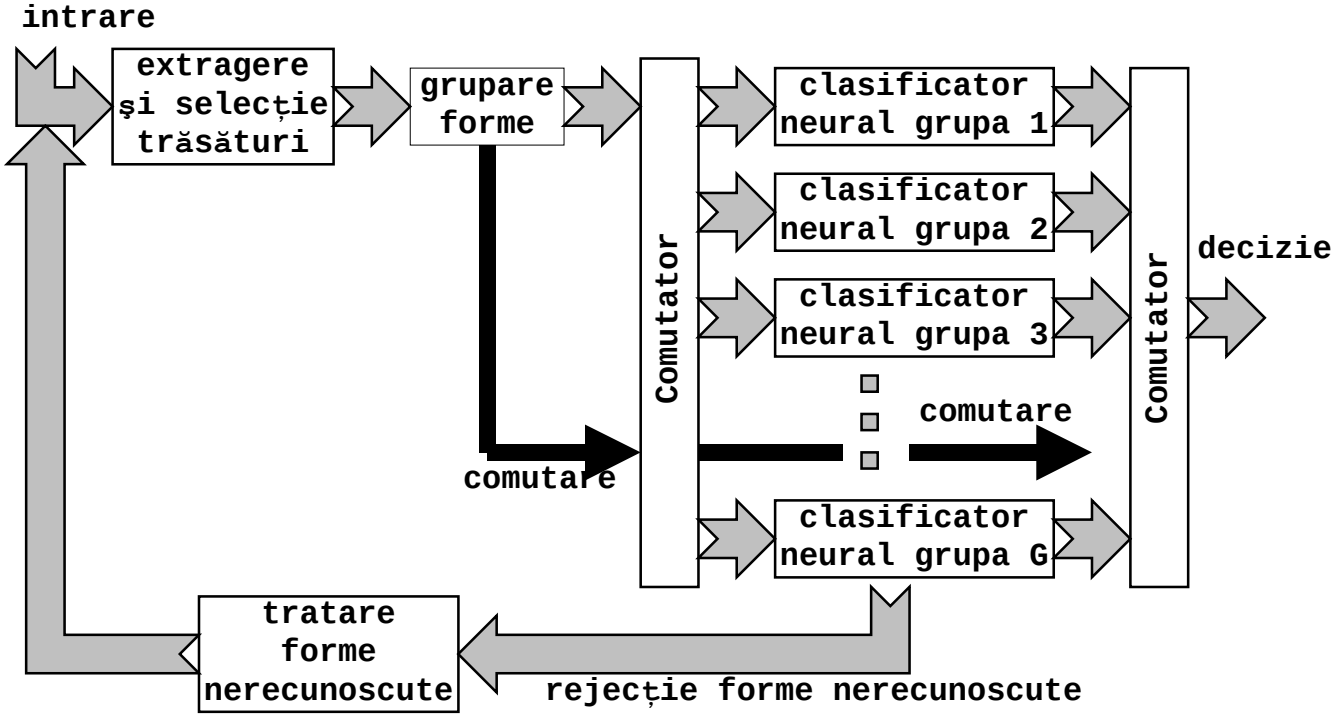


Fig. 11. Sistem de recunoaştere caractere, varianta 2.

Folosind un algoritm de grupare, cum ar fi algoritmul **LBG** (Linde-Buzo-Gray) de cuantizare vectorială, se construiesc prin antrenare nesupervizată un număr (predeterminat sau nu, funcție de algoritmul ales) de **G** grupe, fiecăreia ataşându-i-se câte un clasificator. Trăsăturile extrase se aplică modulului de grupare care indică cărei grupe îi aparține fiecare caracter ce trebuie recunoscut. Odată această informație cunoscută, se selectează, folosind comutatorul, clasificatorul ataşat grupei determinate. Ieşirea lui va furniza decizia de apartenență a caracterului de intrare la una din clasele posibile sau îl va rejecta în cazul eşuării operației de recunoaştere.

Schema simplificată a unui sistem de recunoaştere este:

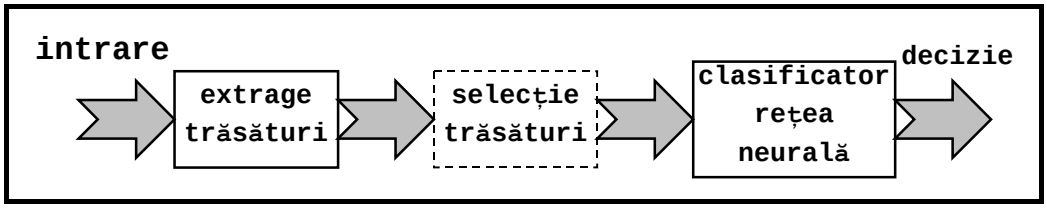


Fig. 12. Schema simplificată a unui sistem de recunoaştere.

În proiectarea acestui clasificator s-a urmărit, pe lângă optimizarea timpului de răspuns (cât mai mic) şi a ratei de recunoaştere (cât mai mare), şi posibilitatea unei implementări hardware, într-o structură dedicată de mare viteză, care să poată rezolva probleme asemănătoare (dar nu identice).

Selectorul de trăsături poate reține cele mai semnificative caracteristici furnizate de extractorul de trăsături. O anume decizie a clasificatorului înseamnă o configurație particulară de activare a neuronilor de pe stratul de ieşire a rețelei.

Primele **F** elemente **(F<L*L)** ale vectorului **Z** se aplică la intrarea unei rețele neurale de tipul ***perceptron cu trei straturi*** şi nelinearitate sigmoidală simetrică (+/-1) la toți neuronii.

Numărul de neuroni folosiți în primul din cele trei straturi depinde de numărul de trăsături extrase. Stratul de ieşire trebuie să aibă un număr de neuroni dependent de numărul de clase din grupa corespunzătoare şi de regula de decizie adoptată.

Dacă numărul de neuroni este egal cu numărul de clase din grupă, regula de decizie adoptată este:

Caracterul C_i *aparține clasei "K" dacă*:

Yk>Prag şi Yj<(1-Prag), oricare $j \neq k$ *, unde 0<Prag<1.*

Se remarcă faptul că, pentru obiectele nerecunoscute, rețeaua neurală trebuie să răspundă în acest caz cu toate ieşirile inactive sau cu mai mult de o ieşire activă. Caracterele nerecunoscute corespund fie caracterelor aparținând altor grupări, fie caracterelor concatenate, caz în care toate rețelele neurale vor trebui să răspundă identic.

Dacă numărul de neuroni este mai mic decât numărul de clase din grupă, regula de decizie adoptată poate fi:

Caracterul C_i *aparține clasei "K" dacă:*

$\{ [K \text{ div } 2^P] = 0$ *şi Yp<(1-Prag)* $\}$ *sau* $\{ [K \text{ div } 2^P] = 0$ *şi Yp>Prag* $\}$,

unde p este numărul de ordine al ieşirii, cu $p = [0.5 + [\log_2 Ng]]$, iar *Ng* - numărul de clase din grupa *"g"*.

În acest caz *p* este lungimea minimă de reprezentare în binar a lui *Ng***,** iar la ieşirea rețelei vor trebui activați doar acei neuroni corespunzători lui *"1"* logic în reprezentarea binară a lui *K*. Pentru obiectele nerecunoscute rețeaua va răspunde activând neuronii de ieşire după o configurație corespunzătoare codului $K + 1$.

Pentru antrenare s-a folosit evident algoritmul "backpropagation" cu factorul de câştig $\eta = 0.3$, factorul de inerție $\alpha = 0.7$.

20.8. Construcția setului de date de antrenament şi test.

Pentru obținerea unui set de antrenament şi test reprezentativ trebuiesc selectate documente tipărite caracteristice unei game cât mai largi de fonturi, mărimi de simboluri şi calități ale tipăriturii. Vor trebui incluse, pentru acelaşi font, variantele normal, **bold,** *italics* şi ***bold-italics***.

Setul de antrenament conține simboluri etichetate din toate clasele şi se remarcă în plus utilizarea unei clase de obiecte, "***clasa de rejecție***", care include, pentru fiecare grupare, caracterele care **nu** aparțin acelei grupări, plus un set de caractere concatenate. Ea permite, pentru fiecare grupare în parte, obținerea unor rezultate net superioare.

Pentru determinarea claselor care aparțin de fiecare grupare, s-a optat pentru utilizarea algoritmului **LBG** (Linde-Buzo-Gray) de cuantizare vectorială, care este în esență un algoritm de grupare cu număr predeterminat de clase. El permite construirea unui număr predeterminat de grupe, conținând formele cele mai apropiate.

Analizând componența fiecărei clase, se poate decide care anume caractere vor fi atribuite fiecărui clasificator. Tot astfel se pot trage concluzii privind probabilitățile de eroare de clasificare pentru diferite clase: vor fi confundate mai uşor obiecte aflate în aceeaşi grupare şi nu în grupări diferite.

Cu cât sunt mai multe grupări, cu atât complexitatea clasificatorului neural este mai mică (scade numărul de neuroni de pe stratul intermediar şi cel de ieşire), ceea ce diminuează timpul de antrenament şi spațiul de memorie ocupat de fiecare clasificator. Pe de altă parte, mărirea numărului de grupări creşte atât timpul global de antrenament, cât şi spațiul total de memorie ocupat. În concluzie, numărul de grupări ale caracterelor trebuie să satisfacă un compromis în ceea ce priveşte timpul de antrenament şi spațiul de memorie ocupat.

De notat faptul că, pentru o implementare paralelă prima variantă optimizează timpul de răspuns.

Construcția efectivă a ***bazei de date*** pentru fiecare clasificator în parte se face selectând un set semnificativ de caractere extrase din documente, toate aparținând grupării aferente, la care se adaugă simbolurile pentru clasa de rejecție, conform celor de mai sus. Pentru fiecare asemenea caracter se parcurg etapele: ***scalare, DCT-2D, ordonare Zig-Zag şi selecție de trăsături***. Rezultatul este un vector de date, care se va salva într-un fişier pe disc.

Este foarte importantă pentru funcționarea corectă a algoritmilor de antrenare (**LBG** şi backpropagation) ordinea de aplicare a datelor de intrare. Studiile efectuate au arătat că pentru algoritmul **LBG** este de preferat selecția unei ordini aleatoare de aplicare a vectorilor de trăsături corespunzători claselor.

Pentru algoritmul de antrenare a perceptronului multistrat s-a optat pentru o ordine aleatoare de aplicare a vectorilor de date, dar intercalând între oricare doi asemenea vectori un vector de trăsături corespunzător unui caracter din clasa de rejecție. O asemenea construcție reuşeşte să elimine majoritatea erorilor de tipul "***caracter recunoscut greşit***" (confuzie de caractere), ştiut fiind faptul că erorile de tipul "***caracter nerecunoscut***" pot fi semnalate şi deci corectate interactiv în faza de exploatare.

20.9. Adaptarea setului de date de antrenament.

În exploatarea sistemului pot apare noi tipuri de fonturi ce trebuiesc recunoscute. De aceea este necesară actualizarea setului de date de antrenament şi reantrenarea periodică a clasificatorului (rețelelor neurale) folosit.

A adauga noile simboluri la setul de antrenament nu este o soluție eficientă deoarece astfel creşte necondiționat dimensiunea setului de antrenament. S-a impus condiția de limitare a numărului de simboluri din setul de antrenament pentru limitarea spațiului de memorie folosit şi a timpului necesar pentru reantrenare.

Soluția utilizată este de a începe cu un număr fix de simboluri în setul de antrenament şi apoi să se înlocuiască succesiv simboluri din acele perechi de simboluri având cele mai mari scoruri de coincidență, cu noile simboluri ce trebuiesc învățate. Algoritmul a fost testat folosind ***caractere nerecunoscute*** sau ***recunoscute greşit*** din setul de test, iar apoi adăugând în setul de antrenament "greşelile" furnizate de exploatarea sistemului de recunoaştere.

Algoritmul de actualizare a setului de antrenament utilizat se rulează **pentru fiecare grup de clase** în parte, începe după o primă antrenare a rețelei neurale şi a fost descris amănunțit în paragraful precedent.

Se elimină doar un singur simbol din fiecare pereche de simboluri corespunzătoare celor mai bune scoruri de coincidență. Se utilizează pentru aceasta regula:

$$\text{IF } \underset{I\le I\le N}{Min}(D_{K,I}^{(C)}) > \underset{I\le I\le N}{Min}(D_{L,I}^{(C)}) \text{ THEN eliminã simbol "L"}$$

ELSE eliminã simbol "K"

Această strategie selectează cele mai reprezentative simboluri pentru setul de antrenament prin eliminarea acelor simboluri care sunt mai apropiate de vecinii lor din aceeaşi clasă.

20.10. Detalii de implementare, rezultate obținute şi concluzii.

Implementarea metodelor descrise anterior o constituie un program scris în Visual C++ 6.0, care este prezentat succint în continuare. El este un mediu integrat care permite încărcarea doar a imaginilor bitmap Windows(BMP), color, în scară de gri, sau binare.

Deoarece sistemul de recunoaştere propriu-zis lucrează doar cu imagini binare (alb-negru), programul include instrumentele necesare conversiei imaginilor color şi în scară de gri. Imaginile color se convertesc mai întâi în imagini în scară de gri, atribuind fiecărui pixel un nivel de gri egal cu luminanța sa. Conversia imaginilor în scară de gri la imagini binare, adică ***binarizarea imaginilor***, se face semiautomat, prin analiza histogramei. Programul furnizează automat un prag de segmentare, dar utilizatorul îl poate schimba şi verifica în acelaşi timp rezultatul segmentării (folosind butonul Preview).

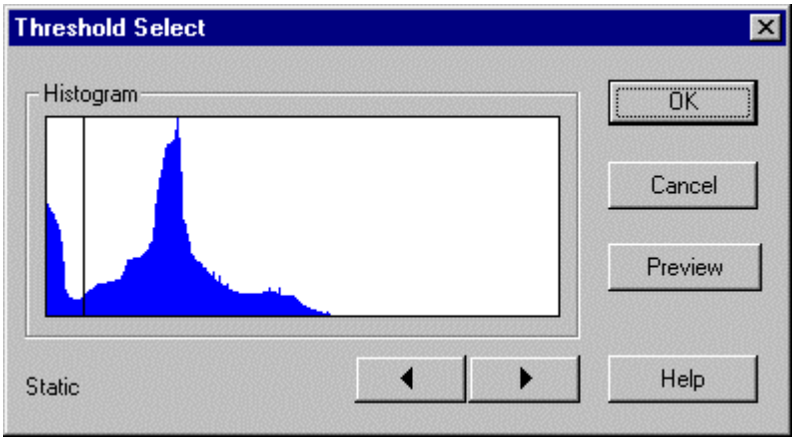


Fig. 12. Binarizarea automată/interactivă a imaginilor.

După obținerea imaginii binare este efectuată ***segmentarea automată a textului*** prin analiza morfologică a obiectelor din imagine. Imaginea următoare prezintă un eşantion de text care va fi folosit în continuare pentru demonstrarea etapelor algoritmului de segmentare.

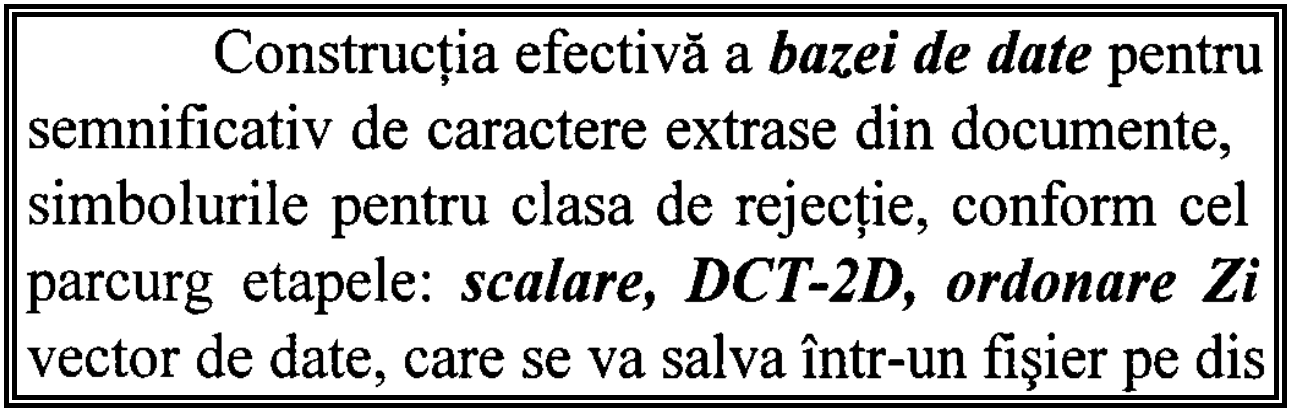


Fig. 13. Imagine de test.

Algoritmul de localizare a obiectelor aplicat imaginii precedente furnizează o listă se structuri de date conținând informația necesară localizării obiectelor din imagine. Algoritmul de urmărire de contur folosit permite în acelaşi timp separarea contururilor exterioare de cele interioare, calculul ariei mărginite de aceste contururi şi eliminarea obiectelor de dimensiune prea mică, apărute de obicei datorită zgomotelor care au afectat imaginea originală.

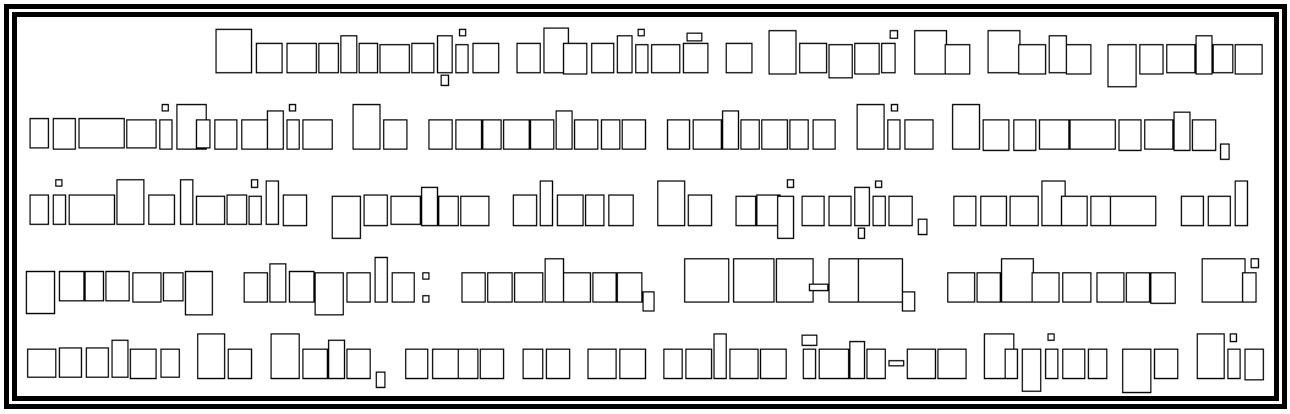


Fig. 14. Localizarea obiectelor din imaginea de test.

Imaginea următoare prezintă rezultatul conectării obiectelor localizate în imagine. Se conectează acele obiecte a căror poziție relativă îndeplineşte setul de condiții descrise anterior. Aici se poate remarca avantajul esențial al acestei metode. Din imagine a fost extrasă în urma localizării toată informația necesară efectuării segmentării şi astfel volumul de date de prelucrat s-a redus dramatic. Segmentarea nu se mai face operând asupra unei imagini ci doar asupra unei liste cu informații despre obiecte.

Conectarea obiectelor poate fi văzută ca fiind construcția unui graf orientat ale cărui noduri sunt obiectele localizate în imagine. Conexiunile dintre obiecte pot fi doar de trei tipuri:

- ***Conexiuni regulare, între caracterele aceluiaşi cuvânt;***
- ***Conexiuni tip separator de cuvinte, între ultimul caracter al unui cuvînt şi primul caracter al cuvântului următor de pe acelaşi rând***
- ***Conexiuni tip ataşament, care unesc obiectele componente ale unui caracter.***

Datorită structurii complexe a textului (caractere de diferite mărimi, formate din două sau mai multe obiecte, grupate în cuvinte, rânduri, paragrafe, coloane), cât şi prezenței liniilor, graficelor şi imaginilor într-un document scanat, în urma conectării obiectelor se pot obține multe conexiuni multiple, care trebuiesc analizate şi eliminate sau etichetate corespunzător.

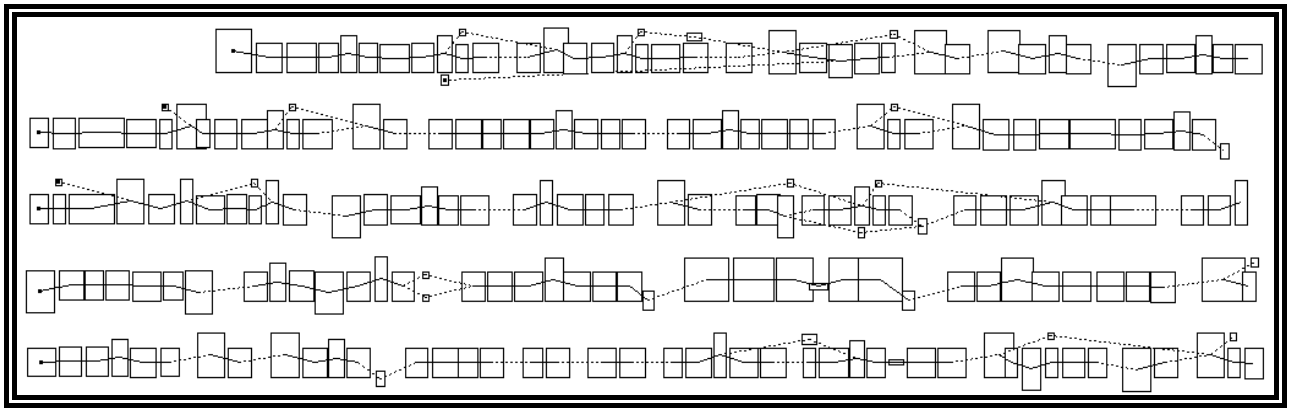


Fig. 15. Conectarea obiectelor din imaginea de test.

În faza următoare se tratează conexiunile divergente, conform regulilor descrise anterior în acest capitol. Ca urmare, după cum se poate remarca în figura următoare, ele vor fi fie eliminate, fie transformate în conexiuni de tip ataşament.

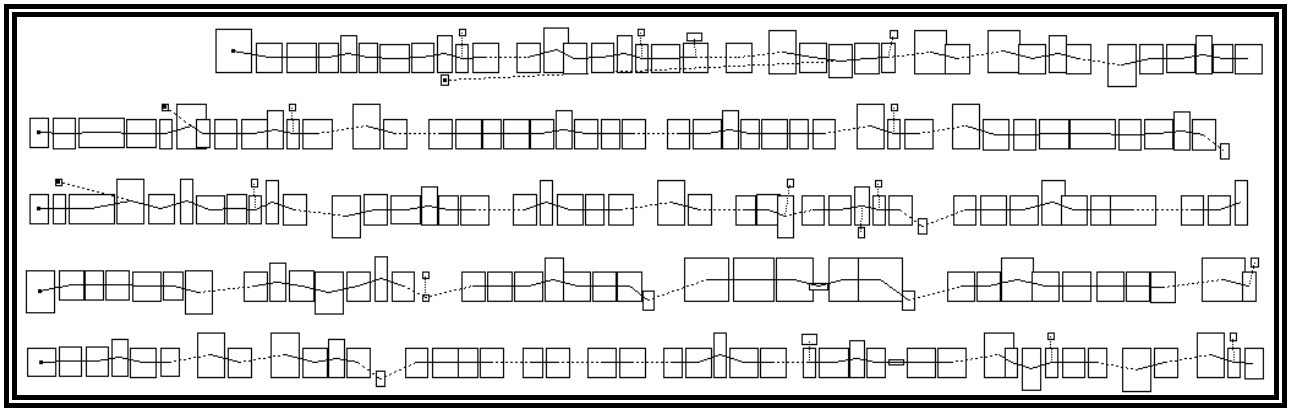


Fig. 16. Tratarea conexiunilor divergente.

În faza finală a algoritmului de segmentare se prelucrează conexiunile convergente, care, ca şi în cazul precedent, fie se transformă în conexiuni de tip ataşament, fie se elimină.

Analiza distanţelor dintre caracterele conectate până acum permite determinarea conexiunilor de tip separator de cuvinte şi eliminarea conexiunilor prea lungi (conexiuni între rânduri coliniare dar din coloane diferite, conexiuni între caractere şi linii, grafice sau imagini prezente în documentul scanat).

Conexiunile multiple rămase după această fază finală trebuiesc eliminate, deoarece ele pot proveni doar de la:

- Formule matematice incluse în text;
- Grafice, linii, imagini incluse în documentul scanat, foarte apropiate de zonele de text.

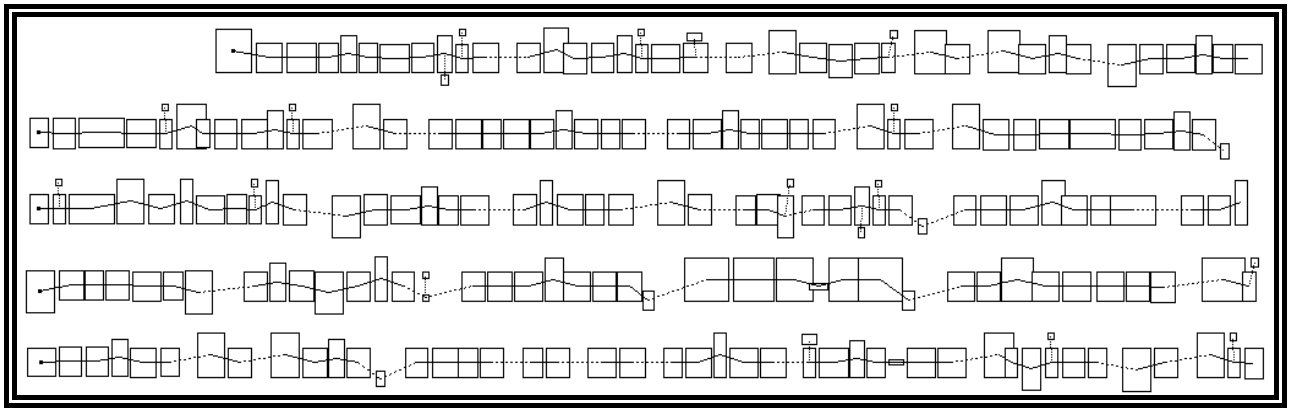


Fig. 17. Tratarea conexiunilor convergente.

Aplicaţia a fost concepută să permită crearea şi întreţinerea unei baze de date de antrenament pentru clasificator. Ca urmare, selectând opţiunea corespunzătoare, după localizarea obiectelor din imagine şi conectarea lor, caracterele (ordonate în ordinea naturală a scrisului) pot fi vizualizate şi apoi etichetate de către utilizator, pentru a forma setul de caractere de antrenament.

Imaginea fiecărui caracter etichetat este salvată sub formă de fişier BMP, eticheta atribuită lui fiind inclusă în numele fişierului. În acest fel utilizatorul poate vizualiza rapid conţinutul bazei de date de antrenament şi îl poate modifica în sensul optimizării lui.

În acest fel fiecare din caracterele etichetate este atribuit unei clase. Unele din obiectele localizate pot fi incluse în clasa de rejecţie, care trebuie să includă orice fel de obiect (fragment de caracter, obiecte formate din mai multe caractere care se ating, etc) a cărui aplicare la intrarea clasificatorului trebuie să dea furnizeze decizia de simbol nerecunoscut.

Fig. 18. Etichetarea caracterelor localizate.

Aplicaţia a fost implementată pe un calculator PC486, 66MHz. Timpul de recunoaştere obţinut în acest stadiu al implementării (localizare, scalare, DCT, reţea neurală) pentru o pagină tipică este de aproximativ 40 secunde.

Este de aşteptat ca noi optimizări să îmbunătăţească considerabil acest timp. Nu se vor face estimări asupra ratei de recunoaştere decât după trecerea unui timp de exploatare a sistemului, cunoscând dependenţa acesteia de componenţa setului de antrenament. Se aşteaptă o îmbunătăţire progresivă a ratei de recunoaştere pe măsura rulării algoritmului de adaptare a setului de antrenament.

Se poate nota dependenţa ratei de recunoaştere de pragul **Th** de decizie. Un prag mare dă un grad mare de încredere în deciziile de recunoaştere corectă, dar măreşte şi probabilitatea erorilor de tipul "caracter nerecunoscut". Un prag optim este în jurul valorii **Th=0.6**.

Pentru micşorarea timpului de răspuns în faza de exploatare când se rulează şi algoritmul de gestiune a setului de antrenament, s-a utilizat următoarea structură ce foloseşte două calculatoare de acelaşi tip, astfel conectate.

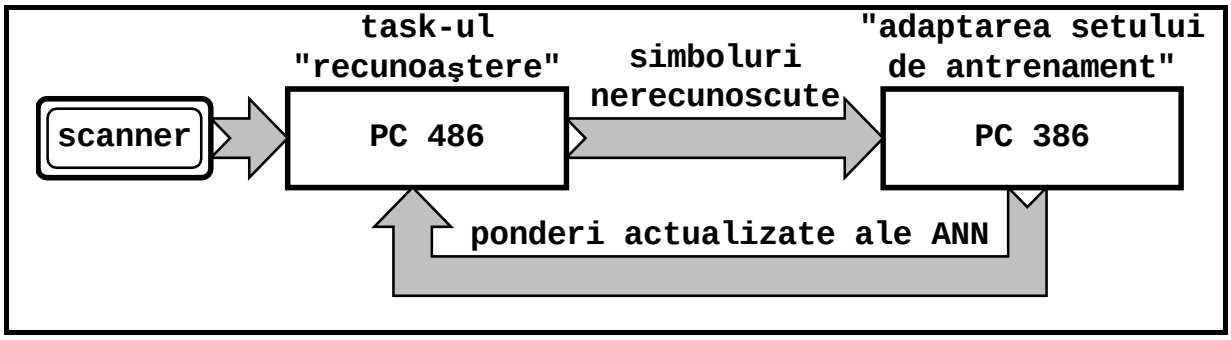
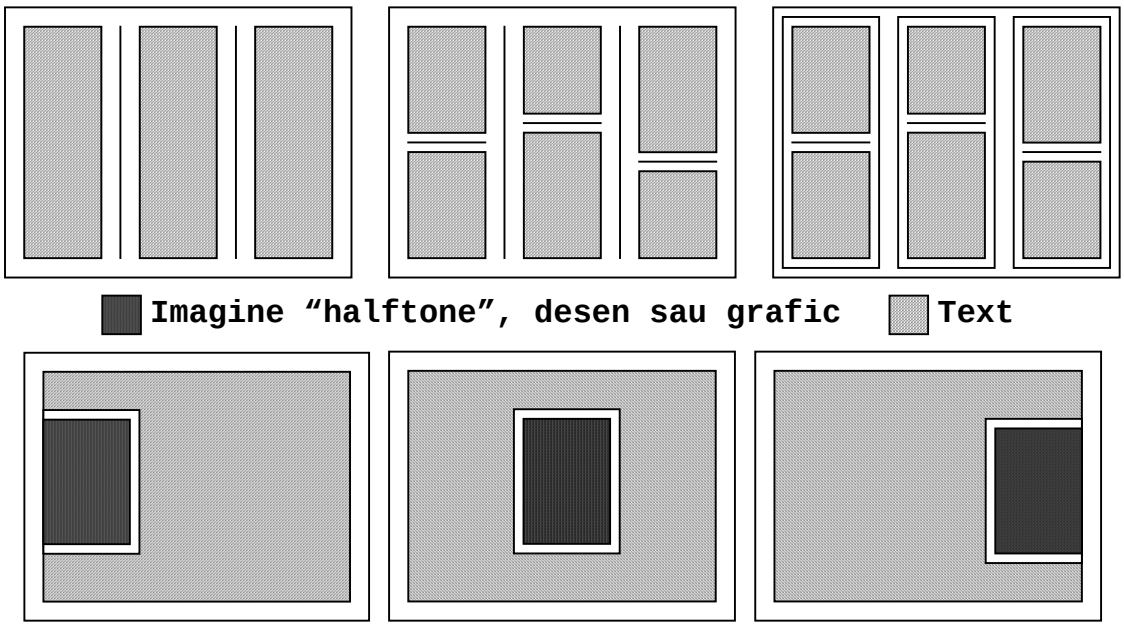


Fig. 18. Utilizarea on-line a algoritmului de adaptare a setului de antrenament.

Rularea aplicaţiei furnizează simboluri nerecunoscute provenite, de obicei, de la un stil nou de documente (alte fonturi, altă calitate a tipăririi). Ele se adaugă setului de test şi apoi se rulează procedura descrisă anterior de adaptare a setului de antrenament. Rezultatele se materializează în actualizarea ponderilor reţelelor neurale care alcătuiesc clasificatorul.



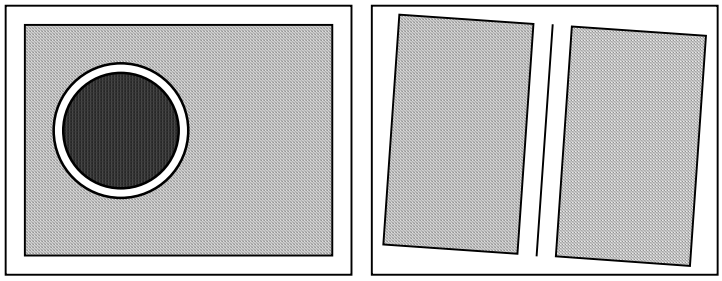


Fig. 19. Imagini tipice de test, prezentate schematic.

În figurile precedente sunt prezentate schematic situaţii tipice de amplasare a textului în imagine, care au fost folosite pentru analiza performanţelor sistemului de recunoaştere, în special ale implementării algoritmului de segmentare a textului. Imaginile de test folosite includ linii verticale şi orizontale de separare a coloanelor de text, dreptunghiuri (casete) care includ coloane de text, imagini/desene înserate în text în diferite poziţii şi zone de text rotite cu un unghi mic faţă de orizontală.

Rândurile de text rotite faţă de orizontală pot apare adesea ca rezultat al amplasării imperfecte a paginilor de text originale la operaţia de scanare. Algoritmul de segmentare a textului prin metoda proiecţiilor eşuează la asemenea imagini, dacă două rânduri succesive nu sunt separate de cel puţin o linie de pixeli albi (altfel proiecţiile lor se suprapun şi segmentarea nu mai este posibilă). În acest caz se folosesc în prealabil algoritmi de "îndreptare a liniilor de text" (deskew), pentru a asigura funcţionarea corectă a metodei proiecţiilor.

Adesea este foarte importantă forma conturului exterior al imaginilor/desenelor înserate în text. Unii algoritmi de segmentare a textului expandează, din motive de viteză, obiectele diferite de text la zone rectangulare. Ca urmare, documente precum cel din ultimul desen (Fig. 19) sunt segmentate incorect de majoritatea sistemelor recunoaştere existente.

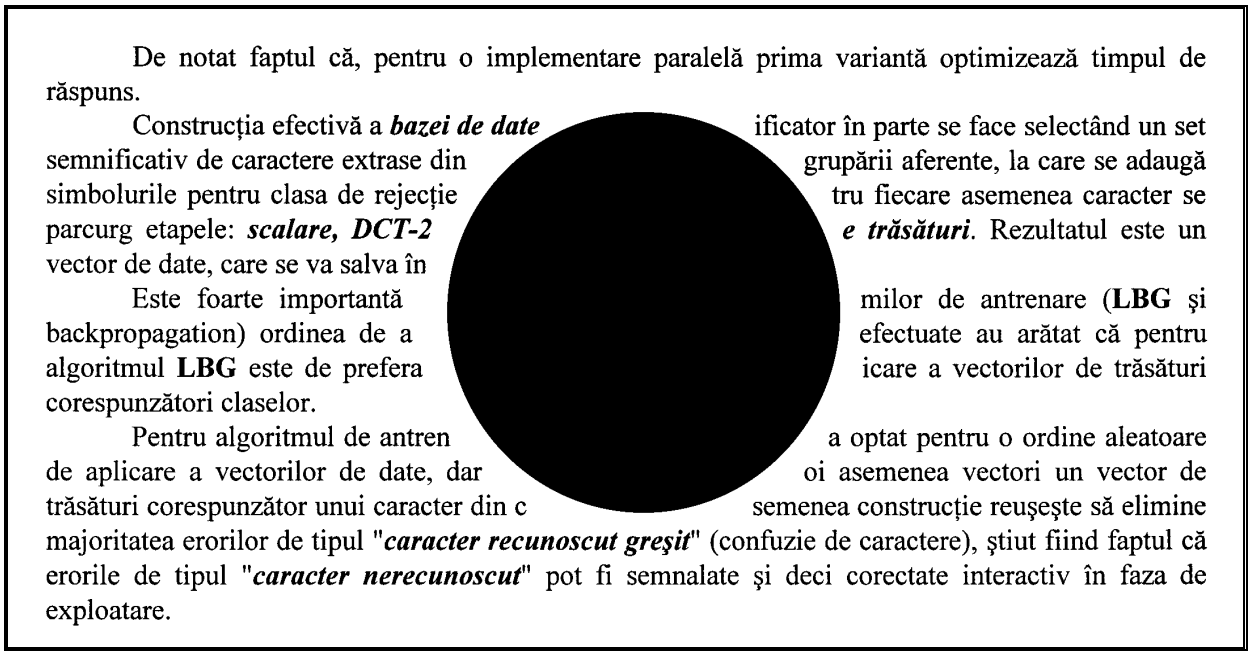
Parametru/Metodă	**Metoda proiecţiilor**	**Analiză de textură**	***Analiză morfologică***
Calitate segmentare	**Calitate slabă**	**Calitate medie**	***Calitate foarte bună***
Viteză de răspuns	**Foarte rapidă (3sec)**	**Foarte lentă (40sec)**	***Viteză medie (10sec)***
Precizie segmentare	**Precizie foarte bună**	**Precizie slabă**	***Precizie foarte bună***
Rânduri înclinate	**Îndreptare rânduri(3sec)**	**Îndreptare rânduri(3sec)**	***Imunitate completă***

Tabelul precedent prezintă sintetic o comparaţie a principalilor parametri (calitate, viteză, precizie, comportare la rânduri de text înclinate) care caracterizează diferiţi algoritmi de segmentare de text de referinţă înglobaţi în diferite produse comerciale sau profesionale de recunoaştere de texte –OCR (Optical Character Recognition). Calitatea medie a segmentării realizate de algoritmul bazat pe analiza de textură se referă la incapacitatea algoritmului de a detecta liniile din imagine, mai ales dacă acestea sunt apropiate de zonele de text. În schimb algoritmul se comportă foarte bine la discriminarea diferitelor zone de text între ele (text scris cu caractere regulare / **bold** / *italic* / ***bold italic*** de dimensiuni apropiate). Viteza medie realizată de algoritmul de segmentare a textului prin analiză morfologică poate fi îmbunătăţită sensibil printr-o implementare optimizată (prezenta implementare se bazează pe utilizarea intensivă a clasei CPtrList din Microsoft Foundation Classes, foarte lentă).

Figura următoare (Fig. 20) prezintă un caz tipic de document scanat – un obiect înserat în text, în aşa fel încât practic textul înconjoară obiectul. Forma nerectangulară (discoidală în acest caz) a obiectului, constituie un test dificil pentru alţi algoritmi de segmentare a textului, majoritatea eşuând.

Algoritmii de segmentare bazaţi pe metoda proiecţiilor, urmărind cel mai adesea o implementare de mare viteză, nu realizează localizarea obiectelor din imagine şi ca urmare, deşi obiectul precedent ar putea fi eliminat din imagine folosind criterii dimensionale, algoritmul etichetează drept text doar primele două şi ultimele trei rânduri din imagine, restul documentului fiind categorisit drept grafic.

Produsele OCR testate au fost Recognita (Caere), Omnipage, Corel. Fiecare din ele prezintă atât variante comerciale cât şi profesionale de produse. Nucleul sistemului de recunoaştere înglobat în aceste produse nu variază esenţial de la o variantă la alta. Opţiunile suplimentare furnizate de variantele profesionale se referă la seturi extinse de caractere pentru limbi diferite, gamă extinsă de formate de fişiere de intrare (imagini), gamă extinsă de fişiere de ieşire (compatibile cu diferite editoare de text), posibilitatea utilizării într-un mediu de producţie, etc. Toate aceste produse de referinţă au eşuat în segmentarea imaginilor următoare.



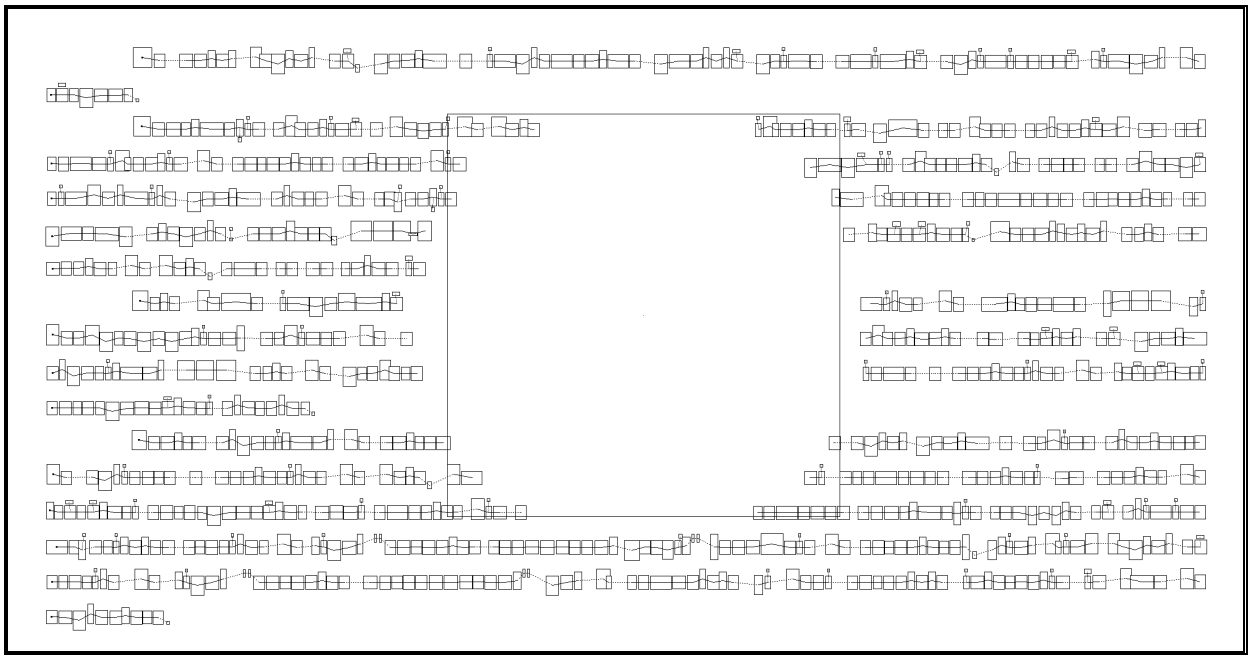


Fig. 20. Imagine de test şi rezultatul segmentării ei cu metoda propusă.

De asemenea rezultate eronate au furnizat OCR-urile de referinţă pentru imaginile următoare, în timp ce algoritmul propus a funcţionat corect, confirmând imunitatea algoritmului la rotaţia imaginii cu unghiuri mici. Imaginea de test reprezintă un paragraf text având rândurile înclinate cu un unghi de 5^0 în sens trigonometric.

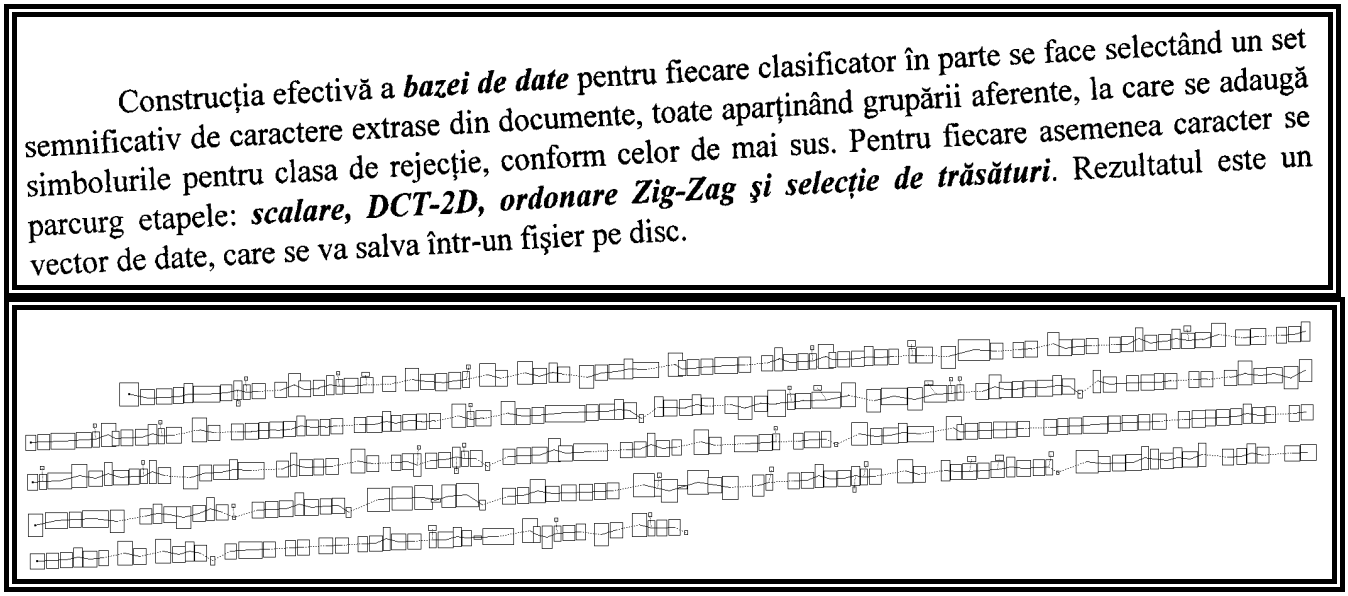


Fig. 21. Imagine de test şi rezultatul segmentării ei cu metoda propusă.

Testele au confirmat robusteţea metodelor de segmentare şi recunoaştere prezentate şi eficacitatea algoritmului de adaptare a setului de antrenament. Algoritmul de segmentare a textului se comportă calitativ mult mai bine decât orice produs similar existent. El este la fel de precis sau chiar mai precis decât algoritmii folosiţi drept referinţă. Deşi timpul de răspuns este mai mare, trebuie ţinut cont că metoda prezentată elimină necesitatea aplicării algoritmului de "îndreptare a rândurilor de text", iar implementarea realizată nu este complet optimizată.

De asemenea a fost confirmată ideea supradimensionării iniţiale a reţelei neurale, tocmai în scopul de a putea învăţa noi simboluri pe durata exploatării, folosind algoritmul de adaptare descris. Trăsătura de bază a acestui algoritm este capacitatea lui de a mări rata de recunoaştere prin construirea unui set de antrenament reprezentativ.

Extensia metodei prezentate la alte aplicaţii de recunoaştere de obiecte în imagini binare este posibilă. În plus este posibilă proiectarea unei structuri hardware dedicate care să rezolve acest gen de probleme. O structură sistolică care să utilizeze un procesor dedicat fiecărui pas de prelucrare (scalare, DCT-2D, ordonare Zig-Zag şi clasificatori neurali) pare a fi o soluţie bună şi foarte rapidă.

Aplicaţia dezvoltată pentru testare, deşi funcţională, nu este încă terminată. Este necesară optimizarea vitezei implementării curente a algoritmului de separare a zonelor de text din imagine. Îmbunătăţirea performanţelor clasificatorului se obţine, în condiţii de exploatare a aplicaţiei, prin rularea algoritmului de gestiune a setului de date de antrenament.

CD-ul ataşat conţine atât codul sursă cât şi programul executabil. O descriere a modului de utilizare a programului poate fi găsită în anexe.

RETELE NEURONALE

Curs 1: Introducere în calculul inteligent

1 Calculul inteligent si rezolvarea problemelor

Din punctul de vedere al rezolvării lor automate, problemele pot fi clasificate în două categorii:

- *Probleme "bine-puse":* caracterizate prin faptul că li se poate asocia un model formal (de exemplu, un model matematic) pe baza căruia se poate dezvolta o metoda de rezolvare cu caracter algoritmic. În această categorie intră probleme clasice în informatică cum ar fi problema căutării într-o bază de date cu informaţii despre persoane folosind drept cheie de căutare codul numeric personal.

- *Probleme "rău-puse":* caracterizate prin faptul că nu pot fi descrise complet printr-un model formal, ci cel mult se cunosc exemple de rezolvare a problemei. O problemă similară celei de mai sus dar care poate fi considerată "rău pusă" este aceea a căutării în baza de date folosind drept indiciu o poză a persoanei căutate.

Rezolvarea unei probleme înseamnă stabilirea unei asocieri între datele de intrare (valori iniţiale, ipoteze etc.) şi răspunsul corect. În cazul problemelor bine-puse această asociere este o relaţie funcţională explicită construită pe baza modelului asociat problemei.

În cazul problemelor rău-puse, însă, o astfel de relaţie explicită nu poate fi pusă în evidenţă, rolul sistemului care rezolvă problema fiind de a dezvolta o relaţie de asociere întrebare-răspuns pe baza unor exemple. Procesul prin care sistemul îsi formează modelul propriu al problemei şi pe baza acestuia relaţia de asociere se numeşte *adaptare* sau chiar *învăţare*.

Pe de altă parte, din punctul de vedere al complexităţii rezolvării şi al relevanţei răspunsului problemele pot fi clasificate în:

- Probleme pentru care este esenţială obţinerea unui răspuns exact indiferent de resursele implicate. Acestea necesită utilizarea unor tehnici exacte.

- Probleme pentru care este preferabil să se obţină un răspuns "aproximativ" folosind resurse "rezonabile" decât un răspuns exact dar folosind resurse foarte costisitoare.

Calculul inteligent este un domeniu al Inteligenţei Artificiale care grupează tehnici de rezolvare a problemelor "rău-puse" sau a celor pentru care modelele formale conduc la algoritmi foarte costisitori.

Principalele direcţii ale calculului inteligent sunt:

- *Calcul neuronal.* Este folosit în principal în rezolvarea problemelor de asociere (clasificare, aproximare, predicţie etc), bazându-se pe extragerea, prin învăţare, a unui model pornind de la exemple. Sursa de inspiraţie o reprezintă structura şi funcţionarea creierului.

- *Calcul evolutiv.* Este folosit în principal în rezolvarea problemelor bazate pe căutarea soluţiei într-un spaţiu mare de soluţii potenţiale (în particular în rezolvarea problemelor de optimizare). Sursa de inspiraţie o reprezintă principiile evoluţionismului de tip darwinist.

- *Calcul fuzzy.* Este folosit atunci când datele problemei (şi relaţiile dintre acestea) nu pot fi descrise exact ci există un grad de incertitudine ("fuzziness"). Ideea de bază este de a înlocui valorile exacte ("crisp") cu valori fuzzy descrise prin funcţii de apartenenţă.

În fiecare dintre cele trei direcţii majoritatea prelucrărilor care se efectuează au *caracter numeric*, fiind necesară o codificare numerică adecvată a problemei. Aceasta motivează parţial prezenţa cuvântului *calcul* în denumirea domeniului. Pe de altă parte în fiecare dintre direcţiile de mai sus se încearcă simularea unor comportamente inteligente ceea ce motivează prezenţa termenului *inteligent*.

Principiul fundamental al calculului neuronal şi al celui evolutiv este de a dezvolta sisteme de calcul inteligent pornind de la implementarea unor *reguli simple*, comportamentul complex al acestor sisteme derivând din aplicarea în *paralel* şi în *manieră interactivă* a acestor reguli. Această abordare de tip bottom-up este în contrast cu abordarea de tip "top-down" specifică altor abordări din Inteligenţa Artificială.

Calculul neuronal şi cel evolutiv fac parte din sfera mai largă a *calculului natural* al cărui principiu este de a prelua idei de rezolvare a problemelor din sistemele naturale (fizice, chimice, biologice, ecologice). Obiectivul principal al calculului natural este de a dezvolta metode de rezolvare a problemelor rău-puse şi a celor nerezolvabile prin metodele tradiţionale (de exemplu, rezolvarea problemelor NP-complete în timp polinomial).

Pe lângă componentele amintite deja, calculul natural mai include *calculul molecular* (DNA Computing), *calculul cu membrane* (Membrane Computing) şi *calculul cuantic* (Quantum Computing). Dacă primele două direcţii sunt deja tradiţionale, ultimele trei sunt încă in primele faze de dezvoltare. O scurtă trecere în revistă a sursei de inspiraţie coresponzătoare acestor domenii este prezentată în tabelul următor.

Sursa de inspiraţie	Model de calcul (soft)	Implementare (hard)
Creier	Calcul neuronal	Electronică
ADN	Calcul evolutiv	Electronică
ADN	Calcul molecular	Biologică
Celula	Calcul membranar	Electronică, biologică
Sistem fizic	Calcul cuantic	Nanoelectronică

2 Specificul calculului neuronal

Din punct de vedere *funcţional* o reţea neuronală este un sistem ce primeşte date de intrare (corespunzătoare datelor iniţiale ale unei probleme) şi produce date de ieşire (ce pot fi interpretate ca răspunsuri ale problemei analizate). O caracteristică esenţială a reţelelor neuronale este capacitatea de a se adapta la mediul informaţional corespunzător unei probleme concrete printr-un proces de învăţare. În felul acesta reţeaua extrage modelul problemei pornind de la exemple.

Din punct de vedere *structural* o reţea neuronală este un ansamblu de unităţi interconectate (neuroni) fiecare fiind caracterizată de o funcţionare simplă. Funcţionarea unităţilor este influenţată de o serie de parametri adaptabili. Astfel o reţea neuronală este un sistem extrem de flexibil.

Structura unităţilor funcţionale, prezenţa conexiunilor şi a parametrilor adaptivi precum şi modul de funcţionare sunt inspirate de creierul uman. Fiecare unitate funcţională primeşte câteva semnale de intrare pe care le prelucrează şi produce un semnal de ieşire. Interacţiunea multor unităţi cu funcţionare simplă conduce la un sistem care poate să rezolve probleme complexe. Funcţionarea sistemului este controlată de un set numeros de parametri ajustabili care permit acestuia să se adapteze cât mai fidel mediului informaţional în care este amplasat (specific problemei de rezolvat). Una dintre cele mai importante caracteristici ale unui sistem neuronal este caracterul său adaptiv, faptul că poate să-şi stabilească parametrii de funcţionare printr-un proces de învăţare bazat pe informaţiile primite. Astfel de sisteme sunt adecvate, astfel, pentru problemele ce sunt dificil sau chiar imposibil de formalizat pentru ele existând doar exemple de rezolvare.

2.1 Motivaţia biologică

În încercarea de a proiecta sisteme inteligente cel mai la îndemână model este chiar creierul uman. Acesta este capabil să prelucreze cantităţi mari de date la un moment dat şi surclasează în mod cert calculatoarele (în special cele seriale) în probleme complexe de tipul înţelegerii scenelor (cum ar fi recunoaşterea unei imagini familiare într-un mediu necunoscut).

Din punct de vedere biologic creierul este constituit dintr-un număr mare de celule (neuronii, cca $10^{10} \div 10^{12}$) care efectuează sarcini simple şi la o viteză nu prea mare (timpul de răspuns la un stimul este de aproximativ 10^{-3}s) dar care sunt puternic interconectate (există cca $10^{14} \div 10^{15}$ interconexiuni) şi lucrează în paralel. Având în vedere faptul că, componentele electronice care stau la baza calculatoarelor actuale au timpi de răspuns mult mai mici (10^{-9}s) şi totuşi sunt surclasate de către creier în rezolvarea unor probleme complexe (vedere, decizii pe baza unor date incomplete etc.), rezultă că puterea computaţională a creierului rezidă în faptul că bilioane de neuroni operează *simultan*. Evident, ar fi de dorit realizarea de sisteme care să lucreze cu viteza componentelor electronice şi să fie caracterizate de conectivitatea creierului.

Dintre caracteristicile creierului care sunt de dorit şi în sistemele artificiale pot fi enumerate: robusteţe şi toleranţă la erori (mor zilnic neuroni, fără ca aceasta să afecteze *semnificativ* performanţele creierului), flexibilitate (fiinţele sunt capabile să se adapteze la un nou mediu prin învăţare), capacitatea de a prelucra informaţie incompletă, nedeterministă sau chiar inconsistentă.

În ceea ce priveşte robusteţea şi toleranţa la erori ea este asigurată de faptul că informaţia este reprezentată în mod distribuit şi nu localizat. Astfel, din punct de vedere cibernetic creierul este un *sistem natural de prelucrare paralel-distribuită a informaţiei*.

Reţelele neuronale pot fi văzute atât ca modele ale creierului cât şi ca sisteme de prelucrare a informaţiei şi rezolvare a problemelor. Acestea sunt două direcţii distincte în domeniul reţelelor neuronale fiind diferite atât din punct de vedere al tehnicilor utilizate cât şi din punct de vedere al aplicaţiilor.

În ceea ce urmează ne vom referi doar la reţele neuronale artificiale destinate rezolvării unor probleme computaţionale.

2.2 Structura unei reţele neuronale artificiale

O reţea neuronală artificială este un ansamblu de unităţi funcţionale amplasate în nodurile unui graf orientat. De-a lungul arcelor grafului circulă semnale care permit unităţilor funcţionale să comunice între ele . Elementele definitorii ale unei reţele neuronale sunt:

- *Arhitectura:* specifică modul în care sunt amplasate şi interconectate unităţile funcţionale. Arhitectura determină şi fluxul informaţional în cadrul reţelei.

- *Funcţionarea:* specifică modul în care fiecare unitate în parte şi reţeaua în ansamblul ei transformă semnalele de intrare în semnale de ieşire. Funcţionarea este influenţată de arhitectură, în special de modul de interconectare a unităţilor.

- *Adaptarea (învăţarea)*: specifică modul de stabilire a parametrilor ajustabili astfel încât reţeaua să poate rezolva anumite probleme. În funcţie de natura informaţiei de care se dispune, învăţarea poate fi *supervizată* sau *nesupervizată*.

Învăţarea constă în modificarea funcţionalităţii reţelei prin modificarea parametrilor şi/sau a structurii acesteia. Procesul de învăţare bazat pe adaptarea parametrilor constă în existenţa unor *reguli de modificare* a parametrilor şi a unui algoritm (de regulă iterativ) de aplicare a acestor reguli. Structura generală a unui algoritm de învăţare bazat pe modificarea parametrilor este:

```
// initializarea (aleatoare) a parametrilor retelei
  init W(0)
// initializarea indicatorului de iteratie
  t=0
// prelucrare repetitiva
  repeat
    W(t+1)=adjust(W(t),X(t)[,d(t)])
    t=t+1
  until <criteriu de stop>
```

unde $W(t)$ reprezintă valorile parametrilor la momentul t, $X(t)$ reprezintă semnalul de intrare, iar $d(t)$ semnalul de învăţare corespunzător (poate fi chiar răspunsul corespunzător intrării $X(t)$ sau doar un indicator de corectitudine/eroare a răspunsului dat de reţea). În cazul învăţării nesupervizate $d(t)$ poate lipsi din regula de învăţare.

2.3 Ce pot şi ce nu pot face reţelele neuronale

Ce pot face reţelele neuronale ? În principiu, pot reprezenta orice funcţie calculabilă. Ceea ce este însă mai important este că pot învăţa orice astfel de funcţie pornind de la exemple. Nu sunt însă eficiente pentru orice problemă, ele fiind utile în rezolvarea problemelor pentru care se dispune de un număr mare de exemple şi pentru care nu pot fi găsite uşor şi rapid reguli de formalizare. Abilitatea unei reţele neuronale de a rezolva o anumită problemă este influenţată şi de modul în care sunt pregătite datele de intrare. Fără o preprocesare atentă a datelor de intrare demersul rezolvării problemei cu reţele neuronale poate fi sortit eşecului.

2.4 Clase de probleme ce pot fi rezolvate cu reţele neuronale

În continuare sunt prezentate câteva dintre categoriile de probleme pentru care reţelele neuronale au fost aplicate cu succes.

Clasificare şi recunoaştere.

Date de intrare. Descriere sintetică a unui obiect (de exemplu descrierea grafică a unei litere sau ansamblul caracteristicilor acesteia). În majoritatea situaţiilor descrierea este un vector de valori numerice obţinute printr-o prelucrare prealabilă (*preprocesare*) a informaţiilor brute.

Date de ieşire. Indicator al clasei căreia îi aparţine obiectul (de exemplu numărul de ordine al literei în cadrul alfabetului).

Abilitatea de clasificare a reţelei este rezultatul unui proces de învăţare pornind de la exemple de clasificare corectă.

Pe lângă exemplul de mai sus, de recunoaştere a caracterelor, alte probleme concrete de clasificare sunt: recunoaşterea vorbirii, clasificarea semnalelor (de exemplu separarea electrocardiogramelor în normale şi anormale), clasificarea celulelor în normale şi anormale, recunoaşterea unor feţe într-o imagine, clasificarea texturilor etc.

Gruparea şi categorizarea datelor. Este similară problemei de clasificare cu diferenţa că antrenarea reţelei se realizează pornind doar de la date de intrare, fără a specifica clasele cărora le aparţin. Clasele sunt construite pornind doar de la similarităţile existente în datele de intrare pe care reţeaua este capabilă să le identifice. În felul acesta reţeaua neuronală descoperă criteriul de grupare. Probleme concrete din această categorie intervin în analiza datelor (în special în domeniul cunoscut sub denumirea de "data mining") şi în compresia datelor.

Aproximare şi estimare. Se referă la la extragerea dependenţei funcţionale dintre două mărimi pornind de la un set de valori ale celor două mărimi. De regulă valorile din set sunt alterate de zgomot (afectate de erori de măsurare sau de altă natură). Folosind acest set reţeaua este antrenată pentru a determina dependenţă dintre cele două mărimi. O dată reţeaua antrenată, pentru orice valoare a argumentului ea furnizează aproximarea valorii asociate. O problemă concretă din aceasta clasă este reprezentată de determinarea dependenţei funcţionale între mărimi măsurate experimental. În aceeaşi clasă se încadrează determinarea parametrilor unor modele din inginerie sau orice altă problemă de asociere. În acest context reţelele neuronale pot fi văzute ca modele de regresie neliniară caracterizate printr-un număr mare de parametri.

Predicţie.

Date de intrare. O succesiune de valori (numită *serie temporală*) pentru care nu este cunoscută o relaţie formală care să le genereze.

Dată de ieşire. Aproximarea următoarei valori din serie.

Antrenarea reţelei se realizează pornind de la valorile cunoscute din serie. Probleme concrete din această clasă sunt predicţia evoluţiei stocurilor, predicţia în meteorologie, predicţie în evoluţia vânzărilor etc.

Optimizare. Diferite probleme din ştiinţă, inginerie şi economie pot fi formulate ca probleme de optimizare constând în necesitatea determinării unei valori care satisface anumite restricţii şi optimizează o funcţie obiectiv. Reţelele neuronale sunt adecvate pentru rezolvarea problemelor de optimizare dificile pentru care este suficient să se obţină soluţii suboptimale. Probleme concrete din această categorie sunt cele ce intervin în proiectarea circuitelor electronice, în alocarea resurselor, în rezolvarea problemelor de rutare în reţele etc.

Stocarea şi regăsirea informaţiei după conţinut. Reţelele neuronale permit stocarea informaţiei astfel încât aceasta să fie ulterior regăsită pornind de la indicii legate de conţinut (indiciile pot fi porţiuni din informaţia stocată). Astfel de sisteme de stocare sunt mai tolerante la erori în raport cu memoriile bazate pe adrese. Aplicaţii concrete sunt în proiectarea bazelor de date multi-media.

Modelare şi control adaptiv. Controlul unui sistem dinamic se referă la a determina un semnal de control, $u(t)$, care asigură producerea unui semnal de ieşire dorit $y(t)$. Reţeaua neuronală este antrenată pornind de la un model de referinţă.

Prelucrarea şi analiza semnalelor.

Date de intrare. Un semnal, care poate fi o imagine sau un semnal sonor.

Date de ieşire. Semnalul transformat (de exemplu prin eliminarea zgomotului) sau o informaţie extrasă din cadrul lui.

2.5 Descriere a unor aplicaţii clasice

În continuare sunt enumerate câteva dintre problmele pentru care reţelele neuronale au fost aplicate

cu succes încă din primii ani de dezvoltare.

2.5.1 Eliminarea adaptivă a zgomotului - "adaptive noise canceling"

La transmiterea unui semnal (sonor sau de altă natură) printr-un canal de comunicaţie (de exemplu o linie telefonică) intervine un zgomot care afectează calitatea transferului. Pentru oameni a separa un sunet dintr-un anumit context sonor (de exemplu a înţelege ceea ce spune o persoană într-o încăpere în care se aude şi muzică) nu este o problemă dificilă. Aceasta este o abilitate de a elimina zgomotul pe care o utilizăm fără să ne gândim explicit la ceea ce facem. În mod analog dacă în timpul unei convorbiri telefonice intervine un zgomot pe linie îl putem ignora deşi zgomotul respectiv nu are o anumită caracterizare. Tocmai datorită acestui fapt este foarte dificil dacă nu imposibil să separăm prin metodele clasice zgomotul de sunetul util. Oamenii sunt capabili să ignore acest zgomot deoarece au învăţat această abilitate din experienţă. Acesta este motivul pentru care această problemă a fost rezolvată utilizând reţele neuronale încă din primii ani ai dezvoltării lor (1950-1960, reţeaua ADALINE proiectată de B. Widrow).

2.5.2 Evaluarea riscului creditului ipotecar

Unul dintre cele mai importante aspecte pe care o bancă trebuie să le ia în considerare înaintea aprobării unui credit unei persoane este evaluarea riscului ca persoana să nu fie capabilă să returneze împrumutul. O reţea antrenată pe baza a câtorva mii de informaţii privind cererile acceptate sau respinse a obţinut o rată de succes de 98% (în timp ce rata de succes a funcţionarilor este în jur de 85%).

2.5.3 Cititor de coduri poştale

Serviciul poştal din SUA manevrează volume mari de corespondenţă. Una dintre sarcinile angajaţilor este clasificarea corespondenţei pe baza codului poştal. Un sistem care asigură clasificarea pe baza codului poştal rezolvă următoarele subprobleme:

- identificarea zonei în care este scris codul;
- împărţirea acestei zone în subzone mai mici care se suprapun pentru izolarea cifrelor.
- recunoaşterea cifrelor;

A doua etapă este repetată în paralel cu recunoaşterea cifrelor până când sunt recunoscute toate cifrele. Reţelele neuronale sunt folosite de regulă în etapa de recunoaştere propriu-zisă. Rata de succes a acestei aplicaţii este de 95%. Ea a reprezentat punctul de pornire pentru aplicaţiile destinate recunoaşterii scrisului de mână (deşi rata de succes a acestor aplicaţii este mult mai mică).

2.5.4 Recunoaşterea vorbirii

Pentru oameni, recunoaşterea vorbirii pare ceva intuitiv dar a realiza o aplicaţie care să facă acest lucru nu mai este deloc un lucru simplu. La începutul anilor 1980, Intel a proiectat o aplicaţie bazata pe o reţea neuronală artificială care putea să identifice vocea unei persoane pe baza unui vocabular limitat. Acest sistem, cunoscut sub numele de sistem de recunoaştere automată a vorbirii, poate identifica vocea unei persoane ce foloseşte un vocabular limitat pe baza căruia a fost antrenată reţeaua de către acea persoană. Această aplicaţie este utilizată în liniile de asamblare unde un inspector poate comunica cu calculatorul printr-o interfaţă vocala pentru a controla procesul de asamblare.

2.5.5 Clasificarea semnalelor produse de "sonar"

"Sonar"-ul este un echipament care permite detectarea obstacolelor submarine. Utilizarea echipamentului necesită un operator experimentat care să recunoască adevăratele obstacole în raport cu cele false. A fost realizată o aplicaţie bazată pe o reţea neuronală antrenată pe baza unor date ale sonarului înregistrate din cât mai multe direcţii posibile. Sistemul a reuşit să identifice 98% dintre obstacolele din setul de antrenare şi 90% din datele noi.

2.6 Calculul neuronal în comparaţie cu calculul clasic

Între abordarea problemelor cu ajutorul reţelelor neuronale artificiale şi cea clasică bazată pe implementarea unui algoritm pornind de la un model formal există o serie de diferenţe. Câteva sunt marcate în continuare.

	Prelucrare neuronală	+/−	*Prelucrare algoritmică*	+/−
1	Foarte multe procesoare fiecare executând un program simplu.	+	Unul sau câteva procesoare executând programe complicate.	−
2	Robusteţe la erori.	+	Un singur bit eronat poate compromite un program.	−
3	Pot rezolva probleme a căror structură logică nu este pe deplin lămurită.	+	Este necesară buna cunoaştere a problemei pentru a o descompune în unităţi logice.	−
4	Sunt sisteme adaptive (se pot adapta la mediul informaţional, pot învăţa din experienţă).	+	Sunt entităţi îngheţate destinate rezolvării unei clase bine precizate de probleme.	−
5	Nu oferă garanţia corectitudinii răspunsului pe care-l dau.	−	Un algoritm corect asigură corectitudinea răspunsului.	+
6	Nu furnizează explicaţia răspunsului pe care-l dau şi sunt dificil de testat.	−	Există metode de testare a algoritmilor şi programelor.	+
7	Produc soluţii aproximative.	−	Produc soluţii exacte.	+

Pe de altă parte pot fi puse în evidenţă, în aceeaşi manieră diferenţele care există între reţelele neuronale si sistemele expert:

	Reţele neuronale	*Sisteme expert*
1	Imită structura şi funcţionarea creierului.	Imită raţionamentul uman.
2	Prelucrare paralelă a informaţiei.	Prelucrare secvenţială a informaţiei.
3	Cunoştinţele sunt reprezentate implicit.	Cunoştinţele sunt reprezentate explicit.
4	Implementează raţionamente de tip inductiv	Implementează raţionamente de tip deductiv
1	Prelucrare preponderent numerică a informaţiei.	Prelucrare preponderent simbolică a informaţiei.
2	Reguli deduse prin învăţare.	Reguli încorporate aprioric în sistem.
3	Fără capacitate explicativă	Posedă modul explicativ.

2.7 Motivaţii pentru studiul reţelelor neuronale

Există mai multe motive de a studia reţelele neuronale. Dintre acestea amintim:

1. Sunt o *alternativă* la paradigma computaţională clasică bazată pe utilizarea unui model formal şi pe proiectarea unor algoritmi a căror comportare nu se modifică pe parcursul utilizării.

2. Integrează rezultate din discipline variate în scopul obţinerii unor arhitecturi simple de calcul. Reformularea unitară, în cadrul teoriei reţelelor neuronale, a unor tehnici deja clasice dar aparţinând unor diverse domenii asigură acoperirea unor zone de graniţă, fapt important în condiţiile în care interdisciplinaritatea câştigă tot mai mult teren.

3. Modelează inteligenţa umană, putând contribui la o mai bună înţelegere a modului cum funcţionează creierul uman.

2.8 Scurt istoric

În continuare sunt enumerate câteva dintre principalele etape parcurse în evoluţia reţelelor neuronale artificiale:

[McCullough and Pitts, 1943] Au propus primul model formal al neuronului punând în evidenţă abilitatea de calcul a acestuia şi posibilitatea de a imita funcţionarea acestuia prin circuite electrice.

[Hebb, 1949] A enunţat principiul adaptării permeabilităţii sinaptice conform căruia de fiecare dată când o conexiune sinaptică este "folosită" permeabilitatea ei creşte. Acest principiu stă la baza adaptării prin modificarea ponderilor sinaptice.

[Rosenblatt, 1957] A dezvoltat o reţea implementată hard, numită perceptron, pentru clasificarea caracterelor tipărite.

[Widrow şi Hoff, 1950-1960] Au dezvoltat algoritmi de învăţare bazaţi pe minimizarea erorii pe setul de antrenare pentru reţele cu un nivel de unităţi funcţionale (ADALINE - ADaptive LINear Element şi MADALINE - Multiple ADaptive LINear Element).

[Minsky şi Papert, 1969] Au publicat cartea "Perceptrons" în care se pun în evidenţă limitele reţelelor cu un singur nivel de unităţi funcţionale. Reprezintă finalul primei etape de dezvoltare a reţelelor neuronale după care a urmat o perioadă de circa 10 ani în care cercetările în domeniu au fost reduse considerabil.

[Hopfield, 1982] Propune o abordare a analizei reţelelor neuronale folosind formalismul fizicii statistice prin punerea în evidenţă a analogiei între reţelele recurente (destinate memorării asociative) şi sistemele de spini magnetici . Marchează începutul unei noi perioade de interes in domeniu caracterizată prin extinderea domeniilor de aplicabilitate şi volumul mare de implementări soft şi hard folosite în aplicaţii practice.

[Rumelhart, Parker, 1985] Este descris, într-o manieră accesibilă, algoritmul de învăţare al reţelelor cu mai multe nivele bazat pe ideea minimizării unei funcţii de eroare calculată pornind de la un set de antrenare (ideea a fost formulată anterior de Werbos însa a devenit cunoscută abia după prezentarea într-o manieraă explicită a lui Rumelhart şi Parker). Algoritmul este cunoscut sub denumirea de "backpropagation" provenită de la faptul că pentru determinarea ajustărilor ce vor fi aplicate ponderilor se propagă prin reţea în sens invers (de la nivelul de ieşire către cel de intrare) un semnal de eroare.

[Carpenter, Grosberg, 1983] Dezvoltă "teoria rezonanţei adaptive" ce conduce la algoritmi de învăţare nesupervizată bazată pe procese de competţie.

[Kohonen, 1980] Dezvoltă reţele cu auto-organizare care modelează procese neuronale dar pot fi folosite şi pentru prelucrarea datelor.

3 Specificul calculului evolutiv

Calculul evolutiv oferă mecanisme de căutare în spaţiul soluţiilor bazate pe principiile evoluţiei naturale (de tip darwinist). Pentru găsirea soluţiei se utilizează o *populaţie* de *căutători*. Elementele populaţiei reprezintă soluţii potenţiale ale problemei. Pentru a ghida căutarea către soluţia problemei asupra populaţiei se transformări specifice evoluţiei naturale:

- *Selecţie.* Elementele populaţiei care se apropie de soluţia problemei sunt considerate adecvate şi sunt favorizate în sensul ca au mai multe şanse de a supravieţui în generaţie următoare precum şi de a participa la generarea de "urmaşi".
- *Încrucişare.* La fel ca în înmulţirea din natură pornind de la două sau mai multe elemente ale populaţiei (numite părinţi) se generează noi elemente (numite urmaşi). În funcţie de calitatea acestora (apropierea de soluţia problemei) urmaşii îşi pot înlocui părinţii.
- *Mutaţie.* Pentru a asigura variabilitatea populaţiei se aplică, la fel ca în natură, transformări cu caracter aleator asupra elementelor populaţiei permiţând apariţia unor trăsături (gene) care doar prin încrucişare şi selecţie nu ar fi apărut în cadrul populaţiei.

În funcţie de modul în care este construită populaţia şi de modul în care este implementată evoluţia, sistemele de calcul evolutiv se încadrează în una dintre următoarele categorii:

- *Algoritmi genetici.* Se folosesc în special pentru rezolvarea unor probleme de optimizare discretă (combinatorială). Populaţia este reprezentată de stări din spaţiul problemei codificate binar (un element al populaţiei este un şir de biţi) iar principalii operatori sunt cei de încrucişare şi selecţie, cel de mutaţie având probabilitate mică de aplicare. Algoritmii genetici au fost propuşi de către Holland în perioada anilor 1960, iniţial ca modele ale evoluţiei şi adaptării la mediu a sistemelor naturale. Ulterior s-a observat că algoritmii genetici pot fi utilizaţi şi ca instrumente eficace în rezolvarea problemelor de optimizare.
- *Programare genetică.* Este o direcţie mai recentă a calculului evolutiv, dezvoltată în jurul anilor 1990 de către Koza. Scopul programării genetice este dezvoltarea unor "modele" de calcul (programe simple).Astfel, populaţia este reprezentată de programe care candidează la rezolvarea problemei. Există diferite reprezentări ale elementelor populaţiei, una dintre cele mai clasice fiind aceea în care se utilizeaza o structura arborescentă pentru reprezentarea programelor. În anumite aplicaţii, cum este regresia simbolică, programele sunt de fapt expresii. De exemplu, "programul-expresie" "a+b*c" poate fi descris prin (+ a (* b c)). O astfel de structură este poate fi uşor codificată în Lisp, astfel că primele implementări din programarea genetică foloseau acest limbaj. Într-o astfel de reprezentare încrucişarea este realizată selectând aleator subarbori din arborele asociat programelor părinte şi interschimbându-le. Ca şi în cazul algoritmilor genetici mutaţia are pondere mică.
- *Strategii evolutive.* Au fost concepute iniţial pentru a rezolva probleme de optimizare în tehnică fiind destinate rezolvării problemelor de optimizare continuă. Populaţia este constituită din elemente din domeniul de definiţie al funcţiei obiectiv. Operatorul principal este cel

de mutaţie dar şi recombinarea este folosită. Pentru strategiile evolutive au fost dezvoltate scheme de adaptare a parametrilor de control (auto-adaptare). La dezvoltarea strategiilor evolutive contribuţii importante au adus Rechenberg şi Schwefel.

- *Programare evolutivă.* Iniţial programarea evolutivă a avut ca obiectiv dezvoltarea unor structuri de calcul (automate) printr-un proces de evoluţia în care operatorul principal este cel de mutaţie. Bazele domeniului au fost puse de către Fogel. Ulterior programarea evolutivă a fost orientată către rezolvarea problemelor de optimizare având aceeaşi sferă de aplicabilitate ca şi strategiile evolutive.

Toate aceste metode se bazează pe faptul că *simulează evoluţia* unei mulţimi (*populaţie*) de structuri informaţionale (*configuraţii* sau *indivizi*) sub acţiunea unor procese similare celor din evoluţia naturală şi anume: *selecţie*, *mutaţie* şi *încrucişare*.

Acţiunea acestor procese este controlată prin intermediul unei funcţii de *performanţă* ("fitness") care măsoară gradul de adecvare a fiecărui individ la *mediul* din care face parte. De exemplu, în cazul rezolvării unei probleme de optimizare (maximizare) funcţia "fitness" este chiar funcţia obiectiv a problemei.

3.1 Structura unui algoritm evolutiv

Majoritatea algoritmilor evolutivi au un caracter iterativ. Structura generală a unui astfel de algoritm (algoritm genetic, strategie evolutivă etc.) este:

```
// initializarea indicatorului de iteratie
     t := 0;
// initializarea (aleatoare) a populatiei (multime de configuratii
// a caror structura depinde de problema)
     init P(t);
// calculul functiei "fitness" pentru fiecare individ al populatiei
     evaluate P(t);
// proces iterativ de evoluţie
  repeat
     // incrementarea indicatorului de iteratie\\
    t = t + 1
    // selectarea unei sub-populatii in scopul reproduceri(parinti)
     P1(t) = select P(t)
    // incrucisarea  "genelor" parintilor avand ca rezultat obtinerea unei noi populatii
     P2(t) = recombine P1(t)
    // perturbarea aleatoare a noii populatii (mutatie)
    P3(t) = mutate P2(t)
    // evaluarea noii populatii(calculul functiei "fitness" pentru noua populatie)
    evaluate P3(t)
    // selectia  supravietuitorilor din cele doua populatii
    // (pe baza valorii functiei fitness)
    P(t+1) = survive (P(t),P3(t))
  until <conditie de stop>
```

Conţia de oprire se poate referi la numărul de iteraţii (generaţii) sau la proprietăţile populaţiei curente (de exemplu, întreaga populaţie converge către o singură valoare).

3.2 Domenii de aplicabilitate

La fel ca şi reţelele neuronale, algoritmii evolutivi se utilizează atunci când nu există altă strategie de rezolvare a problemei şi este acceptat un răspuns aproximativ. Se utilizează în special atunci când problema poate fi formulată ca una de optimizare. Câteva dintre domeniile de aplicabilitate sunt:

- *Planificare.* Majoritatea problemelor de planificare (alegerea traseelor optime ale unor vehicule, rutarea mesajelor într-o reţea de telecomunicaţii, planificarea unor activităti etc.) pot fi formulate ca probleme de optimizare cu sau fără restricţii. Multe dintre acestea sunt probleme din clasa "NP-hard" necunoscându-se algoritmi de rezolvare care să aibă complexitate polinomială. Pentru astfel de probleme algoritmii evolutivi oferă posibilitatea obţinerii în timp rezonabil a unor soluţii sub-optimale de calitate acceptabilă.

- *Proiectare.* Algoritmii evolutivi au fost aplicaţi cu succes în proiectarea circuitelor digitale, a filtrelor dar şi a unor structuri de calcul cum sunt reţelele neuronale. Ca metode de estimare a parametrilor unor sisteme care optimizează anumite criterii se aplică în diverse domenii din inginerie cum ar fi: proiectarea avioanelor, proiectarea reactoarelor chimice, proiectarea structurilor în construcţii etc.

- *Simulare şi identificare.* Simularea presupune să se determine modul de comportare a unui sistem pornind de la un model al acestuia. Identificarea este sarcina inversă a identificării sutructurii sistemului pornind de la modul de comportare. Algoritmii evolutivi sunt utilizaţi atât în simularea unor sisteme din inginerie dar şi din economie (de exemplu pentru modelarea proceselor de competitie in marketing). Identificarea unui model este utilă în special în efectuarea de predicţii în diverse domenii (economie, finanţe, medicină, ştiinţele mediului etc.)

- *Control.* Asemenea reţelelor neuronale algoritmii evolutivi pot fi utilizaţi pentru a implementa controllere on-line asociate sistemelor dinamice (de exemplu pentru a controla roboţii mobili).

- *Clasificare.* Se poate considera că din domeniul calculului evolutiv fac parte şi sistemele de clasificare. Un sistem de clasificare se bazează pe o populaţie de reguli de asociere (reguli de producţie) care evoluează pentru a se adapta problemei de rezolvat (calitatea unei reguli se stabileşte pe baza unor exemple). Evoluţia regulilor are acelaşi scop ca şi învăţărea în reţelele neuronale. Algoritmii evolutivi au fost utilizaţi cu succes în clasificarea imaginilor, în biologie (pentru determinarea structurii proteinelor) sau în medicină (pentru clasificarea electrocardiogramelor).

4 Specificul calculului fuzzy

Calculul fuzzy se caracterizează prin faptul că permite manipularea conceptelor vagi care nu pot fi modelate prin concepte matematice exacte (numere, mulţimi sau funcţii clasice). Probleme în care intervin concepte vagi apar în teoria controlului când sistemele au caracter neliniar iar stările lor nu pot fi descrise în mod exact ci doar prin enunţuri care au un grad de ambiguitate. Să considerăm spre exemplu un sistem de control a temperaturii şi umidităţii aerului într-o încăpere. Regulile pe care trebuie să le respecte un astfel de sistem ar putea fi:

> Dacă temperatura şi umiditatea sunt *scăzute* atunci măreşte viteza ventilatorului (pentru aer cald şi umed).

Dacă temperatura şi umiditatea sunt *ridicate* atunci micşorează viteza ventilatorului (pentru aer cald şi umed).

Regulile de mai sus sunt exemple de reguli fuzzy de inferenţă. Prelucrările efectuate de un sistem expert care prelucrează cunoştinţe vagi se bazează pe teoria mulţimilor fuzzy şi logica fuzzy. Variabilelor lingvistice *scăzut* sau *ridicat* li se pot asocia cuantificări prin aşa numitele funcţii de apartenenţă.

4.1 Structura unui sistem fuzzy

Un sistem fuzzy este constituit din:

- un subsistem de extragere a regulilor fuzzy pornind de la datele problemei (etapa de fuzzificare);
- un subsistem de efectuare a inferenţelor fuzzy (etapa de raţionament)
- un subsistem de transformare a cunoştinţelor fuzzy în date efective (etapa de defuzzificare).

4.2 Domenii de aplicabilitate

Principalele aplicaţii ale calculului fuzzy sunt: recunoaşterea formelor; aproximarea funcţiilor; controlul sistemelor; compresia imaginilor.

Curs 2: Structura şi proiectarea reţelelor neuronale artificiale

Din punct de vedere structural o reţea neuronală este un ansamblu de unităţi funcţionale simple interconectate. Orice reţea neuronală este caracterizată prin:

- Specificul unităţilor funcţionale componente.
- Arhitectura - modul de aranjare şi interconectare a unităţilor funcţionale.
- Funcţionare - procesul prin care reţeaua transformă semnalele de intrare (de exemplu datele de intrare ale problemei de rezolvat) în semnale de ieşire (răspunsul reţelei).
- Învăţare - procesul prin care reţeaua se adaptează la specificul problemei.

1 Unităţi funcţionale

Unitatea funcţională este componenta elementară de prelucrare a informaţiei în cadrul unei reţele neuronale. Funcţionarea acestor unităţi este inspirată de modul de funcţionare a neuronilor biologici fără a ţine însă cont de toate particularităţile acestora. În reţelele artificiale sunt folosite şi modele de unităţi funcţionale care nu au relevanţă biologică.

1.1 Modelul neuronului biologic

Celula nervoasă (neuronul) este constituită din:

- *corp neuronal* – constituit, în principal, din membrană şi "substanţă" intracelulară;
- *axon* – prelungire a corpului neuronal cu rol de canal de ieşire;
- *dendrite* – prelungiri ale corpului neuronal cu rol de canale de intrare.

La nivelul celulei nervoase au loc procese de natură fizico-chimică destul de complexe. Se consideră că neuronul primeşte informaţie (prin intermediul dendritelor) sub formă de *impulsuri nervoase* de natură electrică. Aceste impulsuri provoacă depolarizări ale membranei neuronale adică modificări ale diferenţei de potenţial dintre interiorul şi exteriorul celulei. Potenţialele locale generate pe suprafaţa membranei se "însumează" spaţial şi temporal, iar dacă potenţialul rezultant depăşeşte o anumită valoare *prag*, atunci se generează un potenţial de acţiune (PA) care se transmite de-a lungul axonului.

Cercetările din biologie au condus la concluzia că potenţialele de acţiune au amplitudine constantă şi că aceasta nu depinde de intensitatea stimulilor sosiţi pe neuron. Intensitatea şi durata

stimulilor de intrare influenţează doar frecvenţa potenţialelor de acţiune , întrucât un stimul suficient de intens şi de lung provoacă nu un PA singular (care nu poartă suficientă informaţie) ci o secvenţă de potenţiale de acţiune (*tren de impulsuri*).

La nivelul neuronului se poate considera că informaţia codificată sub forma intensităţii unor stimuli (deci analogic) este transformată într-o informaţie binară (prezenţa sau absenţa potenţialelor de acţiune) sau eventual discretă (frecvenţa potenţialelor de acţiune).

Din punct de vedere biologic lucrurile sunt mult mai complicate, punându-se în evidenţă fenomene ca: existenţa unei perioade refractare după activarea unui neuron, creşterea pragului de activare ("oboseala" neuronului), prezenţa unor descărcări (generare potenţial de acţiune) spontane (în absenţa stimulului).

Informaţia pe care o primeşte un neuron din partea altora este codificată *analogic* datorită faptului că semnalele discrete emise de acei neuroni au fost transformate printr-un proces chimic complex care se desfăşoară la nivelul *sinapsei* (spaţiul dintre terminaţia axonului unui neuron şi dendritele altuia).

Sinapsa poate fi definită ca o joncţiune funcţională în spaţiul dintre două celule excitabile. În momentul în care trenul de impulsuri propagat de-a lungul axonului a ajuns la terminaţia acestuia el provoacă "deschiderea" unor vezicule şi eliberarea unor substanţe chimice purtătoare de informaţie (neurotransmiţători) în spaţiul sinaptic. Cantitatea de neurotransmiţători depinde de numărul de impulsuri din cadrul "trenului" dar şi de alţi factori (unii dintre ei neidentificaţi).

Pe membrana postsinaptică sunt prezente formaţiuni care "recepţionează" neurotransmiţătorii provocând modificări ale permeabilităţii ionice a membranei, deci depolarizări.

După efectul pe care-l provoacă, sinapsele sunt: *excitatoare* (provoacă depolarizare pozitivă) sau *inhibitoare* (provoacă depolarizare negativă). S-a observat că stimulări succesive ale unei perechi de neuroni conduc la creşterea permeabilităţii sinapsei corespunzătoare (de exemplu prin sporirea numărului de receptori). Pe această observaţie biologică se bazează ideea că învăţarea (adaptarea) poate fi realizată prin modificări ale permeabilităţii sinaptice.

1.2 Modelul neuronului formal

O unitate funcţională primeşte semnale din partea altor unităţi şi produce un semnal de ieşire. Atât semnalele de intrare cât şi cel de ieşire sunt codificate numeric iar unitatea efectuează următoarele prelucrări:

1. *integrează* (de exemplu, însumează) semnalele de intrare ţinând cont de *ponderea asociată* fiecăruia;

2. *transformă* semnalul "integrat" prin aplicarea unei funcţii, numită *funcţie de transfer* sau *funcţie de activare* obţinând astfel un număr care reprezintă semnalul de ieşire.

Ponderile asociate conexiunilor sunt parametri ai funcţiei de integrare, care pentru un vector de semnale de intrare, $Y \in R^N$, determină o valoare, x_i numită starea neuronului.

Funcţia de agregare (integrare) asociată unui neuron i care primeşte semnale de la N neuroni este de forma $G_{W_i} : \mathbf{R}^N \to \mathbf{R}$ unde W_i reprezintă mulţimea ponderilor conexiunilor către neuronul i. Componenta w_{ij} reprezintă ponderea conexiunii dintre unitatea j şi unitatea i.

Exemple simple de funcţii de agregare sunt:

1. liniară: $G_{W_i}(Y) = \sum_{j=1}^{N} w_{ij} y_j$.

2. pătratică: $G_{W_i^{(1)}, W_i^{(2)}}(Y) = \sum_{j=1}^{N} w_{ij}^{(1)} y_j + \sum_{j,k=1}^{N} w_{ijk}^{(2)} y_j y_k$.

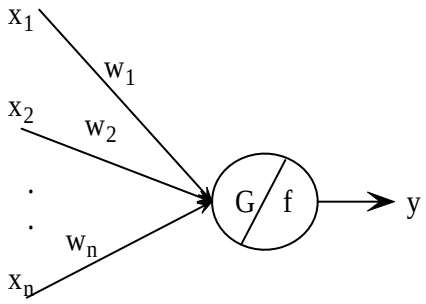


Figura 1: Exemplu de unitate functionala

3. distanţa euclidiană: $G_{W_i}(Y) = \sqrt{(\sum_{j=1}^{N}(w_{ij} - y_j)^2)}$.

Semnalul de ieşire pe care îl produce o unitate funcţională i se determină prin $y_i = f_i(G(Y) - \theta_i)$ unde θ_i reprezintă pragul de activare a unităţii iar f_i funcţia de activare (transfer). Principalele tipuri de funcţii de activare sunt:

1. Funcţii binare deterministe (modelul McCullough-Pitts): $f : R \to B$ cu $B = \{0, 1\}$ iar

$$f(x) = \begin{cases} 1 & \text{dacă } x \geq 0 \\ 0 & \text{dacă } x < 0 \end{cases}$$

adică funcţia Heaviside. O altă variantă este cea cu $B = \{-1, 1\}$, funcţia f fiind de fapt funcţia signum.

2. Funcţii binare stohastice (cu repartiţie de tip Glauber): $f : R \to B$ cu $B = \{0, 1\}$

$$P(f(x) = 1) = \frac{1}{1 + e^{-\beta x}}, \qquad P(f(x) = 0) = \frac{1}{1 + e^{\beta x}}$$

Se poate folosi şi varianta în care $B = \{-1, 1\}$, probabilităţile având aceeaşi formă. Ideea acestor funcţii de transfer este preluată din fizica sistemelor de spini magnetici.

3. Funcţii continue, $f : R \to R$:

   (a) $f(x) = x$ (principalul avantaj al funcţiilor liniare este simplitatea studiului analitic, iar principalul dezavantaj este nemărginirea);

   (b) Dezavantajul de mai sus poate fi eliminat prin folosirea unor funcţii mărginite, liniare pe porţiuni (funcţii de tip "rampă") dar care au dezavantajul de a nu fi netede. Un exemplu simplu de astfel de funcţie este:

$$f(x) = \begin{cases} -1 & \text{dacă } x < -1 \\ x & \text{dacă } -1 \leq x \leq 1 \\ 1 & \text{dacă } x > 1 \end{cases}$$

   (c) Pentru a fi eliminat dezavantajul nenetezimii se utilizează funcţii sigmoidale:

i. (logistica) $f(x) = 1/(1 + \exp(-\beta x),\ \beta > 0;$

ii. (tangenta hiperbolică) $f(x) = \tanh(\beta x) = \dfrac{\exp(2\beta x) - 1}{\exp(2\beta x) + 1},\ \beta > 0;$

Parametrul β permite controlul pantei funcţiilor sigmoidale. Când *beta* tinde către infinit funcţia logistică devine din ce în ce mai apropiată de funcţia Heaviside iar tangenta hiperbolică tinde către funcţia signum.

(d) Dintre funcţiile nemonotone, cele mai des utilizate sunt cele de tip gaussian:

$$f(x) = \exp(-\beta x^2) \qquad \beta > 0.$$

2 Arhitectura

Arhitectura unei retele neuronale se referă la modul în care sunt amplasate unităţile funcţionale (*topologie*) şi la modul în care sunt interconectate (*conectivitate*).

Din punct de vedere al arhitecturii o reţea neuronală poate fi formalizată printr-un graf orientat etichetat în nodurile căruia sunt amplasate unităţile funcţionale şi ale cărui arce specifică conexiunile şi, implicit, fluxul informaţional.

Din punctul de vedere al rolului pe care îl au unităţile funcţionale în cadrul reţelei, ele pot fi descompuse în trei categorii principale:

- *Unităţi de intrare.* Primesc semnale din partea mediului. În cazul în care primesc semnale doar din exterior nu au alt rol decât de a retransmite semnalul primit către alte unităţi din reţea. În această situaţie nu sunt unităţi funcţionale propriu-zise întrucât nu realizează nici o prelucrare asupra semnalului primit.

- *Unităţi ascunse.* Sunt conectate doar cu alte unităţi ale reţelei fără a comunica direct cu mediul extern. Rolul lor este de a colecta semnale, de a le prelucra şi de a distribui semnalul de ieşire către alte unităţi.

- *Unităţi de ieşire.* Colectează semnale de la alte unităţi, le prelucrează şi transmit semnalul pe care îl obţin mediului extern.

În unele reţele cele trei categorii de unităţi formează mulţimi distincte. Cazul cel mai frecvent întâlnit este cel al reţelelor organizate pe nivele: un nivel de unităţi de intrare, unul sau mai multe nivele de unităţi ascunse şi un nivel de ieşire. O situaţie particulară o reprezintă reţelele ce nu conţin unitaţi ascunse ci doar un nivel de unităţi de intrare şi un nivel de unităţi de ieşire.

În alte reţele nu se face distincţie netă între unităţile de intrare şi cele de ieşire: toate unităţile preiau semnale din mediu, le prelucrează şi transmit rezultatul atât unităţilor din reţea cât şi mediului.

Modul de amplasare a unităţilor determină *topologia* reţelei. Din punctul de vedere al acesteia există:

- Reţele în care nu are importanţă (din punctul de vedere al algoritmilor de funcţionare şi/sau de învăţare) poziţia geometrică a unităţilor. Astfel de topologii sunt asociate reţelelor organizate pe nivele (fig. 2) şi reţelelor Hopfield (fig. 3). În reprezentările schematice ale reţelelor organizate pe nivele unităţile aceluiaşi nivel sunt reprezentate grupat deşi poziţia lor nu are semnificaţie pentru procesul de funcţionare şi cel de învăţare.

- Reţele în care este esenţială organizarea geometrică, relaţiile de vecinătate dintre unităţi intervenind în algoritmul funcţionare sau în cel de învăţare. Astfel de topologii sunt cele asociate reţelelor Kohonen (fig. 4, fig. 5) sau reţelelor celulare (fig. 6). Esenţial în acest caz este definirea unei relaţii de vecinăţate între unităţi.

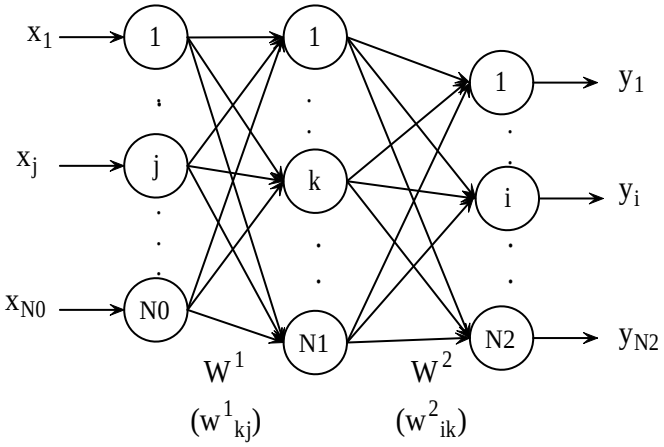


Figura 2: Exemplu de reţea cu un nivel ascuns

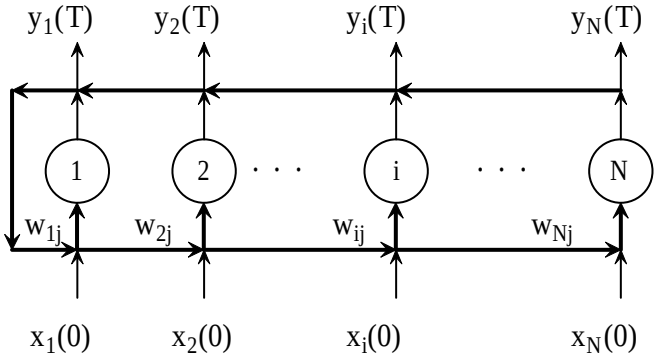


Figura 3: Exemplu de reţea cu conectivitate totală (reţea de tip Hopfield)

Modul de interconectare a unităţilor determină *fluxul de semnale* prin reţea fiind un factor esenţial în stabilirea algoritmului de funcţionare. Din perspectiva prezenţei conexiunilor inverse în graful asociat, reţelele pot fi clasificate în:

- *Reţele de tip "feed-forward".* Nu există conexiuni inverse, fluxul informaţional fiind unidirecţional dinspre setul unităţilor de intrare către cel al unităţilor de ieşire. Conectivitatea între nivele poate fi totală sau locală. Exemple de astfel de reţele sunt ilustrate in fig. 7, 8, 9.
- *Reţele recurente.* Graful asociat conţine conexiuni inverse directe (bucle) sau indirecte (circuite). Exemple de reţele cu conexiuni inverse sunt ilustrate în fig.3, fig.6 şi fig.10 .

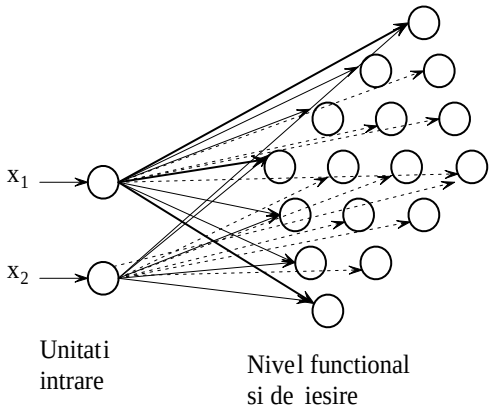


Figura 4: Exemplu de reţea cu organizare geometrică a nivelului de ieşire (reţea Kohonen)

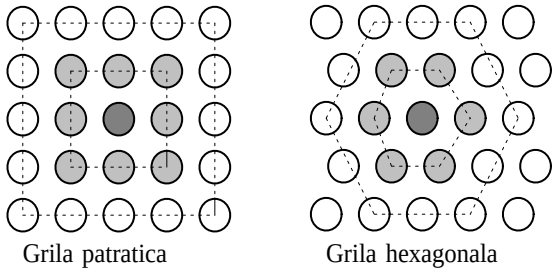


Figura 5: Tipuri de grile bidimensionale utilizate in algoritmii de învăţare de la reţelele Kohonen

2.1 Topologii

Principalele tipuri de topologii sunt:

- *Arbitrară.* Pe mulţimea unităţilor funcţionale nu este definită nici o relaţie de ordine. În acest caz nu are importanţă nici locul şi nici distanţele dintre unităţi. Un model cu o astfel de topologie este modelul Hopfield. De regulă, acestei topologii îi corespunde o conectivitate totală.
- *Pe nivele.* Unităţile sunt împărţite în mai multe submulţimi, numite nivele. În cadrul unui nivel nu are importanţă modul de aranjare a unităţilor. În această categorie intră reţelele feedforward cu unul sau mai multe nivele.
- *Cu structură geometrică.* Unităţile sunt amplasate în nodurile unei grile unidimensionale, bidimensionale sau chiar tridimensionale. În acest caz se poate defini o funcţie distanţă între unităţi. În această categorie intră reţelele de tip Kohonen şi cele celulare.

În practică se utilizează şi arhitecturi mixte în care fiecare nivel poate avea o anumită structură geometrică.

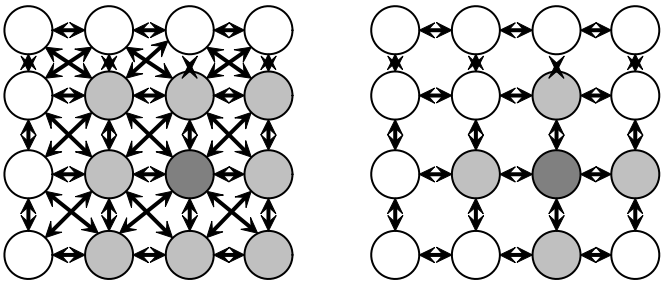


Figura 6: Exemple de reţele celulare

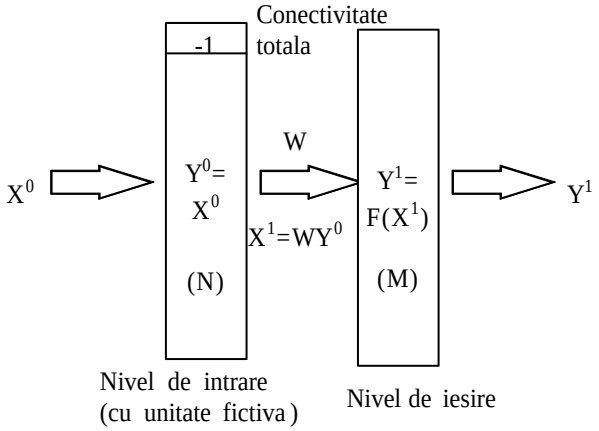


Figura 7: Reprezentare schematică a unei reţele cu un nivel de intrare (N+1 unităţi) şi un nivel de ieşire (M unităţi) şi conectivitate totală între nivelul de intrare şi cel de ieşire.

2.2 Conexiuni

Principalele tipuri de conexiuni sunt:

- Conexiuni între elementele a două nivele distincte. Sunt folosite în special la reţelele organizate pe nivele.
- Conexiuni între elementele aceluiaşi nivel (conexiuni laterale). Se folosesc la reţelele cu structură geometrică.
- Conexiuni inverse prin care o unitate este conectată cu ea însăşi. Pot fi folosite la toate tipurile de topologii.

2.3 Flux informaţional

Fluxul informaţional reprezintă modul în care "curge" informaţia prin reţea (de la unităţile care preiau datele de intrare catre unităţile care produc semnalul de ieşire) În funcţie de modul de interconectare a unităţilor, fluxul informaţional poate fi de una dintre categoriile:

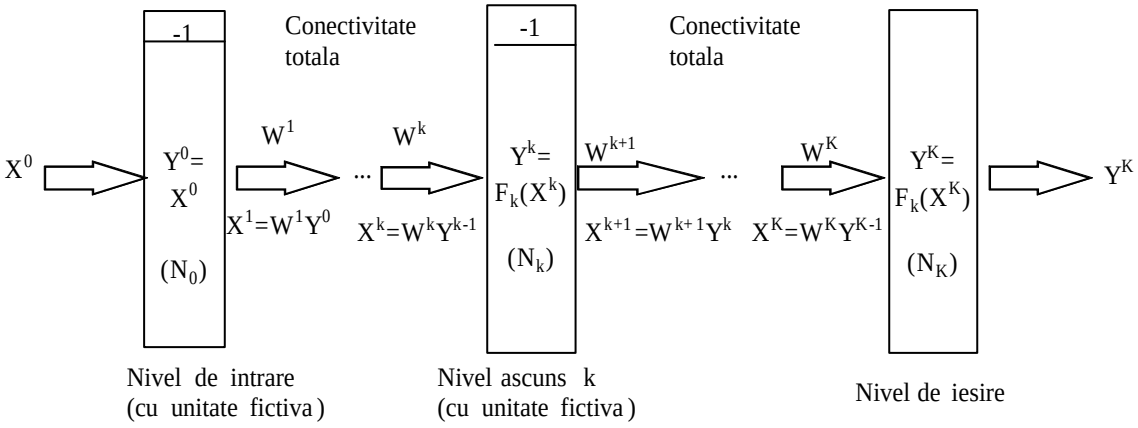


Figura 8: Reprezentare schematică a unei reţele cu K nivele de unităţi funcţionale şi conectivitate totală între nivelele consecutive.

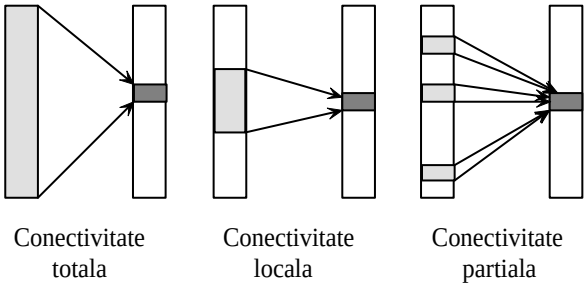


Figura 9: Variante de conectare a unităţilor de pe două nivele

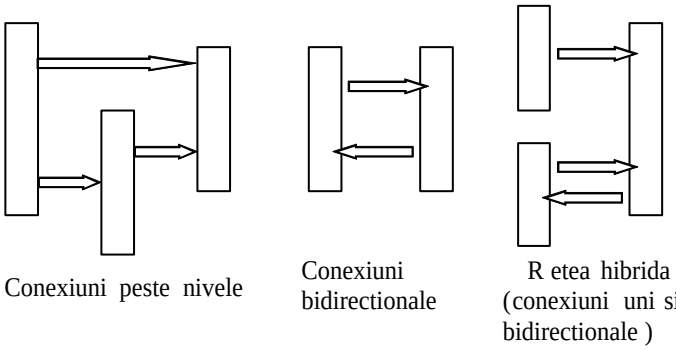


Figura 10: Variante de conectare între două sau mai multe nivele

- *Feedforward:* flux unidirecţional (informaţia circulă doar dinspre nivelul unităţilor de intrare înspre cel al unităţilor de ieşire). Este întâlnit la reţelele organizate pe nivele în care nu există conexiuni inverse.
- *Feedback:* flux multidirecţional (un semnal produs de o unitate poate să ajungă din nou la unitatea respectivă fie direct, fie prin intermediul altor unităţi). Este întâlnit la reţelele cu organizare oarecare şi la cele în care sunt prezente conexiunile laterale sau inverse.

3 Funcţionare

Funcţionarea se referă la modul în care reţeaua transformă un semnal de intrare, X, într-un semnal de ieşire, Y. Ea depinde atât de modul în care funcţionează unităţile cât şi de modul în care sunt interconectate.

Unul dintre parametrii cei mai importanţi ai funcţionării este ansamblul ponderilor asociate tuturor conexiunilor (W). La prima vedere, o reţea neuronală poate fi văzută ca o cutie neagră care primeşte date de intrare şi produce un rezultat (figura 11).

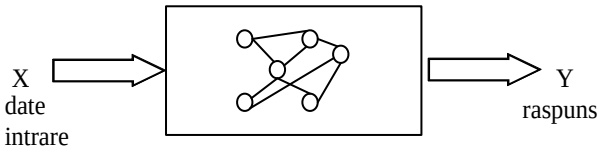


Figura 11: Reţeaua neuronală ca sistem de transformare a semnalelor de intrare în semnale de ieşire.

În funcţie de specificul fluxului informaţional există două moduri principale de funcţionare: *Funcţionare neiterativă.* În cazul reţelelor în care fluxul informaţional este de tip feedforward, semnalul de ieşire, Y, se poate obţine prin aplicarea unei funcţii, F_W (care depinde de parametrii reţelei) , asupra semnalului de intrare, X (figura 12).

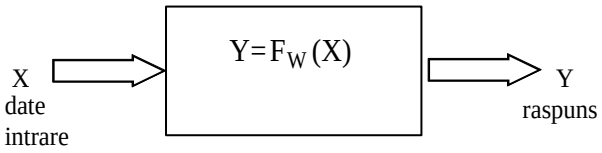


Figura 12: Funcţionarea unei reţele cu flux de tip feedforward

Un exemplu simplu de funcţionare neiterativă (directă) este cel al unei reţele constituită dintr-un nivel de N unităţi de intrare şi un nivel de M unităţi de ieşire. Dacă $X = (x_1, \ldots, x_N)$ reprezintă vectorul semnalelor de intrare în reţea, vectorul semnalelor de ieşire va avea componentele:

$$y_i = f_i(\sum_{j=1}^{N} w_{ij}x_j - \theta_i), \quad i = \overline{1, M}.$$

Funcţionare iterativă. În cazul reţelelor în care sunt prezente conexiuni inverse nu este suficientă o singură "trecere" a semnalului de intrare, X, prin reţea pentru a obţine semnalul de ieşire. Dimpotrivă, în acest caz funcţionarea se desfăşoară în timp, putând fi descrisă printr-un proces iterativ (fig. 13) de forma:

$$X(0) = X, \qquad X(t+1) = F_W(X(t)), \quad t \geq 0.$$

Semnalul de ieşire se consideră ca fiind limita lui $X(t)$ ($X^* = \lim_{t\to\infty} X(t)$). În implementări, limita se aproximează prin $X(T)$, T fiind momentul în care este oprit procesul iterativ. Calitatea acestei aproximări depinde atât de proprietăţile lui F_W cât şi de $X(0)$.

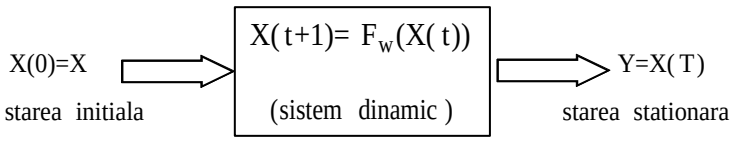


Figura 13: Funcţionarea unei reţele recurente

4 Învăţare

Învăţarea poate fi văzută ca fiind un proces prin care un sistem îşi îmbunătăţeşte performanţele prin achiziţie de cunoaştere.

Pentru reţelele neuronale învăţarea se referă la orice modificare a mulţimii parametrilor (ponderile asociate conexiunilor şi pragurile asociate unităţilor) care asigură o mai bună adecvare a comportării reţelei la problema pentru care a fost proiectată.

Capacitatea de a învăţa este una dintre cele mai importante calităţi ale unei reţele neuronale, prin care aceasta este superioară altor metode. Posibilitatea învăţării din exemple permite evitarea formalizării în detaliu a problemei de rezolvat, un avantaj important mai ales în cazul problemelor pentru care o astfel de formalizare nu există. Prin procesul de învăţare în parametrii reţelei este înglobată implicit o formalizare a problemei, aceasta putând fi utilizată în faza de funcţionare.

Un alt aspect important al reţelelor neuronale este *capacitatea de generalizare* adică de a produce răspunsuri şi pentru date pentru care nu a fost antrenată.

Procesul de învăţare se bazează pe două elemente:

- o mulţime de informaţii (numită şi *set de antrenare*;
- un algoritm de adaptare la informaţiile primite care se bazează pe o regulă de ajustare a parametrilor.

În funcţie de natura informaţiilor primite, învăţarea poate fi de una dintre categoriile:

1. *Nesupervizată (auto-organizare).* Sistemul primeşte doar semnale de intrare din partea mediului şi pe baza acestora descoperă trăsăturile statistice ale populaţiei stimulilor construindu-şi o reprezentare codificată în ponderi a mediului. Această reprezentare poate fi ulterior utilizată pentru alţi stimuli care provin din partea aceluiaşi mediu. Din punct de vedere algoritmic o asemenea metodă de adaptare constă într-un algoritm de construire a ponderilor care poate

fi iterativ sau nu. Regula de ajustare a ponderilor are forma generală: $\Delta w_{ij} = A(y_j, x_i)$, ceea ce sugerează că în procesul de adaptare a ponderii conexiunii dintre unitatea j şi unitatea i intervin doar semnalele de ieşire respectiv de intrare ale celor două unităţi. Aceste metode sunt utilizate pentru aplicaţii de memorare asociativă, grupare a datelor (clustering), analiza componentelor principale.

2. *Supervizată.* Se dispune de un *set de antrenare* (mulţime de exemple) care conţine perechi de forma (I, D) cu I reprezentând semnalul de intrare, iar D răspunsul corect (în cazul învăţării supervizate propriu-zise) sau un indicator de corectitudine (în cazul celei de tip recompensă/ penalizare). Pe baza setului de antrenare ponderile se construiesc iterativ urmărind maximizarea unui indice de performanţă sau minimizarea unei funcţii de eroare. În acest caz regula de ajustare a ponderilor are forma generală $\Delta w_{ij} = A(y_j, x_i, \delta_{ij})$ unde δ_{ij} este un semnal de antrenare determinat pe baza setului de antrenare şi a metodei de optimizare folosite.

   Relativ la învăţarea supervizată apar şi următoarele probleme:

   (a) *Validarea* reţelei "antrenate" : se face de regulă prin reţinerea din setul de antrenare a unui subset de validare (care nu este utilizat în determinarea ponderilor).

   (b) Asigurarea unei bune capacităţi de *generalizare*: se menţine un nivel *acceptabil* de eroare pe setul de antrenare în scopul evitării *supraînvăţării* (învăţarea detaliilor nesemnificative din cadrul exemplelor).

5 Rezolvarea unei probleme folosind reţele neuronale

Rezolvarea clasică (cu algoritmi bine precizaţi pentru fiecare clasă de probleme) a problemelor necesită cunoaşterea a suficiente date despre problemă pentru a o putea descompune în unităţi logice elementare şi pentru a elabora un algoritm care va rămâne " îngheţat" în structura lui, modificându-se doar datele pe care le prelucrează.

Dacă datele despre problemă nu sunt suficiente atunci problema nu poate fi formalizată şi metoda de mai sus nu poate fi aplicată. În aceste situaţii pot fi utilizate reţele neuronale, etapele de rezolvare fiind:

1. Stabilirea unei arhitecturi iniţiale care să fie compatibilă cu problema (de exemplu structura nivelului de intrare în reţea trebuie să fie compatibilă cu numărul de date iniţiale ale problemei) şi alegerea tipului de unităţi funcţionale. Stabilirea gradului de maleabilitate al reţelei prin specificarea parametrilor ajustabili (o reţea va fi cu atât mai generală cu cât va avea mai mulţi parametri ajustabili). Pentru fiecare instanţiere a parametrilor se obţine o anumită funcţie asociată reţelei (două reţele având aceeaşi arhitectură dar valori diferite ale parametrilor pot rezolva două probleme diferite). În anumite cazuri chiar şi arhitectura (de exemplu numărul de unităţi) este maleabilă ea fiind stabilită prin procesul de învăţare.

2. Alegerea unui algoritm de învăţare potrivit cu arhitectura reţelei şi cu cantitatea de informaţie de care se dispune despre problemă. În alegerea algoritmului de învăţare trebuie să se ţină cont de:

   (a) funcţia pe care o poate realiza reţeaua (deci de arhitectură);

   (b) specificul "mediului informaţional" al problemei (de volumul şi natura datelor despre problemă).

3. Antrenarea reţelei pentru a rezolva o anumită problemă. Antrenarea se realizează prin "amplasarea" reţelei în "mediul informaţional" specific problemei şi activarea algoritmului de învăţare.

4. Testarea (validarea) reţelei presupune verificarea corectitudinii răspunsurilor pe care le dă reţeaua când primeşte date de intrare care nu aparţin setului de antrenare dar pentru care se cunoaşte răspunsul corect.

5. Utilizarea propriu-zisă a reţelei.

Primele două etape se referă la proiectarea reţelei iar celelalte la adaptarea (antrenarea) şi utilizarea ei. Adeseori o etapa premergătoare o reprezintă preprocesarea datelor. De exemplu dacă datele de intrare se constituie în vectori de valori numerice o transformare utilă o reprezintă normalizarea acestora.

Curs 3-4: Reţele feedforward cu un singur nivel

1 Arhitectura

Reţelele feed-forward cu un singur nivel de unităţi funcţionale sunt constituite din:

1. Un nivel de N unităţi de intrare caracterizate prin faptul că retransmit semnalul pe care-l primesc. Luând în considerare şi unitatea fictivă utilizată pentru modelarea pragului, nivelul de intrare transmite vectorul $X = (x_0, x_1, \ldots, x_N)^T$ cu $x_0 = -1$.
2. Un nivel de M unităţi de ieşire caracterizate prin funcţii de transfer $f_i : I\!R \to I\!R$ specifice.
3. Un ansamblu de conexiuni între nivelul de intrare şi cel de ieşire, care în cazul cel mai simplu realizează o conectivitate totală. Conexiunilor le sunt asociate ponderile: $w_{ij} \in R$ cu $i \in \{1, \ldots, M\}$ şi $j \in \{0, \ldots, N\}$, w_{i0} reprezentând pragul unităţii de ieşire i.

Structura unei astfel de reţele este ilustrată schematic în figura 1.

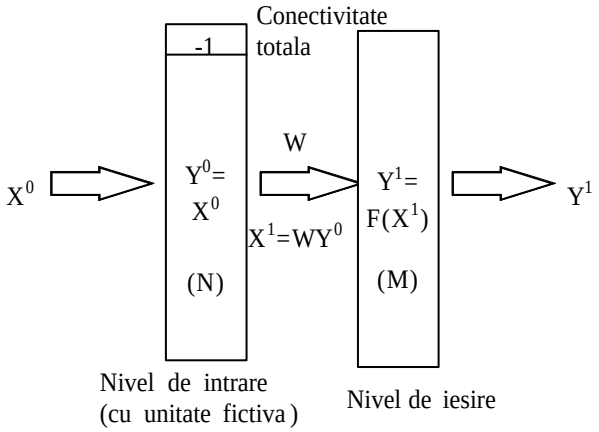


Figura 1: Reprezentare schematică a unei reţele cu un nivel de intrare (N+1 unităţi) şi un nivel de ieşire (M unităţi) şi conectivitate totala între nivelul de intrare şi cel de ieşire.

2 Funcţionare

Funcţionarea unei astfel de reţele depinde de matricea ponderilor conexiunilor $W = (w_{ij})_{i=\overline{1,M},j=\overline{0,N}}$ şi de funcţiile de transfer $(f_i)_{i=\overline{1,M}}$ şi poate fi descrisă prin:

$$y_i = f_i(\sum_{j=0}^{N} w_{ij}x_j), \qquad i \in \{1, \ldots, M\}.$$

Principala problemă rămâne determinarea valorilor w_{ij} care asigură rezolvarea de către reţea a unei anumite probleme, de exemplu efectuarea unor asocieri dintre vectori de intrare şi vectori de ieşire: $X^1 \to d^1, \ldots, X^L \to d^L$ cu $X^l \in I\!R^N$ şi $d_l \in I\!R^M$.

3 Aplicabilitate

Reţelele feedforward (în general, nu numai cele cu un singur nivel) permit realizarea de asocieri între date de intrare $X \in I\!R^N$ şi date de ieşire $Y \in I\!R^M$. Aceste asocieri pot fi formalizate prin funcţii $\phi : D \subset I\!R^N \to I\!R^M$, deci reţelele pot reprezenta funcţii vectoriale. Problematica aproximării funcţiilor poate fi descrisă, în mod simplificat, astfel:

> Presupunem că se cunosc valorile funcţiei ϕ doar pentru o mulţime de valori ale argumentelor (de exemplu se cunosc: $\{(X^1, \phi(X^1)), \ldots, (X^L, \phi(X^L))\}$). Se doreşte estimarea valorilor lui ϕ pentru orice argument X din domeniul său de definiţie.

Despre o funcţie $\psi : R^N \to R^M$ se poate spune că aproximează pe ϕ cu precizia ϵ dacă are loc, de exemplu, $\|\phi(X) - \psi(X)\| < \epsilon$ pentru orice X, $\|\cdot\|$ fiind o normă definită pe R^M.

Diferite probleme concrete pot fi formalizate ca probleme de aproximare de funcţii. De exemplu:

1. Determinarea unei relaţii funcţionale între două mărimi pornind de la date obţinute prin măsurători (*fitarea* datelor experimentale).

2. Efectuarea unei *predicţii* în cadrul unei serii temporale (valori ale unei mărimi ce evoluează în timp) pornind de la un set de valori anterioare. În acest caz se încearcă aproximarea unei funcţii $\phi : D \subset I\!R^k \to I\!R$ care să surprindă legătura dintre k valori anterioare din serie, x_{t-1}, x_{t-2}, ..., x_{t-k} şi valoarea curentă, x_t.

3. *Codificarea/ compresia datelor.* Se caută o funcţie $\phi_1 : I\!R^N \to I\!R^M$, $M < N$ (folosită pentru etapa de codificare) şi o funcţie $\phi_2 : I\!R^M \to I\!R^N$ (folosită pentru etapa de decodificare) astfel încât compusa lor, $\phi_2 \circ \phi_1 : I\!R^N \to I\!R^N$, să fie cât mai apropiată de funcţia identică, $I : I\!R^N \to I\!R^N$, $I(X) = X$.

4. Clasificarea unor vectori din $I\!R^N$ în M clase poate fi văzută ca o asociere între elementul $X \in I\!R^N$ şi un vector indicator al clasei. De exemplu, clasa i poate fi reprezentată de vectorul $(0, \ldots, 1, \ldots, 0)$ singura valoare 1 aflându-se pe poziţia i.

În proiectarea unei reţele neuronale, $\mathcal{R}$, destinate rezolvării unei probleme de asociere (de exemplu, aproximarea funcţiei ϕ) trebuie rezolvate următoarele probleme:

- *Problema reprezentării.* Poate reţeaua $\mathcal{R}$ (caracterizată de o anumită arhitectură, un anumit mod de funcţionare şi de un anumit set de parametrii) să aproximeze funcţia ϕ ? Această problemă este importantă întrucât nu orice reţea poate să aproximeze orice funcţie (de exemplu funcţia XOR nu poate fi aproximată printr-o reţea cu un singur nivel, funcţii liniare de integrare şi funcţii de transfer de tip Heaviside).

- *Problema învăţării.* Presupunem că $\mathcal{R}$ este o reţea a cărei arhitecturi îi permite, teoretic, să aproximeze funcţia ϕ. Se pune problema *cum* să se determine valorile parametrilor reţelei pornind de la informaţiile cunoscute despre ϕ (setul de antrenare) astfel încât aceasta să aproximeze efectiv pe ϕ.

- *Problema generalizării.* Fie $\mathcal{R}$ o reţea antrenată să aproximeze pe ϕ. Se pune problema *cât de bine* aproximează $\mathcal{R}$ pe ϕ pentru date de intrare care nu fac parte din setul de entrenare.

4 Reţele destinate rezolvării problemelor de clasificare

Reţelele din această categorie sunt denumite perceptroni. Termenul "perceptron" provine de la una dintre primele reţele neuronale artificiale implementate. Este vorba despre o reţea neuronală destinată recunoaşterii unor caractere şi care a fost implementată în perioada 1950-1960 de către o echipă coordonată de către Rosenblatt. O astfel de reţea este capabilă să rezolve doar probleme liniar separabile permiţând prin algoritmul de învătare specific (algoritmul perceptronului) determinarea coeficienţilor hiperplanelor care separă clasele.

4.1 Specificul problemelor de clasificare. Liniar separabilitate.

Problematica clasificării, denumită şi recunoaşterea formelor ("pattern recognition"), se referă la încadrarea unei descrieri asociate unui obiect (numită formă, şablon sau tipar) într-o categorie de obiecte *similare*, numită *clasă*.

Un exemplu simplu de clasificare este cel al asocierii dintre o descriere prin matrice de pixeli a unui caracter (de exemplu litera C) şi un indicator al clasei căreia îi aparţine (de exemplu valoarea 3 reprezentaând numărul de ordine al clasei asociate literei C).

Procesul de clasificare depinde de modul de descriere a obiectului. Există două modalităţi principale de descriere a obiectelor:

- *Geometrică.* Obiectului i se asociază un vector cu valori numerice asociate unui set de caracteristici specifice. Acestei modalităţi de descriere îi corespund sistemele de clasificare decizional-statistice.

- *Structurală.* Obiectul este descris într-o manieră simbolică pornind de la un set de primitive. Sistemele de clasificare specifice acestei descrieri se bazează pe o analiză sintactică.

În rezolvarea cu reţele neuronale a problemelor de clasificare se porneşte de la descrierea geometrică. Dacă mulţimea formelor este $\Omega \subset I\!R^N$, aceasta poate fi descompusă într-un set de clase, $\Omega = C_1 \cup C_2 \cup \ldots \cup C_M$. Separarea în clase este asigurată de existenţa unor funcţii de decizie, D_1, D_2, ..., D_M, cu $D_i : I\!R^N \rightarrow I\!R$.

În cazul a două clase, C_1 şi C_2 este suficientă o singură funcţie de decizie $D(X)$ care permite clasificarea în modul următor: $X \in C_1$ dacă şi numai dacă $D(X) > 0$, iar $X \in C_2$ dacă şi numai dacă $D(X) < 0$. Vectorii X cu proprietatea că $D(X) = 0$ se află chiar pe suprafaţa de decizie.

Dacă funcţia de decizie este liniară, $D(X) = w_1x_1 + w_2x_2 + \ldots + w_Nx_N - w_0$, atunci cele două clase se numesc *liniar separabile*, iar suprafaţa de decizie este un hiperplan. Dacă nu există nici o funcţie de decizie liniară care să separe cele două clase atunci ele sunt numite *neliniar separabile*. Pentru cazul particular $N = 2$ liniar separabilitatea este ilustrată în fig. 2.

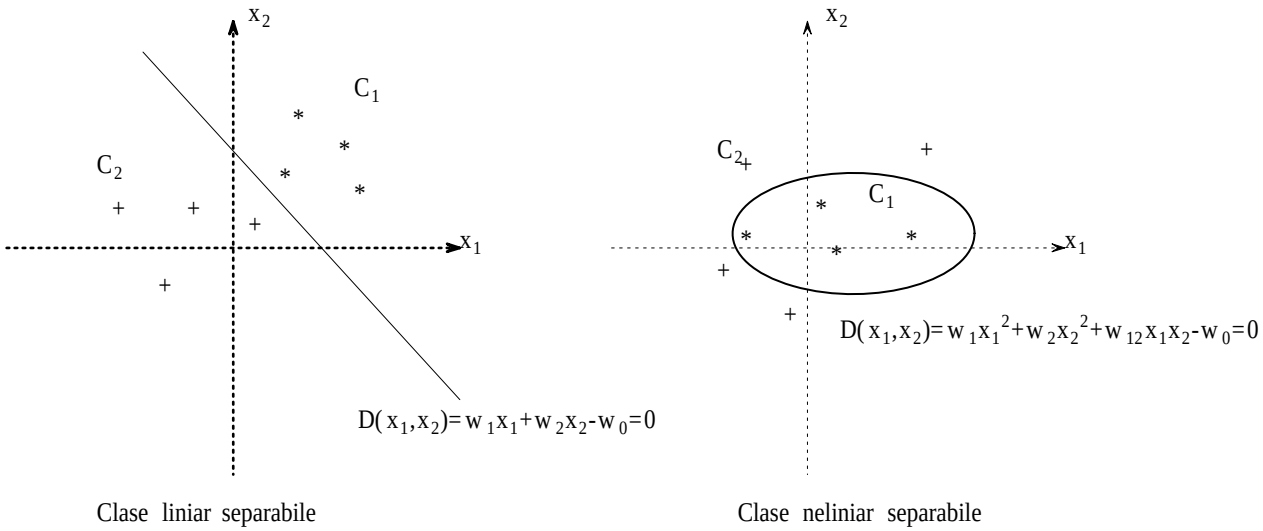


Figura 2: Liniar separabilitate şi neliniar separabilitate în cazul a două clase ($M = 2$, $N = 2$)

Noţiunea de liniar separabilitate poate fi extinsă la cazul $M > 2$ în unul dintre următoarele moduri:

- *Clase puternic liniar separabile.* Dacă există M funcţii liniare, $D_i(X) = w_{i1}x_1 + w_{i2}x_2 + \ldots + w_{iN}x_N - w_{i0}$, cu proprietatea că

  $X \in C_i$ dacă şi numai dacă $D_i(X) > 0$ şi $D_j(X) < 0$ pentru orice $j \in \{1, \ldots, M\}, j \neq i$

  atunci clasele se numesc puternic liniar separabile.

- *Clase simplu liniar separabile.* Dacă există M funcţii liniare, $D_i(X) = w_{i1}x_1 + w_{i2}x_2 + \ldots + w_{iN}x_N - w_{i0}$, cu proprietatea că

  $X \in C_i$ dacă şi numai dacă $D_i(X) > D_j(X)$ pentru orice $j \in \{1, \ldots, M\}, j \neq i$

  atunci clasele se numesc simplu liniar separabile.

Clasele puternic liniar separabile sunt şi simplu liniar separabile. Clasele care nu sunt simplu liniar separabile se numesc neliniar separabile. În cazul a 3 clase liniar separabilitatea este ilustrată în fig. 3.

4.2 Perceptronul simplu

Este o reţea ce conţine o singură unitate funcţională având funcţie de transfer binară (de tip Heaviside) sau bipolară (de tip signum). Este destinată rezolvării problemelor de clasificare în două clase, C_1 şi C_2.

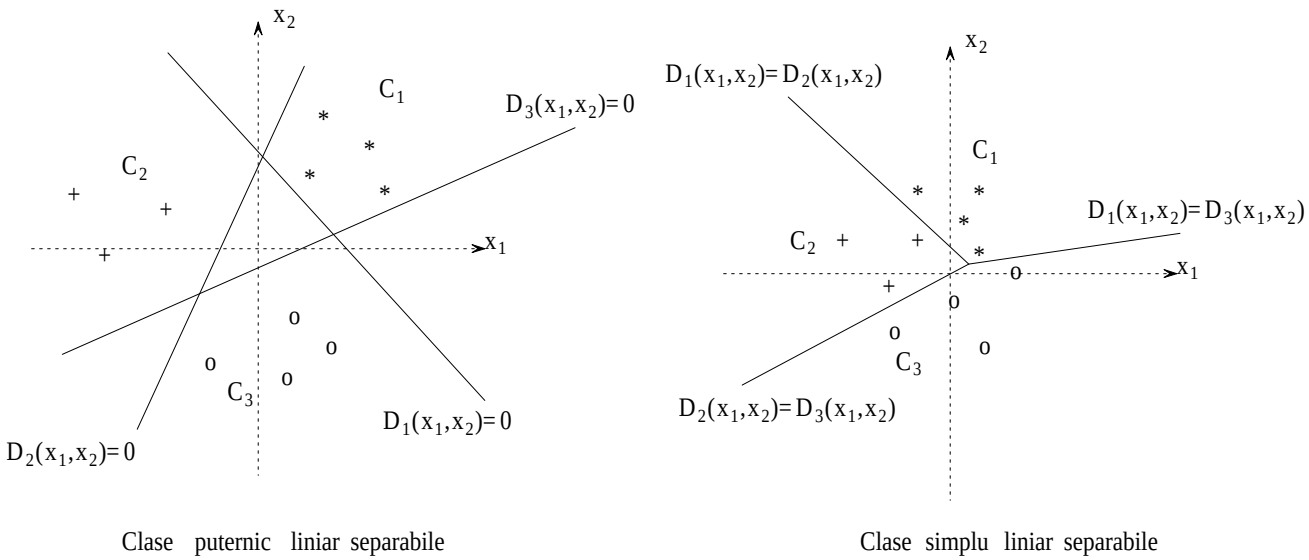


Figura 3: Liniar separabilitate în cazul a 3 clase ($M = 3$, $N = 2$)

Considerând funcţia de transfer de tip signum, pentru un vector de intrare $X = (x_1, \ldots, x_N)^T \in \mathbb{R}^N$ reţeaua produce semnalul de ieşire:

$$y = sgn(\sum_{i=0}^{N} w_i x_i)$$

care poate fi interpretat astfel: dacă $y = -1$ atunci vectorul X aparţine clasei C_1, altfel el aparţine clasei C_2. Corespondenţa dintre valoarea produsă de reţea şi clasa asociată se stabileşte înainte de a iniţia procesul de învăţare (în momentul construirii setului de antrenare).

Un sistem cu o astfel de funcţionare este de fapt un clasificator binar cu funcţie liniară de decizie, coeficienţii acesteia fiind $W = (w_0, w_1, \ldots, w_N)$. Pentru determinarea coeficienţilor funcţiei de decizie se utilizează un set de antrenare: $\{(X^1, d_1), \ldots, (X^L, d_L)\}$ cu $d_l \in \{-1, 1\}$ răspunsul corect corespunzător intrării X^l, pe baza căruia se ajustează iterativ vectorul ponderilor $W = (w_0, w_1, \ldots, w_N)$. Algoritmul de antrenare a perceptronului constă, ca majoritatea algoritmilor de antrenare supervizată, în două etape principale:

- Iniţializarea valorilor ponderilor.
- Ajustarea iterativă a ponderilor până este asigurată comportarea corectă pe setul de antrenare. Aceasta se bazează pe ideea de a modifica ponderile doar dacă răspunsul reţelei nu este corect, ajustându-le astfel încât răspunsul reţelei să se apropie de cel corect.

Structura generală a algoritmului este prezentată în fig. 4.

Observaţii.

1. Regula de ajustare din algoritmul de învăţare al perceptronului simplu se bazează pe modificarea ponderilor doar dacă răspunsul furnizat de reţea, y_l pentru vectorul de intrare X^l este

Iniţializări.

Se aleg valorile iniţiale pentru w_j, $j = \overline{0, N}$

Se iniţializează contorul de iteraţii, $k = 1$

Ajustare iterativă.

REPEAT

FOR $l = \overline{1, L}$ DO

$$y_l = \sum_{j=0}^{N} w_j x_j^l$$

$$w_j = w_j + \frac{1}{2}\eta(d_l - y_l)x_j^l,\ j = \overline{0, N}$$

$k = k + 1$

UNTIL "la ultima parcurgere nu s-a efectuat nici o ajustare" OR $k > k_{max}$

Figura 4: Algoritmul de învăţare pentru perceptronul simplu

diferit de răspunsul corect d_l. Regula de ajustare a ponderilor poate fi descrisă şi prin:

$$\begin{aligned} w_j &= w_j - \eta x_j^l, && \text{dacă } d_l = -1, y_l = 1 \\ w_j &= w_j + \eta x_j^l, && \text{dacă } d_l = 1, y_l = -1 \end{aligned}$$

sau prin $w_j = w_j + \eta d_l x_j^l$.

2. Rata de corecţie, η, poate fi aleasă în diferite moduri:

   (a) η =constant (ex: $\eta = 1$);

   (b) *Algoritmul corecţiei:* η se alege la fiecare iteraţie (şi pentru fiecare exemplu (X^l, d_l)) suficient de mare pentru a garanta că vectorul X^l va fi clasificat corect după ajustarea coeficienţilor. Dacă $W(k)$ este vectorul ponderilor corespunzător iteraţiei k atunci η se alege astfel încât $y_l(k+1)d_l > 0$ adică

$$(W(k) + \eta(k,l)d_l X^l)^T X^l d_l > 0$$

   ceea ce este echivalent cu

$$\eta(k,l) > \frac{|W(k)^T X^l|}{\|X^l\|^2}.$$

   De exemplu se poate alege astfel:

$$\eta(k,l) = \frac{1 + |W(k)^T X^l|}{\|X^l\|^2}.$$

   O altă variantă o reprezintă *corecţia parţială*:

$$\eta(k,l) = \lambda \frac{|W(k)^T X^l|}{\|X^l\|^2}$$

   cu $\lambda \in (0, 2)$.

3. Alegerea valorilor iniţiale pentru W influenţează viteza de convergenţă şi forma clasificatorului. Dacă este posibil este indicat ca W să fie iniţializate pe baza unor informaţii despre problemă. De exemplu se consideră că doi dintre vectorii din setul de antrenare pot reprezenta, la o primă aproximare, prototipuri ale celor două clase. Coeficienţii hiperplanului perpendicular pe dreapta determinată de cele două puncte se consideră a fi valorile iniţiale ale ponderilor.

   Varianta cea mai simplă de iniţializare o reprezintă însă afectarea de valori aleatoare cuprinse în $(0,1)$ sau $(-1,1)$.

4. Perceptronul simplu poate reprezenta funcţii booleene $F : \{-1,1\}^N \to \{-1,1\}$ dar nu oarecare ci dintre cele cu proprietatea că există o transformare liniară $D : I\!R^{N+1} \to I\!R$, $D(X) = \sum_{i=0}^{N} w_i x_i$ cu proprietatea că $F(X) = -1$ dacă şi numai dacă $D(X) \leq 0$ şi $F(X) = 1$ dacă şi numai dacă $D(X) > 0$. Se pune problema dacă pentru orice funcţie booleană cu această proprietate algoritmul perceptronului furnizează o soluţie. Răspunsul este afirmativ şi cuprins în următoarea teoremă.

Teoremă. *Pentru o problemă liniar separabilă algoritmul perceptronului conduce după un număr* **finit** *de iteraţii la coeficienţii unei funcţii de decizie (deci la reprezentarea unei funcţii booleene cu proprietăţile de mai sus).*

Demonstraţie. Problema de clasificare fiind liniar separabilă rezultă că există $W^* \in I\!R^{N+1}$ astfel încât $D(X) = (W^*)^T X$ să fie funcţie de decizie.

Cum $d_l \in \{-1,1\}$ rezultă că relaţiile de ajustare corespunzătoare iteraţiei k pot fi scrise astfel:

$$W(k+1) = W(k) \text{ dacă } d_l W(k)^T X^l > 0$$

$$W(k+1) = W(k) + \eta d_l X^l \text{ dacă } d_l W(k)^T X^l \leq 0$$

Evident $(W^*)^T X^l d_l > 0$, deci putem considera că există $\delta > 0$ cu $\eta d_l (W^*)^T X^l > \delta$.

Deci:

$$W(k+1)^T W^* = W(k)^T W^* + \eta (W^*)^T X^l d_l > W(k)^T W^* + \delta$$

Prin inducţie rezultă imediat că: $W(k+1)^T W^* > W(1)^T W^* + k\delta$.

Pe de altă parte, dacă $W(k+1) \neq W(k)$ are loc:

$$\|W(k+1)\|^2 = \|W(k)\|^2 + 2\eta d_l W(k) X^l + \eta^2 \|X^l\|^2 \leq$$

$$\|W(k)\|^2 + \eta^2 \|X^l\|^2 \leq \|W(k)\|^2 + M$$

Prin inducţie rezultă imediat că: $\|W(k+1)\|^2 \leq \|W(1)\|^2 + kM$. Notând cu $\alpha(k+1)$ unghiul format de $W(k+1)$ şi W^* se observă că:

$$\cos(\alpha(k+1)) = \frac{W(k+1)^T W^*}{\|W(k+1)\| \|W^*\|} > \frac{W(1)^T W^* + k\delta}{\|W^*\| \sqrt{\|W(1)\|^2 + kM}}$$

Dacă presupunem că nu există k^* astfel încât $W(k^*+1) = W(k^*)$ atunci:

$$\lim_{k \to \infty} \cos(\alpha(k+1)) = \infty$$

Absurd, căci $\cos(u) \leq 1$. Deci există $k^* < \infty$ astfel încât $W(k^*+1) = W(k^*)$ adică algoritmul converge în număr finit de paşi. □

4.3 Perceptronul multiplu

Perceptronul multiplu permite clasificarea în mai mult de două clase cu condiţia ca acestea să fie liniar separabile. Din punct de vedere al arhitecturii, perceptronul multiplu este o reţea care conţine un nivel de M unităţi funcţionale total conectate cu cele $N+1$ unităti de intrare.

Ponderile conexiunilor dintre unităţile de intrare şi cele funcţionale (care sunt totodată unităţi de ieşire) pot fi organizate într-o matrice, W, cu M linii şi $N+1$ coloane (pentru a cuprinde şi unitatea fictivă asociată pragului unităţilor funcţionale).

Din punct de vedere al funcţionării, pentru un semnal de intrare $X \in I\!R^N$ se obţine semnalul de ieşire $Y \in I\!R^M$ astfel:

$$Y = WX \qquad \text{adică } y_i = \sum_{j=0}^{N} w_{ij}x_j = W^iX, i = \overline{1,M}$$

W^i fiind linia i din matricea W. Rezultatul obţinut de către reţea se interpretează în modul următor:

Se determină componenta maximă y_{i^*} a vectorului Y:

$$y_{i^*} = \max_{i=\overline{1,M}} y_i$$

iar i^* este considerat indicatorul clasei căreia îi aparţine vectorul de intrare X.

4.3.1 Algoritmul de învăţare

Fie setul de antrenare $\{(X^1, d_1), \ldots, (X^L, d_L)\}$ unde $d_l \in \{1, \ldots, M\}$ este indicatorul clasei căreia îi aparţine X^l. Algoritmul de antrenare al perceptronului multiplu este o extindere a celui al percepronului simplu bazându-se pe ajustarea ponderilor doar în cazul în care reţeaua nu furnizează răspunsul corect. Structura algoritmului este descrisă în fig. 5.

Iniţializări.
  Se aleg valorile iniţiale pentru w_{ij}, $i = \overline{1,M}$, $j = \overline{0,N}$
  Se iniţializează contorul de iteraţii: $k = 1$
Ajustare iterativă.
  REPEAT
    FOR $l = \overline{1,L}$ DO
      $y_i^l = \sum_{j=0}^{N} w_{ij}x_j^l$, $i = \overline{1,M}$
      determină $i^* \in \{1, \ldots, M\}$ cu proprietatea că $y_{i^*}^l \geq y_i^l$, $i = \overline{1,M}$
      IF $i^* \neq d_l$ THEN
        $w_{i^*j} = w_{i^*j} - \eta x_j^l, \quad j = \overline{0,N}$
        $w_{d_lj} = w_{d_lj} + \eta x_j^l, \quad j = \overline{0,N}$
    $k = k + 1$
  UNTIL "la ultima parcurgere nu s-a efectuat nici o ajustare" OR $k > k_{max}$

Figura 5: Algoritmul de învăţare pentru perceptronul multiplu

Observaţiile referitoare la alegerea ratei de învăţare pentru perceptronul simplu rămân valabile şi în cazul perceptronului multiplu. Principalul rezultat referitor la convergenţa acestui algoritm este:

Teoremă. *Dacă cele M clase sunt simplu liniar separabile atunci algoritmul conduce, după un număr finit de paşi, la coeficienţii unor hiperplane separatoare.*

5 Reţele destinate rezolvării problemelor de aproximare

Principalele diferenţe între aceste reţele şi perceptroni sunt legate de modul de interpretare a răspunsului şi de principiul algoritmului de învătare.

Perceptronii sunt consideraţi reţele destinate rezolvării unor probleme de decizie (cum sunt problemele de clasificare) deci apare necesară determinarea unităţii care produce ieşirea maximă, pe când reţelele din această categorie sunt destinate în special rezolvării problemelor de aproximare astfel că răspunsul reţelei este chiar semnalul de ieşire produs.

Cele mai reprezentative reţele uninivel sunt cele cu funcţii de transfer liniare. În această categorie se încadrează reţelele ADALINE (ADAptive LINear Element), cu o singură unitate de ieşire, respectiv MADALINE (Multiple ADAptive LINear Element) cu mai multe unităţi de ieşire. Aceste reţele au fost introduse în perioada anilor 1960 de către Widrow şi Hoff. Spre deosebire de perceptron o astfel de reţea poate fi utilizată pentru a reprezenta funcţii liniare continue, principalele aplicaţii fiind în regresia liniară şi în filtrarea liniară a semnalelor.

Algoritmul propus de Widrow şi Hoff pentru determinarea parametrilor W este tot unul cu caracter supervizat dar bazat pe altă idee şi anume aceea a minimizării unui criteriu de eroare. Cel mai frecvent utilizat criteriu este eroarea pătratică calculată pe setul de antrenare.

Fie $\{(X^1, d^1), \ldots, (X^L, d^L)\}$ setul de antrenare. Eroarea pătratică pe setul de antrenare este:

$$E(W) = \frac{1}{2}\sum_{l=1}^{L} \|y^l - d^l\|^2, \qquad y^l = W^T X^l$$

Se poate lucra şi cu eroarea pătratică medie pe setul de antrenare (obţinută din $E(W)$ prin împărţire la L). Obiectivul algoritmului de învăţare devine astfel minimizarea lui E în raport cu parametrii grupaţi în W.

În cazul reţelelor liniare, E este funcţie pătratică cu minim unic. Pentru determinarea lui W^* care minimizează pe E se pot utiliza instrumente din algebra liniară. În cazul în care $M = 1$, $d_l \in I\!R$ şi cum d_l^2 nu depinde de $W \in I\!R^{N+1}$ rezultă că minimizarea lui $E(W)$ este echivalentă cu minimizarea lui:

$$E(W) = \frac{1}{2}\sum_{l=1}^{L}((W^T X^l)^2 - 2 d_l W^T X^l) =$$

$$= \frac{1}{2} W^T \left(\sum_{l=1}^{L} X^l (X^l)^T\right) W - W^T \left(\sum_{l=1}^{L} d_l X^l\right) = \frac{1}{2} W^T R W - W^T Q$$

unde $R = \sum_{l=1}^{L} X^l (X^l)^T$ şi $Q = \sum_{l=1}^{L} d_l X^l$.

Se observă că matricea R este simetrică şi pozitiv definită, deci $E(W)$ are un minim unic. O primă variantă de determinare a minimului lui $E(W)$ se bazează pe determinarea punctului critic al lui E, adică pe rezolvarea sistemului $RW = Q$, în raport cu W.

O altă modalitate de determinare a lui W^*, care poate fi aplicată şi pentru cazul în care E nu este funcţie pătratică, este de a folosi o metodă iterativă de minimizare. Una dintre cele mai simple este metoda gradientului care conduce la următoarea regulă iterativă de ajustare a ponderilor:

$$W(k+1) = W(k) - \eta \nabla E(W(k))$$

cu $\nabla E(W) = (\partial E(W)/\partial w_1, \ldots, \partial E(W)/\partial w_N)$ reprezentând gradientul lui E iar $\eta > 0$ pasul de descreştere, care în domeniul algoritmilor de învăţare este denumit rată de învăăţare.

Pornind de la această idee a fost dezvoltat algoritmul Widrow-Hoff, în care însă ajustarea se calculează separat pentru fiecare termen pornind de la valoarea erorii corespunzătoare fiecărui exemplu (X^l, d^l) din setul de antrenare:

$$E_l(W) = \frac{1}{2}\|d^l - y^l\|^2 = \frac{1}{2}\sum_{i=1}^{M}(d_i^l - y_i^l)^2 = \frac{1}{2}\sum_{i=1}^{M}(d_i^l - \sum_{j=0}^{N} w_{ij}x_j^l)^2.$$

Ajustarea fiecărei ponderi w_{ij} este determinată de derivata parţială a lui $E_l(W)$:

$$\frac{\partial E_l(W)}{\partial w_{ij}} = -(d_i^l - y_i^l)x_j^l.$$

Algoritmul de învăţare va consta în parcurgerea repetată a setului de antrenare şi în ajustarea ponderilor până la îndeplinirea unei condiţii de oprire. În aceste condiţii structura algoritmului este descrisă în fig. 6.

Observaţii.

1. Valoarea E_* reprezintă eroarea maximă admisă în faza de antrenare şi este denumită toleranţa la antrenare.

2. Datorită convexităţii lui $E(W)$ metodele de minimizare de tip gradient sunt convergente (dar nu în număr finit de paşi ca algoritmul perceptronului). Principala problemă care apare este cea a vitezei de convergenţă, care poate fi controlată, într-o oarecare măsură, prin alegerea ratei de învăţare. De regulă se foloseşte o rată constantă: $\eta \in [0.01, 1]$. Una dintre valorile cel mai frecvent folosite este: $\eta = 0.1$.

3. Iniţializarea ponderilor $(W(1))$ se efectuează, de regulă, cu valori aleatoare din $(0, 1)$ sau chiar $(-1, 1)$.

6 Reţele cu funcţii de transfer continue neliniare

Dacă unităţile de ieşire au funcţii de transfer neliniare atunci reţeaua poate reprezenta şi funcţii continue neliniare (din păcate un singur nivel funcţional nu poate asigura reprezentarea oricărei funcţii). Diferenţa faţă de cazul unităţilor liniare se manifestă atât în funcţionalitate cât şi în algoritmul de învăţare.

Ideea algoritmului de învăţare este tot minimizarea erorii pe setul de antrenare, ajustarea de tip gradient fiind aplicată pentru fiecare exemplu din setul de antrenare asupra lui E_l :

$$E_l(W) = \frac{1}{2}\sum_{i=1}^{M}(d_i^l - f(\sum_{j=0}^{N} w_{ij}x_j^l))^2$$

Iniţializări.

Se aleg valorile iniţiale pentru w_{ij}, $i = \overline{1, M}$, $j = \overline{0, N}$

Se iniţializează contorul de iteraţii, $k = 1$

Ajustare iterativă.

REPEAT

FOR $l = \overline{1, L}$ DO

Ajustare de tip gradient în raport cu E_l

$$y_i^l = \sum_{j=0}^{N} w_{ij} x_j^l, \quad i = \overline{1, M}$$

$$\delta_i^l = d_i^l - y_i^l, \quad i = \overline{1, M}$$

$$w_{ij} = w_{ij} + \eta \delta_i^l x_j^l, \qquad i = \overline{1, M}, j = \overline{0, N}$$

Recalcularea valorii erorii pe setul de antrenare

E=0

FOR $l = \overline{1, L}$ DO

$$y_i^l = \sum_{j=0}^{N} w_{ij} x_j^l, \quad i = \overline{1, M}$$

$E = E + \sum_{i=1}^{M} (d_i^l - y_i^l)^2$

E=E/(2L)

$k = k + 1$

UNTIL $E < E_*$ OR $k > k_{max}$

Figura 6: Algoritmul de învăţare Widrow-Hoff

cu ajutorul unei metode de tip gradient. În acest caz componentele gradientului lui E_l sunt de forma:

$$\frac{\partial E_l(W)}{\partial w_{ij}} = -f'(\sum_{j=0}^{N} w_{ij} x_j^l)(d_i^l - y_i^l) x_j^l$$

iar ajustarea ponderilor ca urmare a prezentării perechii (X^l, d^l) este similară algoritmului Widrow-Hoff cu diferenţa că $\delta_i^l = f'(\sum_{j=0}^{N} w_{ij} x_j^l)(d_i^l - y_i^l)$.

Algoritmul propriu-zis, numit "delta-rule", are aceeaşi structură ca algoritmul Widrow-Hoff diferind doar prin relaţia de calcul pentru δ_i^l .

Observaţii.

1. Datorită neliniarităţii funcţiilor de transfer funţia de eroare nu mai determină o suprafaţă paraboloidală, deci $E(W)$ poate avea mai multe minime. Din acest motiv nu mai este garantată atingerea prin algoritmi de tip gradient a minimului global ci algoritmul se poate "bloca" într-un minim local.

2. Eficienţa algoritmului depinde mai mult de modul de alegere a ponderilor iniţiale şi a ratei de învăţare decât în cazul liniar.

3. Cele mai des utilizate funcţii neliniare de transfer sunt:

   (a) Funcţia logistică:

$$f(u) = \frac{1}{1 + \exp(-\beta u)}$$

cu $f'(u) = \beta f(u)(1 - f(u))$.

(b) Funcţia tanh:

$$f(u) = \frac{\exp(2\beta u) - 1}{\exp(2\beta u) + 1}$$

cu $f'(u) = \beta(1 - (f(u))^2)$.

7 Limite ale reţelelor feed-forward cu un nivel

Una dintre cele mai mari limite ale reţelelor cu un nivel este aceea că nu pot reprezenta unele funcţii simple, cel mai des citat exemplu fiind cel al funcţiilor boolene de paritate: $F : \{0,1\}^N \to \{0,1\}$ cu $F(X) = 0$ dacă X conţine un număr par de componente egale cu 1 şi $F(X) = 1$ în caz contrar.

Cel mai simplu exemplu este cel al funcţiei XOR ($N = 2$). O reţea ar putea reprezenta XOR dacă are două unităţi de intrare şi una de ieşire cu o funcţie de transfer care verifică:

$$f(0) \geq 0 \qquad f(w_1 + w_2) \geq 0 \qquad f(w_1) < 0 \qquad f(w_2) < 0$$

Evident funcţia signum, funcţia liniară sau chiar cea sigmoidală nu pot satisface condiţiile de mai sus.

În schimb prin adăugarea unui nivel intermediar între cel de intrare şi cel de ieşire se poate reprezenta funcţia XOR. Aceasta motivează utilizarea reţelelor feedforward cu mai multe nivele de unităţi funcţionale. De remarcat că aplicabilitatea reţelelor multinivel cu funcţii de transfer liniare pe toate nivelele nu este semnificativ mai largă decât a celor cu un nivel (în fond, o reţea neuronală cu mai multe nivele având funcţii liniare de transfer poate reprezenta doar funcţii liniare). Astfel se explică de ce în majoritatea modelelor funcţiile de transfer ale unităţilor de pe nivelele ascunse sunt neliniare.

Retele feed-forward cu nivele ascunse

Capacitatea de reprezentare restrânsă a reţelelor cu un singur nivel a condus la necesitatea dezvoltării reţelelor cu mai multe nivele. Dintre aceste nivele, unul este de intrare, unul de ieşire iar restul nu comunică cu mediul fiind numite nivele ascunse. Din punct de vedere intuitiv rolul nivelelor ascunse este de a asigura extragerea implicită a unor caracteristici ale datelor de intrare.

Introducerea nivelelor ascunse sporeşte capacitatea de reprezentare a reţelelor feedforward dar ridică dificultăţi în ceea ce priveşte procesul de învăţare, algoritmii de tip "delta" neputând fi aplicaţi în mod direct. Acesta a fost unul dintre principalele motive ale stagnării dezvoltăii reţelelor feedforward cu învăţare supervizată între 1969 (anul în care Papert şi Minsky au pus în evidenţă limitele reţelelor uni-nivel) şi 1985 (anul în care algoritmul BackPropagation, dezvoltat în paralel de mai mulţi cercetători) a devenit cunoscut.

Reţelele din această categorie sunt aproximatori universali fiind utilizate în probleme concrete de asociere cum sunt: clasificare (inclusiv pentru clase neliniar separabile), predicţie, compresie etc.

1 Arhitectura

Dacă specificul problemei nu impune utilizarea unei anumite arhitecturi atunci se poate opta pentru o reţea constând din: un nivel de intrare, un nivel ascuns şi unul de ieşire. În ceea ce priveşte modul de conectare a unităţilor, varianta standard este de a conecta total nivelele vecine. În aceste condiţii singura problemă ce mai trebuie rezolvată este aceea a stabilirii numărului de unităţi de pe fiecare nivel. La stabilirea dimensiunii (complexităţii) reţelei se ţine cont de următoarele lucruri:

- Nivelele de intrare, respectiv de ieşire trebuie să aibă atâtea unităţi câte sunt necesare pentru a reprezenta datele de intrare, respectiv răspunsul reţelei. De exemplu, pentru o reţea proiectată să reprezinte o funcţie $\phi : R^N \to R^M$ se utilizează N unităţi pe nivelul de intrare, respectiv M pe nivelul de ieşire.

- Numărul de unităţi ascunse se stabilesc astfel încât reţeaua să fie suficient de complexă pentru a rezolva problema dar nu mai mult decât este necesar. Stabilirea numărului de unităţi ascunse se bazează fie pe rezultatele teoretice referitoare la capacitatea de reprezentare (rezolvare) a unei anumite arhitecturi, fie pe reguli euristice (de exemplu pentru o reţea cu N unităţi de intrare, M unităţi de ieşire şi un nivel ascuns se poate alege pentru acesta dimensiunea: $\sqrt{MN}$) fie pe tehnici de adaptare a acestora la problema de rezolvat.

Alegerea arhitecturii nu este o problemă simplă întrucât pentru aceeaşi problemă pot exista mai multe arhitecturi care permit rezolvarea ei. O arhitectură adecvată poate simplifica procesul de învăţare şi poate asigura o buna capacitate de generalizare.
Exemplu. Pentru reprezentarea funcţiei XOR există cel puţin două arhitecturi diferite (figura 1).

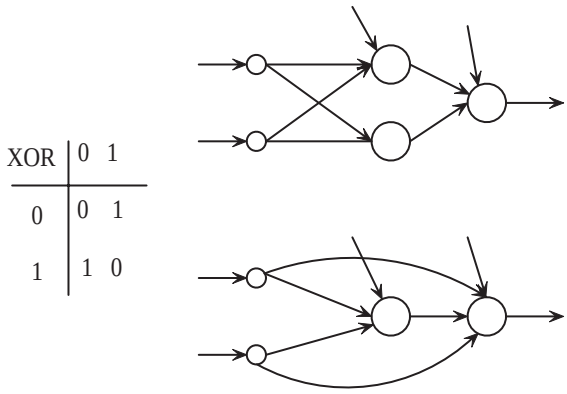


Figura 1: Arhitecturi pentru reprezentarea funcţiei XOR

2 Funcţionare

Modul în care reţeaua funcţionează depinde funcţiile de integrare şi de transfer ale unităţilor funcţionale precum şi de modul de conectare. Pentru o reţea cu K nivele funcţionale (dintre care $K-1$ sunt ascunse), completate cu unitate fictivă (cu excepţia ultimului) pentru modelarea pragului de activare (vezi fig. 2) semnalul, Y^K, de ieşire pentru un semnal de intrare, X^0, este:

$$Y^K = F^K(W^K F^{K-1}(W^{K-1} F^{K-2}(\ldots F^1(W^1 X^0)\ldots)))$$

unde $F^k(U) = (-1, f_1^k(u), \ldots, f_{N_k}^k(u))^T$ pentru $k = \overline{1, K-1}$ iar $F^K(U) = (f_1^K(u), \ldots, f_{N_K}^K(u))^T$. Operaţiile din relaţia de mai sus se efectuează la nivel vectorial, pentru un nivel k, W^k reprezentând matricea (N_k linii şi N_{k-1} coloane) ponderilor conexiunilor către acel nivel.

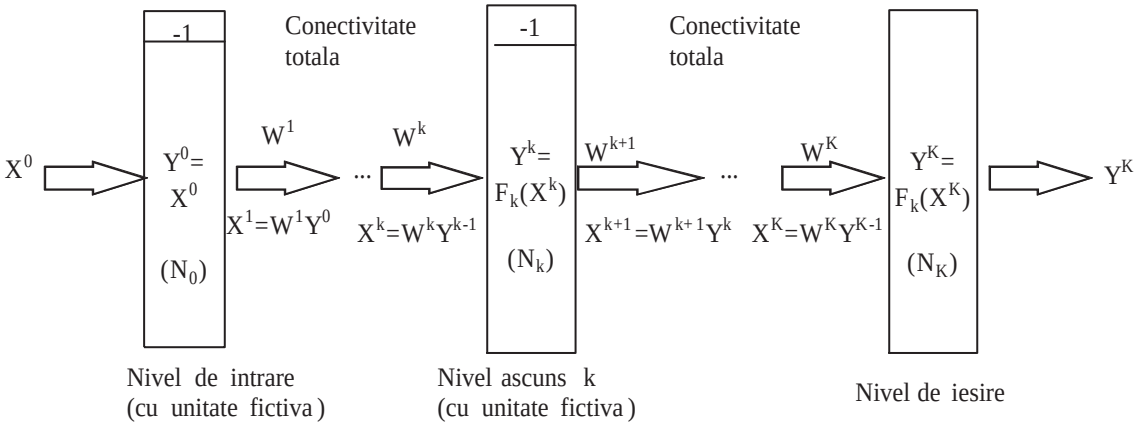


Figura 2: Structura unei reţele multinivel

Din punct de vedere matematic funcţionarea reţelei constă în compunerea unor transformări iar din punct de vedere algoritmic poate fi implementată printr-un algoritm iterativ simplu:

Algoritmul de funcţionare (FORWARD).

Pas 1. Se stabileşte semnalul de X^0 ($x_0^0 = -1$ iar $x_j^0 = x_j$ pentru $j = \overline{1, N_0}$) şi se determină: $x_i^1 = \sum_{j=0}^{N_0} w_{ij}^1 \cdot x_j^0$, $y_i^1 = f^1(x_i^1)$ pentru $i = \overline{1, N_1}$.

Pas 2. Pentru fiecare k de la 1 la $K-1$ se efectuează: $x_i^{k+1} = \sum_{j=0}^{N_k} w_{ij}^{k+1} y_j^k$, $y_i^{k+1} = f^{k+1}(x_i^{k+1})$ pentru $i = \overline{1, N_1}$ (pentru $k < K$ componenta 0 a vectorului Y^k este -1).

Rezultatul obţinut de reţea este interpretat în funcţie natura problemei. Astfel în cazul unei probleme de clasificare reţeaua se completează cu un nivel care determină maximul valorilor obţinute obţinându-se, astfel, indicatorul clasei căreia îi aparţine vectorul de intrare.

3 Capacitate de reprezentare

Problema capacităţii de reprezentare a reţelelor feedforward a fost rezolvată demonstrându-se că reţelele feedforward sunt *aproximatori universali*, adică pentru orice funcţie continuă (sau doar măsurabilă), ϕ, există o reţea feedforward cu cel puţin un nivel ascuns, care aproximează pe ϕ.

Există două abordări principale, bazate pe rezultate din teoria aproximarii funcţiilor, care furnizează condiţii suficiente pentru ca o reţea multi-nivel sa poată aproxima o funcţie arbitrară.

Prima abordare se bazează pe următorul rezultat teoretic [Hornik, 1991], [Cybenko, 1989], [Funahashi, 1989]:

> *Orice funcţie măsurabilă $\phi : D \subset \mathbf{R}^N \to \mathbf{R}^M$ (cu D o mulţime compactă) poate fi aproximată cu o acurateţe arbitrară de către o reţea feedforward caracterizată prin:*
>
> *(i) N unităţi de intrare, M unităţi de ieşire şi un nivel de unităţi ascunse (acurateţea aproximării depinde de numărul unităţilor ascunse).*
>
> *(ii) funcţii de integrare liniare ($G_{W_i}(X) = \sum_{j=0}^{N} w_{ij} x_j$) şi funcţii de transfer, $f : \mathbf{R} \to (a, b)$, continue (sau doar măsurabile) şi crescătoare cu $\lim_{u \to \infty} f(u) = b$, $\lim_{u \to -\infty} f(u) = a$.*

Acest rezultat sugerează modul de alegere a funcţiilor de integrare şi transfer, fără a furniza, însă, suficiente informaţii despre numărul de unităţi de pe nivelul ascuns. Acurateţea aproximării depinde însă de acesta.

A doua abordare porneşte de la o teoremă a lui A.K. Kolmogorov [1957] de reprezentare a funcţiilor continue de mai multe variabile prin funcţii continue de o singură variabilă:

> *Există $\lambda_1, \ldots, \lambda_N \in \mathbf{R}$ şi $\psi_1, \ldots, \psi_{2N+1} \in \mathcal{C}(I)$ cu proprietatea că pentru orice funcţie continuă $\phi : I^N \to \mathbf{R}$, există o funcţie continuă $h : R \to R$ astfel încât:*
>
> $$\phi(x_1, \ldots, x_N) = \sum_{k=1}^{2N+1} h(\lambda_1 \psi_k(x_1) + \ldots + \lambda_N \psi_k(x_N)).$$

Pornind de la acest rezultat, Hecht-Nielsen [1989] a enunţat:

> *Orice funcţie continuă $\phi : [0, 1]^N \to \mathbf{R}^M$ poate fi aproximată printr-o reţea feedforward caracterizată prin:*
>
> *(i) N unităţi de intrare, $(2N + 1)$ unităţi ascunse, M unităţi de ieşire.*

(ii) Unităţile ascunse au funcţia de integrare de forma:

$$g^k(X) = \sum_{j=1}^{N} \lambda_k \varphi(x_j + k\epsilon) + k,$$

iar funcţia de transfer $f^k(u) = u$, $k \in \{1, \ldots, 2N+1\}$. Funcţia φ este continuă monoton crescătoare, nu depinde de ϕ, dar depinde de N, iar ϵ este un număr raţional.

(iii) Unităţile de ieşire au funcţia de integrare:

$$g^i(X) = \sum_{k=1}^{2N+1} h^i(x_k),$$

iar funcţiile de transfer $f^i(u) = u$, $i \in \{1, \ldots, M\}$. h^i sunt funcţii continue care depind de ϕ şi ϵ.

Acest rezultat furnizează sugestii privind alegerea structurii fiecărui nivel, fără a indica însă tipul funcţiilor de integrare şi transfer.

Ambele abordări prezintă dezavantajul că nu conduc direct la metode efective de proiectare a reţelei. Pe de altă parte pentru o problemă concretă o reţea cu două sau mai multe nivele ascunse se poate comporta mai bine decât una cu un singur nivel ascuns, deşi rezultatele teoretice asigură faptul că reţelele cu un nivel ascuns sunt aproximatori universali.

4 Antrenarea unei reţele multinivel

Învăţarea presupune adaptarea reţelei la problema de rezolvat prin utilizarea informaţiilor de care se dispune despre problemă. Se poate desfăşura la unul dintre nivelele:

- *Ponderile conexiunilor.* Este varianta cea mai frecvent folosită. Se presupune că arhitectura este fixată şi rămân de stabilit doar valorile ponderilor.
- *Conexiuni.* Prin învăţare se ajustează nu numai ponderile conexiunilor ci şi conexiunile în sine care pot fi eliminate sau introduse. Partea fixă a reţelei este reprezentată de topologie (numărul unităţilor şi modul de amplasare) iar restul (conexiuni şi ponderi) sunt adaptabile.
- *Arhitectura.* Întreaga arhitectură este ajustabilă, prin procesul de învăţare adăugându-se sau eliminându-se unităţi şi/sau conexiuni şi în acelaşi timp determinându-se ponderile asociate. Este cea mai flexibilă variantă dar conduce la un algoritm complex de învăţare.

În continuare vom analiza cazul adaptaii ponderilor, considerând arhitectura fixată. În cazul problemelor de asociere se dispune de un set de antrenare de forma: $\{(X^1, d^1), \ldots, (X^L, d^L)\}$ unde $X^l \in R^N$ reprezintă data de intrare iar d^l reprezintă răspunsul corect asociat. În aceste condiţii comportarea reţelei poate fi apreciată prin intermediul erorii asociate setului de antrenare. Cel mai frecvent se optează pentru utilizarea erorii medii pătratice:

$$E(W^1, \ldots, W^K) = \frac{1}{L} \sum_{l=1}^{L} E_l(W^1, \ldots, W^K) \text{ unde } E_l(W^1, \ldots, W^K) = \frac{1}{2} \sum_{i=1}^{N_K} (d_i^l - y_i^{K,l})^2 \qquad (1)$$

cu $Y^{K,l}$ vectorul de ieşire corespunzător intrării X^l.

Scopul procesului de învăţare este determinarea reţelei care minimizează funcţia de eroare (1). În cazul în care adaptarea se referă doar la determinarea ponderilor problema învăţarii constă în a determina parametrii care minimizează pe E.

Determinarea ponderilor prin minimizarea funcţiei de eroare. În cazul în care funcţiile de transfer asociate unităţilor au proprietăţi matematice suficient de bune (de exemplu sunt continuu diferenţiabile) pentru rezolvarea problemei de minimizare poate fi folosită o metodă de tip gradient.

Majoritatea algoritmilor de învăţare supervizată bazaţi pe minimizarea unei funcţii de eroare folosesc o metodă de tip gradient astfel că structura lor generală cuprinde două etape principale: iniţializarea parametrilor şi procesul iterativ de ajustare. În funcţie de metoda de minimizare aleasă se poate construi câte un algoritm de învăţare. În continuare vom descrie un algoritm pentru determinarea ponderilor unei reţele feedforward folosind metoda gradientului simplu. Este vorba despre algoritmul BACKPROPAGATION, cel mai răspândit algoritm de antrenare supervizată.

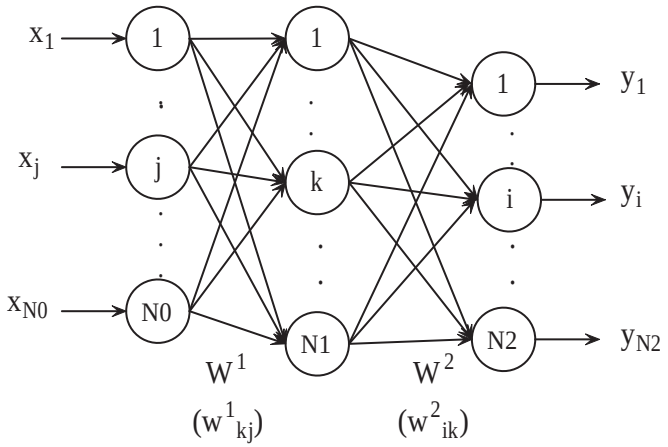


Figura 3: Structura unei reţele multinivel

Pentru a deduce relaţiile de ajustare vom considera o reţea cu un singur nivel ascuns (fig. 3) şi vom indicia elementele vectorilor asociaţi nivelului de intrare cu j, cele ale vectorilor asociaţi nivelului ascuns cu k iar pentru nivelul de ieşire cu i. În felul acesta vom omite indicele nivelului la vectorii X şi Y. Pentru a deduce relaţiile de ajustare calculăm componentele gradientului funcţiei $E_l(W^1, W^2)$ prima dată în raport cu componentele w^2_{ik} ale matricii W^2 după care în raport cu componentele w^1_{kj} ale matricii W^1. Reamintim că $y_i = f(x_i) = f(\sum_{k=0}^{N_1} w^2_{ik} y_k)$, $i = \overline{1, N_2}$ iar $y_k = f(x_k) = f(\sum_{j=0}^{N_0} w^1_{kj} y_j)$ pentru $k = \overline{1, N_1}$ şi -1 pentru $k = 0$. Componentele y_j sunt determinate de vectorul de intrare. Dacă acesta este X^l, atunci $y_0 = -1$ şi $y_j = x^l_j$ pentru $j = \overline{1, N_0}$.

Cu aceste notaţii, derivatele parţiale ale lui $E_l(w^1_{kj}, w^2_{ik})$ pot fi scrise:

$$\frac{\partial E_l(w^1_{kj}, w^2_{ik})}{\partial w^2_{ik}} = -(d^l_i - y_i)\frac{\partial y_i(w^2_{ik})}{\partial w^2_{ik}} = -f'(x_i)(d^l_i - y_i)y_k = -\delta^l_i y_k \tag{2}$$

cu $\delta_i^l \stackrel{not}{=} f'(x_i)(d_i^l - y_i)$.

$$\frac{\partial E_l(w_{kj}^1, w_{ik}^2)}{\partial w_{kj}^1} = -\sum_{i=1}^{N_2} \frac{\partial y_i(w_{kj}^1)}{\partial w_{kj}^1}(d_i^l - y_i) = -\sum_{i=1}^{N_2} f'(x_i) w_{ik}^1 f'(x_k) x_j (d_i^l - y_i) =$$

$$= -x_j f'(x_k) \sum_{i=1}^{N_2} w_{ik}^1 f'(x_i)(d_i^l - y_i) = -x_j f'(x_k) \sum_{i=1}^{N2} w_{ik}^2 \delta_i^l = -x_j \delta_k^l, \tag{3}$$

cu $\delta_k^l = f'(x_k) \sum_{i=1}^{N2} w_{ik}^2 \delta_i^l$.

Modul de calcul al lui δ_k^l sugerează transmiterea prin reţea a semnalului de eroare corespunzător nivelului de ieşire (δ_i^l) în sens invers celui în care circulă semnalele în faza de funcţionare. Aceasta analogie stă la baza denumirii algoritmului: "(error) backpropagation"="propagare înapoi (a erorii)". Pe baza relaţiilor (2) şi (3) se deduc relaţiile de ajustare corespunzătoare exemplului l din setul de antrenare (X^l, d^l). Cum eroarea globală, E, se obţine însumând erorile parţiale, E_l, ajustările corespunzătoare întregului set de antrenare s-ar obţine prin însumarea celor corespunzătoare fiecărui exemplu în parte.

Din punctul de vedere al modului de aplicare a ajustărilor în raport cu parcurgerea setului de antrenare există două variante ale algoritmului Backpropagation:

- *Serială.* (fig. 4) La fiecare iteraţie se parcurge întreg setul de antrenare şi ajustările se operează după prelucrarea fiecărui exemplu. Nu corespunde unei implementări fidele a metodei gradientului însă prezintă avantajul că nu trebuie cumulate toate ajustările înainte de a fi aplicate.

- *Pe blocuri.* (fig. 5) La fiecare iteraţie se parcurge întreg setul de antrenare, se cumulează toate ajustările după care se aplică efectiv. Provine din aplicarea directă a algoritmului de minimizare asupra funcţiei E. Prezintă avantajul că este mai robust în raport cu eventualele erori din datele de intrare.

Etapa 1. Iniţializări. Se iniţializează:
matricile $W^1, \ldots, W^K$ cu valori aleatoare mici (uniform repartizate în (-1,1))
rata de învăţare $\eta(0)$
toleranţa la învăţare E^*
numărul maxim de epoci de antrenare p_{max}
contorul de epoci $p = 0$
Etapa 2. Proces iterativ de ajustare.
REPEAT
FOR $l = \overline{1, L}$ DO
Etapa FORWARD: se aplică algoritmul FORWARD pentru $X^0 = X^l$
Etapa BACKWARD:
$\delta_i^{K,l} = f_K'(x_i^{K,l})(d_i^l - y_i^{K,l})$, $i = \overline{1, N_K}$ (calculul erorii la nivelul de ieşire)
FOR $k = \overline{K-1, 1}$ DO (propagarea erorii înapoi în reţea)
$\delta_j^{k,l} = f_k'(x_j^{k,l}) \sum_{i=1}^{N_{k+1}} w_{ij}^{k+1} \delta_i^{k+1,l}$, $j = \overline{1, N_k}$
Etapa ajustării propriu-zise:
$w_{ij}^k = w_{ij}^k + \eta(p)\delta_i^{k,l} y_j^{k-1}$, $i = \overline{1, N_k}$, $j = \overline{0, N_{k-1}}$, $k = \overline{1, K}$
Recalcularea erorii:
$E = 0$
FOR $l = \overline{1, L}$ DO
Calculează $Y^{K,l}$ aplicând din nou algoritmul FORWARD
$E = E + \|d^l - Y^{K,l}\|^2$
$E = E/L$
p=p+1
UNTIL $(E \leq E^*)$ OR $(p > p_{max})$.

Figura 4: Algoritmul backpropagation în varianta serială

Etapa 1. Iniţializări. Se iniţializează:
matricile $W^1, \ldots, W^K$ cu valori aleatoare mici (uniform repartizate în (-1,1))
rata de învăţare $\eta(0)$
toleranţa la învăţare E^*
numărul maximum de epoci de antrenare p_{max}
contorul de epoci $p = 0$
Etapa 2. Proces iterativ de ajustare.
REPEAT
$\Delta_{ij}^k = 0$ pentru $k = \overline{1,K}$, $i = \overline{1,N_k}$, $j = \overline{0,N_{k-1}}$
FOR $l = \overline{1,L}$ DO
Etapa FORWARD: se aplică algoritmul FORWARD pentru $X^0 = X^l$
Etapa BACKWARD:
$\delta_i^{K,l} = f_K'(x_i^{K,l})(d_i^l - y_i^{K,l})$, $i = \overline{1,N_K}$ (determinarea erorii pentru nivelul de ieşire)
FOR $k = \overline{K-1,1}$ (propagarea erorii înapoi în reţea)
$\delta_j^{k,l} = f_k'(x_j^{k,l}) \sum_{i=1}^{N_{k+1}} w_{ij}^{k+1} \delta_i^{k+1,l}$, $j = \overline{1,N_k}$
Cumularea ajustărilor
$\Delta_{ij}^k = \Delta_{ij}^k + \delta_i^{k,l} y_j^{k-1}$, $i = \overline{1,N_k}$, $j = \overline{0,N_{k-1}}$, $k = \overline{1,K}$
Etapa ajustării propriu-zise:
$w_{ij}^k = w_{ij}^k + \eta(p)\Delta_{ij}^k$, $i = \overline{1,N_k}$, $j = \overline{0,N_{k-1}}$, $k = \overline{1,K}$
Recalcularea erorii:
$E = 0$
FOR $l = \overline{1,L}$ DO
Calculează $Y^{K,l}$ aplicând din nou algoritmul FORWARD
$E = E + \|d^l - Y^{K,l}\|^2$
$E = E/L$
p=p+1
UNTIL $(E \leq E^*)$ OR $(p > p_{max})$.

Figura 5: Algoritmul backpropagation în varianta "pe blocuri"

Variante ale algoritmului Backpropagation

Pornind de la algoritmul BackPropagation standard pot fi dezvoltate diferite variante care diferă între ele prin:

- modul de alegere a ratei de învăţare: constantă sau adaptivă;
- relaţiile de ajustare (determinate de algoritmul de minimizare utilizat, ce poate fi diferit de algoritmul gradientului simplu: algoritmi de tip gradient conjugat, algoritmi de tip Newton, algoritmi aleatori de descreştere, algoritmi genetici etc.);
- modul de iniţializare a parametrilor: iniţializare aleatoare sau bazată pe un algoritm de căutare.
- parcurgerea setului de antrenare (influenţează doar varianta serială a algoritmului): secvenţială sau aleatoare;
- funcţia de eroare: pe lângă eroarea medie pătratică se pot folosi şi măsuri ale erorii specifice problemei de rezolvat (de exemplu în cazul problemelor de clasificare se poate folosi şi eroarea bazată pe entropie);
- criteriul de oprire: pe lângă criteriul bazat pe numărul maxim de epoci şi pe eroarea corespunzătoare setului de antrenare se pot folosi şi criterii legate de eroarea pe setul de validare şi de mărimea ajustărilor din cadrul ultimei epoci.

Motivaţia dezvoltării unor variante ale algoritmului standard o reprezintă faptul că acesta prezintă o serie de inconveniente:

- *Convergenţa lentă*: necesită multe epoci pentru a atinge o valoare suficient de mică pentru eroarea pe setul de antrenare.
- *Blocarea în minime locale*: o data ce algoritmul ajunge într-un minim local al funcţiei de eroare, algoritmul nu permite evadarea din acest minim pentru a atinge optimul global.
- *Stagnarea (paralizarea)*: algoritmul stagnează într-o regiune care nu este neapărat în vecinătatea unui minim local ca urmare a faptului că ajustările aplicate parametrilor sunt foarte mici.
- *Supraantrenarea:* reţeaua asigură o bună aproximare pe setul de antrenare însă posedă o capacitate de generalizare redusă.

1 Problema convergenţei lente.

1.1 Cauze.

Algoritmul BackPropagation standard, fiind bazat pe metoda gradientului simplu, necesită utilizarea unei rate mici de învăţare pentru a asigura convergenţa. Aceasta alegere conduce la necesitatea unui număr mare de epoci.

1.2 Soluţii.

Există soluţii euristice (modificarea ratei de învăţare în funcţie de evoluţia valorii erorii şi varianta momentului) şi soluţii bazate pe utilizarea unei alte metode de minimizare.
Adaptarea ratei de învăţare. Proprietăţile metodei gradientului sugerează că pentru a se asigura convergenţa algoritmului BackPropagation este necesară folosirea unei rate mici de învăţare. Aceasta conduce însă la o durată mare a procesului de învăţare. În cazul unei funcţii pătratice de eroare $E(W) = \frac{1}{2}W^T AW + b^T W + c$ (reţelele liniare) se poate determina valoarea maximă a ratei de învăţare care nu provoacă oscilaţii, $\eta = 2/\lambda_{max}(A)$ ($\lambda_{max}(A)$ este valoarea proprie maximă a a matricii A). O bună viteză de convergenţă poate fi atinsă folosind o valoare a ratei apropiată de cea maximă.

În cazul general nu mai există o singură valoare critică pentru η deoarece funcţia de eroare nu mai este pătratică. Din acest motiv pare natural să se adapteze valoarea lui η în funcţie de caracteristicile suprafeţei de eroare.

Rata de invăţare influenţează comportarea algoritmului de învăţare în modul următor: valori mici ale ratei pot conduce la convergenţă lentă iar valori mari pot conduce la oscilaţii în procesul de căutare a minimului funcţiei de eroare.

Notăm cu $\eta(p)$ rata corespunzătoare epocii p iar cu $E(p)$ şi $E(p-1)$ valorile erorii corespunzătoare epocii p respectiv epocii anterioare. La sfârşitul fiecărei epoci, după calcularea erorii se efectuează următoarea analiză:

- Dacă $E(p) > (1+\gamma)E(p-1)$ atunci se ignoră ajustările efectuate în cadrul epocii p şi se modifica rata: $\eta(p) = a\eta(p-1)$, $0 < a < 1$;
- Dacă $E(p) < (1-\gamma)E(p-1)$ atunci se acceptă ajustările efectuate în cadrul epocii p şi se modifica rata: $\eta(p) = b\eta(p-1)$, $1 < b < 2$;
- Dacă $(1-\gamma)E(p-1) \leq E(p) \leq (1+\gamma)E(p-1)$ atunci se acceptă ajustările efectuate în cadrul epocii p şi rata se păstrează nemodificată.

Valoarea γ utilizată în regulile de mai sus se alege mică (de exemplu, $\gamma = 0.05$).

În afara variantelor euristice de alegere a ratei de învăţare se pot folosi şi metode de minimizare unidimensională pentru a alege la fiecare epocă rata care asigură descreşterea maximă.
Varianta cu moment. O altă tehnică euristică de accelerare a convergenţei algoritmului Backpropagation constă în utilizarea ca ajustări în epoca curentă a unei combinaţii convexe dintre ajustările aplicate la epoca anterioară şi ajustările calculate la epoca curentă pe baza semnalelor de eroare. Intuitiv, utilizarea ajustărilor de la epoca anterioară joacă rolul unei inerţii aplicate în procesul de căutare a minimului funcţiei de eroare. Tehnica momentului poate fi aplicată atât variantei seriale cât şi celei batch a algoritmului. Pentru a ilustra tehnica o vom exemplifica pe structura generală a variantei batch a algoritmului (figura **??**. Pentru a simplifica notatiile atât ponderile cât şi ajustările şi valorile momentului vor fi desemnate prin matricile generice: W, Δ respectiv

Etapa 1. Iniţializări. $W = \text{rand}(-1, 1)$, $M = 0$
Etapa 2. Proces iterativ de ajustare.
REPEAT
  $\Delta := 0$
  FOR $l := \overline{1, L}$ DO
    Etapa FORWARD
    Etapa BACKWARD
    Cumularea ajustărilor
      $\Delta_{ij}^k := \Delta_{ij}^k + \delta_i^{k,l} y_j^{k-1}$, $i = \overline{1, N_k}$, $j = \overline{0, N_{k-1}}$, $k = \overline{1, K}$
  Etapa ajustării propriu-zise:
    $W := W + \eta(1-\alpha)\Delta + \alpha M$ (ajustarea ponderilor)
    $M := \eta(1-\alpha)\Delta + \alpha M$ (reţinerea momentelor pentru epoca următoare)

  Recalcularea erorii: $E(W) = \frac{1}{2L} \sum_{l=1}^{L} \sum_{i=1}^{n} (d_i^l - y_i^l)^2$
  p=p+1
UNTIL $(E \leq E^*)$ OR $(p > p_{max})$.

Figura 1: Structura generală a variantei cu moment

M. Prelucrările corespunzătoare etapelor FORWARD şi BACKWARD sunt cele de la algoritmul BackPropagation standard. Valoarea parametrului asociat termenului moment (α) se alege între 0.2 şi 0.9.
Utilizarea altei metode de minimizare. Metoda gradientului simplu utilizată în deducerea ajustărilor din cadrul algoritmului Backpropagation standard are viteză mică de convergentă. Metode de minimizare caracterizate prin viteză mai mare de convergenţă sunt metodele de tip gradient conjugat şi cele de tip Newton.

2 Problema blocării în minime locale

2.1 Cauze.

Principala cauză a blocării într-un minim local este faptul că metodele de tip gradient sunt metode de optimizare locală, permiţând determinarea punctului de minim aflat în vecinătatea aproximaţiei iniţiale (valorile iniţiale ale ponderilor).

2.2 Soluţii.

Pentru a evita blocarea în minime locale se pot utiliza metode euristice sau poate fi înlocuită metoda de minimizare locală cu una de minimizare globală.
Metode euristice. Se bazează pe ideea că introducerea unor perturbaţii aleatoare în algoritmul de învăţare permite evadarea dintr-un minim local. Cea mai simplă tehnică (care nu implică modificarea algoritmului) este de a relua algoritmul de antrenare de la valori diferite ale ponderilor iniţiale. O altă varianta constă în perturbarea aleatoare, pe parcursul antrenării, a valorilor ponderilor ($w_{ij} := w_{ij} + \xi_{ij}$.

Etapa 1. Iniţializări. $W = \text{rand}(-1, 1)$
Etapa 2. Proces iterativ de ajustare.
Calcul eroare $E(W)$ pe setul de antrenare
REPEAT
Generare ajustari aleatoare Δ ($\Delta_{ij} = rand(-1, 1)$)
Calcul ponderi modificate: $W' = W + \Delta$
Recalculare eroare $E(W')$
IF $E(W') < E(W)$ THEN se accepta ajustările ($W = W'$)
ELSE ponderile rămân nemodificate
p=p+1
UNTIL ($E(W) \leq E^*$) OR ($p > p_{max}$).

Figura 2: Algoritm de căutare aleatoare

Metode de optimizare globală. Utilizând metode de optimizare globală pentru determinarea punctului de minim a funcţiei de eroare este posibilă evitarea blocării în minime locale. Metodele cel mai frecvent utilizate sunt:

- algoritmi de căutare aleatoare;
- algoritmi de tip "simulated annealing" (călire simulată);
- algoritmi genetici.

Considerăm ca exemplu un algoritm simplu de căutare aleatoare (fig 2). Această variantă simplă de algoritm aleator nu este eficientă însă poate fi utilizată ca tehnică de alegere a valorilor iniţiale ale ponderilor în Backpropagation. Rolul căutării aleatoare este să plaseze valorile ponderilor în zona minimului global pe când rolul algoritmului Backpropagation este să rafineze valorile ponderilor.

3 Problema stagnării

Stagnarea algoritmului de învăţare înseamnă stoparea procesului de îmbunătăţire a ponderilor ca urmare a unor valori prea mici ale termenilor de ajustare. Situaţia este întâlnită când se ajunge într-o porţiune plată a suprafeţei de eroare.

3.1 Cauze

Principala cauză a stagnării o reprezintă valorile mari ale semnalelor de intrare în unităţile funcţionale bazate pe funcţii de transfer de tip sigmoidal. Derivatele funcţiilor sigmoidale de transfer devin foarte mici pentru valori mari în modul ale argumentului (de exemplu, pentru argumente mai mari în modul decât 10 derivata tangentei hiperbolice este mai mică decât 10^{-8}) . Cum valorile derivatelor intervin in termenul de ajustare a ponderilor rezultă că ponderile nu vor fi modificate semnificativ. Valori mari ale semnalelor de intrare în unităţile funcţionale de pe nivelele ascunse pot fi produse de valori mari ale ponderilor sau ale semnalelor de intrare în reţea.

3.2 Soluţii

Stagnarea poate fi evitată prin penalizarea valorilor mari ale ponderilor sau prin eliminarea derivatelor din termenii de ajustare.

Prima varianta poate fi aplicată prin adăugarea unui termen suplimentar funcţiei de eroare prin care se penalizează valorile mari ale ponderilor. Astfel în locul erorii medii pătratice, E se foloseşte $E_m(W) = E(W) + \lambda \sum_{i,j} w_{ij}^2$, al doilea termen cuprinzând suma pătratelor tuturor ponderilor reţelei. Parametrul $\lambda > 0$ reflectă importanţa care se acordă penalizării valorilor mari ale ponderilor.

A doua tehnică poate fi implementată prin înlocuirea derivatelor parţiale din ajustări cu valori calculate pe baza unei relaţii de recurenţă. Pe această idee se bazează algoritmul RPROP (Resilient BackPropagation) în care ajustarea corespunzătoare parametrului w_{ij} la epoca p, $\Delta w_{ij}(p)$ se stabileşte în funcţie de semnul derivatei parţiale a funcţiei de eroare:

$$\Delta w_{ij}(p) = \begin{cases} -\Delta_{ij}(p) & \text{if } \frac{\partial E(W(p))}{\partial w_{ij}} > 0 \\ 0 & \text{if } \frac{\partial E(W(p))}{\partial w_{ij}} = 0 \\ \Delta_{ij}(p) & \text{if } \frac{\partial E(W(p))}{\partial w_{ij}} < 0 \end{cases}$$

unde termenii de ajustare sunt determinaţi pe baza regulii:

$$\Delta_{ij}(p) = \begin{cases} a\Delta_{ij}(p-1) & \text{if } \frac{\partial E(W(p-1))}{\partial w_{ij}} \frac{\partial E(W(p))}{\partial w_{ij}} > 0 \\ \Delta_{ij}(p-1) & \text{if } \frac{\partial E(W(p-1))}{\partial w_{ij}} \frac{\partial E(W(p))}{\partial w_{ij}} = 0 \\ b\Delta_{ij}(p-1) & \text{if } \frac{\partial E(W(p-1))}{\partial w_{ij}} \frac{\partial E(W(p))}{\partial w_{ij}} < 0 \end{cases}$$

cu parametrii $0 < b < 1 < a$ (de exemplu, $b = 0.5$, $a = 1.2$). Valorile iniţiale ale termenilor de ajustare, $\Delta(0)$ se aleg relativ mici (de exemplu 0.1). În RPROP, se impune o limită superioară pentru Δ_{ij} astfel că toate valorile ajustărilor sunt trunchiate la o valoare prestabilită Δ_{max} (de exemplu $\Delta_{max} = 5.0$).

4 Problema supraantrenării

Supraantrenarea este situaţia în care eroarea reţelei este mică pe setul de antrenare însă mare pentru date de intrare ce nu aparţin acestui set. Aceasta înseamnă că reţeaua nu are putere de generalizare.

4.1 Cauze

Cauzele supraantrenării pot fi mai multe: număr prea mare de unităţi ascunse, volum mic al setului de antrenare, toleranţă prea mică la antrenare, număr mare de epoci etc. Acestea fac ca reţeaua să modeleze inclusiv zgomotele din setul de antrenare reducând abilitatea reţelei de a generaliza pornind de la exemple. Nu pentru orice problemă oricare dintre elementele enumerate mai sus provocă supraantrenare. Experimente numerice au arătat că un număr prea mare de epoci generează mai frecvent supraantrenare decât un număr mare de unităţi ascunse.

4.2 Soluţii

Atunci când cauza supraantrenării o reprezintă numărul prea mare de epoci, o soluţie simplă o reprezintă controlul numărului de epoci prin modificarea condiţiei de oprire. Aceasta se poate

realiza prin a calcula atât eroarea pe setul de antrenare cât şi eroarea pe un set de validare disjunct de cel de antrenare şi de a opri antrenarea în momentul în care eroarea pe setul de validare începe să crească. Pentru a construi setul de validare, setul de informaţii cunoscute se divide în două seturi disjuncte. O tehnică clasică este cea a validării încrucişate ("cross-validation") caracterizată prin faptul că se construieşte succesiv câte un set de validare prin extragerea a k elemente din setul de antrenare. La fiecare nouă construire a setului de antrenare se aleg alte k elemente. În felul acesta se evită prezenţa sistematică a aceloraşi elemente în setul de antrenare, respectiv în cel de validare.

În cazul în care se presupune că numărul de unităţi ascunse este cauza supraantrenării soluţia este de a aplica tehnici de adaptare a numărului de unităţi ascunse. Există două categorii principale: (i) *constructive*: se porneşte cu un număr mic de unităţi şi dacă reţeaua nu are suficientă capacitatea de reprezentare (de exemplu învăţarea nu progresează) se mai adaugă unităţi; (ii) *destructive*: se porneşte cu un număr mare de unităţi şi dacă reţeaua nu are suficientă capacitate de generalizare atunci se elimină unităţi (pentru eliminare sunt selectate unităţile a căror contribuţie la semnalul de ieşire este relativ mică).

Reţele cu funcţii de transfer radiale

1 Arhitectură şi funcţionare.

Reţelele cu funcţii de transfer radiale, numite şi reţele de tip RBF ("Radial Basis Functions") sunt constituite din N unităţi de intrare, K unităţi ascunse şi M unităţi de ieşire. Diferenţa dintre reţelele feedforward multinivel şi cele RBF constă în funcţiile de integrare şi cele de transfer specifice nivelului ascuns. Semnalul produs de unitatea k aflată pe nivelul ascuns, corespunzător unui semnal de intrare X, este:

$$y_k = g_k(\|X - C^k\|)$$

unde $C^k = (c_{k1}, \ldots, c_{kN})$ este vectorul ponderilor conexiunilor către unitatea ascunsă k numit şi *centrul* sau *prototipul* acesteia. Deci funcţia de integrare asociată unităţilor ascunse se bazează pe calculul unei distanţe dintre vectorul de intrare şi centrul corespunzător, distanţa cea mai frecvent folosită fiind cea euclidiană ($\|X - C^k\|^2 = \sum_{i=1}^{N}(x_j - c_{kj})^2$). Funcţiile de transfer, g_k, sunt diferite de cele sigmoidale, fiind similare funcţiilor *nucleu* folosite în teoria aproximării. Cele mai frecvent folosite sunt cele caracterizate prin simetrie radială care au proprietatea de localitate (produc valori semnificative doar pentru valori mici ale argumentului şi tind către zero în cazul valorilor mari). Exemple de astfel de funcţii sunt cea gaussiană ($g_k(u) = \exp(\frac{-u^2}{2\sigma_k^2})$, cea de tip Cauchy ($g_k(u) = 1/(u^2 + \sigma^2)$) şi $g_k(u) = 1/\sqrt{(u^2 + \sigma^2)}$.

Ieşirea produsă de reţea, pentru un semnal de intrare $X \in R^N$ este vectorul $Y \in R^M$ având componentele:

$$y_i = \sum_{k=1}^{K} w_{ik} g_k(\|X - C^k\|) - w_{i0}, \quad i = \overline{1, M}$$

unde $W = (w_{ik})_{i=\overline{1,N}, k=\overline{0,K}}$ este matricea ponderilor conexiunilor dintre nivelul ascuns şi cel de ieşire, iar $\|cdot\|$ reprezintă norma euclidiană. În continuare vom nota cu z_k semnalul produs de către unitatea ascunsă, k, adică $z_k = g_k(\|X - C^k\|)$. Se poate folosi şi o variantă normalizată, în care z_k se calculează după cum urmează:

$$z_k = \frac{g_k(\|X - C^k\|)}{\sum_{l=1}^{K} g_l(\|X - C^l\|)}.$$

Caracterul de localitate al funcţiei de transfer face ca fiecare unitate ascunsă să producă ieşiri semnificative doar pentru semnale de intrare apropiate de centrul corespunzător unităţii. Fiecare unitate ascunsă reacţionează doar la semnale provenind din zona care le este asociată, zonă numită *câmp receptiv*. O bună comportare a reţelei se obţine dacă domeniul datelor de intrare este acoperit de câmpurile receptive ale unităţilor ascunse. Dimensiunea câmpurilor receptive depinde de parametrul σ_k corespunzător. O valoare mare pentru σ_k induce o dimensiune mare a câmpului receptiv pe când o valoare mică a lui σ_k conduce la un câmp receptiv restrâns.

2 Capacitatea de reprezentare.

La fel ca şi reţelele feedforward cu cel puţin un nivel ascuns de unităţi sigmoidale şi reţelele RBF sunt aproximatori universali:

> *Fie $S \subset R^N$ o mulţime compactă şi $\varphi : S \to R^M$ o funcţie continuă. Pentru orice $\epsilon > 0$ există o reţea RBF cu N unităţi de întrare, M unităţi de ieşire şi K unităţi ascunse astfel încât $\sup_{X \in S} \|\varphi(X) - Y\| < \epsilon$, Y fiind vectorul de ieşire corespunzător vectorului de intrare X.*

Calitatea aproximării depinde de numărul de unităţi ascunse. Diferenţa esenţială dintre reţelele feedforward multinivel şi cele RBF este dată de faptul că primele realizează o aproximare globală iar celelalte o aproximare locală. Reţelele RBF asigură o partiţionare a domeniului datelor de intrare caracterizată prin faptul că pentru fiecare zonă există o unitate ascunsă care produce o valoare semnificativă atunci când primeşte semnale din zona respectivă. Dacă semnalul de intrare se află la frontiera dintre două zone atunci unităţile ascunse corespunzătoare ambelor zone vor produce valori semnificative astfel că răspunsul reţelei va fi o medie ponderată a valorilor asociate acestor unităţi ascunse. În felul acesta, reţeaua asigură o trecere "netedă" de la o zonă la alta.

Reţelele RBF pot fi aplicate pentru rezolvarea problemelor de clasificare, de aproximare şi pentru predicţie în serii temporale.

3 Algoritmi de învăţare.

Parametrii ajustabili ai unei reţele RBF sunt: centrele C^k ale unităţilor ascunse, ponderile conexiunilor către nivelul de ieşire, W şi parametrii σ_k care controlează lărgimile câmpurilor receptive.

În funcţie de modul în care se determină parametrii corespunzători nivelului ascuns şi cei corespunzători nivelului de ieşire există două categorii principale de algoritmi: cu antrenare simultană, respectiv cu antrenare separată a acestor parametri.

În ambele situaţii antrenarea este cu caracter supervizat şi se bazează pe un set de antrenare: $\{(X^1, d^1), \ldots, (X^L, d^L)\}$.

3.1 Determinarea simultană a tuturor parametrilor

Acest mod de antrenare se bazează pe ideea folosită la reţelele feedforward standard constând în aplicarea unei metode de minimizare a erorii pătratice medii calculată pe setul de antrenare, adică:

$$E(C, \sigma, W) = \frac{1}{L} \sum_{l=1}^{L} E_l(C, \sigma, W)$$

$$E_l(C, \sigma, W) = \frac{1}{2} \sum_{i=1}^{M} (d_i^l - (\sum_{k=1}^{K} w_{ik} g_k(\|X - C^k\|) - w_{i0}))^2.$$

Pentru a aplica o metodă de tip gradient pentru minimizarea lui E trebuie calculate derivatele parţiale:

$$\frac{\partial E_l}{\partial w_{i0}}(C, \sigma, W) = d_i^l - y_i^l$$

$$\frac{\partial E_l}{\partial w_{ik}}(C, \sigma, W) = -z_k(d_i^l - y_i^l)$$

$$\frac{\partial E_l}{\partial c_{kj}}(C,\sigma,W) = -\frac{1}{\sigma_k^2}\sum_{i=1}^{M}(d_i^l - y_i^l)w_{ik}z_k(x_j^l - c_{kj})$$

$$\frac{\partial E_l}{\partial \sigma_k}(C,\sigma,W) = -\frac{1}{\sigma_k^3}\sum_{i=1}^{M}(d_i^l - y_i^l)w_{ik}z_k\|X^l - C^k\|^2.$$

Folosind ideea de la varianta serială a algoritmului BackPropagation se obţine un algoritm a cărui structură generală este ilustrată in fig. 1. Fiind un caz particular al algoritmului BackPropagation, adaptat pentru o reţea cu un singur nivel ascuns,

astfel că problemele ce apar la aplicarea algoritmului BP (minime locale, convergenţă lentă etc) apar şi în acest caz.

Iniţializarea aleatoare a tuturor parametrilor
REPEAT
  FOR $l = \overline{1,L}$:
    $w_{ik} = w_{ik} - \eta\frac{\partial E_l}{\partial w_{ik}}(W,\sigma,C), \quad i = \overline{1,M}, k = \overline{0,K}$
    $c_{kj} = c_{kj} - \eta\frac{\partial E_l}{\partial c_{kj}}(W,\sigma,C), \quad k = \overline{1,K}, j = \overline{1,N}$
    $\sigma_k = \sigma_k - \eta\frac{\partial E_l}{\partial \sigma_k}(W,\sigma,C), \quad k = \overline{1,K}$
  Recalcularea erorii
UNTIL *este satisfăcută o condiţie de oprire*

Figura 1: Antrenarea reţelei RBF cu algoritm de tip BackPropagation

3.2 Determinarea separată a parametrilor

Vom considera procesul de antrenare descompus în trei procese distincte care se desfăşoară în următoarea ordine:

- determinarea centrelor $C^k \in R^N$, $k = \overline{1,K}$;
- determinarea lărgimilor câmpurilor receptive, σ_k, $k = \overline{1,K}$;
- determinarea ponderilor conexiunilor dintre nivelul ascuns şi cel de ieşire, w_{ik}, $i = \overline{1,M}$, $k = \overline{0,K}$.

3.2.1 Determinarea centrelor.

Centrele reprezintă vectori prototip selectaţi din domeniul datelor de intrare, pe baza cărora se realizează partiţionarea domeniului de intrare în mici regiuni (câmpurile receptive) de fiecare dintre acestea fiind responsabilă o unitate ascunsă.

În cazul în care setul de antrenare nu este prea mare atunci se poate considera că datele de intrare prezente în acesta determină centrele. Astfel, $K = L$ şi $C^k = X^k$, pentru fiecare $k \in \{1,\ldots,K\}$. Un exemplu simplu de astfel de situaţie îl reprezintă problema aproximării funcţiei booleene XOR. În acest caz datele de intrare sunt vectorii: $(0,0)$, $(0,1)$, $(1,0)$ şi $(1,1)$ iar în cadrul reţelei se consideră 4 unităţi ascunse cărora li se afectează ca centri aceşti vectori.

Dacă, însă, setul de antrenare conţine multe date (unele cu caracter redundant) atunci pentru setarea centrelor se aleg doar o parte dintre acestea. O variantă simplă de alegere o reprezintă

selecţia aleatoare a K date însă o variantă mai adecvată o reprezintă aplicarea unui algoritm de grupare (clustering).

Problematica grupării este următoarea: se dau L vectori $X^1, X^2, \ldots, X^L$ şi se pune problema amplasării lor în $K < L$ clase (grupe sau clusteri) astfel încât elementele aflate într-o clasă să fie mai apropiate între ele decât cele aflate în clase diferite. Noţiunea de apropiere se stabileşte pornind de la criteriul ce a stat la baza grupării. Un astfel de criteriu îl poate reprezenta distanţa euclidiană.

În rezolvarea unei probleme de grupare pot interveni două situaţii: numărul de clase, K, este cunoscut, respectiv este necunoscut.

Un algoritm simplu de grupare, bazat pe cunoaşterea numărului de clase, este *algoritmul mediilor* bazat pe ideea că centrul (prototipul) unei clase poate fi reprezentat de media aritmetică a elementelor clasei, iar afectarea unui vector la o clasă sau alta se face pe criteriul distanţei minime (fig. 2).

Se iniţializează centrele $C^1, C^2, \ldots, C^K$ cu vectori selectaţi aleator din setul de antrenare.
REPEAT
Construirea claselor: fiecare vector X^l este afectat clasei ω_k care verifică:
$d(X^l, C^k) \leq d(X^l, C^i)$, pentru orice $i \in \{1, \ldots, K\}$ d fiind distanţa euclidiană
Recalcularea centrelor: pentru fiecare clasă ω_k se recalculează centrul:
$$C^k = \frac{1}{\text{card } \omega_k} \sum_{X \in \omega_k} X, \quad k = \overline{1, K}.$$
UNTIL *este îndeplinit un criteriu de oprire.*

Figura 2: Algoritm de grupare. Metoda K-mediilor

Observaţii.
1. Algoritmul este oprit fie când la ultima parcurgere clasele nu s-au modificat fie când s-au efectuat un număr mare de iteraţii.
2. Prin acest algoritm se pot alege ca prototipuri vectori ce nu fac parte din setul iniţial de antrenare.
3. Algoritmul mediilor conduce la determinarea acelor centri care minimizează

$$\sum_{k=1}^{K} \sum_{X \in \omega_k} d(X, C^k).$$

Dacă numărul de clase nu este cunoscut se poate folosi un algoritm bazat pe principiul competiţiei (vezi fig. 3 în care numărul claselor este iniţializat cu o valoare mică şi ulterior este mărit dacă este cazul. De exemplu, dacă un vector de intrare din setul de antrenare nu face parte din câmpul receptiv al nici unei unităţi ascunse atunci se adaugă o nouă unitate având centrul determinat chiar de vectorul de intrare.

Valoarea parametrului δ influenţează modul de partiţionare a domeniului datelor de intrare. Valori mici ale lui δ favorizează partiţionarea domeniului de intrare în multe zone, pe când valori mai mari ale lui δ conduce la stabilirea a mai puţine centre. Valoarea η_t controlează mărimea ajustării parametrilor c_{kj}. Când devine suficient de mică, ajustarea este neglijabilă şi algoritmul poate fi stopat. Descreşterea ratei η_t trebuie să fie lentă pentru a nu provoca oprirea algoritmului inaintea ajustării suficiente a centrilor.

Iniţializări:
Se stabileşte numărul iniţial de centre, K.
Se iniţializează valorile centrelor cu elemente selectate aleator din setul de antrenare.
Se iniţializează indicatorul de iteraţie, $t = 1$ şi rata de învăţare, $\eta_1 \in (0, 1)$.
REPEAT
FOR $l = \overline{1, L}$ se efectuează
se determină $k^* \in \{1, \ldots, K\}$ cu $\|X^l - C^{k^*}\| = \min_k \|X^l - C^k\|$;
IF $\|X^l - C^{k^*}\| < \delta$ THEN $C^{k^*} = C^{k^*} + \eta_t(X^l - C^{k^*})$
ELSE se adaugă o nouă unitate ascunsă: $K = K + 1$ şi $C^{K+1} = X^l$
Incrementează indicatorul de iteraţie, $t = t + 1$
Determină nouă valoare η_t a ratei de învăţare, $\eta_t = \eta_0 t^{-\alpha}$, $\alpha \in (0, 1]$.
UNTIL $t > t_{max}$ sau $\eta_t < \epsilon$

Figura 3: Algoritm de determinare a centrilor bazat pe un proces de competiţie

3.2.2 Determinarea lărgimii câmpurilor receptive.

Valoarea parametrilor σ_k controlează modul în care este partiţionat domeniul datelor de intrare. Valori prea mici ale lui σ conduce la o acoperire incompletă a domeniului de intrare, în sensul că pot rămâne porţiuni ce nu aparţin nici unui câmp receptiv. Valori prea mari ale lui σ provoacă mai multe unităţi ascunse să producă valori mari astfel că semnalul produs de reţea va fi unul aproape constant.

O regulă, cu caracter euristic, de determinare a parametrilor σ, după ce în prealabil au fost determinate centrele, este următoarea:

$$\sigma_k = \gamma d(C^k, C^j), \qquad \text{unde } C^j \text{ este centrul cel mai apropiat de } C^k, \text{ iar } \gamma \in [0.5, 1].$$

O regulă similară constă în

$$\sigma_k = \frac{1}{2}(d(C^k, C^i) + d(C^k, C^j)), \qquad \text{unde } C^i \text{ şi } C^j \text{ sunt centrii cei mai apropiaţi de } C^k$$

3.2.3 Determinarea ponderilor conexiunilor către nivelul de ieşire.

O dată ce au fost determinate centrele şi lărgimile câmpurilor receptoare, pentru orice semnal de intrare X^l poate fi determinat semnalul produs de nivelul unităţilor ascunse, Z^l. Astfel se poate considera că trebuie determinate ponderile conexiunilor asociate unei reţele cu un singur nivel de unităţi (liniare) pornind de la setul de antrenare $\{(Z^l, d^L)\}_{l=\overline{1,L}}$. Acest lucru poate fi realizat cu un algoritm clasic de antrenare a unei reţele cu un singur nivel, de exemplu cu algoritmul Widrow-Hoff.

Un caz particular este acela când centrele au fost setate cu elementele setului de antrenare. În aceste condiţii vectorul ponderilor conexiunilor dintre unitatea ascunsă k şi nivelul de ieşire poate fi setat cu vectorul răspunsurilor dorite: d^k. O astfel de setare se poate face pentru reţeaua destinată aproximării funcţiei XOR.

Invăţare nesupervizată bazată pe procese de competiţie

Învăţarea nesupervizată este un proces prin care se extrag informaţii, modele, reguli din date fără a avea la dispoziţie exemple de asociere ci doar folosind relaţiile existente între datele de intrare. Spre deosebire de învăţarea supervizată, cea nesupervizată permite identificarea unor structuri în datele de intrare fără a utiliza informaţii suplimentare.

Reţelele neuronale cu învăţare nesupervizată au aplicaţii în clasificarea, compresia şi analiza datelor.

1 Problematica grupării

Problema grupării datelor constă în identificarea unor clase (clustere) în cadrul datelor astfel încât elementele unei clase să fie suficient de similare iar cele ce aparţin unor clase diferite să fie suficient de disimilare. Fiecare clasă poate fi identificată printr-un element reprezentativ numit prototip al clasei. Un exemplu simplu de grupare în clase reprezentate prin prototipuri este ilustrat în Figura 1.

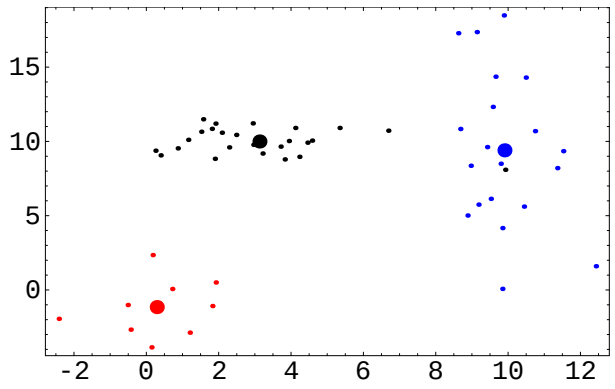


Figura 1: Clustere si prototipuri

Măsuri de similaritate. Pentru a aprecia similaritatea sau disimilaritatea intre date se folosesc măsuri specifice. În cazul în care datele sunt reprezentate prin vectori din R^N că măsură a disimilarităţii poate fi folosită distanţa euclidiană: $d(X,Y) = \sqrt{\sum_{i=1}^{N}(x_i - y_i)^2}$ (cu cât distanţa este mai mare cu atât este mai mare disimilaritatea). Dacă vectorii corespunzători datelor sunt normalizaţi ($\|X\| = \|Y\| = 1$) atunci o măsură a similarităţii poate fi considerat produsul scalar $X^T Y = \sum_{i=1}^{N} x_i y_i$ (cu cât produsul scalar este mai mare cu atât similaritatea este mai mare).

Legătura dintre distanţa euclidiană şi produsul scalar rezultă din relaţia:

$$d^2(X,Y) = \|X\|^2 - 2X^TY + \|Y\|^2 = 2(1 - X^TY)$$

Întrucât $X^TY = \|X\|\|Y\|\cos\alpha$, α fiind unghiul dintre X şi Y rezultă că în cazul datelor normate o altă măsură a similarităţii este cosinusul unghiului dintre cei doi vectori.

Prototipuri. Fiecare clasă poate fi identificată printr-un element reprezentativ numit prototip al clasei. Un astfel de prototip poate fi centroidul clasei (media aritmetică a elementelor clasei) sau medoidul acesteia (elementul din clasă care este cel mai apropiat de media aritmetică a elementelor clasei).

O dată stabilite prototipurile afectarea unui vector la o clasă se realizează pe baza criteriului distanţei minime (vectorul este asociat clasei faţă de prototipul căreia se află la distanţa minimă).

Aplicaţii. Identificarea unor clustere în cadrul datelor are aplicaţii în: identificarea unor categorii de clienţi ai unui sistem de comerţ electronic; identificarea unor profiluri de utilizatori ai unor resurse Web; organizarea unor documente electronice etc. În toate aceste cazuri clasificarea se face pe baza valorilor unor atribute asociate entităţilor ce trebuie clasificate.

O dată identificate clusterele, sistemul poate fi utiliza pentru a efectua predicţii sau recomandări. De exemplu în cazul unui sistem de comerţ electronic la apariţia unui nou client pe baza informaţiilor colectate şi pe baza clusterelor existente se pot face predicţii privind preferinţele acestuia astfel că i se pot face recomandări privind produsele în care este cel mai probabil interesat.

2 Reţele neuronale pentru clustering

Arhitectura. Pentru gruparea în K clase a unor date N-dimensionale arhitectura adecvată constă din:

- un nivel de N unităţi de intrare;
- un nivel cu K unităţi liniare de ieşire.

În cazul în care numărul de clase nu este cunoscut numărul unităţilor de pe nivelul de ieşire este adaptabil, fiind modificat în procesul de învăţare. Conectivitatea dintre cele două nivele este totală iar ponderile conexiunilor pot fi organizate într-o matrice W cu K linii şi N coloane. Linia i a matricii W poate fi interpretată că fiind prototipul clasei i. După antrenare, setul de vectori $\{W^1, \ldots, W^K\}$ reprezintă prototipurile celor K clase.

Funcţionare. În timpul funcţionării reţeaua trebuie să returneze indicele sau prototipul clasei asociate datei de intrare. Clasa asociată datei de intrare este determinată de unitatea de ieşire învingătoare. Unitatea învingătoare este determinată printr-un proces de competiţie care presupune calculul măsurii de similaritate dintre data de intrare, X, şi fiecare dintre vectorii prototip $\{W^1, \ldots, W^K\}$ şi determinarea unităţii ce are asociat prototipul cel mai apropiat de data de intrare. În cazul în care măsura de similaritate este distanţa euclidiană determinarea unităţii învingătoare presupune:

- calcul d_1, d_2, ..., d_K (cu $d_i = d(X, W^i)$);
- determinare $k^* \in \{1, \ldots, K\}$ cu proprietatea că $d_{k^*} \leq d_i$ pentru orice $i \in \{1, \ldots, K\}$.

Indicele k^* al unităţii învingătoare poate fi interpretat ca indice al clasei căreia îi aparţine data de intrare. Dacă există mai multe prototipuri care se află la aceeaşi distanţă faţă de X atunci trebuie să se decidă care este unitatea învingătoare (de exemplu se alege unitatea cu indicele cel mai mic).

Etapa 1. Iniţializarea prototipurilor.
$w_{ij} = \text{rand}(-1, 1), \quad i = \overline{1, K}, \quad j = \overline{1, N}$
$t = 1$
Etapa 2. Proces iterativ de ajustare a prototipurilor.
REPEAT FOR $l = \overline{1, L}$ DO
Determină k^* cu proprietatea $d(X^l, W^{k^*}) \leq d(X^l, W^i)$, $i = \overline{1, K}$
$W^{k^*} = W^{k^*} + \eta(t)(X^l - W^{k^*})$
$t = t + 1$
UNTIL $(\eta(pt) < \epsilon)$ OR $(t > t_{max})$.

Figura 2: Algoritm simplu de tip WTA

Dacă într-o implementare software determinarea unităţii învingătoare este o prelucrare simplă în implementări hard sau în modelele biologice ea este mai complicată, bazându-se pe existenţa unor conexiuni laterale inhibitorii al căror rol este să accentueze diferenţa dintre semnalul produs de unitatea învingătoare şi cele produse de către celelalte unităţi. În felul acesta semnalul produs de către unitatea învingătoare devine semnificativ mai mare astfel că detecţia acesteia devine mai simplă.

Antrenare. Există două variante principale de antrenare:

- Numărul de unităţi funcţionale (K) este prestabilit nemodificându-se pe parcursul antrenării. Un exemplu de algoritm de învăţare din această categorie este cel bazat pe principiul "invingătorul ia totul" (WTA - the winner takes all).
- Numărul unităţilor funcţionale este adaptat pe parcursul procesului de învăţare. Algoritmi din această categorie se bazează pe "teoria rezonanţei adaptive" (ART - adaptive resonance theory).

În continuare este prezentat câte un exemplu de algoritm de învăţare din fiecare categorie. În ambele situaţii considerăm că setul de antrenare este constituit din vectori N-dimensionali, $\{X^1, X^2, \ldots, X^L\}$.

Algoritm simplu de tip WTA. Presupunem că numărul de unităţi funcţionale este K. Un algoritm simplu bazat pe competiţie este ilustrat in fig. 2.

Ideea de bază a ajustării este de a modifica doar vectorul ponderii unităţii învingătoare astfel încât el să devină mai apropiat de data de intrare. Pentru a evita oscilaţia permanentă a ponderilor şi a stabiliza prototipurile se utilizează o rată descrescătoare de învăţare, $\eta(p)$. Valorile lui η trebuie să descrească ($\eta(p+1) < \eta(p)$) însă nu prea rapid pentru a permite ajustarea suficientă a ponderilor. Se consideră că şirul valorilor $\eta(p)$ trebuie să satisfacă următoarele proprietăţi:

$$\lim_{t\to\infty} \eta(t) = 0, \qquad \sum_{t=1}^{\infty} \eta(t) = \infty, \qquad \sum_{t=1}^{\infty} \eta(t^2) < \infty \tag{1}$$

Un exemplu de rată de învăţare care satisface proprietăţile (1) sunt:

$$\eta(t) = \frac{\eta(0)}{t^\alpha}, \quad \alpha \in (0.5, 1], \quad \eta(0) > 0 \tag{2}$$

Chiar dacă nu satisface toate proprietăţile din (1) se mai foloseşte şi $\eta(t) = \eta(0)/\ln(t+1)$.

În practică se obişnuieşte să se normalizeze atât datele de intrare ($\|X^l\| = 1$) cât şi vectorii de ponderi. Pentru a asigura faptul că vectorii de ponderi sunt normalizaţi ajustarea acestora se realizează prin

$$W^{k^*} = \frac{W^{k^*} + \eta(t)(X^l - W^{k^*})}{\|W^{k^*} + \eta(t)(X^l - W^{k^*})\|}.$$

Unul dintre dezavantajele algoritmilor de tip WTA îl reprezintă faptul că este posibil ca anumite unităţi să nu devină niciodată învingătoare. Astfel de unităţi sunt numite "moarte". Pentru a evita apariţia unităţilor moarte se ajustează nu numai ponderile unităţii învingătoare ci şi cele ale celorlalte unităţi însă cu o rată de învăţare mai mică (de exemplu de circa 10 ori mai mică). O altă modalitate de a evita apariţia unităţilor moarte este aşa numitul mecanism al conştiinţei. Acesta se caracterizează prin a lua în considerare la determinarea unităţii învingătoare şi a unui prag care variază invers proporţional cu frecvenţa victoriilor. Astfel se decide că unitatea k^* este învingătoare (pentru intrarea X) dacă:

$$d(X, W^{k^*}) - \theta_{k^*} \leq d(X, W^k) - \theta_k, \qquad \text{pentru } k \in \{1, \ldots, K\},$$

iar cu cât numărul de victorii este mai mare cu atât valoarea pragului descreşte reducând şansa unităţii de a deveni prea frecvent învingătoare.

Algoritm de tip ART. Un alt dezavantaj al algoritmului WTA este faptul că numărul de clase este prestabilit. Există situaţii în care nu se cunoaşte numărul de clase, sarcina de a determina acest număr revenindu-i tot algoritmului de învăţare. În dezvoltarea unui sistem adaptiv (reţea neuronală) de rezolvare a problemelor de grupare intervin două aspecte contradictorii:

- *Adaptabilitate:* capacitatea sistemului de a asimila noi informaţii şi de a identifica noi clustere.
- *Stabilitate:* capacitatea sistemului de a conserva structura de clustere detectată la un moment dat, astfel ca pe parcursul procesului de adaptare sistemul să nu-şi schimbe radical răspunsul pentru o aceeaşi dată de intrare.

Pornind de la această dilemă adaptabilitate-stabilitate a fost dezvoltată de către Carpenter şi Grossberg teoria cunoscută sub numele de teoria rezonanţei adaptive şi au fost dezvoltate modele sofisticate de reţele neuronale care implementează diverse mecanisme cu suport biologic. În continuare este prezentată o variantă simplificată de algoritm de tip ART specific grupării datelor din R^N. Reţeaua neuronală suport este constituită dintr-un nivel de N unităţi de intrare şi un nivel de unităţi funcţională a căror număr poate creşte pe parcursul procesului de învăţare.

Ideea de baza a algoritmului este aceea de a verifica gradul de similaritate între data de intrare şi vectorul de ponderi asociat unităţii învingătoare iar în cazul în care acest grad nu depăşeşte un prag de vigilenţă se consideră că nici o unitate dintre cele existente nu este suficient de reprezentativă pentru data de intrare, motiv pentru care se adaugă o nouă unitate. Structura algoritmului este descrisă în fig. 3.

Parametrul $\rho > 0$ este pragul de vigilenţă şi cu cât se alege mai mic cu atât vor fi create mai multe clustere. Este corelat cu "raza" maximă a clusterelor. În calculul lui card(ω_k) elementele din setul de antrenare sunt afectate la ω_k pe baza distanţei minime.

Algoritmul ART prezentat este o variantă simplificată a algoritmului ART2 propus de Carpenter şi Grossberg şi diferă de algoritmul utilizat la reţelele RBF pentru determinarea centrelor doar prin forma relaţiei de ajustare.

Unul dintre principalele dezavantaje ale acestui algoritm simplu este acela că gruparea pe care o identifică depinde de ordinea în care sunt prezentate datele de intrare.

Iniţializări:
Se stabileşte numărul iniţial de unităţi funcţionale, $K < L$.
Se iniţializează vectorii ponderilor cu elemente distincte selectate aleator din setul de antrenare.
$t = 1$
REPEAT
FOR $l = \overline{1, L}$ se efectuează
se determină $k^* \in \{1, \ldots, K\}$ cu $d(X^l, W^{k^*}) = \min_k d(X^l, W^k)$;
IF $d(X^l, W^{k^*}) < \rho$ OR $K = K_{max}$ THEN $W^{k^*} = (X^l + W^{k^*}\text{card}(\omega_{k^*}))/(1 + \text{card}(\omega_{k^*}))$
($\text{card}(\omega_{k^*})$ reprezintă numărul de elemente din clasa k^*)
ELSE se adaugă o nouă unitate ascunsă: $K = K + 1$ şi $W^{K+1} = X^l$
Incrementează indicatorul de iteraţie, $t = t + 1$
UNTIL $t > t_{max}$

Figura 3: Algoritm de tip ART pentru grupare de vectori din R^N

3 Cuantizare vectorială

Rolul cuantizării (discretizării) vectoriale este de a mapa un domeniu continuu de date vectoriale (din R^N) într-un domeniu discret (o mulţime finită de vectori numiţi prototipuri sau "codebook vectors" cărora li se pot asocia indicatori scalari). Prin cuantizare se asigură partiţionarea domeniului datelor de intrare în regiuni, fiecărei regiuni corespunzându-i un prototip sau un număr de ordine. Astfel fiecare vector din domeniul datelor de intrare poate fi asociat cu un scalar (numărul de ordine al prototipului corespunzător). În felul acesta se poate asigura o compresie a datelor de intrare. Printr-o astfel de compresie se pierde din informaţia purtată de datele iniţiale iar scopul urmărit este să se minimizeze această pierdere.

În acest scop, discretizarea trebuie să conserve proprietăţile distribuţiei de probabilitate asociată sursei de informaţie ce generează datele de intrare (plasarea prototipurilor reprezintă o aproximare a distribuţiei datelor de intrare). De exemplu în zonele în care densitatea datelor de intrare este mare se impune o partiţionare mai fină, deci existenţa mai multor prototipuri.

Asocierea prototipului la o dată de intrare se bazează pe criteriul distanţei minime.

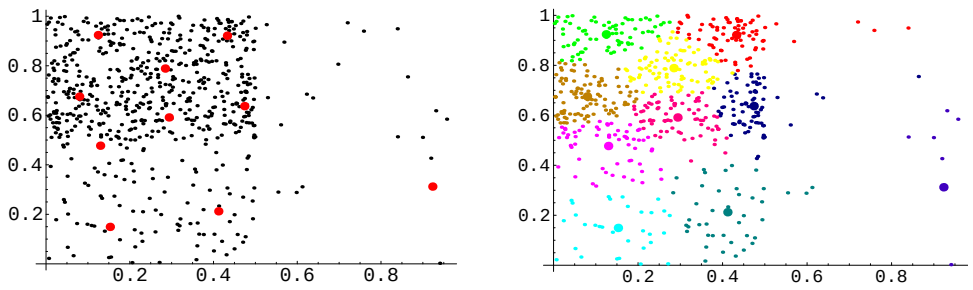


Figura 4: Cuantizare vectoriala folosind algoritmul WTA (100 epoci, $\eta(0) = 0.1$, $\eta(p) = \eta(0)/p$, $K = 10$)(a) Datele de intrare si pozitiile prototipurilor obtinute; (b) Separarea în clase

Dacă numărul de clase este cunoscut pentru identificarea prototipurilor se poate folosi un algoritm simplu de tip WTA. Un exemplu de cuantizare pentru vectori bidimensionali este ilustrat in fig. 4. Într-o astfel de variantă unităţile sunt afectate claselor într-o manieră relativ arbitrară care depinde de modul de iniţializare a ponderilor şi de ordinea de prezentare a datelor de intrare.

Dacă se doreşte utilizarea unei astfel de reţele pentru rezolvarea unei probleme de clasificare şi dacă se cunosc exemple de clasificare corectă atunci se poate utiliza o varianta supervizată a algoritmului cunoscută sub denumirea de LVQ - Learning Vector Quantization. De regulă numărul claselor în care se realizează clasoficarea este mai mic decât numărul clusterelor detectate în cadrul datelor de intrare. În acest caz setul de antrenare este de forma: $\{(X^1, d_1), \ldots, (X^L, d_L)\}$ (X^l sunt vectori N-dimensionali iar d_l este indicator de clasa) iar antrenarea constă în parcurgerea etapelor următoare:

- Se stabilesc K vectori prototip folosind un algoritm de tip WTA pornind de la datele de intrare $\{X^1, \ldots, X^L\}$ din setul de antrenare. Această etapă are ca scop modelarea distribuţiei de probabilitate a datelor de intrare.
- Pentru fiecare prototip se stabileşte indicatorul clasei din care face parte: (i) se afectează (folosind criteriul distanţei minime) vectorii X^l din setul de antrenare la fiecare dintre clusterele asociate prototipurilor; (ii) în cadrul fiecărui cluster se determină indicatorul ce apare cel mai frecvent; acesta va fi indicatorul asociat prototipului respectiv.
- Se ajusteaza iterativ (de-a lungul mai multor epoci $p = \overline{1, p_{max}}$ prototipurile clusterilor folosind setul de antrenare si regulile: (i) pentru X^l se determină prototipul cel mai apropiat (considerăm că acesta este W^{k^*} iar indicatorul asociat acestuia este c_l); dacă $c_l = d_l$ atunci se face ajustarea $W^{k^*} = W^{k^*} + \eta(t)(X^l - W^{k^*})$ iar dacă $c_l \neq d_l$ atunci se face ajustarea $W^{k^*} = W^{k^*} - \eta(t)(X^l - W^{k^*})$. Scopul acestei etape este de a mări distanţa dintre prototipuri şi suprafeţele de separare a diferitelor clase.

4 Mapare topologică şi reţele Kohonen

Mapare topologică. Maparea topologică este o variantă de cuantizare vectorială prin care se asigură conservarea relaţiilor de vecinătate din domeniul datelor de intrare. Acest lucru impune existenţa unei organizări geometrice a nivelului unităţilor funcţionale astfel încât pentru fiecare unitate funcţională să poată fi definită o vecinătate. Astfel poziţia fiecărei unităţi funcţionale trebuie identificată cu un vector de poziţie (dacă nivelul are o structură n-dimensională atunci vectorul de poziţie al unităţii p este $r_p = (r_p^1, \ldots r_p^n)$). Întrucât prin maparea topologică se urmăreşte reducerea dimensiunii datelor de intrare se foloseşte $n \leq 3$: nivel unidimensional ($n = 1$), bidimensional ($n = 2$) sau tridimensional ($n = 3$). Reţelele cu astfel de organizare a nivelului de unităţi funcţionale sunt cunoscute sub numele de reţele Kohonen sau hărţi cu autoorganizare (SOM - self-organizing maps) şi au fost propuse de către Kohonen la inceputul anilor 1980. Reţelele din această categorie au un pronunţat suport biologic întrucât în creier există hărţi corticale care pot fi interpretate ca mapări ale spaţiului stimulilor (hărţi topografice corespunzătoare stimulilor vizuali, hărţi tonotopice corespunzătoare stimulilor auditivi, hărţi senzoriale asociate cu simţul tactil).

Arhitectura. O reţea Kohonen este constituită dintr-un nivel de intrare şi un nivel de unităţi funcţionale structurat geometric (vezi fig. 5 pentru cazul unui nivel funcţional bidimensional). În cazul unui nivel unidimensional unităţile pot fi plasate în nodurile unei diviziuni (echidistante) pe când în cazul bidimensional ele pot fi plasate în nodurile unei grile pătratice (fig. 6(a)), hexagonale (fig. 6(b)) sau chiar neregulată.

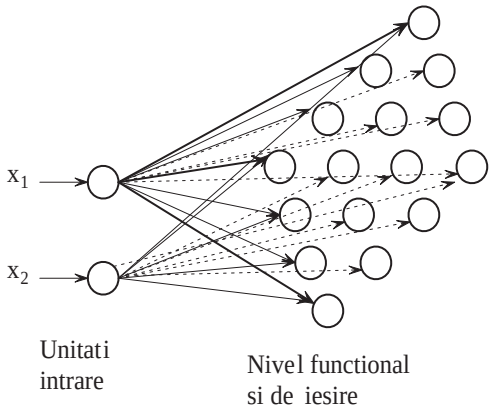


Figura 5: Exemplu de reţea cu organizare geometrică a nivelului de ieşire (reţea Kohonen)

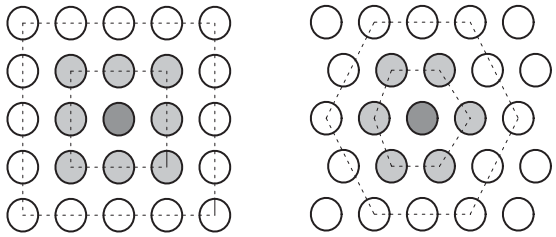


Figura 6: Tipuri de grile bidimensionale utilizate in algoritmii de învăţare de la reţelele Kohonen

Vecinătăţi. O modalitate simplă de a defini un sistem de vecinătăţi în cadrul nivelului de ieşire se bazează pe utilizarea unei distanţe în domeniul $\mathcal{L}$ al vectorilor de poziţie (în cazul unidimensional $\mathcal{L} = \{1, \ldots, M\}$, iar în cazul bidimensional $\mathcal{L} = \{1, \ldots, m\} \times \{1, \ldots, m\}$, $M = m^2$). Câteva dintre distanţele ce pot fi folosite sunt:

- $d_1(r_p, r_q) = \sqrt{\sum_{i=1}^{n} (r_p^i - r_q^i)^2}$ (distanţa euclidiană);
- $d_2(r_p, r_q) = \sum_{i=1}^{n} |r_p^i - r_q^i|$ (distanţa Manhattan);
- $d_3(r_p, r_q) = \max_{i=\overline{1,n}} |r_p^i - r_q^i|$.

În relaţiile de mai sus, $r_p \in \mathcal{L}$ şi $r_q \in \mathcal{L}$ reprezintă vectorii de poziţie ai unităţilor p respectiv q.

Utilizând o astfel de distanţă pentru o unitate p o vecinătate de ordin s va fi constituită din $V_s(p) = \{q | d(r_p, r_q) \leq s\}$. De exemplu în cazul unei grile bidimensionale pătratice vecinătăţile de ordin 1 ale unităţii p ($r_p = (i, j)$) corespunzătoare celor trei distanţe sunt:

$$V_1^{(1)}(i,j) = V_1^{(2)}(i,j) = \{(i-1,j), (i+1,j), (i,j-1), (i,j+1)\}$$

$$V_2^{(3)}(i,j) = \{(i-1,j), (i+1,j), (i,j-1), (i,j+1), (i-1,j-1), (i-1,j+1), (i+1,j-1), (i+1,j+1)\}.$$

Etapa 1. Iniţializări.
$w_{ij} = \text{rand}(-1, 1), \quad i = \overline{1, K}, \quad j = \overline{1, N}$
$t = 1$, $s(t) = s_1$, $\eta(t) = \eta_1$
Etapa 2. Proces iterativ de ajustare.
REPEAT Pentru data de intrare X
Determină p^* cu proprietatea $d(X, W^{p^*}) \leq d(X, W^p)$, $p \in \mathcal{L}$
$W^p = W^p + \eta(t)(X - W^p)$ pentru orice $p \in V_{s(t)}(p^*)$
$t = t + 1$
UNTIL $(\eta(t) < \epsilon)$ OR $(t > t_{max})$.

Figura 7: Algoritm de antrenare pentru o reţea Kohonen

Algoritm de învăţare. Specificul învăţării în reţelele Kohonen este faptul că nu se ajustează doar ponderile unităţii învingătoare ci şi cele corespunzătoare unităţilor din vecinătatea acesteia. Ajustarea acestor ponderi permite "ordonarea" unităţilor astfel încât în faza de funcţionare pentru date de intrare similare să reacţioneze unităţi funcţionale învecinate. Considerând că reţeaua primeşte succesiv date de intrare varianta on-line a algoritmului de învăţare este descrisă în fig. 7.

Atât dimensiunea vecinătăţii cât şi rata de învăţare descresc treptat de la o iteraţie la alta. Prin utilizarea unei dimensiuni variabile a vecinătăţii se pun în evidenţă două faze ale procesului de învăţare:

- *Faza de ordonare.* Corespunde iteraţiilor în care dimensiunea vecinătăţii este relativ mare şi are ca efect aranjarea unităţilor (prin ajustarea ponderilor asociate) astfel încât unor date de intrare similare să le corespundă unităţi apropiate.
- *Faza de ajustare fină.* Corespunde ultimelor iteraţii, în care vecinătăţile sunt reduse (uneori chiar la un singur element, algoritmul fiind în acest caz de tip WTA) şi are ca scop ajustarea ponderilor astfel încât acestea să fie cât mai reprezentative pentru datele de intrare.

Iniţial raza vecinătăţii se alege suficient de mare, astfel incât ajustarea să se extindă asupra tuturor unităţilor. Ulterior raza se micşorează. O modalitate simplă de reducere a vecinătăţii este înjumătăţirea razei după un anumit număr de iteraţii.

Modificarea ratei de învăţare asigură stabilizarea valorii ponderilor. Ca reguli de descreştere se pot folosi cele de la algoritmul WTA. Pentru a diferenţia ajustarea aplicată unităţii învingătoare de cea aplicată unităţilor din vecinătate se poate modifica relaţia de ajustare prin introducerea unei funcţii de vecinătate $\Lambda(p, q)$ a cărei valoare maximă este atinsă pentru unitătea învingătoare. Relaţia de ajustare devine în acest caz:

$$W^p = W^p + \eta(t)\Lambda(p^*, p)(X - W^p)$$

. Dacă funcţia de vecinătate verifică:

$$\Lambda(p, q) = \begin{cases} 1 & \text{dacă } q \in V_s(p) \\ 0 & \text{altfel} \end{cases}$$

atunci se reobţine relaţia iniţială de ajustare. O altă variantă prin care mărimea ajustării depinde de distanţa faţă de unitatea învingătoare este:

$$\Lambda(p, q) = \exp\left(-\frac{d^2(r_p, r_q)}{2s^2}\right)$$

Graficele acestor funcţii sunt reprezentate în fig. 8

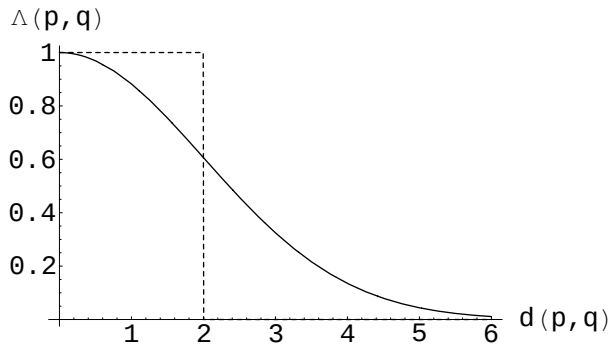


Figura 8: Functii de vecinatate

La fel ca şi în cazul algoritmilor de tip WTA normalizarea datelor de intrare şi a vectorilor cu ponderi după fiecare ajustare poate fi benefică procesului de învătare.

Modul în care o reţea Kohonen se autoorganizează astfel încât se mapeze cât mai fidel domeniul datelor de intrare poate fi ilustrat grafic pentru date de intrare bidimensionale ($N = 2$). În acest caz fiecărei unităţi funcţionale îi corespunde un vector cu două ponderi. Vizualizarea constă în reprezentarea pentru fiecare unitate funcţională a punctului corespunzător vectorului cu ponderi şi în unirea punctelor corespunzătoare unităţilor vecine (în cazul unidimensional punctul asociat unităţii i se uneşte cu cele asociate unităţilor $i-1$ şi $i+1$ iar în cazul bidimensional unitatea (i, j) se uneşte cu unităţile $(i-1, j)$, $(i+1, j)$, $(i, j-1)$, $(i, j+1)$. Exemple pentru cazul unidimensional şi cel bidimensional sunt prezentate în fig 9 (pentru cazul în care datele sunt generate aleator în interiorul unui cerc) şi în fig. 10 (pentru cazul coroanei circulare).

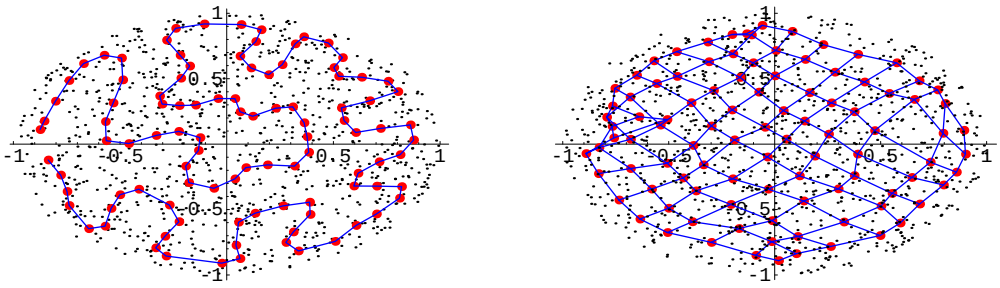


Figura 9: Maparea datelor generate aleator în interiorul unui cerc (a) reţea unidimensională; (b) reţea bidimensională

Aplicaţii ale reţelelor Kohonen. Reţelele Kohonen au fost aplicate cu succes în recunoaşterea vorbirii, controlul braţelor de robot, clasificarea site-urilor Web (WebSOM), rezolvarea problemelor de optimizare combinatorială.

Controlul braţelor de robot. Fiecărei articulaţii a unui braţ de robot îi corespunde un set de

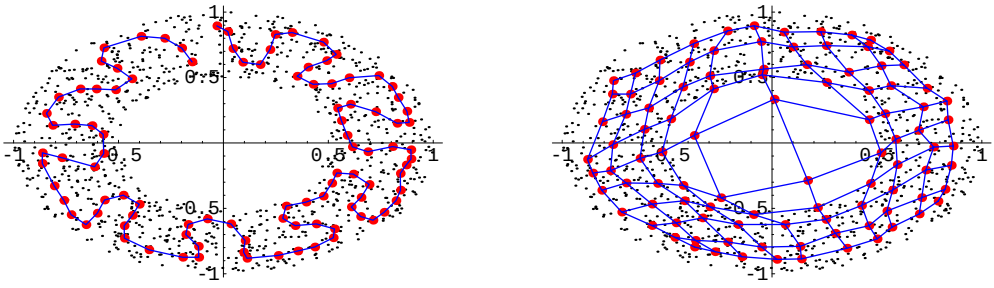


Figura 10: Maparea datelor generate aleator într-o coroană circulară (a) reţea unidimensională; (b) reţea bidimensională

unghiuri a căror valoare determină poziţia extremităţii efectoare a braţului. O reţea Kohonen tridimensională poate fi antrenată cu date de intrare reprezentând seturi de valori ale unghiurilor corespunzătoare mişcării braţului. Reţeaua va realiza o mapare a poziţiilor extremităţii efectoare. Această mapare permite evitarea obstacolelor (zonele cărora nu le corespund date de intrare sunt probabil ocupate de obstacole astfel că aceste zone nu sunt mapate). În practică reţeaua va fi iniţializată cu o mapare ce ţine cont de poziţia obstacolelor procesului de invăţare revenindu-i rolul de a realiza ajustări ulterioare impuse de modificarea condiţiilor de lucru sau de degradarea senzorilor.

Rezolvarea problemei comis-voiajorului. Problema comis-voiajorului (TSP - Travelling Salesman Problem) este o problemă clasică de optimizare care constă în găsirea traseului de cost minim pe care trebuie să îl parcurgă un comis voiajor astfel încât să treacă o singură dată prin fiecare oraş dintr-un set prestabilit. Dacă se consideră că oraşele pot fi reprezentate prin puncte în plan iar costul trecerii de la un oraş o la un oraş o' este proporţional cu distanţa $d(o, o')$ dintre cele două oraşe problema poate fi rezolvată aproximativ printr-o reţea de tip Kohonen unidimensională cu unităţi plasate circular (în cazul a m unităţi se consideră că unitatea 1 si unitatea m sunt vecine. O astfel de reţea este denumită şi reţea elastică. Ideea rezolvării este de a furniza reţelei, ca date de intrare, coordonatele oraşelor. Procesul de adaptare conduce la apropierea unor unităţi de punctele corespunzătoare oraşelor şi cum unităţile formează un inel acesta va tinde către un traseu. Cum mai multe unităţi pot fi atrase către acelaşi punct numărul de unităţi trebuie să fie mai mare decât numărul de oraşe. Traseul obţinut nu este neapărat cel optim însă unul suficient de bun. Astfel tehnica de optimizare cu reţele neuronale nu este una exactă ci una aproximativă.

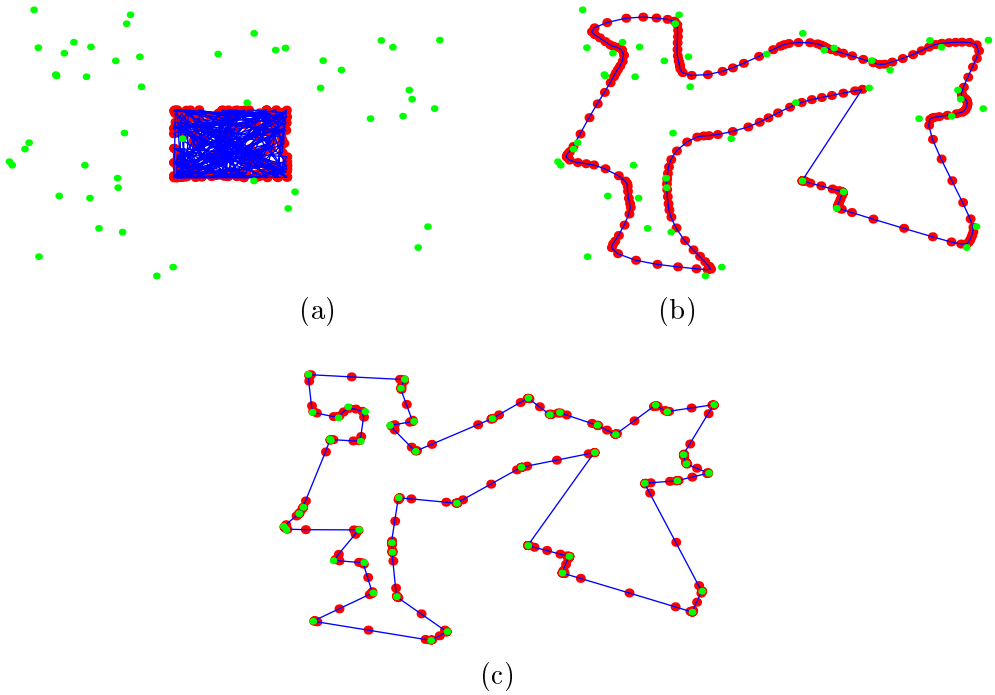


Figura 11: Reţea elastică cu 200 de unităţi (roşu) pentru rezolvarea problemei comis-voiajorului (50 oraşe (verde)). (a) configuraţia iniţială; (b) după 1000 iteraţii; (c) dupa 2000 iteraţii

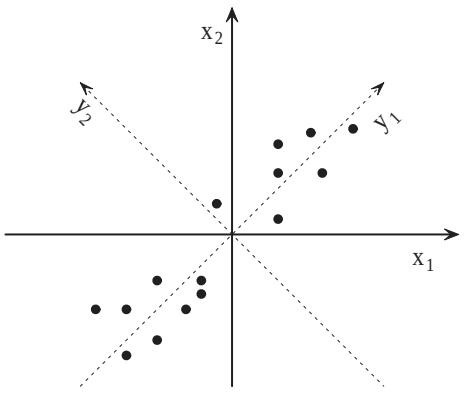
Reţele neuronale pentru analiza în componente principale

1 Analiza în componente principale

Analiza în componente principale (ACP) este o prelucrare statistică a datelor reprezentate vectorial având drept scop principal reducerea dimensiunii acestora cu păstrarea a cât mai mult din informaţia pe care o poartă. Această tehnică este utilă în transformarea datelor înaintea efectuării unor prelucrări de genul clasificării sau grupării precum şi în prelucrarea semnalelor (de exemplu pentru compresia imaginilor).

Din punct de vedere intuitiv ACP constă în transformarea unui ansamblu de informaţii reprezentate prin vectori N dimensionali într-un ansamblu de vectori M dimensionali astfel încât noile componente ale vectorilor să exprime cât mai bine variabilitatea datelor analizate. În fig. 1 este ilustrat cazul în care $N = 2$. Iniţial vectorii sunt reprezentaţi în sistemul de axe de coordonate x_1Ox_2. Se observă însă că prin înlocuirea acestui sistem cu un altul, y_1Oy_2, se poate surprinde structura setului de date reţinând doar componenta corespunzătoare noii axe Oy_1 (numită componentă principală). Axa Oy_1 este direcţia de-a lungul căreia datele variază cel mai mult astfel că dacaă se urmăreşte să se realizeze o clasificare a datelor atunci ar fi suficient să se reţina doar componenta corespunzătoare ei.

În practică, N este mare (de exemplu, datele analizate pot fi un ansamblu de caracteristici ale unui activ financiar) şi se impune reducerea dimensiunii vectorilor pentru a fi mai uşor de prelucrat. Determinarea componentelor principale permite eliminarea redundanţei sau a zgomotului din cadrul datelor permiţând ca informaţia relevantă să poată fi reprezentată prin vectori de dimensiune mai mică.



Ilustrarea componentelor principale

Din punct de vedere formal problema poate fi descrisă în modul următor. Se consideră că informaţia de prelucrat poate fi modelată printr-un vector aleator, $X \in I\!R^N$, caracterizat de o anumită repartiţie. Pentru simplitate vom considera în continuare că vectorul aleator este de medie nulă ($E(X) = 0$). Se pune problema găsirii unei transformări, $T : I\!R^N \to I\!R^M$, (cu $M < N$) şi a inversei sale, $S : I\!R^M \to I\!R^N$, astfel încât vectorul reconstruit, $\hat{X} = S(T(X))$, să fie cât mai apropiat de vectorul iniţial X. Vectorul, Y, obţinut prin transformarea prin T a lui X poate fi considerat o reprezentare condensată a acestuia (ca urmare a faptului că dimensiunea a fost redusă).

În cazul clasic transformarea T căutată este liniară: $T(X) = WX$, W fiind o matrice cu M linii şi N coloane. În statistică se arată că matricea W care asigură cea mai mică eroare la reconstruire (pentru $S(Y) = W^T Y$) se obţine reţinând vectorii proprii ai matricii de corelaţie,$C = M(XX^T)$, corespunzători celor mai mari M valori proprii. Întrucât matricea de corelaţie este simetrică şi semipozitiv definită toate valorile sale proprii vor fi reale şi pozitive. Dacă valorile proprii sunt ordonate descrescător: $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N$ atunci linia W^i a matricii va fi vectorul propriu corespunzător lui λ_i. Vectorul Y nu este altceva decât proiecţia lui X pe spaţiul determinat de cei M vectori proprii selectaţi.

Transformarea obţinută în acest mod este cunsocută şi sub numele de transformarea Karhunen-Loeve şi este utilizată în prelucrarea semnalelor.

Construirea transformării presupune:

- Determinarea matricii de corelaţie (cum media vectorului aleator este nulă ea coincide cu matricea de covarianţă). În cazul în care distribuţia vectorului aleator este cunoscută, matricea de corelaţie poate fi determinată exact iar în caz contrar se estimează pornind de la vectori obţinuţi prin selecţie.
- Determinarea valorilor proprii şi a vectorilor proprii corespunzători. Această prelucrarea se poate realiza utilizând metode numerice adecvate.

În cazul în care se urmăreşte determinarea on-line a matricii W prin ajustarea ei pentru fiecare realizare a vectorului aleator sau în cazul în care N este mare o metodă alternativă o reprezintă utilizarea unui algoritm iterativ de învăţare nesupervizată aplicat unei reţele neuronale cu arhitectură simplă.

2 Arhitectura suport

Reţelele destinate rezolvării problemelor de determinare a componentelor principale au o arhitectură foarte simplă:

- N unităţi de intrare;
- M unităţi funcţionale, total conectate cu unităţile de intrare, având funcţii liniare de transfer.

În aceste condiţii pentru un semnal de intrare $X = (x_1, x_2, \ldots, x_N)$ se obţine semnalul de ieşire $Y = (y_1, y_2, \ldots, y_M)$ caracterizat prin:

$$Y = W \cdot X \text{ adică } y_i = \sum_{j=1}^{N} w_{ij} x_j, \qquad i = \overline{1, M}.$$

Semnalul "reconstruit" pornind de la Y, $\hat{X} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_N)$ are componentele:

$$\hat{X} = W^T \cdot Y \text{ adică } \hat{x}_j = \sum_{i=1}^{M} w_{ij} y_i, \qquad j = \overline{1, N}.$$

Dacă obţinerea semnalului codificat se bazează pe "trecerea" directă prin reţea obţinerea semnalului reconstruit poate fi interpretată ca trecerea prin reţea în sens invers a semnalului codificat.

3 Algoritmi de învăţare

Unul dintre cei mai simpli algoritmi de învăţare este cel propus de Oja. Dezvoltat iniţial pornind de la o regula simplă de învăţare corelativă (regula Hebb, conform căreia ajustarea ponderii w_{ij} trebuie să fie proporţională cu produsul $x_j y_i$) el poate fi văzut şi ca un algoritm de tip Widrow-Hoff aplicat cu scopul de a minimiza eroarea dintre semnalul iniţial şi cel reconstruit (în acest caz setul de antrenare poate fi văzut ca fiind $\{(X^1, X^1), \ldots, (X^L, X^L)\}$ iar invăţarea poate fi interpretată ca fiind *autosupervizată*). Regula de ajustare a ponderilor este:

$$w_{ij} = w_{ij} + \eta y_i (x_j - \hat{x}_j), \qquad i = \overline{1, M}, j = \overline{1, N}. \tag{1}$$

Se observă cu uşurinţă că relaţia 1 este similară celei de la învăţarea Widrow-Hoff cu diferenţa că eroarea este calculată la nivelul de intrare şi ea este înmulţită cu semnalul de ieşire.

Se poate demonstra că prin ajustări succesive liniile matricii W tind către M primii vectori proprii ai matricii de corelaţie neasigurându-se însă o legătură directă între numărul de ordine al liniei şi cel al vectorului propriu (când valorile proprii sunt ordonate descrescător).

Această deficienţă este remediată de o variantă a algoritmului propusă de Sanger în care regula de ajustare este:

$$w_{ij} = w_{ij} + \eta y_i (x_j - \tilde{x}_j^i), \quad \tilde{x}_j^i = \sum_{k=1}^{i} w_{kj} y_k \qquad i = \overline{1, N}, j = \overline{1, M}. \tag{2}$$

Structura generală a algoritmului, atât în varianta Oja cât şi în varianta Sanger, este ilustrată în fig. 1.

Se iniţializează matricea W
$t = 0$ (iniţializarea indicatorului de iteraţie)
REPEAT
  ajustarea ponderilor pentru semnalul curent, X, folosind (1) sau (2)
  $t = t + 1$
  determinarea noii valori a ratei de învăţare
UNTIL *este îndeplinit un criteriu de oprire.*

Figura 1: Structura generală a algoritmilor Oja şi Sanger

Reţele recurente

Reţelele neuronale recurente se caracterizează prin prezenţa conexiunilor inverse, semnalul produs de către o unitate ajungând, fie direct fie indirect (prin intermediul altor unităţi), la unitatea de la care a pornit. Graful asociat unei reţele recurente conţine cicluri, iar acest lucru modifică atât modul de funcţionare al reţelei cât şi potenţialul său aplicativ. Din punct de vedere al gradului de conectivitate există:

- *Reţele total recurente.* Acestea se caracterizează prin conectivitate totală (fiecare unitate este conectată cu toate celelalte şi, uneori, chiar cu sine însăşi). Reţelele din această categorie sunt cunsocute şi sub denumirea de reţele Hopfield (sau modelul Hopfield) şi au fost dezvoltate cu scopul de a implementa *memorii asociative* şi de a rezolva probleme de optimizare.
- *Reţele parţial recurente.* Acestea se caracterizează prin faptul că o unitate este conectată doar cu o parte dintre celelalte unităţi însă, spre deosebire de reţelele feed-forward, există şi conexiuni inverse. Din această categorie fac parte *reţelele cu unităţi contextuale* utilizate în special în prelucrarea seriilor temporale şi *reţelele celulare* utilizate în special în prelucrarea imaginilor.

Existenţa conexiunilor inverse face ca reţelele recurente să se comporte ca nişte sisteme dinamice. Astfel semnalul de iesîre nu mai este obţinut prin calcul direct pornind de la semnalul de intrare ci este nevoie de un proces iterativ prin care se estimează starea staţionară către care tinde reţeaua.

Din punct de vedere al modului de funcţionare, reţelele parţial recurente pot fi văzute ca şi cazuri particulare ale reţelelor total recurente, astfel că este suficient ca principiul funcţionării să fie ilustrat pentru acestea din urmă.

1 Reţele total recurente. Modelul Hopfield.

Vom ilustra modul de funcţionare a unei reţele recurente pentru cazul modelului Hopfield.

Arhitectura. Modelul Hopfield este o reţea constituită din N unităţi total interconectate, în care fiecare unitate joacă simultan rol de unitate de intrare şi de ieşire (vezi fig. 1).

El poate fi utilizat atât pentru simularea *memoriilor asociative* (sisteme dinamice ce permit stocarea informaţiilor prin intermediul parametrilor lor şi regăsirea acestora pornind de la exemplare incomplete sau afectate de zgomot) cât şi pentru rezolvarea problemelor de optimizare combinatorială. Funcţiile de activare se aleg în conformitate cu problema ce trebuie rezolvată (de exemplu, la implementarea memoriilor asociative se utilizează în special funcţia signum).

Funcţionare. Prezenţa conexiunilor inverse face ca în deducerea relaţiilor ce descriu funcţionarea reţelei să se ţină cont de evoluţia în timp a caracteristicilor reţelei (semnale de intrare şi de ieşire din unităţile funcţionale). Pentru a descrie relaţiile de funcţionare ale reţelei, pentru fiecare neuron i vom folosi următoarele notaţii:

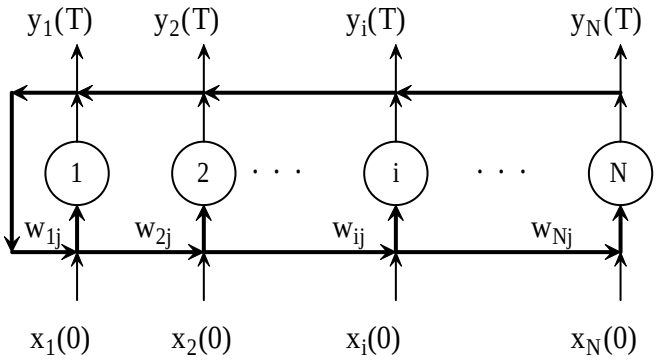


Figura 1: Arhitectura modelului Hopfield

- $x_i(t)$ - potenţialul neuronului la momentul t (suma ponderată a semnalelor provenite de la celelalte unităţi);
- $y_i(t) = f_i(x_i(t))$ - semnalul de ieşire produs de neuron (f_i este funcţia de activare);
- $I_i(t)$ - semnal de intrare primit din partea mediului;
- w_{ij} - ponderea conexiunii dintre unitatea j şi unitatea i. Ansamblul tuturor ponderilor formează o matrice pătratică $W = (w_{ij})_{i=\overline{1,N}, j=\overline{1,N}}$.

Întrucât în majoritatea aplicaţiilor toate unităţile au acelaşi tip de funcţie de activare vom considera că $f_i = f$ pentru $i = \overline{1,N}$. Prezenţa conexiunilor inverse face ca semnalul produs de către o unitate să revină direct (prin bucle) sau indirect (transformat prin intermediul altor unităţi) la unitatea de la care a pornit. Din acest motiv funcţionarea constă de fapt în evoluţia *stării* reţelei până în momentul atingerii unei stări staţionare (care va fi interpretată ca răspunsul reţelei). Orice reţea recurentă poate fi modelată printr-un *sistem dinamic* a cărui evoluţie poate fi descrisă în timp discret (prin intermediul unui sistem de ecuaţii cu diferenţe - relaţii de recurenţă) sau în timp continuu (prin intermediul unui sistem de ecuaţii diferenţiale).
Evoluţie în timp discret. Este adecvată situaţiei în care reţeaua va fi simulată soft. Să considerăm că la un moment dat, t, starea reţelei este determinată de către vectorul $Y(t) = (y_1(t), \ldots, y_N(t))$ al semnalelor pe care le produc unităţile. Există două strategii de modificare a stării reţelei:

- *Asincronă.* La un moment dat o singură unitate (i^*) îşi schimbă starea, astfel că evoluţia stării reţelei poate fi descrisă de:

$$\begin{cases} y_i(t+1) = f\left(\sum_{j=1}^{N} w_{ij} y_j(t) + I_i(t)\right) & i = i^* \\ y_i(t+1) = y_i(t) & i \neq i^* \end{cases} \tag{1}$$

  Alegerea unităţii i^* se poate efectua într-o manieră aleatoare (prin selecţie fără revenire a unui indice din $\{1, \ldots, N\}$) sau într-o manieră secvenţială (prima dată se selectează unitatea 1, după aceea unitatea 2 ş.a.m.d).

- *Sincronă.* Unităţile îşi schimbă simultan starea:

$$y_i(t+1) = f\left(\sum_{j=1}^{N} w_{ij} y_j(t) + I_i(t)\right), \qquad i = \overline{1,N} \tag{2}$$

O astfel de evoluţie poate fi implementată cu uşurinţă în paralel.

Pentru a putea implementa funcţionarea reţelei, relaţiile (1) şi (2) trebuie completate cu starea iniţială ($Y(0) = Y^0$) şi cu o condiţie de oprire de tipul $\|Y(t+1) - Y(t)\| < \epsilon$ ($\|\cdot\|$ reprezintă o normă în R^N iar $\epsilon > 0$ este o măsură a erorii apriori acceptată pentru aproximarea *punctului fix* al procesului iterativ).

Evoluţie în timp continuu. Este adecvată pentru a modela funcţionarea implementărilor hard. În implementarea prin circuite electrice fiecare unitate este realizată dintr-un circuit care conţine un amplificator (modelează funcţia de activare, f) precum şi un rezistor (r_i) conectat în paralel cu un condensator (C_i) (vezi figura 2). Starea fiecărei unităţi, x_i, reprezintă tensiunea de intrare, iar y_i reprezintă tensiunea de ieşire din amplificator. Unitatea i este conectată cu unitatea j prin intermediul unui rezistor, a cărui rezistenţă, R_{ij}, determină ponderea conexiunii.

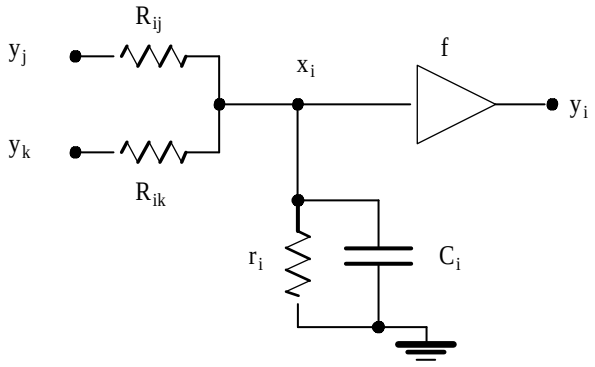


Figura 2: Circuit pentru implementarea unei unităţi şi a conexiunii cu alte unităţi

Cu aceste notaţii ecuaţiile circuitului pot fi scrise:

$$\tau_i \frac{dx_i(t)}{dt} = -x_i(t) + \sum_{j=1}^{N} w_{ij} f(x_j(t)) + I_i(t), \qquad i = \overline{1,N} \tag{3}$$

unde $\tau_i = R_i C_i$, $1/R_i = 1/r_i + \sum_{i=1}^{N} 1/R_{ij}$ şi $w_{ij} = R_i/R_{ij}$. Pentru a simplifica relaţiile se consideră că $\tau_i = 1$.

În ambele variante, pentru a putea determina răspunsul reţelei (astfel încât reţeaua să poată rezolva o sarcină computaţională) este necesar ca procesul de evoluţie să se *stabilizeze* într-o *stare staţionară* , $X = (x_1, \ldots, x_N)$, care verifică

$$x_i = \sum_{j=1}^{N} w_{ij} f(x_j) + I_i, \qquad i = \overline{1, N}. \tag{4}$$

Dacă se consideră ca starea reţelei este reprezentată de vectorul semnalelor de ieşire, $Y = (y_1, \ldots, y_N)$ atunci starea staţionară verifică

$$y_i = f(\sum_{j=1}^{N} w_{ij} y_j + I_i), \qquad i = \overline{1, N}. \tag{5}$$

Problema care apare este de a stabili în ce condiţii reţeaua evoluează către o stare staţionară şi cum este influenţat acest lucru de starea iniţială. În domeniul sistemelor dinamice probleme de acest tip sunt abordate utilizând metoda Liapunov.

Proprietăţi de stabilitate şi funcţii Liapunov. Comportarea unei reţele recurente, descrisă prin evoluţia stării sale (de exemplu $X(t)$) se poate încadra într-una dintre situaţiile:

- $X(t)$ tinde către o stare staţionară X^* (numită şi *punct fix* al dinamicii reţelei).
- $X(t)$ oscilează permanent între două sau mai multe stări posibile (comportare *periodică*).
- $\|X(t)\|$ devine din ce în ce mai mare pe măsură ce t creşte.
- $X(t)$ are o comportare haotică.

Comportarea cea mai utilă, atât din perspectiva utilizării ca memorii asociative cât şi pentru a rezolva probleme de optimizare este cea corespunzătoare primei situaţii. Din punct de vedere practic o proprietate utilă a stării staţionare X^* este cea de *asimptotic stabilitate*. Din punct de vedere intuitiv, proprietatea de stabilitate poate fi ilustrată prin starea de echilibru a unei bile aflată pe o suprafaţă convexă (stare asimptotic stabilă), orizontală (stare stabilă) respectiv concavă (stare instabilă) (vezi figura 3).

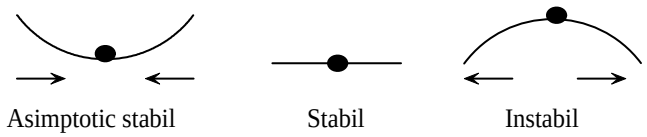


Figura 3: Proprietatea de stabilitate a unui punct de echilibru

Pentru a defini formal aceste noţiuni să considerăm un sistem descris prin

$$\frac{dX(t)}{dt} = F(X(t)), \qquad t \geq 0 \tag{6}$$

iar $X^* = F(X^*)$ un punct fix al acestuia. Pentru sistemul (6) completat cu o condiţie iniţială $X(0) = X^0$ vom nota prin $X(t; X^0)$ unica soluţie ce corespunde condiţiei. Asimptotic stabilitatea constă din două proprietăţi:

Stabilitate:

> X^* este stabilă dacă pentru orice $\epsilon > 0$ există $\delta(\epsilon) > 0$ astfel încât pentru orice X^0 cu $\|X^0 - X^*\| < \delta$ are loc $\|X(t; X^0) - X^*\| < \epsilon$ pentru orice $t > 0$.

Atractivitate:

> *X^* este atractivă dacă există $\delta > 0$ astfel încât pentru orice X^0 cu $\|X^0 - X^*\| < \delta$ are loc $\lim_{t\to\infty} X(t; X^0) = X^*$.*

Mulţimea $\mathcal{A}(X^*) = \{X^0 | \lim_{t\to\infty} X(t; X^0) = X^*\}$ se numeşte *regiune de atracţie* a lui X^*. Dacă $\mathcal{A}(X^*) = R^N$ atunci X^* este *global atractivă* (dacă are şi proprietatea de stabilitate este *global asimptotic stabilă*) altfel este *local atractivă* (respectiv *local asimptotic stabilă*). Global asimpotic stabilitatea este o proprietate utilă în cazul problemelor de optimizare pe când local asimptotic stabilitatea este utilă în modelarea prin reţele recurente a memoriilor asociative.

Verificarea proprietăţilor de asimptotic stabilitate pe baza definiţiilor este dificilă când soluţia nu este cunoscută explicit (cum este cazul reţelelor neuronale). În acest caz un instrument util este metoda lui Liapunov la baza căreia stă următoarea teoremă.

> *Teorema de stabilitate a lui Liapunov.* Dacă există o funcţie $V : R^N \to R$ mărginită inferior astfel încât $\frac{dV(X(t))}{dt} < 0$ pentru orice $t > 0$ atunci orice soluţie staţionară, X^*, a lui (6) este asimptotic stabilă.

Funcţia V se numeşte funcţie Liapunov şi este într-un fel analoagă noţiunii de energie din fizică, în sensul că X^* este un punct de minim al lui V iar o poziţie de echilibru stabil (vezi figura 3) are asociată o energie minimă.

Rezultatele de mai sus au corespondent şi pentru sistemele discrete de forma:

$$X(t+1) = -X(t) + F(X(t))$$

diferenţa principală fiind dată de faptul că derivata funcţiei Liapunov este înlocuită cu o diferenţă de forma: $V(X(t+1)) - V(X(t))$.

Principalele rezultate privind existenţa funcţiilor Liapunov pentru reţele neuronale sunt:

> *Funcţie Liapunov pentru reţele cu dinamică discretă asincronă (1).* Dacă $w_{ij} = w_{ji}$ pentru orice $i \neq j$, $w_{ii} = 0$ pentru orice i, f este funcţia signum (cu valori în $\{-1, 1\}$) sau Heaviside (cu valori în $\{0, 1\}$), iar semnalul de intrare este constant ($I_i(t) = I_i$) atunci
>
> $$V(y_1, \ldots, y_N) = -\frac{1}{2}\sum_{i,j=1}^{N} w_{ij} y_i y_j - \sum_{i=1}^{N} y_i I_i \tag{7}$$
>
> este funcţie Liapunov pentru (1), singurele soluţii asimptotic stabile fiind punctele staţionare.

> *Funcţie Liapunov pentru reţele cu dinamică continuă (3).* Dacă $w_{ij} = w_{ji}$ pentru orice $i \neq j$, $\tau_i = 1$ pentru orice i, f este funcţie diferenţiabilă strict crescătoare, iar semnalul de intrare este constant ($I_i(t) = I_i$) atunci
>
> $$V(x_1, \ldots, x_N) = -\frac{1}{2}\sum_{i,j=1}^{N} w_{ij} f(x_i) f(x_j) - \sum_{i=1}^{N} f(x_i) I_i + \sum_{i=1}^{N} \int_0^{f(x_i)} f^{-1}(z) dz \tag{8}$$
>
> este funcţie Liapunov pentru (3), iar singurele soluţii asimptotic stabile sunt punctele staţionare.

2 Memorii asociative

O memorie asociativă este un sistem de stocare a informaţiei care permite regăsirea acesteia pe baza conţinutului. Spre deosebire de memoriile adresabile clasice unde informaţia este stocată localizat şi regăsirea se bazează pe cunoaşterea adresei unde a fost stocată, în memoriile asociative informaţia este stocată în manieră distribuită iar regăsirea se bazează pe indicii reprezentate de porţiuni din informaţie sau variante afectate de zgomot ale acesteia. Memoriile asociative pot fi utilizate pentru regăsirea informaţiei pe baza conţinutului. Un exemplu simplu este cel al stocării unor şabloane (de exemplu reprezentări ale unor caractere) şi al regăsirii acestora pornind de la variante alterate ale acestora. Principiul memoriilor asociative este mai apropiat de cel al memoriei umane decât modul de funcţionare a memoriilor adresabile.

Memoriile asociative pot fi implementate în hardware (implementări de natură electronică sau optică) sau pot fi simulate soft. Deşi prima varianta este cea cu mai mare potenţial practic în continuare vom analiza simularea soft a memoriilor asociative utilizând modelul Hopfield.

Vom considera că datele care trebuie stocate sunt codificate sub forma unor vectori N-dimensionali ale căror elemente aparţin mulţimii $\{-1, 1\}$. Dacă datele de stocat reprezintă pattern-uri grafice reprezentate prin matrici de pixeli atunci vectorul N-dimensional se obţine prin liniarizarea matricii şi prin asocierea valorii -1 fiecărui pixel inactiv şi a valorii 1 fiecărui pixel activ.

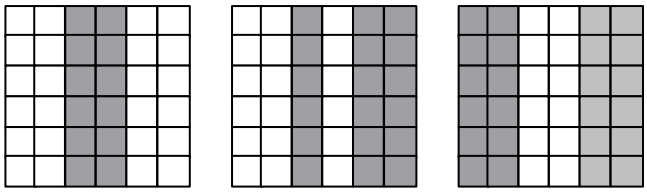


Figura 4: Exemple de şabloane

În Figura 4 sunt ilustrate câteva şabloane reprezentate pe matrici 6×6. Acestor şabloane le corespund vectori cu $N = 36$ componente. De exemplu, pentru şablonul (a) vectorul asociat este $(-1,-1,1,1,-1,-1,-1,-1,1,1,-1,-1,-1,-1,1,1,-1,-1,-1,-1,1,1,-1,-1,-1,-1,1,1,-1,-1,-1,-1,1,1,-1,-1)$

Ca în orice memorie şi în memoriile asociative trebuie asigurate două mecanisme: de stocare a informaţiilor şi de regăsire a acestora.

Stocarea informaţiilor. Dacă se doreşte stocarea unor vectori N-dimensionali atunci trebuie utilizată o reţea total interconectată constituită din N neuroni. În acest caz ponderile conexiunilor pot fi organizate într-o matrice pătratică, $W = (w_{ij})_{i=\overline{1,N},j=\overline{1,N}}$. Stocarea unui set de vectori, $\{X^1, \ldots, X^L\} \in \{-1, 1\}$ este echivalentă cu alegerea ponderilor astfel încât aceşti vectori să corespundă unor stări staţionare ale dinamicii reţelei. Rezultatele din secţiunea anterioară afirmă că dacă matricea ponderilor este simetrică şi se lucrează cu o dinamică asincronă atunci punctele fixe sunt asimptotic stabile. Aceasta va permite ca pornind din configuraţii aflate în regiunea de atracţie a punctelor fixe să se ajungă la acestea prin dinamica naturală a reţelei.

Cele mai cunoscute metode de determinare a ponderilor sunt: regula Hebb şi regula pseudo-inversei (cu varianta sa iterativă reprezentată de algoritmul Diederich-Opper).

Regula Hebb. Este cea mai simplă regulă de stocare şi se bazează pe principiul lui Hebb care afirmă

că tăria sinapsei dintre doi neuroni activaţi simultan creşte. Modalitatea de calcul a ponderilor este în acest caz:

$$w_{ij} = \frac{1}{N}\sum_{l=1}^{L} x_i^l x_j^l, \qquad i = \overline{1,N}, j = \overline{1,N} \tag{9}$$

Dacă se doreşte ca reţeaua să nu conţine autoconexiuni atunci elementele diagonalei principale a matricii de ponderi se setează explicit pe 0 ($w_{ii} = 0$)

Se poate verifica prin calcul direct că dacă vectorii de stocat sunt ortogonali doi câte doi ($(X^k)^T \cdot X^l = 0$ pentru orice $k \neq l$) atunci elementele setului $\{X^1, \ldots, X^L\}$ vor fi puncte fixe ale dinamicii sistemului. Condiţia de ortogonalitate semnifică, din punct de vedere practic, că vectorii sunt necorelaţi, deci suficient de diferiţi între ei. De exemplu vectorii corespunzători şabloanelor (a) şi (b) din Figura 4 sunt ortogonali (numărul poziţiilor în care diferă coincide cu numărul poziţiilor în care coincid). În schimb şabloanele (a) şi (c) nu pot fi considerate necorelate întrucât una este complementara celeilalte. În acest caz produsul scalar este $-N$.

Dacă N este impar condiţia de ortogonalitate nu poate fi satisfăcută exact. În acest caz este suficient ca produsul scalar să fie suficient de mic (de exemplu 1).

Regula pseudo-inversei. Permite stocarea ca puncte fixe nu numai a vectorilor ortogonali doi câte doi ci a mulţimilor de vectori liniar independenţi. Modul de calcul al ponderilor este în acest caz:

$$w_{ij} = \frac{1}{N}\sum_{l,k} x_i^l (Q^{-1})_{lk} x_j^k, \qquad i = \overline{1,N}, j = \overline{1,N} \tag{10}$$

unde $(Q^{-1})_{lk}$ este elementul de pe linie l coloana k a inversei matricii Q ale cărei elemente sunt de forma:

$$Q_{kl} = \frac{1}{N}\sum_{i=1}^{N} x_i^k x_i^l \tag{11}$$

Se poate verifica că dacă matricea Q este inversabilă atunci toate elementele setului $\{X^1, \ldots, X^L\}$ sunt puncte fixe ale dinamicii reţelei.

Pentru a evita calculul matricii inverse s-a dezvoltat un algoritm iterativ ce permite aproximarea acesteia şi a cărui structură este apropiată de cea a algoritmului Widrow-Hoff.

$t := 0$
Se iniţializează matricea $W(0)$ folosind regula Hebb
REPEAT
  FOR $l := 1, L$ DO
    $y_i^l := \sum_{j=1}^{N} w_{ij}(t) x_j^l$, $i = \overline{1,N}$
    $w_{ij}(t+1) := w_{ij}(t) + \frac{1}{N}(x_i^l - y_i^l)x_j^l$, $i = \overline{1,N}$, $j = \overline{1,N}$
  $t := t + 1$
UNTIL $\|W(t) - W(t-1)\| < \epsilon$

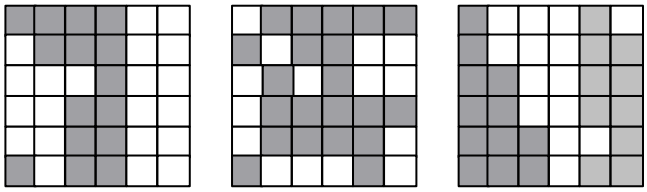
Figura 5: Algoritmul Diederich-Opper

Regăsirea informaţiei. Regăsirea informaţiei este asigurată prin evoluţia naturală a reţelei către unul dintre punctele sale fixe (printre care se află şi vectorii stocaţi) în condiţiile în care starea iniţială a reţelei este setată pe indiciul pornind de la care se doreşte regăsirea. În cazul unui simulator de memorie asociativă regăsirea va consta în fixarea stării iniţiale: $y_i(0) = x_i^s$ $(i = \overline{1,N})$, X^s fiind vectorul corespunzător indiciului, şi în efectuarea procesului iterativ:

$$y_i(t+1) = \text{sgn}\left(\sum_{j=1}^{N} w_{ij} y_j(t)\right), \quad i = \overline{1,N}$$

până când se atinge starea staţionară ($Y(t+1) = Y(t)$ sau distanţa dintre aceştia este suficient de mică).

Întrucât regulile de stocare prezentate mai sus conduc toate la matrici simetrice de ponderi, dacă se lucrează cu dinamică asincronă rezultatele teoretice referitoare la asimptotic stabilitatea stărilor staţionare garantează faptul că se va ajunge la o stare staţionară.



(a) (b) (c)

Figura 6: Variante incomplete sau afectate de zgomot

In figura 6 sunt ilustrate trei exemple de indicii de la carea se poate iniţia regăsirea şabloanelor stocate. Acestea sunt obţinute prin modificarea aleatoare a unor poziţii din şablonul (a) (fig. 4): varianta (a) este obţinută prin modificarea a maxim 25% dintre poziţii, varianta (b) este obţinută prin modificarea a maxim 50% dintre poziţii iar varianta (c) este obţinută prin modificarea a maxim 75% dintre poziţii. Pentru o reţea ale cărei ponderi au fost stabilite pe baza şabloanelor (a) şi (b) din figura 4, pornind de la indiciul (a) sau de la indiciul (b) (fig. 6) se ajunge într-o singura iteraţie la şablonul stocat (a). Pornind de la indiciul (c) se ajunge însă la complementarul primului şablon stocat (vezi fig. 4 (c)).

Capacitatea de stocare. Cantitatea de informaţie care poate fi stocată şi regăsită este denumită capacitate de stocare. Pentru memoriile asociative de tip Hopfield capacitatea de stocare poate fi definită în funcţie de eroarea acceptată în faza de regăsire. Dacă se doreşte ca informaţia să poată fi regăsită exact atunci capacitatea de stocare are ordinul $O(N/(4\ln N))$. Dacă în faza de regăsire se acceptă prezenţa cu o anumită probabilitate (de exemplu $P_{err} = 0.005$) a unor componente eronate (valori 1 şi -1 inversate) capacitatea de stocare este de ordinul $0.15N$.

Atractori paraziţi. Unul dintre principalele dezavantaje ale memoriilor asociative de tip Hopfield este faptul că pe lângă datele care se doresc a fi explicit stocate se stochează în mod implicit şi altele. De exemplu se observă uşor că aplicând regula Hebb prin stocarea unui vector se stochează implicit şi complementarul său (de exemplu la stocarea $(1,-1,1)$ se stochează şi $(-1,1,-1)$). Pe lângă aceşti vectori complementari se mai pot stoca implicit şi combinaţii liniare ale vectorilor din setul de memorat. Pentru evitarea atractorilor paraziţi fie se modifică regula de stocare fie dinamica de regăsire. O variantă de modificare a acesteia din urmă este aceea în care se utilizează unităţi funcţionale cu comportare aleatoare:

$$P(y_i(t+1) = 1|Y(t)) = \frac{1}{1+\exp(-\beta\sum_{j=1}^{n} w_{ij} y_j(t))}$$

$$P(y_i(t+1) = -1|Y(t)) = \frac{1}{1+\exp(\beta\sum_{j=1}^{n} w_{ij} y_j(t))}.$$

Utilizarea unităţilor funcţionale aleatoare permite evadarea din regiunile de atracţie ale atractorilor nedoriţi (a căror dimensiune de este de regulă mai redusă).

3 Reţele cu unităţi contextuale. Prelucrarea seriilor temporale.

O serie temporală este o serie de date înregistrate la momente succesive de timp care permite descrierea unei mărimi ce evoluează în timp (de exemplu preţul unui tip de acţiuni, rata de schimb valutar, temperatura atmosferică etc.). Elementele seriei pot fi valori scalare sau vectoriale. În continuare vom analiza cazul mai simplu al valorilor scalare însă cazul vectorial este principial similar.

Una dintre principalele prelucrări ce intervin în prelucrarea seriilor temporale este predicţia valorii (valorilor) viitoare din serie. În scopul efectuării unei predicţii este utilă punerea în evidenţă a unui model care surprinde dependenţa între valoarea viitoare a seriei şi valoarea curentă şi eventual cele anterioare. Pentru o serie scalară $x(t)$ un model este o funcţie $F : R^{p+s} \to R$ ce satisface:

$$x(t+1) = F(x(t), x(t-1), \ldots, x(t-p+1), \varphi_1, \varphi_2, \ldots, \varphi_s)$$

cu φ_i parametri ce modelează factorii externi ce influenţează valoarea viitoare a seriei. În funcţie de proprietăţile funcţiei F modelele sunt: liniare sau neliniare, deterministe sau nedeterministe (componentele aleatoare se introduc pentru a modela factorii necunoscuţi şi/sau erorile de măsurare). În statistică sunt utilizate diverse modele. Unul dintre acestea, cunoscut ca model autoregresiv de ordin p, $AR(p)$ se bazează pe relaţia:

$$x(t+1) = \sum_{i=0}^{p-1} \alpha_i x(t-i) + \epsilon \tag{12}$$

unde ϵ este o variabilă aleatoare cu repartiţie normală şi medie nulă.

Reţelele neuronale permit modelarea unor dependenţe neliniare. Atunci când ordinul modelului, p, este cunoscut se pot utiliza reţele fără recurenţe dar în cazul în care p nu este cunoscut reţelele recurente sunt mai adecvate.

Reţele feedforward cu ferestre temporale. Permit efectuarea de predicţii în cazul în care se cunoaşte numărul de valori anterioare care influenţează valoarea viitoare. Arhitectura acestor reţele, ilustrată în fig.7(a), se caracterizează prin faptul că nivelul de intrare conţine pe lângă unitatea ce primeşte semnalul curent, $x(t)$, alte $p-1$ unităţi care transmit semnalele anterioare: $x(t-1)$, ..., $x(t-p+1)$. Se poate considera că aceste unităţi nu comunică direct cu mediul ci îşi transmit succesiv semnalele primite de la prima unitate. În felul acesta dependenţa temporală specifică seriei este transformată într-una spaţială.

Dacă pe nivelul ascuns există K unităţi având funcţia de transfer f, semnalul produs de reţea (aproximarea următoarei valori a seriei) este:

$$y = \sum_{k=1}^{K} w_k^2 f(\sum_{j=0}^{p-1} w_{kj}^1 x(t-j))$$

Întrucât reţeaua nu conţine conexiuni inverse funcţionarea este cea specifică unei reţele feedforward. În ceea ce priveşte antrenarea, aceasta poate fi asigurată aplicând algoritmul BackPropagation cu condiţia ca setul de antrenare să fie pregătit adecvat. Folosind setul de valori cunoscute ale seriei: $x(1)$, $x(2)$,...,$x(p)$,$x(p+1)$, ...$x(t)$ elementele din setul de antrenare vor fi perechi de forma $((x(l), x(l-1), \ldots, x(l-p+1)), x(l+1))$ pentru $l = \overline{p, t}$.

Reţele de tip Elman. Principalul dezavantaj al reţelelor cu ferestre temporale este faptul că necesită cunoaşterea parametrului p. Reţelele Elman permit modelarea implicită a dependenţei temporale prin utilizarea unor unităţi contextuale ce joacă rolul de memorie de scurtă durată. Acest tip de reţea a fost dezvoltată în 1990, iniţial pentru a învăţa structura gramaticală a unui set de propoziţii generate aleator pe baza unui vocabular limitat. Scopul reţelei era predicţia următorului cuvânt din reţea pe baza celui curent.

Arhitectura reţelei (descrisă în fig.7(b)) constă dintr-o reţea feedforward cu un nivel ascuns completată cu un nivel de unităţi contextuale care comunică doar cu unităţile ascunse. Fiecărei unităţi ascunse îi corespunde o unitate contextuală ce conţine la momentul t valoarea produsă de unitatea ascunsă corespunzătoare la momentul $t-1$. Singurele conexiuni inverse sunt cele de la unităţile ascunse către unităţile contextuale corespunzătoare. Ponderile acestor conexiuni sunt fixate pe 1, realizându-se de fapt o copiere a semnalului de ieşire a nivelului ascuns în nivelul contextual. Notând cu $h_i(t)$ semnalul produs la momentul t de către unitatea ascunsă i, cu W^x vectorul cu ponderile conexiunilor dinspre unitatea de intrare către nivelul ascuns, cu W^c matricea ponderilor conexiunilor directe dintre unităţile contextuale şi cele ascunse şi cu W^2 vectorul cu ponderi ale conexiunilor către nivelul de ieşire, semnalul de ieşire, corespunzător intrării $x(t)$ se calculează prin:

$$y = \sum_{k=1}^{K} w_k^2 h_k(t)$$

$$h_k(t) = f\left(w_k^x x(t) + \sum_{j=1}^{K} w_{kj}^c h_j(t-1)\right)$$

Se consideră că $h_j(0) = 0$. Deşi sunt prezente conexiuni inverse funcţionarea nu este de tip dinamic. Ordinul modelului nu apare explicit în arhitectura reţelei însă abilitatea ei de predicţie este influenţată de numărul de unităţi ascunse.

Întrucât ponderile conexiunilor inverse sunt fixate antrenarea este similară cu cea din cazul reţelelor feedforward. Setul de antrenare conţine perechi de forma $(x(l), x(l+1))$, $l = \overline{1, t-1}$. Ca algoritm de învăţare se poate folosi BackPropagation.

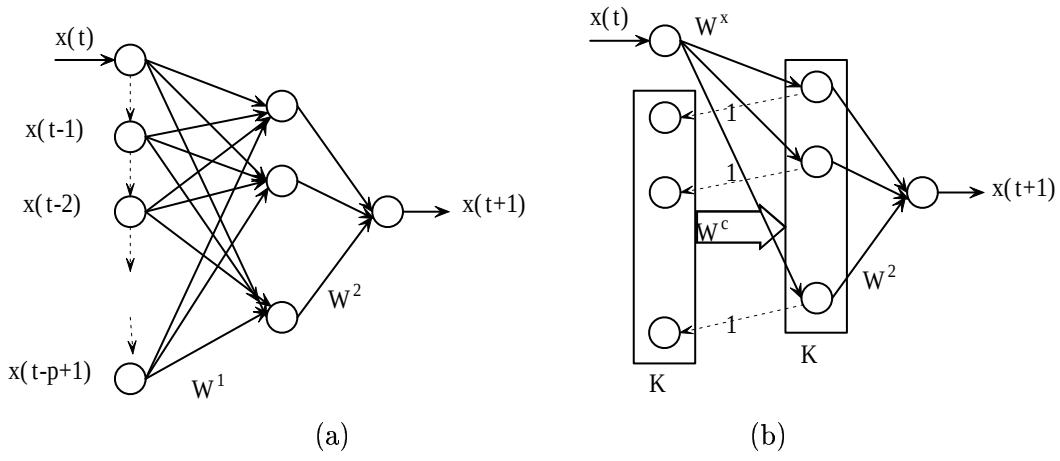


Figura 7: Arhitecturi specifice prelucrării seriilor temporale. (a) Reţea feedforward cu fereastră temporală; (b) Reţea de tip Elman.

4 Reţele celulare.

Reţelele celulare se caracterizează prin faptul că unităţile funcţionale (care, la fel ca în cazul modelului Hopfield, sunt în acelaşi timp unităţi de intrare şi de ieşire) sunt plasate în nodurile unei grile geometrice (similar nivelului de unităţi funcţionale ale reţelelor de tip Kohonen) şi sunt conectate doar cu unităţile din vecinătatea lor. Aceasta conectivitate locală face ca aceste reţele să poate fi uşor implementate prin circuite electronice. Aceasta permite realizarea unor chip-uri care rezolvă rapid anumite probleme (de exemplu, prelucrări de imagini). Pe de altă parte, simularea soft a acestor reţele permite modelarea unor fenomene neliniare, a unor dinamici haotice sau a unor sisteme fizice.

Arhitectura. Ca şi în cazul reţelelor de tip Kohonen cel mai frecvent model este cel bazat pe o grilă bidimensională în nodurile căreia sunt plasate unităţile funcţionale. Fiecare unitate poate fi astfel identificată prin poziţia ei în cadrul grilei: $p = (i, j) \in \{1, \ldots, n\} \times \{1, \ldots, n\} = \mathcal{L}$ şi are asociat un set de vecini. Vecinătatea de ordin s a unităţii p se defineşte utilizând o funcţie distanţă: $V_s(p) = \{q \in \mathcal{L} | d(p, q) \leq s\}$. Se poate utiliza distanţa euclidiană sau distanţa Manhattan. Fiecare unitate p este conectată cu toate unităţile din $V_s(p)$. În figura 8 este ilustrată arhitectură unei reţele cu 25 de unităţi bazată pe o grilă pătratică şi vecinătăţi de ordin 1.

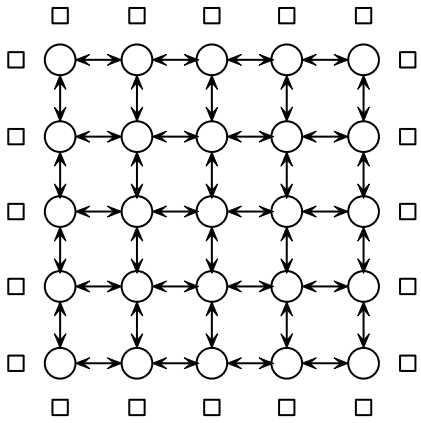


Figura 8: Reţea celulară cu grilă pătratică şi vecinătăţi de ordin 1 .

Funcţionare. Funcţia de transfer cel mai frecvent utilizată la reţelele celulare este cea rampă: $f(u) = (|u + 1| - |u - 1|)/2$. Pentru a descrie relaţiile de funcţionare ale reţelei folosim următoarele notaţii:

- $X(t) = \{x_p(t) | p \in \mathcal{L}\}$ reprezintă ansamblul semnalelor de intrare în unităţile funcţionale;
- $Y(t) = \{y_p(t) | p \in \mathcal{L}\}$ reprezintă ansamblul semnalelor de ieşire, $y_p(t) = f(x_p(t))$;
- $U(t) = \{u_p(t) | p \in \mathcal{L}\}$ reprezintă ansamblul semnalelor de control transmise unităţilor;
- $I(t) = \{i_p(t) | p \in \mathcal{L}\}$ reprezintă ansamblul pragurilor corespunzătoare unităţilor;
- a_{pq} este ponderea conexiunii dintre unitatea q şi unitatea p;
- b_{pq} este ponderea conexiunii de la componenta q a semnalului de control la unitatea funcţională p;

Cu aceste notaţii funcţionarea reţelei poate fi descrisă prin:

$$\frac{dx_p(t)}{dt} = -x_p(t) + \sum_{q \in V(p)} a_{pq} y_q(t) + \sum_{q \in V(p)} b_{pq} u_q(t) + i_p(t), \quad p \in \mathcal{L} \tag{13}$$

Pentru unităţile aflate pe frontieră se poate considera că ele primesc semnale şi de la nişte unităţi fictive (marcate în figura 8 prin pătrate). Semnalul pe care unitatea p de pe frontieră îl primeşte de la unitatea fictivă corespunzătoare va fi notat cu z_p. Alegerea valorii pentru aceste semnale depinde de problema pentru care se proiectează reţeaua însă în multe aplicaţii se consideră că $z = 0$.

Relaţia (13) poate fi simplificată dacă ponderile conexiunilor dintre unităţile vecine nu depind de poziţia acestora. Aceasta înseamnă că pentru o unitate $p = (i, j)$ ponderea conexiunii dintre ea şi unitatea vecină $q = (i-1, j)$ este aceeaşi cu ponderea conexiunii dintre unitatea $p' = (i', j')$ şi unitatea $q' = (i'-1, j')$. Astfel ponderea unei conexiuni depinde doar de poziţia relativă a celor două unităţi vecine şi nu de poziţia lor absolută în cadrul reţelei: $a_{pq} = a_{p'q'}$. Valoarea comună a celor două ponderi se notează cu $a_{-1,0}$.

Astfel de reţele se numesc reţele celulare cu şabloane independente de poziţie ("cloning template cellular neural network").

Dacă se lucrează cu vecinătăţi de ordinul 1, de forma:

$$V_1(i,j) = \{(i-1,j-1), (i-1,j), (i-1,j+1), (i,j-1), (i,j), (i,j+1), (i+1,j-1), (i+1,j), (i+1,j+1)\}$$

parametrii lor sunt reprezentaţi de două matrici şablon:

$$A = \begin{bmatrix} a_{-1,-1} & a_{-1,0} & a_{-1,1} \\ a_{0,-1} & a_{0,0} & a_{0,1} \\ a_{1,-1} & a_{1,0} & a_{1,1} \end{bmatrix} \qquad B = \begin{bmatrix} b_{-1,-1} & b_{-1,0} & b_{-1,1} \\ b_{0,-1} & b_{0,0} & b_{0,1} \\ b_{1,-1} & b_{1,0} & b_{1,1} \end{bmatrix} \tag{14}$$

Presupunând că toate unităţile au asociată aceeaşi valoare a pragului, I, numărul total de parametri ai reţelei este 19 fapt ce facilitează procesul de determinare a acestora.

Cu aceste notaţii sistemul de ecuaţii (13) poate fi rescris într-o formă mai simplă:

$$\frac{dx_{i,j}(t)}{dt} = -x_{i,j}(t) + \sum_{(k,l) \in V^*} a_{k,l} y_{i+k,j+l}(t) + \sum_{(k,l) \in V^*} b_{k,l} u_{i+k,j+l}(t) + I, \qquad i,j \in \{1, \ldots, n\} \tag{15}$$

cu

$$V^* = \{(-1,-1), (-1,0), (-1,1), (0,-1), (0,0), (0,1), (1,-1), (1,0), (1,1)\}$$

La simularea soft a acestor reţele sistemul (13) sau sistemul (15) completat cu o condiţie iniţială $x_p(0) = x_p^0$, $p \in \mathcal{L}$ se rezolvă numeric (utilizând de exemplu metoda Euler explicită). În aceste condiţii răspunsul reţelei se obţine efectuând procesul iterativ:

$$x_p(t+1) = (1-h)x_p(t) + h(\sum_{q \in V(p)} a_{pq} y_q(t) + \sum_{q \in V(p)} b_{pq} u_q(t) + i_p(t)), \quad p \in \mathcal{L}$$

respectiv

$$x_p(t+1) = (1-h)x_p(t) + h(\sum_{(k,l) \in V^*} a_{k,l} y_{i+k,j+l}(t) + \sum_{(k,l) \in V^*} b_{k,l} u_{i+k,j+l}(t) + I), \qquad i,j \in \{1, \ldots, n\}$$

Parametrul h din relaţiile de mai sus reprezintă pasul de integrare utilizat de metoda numerică.

Utilizarea în prelucrarea imaginilor. Reţelele celulare cu şabloane independente de poziţie pot fi utilizate pentru implementarea unor filtre din prelucrarea imaginilor alb-negru. Presupunem că imaginea care trebuie prelucrată este constituită din $n \times n$ pixeli şi este codificată printr-o matrice Q ale cărei elemente corespund pixelilor (-1 pentru un pixel alb şi 1 pentru un pixel negru).

Alegând adecvat şabloanele A şi B, semnalul de control U, pragul I, valorile de pe frontieră, z şi starea iniţială a reţelei pot fi implementate diverse filtre din prelucrarea imaginilor. Pentru unele prelucrări pot exista mai multe variante de determinare a parametrilor. La prelucrările simple, determinarea parametrilor se bazează pe regulile locale de transformare a pixelilor ce asigură transformarea dorită a imaginii (de exemplu la eliminarea zgomotului cea mai simplă regulă este cea a majorităţii: un pixel alb încojurat de pixeli negri se transformă în negru şi invers). În cazul unor prelucrări complexe determinarea parametrilor se realizează prin învăţare (în cazul reţelelor celulare învăţarea se bazează pe diverse metode de optimizare, cum ar fi algoritmii evolutivi).

Câteva dintre cele mai simple prelucrări de imagini şi parametrii reţelei celulare corespunzătaore sunt prezentaţi în continuare.

Determinarea contururilor. Se aleg următorii parametri:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{16}$$

$I = -1$, $z = 0$. Semnalul de control este reprezentat de imaginea de prelucrat ($U(t) = Q$) iar starea iniţială a reţelei poate fi aleasă arbitrar.

Eliminarea zgomotului (pixeli izolaţi). Se aleg următorii parametri:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{17}$$

$I \in (-1, 1)$, $z = 0$. Semnalul de control este reprezentat de imaginea de prelucrat ($U(t) = Q$) iar starea iniţială a reţelei se fixează pe 0 ($x_{i,j}(t) = 0$ pentru orice i şi j).

SISTEM DE RECUNOAŞTERE AUTOMATĂ A VORBIRII BAZAT PE REŢELE NEURONALE FUZZY

În aceast ă lucrare este prezentat un system de recunoaştere automată a vorbirii bazat pe reţele neuronale fuzzy, ca elemente de clasificare şi recunoaştere a eşantioanelor vocale prezentate sistemului. Sistemul prezentat este capabil să realizeze recunoaşterea frazelor discrete, independent de vorbitor, pe baza unor serii de timp ale rapoartelor de formanţi. Sistemul a fost implementat pe un PC cu soundcard şi microfon, iar aplicaţiile software utilizate au fost programul GoldWave, pentru înregistrarea on-line a eşantioanelor vocale şi mediul de programare MatLab, pentru prelucrarea semnalelor şi simularea reţelei neuronale.

1 Introducere

Interesul actual generat de reţelele neuronale fuzzy este determinat în mare parte de speranţa că aceste tehnologii, bazate pe modelul creierului uman, vor fi capabile să rezolve categorii de probleme aflate acum mult dincolo de capacităţile de rezolvare ale calculatoarelor din ziua de azi [1] (funcţii extrem de complexe, din punct de vedere al posibilităţilor de calcul, pentru calculatoarele numerice convenţionale, cum sunt înţelegerea vorbirii sau a stimulilor vizuali, sunt mult mai eficient realizate de sistemele neurologice ale organismelor biologice). Studiul reţelelor neuronale a determinat apariţia unor categorii de arhitecturi de reţele care să modeleze capabilităţile creierului uman.

Recunoaşterea automată a vorbirii reprezintă un element important în sistemele de înţelegere a vorbirii, care tinde să îmbunătăţească uşurinţa cu care oamenii interacţionează cu calculatorul. Astfel, comunicaţia cu calculatorul prin vorbire permite utilizatorilor accesarea de la distanţă a serviciilor furnizate de calculator chiar şi prin linii telefonice standard. Cerinţele pentru aceste tehnologii sunt determinate în principal de raţiuni economice, dar şi de dezvoltarea de noi tehnologii în domeniul telefoniei.

Astfel recunoaşterea automată a vorbirii se referă la utilizarea unor instrumente pentru a realiza identificarea unor elemente lingvistice prezente în exprimarea limbajului uman. Sistemele de recunoaştere automată a vorbirii reprezintă primul pas în dezvoltarea unui sistem care recunoaşte cuvintele unei persoane, interpretează sensul şi furnizează un răspuns adecvat.

Actualmente atât din raţiuni economice, cât şi financiare se doreşte realizarea unor sisteme de recunoaştere automată a vorbirii capabile să înveţe rapid noi seturi de eşantioane vocale, independent de vorbitor sau de limbaj. Ideal, eşantioanele vocale trebuie memorate într-o manieră modulară astfel încât aceste submulţimi de informaţii să poată fi combinate în pachete adaptate fiecărei noi aplicaţii, fără necesitatea unei reinstruiri pentru extinderea sistemului.

2 Etapele procesului de recunoaştere automată a vorbirii

Procesul de recunoaştere automată a vorbirii poate fi descris ca un proces în patru etape şi anume: înregistrarea eşantioanelor vocale, prelucrarea semnalelor, extragerea caracteristicilor şi clasificarea exprimărilor [2].

Prima etapa este cea a înregistrării eşantioanelor vocale, în care este extreme de importantă recepția eficientă şi conversia semnalului acustic provenit de la vorbitor într-un echivalent electronic care poate fi memorat pentru prelucrări ulterioare.

A doua etapă este etapa de prelucrare a semnalelor şi constă în principal în analiza spectrală a semnalului electronic pentru a obține o reprezentare parametrizată care codifică informația spectrală.

Etapa a treia este etapa de extragere a caracteristicilor şi se referă în principal la eliminarea informațiilor nerelevante sau redundante din prezentarea parametrizată obținută în etapa a doua. Se obține astfel o compresie a datelor şi o reducere a încărcării pentru prelucrările ulterioare.

Ultima etapă este etapa de clasificare a exprimării şi constă în identificarea cuvintelor rostite. Această etapă implică recunoaşterea fonemelor, grupelor de foneme, cuvinte sau fraze, cu ajutorul rețelelor neuronale, a modelării Markov ascunse [3].

3 Sistem de recunoaştere automată a vorbirii bazat pe rețele neuronale fuzzy

Sistemul de recunoaştere automată a vorbirii bazat pe rețele neuronale fuzzy a fost implementat pe un PC cu soundcard şi microfon, iar aplicațiile software utilizate au fost programul GoldWave, pentru înregistrarea on-line a eşantioanelor vocale şi mediul de programare MatLAB, pentru prelucrarea semnalelor şi simularea rețelei neuronale.

Prezentarea sistemului de recunoaştere automată (ASR – Automatic Speeck Recognition) se face pe baza celor patru etape ale procesului de recunoaştere automată a vorbirii, descrise anterior.

3.1 Înregistrarea eşantioanelor vocale

Necesitatea efectuării unor teste cât mai apropiate de realitate exclud utilizarea unor eşantioane vocale înregistrate într-un interval restrâns de timp. Este necesară deci o bază de date de eşantioane vocale înregistrate pe o perioadă mare de timp, pentru a îngloba astfel variațiile zilnice care apar în vocea unui vorbitor. Pentru optimizare, s-a utilizat un set de mai multe fraze, înregistrate într-un interval larg de timp şi memorate în fişiere pe calculator.

Pentru a obține un sistem robust este important să se țină cont de diverşi factori implicați în metodele utilizate privind mediul acustic, sistemul de microfoane, sistemul de transmitere şi variabilitatea specifică vorbirii, astfel încât eşantionarea şi memorarea semnalelor să se realizeze cu minimum de distorsiuni.

Toate eşantioanele vocale au fost numerizate utilizând placa de sunet a calculatorului. Semnalele audio au fost numerizate utilizând o rată de eşantionare de 22050 Hz cu o rezoluție de 16 biți.

3.2 Prelucrarea semnalelor

În prima etapă, semnalele vocale au fost eşantionate şi convertite într-un format digital, cu o rată de eşantionare de 22050 Hz şi o rezoluție de 16 biți pe eşantion. Înregistrările cu ajutorul telefonului au fost eşantionate cu o rată de 8000 Hz, de o rezoluție de 16 biți pe eşantion.

În etapa a doua s-a realizat analiza spectrală a semnalelor eşantionate, cu ajutorul unui program elaborat în limbajul MatLAB, care utilizează o fereastră Hamming de 512 puncte de eşantionare. Amplitudinea semnalului în domeniul timp a fost normalizată înainte de calcularea spectrogramei. Pragul relativ al zgomotului de fond variază puternic între eşantioanele vocale, astfel încât utilizarea unui prag fix al zgomotului de fond poate constitui o sursă de erori.

Eşantioanele vocale au fost împărţite în segmente de lungime fixată (1 s), s-a calculat spectograma pentru fiecare segment, iar spectograma finală a fost obţinută concatenând spectogramele rezultate pentru fiecare segment în parte.

Pentru îndeplinirea cerinţelor de memorare, s-au memorat în fişierul de ieşire doar părţile din spectogramă care reprezentau frecvenţele mai mici de 5000 Hz.

Spectrograma finală obţinută cu ajutorul programului elaborat în limbajul MatLAB este reprezentată în figura 1.

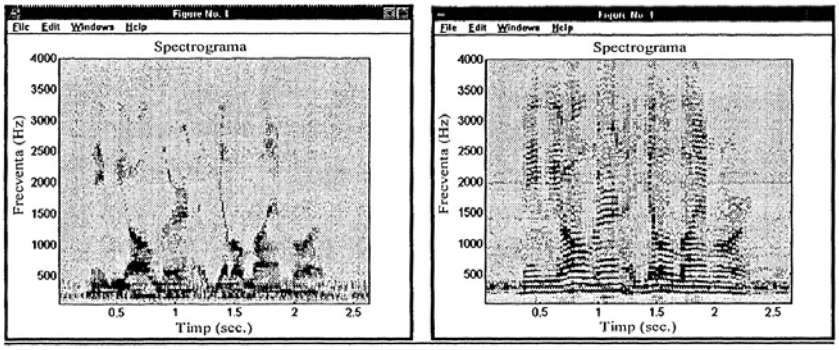


Fig. 1: Spectrograma totală a semnalului.

3.3 Extragerea caracteristicilor

Etapa a treia se referă la izolarea caracteristicilor lingvistice din spectogramele eşantioanelor vocale, după eliminarea datelor nerelevante, pentru a reduce încărcarea în etapele de calcul ulterioare.

Pentru filtrarea zgomotelor de fond care apar evident în esantioanele vocale s-a construit o mască de zgomot, utilizând eşantioane cu zgomot de fond prelevate la diferite intervale de timp. Pentru construcţia acestei măşti, s-a calculat o spectogramă pe o secundă a zgomotului de fond eşantionat, s-a identificat amplitudinea maximă pentru fiecare bandă de frecvenţă din spectogramă zgomotului, obţinându-se astfel un vector-mască. Acest spectru de valori maxime ale amplitudinii zgomotului a fost scăzut din fiecare secvenţa de timp a spectogramelor calculate mai sus, eliminându-se astfel orice contribuţie posibilă a zgomotului de fond. Valorile negative obţinute în urma calculelor au fost considerate nule. Utilizarea acestei metode de mascare a zgomotului a îmbunătăţit identificarea frecvenţelor formanţilor. Vectorii de mascare obţinuţi sunt prezentaţi în fig. 2, iar spectrograma semnalului după mascarea zgomotului de fond este indicată în fig. 3.

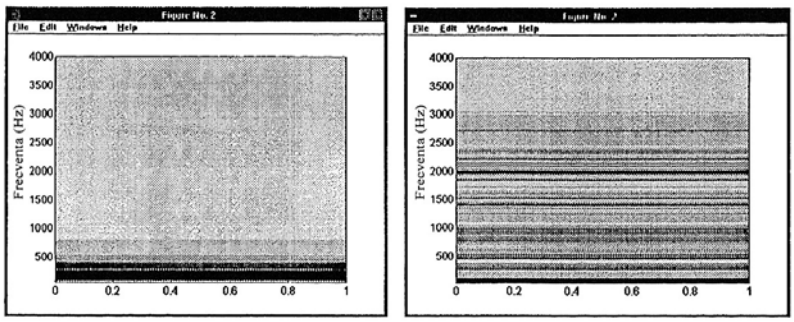


Fig. 2: Masca de zgomot în cazul înregistrării
cu ajutorul microfonului/telefonului.

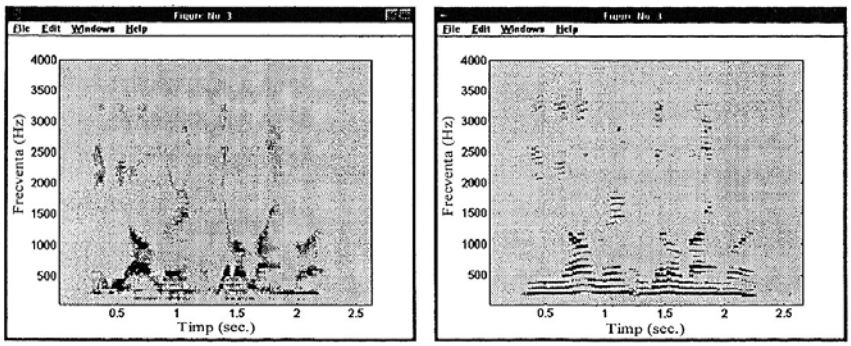


Fig. 3: Spectrograma semnalului după mascarea zgomotului.

Eliminarea perioadelor de pauză din eşantionul vocal s-a realizat pe baza unei curbe de energie pe termen scurt, obţinută din spectograma eşantionului. Fiecare punct al curbei corespunde sumei amplitudinilor în frecvenţă pentru o secvenţă de timp dată. Deoarece formanţii apar doar în porţiunile de eşantion în care se vorbeşte, este importantă izolarea acestor porţiuni pentru a obţine îmbunătăţirea înregistrării formanţilor.

Întrucât aceste porţiuni au energie relativă mai mare în comparaţie cu perioadele de pauză, toate secvenţele de timp în care curba de energie scade sub un anumit prag (fixat la un procent din maximul de energie din curbă) sunt eliminate.

Pentru fiecare bandă de frecvenţă dintr-o secvenţă de timp dată dintr-o spectogramă, se compară amplitudinea frecvenţei cu cele ale frecvenţelor învecinate într-o fereastră de dimensiuni fixe, şi în care banda de frecvenţă respectivă este poziţionată la mijlocul ferestrei. După ce au fost identificate toate maximale locale, se memorează doar acelea cu amplitudinea cea mai mare ca fiind valori utile pentru localizarea frecvenţelor formanţilor.

Prin procesul de separare se elimină secvenţele de timp din spectogramă care au estimări eronate ale frecvenţelor formalilor. Dacă prima frecvenţă este estimată la mai mult de 1200 HZ, sau dacă nu există trei estimări pentru o secvenţă dată, atunci secvenţa respectivă este eliminată. Rezultatul obţinut reprezintă o serie de estimări considerate corespunzătoare, care sunt apoi memorate într-un vector formant.

3.4 Clasificarea exprimărilor

În această etapă, reţeaua neuronală fuzzy este utilizată pentru a clasifica vectorii reprezentând esantioanele vocale obţinuţi în etapa a treia. Reţelele neuronale fuzzy bazate pe teoria rezonanţei adaptive combină logica fuzzy şi procesele bazate pe teoria rezonanţei adaptive de instruire, rezonanţă, selectarea clasei. Valorile de intrare variază continuu între zero şi unu, astfel că intrările pot fi atât analogice cât şi binare, sistemul fiind capabil să realizeze o instruire nesupervizată la prezentarea la intrare a unor modele arbitrare în timp ce se menţin stabilitatea şi plasticitatea, utilizând complet capacităţile de memorie. Astfel arhitectura reţelelelor neuronale fuzzy bazate pe teoria rezonanţei adaptive diferă de cea a reţelelor neuronale obişnuite, avantajul principal fiind acela că sistemul este proiectat să funcţioneze corect şi autonom, să înveţe într-o manieră cât mai stabilă şi într-un mediu care variază continuu când este expus unei secvenţe arbitrare de modele la intrare, până la utilizarea completă a capacităţilor de memorie [4].

Figura 4 prezintă procesul de clasificare prin trei grafice. În primul grafic se află reprezentarea ca vector raport tri-format (TRFV – Tri Formant Ratio Vector) a eşantionului vocal prezentat reţelei. Cel de-al doilea grafic indică vectorul regăsit de reţeaua propusă ca răspuns la intrarea reţelei, iar ultimul grafic indică o comparaţie între cei doi vectori.

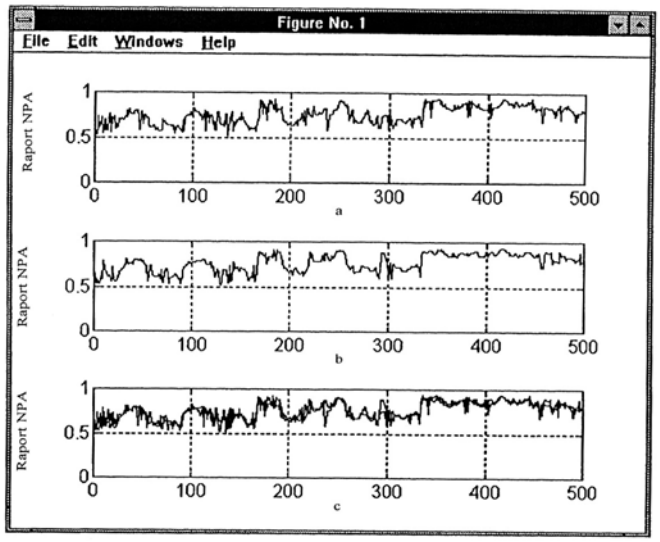


Fig. 4: Recunoaşterea formelor cu ajutorul reţelei neuronale pentru un semnal obţinut prin microfon.

4 Concluzii

Lucrarea demonstrează aplicabilitatea reţelelor neuronale fuzzy în cazul procesului de recunoaştere automată a vorbirii prin implementarea unui sistem care utilizează reţeaua fuzzy ca element de recunoaştere. Sistemul de recunoaştere automată a vorbirii prezentat este capabil să realizeze recunoaşterea frazelor discrete, independent de vorbitor, pe baza unor serii de timp ale rapoartelor de formanţi.

Procesul de conversie a semnalului acustic într-o reprezentare corespunzătoare pentru intrarea în blocul de recunoaştere a fost descris ca un proces de recunoaştere automată a vorbirii în patru etape, care include înregistrarea eşantioanelor vocale, prelucrarea semnalelor, extragerea caracteristicilor şi clasificarea exprimărilor.

Au fost utilizate atât microfonul cât şi telefonul în etapa de înregistrare a sunetelor pentru a măsura şi compara performanţele sistemului în ambele cazuri de transmitere a semnalului. Semnalele acustice au fost numerizate şi convertite într-o spectrogramă. Pentru obţinerea primilor trei formanţi s-a realizat reducerea zgomotelor, a perioadelor de pauză şi o normalizare a amplitudinii în frecvenţă. Formanţii au fost apoi combinaţi secvenţial într-o reprezentare vector raport tri-format, care reduce puternic diferenţele între diverse reprezentări ale frazelor rostite, diferenţe datorate vârstei şi sexului vorbitorului. Reprezentarea a fost apoi normalizată în gama de valori fuzzy şi s-a realizat apoi o limitare a vectorului raport tri-format la dimensiunea stratului de intrare al reţelei neuronale.

Cercetări ulterioare pot fi realizate pentru a îmbunătăţi analiza erorilor, pentru dezvoltarea unor tehnici de normalizare a amplitudinii în frecvenţă care să adapteze atenuarea spectrală a unei conexiuni telefonice. Pot fi obţinute îmbunătăţiri majore prin utilizarea unui sistem de testare on-line, pentru a profita de avantajul măştii de zgomot adaptată pentru fiecare eşantion vocal, şi prin îmbunătăţirea rezoluţiei temporale pentru detecta schimbările rapide în semnalul vocal prin utilizarea unei rate de eşantionare mai mari. În final, sistemul poate fi făcut mai rapid prin compilarea sa ca un program executabil.

Sistemele de recunoaştere automată a vorbirii reprezintă primul pas în dezvoltarea unui sistem care recunoaşte cuvintele unei persoane, interpretează sensul şi furnizează un răspuns adecvat. Cercetările efectuate în acest domeniu sunt importante deoarece dezvoltările ulterioare vor permite

interacțiunea cu calculatorul cu ajutorul vorbirii, uşurând astfel accesul persoanelor neinstruite în domeniul informaticii la resurse controlate de calculator. Aceste noi tehnologii permit executarea de tranzacții on-line, creare de documente şi trimitere de faxuri prin simple dictări către un microfon ataşat calculatorului, posibilitatea realizării de traduceri bi-direcționale pentru o gamă largă de limbi de circulație internațională.

Referințe

[1] D. Dumitrescu, H. Costing. *Rețele neuronale – Teorie şi aplicații*, Editura Teora, Bucureşti, 1996
[2] N. Dixon, T. Martin. Automatic Speeck & Speaker Recognition, *IEEE Press*, New York, 1999.
[3] W. A. Ainsworth, *Speech Recognition by Machine*, Peter Peregrinus Ltd., London, 1988.
[4] G. Carpenter, S. Grossberg, The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network, *IEEE Computer*, March, 1991.

PRELUCRAREA ŞI ANALIZA IMAGINILOR

dr. dipl. ing. Catalin DUMITRESCU

Cuprins

Capitolul 1

INTRODUCERE

Prelucrarea şi analiza imaginilor (numită adeseori prescurtat doar prelucrarea imaginilor) s-a născut datorită ideii şi necesităţii de a înlocui observatorul uman printr-o maşină. Este important de precizat că analiza imaginilor a mers mai departe decât simpla înlocuire a observatorului uman, deoarece au apărut soluţii novatoare pentru probleme cu care acesta nu mai fusese confruntat - ca în cazul imaginilor non-vizibile (imagini acustice, ultrasonore, radar). După cum se remarcă în [9], prelucrarea imaginilor înglobează posibilitatea de a dezvolta maşina totală de viziune, capabilă să realizeze funcţiile vizuale ale oricărei vieţuitoare (desigur, după realizarea a importante dezvoltări teoretice şi tehnologice).

> "Image processing holds the possibility of developing the ultimate machine that could perform the visual functions of all living beings".

Trebuie remarcată terminologia anglo-saxonă (originală), în care disciplina este denumită *Digital Image Processing*, deci prelucrarea digitală a imaginilor. Prin prelucrarea digitală a imaginilor se înţelege prelucrarea pe un calculator digital a unor date bidimensionale (imagini). Termenul cheie este cuvântul *digital*, înlocuit adesea în mod eronat în multe traduceri româneşti cu termenul de numeric. După cum o arată dicţionarul limbii române moderne, definiţia cuvântului *numeric* este aceea de

> "care aparţine numerelor, privitor la numere, exprimat prin numere".

Rezultatul oricărui calcul este numeric. Termenul *digital* înseamnă însă

> "ceea ce este referitor la reprezentarea informaţiei discrete în calculatoare"

Deci atâta vreme cât acceptăm ideea că unealta de lucru în prelucrarea imaginilor este calculatorul, şi acesta la rândul său este digital, atunci şi prelucrarea este la rândul ei digitală, ca un caz particular al oricărei prelucrări numerice. Desigur că există însă şi prelucrări de imagini care sunt analogice - aşa cum sunt toate prelucrările ce au loc în cadrul lanţului de transmisie şi recepţie a imaginii standard de televiziune.

1.1 Imagini digitale

La început, imaginile sunt semnale, dar nu funcţii temporale, ci funcţii definite pe un domeniu spaţial. Orice imagine este o structură bidimensională (tablou, matrice) de date. Un element al imagini se numeşte *pixel* (cuvânt preluat din engleză, unde provine de la **pic**ture **el**ement). Aceste date pot fi numere naturale, reale sau complexe, reprezentate însă pe un număr finit de biţi. După tipul datelor din acestă structură bidimensională, imaginile prelucrate pot fi împărţite în mai multe categorii:

- imagini scalare, în care fiecare componentă este un scalar (un unic număr); ca exemple de astfel de imagini se pot da imaginile monocrome (în care punctele au doar două valori posibile, ce corespund unui conţinut binar al imaginii, în general alb-negru) şi imaginile cu nivele de gri (de tipul imaginii de luminanţă de pe ecranele televizoarelor alb-negru).
- imagini vectoriale, în care fiecare componentă este un vector de numere; cazul particular cel mai de interes este acela al imaginilor color, în care vectorul are trei elemente (ce corespund celor trei constituente de bază ale oricărei culori); în general, pentru imaginile multicomponentă, vectorul asociat fiecărui punct din imagine are mai multe elemente (caz ce corespunde imaginilor preluate în mai multe benzi de frecvenţă, aşa cum sunt imaginile de teledetecţie ale sateliţilor, imaginile de termodetecţie în benzile de infraroşu,...). Tot în categoria imaginilor vectoriale intră însă şi imaginile stereo (o pereche de imagini ale aceleiaşi scene, luate din unghiuri diferite) şi secvenţele de imagini.

Conform datelor prezentate în [11], dintre imaginile prelucrate în aplicaţii funcţionale, 20 % sunt alb-negru, 32 % sunt cu nivele de gri, 20 % sunt color, 10 % sunt imagini stereoscopice şi 18 % sunt secvenţe de imagini.

În mod clasic, valoarea unui element al unei imagini este o măsură a intensităţii luminoase în punctul considerat; acesta nu este însă decât un caz particular. După natura lor, imaginile pot fi clasificate ca imagini abstracte, imagini non-vizibile şi imagini vizibile [2]. Imaginile abstracte sau modelele sunt de fapt funcţii [matematice], continue sau discrete, de două variabile. Imaginile non-vizibile, care, evident, nu pot fi percepute în mod direct de ochiul uman, sunt de fapt achiziţii ale unor câmpuri bidimensionale de parametri fizici

(presiune, temperatură, presiune, densitate, ...). În fine, imaginile ce pot fi percepute în mod direct de către ochiul uman (deci imaginile vizibile) sunt la rândul lor imagini optice, generate ca distribuţii de intensitate luminoasă (aşa ca hologramele, imaginile de interferenţă şi difracţie) sau imagini propriu-zise (de luminanţă - în sensul curent al termenului, ce se referă la fotografii, desene, picturi, schiţe, scheme şi altele din aceeaşi categorie).

O altă împărţire a imaginilor scalare se poate face după semnificaţia ce se dă valorii numerice a pixelilor. Vom distinge astfel imagini de intensitate şi imagini indexate. O imagine de intensitate este o imagine în care valoarea fiecărui pixel este o măsură directă a intensităţii luminoase sau a mărimii fizice preluate de senzor, ca de exemplu în imaginile cu nivele de gri. Pixelii unei imagini de intensitate pot avea orice fel de valori: reale sau naturale (depinzând dacă imaginea este sau nu cuantizată).

O imagine indexată este acea imagine în care valoarea fiecărui pixel este un indice prin care se regăseşte informaţia de culoare asociată pixelului respectiv. Deci, pentru afişarea sau reprezentarea unei imagini indexate este necesară o informaţie suplimentară, de asociere între indici şi culori. Această asociere se face prin intermediul tabelei de culoare. Tabela de culoare este o matrice în care fiecare linie conţine descrierea unei culori (deci cele trei componente ce definesc culoarea - în mod tipic intensităţile relative de roşu, verde şi albastru ce compun culoarea dată printr-un amestec aditiv). Deci tabela de culoare are trei coloane; numărul de linii al tabelei de culoare este egal cu numărul de culori din imaginea reprezentată şi este în mod tipic o putere a lui doi (16, 256, ...). Indicele (valoarea pixelului) va fi numărul de ordine al liniei din tabela de culoare pe care se găseşte descrierea culorii. Este evident că valorile pixelilor unei imagini indexate nu pot fi decât numere naturale (deoarece sunt indici într-o matrice).

Această tehnică este folosită şi în grafica pe calculator. Afişarea imaginilor pe ecranul monitorului se face corespunzător unui anumit mod grafic, determinat din placa video a calculatorului. Un mod video defineşte numărul maxim de culori ce pot fi utilizate simultan şi dimensiunile ecranului (în pixeli de afişaj). Culorile utilizate la un moment dat sunt grupate într-o paletă de culori de afişare. Paleta de afişare este o structură logică definită în BGI[1] (Borland Graphics Interface), pentru programare în sesiuni de tip DOS, ca:

```
struct palettetype {
  unsigned char size;
  int colors[MAXCOLORS+1]; }
```

Modificarea unei culori din paletă (o intrare a tabelului) se face cu:

```
void far setpalette(int index_culoare, int culoare);
```

[1]Exemplele de cod C prezentate în lucrare corespund mediilor integrate Borland (Borland C++ 3.1, Turbo C 2.0)

Afişarea unui pixel cu o anumită culoare se face cu:

```
putpixel(int pozx, int pozy, int index_culoare);
```

Sub Windows, este suportată şi specificarea directă a culorii de afişat (sub forma unui triplet RGB[2]), sistemul de operare aproximând culoarea respectivă cu cea mai apropiată culoare disponibilă din paleta de lucru curentă (în acest fel, utilizatorul poate neglija existenţa acesteia).

Pentru o imagine cu nivele de gri, componentele de roşu, verde şi albastru ale fiecărei culori din paleta de culoare sunt identice. Dacă specificarea componentelor de culoare se face prin numere de 8 biţi (deci între 0 şi 255, adică cazul cel mai des folosit), tabela de culoare va avea 256 de culori (tonuri de gri) diferite. Specificarea acestora se va face cu indecşi între 0 şi 255, alocaţi conform convenţiei 0 - negru, 255 - alb. În acest fel, pentru o imagine indexată cu nivele de gri, nu mai este necesară specificarea tabelei de culoare; culorii reprezentate de indexul i îi corespunde nivelul de gri i, adică tripletul RGB (i, i, i).

Modelul imagini indexate este un caz particular de folosire a tehnicii dicţionar (sau tehnicii tabelului de echivalenţă - *Look Up Table* - LUT): o tehnică de regăsire a unei cantităţi de informaţie folosind asocierea unei chei de căutare mult mai mici.

1.2 Structura unui sistem de prelucrarea şi analiza imaginilor

Structura tipică a unui sistem de prelucrarea (evident digitală) şi analiza imaginilor este alcătuită din punct de vedere funcţional dintr-un număr mic de blocuri (vezi figura 1.1):

- sistemul de formare a imaginii (de exemplu sistemul de lentile al camerelor de luat vederi): strânge radiaţia electromagnetică a obiectului studiat pentru a forma imaginea trăsăturilor de interes
- convertorul de radiţie: converteşte radiaţia electromagnetică din planul imaginii într-un semnal electric.

Sistemul de formare a imaginii şi convertorul de radiaţie formează senzorul; acesta realizează o proiecţie plană (bidimensională) a scenei reale (care este în general tridimensională). Un studiu realizat în Germania în anul 1996 [11] prin inventarierea sistemelor de

[2]Red Green Blue - Roşu, Verde, Albastru: sistemul primar de reprezentare a culorilor.

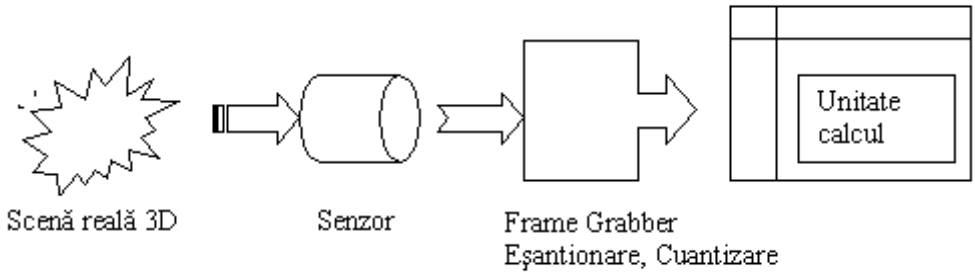


Fig. 1.1: Schema generală a unui sistem de analiza şi prelucrarea imaginilor

preluare a imaginilor folosite în industrie indică o distribuţie a tipurilor de senzori după gama de radiaţie captată conform tabelului 1.1:

Domeniu electromagnetic	Infraroşu depărtat	Infraroşu apropiat	Vizibil	Ultraviolet	Radar (microunde)	Radiaţie X
Procent	5 %	25 %	40 %	10 %	10 %	10 %

Tabel 1.1: Tipuri de senzori folosiţi în prelucrarea imaginilor

- sistemul de achiziţie (echivalent unui *frame-grabber* sau *video-blaster*): converteşte semnalul electric al senzorului într-o imagine digitală, pe care o stochează; acesta nu este altceva decât un dispozitiv de eşantionare (discretizare a suportului imaginii) şi cuantizare (discretizare a domeniului de valori a imaginii).
- sistemul de prelucrare (în mod tipic un calculator, fie el PC sau staţie de lucru); în această categorie se încadrează însă şi maşinile specializate de calcul, calculatoarele de proces, ...
- software-ul specializat: implementează algoritmii de prelucrare şi analiză

În [11] se arată că unitatea de prelucrarea hardware (deci calculatorul) folosită la aplicaţiile de prelucrarea imaginilor funcţionale la acea dată este în cea mai mare majoritate a cazurilor un PC obişnuit, cu performanţe standard; datele sunt sintetizate în tabelul 1.2:

Platformă hardware	Procent din piaţă
PC standard, bus ISA, Windows 3.1, 95, NT	40 %
Calculatoare industriale (procesoare Intel), bus VME	15 %
PC standard cu acceleratoare specializate, bus VLB, PCI	15 %
Staţii de lucru (workstations)	10 %
Maşini specializate	10 %
Calculatoare Macintosh, calculatoare paralele (transputere), altele	10 %

Tabel 1.2: Unităţi de calcul folosite în prelucrarea imaginilor

Sistemul software specializat care este responsabil cu realizarea efectivă a unei sarcini concrete poate fi descompus în mai multe module, nu neapărat bine separate şi nu neapărat prezente împreună: îmbunătăţirea, restaurarea, compresia, segmentarea şi analiza [9].

Blocul de îmbunătăţire a imaginilor are ca scop accentuarea anumitor trăsături [ale obiectelor conţinute în imagine] pentru uşurarea unor sarcini ulterioare de analiză automată sau interpretare prin afişare. Asemenea metode pot fi utile la extragerea trăsăturilor caracteristice ale obiectelor, eliminarea zgomotelor suprapuse imaginii, mărirea confortului vizual. Aceşti algoritmi nu măresc conţinutul informaţional al imaginii şi sunt în general interactivi şi puternic dependenţi de aplicaţie.

Restaurarea imaginilor se referă la eliminarea sau minimizarea efectelor unor perturbaţii şi a unor degradări. Perturbaţiile reprezintă în general zgomotele (modelate ca procese aleatoare) ce se suprapun în cursul achiziţiei imaginii (din cauza senzorului şi a lanţului de transmisiune şi captare); degradările sunt cauzate de imperfecţiunile şi limitările deterministe ale senzorului (efecte de apertură, timp de expunere, deficienţe geometrice ale sistemului de lentile, ...).

Compresia imaginilor se referă la reducerea volumului de date (numărului de biţi) cu care este reprezentată imaginea, printr-o transformare reversibilă - imaginea trebuie să poată să fie recuperată integral (sau cu diferenţe foarte mici, controlabile) din versiunea sa comprimată.

Segmentarea este procesul de descompunere a unei imagini (sau scene) în elementele (obiectele) sale constituente. Adeseori, segmentarea este strâns legată de algoritmii de analiză, al căror scop este de a realiza măsurători cantitative sau evaluări calitative asupra unor anumite categorii de obiecte, prezente în imaginea dată.

Sfera de aplicabilitate a tehnicilor de prelucrarea şi analiza imaginilor este deosebit de largă; practic, în orice domeniu de activitate se pot găsi numeroase aplicaţii. Această clasă de aplicaţii extrem de specifice a fost caracterizată drept “consumul imaginii” [1] (imaginea folosită în vederea analizei, deci a luării unor decizii). Imaginile preluate de către sateliţi pot fi folosite la descoperirea resurselor terestre, cartografiere geografică, predicţia recoltelor, urmărirea dezvoltării urbane, urmărirea vremii, controlul şi prevenirea incendi-

ilor şi inundaţiilor, precum şi alte aplicaţii ecologice şi militare. Aplicaţiile transmisiei şi compresiei imaginilor se regăsesc în televiziunea digitală, sistemele de teleconferinţă, transmisiile fax, birotică, comunicaţia pe reţele distribuite, sisteme de securitate cu circuit închis, aplicaţii militare. În aplicaţiile medicale sunt prelucrate radiografiile cu raze X, angiogramele, echografiile, tomografiile, imaginile de rezonanţă magnetică nucleară. Acestea pot fi utilizate pentru monitorizarea pacienţilor şi pentru descoperirea şi identificarea de boli şi tumori.

O largă clasă de aplicaţii sunt cele industriale, în care componentele de prelucrarea şi analiza imaginilor sunt folosite în sisteme mai mari de asigurare a calităţii produselor (metrologie, controlul calităţii - inclusiv defectoscopie nedistructivă). Soluţiile sunt extrem de specifice, puternic legate de procesul de fabricaţie urmărit şi tind să devină din ce în ce mai utilizate odată cu impunerea normelor de "calitate totală" ale standardului ISO9000 (se poate urmări [10] pentru aplicaţii specifice ale diferitelor firme germane). Din acest punct de vedere este interesantă comparaţia între câteva caracteristici ale sistemului vizual şi de prelucrare uman şi un sistem de prelucrarea şi analiza imaginilor [8], folosite pentru aplicaţii industriale, prezentată în tabelul 1.3.

Criterii	Om	Sistem de prelucrarea imaginilor
Obiectivitate	NU	DA
Control 100%	NU	DA
Rată de eroare	MARE	MICĂ
Rată de lucru	MICĂ	MARE
Rezistenţă la oboseală	MICĂ	MARE
Iluzie optică	DA	NU
Prelucrare statistică	Greu realizabil	DA
Reproductibilitate	Greu realizabil	DA
Măsurare geometrică	Cu instrumente auxiliare	DA
Recunoaştere de forme	DA	DA

Tabel 1.3: Comparaţia între caracteristici esenţiale ale sistemului vizual uman şi sistemele de prelucrarea şi analiza imaginilor

1.3 Stocarea imaginilor

Se poate considera că există două moduri de stocare a imaginilor: stocarea în memoria de lucru a unităţii de prelucrare a imaginii de lucru (care este o stocare de scurtă durată - doar pe durata prelucrării efective) şi stocarea de lungă durată imaginilor, în fişiere, pe suporturi externe de memorie (benzi, discuri, etc.). Diferenţa esenţială între cele două tipuri de stocare este aceea că în memorie imaginea va fi reprezentată complet, în formă necomprimată, pentru a permite accesul rapid direct la informaţia fiecărui pixel.

1.3.1 Stocarea imaginilor în memorie

Principalul limbaj de programare utilizat pentru aplicaţii cu calcule intensive rămâne încă limbajul C (C++). Stocarea imaginilor se va face, evident, prin intermediul unor variabile ce implementează structuri de date bidimensionale. Ceea ce este deosebit este modul de declarare a acestora: declararea statică nu este convenabilă din cauza dimensiunilor în general mari ale imaginilor, şi deci este necesară o declarare dinamică. Particularitatea este dată de memorarea separată a fiecărei linii (sau coloane) a matricii într-un vector alocat dinamic, şi gruparea adreselor de început a acestora într-un vector de pointeri, la care se va reţine adresa de început (deci un alt pointer). Dacă considerăm un tip generic de date pentru componentele matricii (caracter, sau întreg, sau real), atunci o secvenţă C de declarare a unui imagini poate fi:

```
tip ** imagine;
unsigned int contor;
imagine=(tip**) malloc(nr_linii*sizeof(tip*));
for (contor=0;contor<nr_linii;contor++)
  imagine[contor]=(tip*) malloc(nr_coloane*sizeof(tip));
```

Se remarcă folosirea constantelor *nr_linii* şi *nr_coloane* (cu semnificaţie evidentă) şi a tipului generic *tip* pentru valoarea pixelilor. Linia a 3-a alocă spaţiul pentru un masiv de pointeri la date de tip pointer; spaţiul de memorie necesar (argumentul funcţiei *malloc*) este dat de numărul de pointeri la liniile imaginii ce înmulţeşte dimensiunea unui astfel de pointer (*sizeof(tip*)*). Valoarea *imagine[contor]* este adresa de început a spaţiului de memorie la care încep valorile pixelilor de pe linia *contor*; aceştia sunt stocaţi într-un vector declarat de *malloc(nr_coloane*sizeof(tip))*. Trebuie remarcată conversia de tip (cast) obligatorie ce însoţeşte fiecare alocare de memorie (se ştie că funcţia *malloc* întoarce un pointer la *void*). De asemenea se observă că secvenţa anterioară nu face nici un fel de verificare a succesului operaţiei de alocare de memorie (verificarea faptului că valoarea returnată de funcţia *malloc* nu este *NULL*). În mod implicit, la compilare, toţi pixelii (toate valorile matricii *imagine*) sunt iniţializaţi cu 0.

Spre deosebire de C, limbajul Matlab[3] aduce mari simplificări. Există un singur tip de date, reprezentate pe 8 octeţi (caracteristică ce se schimbă începând cu versiunea 5.0, ce admite valori reale, întregi sau caracter, declarate explicit). Orice variabilă Matlab este creată în momentul folosirii sale în membrul stâng al unei expresii (deci nu este necesară declararea prealabilă folosirii); orice variabilă este o matrice (scalarul este o matrice de o linie şi o coloană). Funcţiile returnează matrici. Secvenţa C anterioară este echivalentă cu:

```
imagine=zeros(nr_linii,nr_coloane);
```

[3] Codurile Matlab prezentate în lucrare corespund versiunii Matlab 4.2c.

1.3.2 Stocarea imaginilor în fişiere

Un fişier este entitatea logică de organizare a informaţiei înscrise pe mediile magnetice de stocare şi se compune dintr-un şir de octeţi. Pentru stocarea imaginii este necesar ca aceşti octeţi să conţină informaţia aferentă pixelilor precum şi informaţie despre tipul imaginii: dimensiunile acesteia, dacă este sau nu indexată, dacă are sau nu o tabelă de culoare ataşată, dacă este sau nu comprimată şi după ce metodă. Anumite structuri de fişiere au fost impuse de-a lungul timpului de firme producătoare de software sau de organisme de standardizare, căpătând denumirea de formate de imagini. Formatele de imagini s-au făcut cunoscute mai ales după extensia standard a fişierelor ce conţin imaginile stocate după formatul respectiv: BMP, TIF, GIF, PCX, JPG ... . În cele ce urmează ne vom referi la formatele RAW(cunoscut şi ca IMG), unul dintre cele mai rudimentare formate de fişiere imagine, şi Windows Bitmap -BMP al firmei Microsoft, care este unul dintre cele mai larg recunoscute formate de fişiere.

Un fişier RAW conţine imagini indexate cu nivele de gri, de formă pătrată. Fişierul nu are antet (dimensiunile imaginii fiind deduse din dimensiunea fişierului ce o conţine) şi nu conţine nici tabel de culoare (acesta are toate componentele liniei i egale între ele, reprezentând griuri). Fiecare pixel al imaginii este codat cu numărul corespunzător de biţi (4, 8, etc.); imaginea este baleiată în ordinea normală (începând cu prima linie a imaginii, de la stânga la dreapta).

Un fişier BMP[4] are trei componente consecutive: un antet de fişier (BITMAPFILEHEADER), o structură de informaţie a imaginii(BITMAPINFO) şi codarea pixelilor. Antetul de fişier (BITMAPFILEHEADER) conţine informaţii asupra tipului, dimensiunii şi reprezentării fişierului Bitmap independent de dispozitiv (DIB - Device Independent Bitmap); semnificaţiile componentelor sunt date în tabelul 1.4.

```
typedef struct tagBITMAPFILEHEADER{
WORD bfType;
DWORD bfType;
WORD bfReserved1;
WORD bfReserved2;
DW bfOffBits;
}BITMAPFILEHEADER;
```

Structura de informaţie a imaginii (BITMAPINFO) conţine informaţii asupra dimensiunilor şi culorilor unui DIB, şi este alcătuită din două componente: antetul structurii de informaţii (BITMAPINFOHEADER), a cărui componente sunt descrise în tabelul 1.5 şi tabelul de culoare, format din structuri RGBQUAD.

[4]Denumirile componentelor logice ale fişierului sunt cele standardizate de Microsoft.

Câmp	Descriere
bfType	Specifică tipul de fişier; trebuie să conţină caracterele BM
bfSize	Specifică lungimea fişierului în DWORD
bfReserved1	Câmp rezervat, valoare 0
bfReserved2	Câmp rezervat, valoare 0
bfOffBits	Specifică deplasamentul în octeţi de la sfârştul structurii BITMAPFILEHEADER până la zona din fişier ce conţine pixelii codaţi

Tabel 1.4: Descrierea câmpurilor structurii BITMAPFILEHEADER

Câmp	Descriere
biSize	Numărul de octeţi ai structurii BITMAPINFOHEADER
biWidth	Lăţimea imaginii, în pixeli
biHeight	Înălţimea imaginii, în pixeli
biPlanes	Numărul de plane de culoare ale dispozitivului de afişaj (1)
biBitCount	Numărul de biţi cu care se codează un pixel; poate fi 1, 4, 8 sau 24
biCompression	Tipul de compresie utilizată: BI_RGB fără compresie, BI_RLE8 sau BI_RLE4 pentru compresie de tip RLE cu cuvinte de respectiv 8 sau 4 biţi
biSizeImage	Dimensiunea imaginii în octeţi
biXPelsPerMeter	Rezoluţia pe orizontală a dispozitivului ţintă (în pixeli pe metru)
biYPelsPerMeter	Rezoluţia pe verticală a dispozitivului ţintă (în pixeli pe metru)
biClrUsed	Numărul de culori utilizate în imagine; dacă este 0, imaginea foloseşte toate culorile disponibile ale paletei
biClrImportant	Numărul de culori considerate importante; dacă este 0, toate culorile sunt luate în considerare

Tabel 1.5: Descrierea câmpurilor structurii BITMAPINFOHEADER

```
typedef struct tagBITMAPINFOHEADER{
DWORD biSize;
DWORD biWidth;
DWORD biHeight;
WORD biPlanes;
WORD biBitCount;
DWORD biCompression;
DWORD biSizeImage;
DWORD biXPelsPerMeter;
DWORD biYPelsPerMeter;
DWORD biClrUsed;
DWORD biClrImportant;
} BITMAPINFOHEADER;
```

Structura RGBQUAD descrie o culoare prin componentele sale de roşu, verde şi albastru, şi un câmp rezervat având valoarea 0.

```
typedef struct tagRGBQUAD{
BYTE rgbBlue;
BYTE rgbGreen;
BYTE rgbRed;
BYTE rgbReserved;
}RGBQUAD;
```

Codarea pixelilor se face după câteva reguli. Fiecare pixel va fi codat pe biBitCount biţi; dacă biBitCount este 1, 4 sau 8, imaginea va fi indexată şi fişierul conţine tabela de culoare asociată imaginii. Codurile pixelilor se grupează pe octeţi (deci pentru o codare de 4 biţi per pixel, fiecare octet de cod va corespunde la doi pixeli alăturaţi). Dacă biBitCount este 24, pentru fiecare pixel se asociază direct trei octeţi, ce reprezintă componentele de roşu, verde şi albastru ale culorii respective; această imagine se numeşte *True Color* şi nu mai are un tabel de culoare asociat. Denumirea de *True Color* (culoare adevărată) provine din faptul că numărul total de culori ce se pot astfel reprezenta (2^{24}) depăşeşte limita sensibilităţii umane de discernere a culorilor.

Codarea se face independent pe fiecare linie orizontală a imaginii. Codurile (indexurile) tuturor pixelilor unei linii sunt concatenate; şirul rezultat trebuie să fie multiplu de 32 de biţi (sau de 4 octeţi, sau să conţină un număr întreg de DWORD's). Dacă acestă constrângere nu este respectată, linia respectivă se completează cu numărul necesar de biţi (care, în mod evident, nu vor fi utilizaţi la citirea imaginii din fişier). Codarea imaginii începe cu ultima linie, şi pentru fiecare linie baleiajul este normal (de la stânga la dreapta).

Capitolul 2

TEHNICI DE ÎMBUNĂTĂŢIRE A IMAGINILOR

Îmbunătăţirea imaginilor este o sintagmă generală ce se referă la o clasă largă de operaţii al căror scop este mărirea detectabilităţii componentelor imaginii. Detectabilitatea componentelor este legată mai mult de percepţia vizuală a unui observator uman decât de o analiză automată cantitativă. Percepţia vizuală de referinţă este cea a unui expert uman în domeniul aplicaţiei din care provine imaginea.

Aşadar criteriile de evaluare ale calităţii unei imagini sunt subiective şi specifice aplicaţiei. În [2] se face analogia operaţiilor de îmbunătăţire a imaginilor cu reglajul tonalităţii muzicii ascultate; în funcţie de ascultător, se vor favoriza componentele înalte sau joase, sau nici unele. Ca o consecinţă, procesul de îmbunătăţire va fi interactiv, transformările efectuate trebuind să fie validate (cel puţin în etapa de proiectare sau probă) de către un utilizator uman.

Principiul (aproape unanim acceptat) este că îmbunătăţirea calităţii unei imagini se face fără a lua în considerare nici o informaţie asupra imaginii originale sau asupra procesului de degradare (prin care imaginea nu este suficient de bună). Conform acestui punct de vedere chiar şi o imagine originală (nedegradată) poate fi îmbunătăţită, obţinând o imagine falsificată, dar subiectiv preferabilă [18]. În general, calitatea subiectivă a unei imagini poate fi apreciată pe baza contrastului sau accentuării elementelor de contur (muchii, frontiere, linii, margini) şi pe baza netezimii în regiunile uniforme.

Creşterea uniformităţii regiunilor este însă asimilată eliminării unui eventual zgomot suprapus imaginii, operaţie denumită în mod clasic filtrare. Filtrarea ce are ca scop eliminarea zgomotului va fi studiată în următoarele capitole.

Din punctul de vedere al metodelor utilizate, putem distinge mai multe tipuri de operaţii de îmbunătăţire:

- operaţii punctuale, prin care se realizează o corespondenţă de tip "unu la unu" între vechea valoare a nivelului de gri şi noua valoare a acestuia, pentru fiecare pixel al imaginii. Tot în acestă categorie vom include şi operaţiile de pseudocolorare, care se referă la afişarea imaginii folosind o paletă de culoare modificată.
- operaţii locale (sau de vecinătate), prin care noua valoare a nivelului de gri într-un pixel este obţinută din vechea valoare a pixelului repectiv şi din valorile unor pixeli vecini pixelului considerat.
- operaţii integrale, în care noua valoare a unui pixel este dependentă de valorile tuturor pixelilor imaginii

În acest capitol vom studia doar operaţiile punctuale şi de pseudocolorare.

Prin îmbunătăţire, unei imagini nu i se adaugă nici o informaţie nouă faţă de cea ce exista iniţial [9] (deci nu se adaugă nimic imaginii), ci doar este prezentat altfel conţinutul iniţial al acesteia. Deşi la o examinare superficială afirmaţia este corectă, putem găsi măcar două obiecţii (sau contraexemple) la această formulare:

- din punctul de vedere al utilizatorului, informaţia, chiar dacă există, nu poate fi folosită, deci este asimilabil nulă. Acesta este cazul imaginilor obţinute în condiţii extreme de iluminare, ce prezintă un contrast foarte slab (imagini subexpuse sau supraexpuse) [5].
- din punctul de vedere al teoriei informaţiei, informaţia din imagine poate fi asimilată entropiei procesului aleator ale cărui realizări particulare sunt valorile de gri ale pixelilor. Entropia se modifică însă la orice transformare ce afectează distribuţia nivelelor de gri din imagine.

2.1 Operaţii punctuale de modificare a contrastului

Operaţiile punctuale de modificare a contrastului (numite şi transformări ale nivelului de gri) sunt asocieri (*mapping*, în engleză) ce leagă nivelul de gri original de noua sa valoare. O asemenea asociere nu este altceva decât o funcţie:

$$v = T(u),\ u \in [0; L-1] \tag{2.1}$$

În [5] se stabilesc ca necesare condiţiile ca:

- transformarea T să păstreze gama admisibilă de valori ale imaginii (dacă nivelele de gri au fost reprezentate pe L nivele de cuantizare, atunci $0 \leqslant T(u) \leqslant L-1$, $\forall u \in [0; L-1]$)

- transformarea T să fie monotonă (crescătoare sau descrescătoare) pentru a păstra ordinea între nivelele de gri

2.1.1 Modificarea liniară a contrastului

Cea mai des folosită tehnică de modificare liniară a contrastului este o transformare liniară pe porţiuni, dată de:

$$v = \begin{cases} \frac{\alpha}{T_1} u, \, 0 \leq u < T_1 \\ \alpha + \frac{\beta - \alpha}{T_2 - T_1} \left(u - T_1 \right), \, T_1 \leq u < T_2 \\ \beta + \frac{L-1-\beta}{L-1-T_2} \left(u - T_2 \right), \, T_2 \leq u < L \end{cases} \tag{2.2}$$

În formula anterioară, parametrii de control sunt T_1, T_2, α şi β; aceştia sunt grupaţi câte doi, definind punctele (T_1,α) şi (T_2,β). Aceste două puncte de control, împreună cu punctele fixe $(0, 0)$ şi $(L-1, L-1)$ vor defini cele trei segmente de dreaptă ce apar în formula (2.2). Rezultatul aplicării unei asemenea operaţii punctuale se obţine modificând valoarea (nivelul de gri) fiecărui pixel al imaginii iniţiale, u, conform (2.2), obţinând noul nivel de gri v. Transformarea poate fi făcută în două moduri: fie se repetă calculele de la (2.2) pentru fiecare pixel, baleind imaginea, fie noile valori ale contrastului se calculează de la început pentru toate nivelele de gri posibile (între 0 şi $L-1$) şi apoi aceste modificări se aplică imaginii. Codul C următor implementează a doua variantă de calcul, care este mai rapidă (calculele nivelelor de gri au fost separate de ciclul de baleiere al imaginii şi au fost eliminate structurile condiţionale *if* - impuse de definiţia de tip "acoladă" - prin separarea domeniilor de calcul în trei cicluri). Trebuie remarcată de asemenea definirea nivelului de gri ca *unsigned int*, ceea ce crează posibilitatea folosirii unui număr de nivele de gri mai mare de 256 (pentru care ar fi fost suficient un *unsigned char*).

```
unsigned int T1,T2,alfa,beta,gri_vechi,i,j;
gri_nou=(unsigned int *) malloc (L*sizeof(unsigned int));
for (gri_vechi=0;gri_vechi<T1;gri_vechi++)
  gri_nou[gri_vechi]=alfa*gri_vechi/T1;
for (gri_vechi=T1;gri_vechi<T2;gri_vechi++)
  gri_nou[gri_vechi]=alfa+(beta-alfa)*(gri_vechi-T2)/(T2-T1);
for (gri_vechi=T2;gri_vechi<L;gri_vechi++)
  gri_nou[gri_vechi]=beta+(L-1-beta)*(gri_vechi-T1)/(L-1-T2);
for (i=0;i<NRLIN;i++)
  for (j=0;j<NRCOL;j++)
    imagine_noua[i][j]=gri_nou[imagine_veche[i][j]];
```

Vom prezenta în cele ce urmează un exemplu; în figura 2.1 este prezentată pe de o parte imaginea originală "lena" (una dintre fotografiile impuse ca standard de fapt în raportarea rezultatelor obţinute), iar alăturat imaginea obţinută cu o modificare liniară pe porţiuni

a contrastului, determinată de parametrii $T_1 = 30$, $T_2 = 100$, $\alpha = 20$, $\beta = 200$. În figura 2.2 este prezentată funcţia de transformare a nivelelor de gri ($T(u)$).

Fig. 2.1: Imagine originală şi imagine cu contrastul modificat

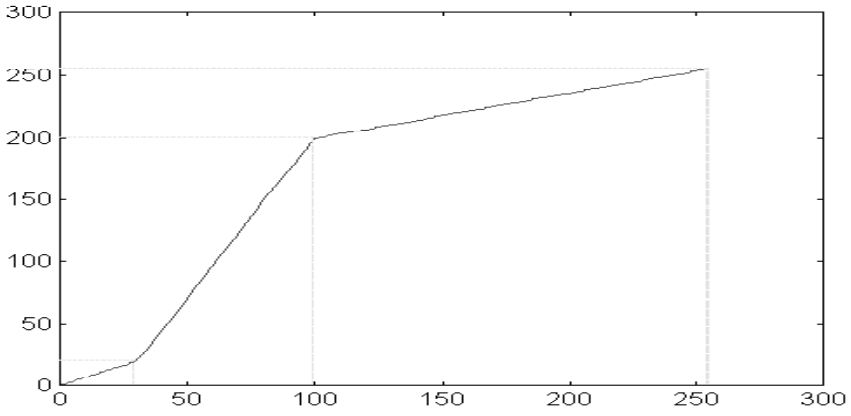


Fig. 2.2: Îmbunătăţire liniară pe porţiuni, definită de punctele (0,0), (30,20), (100,200), (255,255)

Vizibilitatea componentelor scenei este în general determinată în cea mai mare parte de contrastul zonei din imagine; contrastul este o măsură proporţională cu diferenţa dintre luminozitatea anumitor pixeli (nivelul lor de gri). Pentru a putea prevedea deci efectele unei operaţii de îmbunătăţire de tipul prezentat asupra contrastului este deci suficientă studierea diferenţelor de nivele de gri între o aceeaşi pereche de pixeli înainte şi după efectuarea transformării. La limită, este posibil ca pixelii să aibă nivelul de gri original diferit cu doar o unitate (cuanta minimă), şi atunci modificarea contrastului va fi dată

de diferenţa valorilor transformate, adică de derivata funcţiei de transformare:

$$C = \frac{\Delta v}{\Delta u} = \frac{T(u_2) - T(u_1)}{u_2 - u_1} = \frac{dT(u)}{du} = T'(u) \tag{2.3}$$

Pentru funcţia liniară pe porţiuni este evident că derivata va fi constantă pe aceleaşi intervale, având valoarea egală cu panta segmentului de dreaptă.

$$C = \begin{cases} \frac{\alpha}{T_1}, \text{ dacă } u \in [0; T_1] \\ \frac{\beta-\alpha}{T_2-T_1}, \text{dacă } u \in [T_1; T_2] \\ \frac{L-1-\beta}{L-1-T_2}, \text{ dacă } u \in [T_2; L-1] \end{cases} \tag{2.4}$$

Dacă pe un interval această pantă este subunitară, atunci diferenţa între nivelele alăturate de gri se micşorează şi deci contrastul scade; dacă din contră, panta dreptei este supraunitară, diferenţa dintre nivelele de gri alăturate se măreşte şi contrastul va creşte. Spre exemplu, în transformarea din figura 2.2, pe intervalul [30,100] contrastul creşte, iar pe intervalele [0,30] şi [100,255] contrastul scade. Aceste efecte sunt uşor vizibile pe imaginile din figura 2.1: de exemplu scăderea contrastului pentru nivelele de gri de la capătul superior al gamei admise (dinspre alb) se observă prin dispariţia detaliilor luminoase din imagine (panglica lată de la pălărie); creşterea contrastului pe intervalul [30,100] (deci griuri închise) este vizibil în zona penei de pălărie, ale cărei detalii sunt acum mult mai sesizabile.

În funcţie de alegerea celor patru parametri, se pot obţine câteva cazuri particulare de interes ce poartă denumiri specifice.

Fig. 2.3: Imagine originală şi imagine binarizată cu pragul 125

Dacă $T1 = T2$ şi $\alpha = 0$, $\beta = L - 1$, se obţine prăguirea sau binarizarea ("thresholding") (vezi figura 2.4); în imaginea rezultată nu există decât alb şi negru (figura 2.3); toate

nivelele de gri iniţiale a căror valoare era mai mică decât T_1 fiind negre şi toate nivelele de gri iniţiale mai mari ca T_1 devenind albe. După cum se va vedea la capitolul de segmentare orientată pe regiuni (capitolul 8), aceasta este şi una dintre tehnicile cele mai simple de segmentare. În urma acestei transformări, contrastul este maximizat la nivelul întregii imagini.

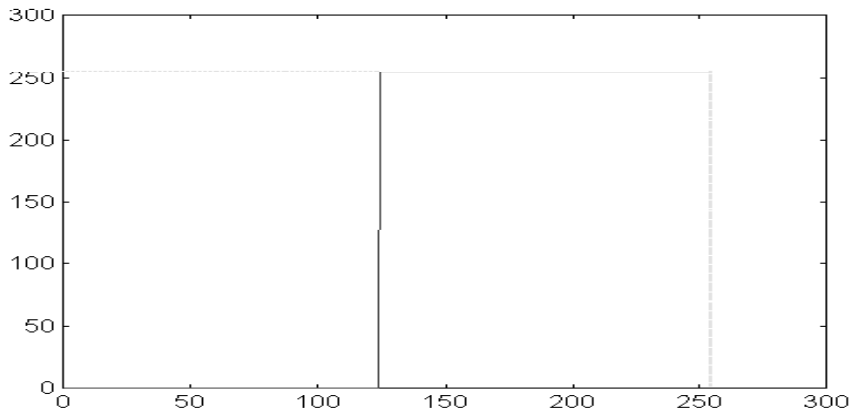


Fig. 2.4: Transformarea de binarizare

Dacă $\alpha = 0$ şi $\beta = L - 1$ se obţine operaţia de întindere maximă a contrastului (*contrast streching*) (vezi figura 2.5) pentru intervalul $[T_1; T_2]$. Nivelele de gri care se găsesc în afara acestui interval vor fi înlocuite fie cu alb, fie cu negru.

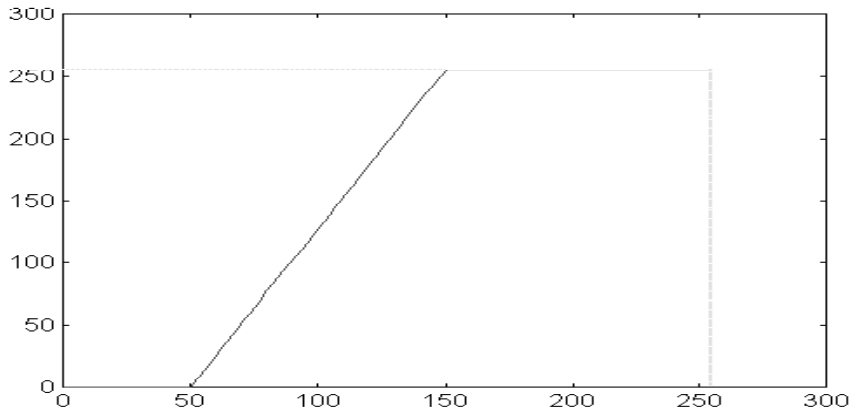


Fig. 2.5: Transformarea de întindere maximă a contrastului

2.1.2 Modificarea neliniară a contrastului

Principalul dezavantaj al tehnicii liniare pe porţiuni prezentate este faptul că modificarea contrastului este aceeaşi pe un întreg interval de nivele de gri, şi nu este posibilă o modificare neuniformă a contrastului pe întregul interval de nivele de gri sau în jurul unui anume nivel de gri. Tehnicile neliniare au aceste proprietăţi.

O primă variantă este compandarea domeniului [9], definită de o curbă logaritmică şi cu punctele fixe $(0,0)$ şi $(L-1, L-1)$:

$$v = T(u) = \frac{L-1}{\lg L} \lg(1+u) \tag{2.5}$$

Contrastul va varia neuniform de-a lungul scalei de gri, mărindu-se la capătul inferior (negru) şi micşorându-se la capătul superior (alb). În mod reciproc se poate defini expandarea domeniului, ca transformare inversă celei de compandare, şi deci având o alură exponenţială:

$$v = T(u) = (L-1)\frac{e^u - 1}{e^{L-1} - 1} \tag{2.6}$$

Contrastul va varia neuniform de-a lungul scalei de gri, mărindu-se la capătul superior (alb) şi micşorându-se la capătul inferior (negru). Termenii de compandare şi de expandare au fost daţi prin asemănare cu transformările folosite în teoria codării şi cuantizării (ce intervin în metodele de cuantizare a semnalului vocal pentru telefonia digitală, cunoscute sub numele de legea A în Europa şi legea μ în America). Trebuie însă subliniat că în prelucrarea imaginilor aceste transformări nu afectează domeniul de valori, care rămâne $[0, L-1]$.

Alte transformări neliniare pot fi obţinute prin folosirea unor funcţii de tip putere; şi acestea au nivelele de gri extreme ca puncte fixe ($(0,0)$ şi $(L-1, L-1)$). O primă variantă este funcţia putere:

$$v = T(u) = (L-1)\left(\frac{u}{L-1}\right)^r \tag{2.7}$$

După valorile parametrului-putere r se pot obţine două comportări diferite: pentru $r < 1$ comportarea este de acelaşi tip cu al funcţiei de compandare logaritmice, iar pentru $r > 1$ comportarea este de tipul funcţiei de expandare. Trebuie remarcat că legile de variaţie ale contrastului vor fi însă diferite.

Există însă şi o variantă la care se mai adaugă un punct fix (T, T), funcţia devenind cu două intervale de definiţie:

$$v = T(u) = \begin{cases} T\left(\frac{u}{T}\right)^r, \text{ dacă } u \in [0;T] \\ L-1-(L-1-T)\left(\frac{L-1-u}{L-1-T}\right)^r, \text{ dacă u} \in [T, L-1] \end{cases} \tag{2.8}$$

Funcţia are o alură de tipul celei prezentate în figura 2.6.

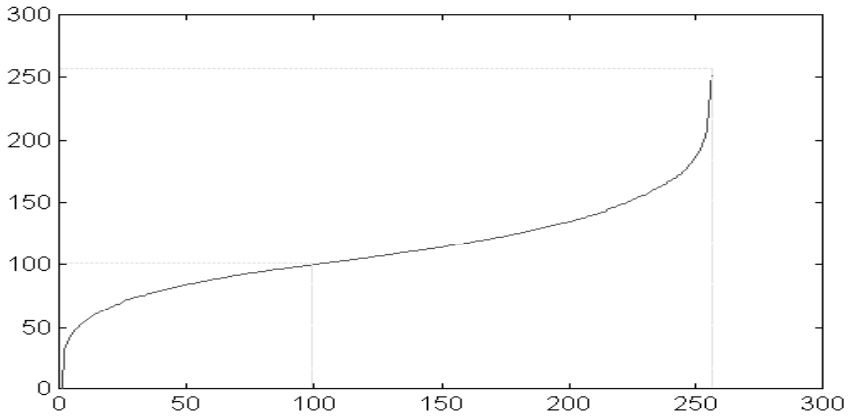


Fig. 2.6: Modificare neliniară a contrastului, cu trei puncte fixe

În [9], în cadrul operaţiilor punctuale de îmbunătăţire a imaginilor sunt prezentate şi operaţii aritmetice simple, ca de exemplu negativarea. Negativarea este descrisă de:

$$v = T(u) = L - 1 - u \tag{2.9}$$

Efectul acesteia de modificare a contrastului se bazează doar pe caracteristicile sistemului vizual uman, pentru care contrastul depinde de diferenţa de luminozitate între pixeli aparţinând unui obiect, respectiv fundalului, raportată la luminanţa medie a fundalului. O categorie aparte de aplicaţii în care sunt utile asemenea inversiuni este analiza imaginilor medicale, care pentru o analiză automată trebuiesc inversate; un astfel de exemplu este imaginea angiografică din figura 2.7.

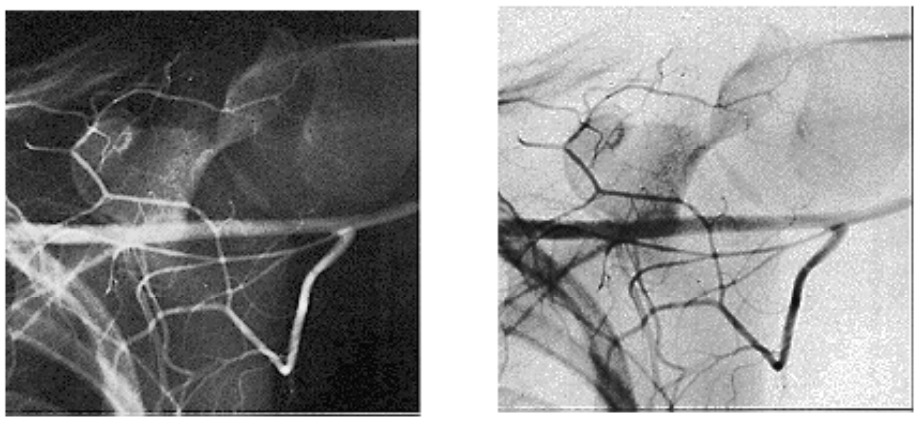


Fig. 2.7: Imagine originală şi negativată (dintr-o aplicaţie medicală)

Alte operaţii posibile (tratate în [9], dar de mai mică semnificaţie practică) sunt repre-

zentarea planelor de bit (pentru fiecare pixel al imaginii, fiecare bit al reprezentării binare a nivelului de gri este considerat ca valoarea unui pixel al unei imagini binare), transformarea de lipire *clipping* (care păstrează nemodificate nivelele de gri dintr-un anumit interval şi le anulează pe celelalte - imaginea rezultată conţinând obiectele de interes pe fond negru) şi transformarea de tăiere *slicing* (care transformă nivelele de gri dintr-un interval fixat în alb şi tot restul în negru; nu este altceva decât un alt tip de binarizare).

2.2 Pseudocolorarea

Pseudocolorarea este o tehnică de îmbunătăţire a vizibilităţii anumitor componente ale imaginii (sau a imaginii în ansamblu) prin modificarea paletei de culoare cu care imaginea este afişată (reprezentată). Aceasta înseamnă că pentru anumite nivele de gri, afişarea nu se va mai face cu culoarea a cărei componente sunt toate egale cu indexul (nivelul de gri), ci cu o altă culoare. Acestă definiţie acoperă însă şi cazul operaţiilor de modificare a contrastului prezentate anterior; funcţia $v = T(u)$ nu este altceva decât o funcţie de construcţie a unei noi palete de culoare pentru aceeaşi imagine. Spre exemplu, codul modificării neliniare de contrast devine:

```
unsigned int T1,T2,alfa,beta,gri_vechi;
gri_nou=(unsigned int *) malloc (L*sizeof(unsigned int));
for (gri_vechi=0;gri_vechi<T1;gri_vechi++)
  gri_nou[gri_vechi]=alfa*gri_vechi/T1;
for (gri_vechi=T1;gri_vechi<T2;gri_vechi++)
  gri_nou[gri_vechi]=alfa+(beta-alfa)*(gri_vechi-T2)/(T2-T1);
for (gri_vechi=T2;gri_vechi<L;gri_vechi++)
  gri_nou[gri_vechi]=beta+(L-1-beta)*(gri_vechi-T1)/(L-1-T2);
for (gri_vechi=0;gri_vechi<L;gri_vechi++)
  setpalette(gri_vechi,(color) gri_nou[gri_vechi]);
```

Ultima buclă *for* a codului modifică paleta de culoare pentru fiecare index (intrare) a acesteia, conform noilor valori calculate. Remarcaţi cast-ul (schimbarea de tip) la tipul *color*; acesta este inserat mai mult ca o măsură de atenţionare: culoarea (deci variabila de tip *color)* trebuie obţinută din scalarul nivel de gri în conformitate cu regulile de descriere a culorilor valabile în respectivul mod grafic.

Ideea de bază în pseudocolorare este de a folosi culori pentru a pune în evidenţă zone de interes din imagini cu nivele de gri (există şi varianta colorării false - *false coloring*, care transformă o imagine color într-o altă imagine color, dar cu un contrast mult mai pronunţat şi artificial între elementele sale). Acestă idee este normală dacă se are în vedere faptul că ochiul (sistemul vizual) uman distinge ceva mai puţin de 256 nuanţe de gri, deşi diferenţiază câteva milioane de culori [9].

O aplicaţie interesantă a pseudocolorării este prezentată în [2]. La NASA, la începuturile erei digitale în tehnicile de achiziţie a imaginilor, era necesară digitizarea unor imagini de microscopie, pentru care iluminarea era reglată manual, până la o vizibilitate maximă a tuturor detaliilor. Pentru că operatorul nu putea distinge clar şi precis care era iluminarea cea mai potrivită (care nici nu producea suprailuminare şi nici nu lăsa umbre mai mari decât era necesar), la afişarea în timp real a imaginii achiziţionate pe monitoarele de control, paleta de griuri a fost modificată pe ultima poziţie (alb), unde s-a inserat roşu. Atunci instrucţiunile pentru operator erau: creşte curentul prin lampă (iluminarea) până când în imagine apare roşu, după care redu curentul până când culoarea roşie dispare. Rezultatul acestei tehnici simple au fost mii de imagini digitizate corect.

În final merită poate amintită remarca (destul de acidă) din [2]:

> "Deşi prin natura sa este un detaliu al tehnicilor de afişare, pseudocolorarea a fost adesea glorificată prin termeni ca prelucrare prin pseudocolorare sau analiză prin pseudocolorare. Pseudocolorarea rămâne un accesoriu favorit al vânzătorilor, care o utilizează adesea în demonstraţiile produselor [software], deoarece poate stârni interesul în ochii clienţilor mult mai repede decât orice altă metodă de afişare cunoscută. Cercetările mele au adus la lumină o listă dureros de scurtă a aplicaţiilor demonstabil productive a pseudocolorării"

2.3 Operaţii de contrastare bazate pe histograma imaginii

Pentru o imagine f de $M \times N$ pixeli şi L nivele de gri, histograma este definită (2.10) ca probabilitatea (frecvenţă relativă) de apariţie în imagine a diferitelor nivele de gri posibile.

$$h(i) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \delta(i - f(m,n)) \quad , \quad i = 0, 1, ...L-1 \tag{2.10}$$

Din punct de vedere statistic, putem considera valoarea fiecărui pixel al imaginii ca o realizare particulară a unei variabile aleatoare asociată nivelelor de gri, caz în care histograma (2.10) este funcţia de densitate de probabilitate a acestei variabile aleatoare. Fiind o funcţie de densitate de probabilitate, histograma oricărei imagini verifică condiţia de normare $\sum_{i=0}^{L-1} h(i) = 1$.

Din punct de vedere practic, calculul histogramei unei imagini înseamnă parcurgerea punct cu punct a imaginii şi contorizarea numărului de nivele de gri întâlnite. Presupunând L nivele de gri posibile în imaginea de dimensiuni $NRLIN$ şi $NRCOL$, codul

următor alocă pentru fiecare nivel de gri posibil câte un *unsigned int*, ce va contoriza numărul de apariţii ale nivelului de gri respectiv. Pentru fiecare punct al imaginii se incrementează poziţia din histogramă ce corespunde valorii de gri din acel pixel. Ceea ce rezultă în final diferă de histograma descrisă de (2.10) prin constanta de normare "numărul total de puncte ale imaginii" (deci MN sau *NRLIN*NRCOL*); este evident că valorile obţinute pot fi normate pentru a obţine o funcţie de densitate de probabilitate prin împărţirea cu *NRLIN*NRCOL* şi definirea histogramei ca fiind formată din *real* sau *float*.

```
histo=(unsigned int*)malloc(L*sizeof (unsigned int));
for (i=0;i<L;i++)
  histo[i]=0;
for (i=0;i<NRLIN;i++)
  for (j=0;j<NRCOL;j++)
    histo[imagine[i][j]]++;
```

În Matlab, implementarea oricărei funcţii este cu atât mai eficientă (din punctul de vedere al timpului de rulare) cu cât sunt evitate structurile repetitive (în particular buclele *for*). Cum, pentru calculul histogramei, ciclarea nu poate fi evitată (deci trebuie parcurse fie toate punctele imaginii, fie toate nivelele de gri), se alege varianta care implică repetarea minimă a ciclării:

```
histo=zeros(1,L);
for i=1:L
  p=find(imagine==i);
  histo(i)=length(p);
% histo(i)=length(p)/prod(size(imagine));
end
```

Funcţia *find* (funcţie standard a pachetului Matlab) returnează poziţiile (indicii) la care este găsită valoarea i în matricea *imagine* (adică întoarce poziţiile punctelor ce au valoarea i); numărând aceste puncte (deci calculând lungimea vectorului în care sunt stocate) se găseşte numărul de puncte ce au respectivul nivel de gri. Normarea histogramei se poate face fără a modifica declaraţiile de definire a structurilor (pentru Matlab este indiferent dacă se stochează întregi sau numere cu parte zecimală). Acestă structură este mai rapidă decât cea prin care se parcurgea toată imaginea deoarece se foloseşte o singură buclă de lungime L (şi nu două bucle imbricate, de lungime totală *NRLIN*NRCOL),* iar funcţia *find* este rapidă, fiind o funcţie precompilată.

Histograma imaginii oferă informaţii asupra plasamentului în "nuanţă" a conţinutului imaginii (vezi figura 2.8). La majoritatea imaginilor există o distribuţie neuniformă a nivelelor de gri; există nivele de gri predominante şi există nivele de gri folosite puţin sau

deloc. Operaţiile de îmbunătăţire a imaginilor (pentru îmbunătăţirea percepţiei vizuale) au ca scop redistribuirea nivelelor de gri, astfel ca acestea să ocupe întreaga gamă de variaţie disponibilă, în mod uniform: aceasta este egalizarea de histogramă [9], [18].

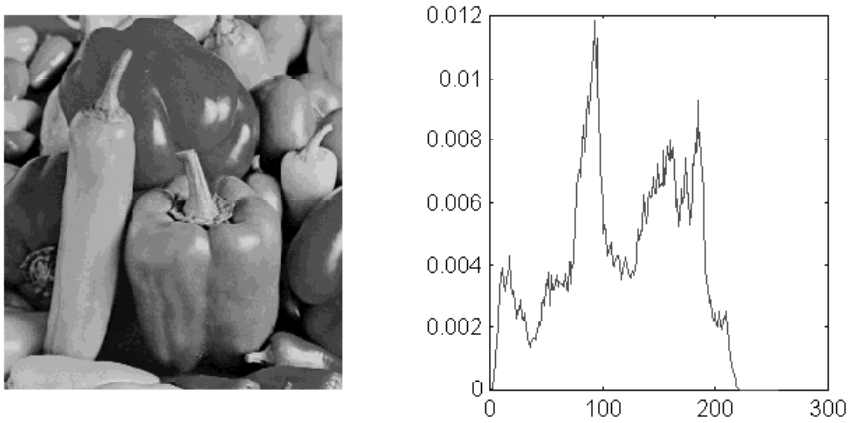


Fig. 2.8: Imagine cu nivele de gri şi histograma acesteia

Scopul egalizării de histogramă este deci obţinerea unei distribuţii uniforme a nivelelor de gri; imaginea rezultată va prezenta cea mai mare îmbunătăţire a contrastului, distribuit regulat în întreaga gamă dinamică a nivelelor de gri. Din punct de vedere matematic, egalizarea de histogramă înseamnă transformarea unei distribuţii oarecari (descrisă de histograma imaginii iniţiale) într-o distribuţie uniformă. Considerând variabilele aleatoare $X(\xi, x)$ şi $Y(\xi, y)$ legate prin $Y = g(X)$, atunci între funcţiile de densitate de probabilitate a celor două variabile aleatoare există relaţia [16]:

$$f_Y(y) = f_X(x) \frac{1}{\left|(g^{-1}(y))'\right|}|_{x=g^{-1}(y)}$$

Dacă dorim ca funcţia de densitate de probabilitate $f_Y(y)$ să fie uniformă (în condiţiile în care funcţia de densitate de probabilitate $f_X(x)$ este dată), atunci înseamnă că vom avea $\left|(g^{-1}(y))'\right| = \frac{1}{k} f_X\left(g^{-1}(y)\right)$. Rezolvarea acestei ecuaţii produce soluţia $y = g(x) = \int_{-\infty}^{x} f_X(t)dt$.

Pentru cazul particular al imaginilor, variabila aleatoare X ia valori naturale - nivelele de gri. Funcţia de densitate de probabilitate $f_X(x)$ este histograma [normată] a imaginii iar funcţia de transformare devine $y = g(x) = \int_{0}^{x} f_X(t)dt$. Ţinând seama că valorile de gri sunt discrete, integrala se transformă în sumă şi acestă formă nu este altceva decât

histograma cumulativă a imaginii; dacă h este histograma imaginii, atunci

$$g(x) = H(x) = \sum_{i=0}^{x} h(i) \tag{2.11}$$

Valorile funcţiei trebuie însă redistribuite în intervalul permis de valori de gri, ceea ce duce la deducerea formulei care exprimă noile valori de gri:

$$v = \left[\frac{H(u) - H(0)}{MN - H(0)} (L-1) + 0.5 \right] \tag{2.12}$$

O variantă de cod care realizează egalizarea de histogramă este prezentată în continuare. Trebuie remarcat că modul de calcul al histogramei cumulative este de tip iterativ, bazându-se pe formula $H(x) = H(x-1) + h(x)$, ce se poate deduce imediat din (2.11). Mai trebuie de asemenea remarcat că se foloseşte o histogramă nenormată.

```
gri_nou=(unsigned int *) malloc (L*sizeof(unsigned int));
histo_cum=(unsigned int *) malloc (L*sizeof(unsigned int));
tot=NRLIN*NRCOL;
histo_cum[0]=histo[0];
for (i=1;i<L;i++)
  histo_cum[i]=histo_cum[i-1]+histo[i];
for (i=0;i<L;i++)
  gri_nou[i]=round((histo_cum[i]-histo_cum[0])*(L-1)/(tot-histo_cum[0]));
```

Figurile următoare prezintă o imagine originală ("lena") şi rezultatul egalizării de histogramă, precum şi histogramele originale şi egalizate (figura 2.9). Ceea ce se remarcă cu uşurinţă este diferenţa dintre histograma obţinută şi histograma perfect uniformă dorită. Unul dintre efectele secundare notabile este introducerea de nivele de gri lipsă în histograma egalizată (aspectul "în pieptene" al acesteia).

Imagine originală

Imagine după egalizarea de histogramă

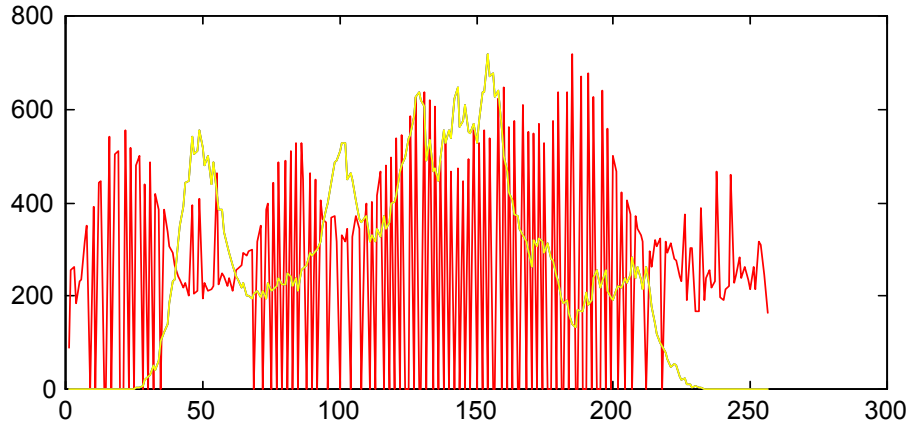


Fig. 2.9: Histograma imaginii "lena", înainte şi după egalizare

Alte variante de egalizare a histogramei [18] folosesc histograme modificate ale imaginii (prin contabilizarea doar a anumitor pixeli) sau limitează amplitudinea vârfurilor histogramei, redistribuind uniform pixelii în exces.

Tehnica de egalizare a histogramei poate fi extinsă prin realizarea unei histograme de formă impusă (dar oarecare) a imaginii rezultat; această metodă este denumită specificare de histogramă [9], [5].

O ultimă remarcă se poate referi la validitatea introducerii tehnicilor de îmbunătăţire a histogramei în categoria de operaţii punctuale. Majoritatea autorilor consideră că orice operaţie care este echivalentă cu modificarea paletei de culoare a imaginii este o operaţie punctuală. În acelaşi timp însă, noile valori de gri ale fiecărui punct sunt calculate pe baza histogramei imaginii, deci pe baza unei măsuri ce ia în calcul valorile din întreaga imagine; din acest punct de vedere, egalizarea de histogramă poate fi inclusă în categoria operaţiilor integrale.

Capitolul 3

FILTRAREA LINIARĂ A IMAGINILOR

Odată cu operaţiile de filtrare liniară, se începe studiul operatorilor de vecinătate: operatorii al căror rezultat depinde atât de valoarea punctului în care au fost aplicaţi, cât şi de valorile unui număr de alte puncte vecine (nu neapărat topologic) punctului curent de calcul. Filtrarea se cheamă liniară pentru că operaţia verifică principiul superpoziţiei (liniarităţii): pentru două imagini f_1 şi f_2, doi scalari a_1 şi a_2 şi operatorul liniar L avem

$$L(a_1 f_1 + a_2 f_2) = a_1 L(f_1) + a_2 L(f_2) \tag{3.1}$$

Operaţia de filtrarea liniară calculează noua valoare a unui pixel al imaginii (din poziţia (m, n)) ca o combinaţie liniară (medie ponderată) a unui număr de valori din imaginea originală:

$$v(m, n) = \sum_{(k,l) \in W} w_{kl} f(m - k, n - l) \tag{3.2}$$

W este vecinătatea punctului curent în care se face calculul; W este numită mască sau fereastră de filtrare şi este o formă plană, descrisă ca o mulţime de puncte din spaţiul cartezian, în coordonate relative (deci în alt sistem de coordonate decât sistemul de coordonate a imaginii). Scalarii w_{kl} sunt ataşaţi poziţiilor (k, l) din fereastra de filtrare şi poartă numele de coeficienţi ai filtrului. Spre exemplu, o mască simplă de mediere este media aritmetică a valorilor dintr-o vecinătate pătrată de 3 x 3 pixeli, centrată în pixelul curent; pentru acest caz, toţi coeficienţii vor avea valoarea $1/9$ şi masca de filtrare W va fi: $W = \{(0,0), (-1,0), (1,0), (0,-1), (-1,-1), (1,-1), (0,1), (-1,1), (1,1)\}$. Pentru acest caz particular, expresia (3.2) devine:

$$v(m, n) = \sum_{k=-1}^{1} \sum_{l=-1}^{1} \frac{f(m - k, n - l)}{9} \tag{3.3}$$

adică, desfăşurat, $v(m,n) = \frac{f(m+1,n-1)}{9} + \frac{f(m+1,n)}{9} + \frac{f(m+1,n+1)}{9} + \frac{f(m,n)}{9} + \frac{f(m,n-1)}{9} + \frac{f(m,n+1)}{9} + \frac{f(m+1,n-1)}{9} + \frac{f(m+1,n)}{9} + \frac{f(m+1,n+1)}{9}$. Acelaşi lucru se poate exprima şi prin specificarea matricială a măştii de filtrare şi marcarea originii acesteia, ca $W = \begin{pmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & \boxed{1/9} & 1/9 \\ 1/9 & 1/9 & 1/9 \end{pmatrix}$.

Formula de definiţie (3.2) nu este altceva decât suma produselor punct cu punct a coeficienţilor măştii şi a valorilor pixelilor imaginii din zonă de imagine peste care a fost suprapusă masca. Această sumă de produse se calculează pentru fiecare punct al imaginii, deplasând masca. Descrierea algoritmului nu este însă altceva decât descrierea plastică a unei operaţii de convoluţie bidimensională, în care, prin convenţie, masca de filtrare este considerată nucleul de convoluţie. Deplasarea măştii (ferestrei de filtrare) a condus la adoptarea denumirii de tehnică a ferestrei glisante; aplicarea acesteia se poate descrie simplu:

- se plasează originea ferestrei de filtrare (pe rând) în fiecare punct al imaginii şi se selectează punctele imaginii situate în interiorul ferestrei (putem imagina fereastra de filtrare ca fiind o apertură într-o placă opacă; într-o anumită poziţie a acesteia valorile selectate din imagine sunt valorile punctelor ce se "văd" prin apertură).
- pentru fiecare poziţie se face suma produselor punct cu punct coeficient mască - valoare pixel.

Fereastra de filtrare este deci definită de formă (mulţimea W) şi valori (coeficienţii w_{kl}). Cele mai simple ferestre de filtrare sunt cele de formă pătrată, având originea în centru (deci fiind pătrate de dimensiuni impare: 3 x 3, 5 x 5, etc.) - de tipul celei prezentate în exemplul anterior. Fereastra de filtrare are în general o dimensiune mult mai mică decât dimensiunile imaginii (mai sunt numite şi nuclee mici, făcând referinţă la operaţia de convoluţie). Codul care urmează exemplifică această cea mai simplă filtrare, cu un nucleu de dimensiune 3 x 3. Se remarcă alocarea statică a coeficienţilor w ai filtrului (numere reale) şi alocarea dinamică a imaginii rezultat (considerată aici ca având valori întregi). Calculul valorilor pixelilor din imaginea filtrată s-a făcut decupând câte un rând şi o coloană de la extremităţile imaginii originale (pentru a evita efectele de margine, în care masca "debordează" în afara imaginii), acestea fiind completate în imaginea rezultat prin copierea valorilor originale. O altă variantă de evitare a efectelor de margine este bordarea (completarea) imaginii la capete cu câte o linie şi o coloană cu valori nule. De asemenea, se poate constata că în expresia de calcul efectiv indicii tabelului de coeficienţi ai măştii au fost deplasaţi, astfel încât indicele minim să fie 0, şi nu negativ.

```
real w[3][3];
img_out=(int**)malloc(NRLIN*sizeof(int*));
for (i=0;i<NRLIN;i++)
```

```
  img_out[i]=(int*)malloc(NRCOL*sizeof(int));
for (i=0;i<NRCOL;i++)
{ img_out[0][i]=img[0][i];
  img_out[NRLIN-1][i]=img[NRLIN-1][i];}
for (i=0;i<NRLIN;i++)
{ img_out[i][0]=img[i][0];
  img_out[i][NRCOL-1]=img[i][NRCOL-1];}
for (i=1;i<NRLIN-1;i++)
 for (j=1;j<NRCOL-1;j++)
  img_out[i][j]=round(w[1][1]*img[i][j]+w[1][0]*img[i][j-1]+
    w[1][2]*img[i][j+1]+w[0][1]*img[i-1][j]+w[0][0]*img[i-1][j-1]+
    w[0][2]*img[i-1][j+1]+w[2][1]*img[i+1][j]+w[2][0]*img[i+1][j-1]+
    w[2][2]*img[i+1][j+1]);
```

Pentru o imagine pătrată de dimensiune N şi o mască de filtrare pătrată de dimensiune n, numărul de operaţii necesare unei filtrări liniare este de N^2n^2 înmulţiri şi $N^2(n^2-1)$ adunări, adică complexitatea de calcul este $O[N^2]$, atât faţă de dimensiunile imaginii cât şi faţă de dimensiunile ferestrei de filtrare. Aceasta duce la constrângerea practică de a folosi ferestre de filtrare cât mai mici (şi de a încerca descompunerea ferestrelor mari de filtrare într-o succesiune de ferestre mai mici - spre exemplu o filtrare cu un nucleu de 5 x 5, care ar necesita 25 de operaţii pentru fiecare pixel, poate fi echivalată cu două filtrări succesive cu nuclee de 3 x 3 pixeli, care necesită doar 18 operaţii pentru fiecare pixel). În figurile următoare se prezintă efectul unei operaţii de mediere cu un nucleu de 7 x 7: se remarcă faptul că imaginea rezultat este mai neclară şi ceţoasă (efect de *blur*) şi că toate contururile au devenit mai vagi.

Imagine originală

Rezultatul unei filtrări de mediere

3.1 Filtrarea liniară de netezire

Efectul de înceţoşare a unei imagini poate fi considerat şi ca un efect de îmbunătăţire a uniformităţii regiunilor [18]. Aceasta înseamnă că se elimină micile diferenţe dintre valorile pixelilor aparţinând unei aceleiaşi regiuni (zone cu intensitate luminoasă relativ constantă). Acesta este fundamentul metodelor de reducere a zgomotului alb aditiv gaussian (normal) suprapus imaginii: un asemenea zgomot ce afectează o regiune absolut uniformă (în care toţi pixelii ce o formează au aceeaşi valoare) produce variaţia valorilor din interiorul acesteia, deci micşorează uniformitatea. Variaţiile introduse sunt cu atât mai mari cu cât puterea zgomotului este mai mare; pentru un zgomot de medie nulă (aşa cum este zgomotul alb gaussian aditiv) puterea este identică cu varianţa. Teoria proceselor aleatoare [16] arată că, pentru o combinaţie liniară de realizări ale unei variabile aleatoare $y = \sum_{i=1}^{n} a_i x_i$, varianţa noii variabile aleatoare este proporţională cu varianţa variabilei aleatoare din care au provenit realizările particulare: $\sigma_y^2 = \sum_{i=1}^{n} a_i^2 \sigma_x^2$. Deci, prin medierea a n realizări ale unei variabile aleatoare, varianţa este redusă de n ori ($a_i = \frac{1}{n}$).

Măsurile de calitate folosite pentru a caracteriza în mod obiectiv calitatea unei imagini (sau rezultatul unei prelucrări) sunt extensii bidimensionale ale măsurilor de calitate folosite pentru caracterizarea semnalelor unidimensionale. Dacă considerăm f imaginea originală (corectă) şi g imaginea a cărei calitate trebuie determinată (g este considerată că provine din f prin suprapunerea unui zgomot aditiv), rapoartele de calitate cele mai utilizate sunt: raportul semnal zgomot (Signal to Noise Ratio - SNR) (3.4), raportul semnal de vârf zgomot (Peak Signal to Noise Ratio - PSNR) (3.5), eroare pătratică medie (Mean Squared Error - MSE) (3.6) şi eroarea medie absolută (Mean Absolute Error - MAE) (3.7).

$$SNR = 10 \log \frac{\sum_{n=1}^{N} \sum_{m=1}^{M} f^2(n,m)}{\sum_{n=1}^{N} \sum_{m=1}^{M} \left(g(m,n) - f(n,m)\right)^2} \text{ dB} \tag{3.4}$$

$$PSNR = 10 \log \frac{NM \left(\max_{n,m} f(n,m)\right)^2}{\sum_{n=1}^{N} \sum_{m=1}^{M} \left(g(m,n) - f(n,m)\right)^2} \text{ dB} \tag{3.5}$$

$$MSE = \frac{1}{NM} \sum_{n=1}^{N} \sum_{m=1}^{M} \left(g(m,n) - f(n,m)\right)^2 \tag{3.6}$$

$$MAE = \frac{1}{NM} \sum_{n=1}^{N} \sum_{m=1}^{M} |g(m,n) - f(n,m)| \tag{3.7}$$

Fig. 3.1: Imagine degradată de un zgomot aditiv gaussian

Fig. 3.2: Reducerea zgomotului prin filtrarea de mediere

Figurile următoare prezintă o imagine degradată cu un zgomot aditiv, alb, gaussian, de medie nulă (figura 3.1 SNR=16.97 dB, PSNR=22.43 dB, MAE=15.22) şi imaginea filtrată cu un filtru liniar de mediere aritmetică, cu fereastră 3 x 3 (figura 3.2 SNR=19.33 dB, PSNR=24.79 dB, MAE=9.31). Se poate observa atât o calitate vizuală mai bună (regiuni mai uniforme) cât şi factori de calitate mai buni (SNR şi PSNR mai mari, MAE mai mic).

Un caz particular de interes este considerarea regiunilor constante (absolut uniforme) ale imaginii; în aceste regiuni toţi pixelii vor avea aceeaşi valore, şi deci uniformitatea nu mai poate fi îmbunătăţită. În acelaşi timp, operaţia de filtrare de netezire trebuie să păstreze această valoare constantă a pixelilor (pe care o vom nota cu μ); înlocuind în formula de

definiţie a filtrării liniare (3.2), vom obţine că

$$\mu = \sum_{(k,l)\in W} w_{kl}\mu$$

adică:

$$\sum_{(k,l)\in W} w_{kl} = 1 \tag{3.8}$$

Această condiţie (suma coeficienţilor măştii de filtrare să fie unitară) se numeşte condiţia de normare a nucleelor de filtrare de netezire şi defineşte un astfel de nucleu.

Nucleele de filtrare folosite nu trebuie limitate strict la nucleele ce realizează media aritmetică într-o vecinătate pătrată, centrată în punctul considerat. Măştile următoare realizează medieri ponderate: $W_1 = \begin{pmatrix} 1/16 & 1/16 & 1/16 \\ 1/16 & \boxed{1/2} & 1/16 \\ 1/16 & 1/16 & 1/16 \end{pmatrix}$, $W_2 = \begin{pmatrix} 0 & 1/5 & 0 \\ 1/5 & \boxed{1/5} & 1/5 \\ 0 & 1/5 & 0 \end{pmatrix}$, $W_3 = \begin{pmatrix} 0 & 1/8 & 0 \\ 1/8 & \boxed{1/4} & 1/8 \\ 0 & 1/8 & 0 \end{pmatrix}$, $W_4 = \begin{pmatrix} \boxed{1/2} & 1/6 \\ 1/6 & 1/6 \end{pmatrix}$. Nucleul de filtrare poate avea orice formă (deci există o libertate totală în definirea mulţimii W (3.2)); în acelaşi timp însă, practica a demonstrat suficienţa considerării unor forme regulate (pătrate, centrate) pentru mască. Orice mască poate fi extinsă cu puncte ce au ataşat un coeficient nul, pentru a rezulta o mască pătrată centrată; spre exemplu, masca W_4 poate fi scrisă şi ca $W_4 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \boxed{1/2} & 1/6 \\ 0 & 1/6 & 1/6 \end{pmatrix}$; la fel s-a procedat şi cu măştile W_2 şi W_3. Trebuie remarcat că toate aceste măşti respectă condiţia de normare a unui nucleu de netezire (3.8).

3.2 Filtrarea liniară de contrastare

Contrastarea unei imagini are ca obiectiv îmbunătăţirea perceperii vizuale a contururilor obiectelor (îmbunătăţirea detectabilităţii componentelor scenei de-a lungul frontierelor acestora). În principiu, acest deziderat se poate realiza prin modificarea valorilor pixelilor aflaţi de o parte şi de alta a unei frontiere comune.

O experienţă de percepţie vizuală subiectivă ("benzile lui Mach") a pus în evidenţă faptul că sistemul vizual uman are tendinţa de a adânci profilul zonelor de tranziţie dintre regiuni uniforme. Studiul fiziologiei sistemului vizual a demonstrat că acesta se realizează prin prelucrări de tip derivativ ce apar în diferitele etape pe care le parcurge informaţia vizuală; efectul global poate fi descris ca scăderea din semnalul original a unei derivate secunde a acestuia, ponderată corespunzător (aşa cum prezintă figurile 3.3 şi 3.4, pentru cazul unidimensional - secţiunea unei frontiere între regiunile imaginii).

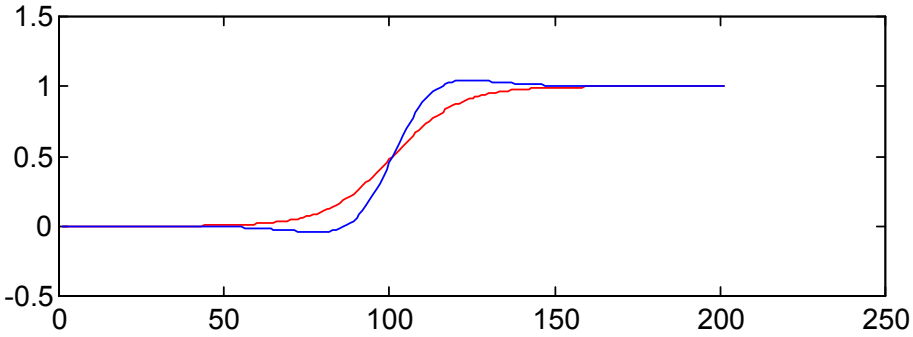


Fig. 3.3: Profilul original de tranziţie şi profilul adâncit prin scăderea derivatei secunde

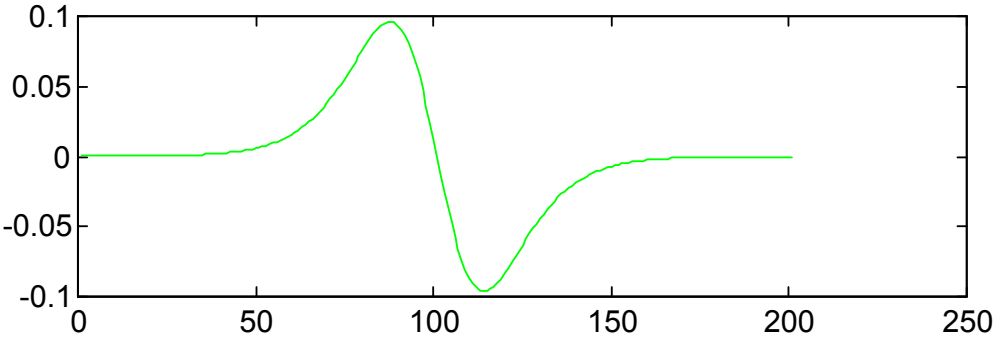


Fig. 3.4: Derivata secundă a profilului original prezentat anterior

Implementarea unei derivate secunde pentru cazul discret va trebui să ia în considerare existenţa a două direcţii de derivare de bază şi modul de definire a derivatei în cazul discret (de exemplu prima derivată pe direcţia orizontală poate fi $f'(m,n) = f(m,n+1) - f(m,n)$ sau $f'(m,n) = f(m,n) - f(m,n-1)$ sau $f'(m,n) = f(m,n+1) - f(m,n-1)$). Măşti obişnuite de implementare a unei derivate secunde bidirecţionale (operator Laplacian) pot fi [9], [19], [5]: $W_5 = \begin{pmatrix} 0 & -1/4 & 0 \\ -1/4 & \boxed{1} & -1/4 \\ 0 & -1/4 & 0 \end{pmatrix}$, $W_6 = \begin{pmatrix} 1/4 & -1/2 & 1/4 \\ -1/2 & \boxed{1} & -1/2 \\ 1/4 & -1/2 & 1/4 \end{pmatrix}$, $W_7 = \begin{pmatrix} -1/8 & -1/8 & -1/8 \\ -1/8 & \boxed{1} & -1/8 \\ -1/8 & -1/8 & -1/8 \end{pmatrix}$.

Pentru un astfel de operator de derivare este esenţial ca răspunsul său pentru pixeli din interiorul unei regiuni absolut uniforme (de valoare μ) să fie nul (derivata unei constante este nulă); pentru aceasta, din formula de definiţie a filtrării liniare (3.2) avem că $0 =$

$\sum_{(k,l)\in W} w_{kl}\mu$, adică:

$$\sum_{(k,l)\in W} w_{kl} = 0 \tag{3.9}$$

Aceasta este condiţia de normare a unui nucleu de filtrare derivativă, pentru care suma coeficienţilor măştii trebuie deci să fie nulă. Se remarcă faptul că nucleele de derivată secundă (operatori laplacieni) prezentate anterior verifică acestă condiţie de normare (3.9). Figurile următoare prezintă o imagine slab contrastată şi varianta sa îmbunătăţită prin contrastare cu nucleul laplacian W_5.

Imagine originală

Contrast îmbunătăţit

3.3 Filtrarea liniară adaptivă

Termenul de adaptiv se referă la capacitatea filtrului de a-şi ajusta comportarea (care este determinată de caracteristicile sale de definiţie) în funcţie de anumite criterii, urmărind optimizarea unor măsuri de calitate. În mod curent, optimizarea măsurilor de calitate se traduce în prelucrarea semnalelor unidimensionale prin minimizarea erorii pătratice medii (sau, corespunzător, maximizarea raportului semnal zgomot). Pe lângă măsurile obiective de calitate, prelucrarea imaginilor adaugă şi o măsură subiectivă: confortul vizual al unui observator, pentru care imaginea dată arată bine, sau mai bine decât o alta. Prin procesul de adaptare, în fiecare nou punct prelucrat structura filtrului este alta, luând în considerare caracteristicile locale pixelului curent prelucrat.

Atâta timp cât (cel puţin teoretic) în fiecare punct al imaginii operaţia este diferită (deoarece se face cu un filtru diferit), filtrarea globală nu mai este liniară (nu mai respectă principiul superpoziţiei (3.1)).

Adaptarea filtrelor liniare nu poate urmări decât două variante: modificarea formei ferestrei de filtrare, sau modificarea coeficienţilor unei ferestre de filtrare de formă fixată.

Exemplul cel mai folosit de modificare a modificare a formei ferestrei de filtrare este filtrul de netezire direcţională adaptivă [9]. Termenul de filtrare direcţională se referă la forma puternic orientată a ferestrei de filtrare (deci fereastra de filtrare nu mai este pătrată, ci va avea o formă liniară, orientată după o anumită direcţie). Pentru fiecare punct al imaginii se pot considera mai multe orientări (deci mai multe direcţii) posibile pentru masca de mediere. Pentru fiecare dintre direcţiile alese se obţine în urma filtrării o valoare. În mod ideal, dorim ca valoarea obţinută prin filtrare să nu difere în mod semnificativ de valoarea iniţială din pixelul considerat. O diferenţă semnificativă poate însemna că punctul curent este situat pe un contur, şi acceptarea acestei valori mult diferite de cea iniţială produce efectul de înceţoşare a imaginii (caracteristic oricărei filtrări de netezire). Soluţia adusă de filtrul adaptiv este de a alege din setul de valori obţinute pentru diferitele ferestre de filtrare pe aceea care este cea mai apropiată de valoarea iniţială din punctul curent. Fragmentul de cod următor prezintă un caz particular de filtrare direcţională adaptivă: ferestrele direcţionale au trei puncte şi sunt alese după cele patru direcţii principale (orizontal, vertical şi cele două diagonale, adică direcţiile orientate la 0°, 45°, 90° şi 135° faţă de orizontală).

```
int val_temp1,val_temp2,out0,out45,out90,out135;
for (i=1;i<NRLIN-1;i++)
 for (j=1;j<NRCOL-1;j++) {
  out0=round((img[i][j]+img[i][j-1]+img[i][j+1])/3);
  out90=round((img[i][j]+img[i-1][j]+img[i+1][j])/3);
  out45=round((img[i][j]+img[i-1][j+1]+img[i+1][j-1])/3);
  out135=round((img[i][j]+img[i-1][j-1]+img[i+1][j+1])/3);
  if (abs(out0-img[i][j])>abs(out90-img[i][j]))
   val_temp1=out90;
  else
   val_temp1=out0;
  if (abs(out45-img[i][j])>abs(out135-img[i][j]))
   val_temp2=out135;
  else
   val_temp2=out45;
  if (abs(val_temp1-img[i][j])>abs(val_temp2-img[i][j]))
   img_out[i][j]=val_temp2;
  else
   img_out[i][j]=val_temp1; }
```

În cod nu a mai fost inclusă alocarea de memorie pentru imaginea de ieşire *img_out* şi nici partea de copiere a valorii marginilor imaginii; se remarcă folosirea a patru variabile suplimentare *out0*, *out45*, *out90*, *out135* care primesc valorile obţinute prin filtrarea direcţională cu masca orientată la 0°, 45°, 90° şi 135°. Variabilele *val_temp1* şi *val_temp2* sunt folosite apoi pentru a calculul valorii celei mai apropiate de valoarea originală.

Ideea de a accepta valoarea produsă de medierea valorilor selectate de fereastra filtrului

numai dacă acestă valoare este suficient de apropiată de valoarea originală se poate aplica şi pentru filtrarea de netezire cu fereastră şi coeficienţi ficşi [5]. În acest caz rezultatul netezirii într-un punct va fi acceptat doar dacă diferenţa faţă de valoarea originală este mai mică decât un prag impus:

$$g(n,m) = \begin{cases} \sum\limits_{(k,l)\in W} w_{kl} f(m-k, n-l), \text{ dacă } \left| f(m,n) - \sum\limits_{(k,l)\in W} w_{kl} f(m-k,n-l) \right| < T \\ f(m,n), \text{ în rest} \end{cases}$$

O variantă clasică de filtrare liniară adaptivă, realizată prin recalcularea coeficienţilor măştii de filtrare în fiecare punct al imaginii este prezentată în [18]. Acest filtru este o combinaţie liniară convexă a imaginii zgomotoase şi a imaginii obţinute din aceasta printr-o filtrare liniară de mediere (cu fereastră constantă), adică: $\hat{f} = \alpha f + \beta \bar{f}$. O combinaţie liniară convexă se obţine doar dacă $\alpha + \beta = 1$. Deci

$$\hat{f} = \alpha f + (1-\alpha)\overline{f} \tag{3.10}$$

Imaginea zgomotoasă f provine dintr-o imagine corectă g, la care s-a adăugat un zgomot alb aditiv de medie nulă n ($\overline{n} = 0$) şi necorelat cu imaginea ($\overline{ng} = \overline{n}\,\overline{g} = 0$): $f = g + n$. Atunci

$$\hat{f} = \alpha(g+n) + (1-\alpha)\overline{(g+n)} = \alpha g + \alpha n + (1-\alpha)\overline{g} \tag{3.11}$$

Ceea ce defineşte filtrul adaptiv este coeficientul α; acesta este calculat local impunând optimizarea unei măsuri de calitate obiective - eroarea pătratică medie între imaginea originală g şi rezultatul filtrării $\widehat{f}$.

$$\varepsilon^2 = \left(\widehat{f} - g\right)^2 = (\alpha n + (\alpha-1)(g-\overline{g}))^2 = \alpha^2 n^2 + 2\alpha(\alpha-1)ng - 2\alpha(\alpha-1)n\overline{g} + (\alpha-1)^2(g-\overline{g})^2$$

$$\overline{\varepsilon^2} = \overline{\alpha^2 n^2 + 2\alpha(\alpha-1)ng - 2\alpha(\alpha-1)n\overline{g} + (\alpha-1)^2(g-\overline{g})^2} = \alpha^2\overline{n^2} + (\alpha-1)^2\overline{(g-\overline{g})^2}$$

$$\overline{\varepsilon^2} = \alpha^2\sigma_n^2 + (\alpha-1)^2\sigma_g^2 \tag{3.12}$$

Formula (3.12) arată că eroarea pătratică medie este suma ponderată dintre varianţa (puterea) zgomotului σ_n^2 şi varianţa imaginii originale σ_g^2. Găsirea coeficientului α optim înseamnă găsirea minimului lui $\overline{\varepsilon^2}$, deci anularea derivatei acestuia.

$$\frac{d\overline{\varepsilon^2}}{d\alpha} = 2\alpha\sigma_n^2 + 2(\alpha-1)\sigma_g^2 = 0$$

$$\alpha = \frac{\sigma_g^2}{\sigma_g^2 + \sigma_n^2} = 1 - \frac{\sigma_n^2}{\sigma_f^2} \tag{3.13}$$

De aici rezultă expresia finală a filtrului:

$$\widehat{f} = \left(1 - \frac{\sigma_n^2}{\sigma_f^2}\right) f + \frac{\sigma_n^2}{\sigma_f^2}\overline{f} \tag{3.14}$$

Se observă că trebuiesc cunoscute puterea de zgomot şi varianţa locală (în fiecare punct) al imaginii zgomotoase. Adaptarea se face între două cazuri limită: dacă puterea de zgomot este mult mai mare decât varianţa valorilor imaginii originale ($\sigma_n^2 \gg \sigma_g^2$) atunci, din prima parte a formulei (3.13) rezultă $\alpha = 0$ şi deci filtrul optim este un simplu filtru de mediere; dacă varianţa valorilor din imagine este mult mai mare decât puterea de zgomot ($\sigma_f^2 \gg \sigma_n^2$, deci este de presupus că punctul curent considerat este situat pe un contur) atunci $\alpha = 1$ şi filtrul optim este un filtru identitate (trece tot). Prezentăm în continuare codul Matlab ce implementează acestă filtrare adaptivă:

```
[NRLIN,NRCOL]=size(img);
for i=2:NRLIN-1
 for j=2:NRCOL-1
   temp=img(i-1:i+1,j-1:j+1);
   temp=temp(:);
   varianta_img=std(temp);
   alfa=1-varianta_zg/varianta_img;
   img_out(i,j)=alfa*img(i,j)+(1-alfa)mean(temp);
 end
end
```

Se folosesc funcţiile Matlab *mean* (pentru a calcula media valorilor selectate de fereastra de filtrare pătrată de 3 x 3) şi *std* (pentru a calcula varianţa locală a imaginii). Se presupune că puterea de zgomot *varianta_zg* este un parametru cunoscut.

În capitolul următor se va introduce o nouă posibilitate de caracterizare a efectelor filtrării liniare a imaginilor: reprezentarea în domeniul frecvenţelor spaţiale (o extensie directă a spectrului semnalelor unidimensionale). De asemenea vom arăta că rămâne valabilă teorema convoluţiei, prin care vom introduce şi o nouă modalitate de implementare a filtrelor liniare: filtrarea în domeniul de frecvenţă.

Capitolul 4

TRANSFORMĂRI INTEGRALE UNITARE DISCRETE

4.1 Generalităţi

Această categorie de prelucrări include operaţii de tip integral – totalitatea pixelilor imaginii iniţiale contribuie la obţinerea valorii fiecărui pixel din imaginea rezultat. În [9] se consideră că termenul de transformări de imagine se referă în mod uzual la o clasă de matrici unitare folosite pentru reprezentarea imaginilor. Orice imagine poate fi reprezentată ca o serie de matrici de bază ortonormale. Pentru o imagine pătrată $\mathbf{U}$[1] de dimensiune N, o dezvoltare în serie este

$$\mathbf{U} = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} \mathbf{A}_{kl}^{*} V(k,l) \tag{4.1}$$

unde $\mathbf{A}_{kl}^{*}$ sunt matricile de bază[2] (de dimensiuni $N \times N$) iar $V(k,l)$ sunt coeficienţii dezvoltării în serie. Exprimând relaţia (4.1) la nivelul fiecărui pixel al imaginii $\mathbf{U}$ obţinem

$$U(m,n) = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} a_{kl}^{*}(m,n) V(k,l) \tag{4.2}$$

Calculul coeficienţilor $V(k,l)$ ai transformării se poate face în condiţiile impuse de orto-

[1] În acest capitol vom folosi notaţiile matriciale şi vectoriale clasice: litere mari, drepte şi groase pentru matrici ($\mathbf{A}$ este o matrice), litere mici, drepte şi groase pentru vectori ($\mathbf{v}$ este un vector) şi litere înclinate pentru elementele vectorilor şi matricilor ($A(i,j)$, $v(m)$).

[2] Numite uneori şi imagini de bază.

gonalitate şi completitudine a matricilor de bază $\mathbf{A}_{kl}^*$ prin:

$$V(k,l) = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} a_{kl}(m,n)U(m,n) \tag{4.3}$$

Mulţimea coeficienţilor transformării $V(k,l)$ formează matricea $\mathbf{V}$, transformata imaginii. Atunci relaţia (4.2) este transformarea directă a imaginii, iar relaţia (4.3), prin care se obţine imaginea din transformata sa, este transformata inversă. Se poate remarca faptul că ambele transformări (directă şi inversă) necesită N^4 operaţii de înmulţire, şi deci o complexitate $O(N^4)$.

Condiţia de ortonormalitate a matricilor de bază se exprimă ca

$$\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} a_{kl}(m,n)a_{k'l'}^*(m,n) = \delta(k-k', l-l'),\ \forall k,l,k',l' \tag{4.4}$$

şi asigură faptul că orice dezvoltare în serie truncheată minimizează eroarea pătratică de aproximare.

Condiţia de completitudine a matricilor de bază se exprimă ca

$$\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} a_{kl}(m,n)a_{kl}^*(m',n') = \delta(m-m', n-n'),\ \forall m,n,m',n' \tag{4.5}$$

şi asigură faptul că baza de matrici folosită este completă.

O transformare de tip (4.2) este deci caracterizată de cei N^4 coeficienţi $a_{kl}(m,n)$, ce pot fi interpretaţi ca valori ai unei funcţii de patru variabile (k,l,m,n), câte două pentru fiecare imagine (iniţială şi transformată). Această funcţie se numeşte nucleul transformării. Prin definiţie, o transfomare se zice separabilă dacă nucleul ei este separabil după perechi de variabile corespondente:

$$a_{kl}(m,n) = a_k(m)b_l(n) = a(k,m)b(l,n) \tag{4.6}$$

Impunând condiţiile (4.4) şi (4.5) acestei noi forme a coeficienţilor transformării, rezultă că matricile $\mathbf{A} = \{a(k,m)\}$ şi $\mathbf{B} = \{b(l,n)\}$ trebuie să fie matrici unitare:

$$\mathbf{A}\mathbf{A}^{*T} = \mathbf{A}^T\mathbf{A}^* = \mathbf{I}_N \tag{4.7}$$

$$\mathbf{B}\mathbf{B}^{*T} = \mathbf{B}^T\mathbf{B}^* = \mathbf{I}_N$$

Pentru o transformare separabilă, relaţiile de definiţie (4.2) şi (4.3) pot fi rescrise sub formă matricială ca:

$$\mathbf{V} = \mathbf{A}\mathbf{U}\mathbf{B}^T \tag{4.8}$$

$$\mathbf{U} = \mathbf{A}^{*T}\mathbf{V}\mathbf{B}^* \tag{4.9}$$

Numărul de înmulţiri necesare pentru a realiza oricare dintre transformările (4.8) sau (4.9) este doar N^3 (deci o complexitate $O(N^3)$). Proprietatea de separabilitate conduce deci la reducerea complexităţii calculelor cu un ordin de mărime; în practică se folosesc numai transformări separabile, pentru care, în plus, $\mathbf{A} = \mathbf{B}$. În acest caz particular, relaţia (4.8) se poate scrie ca

$$\mathbf{V} = \mathbf{AUA}^T = \mathbf{A}(\mathbf{AU}^T)^T = (\mathbf{U}^T\mathbf{A}^T)^T\mathbf{A}^T$$

ceea ce înseamnă că se poate face o transformare unidimensională pe fiecare linie sau coloană a lui $\mathbf{U}$ urmată de aceeaşi transformare pe fiecare coloană sau linie a rezultatului. Astfel, transformările practic utilizate în prelucrarea imaginilor (matricilor) sunt (paradoxal ?) unidimensionale.

Aceste observaţii ne conduc la concluzia că pentru acoperirea cazurilor utilizate în mod curent este suficientă studierea proprietăţilor transformatelor integrale unitare unidimensionale.

4.2 Proprietăţile transformatelor unitare unidimensionale

În cele ce urmează vom considera semnalul de intrare $\mathbf{u} = (u(0), u(1), ..., u(N-1))$ şi transformata sa $\mathbf{v}$, obţinută prin folosirea matricii unitare $\mathbf{A}$. Relaţiile de transformare (directă şi inversă) sunt $\mathbf{v} = \mathbf{Au}$ şi $\mathbf{u} = \mathbf{A}^{*T}\mathbf{v}$.

1. Energia semnalului se conservă printr-o transformare unitară.

$$E_{\mathbf{v}} = \|\mathbf{v}\|^2 = \mathbf{v}^{*T}\mathbf{v} = (\mathbf{Au})^{*T}\mathbf{Au} = \mathbf{u}^{*T}\mathbf{A}^{*T}\mathbf{Au} = \mathbf{u}^{*T}\mathbf{u} = \|\mathbf{u}\|^2 = E_{\mathbf{u}} \qquad (4.10)$$

   Energia vectorului semnal este de fapt lungimea (Euclidiană) a acestuia în spaţiul N-dimensional de reprezentare. Conservarea energiei este deci echivalentă cu conservarea lungimii vectorului, deci transformarea unitară este o rotaţie în spaţiul semnalului. Aceasta înseamnă că baza de reprezentare a lui $\mathbf{u}$ este rotită, iar $\mathbf{v}$ este proiecţia lui $\mathbf{u}$ pe noua bază.

2. Energia medie a semnalului se conservă printr-o transformare unitară.

$$\overline{\mathbf{v}}=\overline{\mathbf{Au}}=\mathbf{A}\overline{\mathbf{u}} \qquad (4.11)$$

$$\overline{E_{\mathbf{v}}} = \overline{\mathbf{v}}^{*T}\overline{\mathbf{v}} = (\mathbf{A}\overline{\mathbf{u}})^{*T}\mathbf{A}\overline{\mathbf{u}} = \overline{\mathbf{u}}^{*T}\mathbf{A}^{*T}\mathbf{A}\overline{\mathbf{u}}=\overline{\mathbf{u}}^{*T}\overline{\mathbf{u}} = \overline{E_{\mathbf{u}}} \qquad (4.12)$$

   Corelaţiei componentelor semnalului se modifică; dacă notăm cu $\mathbf{C}$ matricea de covariaţie atunci:

$$\mathbf{C}_{\mathbf{u}} = \overline{(\mathbf{u}-\overline{\mathbf{u}})(\mathbf{u}-\overline{\mathbf{u}})^{*T}} \qquad (4.13)$$

$$\mathbf{C}_{\mathbf{v}} = \overline{(\mathbf{v}-\overline{\mathbf{v}})(\mathbf{v}-\overline{\mathbf{v}})^{*T}} = \overline{\mathbf{A}(\mathbf{u}-\overline{\mathbf{u}})(\mathbf{u}-\overline{\mathbf{u}})^{*T}\mathbf{A}^{*T}} = \mathbf{A}\mathbf{C}_{\mathbf{u}}\mathbf{A}^{*T} \tag{4.14}$$

Majoritatea transformărilor unitare au tendinţa de a aglomera o mare parte a energiei medii a semnalului în relativ puţini coeficienţi ai transformării. Deoarece energia totală se conservă prin transfomare, mulţi coeficienţi ai transformării vor conţine foarte puţină energie. Energia medie a coeficienţilor transformării va avea o distribuţie neuniformă, chiar dacă în secvenţa de intrare aceasta era uniform distribuită.

Dacă componentele lui $\mathbf{u}$ sunt puternic corelate, coeficienţii transformării vor fi decorelaţi (termenii matricii de covariaţie care nu sunt pe dioagonala principală vor avea valori mici comparativ cu valorile de pe diagonală).

3. Entropia unui vector cu componente aleatoare se conservă printr-o transformare unitară

$$H(\mathbf{u}) = \frac{N}{2}\log_2(2\pi e\,|\mathbf{C}_{\mathbf{u}}|^{1/N}) = \frac{N}{2}\log_2(2\pi e\,|\mathbf{C}_{\mathbf{v}}|^{1/N}) = H(\mathbf{v}) \tag{4.15}$$

Deoarece entropia este o măsură a cantităţii de informaţie (informaţia medie) înseamnă ca transformările unitare păstrează informaţia conţinută în semnal.

4.3 Transformata Fourier discretă

Transformata Fourier este poate cea mai importantă transformare integrală unitară, ce asigură trecerea între spaţiul semnalului şi spaţiul de frecvenţe ale semnalului. După cum semnalul este "clasic" (temporal, şi deci unidimensional) sau cu suport spaţial bidimensional (imagine), spaţiul de frecvenţă marchează frecvenţe propriu-zise sau frecvenţe spaţiale.

Transformata Fourier discretă bidimensională este o transformare separabilă, în care nucleul se poate descompune în termeni identici $\mathbf{A} = \mathbf{B} = \mathbf{F}$. Matricea $\mathbf{F}$ a transformării este o matrice unitară, ce verifică (4.7). Elementele matricii transformării sunt exponenţialele complexe:

$$F(k,n) = \frac{1}{\sqrt{N}}e^{-j\frac{2\pi}{N}kn} = \frac{1}{\sqrt{N}}w_N^{kn} \tag{4.16}$$

Separabilitatea ne permite deci să studiem majoritatea proprietăţilor transformării pe cazul unidimensional, urmând ca rezultatele să fie uşor extinse pentru cazul bidimensional. Folosind notaţiile introduse în secţiunea 4.2, putem scrie deci transformata Fourier unidimensională directă ca:

$$v(k) = \sum_{n=0}^{N-1} u(n)w_N^{kn},\ k = \overline{0, N-1} \tag{4.17}$$

iar transformarea inversă ca

$$u(n) = \frac{1}{N} \sum_{k=0}^{N-1} v(k) w_N^{-kn},\ n = \overline{0, N-1} \tag{4.18}$$

Relaţiile se extind imediat la cazul bidimensional:

$$v(k,l) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} u(m,n) w_N^{kn+ml},\ k,l = \overline{0, N-1} \tag{4.19}$$

$$u(m,n) = \frac{1}{N^2} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} v(k,l) w_N^{-(kn+ml)},\ m,n = \overline{0, N-1} \tag{4.20}$$

Trebuie totuşi remarcat că formele prezentate ale transformărilor nu mai sunt unitare (apare o asimetrie între transformarea directă şi cea inversă prin factorul de scalare de tip $1/N$). Matematic, definiţia (4.16) este corectă, dar în practică se folosesc formele neunitare prezentate în (4.17), (4.18) şi (4.19), (4.20).

4.3.1 Proprietăţile fundamentale ale transformatei Fourier

Dintre numeroasele proprietăţi ale transformatei Fourier [9], [19] ne vom referi doar la două proprietăţi fundamentale: simetria centrală a spectrului de frecvenţă (având drept consecinţă importantă faptul că este necesar acelaşi spaţiu de memorie pentru a reprezenta fie spectrul unei imagini, fie imaginea propriu-zisă) şi teorema convoluţiei circulare (care face legătura între filtrarea în domeniul spaţial şi în domeniul de frecvenţă).

Transformata Fourier a unei secvenţe (matrici) reale este complex conjugată faţă de mijlocul său [9]. Aceasta înseamnă că

$$v(\frac{N}{2} - k) = v^*(\frac{N}{2} + k),\ k = \overline{0, \frac{N}{2}} \tag{4.21}$$

$$v(\frac{N}{2} - k, \frac{N}{2} - l) = v^*(\frac{N}{2} + k, \frac{N}{2} + l),\ k,l = \overline{0, \frac{N}{2}}$$

Relaţiile arată că există o simetrie centrală a spectrelor de frecvenţă, în centru aflându-se o valoare reală. Dar singura valoare reală din spectre este cea ce corespunde componentei de fecvenţă nulă (componenta medie), şi deci este necesară o interschimbare a jumătăţilor de spectru (în cazul secvenţelor unidimensionale) sau a sferturilor de spectru (în cazul imaginilor), astfel încât componenta de frecvenţă nulă să fie în centru. În figura 4.1 este reprezentat modulul spectrului de frecvenţă al imaginii "lena", reprezentat cu 256 nivele de gri (valorile au fost scalate după o reprezentare logaritmică) astfel încât valorile mai mari corspund unei nuanţe mai deschise. Se observă în partea dreaptă spectrul central simetrizat, în care valorile maxime (corespunzâd frecvenţelor cele mai mici, inclusiv

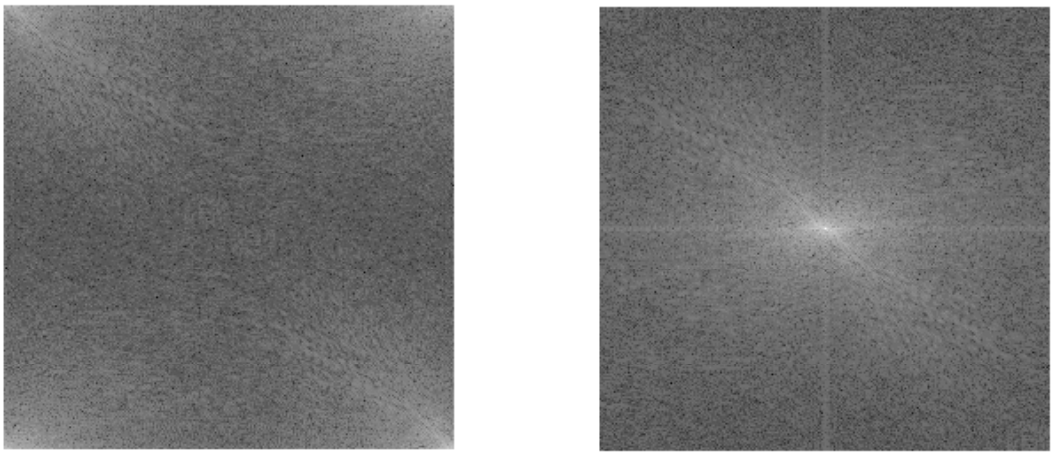


Fig. 4.1: Spectrele de frecvenţă (Fourier) ale imaginii "lena", înainte şi după aranjarea cu simetrie centrală.

zero) se găsesc în centrul imaginii, iar în partea stângă spectrul original obţinut în urma transformării Fourier, în care maximele se găsesc pe cele patru colţuri.

Ca şi în cazul unidimensional, şi pentru imagini legătura dintre frecvenţele spaţiale şi obiectele din domeniul spaţial (imagine) se păstrează: frecvenţele spaţiale ridicate corespund detaliilor, obiectelor mici, contururilor, zgomotului. În figura 4.2 este prezentat spectrul de frecvenţă al imaginii "lena" degradată de zgomot impulsiv de tip sare şi piper (figura 5.1); zgomotul impulsiv corespunde apariţiei a numeroase detalii (obiecte mici – punctele de zgomot) şi variaţii ale nivelului de gri. Efectul evident este acela de mărire a valorilor din spectru corespunzătoare frecvenţelor înalte şi, corespunzător, diminuarea importanţei relative a frecvenţelor joase.

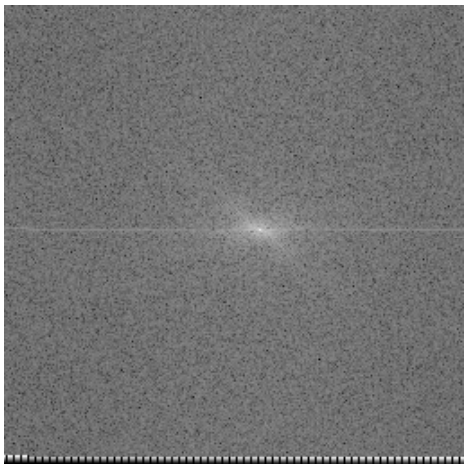


Fig. 4.2: Spectrul de frecvenţă al imaginii "lena" degradată de zgomot impulsiv.

Prelucrarea imaginilor (fie ele scalare sau vectoriale) presupune executarea, prin intermediul unui bloc funcţional ce poate fi presupus iniţial de tip "cutie neagră", a unei operaţii de tip *Image In, Image Out* [2] (spre deosebire de tehnicile de analiză ce pot fi caracterizate ca operaţii *Image In, Description Out*). Transformarea imaginii de intrare în imaginea rezultat de ieşire (ce prezintă aceleaşi caracteristici majore, simbolice şi perceptuale[3]) se numeşte în mod generic filtrare[4]. Funcţia unui filtru este de a transforma un semnal dat într-un alt semnal, mai potrivit unei anume aplicaţii date.

Definiţiile filtrării (sau ale dispozitivului care o realizează, filtrul) ce se regăsesc în dicţionarele lingvistice de uz general fac apel la noţiunea de semnal electric şi de frecvenţe asociate componentelor acestuia. Descrierea acţiunii filtrului este descrierea unei operaţii liniare efectuate în domeniul frecvenţelor. Extinderea acestor definiţii la cazul imaginilor presupune în primul rând definirea spectrului de frecvenţă asociat unui semnal cu suport [spaţial] multidimensional şi, implicit, introducerea noţiunii de frecvenţă spaţială. Utilizarea unei prelucrări în domeniul frecvenţelor presupune apoi identificarea benzilor de frecvenţă corepunzătoare entităţilor ce se doresc păstrate, respectiv eliminate, asociere ce nu este întotdeauna simplă. În practică, aceasta înseamnă determinarea unui spectru de frecvenţe transformat, care apoi este transformat prin transformata Fourier inversă în imaginea filtrată – aceasta este filtrarea în domeniul frecvenţă.

Filtrarea liniară a imaginilor se bazează pe convoluţia [circulară] între imaginea de prelucrat şi un nucleu de filtrare (a se vedea capitolul 3); teoria transformatelor unitare [9] arată că o asemenea operaţie este echivalentă unui produs între spectrul Fourier al imaginii şi spectrul Fourier al nucleului de filtrare; aceasta este teorma convoluţiei. În cazul unidimensional, dacă secvenţele $\mathbf{x}$ şi $\mathbf{y}$ au aceeaşi lungime, şi $\circledast$ este operatorul de convoluţie circulară (definit în (4.23)) atunci:

$$\text{Fourier}(\mathbf{x} \circledast \mathbf{y}) = \text{Fourier}(\mathbf{x})\,\text{Fourier}(\mathbf{y}) \tag{4.22}$$

$$\mathbf{x} \circledast \mathbf{y}(n) = \sum_{k=0}^{N-1} x\left((n-k) \bmod N\right) y(k),\ n = \overline{0, N-1} \tag{4.23}$$

Teorema convoluţiei oferă deci instrumentul prin care se pot echivala operaţiile de filtrare realizate în domeniile spaţial (filtrarea liniară, prezentată în capitolul 3) şi de frecvenţă. Aceasta înseamnă că un nucleu mic de convoluţie (mască de filtrare) este bordat cu zerouri până la obţinerea dimensiunii imaginii (teorema convoluţiei circulare se referă la

[3]Prin prisma acestei definiţii, transfomările unitare ale imaginilor (deci reprezentarea într-un spaţiu spectral) nu sunt filtrări; spectrul unei imagini, deşi este echivalent acesteia din punct de vedere informaţional, nu are aceleaşi caracteristici perceptuale.

[4]Filtrul este un sistem de circuite electrice, sonore, etc. cu care se selectează dintr-un complex de oscilaţii cu frecvenţe diferite, oscilaţiile cu frecvenţele cuprinse între anumite limite. (Dicţionarul Explicativ al Limbii Române, 1995)

Filtrul este un dispozitiv ce amortizează selectiv oscilaţiile cu anumite frecvenţe şi nu afectează oscilaţiile având alte frecvenţe. (Webster Encyclopedic Unabridged Dictionary, 1995)

Filtrul este un dispozitiv destinat favorizării sau inhibării trecerii anumitor componente de frecvenţă a unui semnal electric. (Larousse, 1994)

secvenţe de aceeaşi lungime) şi apoi transformat Fourier; ceea ce se obţine este răspunsul în frecvenţă al filtrului. Figura 4.3 prezintă răspunsul în frecvenţă a filtrului de mediere aritmetică realizat cu un nucleu pătrat, centrat de 3 x 3; se remarcă comportamentul de tip filtru trece jos. Acest comportament este leagt de efectul de "blur" al filtrului de mediere, pus în evidenţă prin reducerea vizibilităţii contururilor imaginii, şi deci înlăturarea frecvenţelor înalte ce le sunt asociate.

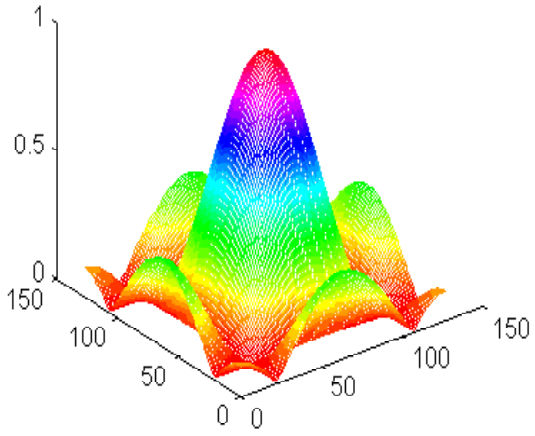


Fig. 4.3: Răspunsul în frecvenţă a unui nucleu 3 x 3 de mediere aritmetică.

4.3.2 Transformata Fourier rapidă

Implementarea directă a relaţiei de definiţie a transformării Fourier discrete unidimensionale (4.17) necesită evident N^2 înmulţiri complexe şi $N(N-1)$ adunări, ceea ce duce la o complexitate $O(N^2)$. Realizarea unei implementări mai rapide a transformării Fourier (FFT - *Fast Fourier Transform*) a constituit unul dintre momentele cele mai importante pentru domeniul prelucrării digitale a semnalelor. Tehnica iniţială a fost divizarea în timp: cele N eşantioane ale secvenţei se împart în două, după indicii pari şi respectiv impari:

$$v(k) = \sum_{n=0}^{N-1} u(n) \exp(-j\frac{2\pi}{N}kn) =$$

$$= \sum_{n=0}^{N/2-1} u(2n) \exp(-j\frac{2\pi}{N}k \cdot 2n) + \sum_{n=0}^{N/2-1} u(2n+1) \exp(-j\frac{2\pi}{N}k(2n+1)) =$$

$$= \sum_{n=0}^{N/2-1} u(2n) \exp(-j\frac{2\pi k}{N/2}2n) \exp(j\frac{2\pi k}{N}2n) + \exp(-j\frac{2\pi k}{N}) \sum_{n=0}^{N/2-1} u(2n+1) \exp(-j\frac{2\pi k}{N/2}2n) =$$

$$= \sum_{n=0}^{N/2-1} u_{par}(n) \exp(-j\frac{2\pi k}{N/2}n) + \exp(-j\frac{2\pi k}{N}) \sum_{n=0}^{N/2-1} u_{impar}(n) \exp(-j\frac{2\pi k}{N/2}n) =$$

$$= v_{par}(k) + \exp(-j\frac{2\pi k}{N}) v_{impar}(k) \tag{4.24}$$

În (4.24) am notat cu $\mathbf{u}_{par}$ secvenţa obţinută din toate valorile pare ale secvenţei iniţiale, iar cu $\mathbf{u}_{impar}$ secvenţa obţinută din toate valorile impare ale secvenţei iniţiale. Această relaţie exprimă posibilitatea construirii transformării Fourier a secvenţei $\mathbf{u}$ de lungime N prin transformările Fourier ale unor jumătăţi de secvenţă (de lungime $N/2$) – $\mathbf{v}_{par}$ şi $\mathbf{v}_{impar}$.

Dacă notăm cu $T(N)$ timpul de calcul al transformatei Fourier a secvenţei de lungime N, atunci avem:

$$T(N) = 2T\left(\frac{N}{2}\right) + N \tag{4.25}$$

În mod evident $T(1) = 1$ (transformata Fourier a secvenţei de lungime 1 este identică cu secvenţa); dezvoltând ecuaţia recurentă din (4.25) obţinem:

$$T(N) = 2T\left(\frac{N}{2}\right) + N = 2\left(2T\left(\frac{N}{4}\right) + \frac{N}{2}\right) + N = 4T\left(\frac{N}{4}\right) + 2N =$$

$$= ... = 2^i T\left(\frac{N}{2^i}\right) + iN \tag{4.26}$$

La limită, când i este ales astfel ca $N = 2^i$ vom avea:

$$T(N) = NT(1) + N\log_2 N = N\log_2 N + N$$

ceea ce este echivalent cu o complexitate de $O(N \log N)$. Sporul de viteză obţinut prin FFT faţă de implementarea clasică este deci proporţional cu $N/\log_2 N$ (deci aproape un ordin de mărime pentru o lungime tipică de secvenţă $N = 256$).

Relaţia de implementare de bază (4.24) mai poate fi scrisă şi ca:

$$v(k) = v_{par}(k) + w_N^k v_{impar}(k),\ k = \overline{0, ..., \frac{N}{2} - 1} \tag{4.27}$$

$$v(k) = v_{par}(k) - w_N^k v_{impar}(k),\ k = \overline{\frac{N}{2}, ..., N - 1}$$

ceea ce semnifică faptul că două valori ale transformărilor Fourier ale secvenţelor pe jumătate sunt combinate liniar pentru a genera două valori, simetrice faţă de centru, ale secvenţei originale. Blocul ce realizează aceste prelucrări este numit celulă "*butterfly*", caracterizat de o operaţie de înmulţire-adunare[5] (figura 4.4).

[5] Operaţia de îmulţire-adunarte este una dintre operaţiile de bază (cablate) din setul de instrucţiuni al procesoarelor de semnal (DSP).

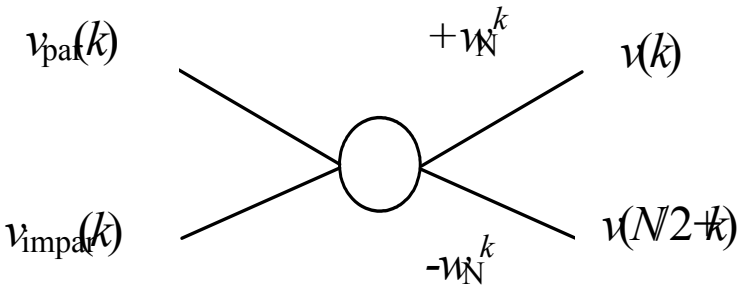


Fig. 4.4: Celula de bază butterfly

Construind etapele succesive de înjumătăţire a lungimii secvenţei, pentru o secvenţă de lungime $N = 8$ se ajunge la schema din figura 4.5. Se remarcă reordonarea eşantioanelor secvenţei de intrare după ordinea de bit inversă *bit-reverse* (forma binară a noului indice este forma binară a vechiului indice reflectată).

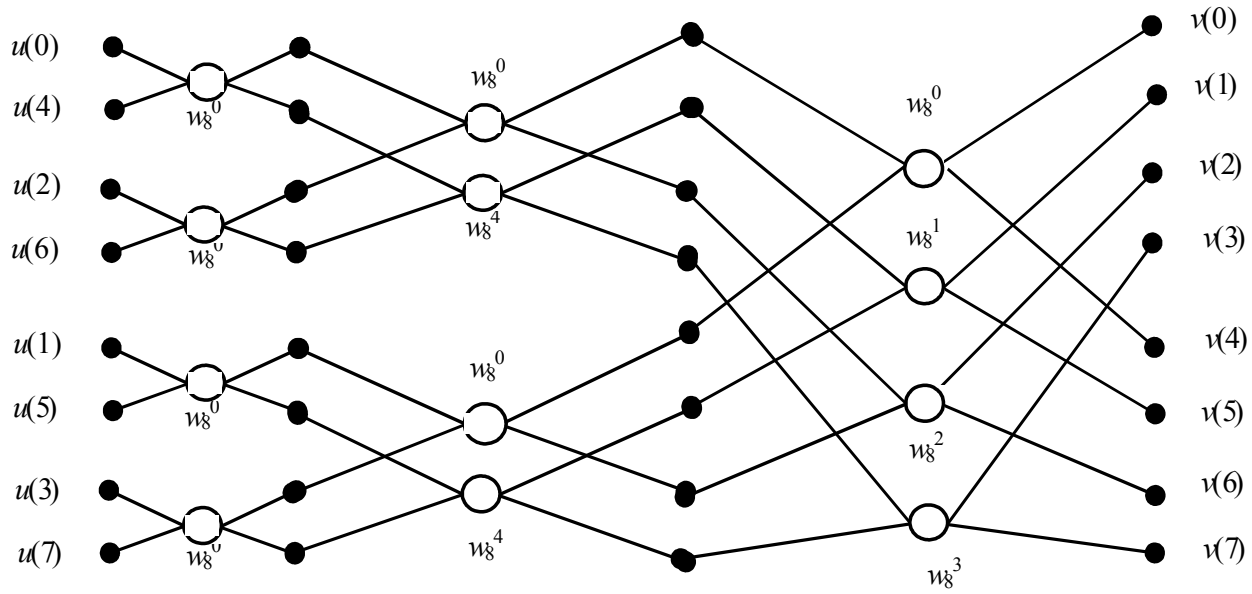


Fig. 4.5: Schema FFT pentru o secvenţă de lungime 8.

Implementările clasice ale FFT folosesc o reordonare *in-place* a secvenţei de intrare (în acelaşi spaţiu de memorie, prin interschimbări) pentru numere complexe (scrise în ordinea parte reală, parte imaginară). Acelaşi cod este folosit atât pentru transformata directă cât şi pentru transformata inversă, doar prin modificarea unui parametru de semn ce intervine la calculul coeficienţilor Fourier w_N. Rezultatul (transformata) ia locul datelor

de intrare, în acelaşi spaţiu de memorie (deci din nou o implementare *in-place*). În continuare prezentăm codul C din [13].

```
void fft(float data,integer nn,isign)
integer m,n,mmax,i,j,step;

n=nn<<1;
j=1;
for (i=1;i<n;i+=2){
  if (j>i) {
    swap(data[j],[data[i]);
    swap(data[j+1],[data[i+1]);}}
m=n>>1;
while (m>=2 && j>m) {
  j-=m;
  m>>=1;}
j+=m;}

mmax=2;
while (n>mmax) {
  step=2*mmax;
  teta=2*PI/(isign*mmax);
  wtemp=sin(teta/2);
  wpr=-2*wtemp*wtemp;
  wpi=sin(teta);
  wr=1;
  wi=0;
  for (m=1;m<mm1x;m+=2) {
    for (i=m;i<=n;i+=step) {
      j=i+mmax;
      tempr=wr*data[j]-wi*data[j+1];
      tempi=wr*data[j+1]+wi*data[j];
      data[j]=data[i]-tempr;
      data[j+1]=data[i+1]-tempi;
      data[i]+=tempr;
      data[i+1]+=tempi;}
    wr=(wtemp=wr)*wpr-wi*wpi+wr;
    wi=wi*wpr+wtemp*wpi+wi;}
  mmax=step;}
```

Prima parte a codului realizează reordonarea secevenţei în ordinea inversă de bit. În partea a doua a codului se realizează calculul efectiv al celulelor *butterfly*, corespunzând transformărilor unor secvenţe de dimensiune *mmax*, şi deci preluării unor date situate la

distanţa *mmax* între indici (vizibil şi în schema din figura 4.5).

Transformata rapidă bidimensională implică realizarea câte unei transformări unidimensionale pentru fiecare linie şi coloană a imaginii (conform principiului separabilităţii); aceasta conduce la o complexitate $O(N^2 \log N)$.

4.4 Alte transformări

În cele ce urmează vom prezenta pe scurt alte transformări unitare folosite în prelucrarea imaginilor; apariţia acestora s-a datorat adaptării mai bune decât transformata Fourier la compactarea energiei şi decorelarea spectrală a anumitor clase de semnale.

4.4.1 Transformata cosinus

Transformata cosinus este o transformată unitară separabilă, caracterizată de $\mathbf{A} = \mathbf{B} = \mathbf{C}$. Elementele matricii $\mathbf{C}$ sunt date de

$$C(k,n) = \frac{1}{\sqrt{N}} \begin{cases} 1, \text{ dacă } k = 0 \\ \sqrt{2}\cos\frac{(2n+1)\pi k}{2N}, \text{ dacă } k = \overline{1, N-1} \end{cases}, n = \overline{0, N-1} \tag{4.28}$$

Transformările directă şi inversă a unei secvenţe $\mathbf{u}$ sunt definite ca:

$$v(k) = \alpha(k) \sum_{n=0}^{N-1} u(n) \cos\frac{(2n+1)\pi k}{2N},\ k = \overline{0, N-1} \tag{4.29}$$

$$u(n) = \sum_{k=0}^{N-1} \alpha(k) v(k) \cos\frac{(2n+1)\pi k}{2N},\ n = \overline{0, N-1} \tag{4.30}$$

unde

$$\alpha(k) = \frac{1}{\sqrt{N}} \begin{cases} 1,\ k = 0 \\ \sqrt{2},\ k \neq 0 \end{cases} \tag{4.31}$$

Una dintre proprietăţile importante ale transformatei cosinus se referă la legătura acesteia cu transformata Fourier: transformata cosinus a unei secvenţe nu este partea reală a transformatei Fourier a aceleiaşi secvenţe. Transformata cosinus se poate obţine prin transformarea Fourier a unei secvenţe rearanjate sau a unei secvenţe dublate şi simetrizate central. Fie N par; şi definim

$$\begin{matrix} \widetilde{u_1}(n) = u(2n) \\ \widetilde{u_1}(N-n-1) = u(2n+1) \end{matrix}, n = \overline{0, \frac{N}{2} - 1} \tag{4.32}$$

$$\widetilde{u_2}(n) = \begin{cases} u(N-n-1),\ 0 \leqslant n < N \\ u(n-N),\ N \leqslant n < 2N \end{cases} \tag{4.33}$$

Aceasta înseamnă că secvenţa $\widetilde{\mathbf{u}_1}$ este $u(0), u(2), u(4), ..., u(N-2), u(N-1), u(n-3), ..., u(3), u(1)$, iar secvenţa $\widetilde{\mathbf{u}_2}$ este $u(N-1), u(N-2), ..., u(1), u(0), u(0), u(1), ..., u(N-2), u(n-1)$. Atunci:

$$COS(\mathbf{u}) = \text{Re}\left[\alpha(k)\exp(-\frac{j\pi k}{2N})\,\text{Fourier}(\widetilde{\mathbf{u}_1})\right] \tag{4.34}$$

$$COS(\mathbf{u}) = \text{Fourier}(\widetilde{\mathbf{u}_2})\exp(-\frac{\pi k}{2N}),\ k = \overline{0, ..., N-1} \tag{4.35}$$

O consecinţă imediată a acestor relaţii este posibilitatea realizării unei transformări cosinus rapide (de complexitate $O(N \log N)$) folosind transformata Fourier rapidă.

O altă proprietate importantă a transformatei cosinus este accea că vectorii bazei cosinus (coloanele matricii $\mathbf{C}$) sunt vectori proprii pentru orice matrice simetrică tridiagonală de forma:

$$\mathbf{Q} = \begin{pmatrix} 1-\rho & -\rho & 0 & ... & ... & 0 \\ -\rho & 1-\rho & -\rho & 0 & ... & 0 \\ 0 & -\rho & 1-\rho & -\rho & ... & 0 \\ ... & ... & ... & ... & ... & ... \\ 0 & ... & 0 & -\rho & 1-\rho & -\rho \\ 0 & ... & ... & 0 & -\rho & 1-\rho \end{pmatrix} \tag{4.36}$$

Dacă notăm cu $\mathbf{c}_k$ coloana k a matricii transformării cosinus, atunci:

$$\mathbf{Q}\mathbf{c}_k = \lambda_k \mathbf{c}_k$$

Această proprietate revelă faptul că transfomata cosinus se comportă precum transformata Karhunen-Loeve pentru orice clasă de semnale a căror matrice de covariaţie este de forma (4.36). Dar această formă este tipică secvenţelor staţionar Markov de ordinul 1, cu parametru de corelaţie mare ($\rho \to 1$), model adeseori folosit în cazul imaginilor. Deci transformata cosinus este similară transformării optime de decorelare Karhunen-Loeve pentru o clasă largă de imagini naturale (alte comentarii referitoare la această proprietate şi la aplicaţiile ei practice se găsesc în secţiunea 7.2.2, referitoare la compresia imaginilor cu transformate).

4.4.2 Transformata sinus

Transformata sinus este o transformată unitară separabilă, caracterizată de $\mathbf{A} = \mathbf{B} = \mathbf{S}$. Elementele matricii $\mathbf{S}$ sunt date de

$$S(k,n) = \sqrt{\frac{2}{N+1}}\sin\frac{\pi(k+1)(n+1)}{N+1},\ k,n = \overline{0, N-1} \tag{4.37}$$

Transformările directă şi inversă a unei secevenţe $\mathbf{u}$ sunt definite ca:

$$v(k) = \sqrt{\frac{2}{N+1}} \sum_{n=0}^{N-1} u(n) \sin \frac{\pi(k+1)(n+1)}{N+1},\ k = \overline{0, N-1} \tag{4.38}$$

$$u(n) = \sqrt{\frac{2}{N+1}} \sum_{k=0}^{N-1} v(k) \sin \frac{\pi(k+1)(n+1)}{N+1},\ n = \overline{0, N-1} \tag{4.39}$$

Una dintre proprietăţile importante ale transformatei sinus se referă la legătura acesteia cu transformata Fourier: transformata sinus a unei secvenţe nu este partea imaginară a transformatei Fourier a aceleiaşi secvenţe. Transformata sinus se poate obţine prin transformarea Fourier a extensiei antisimetrice a secvenţei. Fie N par; şi definim

$$\begin{array}{l} \widetilde{u}(0) = \widetilde{u}(N+1) = 0 \\ \widetilde{u}(n) = -u(N-n) \quad , n = \overline{1, N} \\ \widetilde{u}(N+2+n) = u(n) \end{array} \tag{4.40}$$

Aceasta înseamnă că secvenţa $\widetilde{\mathbf{u}}$ este $0, -u(N-1), -u(N-2), ..., -u(1), -u(0), 0, u(0), u(1), ..., u(N-1)$. Atunci

$$SIN(\mathbf{u}) = \text{Im}\left[-j(-1)^k \,\text{Fourier}(\widetilde{\mathbf{u}})\right] \tag{4.41}$$

O consecinţă imediată a acestei relaţii este posibilitatea realizării unei transformări sinus rapide (de complexitate $O(N \log N)$) folosind transformata Fourier rapidă.

O altă proprietate importantă a transformatei sinus este aceea că vectorii bazei sinus (coloanele matricii $\mathbf{S}$) sunt vectori proprii pentru orice matrice simetrică tridiagonală de forma:

$$\mathbf{Q} = \begin{pmatrix} 1 & -\rho & 0 & ... & 0 \\ -\rho & 1 & 0 & ... & 0 \\ ... & ... & ... & ... & ... \\ 0 & ... & 0 & 1 & -\rho \\ 0 & ... & 0 & -\rho & 1 \end{pmatrix} \tag{4.42}$$

Dacă notăm cu $\mathbf{s}_k$ coloana k a matricii transformării sinus, atunci:

$$\mathbf{Q}\mathbf{s}_k = \lambda_k \mathbf{s}_k$$

Această proprietate revelă faptul că transfomata sinus se comportă precum transformata Karhunen-Loeve pentru orice clasă de semnale a căror matrice de covariaţie este de forma (4.42). Dar această formă este tipică secvenţelor staţionar Markov de ordinul 1, cu parametru de corelaţie $\rho \in [-0.5; 0.5]$.

Capitolul 5

FILTRAREA NELINIARĂ A IMAGINILOR

Operaţiile de filtrare studiate până în prezent au fost caracterizate esenţial (din punct de vedere matematic) de către principiul liniarităţii (sau superpoziţiei). Istoric, acestea au fost primele operaţii utilizate, având un suport teoretic solid şi desigur, având mai multe caracteristici utile. O mare parte a aplicaţiilor curente de prelucrare a imaginilor probabil că se pot rezolva suficient de precis folosind doar operaţii liniare. Limitările operaţiilor liniare de filtrare apar însă în momentul în care se consideră zgomote ce nu au distribuţie normală, şi, mai mult, nu mai sunt aditive.

Exemplul cel mai simplu de astfel de zgomot este zgomotul impulsiv, caracterizat de impulsuri pozitive şi negative de amplitudine maxim posibilă (relativ la numărul de nivele de cuantizare ale imaginii) care afectează prin înlocuire o parte din pixelii imaginii. Aceasta înseamnă că imaginea va fi uniform acoperită de puncte foarte închise (negre) şi foarte deschise (albe), rezultatul fiind ceea ce se numeşte zgomot de tip sare şi piper (vezi figura 5.1). Asemenea erori apar în porţiunile de achiziţie şi transmisiune ale sistemului de prelucrarea şi analiza imaginilor (din cauza erorilor în convertorul analog-digital, din cauza erorilor de înregistrare pe mediile magnetice de stocare sau de pe canalul de transmisiune); este posibil ca asemenea valori să apară şi ca urmare a unui zgomot aditiv caracterizat de o distribuţie "cu coadă lungă" - deci care asigură probabilităţi nenule de apariţie a unor valori mult diferite de medie. Aceste valori aberante (mult mai mari sau mai mici decât valorile normale ale regiunii în care apar) sunt numite "outlier": valori extreme sau erori grosiere.

Dacă o asemena imagine ar fi filtrată liniar (cu un filtru de mediere sau poate cu un filtru de reliefare) rezultatul ar fi o accentuare a punctelor de zgomot şi o corupere a punctelor cu valori corecte din jurul lor.

Zgomotul impulsiv nu este singurul model de degradare neliniară: există zgomote de-

Fig. 5.1: Imagine afectată de zgomot impulsiv; 10% dintre pixeli au valorile modificate

pendente de valoarea imaginii din pixelul afectat, precum şi compuneri neaditive ale zgomotului cu imaginea originală (multiplicativă, convolutivă) [9].

Este deci evident că se impune considerarea şi a altor metode de filtrare, care nu mai respectă principiul superpoziţiei, şi deci sunt neliniare. O clasă esenţială de astfel de metode de filtrare sunt cele bazate pe ordonare - rank order filtering [12]. Ideea esenţială este aceea că operaţia de ordonare plasează valorile aberante (extremale) la capetele şirului de valori (deci într-o zonă bine definită), permiţând astfel identificarea şi eliminarea acestora.

5.1 Filtrarea de ordine

Filtrele de ordine sunt operatori locali: filtrul este definit de o fereastră (mască), care selectează din imaginea de prelucrat un număr de vecini ai pixelului curent (se respectă deci acelaşi model de prelucrare al ferestrei glisante, întâlnit şi la filtrările liniare în domeniul spaţial). Valorile selectate de fereastra de filtrare sunt apoi ordonate crescător; dacă valorile selectate de o fereastră cu n poziţii sunt $\{x_1, x_2, ..., x_n\}$, atunci acelaşi set de valori, ordonate crescător este $\{x_{(1)}, x_{(2)}, ..., x_{(n)}\}$. Valorile $x_{(i)}$ se numesc statistici de ordine de ordinul i şi au proprietatea

$$x_{(1)} \leqslant x_{(2)} \leqslant ... \leqslant x_{(n)} \tag{5.1}$$

În mod evident statistica de ordinul 1, $x_{(1)}$, este valoarea minimă, iar statistica de ordinul n, $x_{(n)}$, este valoarea maximă.

Ieşirea filtrului de ordine de ordin k este statistica de ordinul k, cu $k \in [1; n]$:

$$\text{rank}_k\{x_1, x_2, ..., x_n\} = x_{(k)} \tag{5.2}$$

În acest fel, ieşirea unui filtru de ordine este una dintre valorile selectate de fereastra de filtrare, şi deci nu se crează la ieşire valori noi (aşa cum se întâmplă la filtrarea liniară); acesta este un avantaj atât în ceea ce priveşte păstrarea valorilor originale din imagine, cât şi din punctul de vedere al simplificării calculelor (nu mai sunt necesare operaţii cu numere reale, ce trebuie apoi rotunjite sau truncheate la numere naturale).

Filtrele de ordine nu sunt liniare din cauza operaţiei de ordonare (ce nu respectă principiul superpoziţiei); în general putem scrie:

$$\text{rank}_k\{x_1+y_1, x_2+y_2, ..., x_n+y_n\} \neq \text{rank}_k\{x_1, x_2, ..., x_n\} + \text{rank}_k\{y_1, y_2, ..., y_n\} \quad (5.3)$$

Filtrele de ordine comută însă cu operaţiile de modificare liniară a valorilor:

$$\text{rank}_k\{\alpha x_1+\beta, \alpha x_2+\beta, ..., \alpha x_n+\beta\} = \alpha\,\text{rank}_k\{x_1, x_2, ..., x_n\} + \beta \quad (5.4)$$

Din punct de vedere statistic, interesează funcţiile de distribuţie ale ieşirii filtrului (funcţie de densitate de probabilitate, funcţie de repartiţie) atunci când funcţiile de distribuţie ale valorilor de intrare sunt presupuse cunoscute. Modelul folosit în general se bazează pe ipoteza de independenţă şi distribuţie identică a valorilor selectate de fereastra filtrului (ceea ce conduce, implicit, la ipoteza că valorile pixelilor imaginii sunt variabile aleatoare independente, identic distribuite[1]). Dacă notăm cu $f(x)$ şi $F(x)$ funcţiile de densitate de probabilitate şi respectiv repartiţie a valorilor pixelilor din imaginea ce se filtrează cu un filtru de ordine de ordin k cu fereastră de n elemente, atunci ceea ce interesează este funcţia de densitate de probabilitate a ieşirii filtrului de ordine, $f_k(x)$.

Fie t o valoare oarecare din domeniul de variaţie al variabilelor aleatoare; atunci probabilitatea evenimentului ca ieşirea filtrului de ordine să se găsească în intervalul $[t; t+dt]$ este $f_k(t)dt$. Evenimentul de interes este deci

$$t \leqslant \text{rank}_k\{x_1, x_2, ..., x_n\} \leqslant t+dt \quad (5.5)$$

adică

$$t \leqslant x_{(k)} \leqslant t+dt \quad (5.6)$$

Statisticile de ordine sunt obţinute prin ordonarea crescătoare a valorilor x_i, ordonare crescătoare care poate fi interpretată ca o permutare oarecare a indicilor valorilor ($\{1, 2, ..., n\} \longrightarrow \{i_1, i_2, ..., i_n\}$). În acest caz, evenimentul de interes exprimat de (5.6) este realizat prin mai multe evenimente elementare de tipul:

$$x_{i_1}, x_{i_2}, ..., x_{i_{k-1}} \leqslant t \leqslant x_{i_k} \leqslant t+dt \leqslant x_{i_{k+1}}, ..., x_{i_n} \quad (5.7)$$

Probabilitatea evenimentului elementar din (5.7) este dată de probabilitatea ca $k-1$ valori să fie mai mici decât t (deci $F^{k-1}(t)$), probabilitatea ca $n-k$ valori să fie mai mari

[1] Această ipoteză a mai fost făcută şi pentru deducerea transformării de egalizare a histogramei imaginilor.

ca $t + dt$ (deci $(1 - F(t))^{n-k}$) şi probabilitatea ca o valoare să fie cuprinsă în intervalul $[t; t + dt]$ (deci $f(t)dt$), adică:

$$P = F^{k-1}(t)(1 - F(t))^{n-k} f(t)dt$$

Numărul total de evenimente de tipul (5.7) este nC_{n-1}^{k-1} (n moduri de alege o valoare din cele n, la fiecare dintre aceste alegeri corespunzând C_{n-1}^{k-1} moduri de a alege din cele $n-1$ valori rămase $k-1$ valori care să fie mai mici ca t). Atunci:

$$\Pr\left(t \leqslant x_{(k)} \leqslant t + dt\right) = nC_{n-1}^{k-1} F^{k-1}(t)(1 - F(t))^{n-k} f(t)dt \tag{5.8}$$

şi deci funcţia de densitate de probabilitate a ieşirii filtrului de ordine de ordin k este:

$$f_k(x) = nC_{n-1}^{k-1} F^{k-1}(x)(1 - F(x))^{n-k} f(x) \tag{5.9}$$

În [12] sunt prezentate diferite utilizări ale funcţiei de densitate de probabilitate a ieşirii filtrelor de ordine, mai ales pentru determinarea comportării asimptotice ($n \longrightarrow \infty$) a filtrelor.

Proprietăţile deterministe ale filtrelor de ordine se referă la comportamentul acestora în zonele tipice ale unei imagini: porţiuni netede şi contururi. Filtrele de ordine nu afectează contururile (zonele de pantă abruptă) şi păstează valoarea zonelor uniforme. Aceste comportări au fost extinse la investigarea clasei mai generale de semnale rădăcină - semnale ce nu sunt modificate prin filtrare.

Orice secvenţă $\{x_i\}$ monotonă (crescătoare sau descrescătoare) este un semnal rădăcină pentru filtrele de ordine. În plus, filtrele de ordine comută cu orice funcţie monotonă g:

$$\text{rank}_k\left(g(x_i)\right) = g(\text{rank}_k(x_i))$$

Pe lângă metodele analitice de construire a semnalelor rădăcină [12], în practică, semnalele rădăcină se obţin prin filtrarea repetată a unui semnal oarecare, până la obţinerea invarianţei. Să considerăm spre exemplu secvenţa unidimensională $x = \{0, 1, 1, 3, 1, 3, 2, 3, 3, 2, 1, 1\}$ pe care o vom filtra cu o fereastră de filtrare de dimensiune 3, centrată, cu un filtru de ordine de ordin 2. După prima filtrare se obţine secvenţa $x1 = \{0, 1, 1, 1, 3, 2, 3, 3, 3, 2, 1, 1, 0\}$, iar după a doua filtrare se obţine secvenţa $x2 = \{0, 1, 1, 1, 2, 3, 3, 3, 3, 2, 1, 1, 0\}$. Această secvenţa este invariantă la filtrările următoare şi este deci un semnal rădăcină. Figura 5.4 prezintă secvenţele x, $x1$, $x2$.

5.1.1 Filtrul median

Filtrul median este un filtru de ordine a cărui ieşire este statistica de ordine de ordin central a setului de valori selectate de fereastra de filtrare.

$$\text{median}\{x_1, x_2, ..., x_n\} = \begin{cases} x_{\left(\frac{n+1}{2}\right)} \text{ dacă } n \text{ este impar} \\ \frac{1}{2}x_{\left(\frac{n}{2}\right)} + \frac{1}{2}x_{\left(\frac{n}{2}+1\right)} \text{ dacă } n \text{ este par} \end{cases} \tag{5.10}$$

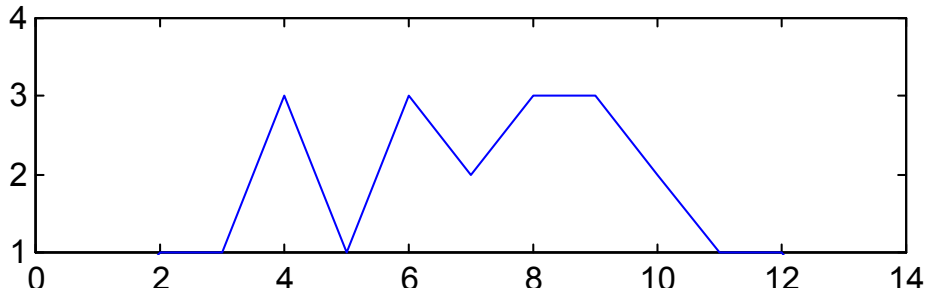


Fig. 5.2: Semnal iniţial oarecare; prin filtrări de ordine repetate acesta se va transforma într-un semnal rădăcină.

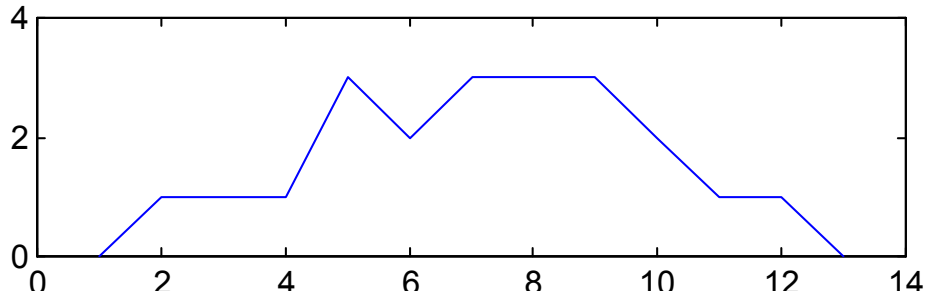


Fig. 5.3: Rezultatul primei filtrări a secvenţei iniţiale.

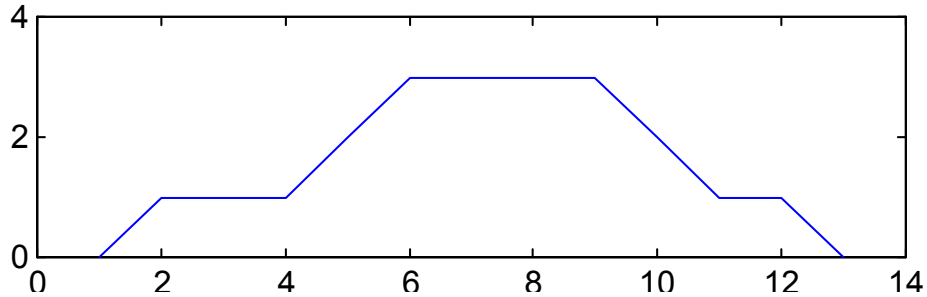


Fig. 5.4: Semnal rădăcină al filtrului de ordine considerat, obţinut după două iteraţii ale filtrării.

Ieşirea filtrului median este deci valoarea din centrul secvenţei ordonate; în cazul secvenţelor (ferestrelor de filtrare) de dimensiune pară, această poziţie centrală nu există, şi atunci este definită ca media aritmetică a valorilor vecine centrului imaginar. În practică, filtrul median se foloseşte doar cu ferestre de filtrare de dimensiune impară, ceea ce conduce la posibilitatea de a scrie:

$$\text{median}\{x_1, x_2, ..., x_n\} = \text{rank}_{\frac{n+1}{2}}\{x_1, x_2, ..., x_n\} \tag{5.11}$$

Aceasta înseamnă că dacă fereastra de filtrare are 3 puncte, medianul este statistica de ordinul 2[2], pentru o fereastră filtrare cu 9 puncte, medianul este statistica de ordinul 5, iar pentru o fereastră de filtrare de 25 de puncte, medianul este statistica de ordinul 13. Figura 5.5 prezintă rezultatul filtrării cu un filtru median cu fereastră pătrată de 3 x 3 puncte a imaginii degradate cu zgomot impulsiv din figura 5.1.

Fig. 5.5: Imagine filtrată cu filtru median cu fereastră de 3 x 3.

Efectele de filtrare a zgomotului impulsiv (de tip sare şi piper) sunt evidente; valorile punctelor de zgomot sunt 0 sau 255 şi deci, după ordonare se vor afla la "capetele" şirului de valori; ieşirea filtrului median, fiind statistica de ordin central, este situată departe de valorile extreme. În imaginea filtrată median (figura 5.5) se observă totuşi existenţa unor puncte albe şi negre (puncte de zgomot) ce nu au putut fi eliminate prin filtrare; spunem că în acele poziţii filtrul a fost străpuns de impulsuri (care deci au ajuns la ieşirea filtrului). Probabilitatea de străpungere a unui filtru median este dependentă de procentul de puncte de zgomot din imagine şi de dimensiunea ferestrei de filtrare folosite. Străpungerea filtrului median se produce atunci când în fereastra de filtrare avem măcar $\frac{n+1}{2}$ impulsuri de zgomot de aceeaşi valoare.

O variantă a filtrului median cu fereastră de filtrare pătrată este filtrul median separabil. Separabilitatea semnifică (ca şi în cazul transformărilor unitare) realizarea operaţiei întâi

[2]Aceasta înseamnă deci că exemplul de construire a semnalului rădăcină prezentat anterior este realizat pentru un filtru median.

pe linii, şi apoi pe coloanele imaginii, pe baza unor ferestre de filtrare liniare. Este evident că rezultatul acestei operaţii de filtrare nu coincide cu rezultatul filtrării mediane cu fereastră pătrată[3] (figura 5.6 prezintă rezultatul filtrării mediane separabile cu ferestre de dimensiune 3 a imaginii cu zgomot impulsiv din figura 5.1; rezultatul filtrării mediane cu fereastră pătrată 3 x 3 este prezentat în figura 5.5).

Fig. 5.6: Rezultatul filtrării cu un filtru median separabil.

Se remarcă faptul că această structură de filtrare, deşi prezintă avantaje din punctul de vedere al timpului de calcul necesar, se străpunge mai uşor decât filtrul median cu fereastră pătrată din care a derivat.

5.1.2 Filtrele de ordine ponderate şi structurile multietaj

Atât filtrele liniare cât şi filtrele de ordine se bazează pe acelaşi principiu al ferestrei glisante. Prelucrările realizate asupra valorilor selectate de această fereastră de filtrare sunt evident diferite, dar se poate totuşi remarca faptul că structura de filtrare liniară permite o flexibilitate mai mare (se pot realiza un număr infinit de filtre liniare pentru o aceeaşi fereastră de filtrare, prin varierea coeficienţilor de ponderare). Singurul factor de reglaj al filtrelor de ordine este ordinul statisticii selectate la ieşire[4].

Este evident că ponderarea filtrelor de ordine nu se poate face prin multiplicarea valorilor selectate de fereastra de filtrare cu diferiţi coeficienţi, aşa ca în cazul filtrelor liniare.

[3] Filtrul de minim (filtrul de ordine de ordin 1) şi filtrul de maxim (filtrul de ordine de ordin n) sunt singurele filtre de ordine separabile.

[4] Zamperoni a propus adaptarea ordinului statisticii folosite ca ieşire a filtrului de ordine prin $k = \left[\frac{1}{2} + \sum_{i=1}^{n} \frac{x_{(i)} - x_{(1)}}{x_{(n)} - x_{(1)}}\right]$.

Ponderarea are ca scop întărirea influenţei anumitor pixeli din fereastra de filtrare (de exemplu pixelul central - curent prelucrat - care ar fi de aşteptat ca să aibă o influenţă mai puternică asupra rezultatului filtrării decât vecinii săi). Având în vedere că rezultatul filtrării (deci valoarea unei anumite statistici) este determinată în urma ordonării, modalitatea de a întări influenţa unei anume valori este de a o repeta de mai multe ori, crescând astfel probabilitatea de a o regăsi ca statistica de ordin dorit. Aşadar, ponderarea filtrelor de ordine se referă la repetarea de un număr fixat de ori a valorilor selectate de fereastra de filtrare. Mulţimea de valori ce rezultă se numeşte multiset. Ponderile w_i ataşate fiecărei poziţii de filtrare vor fi numere naturale nenule, ce exprimă numărul de repetiţii al valorii corespunzătoare în multiset.

Spre exemplu să considerăm porţiunea de imagine selectată de o fereastră de filtrare de dimensiune 3 x 3, ale cărei valori sunt $\begin{pmatrix} 1 & 3 & 3 \\ 2 & 2 & 1 \\ 4 & 3 & 5 \end{pmatrix}$; statistica mediană ce corespunde acestei situaţii este 3 (setul ordonat de valori fiind 1, 1, 2, 2, 3, 3, 3, 4, 5). Dacă considerăm acum filtrul median ponderat cu masca $W = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 3 & 2 \\ 1 & 2 & 1 \end{pmatrix}$, multisetul rezultant este 1, 3, 3, 3, 2, 2, 2, 2, 2, 1, 1, 4, 3, 3, 5 (sau ordonat 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 4, 5) ceea ce face ca medianul să fie 2 (şi deci o valoare mai apropiată de valoarea originală din punctul curent prelucrat). Masca de ponderare W semnifică faptul că valoarea centrală este repetată de 3 ori, valorile de pe poziţiile vecinilor imediaţi (orizontali şi verticali) sunt repetate de câte 2 ori, iar valorile de pe poziţiile vecinilor de pe diagonală sunt considerate o singură dată. Filtrul de ordine rezultat se poate scrie ca:

$$\text{rank}_k(x_i \diamond w_i) = x_{(k)}$$

Este evident că multisetul de valori conţine $\sum_{j=1}^{n} w_i$ termeni, şi deci ordinul statisticii maxime este $\sum_{j=1}^{n} w_i$, iar ordinul statisticii mediane este $\frac{1}{2}\left(\sum_{j=1}^{n} w_i + 1\right)$ (presupunând un număr impar de valori în multiset). Un caz particular destul de des utilizat este acela al filtrelor de ordine central ponderate, în care toţi coeficienţii de ponderare (repetiţie) sunt 1, cu excepţia coeficientului ce corespunde pixelului curent prelucrat, ce are o valoare mai mare ca 1.

Structurile multietaj grupează câteva filtre de ordine (median, minim, maxim) aplicate succesiv, cu ferestre de filtrare diferite. Cele mai răspândite structuri sunt cele de tip median multietaj (median din median pe orizontală, median pe verticală şi valoarea curentă), reprezentat în figura 5.8 şi max/min-median (maxim sau minim din medianele pe verticală, orizontală şi cele două diagonale ale ferestrei de filtrare pătrate centrate în punctul curent prelucrat), reprezentat în figura 5.7.

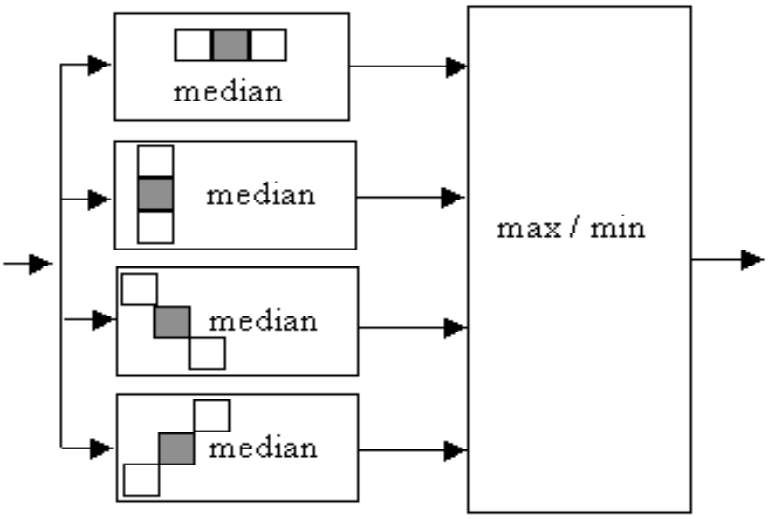


Fig. 5.7: Structră multietaj de tip max / min - median, bazată pe o fereastră de filtrare de dimensiune 3.

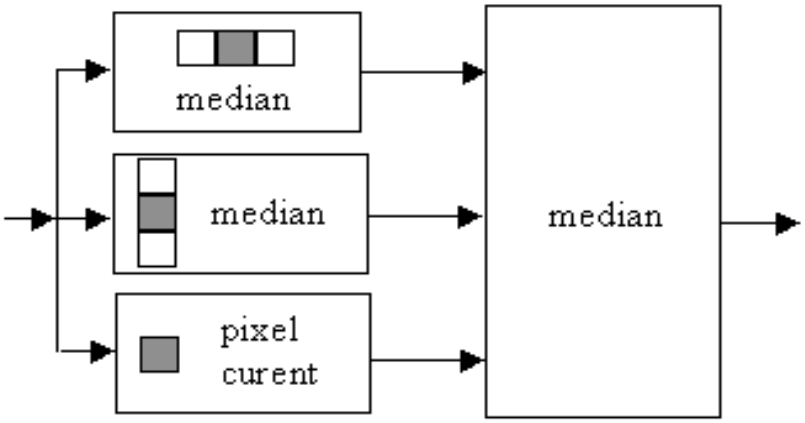


Fig. 5.8: Structură de filtrare de tip median - median, bazată pe o fereastră de filtrare pătrată de dimensiune 3.

5.2 Filtre de ordine de domeniu

Filtrele de ordine de domeniu (filtre LUM – *Lower-Upper-Middle* [18]) introduc o posibilitate de adaptare şi reglare a filtrelor de ordine.

Valoarea y de ieşire a filtrului LUM de netezire este definită ca:

$$y = \begin{cases} x_{(k)}, \text{ dacă } x^* < x_{(k)} \\ x_{(n-k+1)}, \text{ dacă } x^* > x_{(n-k+1)} \\ x^*, \text{ în rest} \end{cases} \tag{5.12}$$

unde x^* este valoarea ce corespunde originii ferestrei de filtrare cu n poziţii, iar k este parametrul de reglaj al filtrului. Valoarea k este un întreg din intervalul $[0; \frac{n+1}{2}]$; dacă $k = 0$ filtrul de comportă ca un filtru trece tot ($y = x^*$), iar dacă $k = \frac{n+1}{2}$, filtrul este un filtru median ($y = x_{(\frac{n+1}{2})}$).

Valoarea y de ieşire a filtrului LUM de conturare este definită ca:

$$y = \begin{cases} x_{(l)}, \text{ dacă } x_{(l)} < x^* < \frac{x_{(l)}+x_{(n-l+1)}}{2} \\ x_{(n-l+1)}, \text{ dacă } \frac{x_{(l)}+x_{(n-l+1)}}{2} < x^* < x_{(n-l+1)} \\ x^*, \text{ în rest} \end{cases} \tag{5.13}$$

unde x^* este valoarea ce corespunde originii ferestrei de filtrare cu n poziţii, iar l este parametrul de reglaj al filtrului. Valoarea k este un întreg din intervalul $[0; \frac{n+1}{2}]$; dacă $l = 0$ filtrul de comportă ca un filtru de reliefare extremă, iar dacă $l = \frac{n+1}{2}$, filtrul este un filtru trece tot ($y = x^*$).

5.3 L-filtre

L-filtrele [12] au fost introduse din dorinţa de a oferi o mai mare flexibilitate operaţiilor de filtrare neliniară bazate pe ordonare; pasul firesc de dezvoltare a fost combinarea tuturor statisticilor de ordine în valoarea de la ieşirea filtrului. Ieşirea unui L-filtru este o combinaţie liniară a statisticilor de ordine:

$$L(\{x_1, x_2, ..., x_n\}) = \sum_{i=1}^{n} w_i x_{(i)} \tag{5.14}$$

Este evident că prin varierea coeficienţilor w_i se pot obţine diferite filtre, inclusiv filtrele de ordine (un singur coeficient egal cu 1, iar restul nuli). Coeficienţii celor mai uzuale L-filtre sunt prezentaţi în tabelul 5.1

Filtru	Coeficienţi
Filtru de ordine de ordin k, rank_k	$a_k = 1$, $a_i = 0$, $i \neq k$
Filtru de mediere aritmetică	$a_i = 1/n$
Filtru de mijloc	$a_1 = a_n = 0.5$, $a_i = 0$, $i \notin \{1, n\}$
Filtru de cvasi-mijloc	$a_k = a_{n+1-k} = 0.5$, $a_i = 0$, $i \notin \{k, n+1-k\}$
Medie α-reglabilă	$a_k = 1/n(1-2\alpha)$, $i \in [\alpha n + 1; n - \alpha n]$ $a_i = 0$, $i \notin [\alpha n + 1; n - \alpha n]$

Tabel 5.1: Coeficienţii celor mai uzuale L-filtre.

Se poate demonstra cu uşurinţă că coeficienţii L-filtrelor trebuie să îndeplinească aceleaşi condiţii de normare ca şi coeficienţii filtrelor liniare: să aibă suma 1 pentru un filtru de netezire (conform (3.8)) şi suma 0 pentru un filtru de reliefare (conform (3.9)). Toate filtrele prezentate în tabelul 5.1 sunt deci filtre de netezire, al căror scop este mărirea uniformităţii regiunilor imaginii prin eliminarea diferitelor zgomote suprapuse acesteia. Tabelul 5.2 prezintă filtrele cele mai potrivite pentru eliminarea diferitelor tipuri de zgomote.

Tip zgomot	Filtru
impulsiv (sare şi piper)	filtru de ordine (median, maxim, minim)
gaussian	filtru de mediere aritmetică
impulsiv + gaussian	filtru de tip medie α-reglabilă
uniform	filtru de mijloc

Tabel 5.2: L-filtre utilizate pentru reducerea diferitelor tipuri de zgomote.

Un exemplu de filtru de reliefare (cu comportare derivativă, care să satisfacă deci (3.9)) este gradientul morfologic – diferenţa între valoarea maximă şi minimă din fereastra de filtrare curentă:

$$grad\{x_1, x_2, ..., x_n\} = x_{(n)} - x_{(1)}$$

Folosirea unui asemenea operator produce o imagine în care valoarea fiecărui pixel este invers proporţională cu gradul de uniformitate (netezime) al vecinătăţii acestuia (vezi figura 5.9).

5.4 Aspecte de implementare

Implementarea filtrelor neliniare de ordine sau a L-filtrelor urmează aceleaşi principii ca şi oricare tehnică de tip fereastră glisantă (ca filtrarea liniară) în ceea ce priveşte

Fig. 5.9: Operatorul gradient morfologic aplicat imaginii filtrate din figura 5.5.

problemele legate de realizarea formei ferestrei de filtrare şi efectele de margine. Ceea ce este particular filtrelor din această categorie este etapa de ordonare a valorilor selectate de fereastra de filtrare în fiecare poziţie.

În teoria algoritmilor au fost dezvoltaţi numeroşi algoritmi de sortare (ordonare crescătoare), a căror complexitate de calcul variază de la $O(n^2)$ (bubble sort, sortare prin determinare succesivă de extreme), $O(n \log n)$ (sortare prin interclasare sau inserţie) până la $O(n)$ (pentru determinare doar a extremelor şi a statisticii mediane). Trebuie însă remarcat că o metodă de sortare de complexitate $O(n^2)$ nu este neapărat mai neeficientă decât o metodă de sortare $O(n)$: complexitatea unui algoritm exprimă comportarea asimptotică a numărului de operaţii necesare, în condiţiile în care dimensiunea caracteristică n a setului de date ce se prelucrează este foarte mare (la limită $n \longrightarrow \infty$). Sortările folosite în implementarea filtrelor neliniare de ordine prelucrează seturi mici de date (să nu uităm că rareori o fereastră de filtrare depăşeşte dimensiunea de 25 de pixeli), şi deci complexitatea $O()$ nu este un criteriu valid de alegere a unui algoritm de sortare.

Codul Matlab ce urmează implementează un L-filtru cu o fereastră pătrată de dimensiune 3 x 3, centrată; ponderile filtrului sunt stocate în vectorul w (1 linie, 9 coloane, prima poziţie corespunzând coeficientului statisticii minime). L-filtrul este aplicat imaginii *in* (de dimensiuni *nrlin* şi *nrcol*); efectele de margine sunt evitate prin neprelucrarea primei şi ultimei linii, respectiv coloane a imaginii. Rezultatul este înscris în imaginea *out*.

```
[nrlin,nrcol]=size(in);
out=in;
for i=2:nrlin-1
 for j=2:nrcol-1
  temp=in(i-1:i+1,j-1:j+1);
  temp=sort(temp(:));
```

```
  out(i,j)=fix(w*temp);
 end
end
```

Se remarcă folosirea funcţiei standard *sort* ce realizează operaţia de ordonare crescătoare a valorilor; ieşirea L-filtrului este produsul scalar al vectorului de coeficienţi ai filtrului cu vectorul statisticilor de ordine.

În cele ce urmează vom considera un exemplu de implementare a unui filtru cu ordonare după rang relizat cu o fereastră centrată de 3 x 3 pixeli, pentru filtrarea unei imagini cu *NRLIN* linii şi *NRCOL* coloane; prima şi ultima linie şi coloană sunt identice cu cele din imaginea ce se filtrează (remarcaţi buclele *for* de alocare a spaţiului pentru imaginea filtrată şi iniţializarea valorilor acesteia).

```
unsigned char temp[9],dummy;
boolean sortat;
img_out=(int**)malloc(NRLIN*sizeof(int*));
for (i=0;i<NRLIN;i++)
  img_out[i]=(int*)malloc(NRCOL*sizeof(int));
for (i=0;i<NRCOL;i++)
{ img_out[0][i]=img[0][i];
  img_out[NRLIN-1][i]=img[NRLIN-1][i];}
for (i=0;i<NRLIN;i++)
{ img_out[i][0]=img[i][0];
  img_out[i][NRCOL-1]=img[i][NRCOL-1];}
for (i=1;i<NRLIN-1;i++)
 for (j=1;j<NRCOL-1;j++)
 { for (k=0;k<9;k++)
    temp[k]=img[i-1+k/3][j-1+k%3];
   sortat=false;
   while (! sortat) do
   {  sortat=true;
      for (k=0;k<8;k++)
      if (temp[k]>temp[k+1])then
       { dummy=temp[k+1];
         temp[k+1]=temp[k];
         temp[k]=dummy;
         sortat=false;
        }
    }
   img_out[i][j]=temp[5];
}
```

Valorile selectate de fereastra de filtrare sunt stocate în vectorul de 9 poziţii *temp*. Citirea acestor valori se face printr-un ciclu, bazându-se pe introducerea unei relaţii de echivalenţă între ordinea de baleiaj a ferestrei şi poziţia valorilor în vector (economia nu este de timp de execuţie ci de timp de scriere a codului). Această alocare este echivalentă cu liniile de cod:

```
temp[0]=img[i-1][j-1];temp[1]=img[i-1][j];
temp[2]=img[i-1][j+1];temp[3]=img[i][j-1];
temp[4]=img[i][j];temp[5]=img[i][j+1];
temp[6]=img[i+1][j-1];temp[7]=img[i+1][j];
temp[8]=img[i+1][j+1];
```

Sortarea implementată este de tip bubble sort; fiecare poziţie a vectorului ce conţine valorile extrase de fereastra de filtrare este comparată cu poziţia următoare, iar dacă ordinea crescătoare nu este respectată, cele două valori sunt interschimbate; indicatorul *sortat* indică efectuarea a măcar unei interschimbări, şi deci necesitatea reluării verificării ordinii. Filtrul de ordine folosit este medianul (se preia în imaginea rezultat statistica de ordine cu numărul 5, deci statistica mediană).

Structura de program folosită anterior realizează ordonarea vectorului de valori pentru fiecare poziţie a ferestrei de filtrare; o serioasă economie de timp se poate realiza dacă se ţine seama că la mutarea ferestrei de filtrare de la un pixel al imaginii la un pixel vecin acestuia, se modifică un număr relativ mic de valori (restul rămânând neschimbate). Aceasta poate conduce la ideea păstrării unei părţi a setului de valori ordonate, în care să se intercaleze valorile noi.

O altă variantă de determinare a statisticilor de ordine se bazează pe folosirea histogramei valorilor din fereastra de filtrare. Histograma (2.10) reprezintă numărul de puncte ce au o anumită valoare şi este echivalentă cu funcţia de densitate de probabilitate a unei variabile aleatoare. Pentru aceeaşi varibilă aleatoare există însă şi funcţia de repartiţie – primitiva funcţiei de densitate de probabilitate; aşadar putem asocia oricărei histograme h o histogramă cumulativă (5.15):

$$H(i) = \sum_{j=0}^{i} h(j), \text{ cu } i = 0, 1, ..., L-1 \tag{5.15}$$

Histograma cumulativă în punctul i va reprezenta deci numărul de puncte (pixeli) a căror valoare (nivel de gri) este mai mică decât i. Determinarea valorii statisticilor de ordine pentru un set de valori pe baza histogramei acestora este imediată. Să considerăm următorul exemplu: setul de valori selectate de fereastra de filtrare este {1, 2, 3, 3, 4, 0, 1, 1, 2, 1, 1, 0, 0, 1, 2, 7, 7, 6, 0, 1, 6, 6, 5, 5, 0}(valori din intervalul 0–7). Histograma acestui set este $h = \begin{pmatrix} 5 & 7 & 3 & 2 & 1 & 2 & 3 & 2 \end{pmatrix}$ iar histograma cumulativă este $H = \begin{pmatrix} 5 & 12 & 15 & 17 & 18 & 20 & 23 & 25 \end{pmatrix}$. Atunci statisticile de ordine de ordin 1–5

($H(0)$) au valoarea 0, statisticile de ordine de ordin 6 ($H(0)+1$) – 12 ($H(1)$) au valoarea 1, statisticile de ordine de ordin 13 ($H(1)+1$) – 15 ($H(2)$) au valoarea 2, statisticile de ordine de ordin 16 ($H(2)+1$) – 17 ($H(3)$) au valoarea 3, statistica de ordine de ordin 18 ($H(3)+1 = H(4)$) are valoarea 4, statisticile de ordine de ordin 19 ($H(4)+1$) – 20 ($H(5)$) au valoarea 5, statisticile de ordine de ordin 21 ($H(5)+1$) – 23 ($H(6)$) au valoarea 6, statisticile de ordine de ordin 24 ($H(6)+1$) – 25 ($H(7)$) au valoarea 7. Regula de decizie generală este: statisticile de ordine de ordin $H(j-1)+1$ până la $H(j)$ au valoarea j, cu $j = 0, 1, 2, ..., L-1$ şi $H(-1) = 0$.

Deci, pentru o statistică oarecare de ordin r, trebuie identificat numărul j astfel ca:

$$H(j-1)+1 \leqslant r \leqslant H(j) \tag{5.16}$$

Problemele ce apar la o asemenea implementare sunt legate în primul rând de timpul de determinare a numărului j ce satisface (5.16), timp ce este direct proporţional cu numărul de nivele de gri din imagine (şi deci lungimea vectorului histogramă sau histogramă cumulativă). Se poate însă observa că histograma într-o fereastră de filtrarea va selecta relativ puţine valori diferite, şi atunci histograma va fi un vector rar (*sparse*), cu multe intrări nule, iar histograma cumulativă va avea, corespunzător, multe porţiuni constante. În aceste condiţii, o reprezentare mai eficientă (atât ca memorie ocupată, cât şi ca timp de căutare) este memorarea doar a tranziţiilor (poziţie, valoare) din histograma cumulativă, adică doar memorarea histogramei.

Capitolul 6

ELEMENTE DE MORFOLOGIE MATEMATICĂ

În mod tradiţional, prelucrarea semnalelor multidimensionale (şi a imaginilor în particular) a fost bazată pe exploatarea conceptelor şi teoriei sistemelor liniare şi a transformatei Fourier ([9], [2]). Deşi aceste abordări clasice sunt justificate şi dau rezultate în multe cazuri, aplicarea lor este limitată de problema fundamentală impusă de semnalele de tip imagine: modul de reprezentare a formei sau structurii geometrice existente într-un semnal.

Morfologia matematică, dupa cum indică şi numele (*morphos* – formă, *logos* – ştiinţă, deci ştiinţa formelor), realizează o abordare axată pe formă a prelucrării imaginilor. Folosită corespunzător, morfologia matematică conduce la prelucrări ce simplifică structura imaginii, păstrând caracteristicile esenţiale de formă şi eliminând irelevanţele.

Ideea de bază a oricărei prelucrări morfologice este considerarea imaginii ca un ansamblu (mulţime, reuniune de părţi) asupra căruia se aplică transformări a căror esenţă este comparaţia cu mulţimi (ansambluri) mai simple, numite *elemente structurante*. Scopul acestor transformări este extragerea de forme mai simple din formele iniţiale (complexe) ale imaginii.

Morfologia matematică este utilizată ca o abordare naturală a proceselor de vedere artificială, deoarece trăsăturile şi respectiv identificarea obiectelor sunt corelate cu forma. Principalele aplicaţii sunt în domeniile roboticii, microscopiei electronice, imagisticii biomedicale, telemetriei, inspecţiei automate a produselor, analizei de scenă. Aplicaţiile industriale sunt impulsionate şi de continua dezvoltare şi îmbunatatire a arhitecturilor de calcul ce implementează transformări morfologice.

6.1 Transformări morfologice de bază

6.1.1 Transformarea Hit or Miss

Transformarea *Hit or Miss* a fost introdusă în [14] şi reprezintă poate cea mai primară şi evidentă exemplificare a conceptului de studiu al formei. Orice formă poate fi considerată ca o reuniune de componente (blocuri, regiuni, scheme) individuale independente, plasate în diverse poziţii; a recunoaşte forma implică identificarea locală a părţilor sale componente şi deci o operaţie simplă de potrivire de măşti (*pattern matching*).

Rezultatul aplicării acestei transformări de identificare (numită uneori şi transformarea "totul sau nimic") este o mulţime ale cărei puncte satisfac criteriul de identificare a unei vecinătăţi cu masca aplicată.

Transformarea *Hit or Miss* se defineşte pe baza unei partiţii $(B_1,\ B_2)$ a elementului structurant B: $B = B_1 \cup B_2$ şi $B_1 \cap B_2 = \emptyset$ ca:

$$A \circledast B = \{\mathbf{x}|(B_1)_\mathbf{x} \subseteq A\} \cap \{\mathbf{x}|(B_2)_\mathbf{x} \subseteq A^C\} \tag{6.1}$$

În (6.1) A este mulţimea căreia i se aplică transformarea, A^C este complementara mulţimii A, B este elementul structurant şi $(B_i)_\mathbf{x}$ este mulţimea B_i translatată cu vectorul $\mathbf{x}$. Trebuie remarcat faptul că oricărui element structurant trebuie să i se ataşeze o origine. Figura 6.1 prezintă o exemplificare a transformării *Hit or Miss*.

Se poate observa că această transformare produce ca rezultat o mulţime de puncte ce satisfac concomitent un set de condiţii de tipul $(B_i)_\mathbf{x} \subseteq A_i$, unde $\{A_i\}$ formează o partiţie a spaţiului.

Este evident că $\{A, A^C\}$ formează o partiţie, dar aceasta nu este singura posibilă. Anumite aplicaţii practice pot impune însă situaţii mai complexe, a căror rezolvare depăşeşte cadrul morfologiei binare [14]. Astfel se pot cita aşa numitele "cazuri": cazul petrografic (provenit din analiza hidrocarburilor şi a mineralelor), în care p componente împart exact spaţiul (fără a lăsa locuri libere) şi cazul geografic, în care p componente nu ocupă întreg spaţiul şi se pot întrepătrunde (cazul zonelor de pădure cu diferite specii de copaci). Figura 6.2 prezintă o astfel de transformare.

Transformarea *Hit or Miss*, în forma prezentată, prezintă un interes mai mult teoretic, datorită situării sale la baza construcţiei morfologiei matematice. Se va arăta însă că este posibilă exprimarea acesteia şi în funcţie de transformările morfologice fundamentale larg utilizate (erodarea şi dilatarea).

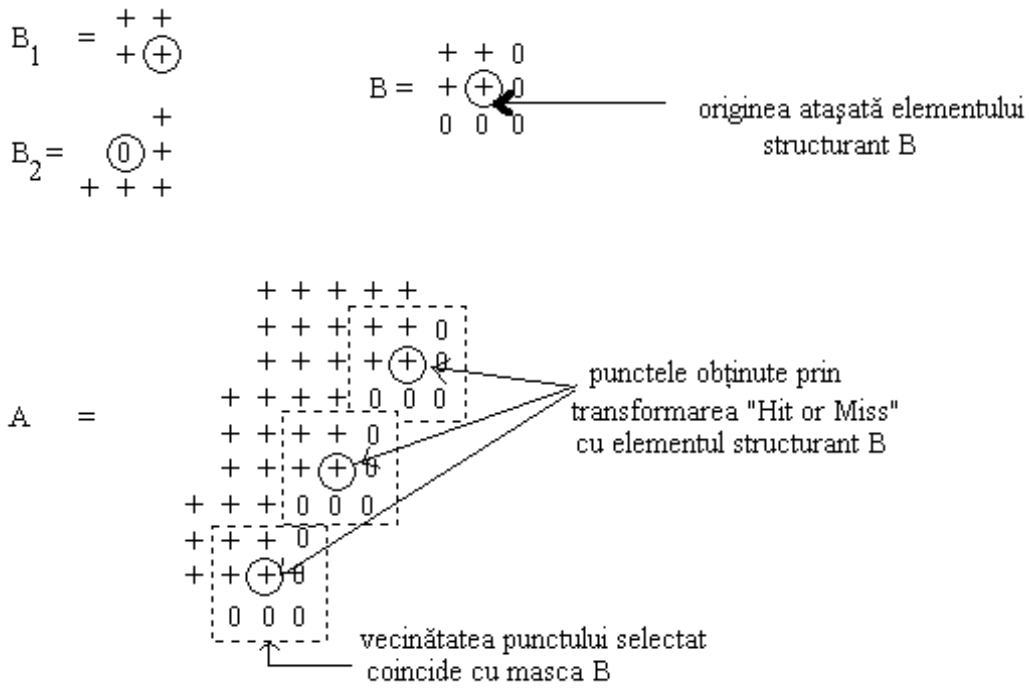


Fig. 6.1: Exemplificare a transformării de identificare a configuraţiilor locale (Hit or Miss); în particular exemplul prezintă determinarea punctelor de tip "colţ dreapta-jos".

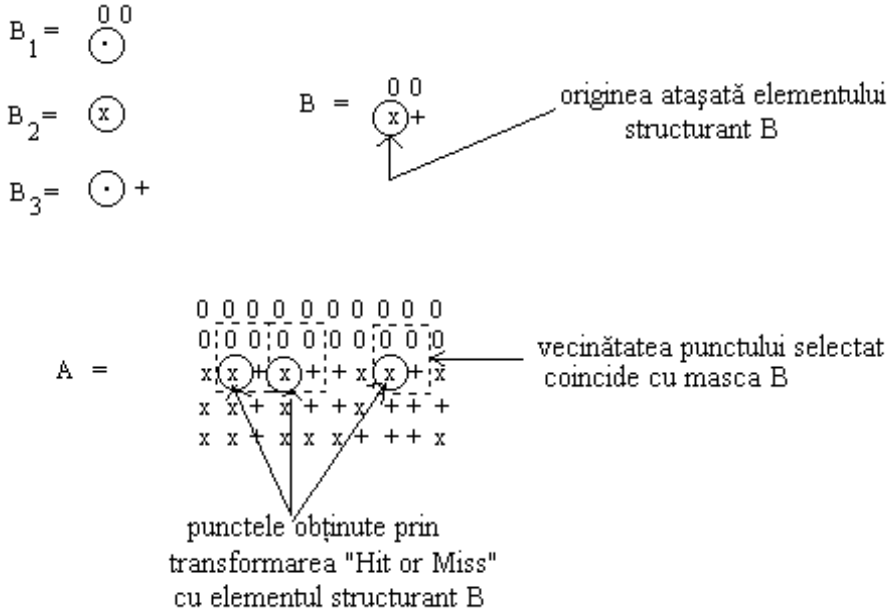


Fig. 6.2: Exemplu de transformare Hit or Miss extinsă pentru o partiţie cu trei mulţimi a spaţiului imaginii.

6.1.2 Erodarea morfologică

Erodarea morfologică a mulţimii A prin elementul structurant B se defineşte ca mulţimea punctelor (elementelor) cu care se poate translata elementul structurant astfel încât acesta să fie inclus în mulţimea de prelucrat A:

$$A \ominus B = \{\mathbf{x} | B_{\mathbf{x}} \subseteq A\} \tag{6.2}$$

Erodare morfologică a mulţimii A este transformata Hit or Miss a mulţimii cu un element structurant $B = B_1$ ($B_2 = \emptyset$):

$$A \ominus B = \{\mathbf{x} | (B_1)_{\mathbf{x}} \subseteq A\} \cap \{\mathbf{x} | (\emptyset)_{\mathbf{x}} \subseteq A^C\} = \{\mathbf{x} | (B)_{\mathbf{x}} \subseteq A\}$$

Această relaţie de definiţie se mai poate exprima ca:

$$\begin{aligned} A \ominus B &= \{\mathbf{x} | B_{\mathbf{x}} \subseteq A\} = \{\mathbf{x} | \forall \mathbf{b} \in B, \exists \mathbf{a} \in A \text{ astfel încât } \mathbf{b} + \mathbf{x} = \mathbf{a}\} = & (6.3) \\ &= \{\mathbf{x} | \forall \mathbf{b} \in B, \exists \mathbf{a} \in A \text{ astfel încât } \mathbf{x} = \mathbf{a} - \mathbf{b}\} = \bigcap_{\mathbf{b} \in B} A_{-\mathbf{b}} = \bigcap_{\mathbf{b} \in B^S} A_{\mathbf{b}} & (6.4) \end{aligned}$$

Figura 6.3 prezintă rezultatul erodării unor mulţimi discrete, iar figura 6.4 prezintă rezultatul erodării unor mulţimi continue.

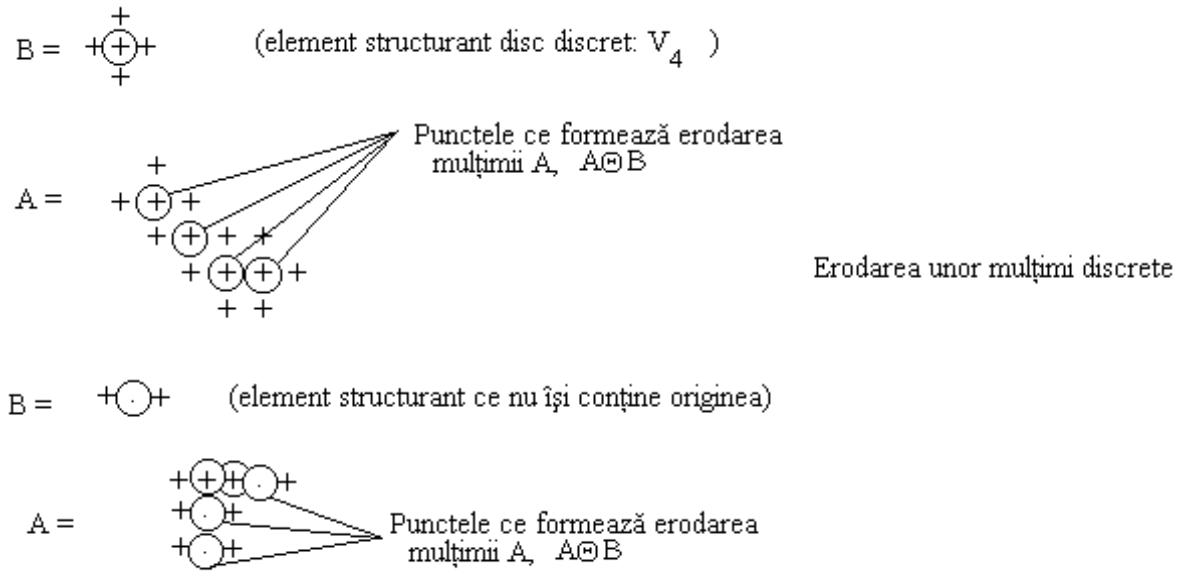


Fig. 6.3: Exemple de erodare a unor mulţimi discrete cu diferite elemente structurante, ce îşi conţin (sau nu) originea.

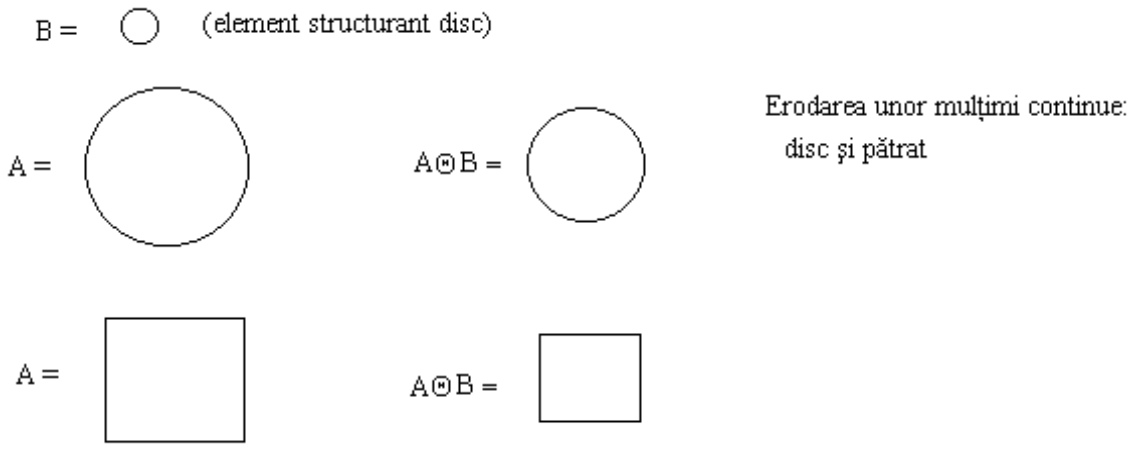


Fig. 6.4: Rezultatul erodării unor mulţimi continue cu un element structurant de tip disc (originea elementului structurant este centrul discului).

6.1.3 Dilatarea morfologică

Dilatarea morfologică a mulţimii A cu elementul structurant B se defineşte ca mulţimea punctelor (elementelor) cu care se poate translata elementul structurant astfel încât acesta să aibă puncte comune cu mulţimea de prelucrat A:

$$A \oplus B = \{\mathbf{x} | B_{\mathbf{x}} \cap A \neq \emptyset\} \tag{6.5}$$

Erodare morfologică a mulţimii A este complementara transformatei Hit or Miss a mulţimii cu un element structurant $B = B_2$ ($B_1 = \emptyset$):

$$(A \oplus B)^C = \{\mathbf{x} | (B_1)_{\mathbf{x}} \subseteq A\} \cap \{\mathbf{x} | (B_2)_{\mathbf{x}} \subseteq A^C\} = \{\mathbf{x} | (\emptyset)_{\mathbf{x}} \subseteq A\} \cap \{\mathbf{x} | (B_2)_{\mathbf{x}} \subseteq A^C\} = \{\mathbf{x} | (B)_{\mathbf{x}} \subseteq A^C\}$$

$$A \oplus B = \{\mathbf{x} | (B)_{\mathbf{x}} \subseteq A^C\}^C = \{\mathbf{x} | (B)_{\mathbf{x}} \nsubseteq A^C\} = \{\mathbf{x} | B_{\mathbf{x}} \cap A \neq \emptyset\}$$

Relaţia de definiţie mai poate fi exprimată şi ca:

$$A \oplus B = \{\mathbf{x} | B_{\mathbf{x}} \cap A \neq \emptyset\} = \{\mathbf{x} | \exists \mathbf{b} \in B, \exists \mathbf{a} \in A \text{ astfel încât } \mathbf{b} + \mathbf{x} = \mathbf{a}\} = \tag{6.6}$$

$$= \{\mathbf{x} | \exists \mathbf{b} \in B, \exists \mathbf{a} \in A \text{ astfel încât } \mathbf{x} = \mathbf{a} - \mathbf{b}\} = \bigcup_{\mathbf{b} \in B} A_{-\mathbf{b}} = \bigcup_{\mathbf{b} \in B^S} A_{\mathbf{b}} \tag{6.7}$$

Figura 6.5 prezintă rezultatul erodării unor mulţimi discrete, iar figura 6.6 prezintă rezultatul dilatării unor mulţimi continue.

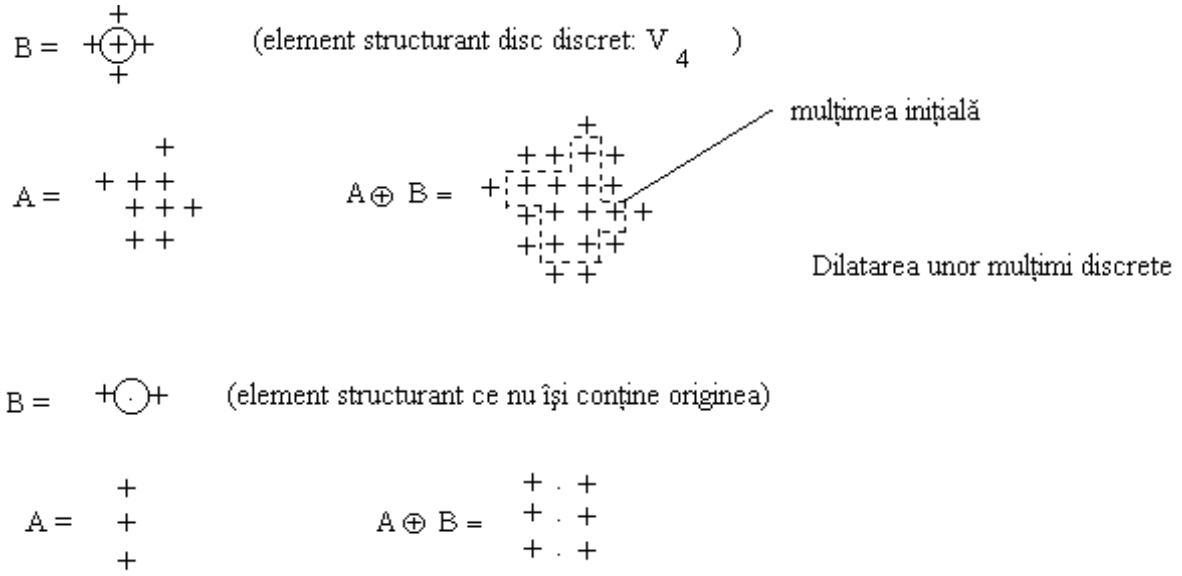


Fig. 6.5: Exemple de dilatare a unor mulţimi discrete cu diferite elemente structurante, ce îşi conţin (sau nu) originea.

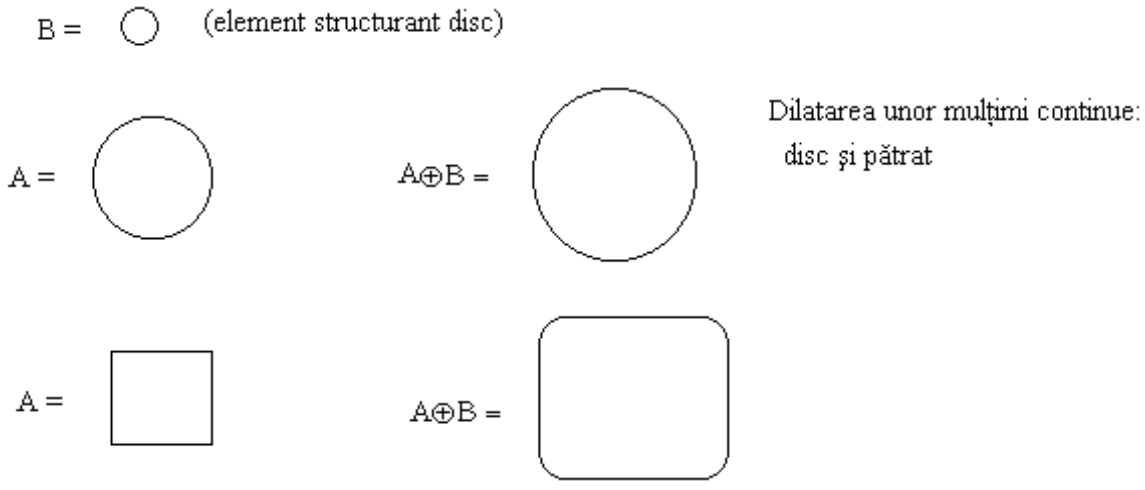


Fig. 6.6: Rezultatul dilatării unor mulţimi continue cu un element structurant de tip disc (originea elementului structurant este centrul discului).

6.1.4 Proprietăţile erodării şi dilatării

Se observă că, în general, efectul operaţiei de dilatare este de a mări obiectul; acesta creşte, "se umflă", corespunzător formei şi dimensiunii elementului structurant. Efectul erodării este, după cum am văzut, o micşorare a obiectului. Modificarea dimensiunii obiectului este strict determinată de forma elementului structurant: un element structurant simetric (disc, segment de dreaptă centrat în origine) provoacă o modificare simetrică a dimensiunilor; dacă elementul structurant nu este simetric, efectele se vor manifesta asupra obiectului pe direcţia elementului structurant şi în sens contrar acestuia. Efectele erodării şi dilatării vor fi discutate în continuare, pe baza proprietăţilor matematice ale acestora.

Dilatarea şi erodarea formează o pereche de operaţii duale:

$$(A \oplus B)^C = A^C \ominus B \text{ şi } (A \ominus B)^C = A^C \oplus B \tag{6.8}$$

Demonstraţia este imediată:

$A \oplus B = \{\mathbf{x} | B_\mathbf{x} \cap A \neq \emptyset\}$, de unde rezultă

$$(A^C \oplus B)^C = \{\mathbf{x} | B_\mathbf{x} \cap A^C \neq \emptyset\}^C = \{\mathbf{x} | B_\mathbf{x} \cap A^C = \emptyset\} = \{\mathbf{x} | B_\mathbf{x} \subseteq A\} = A \ominus B \tag{6.9}$$

şi $A \ominus B = \{\mathbf{x} | B_\mathbf{x} \subseteq A\}$ de unde rezultă

$$(A^C \ominus B)^C = \{\mathbf{x} | B_\mathbf{x} \subseteq A^C\}^C = \{\mathbf{x} | B_\mathbf{x} \nsubseteq A^C\} = \{\mathbf{x} | B_\mathbf{x} \cap A \neq \emptyset\} = A \oplus B \tag{6.10}$$

Interpretarea dualităţii ca obţinerea aceloraşi efecte, dar asupra unor mulţimi complementare, este corectă: dacă dilatarea va mări obiectul (mulţimea), aceasta înseamnă că va micşora în acelaşi timp fundalul (complementara mulţimii), deci va fi echivalentă cu o erodare a fundalului. Dacă erodarea micşorează obiectul (mulţimea), aceasta înseamnă o mărire simultană a fundalului (complementara mulţimii), deci o dilatare a acestuia.

Un caz particular ce poate apare este acela al elementelor structurante ce nu conţin originea. Folosind un asemenea element structurant, rezultatele operaţiilor morfologice vor fi "translatate" faţă de poziţia la care ar fi fost de aşteptat să apară.

Legătura între translaţia elementului structurant (faptul că acesta nu conţine originea poate fi interpretat ca o deplasare) sau a mulţimii (obiectului) care se prelucrează şi deplasarea rezultatului operaţiilor morfologice este dat de proprietăţile de invarianţă la translaţie:

$$A_t \oplus B = (A \oplus B)_t \text{ şi } A_t \ominus B = (A \ominus B)_\mathbf{t} \tag{6.11}$$

$$A \oplus B_\mathbf{t} = (A \oplus B)_{-\mathbf{t}} \text{ şi } A \ominus B_\mathbf{t} = (A \ominus B)_{-\mathbf{t}} \tag{6.12}$$

Aceste proprietăţi de invarianţă la translaţie (translaţia obiectului şi translaţia elementului structurant) pot avea mai multe interpretări. În primul rând, invarianţa la translaţie asigură faptul că forma rezultată prin dilatarea sau erodarea unei mulţimi este aceeaşi, indiferent de poziţia mulţimii prelucrate. În al doilea rând, relaţiile enunţate asigură suficienţa considerării doar a elementelor structurante ce conţin originea; un element structurant ce nu conţine originea poate fi obţinut prin translaţie dintr-un element structurant ce conţine originea; mulţimea (forma) rezultată în urma prelucrării va trebui translatată în sens opus elementului structurant.

În acelaşi grup de proprietăţi de invarianţă a operaţiilor morfologice se încadrează şi invarianţa la scalare (omotetie). Dacă λ este parametrul nenul de scalare, atunci:

$$\frac{1}{\lambda}(\lambda A \oplus B) = A \oplus \frac{1}{\lambda}B \text{ şi } \frac{1}{\lambda}(\lambda A \ominus B) = A \ominus \frac{1}{\lambda}B \tag{6.13}$$

Într-adevăr,

$$\frac{1}{\lambda}(\lambda A \oplus B) = \frac{1}{\lambda}(\lambda A + B^S) = \frac{1}{\lambda}\{\mathbf{x}|\mathbf{x} = \lambda\mathbf{a} - \mathbf{b}, \mathbf{a} \in A, \mathbf{b} \in B\} = \{\mathbf{x}|\mathbf{x} = \mathbf{a} - \frac{1}{\lambda}\mathbf{b}, \mathbf{a} \in A, \mathbf{b} \in B\} =$$

$$= A + \frac{1}{\lambda}B^S = A \oplus \frac{1}{\lambda}B$$

$$\frac{1}{\lambda}(\lambda A \ominus B) = \frac{1}{\lambda}(\lambda A^C \oplus B)^C = (\frac{1}{\lambda}(\lambda A^C \oplus B))^C = (A^C \oplus \frac{1}{\lambda}B)^C = A \ominus \frac{1}{\lambda}B$$

Relaţiile de invarianţă la scalare afirmă că rezultatul unei transformări morfologice aplicate unei versiuni scalate a formei A este identic cu aceeaşi scalare aplicată rezultatului transformării morfologice a formei A prin acelaşi element structurant scalat invers.

Proprietăţile de monotonie a unei transformări nu pot fi neglijate la descrierea acesteia. Atât dilatarea cât şi erodarea sunt crescătoare faţă de mulţimea ce se prelucrează (forma): dacă $A_1 \subseteq A_2$, atunci

$$A_1 \oplus B \subseteq A_2 \oplus B \text{ şi } A_1 \ominus B \subseteq A_2 \ominus B \tag{6.14}$$

Monotonia faţă de elementul structurant folosit este diferită: dilatarea este crescătoare, iar erodarea este descrescătoare: dacă $B_1 \subseteq B_2$, atunci:

$$A \oplus B_1 \subseteq A \oplus B_2 \text{ şi } A \ominus B2 \subseteq A \ominus B1 \tag{6.15}$$

Aceste proprietăţi subliniază corectitudinea interpretării dilatării ca o adăugare, ca o îngroşare, iar a erodării ca o subţiere; grosimea stratului adăugat sau îndepărtat este

dată de elementul structurant. Deci cu cât elementul structurant este mai mare, cu atât corpul se va mări mai mult în urma dilatării sau se va micşora mai mult în urma erodării.

În general, dilatarea este extensivă, adică:

$$A \subseteq A \oplus B. \tag{6.16}$$

O condiţie suficientă pentru ca această proprietate să fie adevărată este ca elementul structurant să conţină originea, $\{0_n\} \in B$. Atunci

$$A \oplus B = \bigcup_{\mathbf{b} \in B^S} A_{\mathbf{b}} = A \cup (\bigcup_{\mathbf{b} \in B^S - \{0_n\}} A_{\mathbf{b}}) \supseteq A \tag{6.17}$$

Condiţia ca elementul structurant să conţină originea nu este însă şi o condiţie necesară (a se vedea figura 6.7).

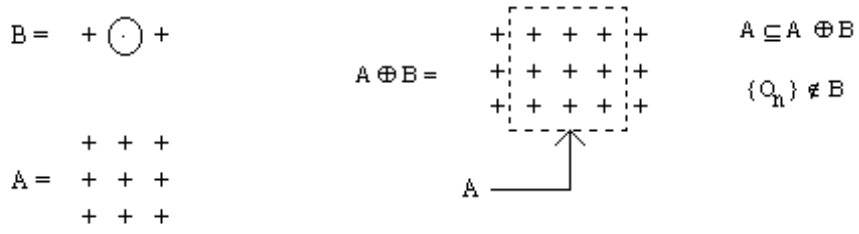


Fig. 6.7: Deşi elementul structurant nu îşi conţine originea, rezultatul dilatării include forma originală.

În general erodarea este antiextensivă, adică:

$$A \ominus B \subseteq A \tag{6.18}$$

O condiţie suficientă pentru ca această proprietate să fie adevărată este ca elementul structurant să conţină originea, $\{0_n\} \in B$. Atunci

$$A \ominus B = \bigcap_{\mathbf{b} \in B^S} A_{\mathbf{b}} = A \cap (\bigcap_{\mathbf{b} \in B^S - \{0_n\}} A_{\mathbf{b}}) \subseteq A \tag{6.19}$$

Condiţia ca elementul structurant să conţină originea nu este însă şi o condiţie necesară (a se vedea figura 6.8).

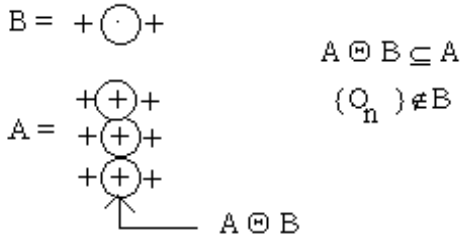


Fig. 6.8: Deşi elementul structurant nu îşi conţine originea, rezultatul erodării este inclus în mulţimea iniţială.

Dilatarea este pseudocomutativă, proprietate care se exprimă ca

$$A \oplus B = (B \oplus A)^S \tag{6.20}$$

$$A \oplus B = A + B^S = (A^S + B)^S = (B + A^S) = (B \oplus A)^S \tag{6.21}$$

Proprietatea de asociativitate a dilatării se scrie ca

$$A \oplus (B \oplus C) = (A \oplus B) \oplus C^S \tag{6.22}$$

Într-adevăr:

$$A \oplus (B \oplus C) = A + (B \oplus C)^S = A + (B + C^S)^S = A + B^S + C = (A \oplus B) + C = (A \oplus B) \oplus C^S \tag{6.23}$$

Erodarea nu este nici comutativă (sau pseudocomutativă) şi nici asociativă. Mai mult, "asociativitatea" erodării se scrie ca:

$$(A \ominus B) \ominus C = A \ominus (B \oplus C) \tag{6.24}$$

Într-adevăr:

$$(A \ominus B) \ominus C = ((A \ominus B)^C \oplus C)^C = ((A^C \oplus B) \oplus C)^C = (A^C \oplus (B \oplus C))^C = A \ominus (B \oplus C) \tag{6.25}$$

Proprietăţile de asociativitate ale dilatării şi erodării pot fi interpretate şi ca o "descompunere" a elementului structurant; dacă un element structurant poate fi considerat ca descompus prin dilatare în $X = B \oplus C$, atunci operaţia morfologică efectuată prin elementul structurant respectiv este echivalentă cu operaţiile morfologice prin "părţile" descompunerii elementului structurant, efectuate succesiv, adică:

$$A \oplus X = (A \oplus B) \oplus C^S \text{ şi } A \ominus X = (A \ominus B) \ominus C$$

Dilatarea şi erodarea sunt transformări (operaţii) ce nu păstrează numărul de conexităţi (numărul de obiecte). Rezultatul unei erodări poate fi o mulţime vidă (dacă elementul structurant nu poate fi inclus în formă pentru nici o poziţie) sau mai multe forme (componente conexe). Rezultatul dilatării unei mulţimi formate din mai multe componente conexe poate fi o singură componentă conexă; "găurile" conţinute într-o formă pot fi umplute.

Aceste proprietăţi ale erodării şi dilatării sunt fundamentul folosirii practice a acestora: prin erodare se pot elimina dintr-o mulţime componentele conexe ce sunt mai mici decât elementul structurant folosit (sau sunt mult diferite de acesta din punctul de vedere al formei) - aplicaţiile ce ar putea folosi o asemenea comportare sunt separarea obiectelor după formă şi dimensiune şi eliminarea zgomotului suprapus scenei. Prin dilatare se pot grupa într-o singură entitate obiecte apropiate şi se pot umple golurile înglobate în obiecte.

Dilatarea şi erodarea nu sunt operaţii inversabile (nu admit o transformare inversă). Cu atât mai mult, dilatarea nu este inversa erodării şi erodarea nu este inversa dilatării; exemplele prezentate în figurile 6.9 şi 6.10 ilustrează această afirmaţie.

$$(A \ominus B) \oplus B \neq A \text{ şi } (A \oplus B) \ominus B \neq A \tag{6.26}$$

O altă categorie de proprietăţi se referă la distributivitatea operaţiilor morfologice faţă de operaţiile cu mulţimi. Pentru dilatare avem:

$$(A \cup B) \oplus C = (A \oplus C) \cup (B \oplus C) \tag{6.27}$$

$$(A \oplus C) \cup (B \oplus C) = (\bigcup_{\mathbf{c} \in C^S} A_{\mathbf{c}}) \cup (\bigcup_{\mathbf{c} \in C^S} B_{\mathbf{c}}) = (\bigcup_{\mathbf{c} \in C^S} A_{\mathbf{c}} \cup B_{\mathbf{c}}) = \bigcup_{\mathbf{c} \in C^S} (A \cup B)_{\mathbf{c}} = (A \cup B) \oplus C$$

$$A \oplus (B \cup C) = (A \oplus B) \cup (A \oplus C) \tag{6.28}$$

$$(A \oplus B) \cup (A \oplus C) = (A + B^S) \cup (A + C^S) = (B^S + A) \cup (C^S + A) = (\bigcup_{\mathbf{a} \in A} B^S_{\mathbf{a}}) \cup (\bigcup_{\mathbf{a} \in A} C^S_{\mathbf{a}}) =$$

$$= \bigcup_{\mathbf{a} \in A} (B^S_{\mathbf{a}} \cup C^S_{\mathbf{a}}) = \bigcup_{\mathbf{a} \in A} (B \cup C)^S_{\mathbf{a}} = (B \cup C)^S + A = A + (B \cup C)^S = A \oplus (B \cup C)$$

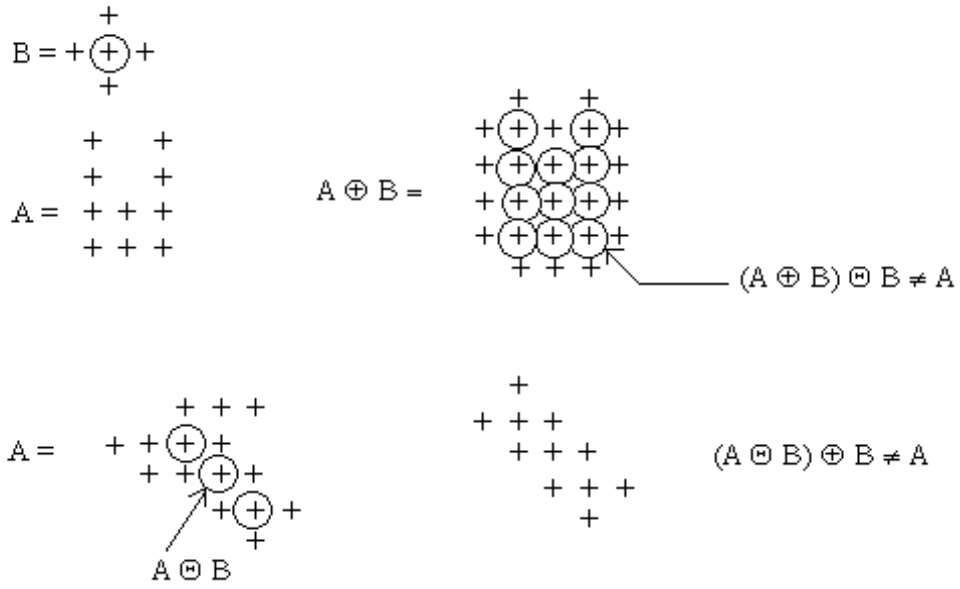


Fig. 6.9: Dilatarea şi erodarea nu sunt transformări inverse una alteia; ilustrare pentru mulţimi discrete.

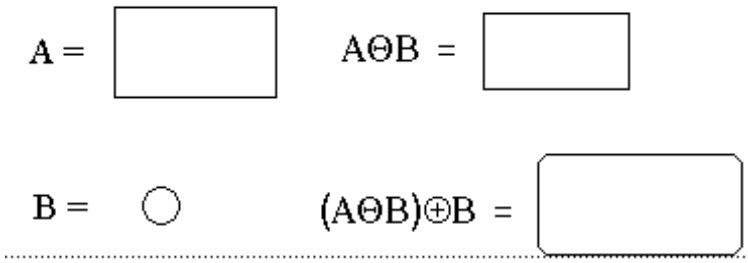


Fig. 6.10: Dilatarea şi erodarea nu sunt transformări inverse una alteia; ilustrare pentru mulţimi continue.

Această din urmă relaţie poate fi interpretată şi ca o descompunere prin reuniune a elementului structurant $X = B \cup C$, caz în care operaţia morfologică efectuată prin elementul structurant respectiv este echivalentă cu operaţiile morfologice prin "părţile" descompunerii elementului structurant, efectuate succesiv, adică:

$$A \oplus X = (A \oplus B) \cup (A \oplus C) \tag{6.29}$$

Erodarea are proprietăţi asemănătoare dilatării faţă de intersecţia mulţimilor:

$$A \ominus (B \cup C) = (A \ominus B) \cap (A \ominus C) \tag{6.30}$$

$$(A \ominus B) \cap (A \ominus C) = (\bigcap_{\mathbf{t} \in B^S} A_{\mathbf{t}}) \cap (\bigcap_{\mathbf{t} \in C^S} A_{\mathbf{t}}) = \bigcap_{\mathbf{t} \in B^S \cup C^S} A_{\mathbf{t}} = \bigcap_{\mathbf{t} \in (B \cup C)^S} A_{\mathbf{t}} = A \ominus (B \cup C)$$

$$(A \cap B) \ominus C = (A \ominus C) \cap (B \ominus C)$$

$$(A \ominus C) \cap (B \ominus C) = (\bigcap_{\mathbf{c} \in C^S} A_{\mathbf{c}}) \cap (\bigcap_{\mathbf{c} \in C^S} B_{\mathbf{c}}) = \bigcap_{\mathbf{c} \in C^S} (A_{\mathbf{c}} \cap B_{\mathbf{c}}) = \bigcap_{\mathbf{c} \in C^S} (A \cap B)_{\mathbf{c}} = (A \cap B) \ominus C$$

Înrudite cu proprietăţile de distributivitate deja enunţate, există o serie de inegalităţi (relaţii de incluziune) de aceeaşi natură:

$$A \oplus (B \cap C) \subseteq (A \oplus B) \cap (A \oplus C) \text{ şi } (B \cap C) \oplus A \subseteq (B \oplus A) \cap (C \oplus A)$$

$$A \ominus (B \cap C) \supseteq (A \ominus B) \cup (A \ominus C) \text{ şi } (B \cap C) \ominus A \supseteq (B \ominus A) \cap (C \ominus A)$$

Această tratare [aproape] exhaustivă a proprietăţilor operaţiilor morfologice de bază (dilatare, erodare) se regăseşte în majoritatea monografiilor de morfologie matematică, ca de exemplu în [14], [7]. Deşi, în esenţă, proprietăţile tratate sunt aceleaşi, se impun câteva precizări.

Notaţiile introduse în acest text sunt diferite pentru operaţiile Minkowski (adunare $+$ şi scădere $-$) şi pentru operaţiile morfologice (dilatare $\oplus$ şi erodare $\ominus$). Legătura dintre aceste operaţii este dată de:

$$A \oplus B = A + B^S \text{ şi } A \ominus B = A - B^S \tag{6.31}$$

În majoritatea lucrărilor de morfologie însă, nu se face nici o distincţie de notaţie între aceste operaţii, ceea ce poate genera confuzii. Chiar în condiţiile păstrării aceloraşi notaţii, există abordări diferite privind definirea operaţiilor morfologice. În [7] dilatarea şi erodarea sunt identice cu adunarea, respectiv scăderea Minkowski (deci fără simetrizarea elementului structurant). Avantajul unei asemenea formalizări este existenţa proprietăţilor de comutativitate şi asociativitate a dilatării (determinate direct de comutativitatea şi asociativitatea adunării Minkowski). În plus, rezultatul dilatării este o "creştere" a formei pe direcţia şi în sensul elementului structurant folosit.

Dezavantajul acestei abordări este acela că trebuie redefinit conceptul de dualitate 6.8 (dacă se doreşte păstrarea proprietăţii fundamentale de dualitate a dilatării şi erodării), după cum urmează:

$$Dual(Op(A,B))|_{B\ parametru} = (Op(A^C, B^S))^C \tag{6.32}$$

În [14] este preferată definirea dilatării unei mulţimi ca adunarea Minkowski a mulţimii cu elementul structurant simetrizat.

6.1.5 Aspecte de implementare

Din punct de vedere practic (al implementării), elementul structurant poate fi interpretat analog suportului ferestrei de filtrare folosită pentru orice operaţie bazată pe principiul ferestrei glisante: cu valorile selectate din imagine "se face ceva". Va trebui însă introdus un mecanism de specificare a formei ferestrei de filtrare (elementului structurant) care să permită modificarea relativ simplă a acesteia (puterea morfologiei matematice constă în alegerea a diferite tipuri de elemente structurante optime unei anumite prelucrări). Convenţia de reprezentare a imaginilor binare este de a avea asociate valori nule punctelor de fundal şi valori pozitive (1 sau 255) punctelor obiect[1]. În aceste condiţii, putem găsi o echivalenţă intuitivă simplă operaţiilor morfologice de erodare şi dilatare. Rezultatul operaţiei de erodare într-un pixel este nenul dacă şi numai dacă elementul structurant plasat cu originea în acel punct este inclus în forma de prelucrat, şi deci, dacă şi numai dacă toate valorile selectate de elementul structurant sunt nenule; aceasta înseamnă că putem implementa operaţia de erodare printr-o operaţie de minim. Rezultatul operaţiei de dilatare într-un pixel este nul dacă şi numai dacă elementul structurant plasat cu originea în acel punct nu are nici un punct comun cu forma de prelucrat, şi deci, dacă şi numai dacă toate valorile selectate de elementul structurant sunt nule; aceasta înseamnă că putem implementa operaţia de dilatare printr-o operaţie de maxim. În cele ce urmează prezentăm codul pentru operaţia de erodare.

```
# define SIZE_MAX_ES=16;
```

[1]Reprezentarea grafică a acestei imagini, realizată cu tabela de nivele de gri normală va afişa deci obiecte albe pe un fundal negru.

```
int min,min_i,max_i,min_j,max_j;
integer es[SIZE_MAX_ES][2];
min_i=0;max_i=0,min_j=0,max_j=0;
for (k=0;k<dim_es;k++) {
  if (min_i > es[k][1])
    min_i=es[k][1];
  if (max_i < es[k][1])
    max_i=es[k][1];
  if (min_j > es[k][2])
    min_j=es[k][2];
  if (max_j < es[k][2])
    max_j=es[k][2]; }
for (i=-min_i;i<NRLIN-max_i;i++)
 for (j=-min_j;j<NRCOL-max_j;j++) {
   min=MAXINT;
   for (k=0;k<dim_es;k++)
     if (min > img[i+es[k][1]][j+es[k][2]]) then
       min=img[i+es[k][1]][j+es[k][2]];
   img_out[i][j]=min;  }
```

Elementul structurant folosit este reprezentat prin matricea de întregi *es*, având cel mult 16 linii şi 2 coloane. Fiecare linie a matricii corespunde unui punct din elementul structurant, reprezentat prin coordonatele faţă de originea elementului structurant: *es[k][1]* şi *es[k][2]*. De exemplu, elementul structurant orizontal, centrat, având trei puncte este reprezentat prin $es = [0\ -1; 0\ 0; 0\ 1]^T$. Numărul de puncte ce compun elementul structurant este *dim_ es*. Prima buclă *for* determină dimensiunile dreptunghiului de încadrare a elementului structurant, pentru a putea evita efectele de margine la prelucrarea imaginii.

6.2 Transformări morfologice derivate

Vom numi operaţie (transformare) morfologică derivată operaţia morfologică construită ca o combinaţie de transformări de bază: erodări, dilatări şi operaţii clasice ansambliste.

6.2.1 Deschiderea şi închiderea

După cum s-a arătat, erodarea şi dilatarea nu sunt transformări inverse una alteia (deci alternarea lor va produce un rezultat diferit de mulţimea originală ce a fost prelucrată). Această observaţie conduce la ideea utilizării unor iterări ale operaţiilor morfologice de

bază, obţinând astfel operaţii mai complexe, ale căror proprietăţi le fac mai adecvate utilizării în scopuri practice.

Deschiderea morfologică a mulţimii A prin elementul structurant B se defineşte ca erodarea mulţimii cu elementul structurant respectiv, urmată de dilatarea cu elementul structurant simetrizat:

$$A \bigcirc B = (A \ominus B) \oplus B^S \tag{6.33}$$

Închiderea morfologică a mulţimii A prin elementul structurant B se defineşte ca dilatarea mulţimii cu elementul structurant respectiv, urmată de erodarea cu elementul structurant simetrizat:

$$A \bullet B = (A \oplus B) \ominus B^S \tag{6.34}$$

Proprietatea de bază a deschiderii şi închiderii morfologice este aceea că sunt transformări duale una alteia (proprietate ce derivă din dualitatea blocurilor constituente de bază, dilatarea şi erodarea). Această proprietate permite interpretarea rezultatelor unei operaţii şi deducerea proprietăţilor acesteia pe baza caracteristicilor dualei sale.

$$(A \bigcirc B)^C = A^C \bullet B \text{ şi } (A \bullet B)^C = A^C \bigcirc B \tag{6.35}$$

Demonstraţia acestor proprietăţi este imediată:

$$(A^C \bigcirc B)^C = ((A^C \ominus B) \oplus B^S)^C = ((A \oplus B)^C \oplus B^S)^C = (A \oplus B) \ominus B^S = A \bullet B$$

$$(A^C \bullet B)^C = ((A^C \oplus B) \ominus B^S)^C = ((A \ominus B)^C \ominus B^S)^C = (A \ominus B) \oplus B^S = A \bigcirc B$$

În mod evident, rezultatul unei deschideri sau al unei închideri este diferit de mulţimea ce a fost prelucrată. Relaţia dintre rezultatul prelucrării şi mulţimea iniţială este dată de proprietăţile de extensivitate ale transformărilor.

Deschiderea morfologică este antiextensivă, adică $A \bigcirc B \subseteq A$

Închiderea morfologică este extensivă, adică $A \subseteq A \bullet B$

Deci, pentru a sintetiza, putem afirma că:

$$A \bigcirc B \subseteq A \subseteq A \bullet B \tag{6.36}$$

Relaţiile pot fi interpretate ca o modificare sigură a mulţimii: prin deschidere se vor elimina o parte dintre elementele mulţimii ce se prelucrează, iar prin închidere se adaugă elemente noi mulţimii.

Proprietatea de idempotenţă a operaţiilor introduce o limitare a modificărilor: mulţimea obţinută după o deschidere sau închidere este invariantă la repetarea operaţiei:

$$(A \bullet B) \bullet B = A \bullet B \text{ şi } (A \bigcirc B) \bigcirc B = A \bigcirc B \tag{6.37}$$

Relaţiile se demonstrează folosind proprietăţile deja enunţate ale deschiderii şi închiderii (extensivitate) şi proprietăţile de monotonie ale operaţiilor morfologice de bază.

Închiderea şi deschiderea moştenesc o parte dintre proprietăţile operaţiilor morfologice de bază: invarianţa la translaţie şi la scalare, monotonia faţă de mulţimea prelucrată şi faţă de elementul structurant folosit. Din punctul de vedere al acestor proprietăţi, deschiderea se comportă analog erodării iar închiderea are o comportare analoagă dilatării.

$$A_{\mathbf{t}} \bigcirc B = (A \bigcirc B)_{\mathbf{t}} \text{ şi } A_{\mathbf{t}} \bullet B = (A \bullet B)_{\mathbf{t}}$$

$$\tfrac{1}{\lambda}(\lambda A \bigcirc B) = A \bigcirc \tfrac{1}{\lambda} B \text{ şi } \tfrac{1}{\lambda}(\lambda A \bullet B) = A \bullet \tfrac{1}{\lambda} B$$

$$\text{Dacă } A_1 \subseteq A_2 : A_1 \bigcirc B \subseteq A_2 \bigcirc B \text{ şi } A_1 \bullet B \subseteq A_2 \bullet B.$$

$$\text{Dacă } B_1 \subseteq B_2 : A \bigcirc B_2 \subseteq A \bigcirc B_1 \text{ şi } A \bullet B_1 \subseteq A \bullet B_2$$

În ceea ce priveşte proprietăţile legate de comportarea faţă de translaţie a operatorilor de deschidere şi închidere, merită subliniat faptul că proprietatea este identică cu cea a erodării şi dilatării doar la translaţia mulţimii prelucrate (rezultatul unei deschideri sau închideri a unei mulţimi este acelaşi, indiferent de poziţia spaţială a mulţimii). În cazul translatării elementului structurant, rezultatul operaţiei este acelaşi, invariant la translaţie (ca rezultat a iterării erodării şi dilatării cu elemente structurante simetrice).

Pentru realizarea efectivă a operaţiilor de deschidere şi închidere este importantă exprimarea acestora ca operaţii la nivelul elementelor mulţimilor ce se prelucrează (şi nu ca o iterare de operaţii morfologice de bază). Deschiderea mai poate fi exprimată şi ca mulţimea elementelor structurante translatate ce sunt incluse în mulţimea de prelucrat:

$$A \bigcirc B = \bigcup_{B_{\mathbf{x}} \subseteq A} B_{\mathbf{x}} \tag{6.38}$$

Închiderea mai poate fi exprimată şi ca mulţimea punctelor pentru care toate elementele structurante translatate ce le conţin au puncte comune cu mulţimea de prelucrat:

$$A \bullet B = \bigcap_{B_{\mathbf{x}} \cap A \neq \emptyset} B_{\mathbf{x}} \tag{6.39}$$

Pe baza acestor exprimări se pot deduce şi efectele practice ale deschiderii şi închiderii asupra formelor (mulţimilor). Prin deschidere, formele mai mici ca elementul structurant folosit vor fi eliminate, se lărgesc golurile înglobate în obiecte, contururile sunt netezite prin teşirea convexităţilor iar obiectele unite prin istmuri sunt separate. Datorită proprietăţii de dualitate, închiderea va avea aceleaşi efecte asupra fundalului (complementarei obiectelor) pe care le are deschiderea asupra mulţimilor. Închiderea umple golurile înglobate în obiecte (dacă aceste găuri sunt mai mici decât elementul structurant folosit), netezeşte contururile formelor prin umplerea concavităţilor şi uneşte obiectele foarte apropiate (umple "strâmtorile").

Efectele operaţiilor de deschidere şi închidere pot fi considerate ca analoage efectelor unei filtrări de netezire a formelor şi eliminare a zgomotului (zgomot interpretat ca obiecte şi găuri de mici dimensiuni). În [14], în cadrul teoriei algebrice a morfologiei matematice, un filtru este definit ca o operaţie crescătoare şi idempotentă. Trebuie făcută deci distincţia între filtrul algebric şi filtrul obişnuit, în sensul teoriei prelucrării semnalelor.

6.2.2 Filtrele alternate secvenţial

Filtrele morfologice sunt transformari neliniare ale semnalului, care modifică local caracteristici geometrice (de formă). Teoria algebrică generală a filtrelor [14] defineşte o transformare ca filtru dacă aceasta este crescătoare şi idempotentă, ceea ce înseamnă deci că deschiderea şi închiderea sunt filtre, în timp ce erodarea şi dilatarea nu sunt filtre.

Dacă o imagine (mulţime) este filtrată prin deschideri după elemente structurante (discuri de rază r) crescătoare, aceasta este echivalent cu a aplica deschiderea după elementul structurant cu raza cea mai mare. Totuşi, se pot construi filtre ce actionează asupra obiectelor şi detaliilor în mod gradat, pe masura iterării: aceste sunt filtrele alternate secvenţial, ce alternează deschideri şi închideri după elemente structurante de dimensiune crescătoare:

$$FAS(A) = (((((A \bigcirc B) \bullet B) \bigcirc 2B) \bullet 2B) \bigcirc 3B) \bullet ... \tag{6.40}$$

sau

$$FAS(A) = (((((A \bullet B) \bigcirc B) \bullet 2B) \bigcirc 2B) \bullet 3B) \bigcirc ... \tag{6.41}$$

6.2.3 Operatori morfologici de contur

O problema curentă a prelucrărilor de imagini este extragerea punctelor de contur. Cazul în care imaginea este binară constituie o simplificare a metodelor şi calculelor. Păstrând

interpretarea pixelilor ca fiind puncte ale obiectului sau ale fundalului, punctele de contur sunt acele puncte de obiect ce au cel puţin un punct de fundal vecin. În general, vecinătatea folosită este cea de tip disc unitar, a cărei formă va depinde puternic de tipul de metrică folosită.

Există trei extractoare morfologice de contur: conturul interior (6.42), conturul exterior (6.43) şi gradientul morfologic (6.44).

$$\delta A = A - A \ominus B \tag{6.42}$$

$$\Delta A = A \oplus B - A \tag{6.43}$$

$$grad\ A = A \oplus B - A \ominus B \tag{6.44}$$

6.3 Extinderea morfologiei matematice la nivele de gri

Pâna în acest moment s-au prezentat operaţiile morfologice clasice, adică aplicate asupra unor seturi (mulţimi). Acestea prezintă limitarea implicită în aplicarea numai asupra unei categorii particulare de imagini, şi anume cele a căror structură poate fi uşor şi imediat asociată unor mulţimi, adică imaginile binare. Pentru extinderea operaţiilor morfologice la funcţii se vor construi transformări ce permit trecerea de la o reprezentare de tip funcţie la o reprezentare de tip mulţime.

În cele ce urmează ne vom referi la o submulţime A a spaţiului discret n-dimensional, adică $A \subseteq \mathbf{Z}^n$. Elementele mulţimii A sunt puncte, reprezentate prin vectorul n-dimensional al coordonatelor, $\mathbf{x} \in A$, cu $\mathbf{x} = (x_1, x_2, ..., x_n)$. Ordinea coordonatelor este arbitrară.

Vom numi primele $n-1$ coordonate ale unui punct $\mathbf{x}$ *domeniu spaţial* al mulţimii şi cea de-a n-a coordonată, *suprafaţa mulţimii*. Vom nota aceste "părţi" ale mulţimii A cu $A_{(1,n-1)}$ şi respectiv $A_{(n,n)}$. Putem avea n moduri diferite de a alege aceste partiţii ale coordonatelor (corespunzând modului în care se poate alege o coordonată dintr-un total de n).

Vârful (top) sau s*uprafaţa de vârf* a unei mulţimi A este funcţia $T(A) : A_{(1,n-1)} \longrightarrow A_{(n,n)}$ definită de:

$$T(A)(\mathbf{z}) = \max\{y \mid (\mathbf{z}, y) \in A, \forall \mathbf{z} \in A_{(1,n-1)}\} \tag{6.45}$$

În acest mod se introduce o funcţie, pornind de la o mulţime; funcţia este cu valori întregi (funcţie scalară) de argument vectorial $n-1$-dimensional. Pentru exemplificare considerăm cazul în care $n = 2$, deci fiecare punct al mulţimii A este un vector bidimensional, $\mathbf{x} = (i, j)$. Considerăm coordonata i ca fiind domeniul spaţial şi coordonata j ca

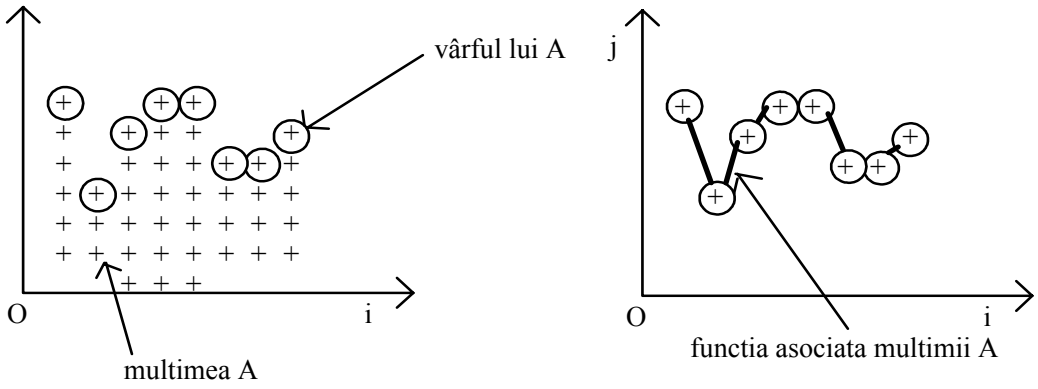


Fig. 6.11: Exemplificare pentru funcţia vârf a unei mulţimi.

fiind suprafaţa mulţimii. Atunci, prin transformarea vârf, asociem mulţimii A o funcţie definită pe domeniul său spaţial şi cu valori în suprafaţa sa.

Umbra unei funcţii oarecare f este definită ca transformarea $f : B \subseteq \mathbf{Z}^{n-1} \longrightarrow C \subseteq \mathbf{Z}$ dată de:

$$U(f) = \{(\mathbf{z}, y) \mid \mathbf{z} \in B, y \in \mathbf{Z},\ y \leqslant f(\mathbf{z})\} \tag{6.46}$$

În acest mod, plecând de la o funcţie de variabilă vectorială $n-1$-dimensională, cu valori scalare, se obţine o mulţime ale cărei elemente sunt vectori n-dimensionali. Mulţimea obţinută este semideschisă (nemarginită).

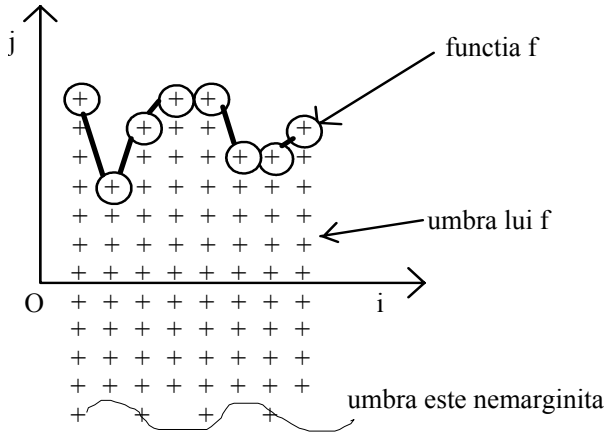


Fig. 6.12: Determinarea umbrei unei mulţimi.

În mod evident, vom avea următoarele relaţii între cele două transformări mulţime-funcţie (vârful şi umbra):

$$T(U(F)) = f \text{ şi } A \subseteq U(T(A)) \tag{6.47}$$

Operaţiile morfologice pentru funcţii se definesc prin intermediul transformării acestora în mulţimi (reducând astfel operaţiile pe funcţii la operaţii pe mulţimi, aşa cum au fost prezentate la morfologia binară):

$$f \ominus g = T\left(U(f) \ominus U(g)\right) \text{ şi } f \oplus g = T\left(U(f) \oplus U(g)\right) \tag{6.48}$$

Ca şi în cazul morfologiei pe mulţimi, g se numeşte element structurant (funcţie structurantă). Se numeşte element structurant *flat*, elementul structurant pentru care $g(\mathbf{y}) = 0, \forall \mathbf{y} \in Supp(g)$. Aceste elemente structurante de tip *flat* sunt echivalente cu cele folosite în morfologia clasică pe mulţimi.

Forma echivalentă a definiţiilor erodării şi dilatării (folosite în mod efectiv în practică) este, pentru $\forall \mathbf{x} \in Supp(f)$:

$$f \ominus g = \min_{\mathbf{y} \in Supp(g)} \left\{ f(\mathbf{x} - \mathbf{y}) - g(\mathbf{y}) \right\}, \; f \oplus g = \max_{\mathbf{y} \in Supp(g)} \left\{ f(\mathbf{x} - \mathbf{y}) + g(\mathbf{y}) \right\} \tag{6.49}$$

Dacă se utilizează elemente structurante plate (deci caracterizate doar de forma suportului, şi nu şi de valori asociate acestora), cele două relaţii de definiţie devin identice cu filtrările de ordine de rang minim şi respectiv maxim:

$$f \ominus g = \min_{\mathbf{y} \in Supp(g)} \left\{ f(\mathbf{x} - \mathbf{y}) \right\}, \; f \oplus g = \max_{\mathbf{y} \in Supp(g)} \left\{ f(\mathbf{x} - \mathbf{y}) \right\} \tag{6.50}$$

Capitolul 7

METODE DE COMPRESIE A IMAGINILOR

Termenul de compresie a imaginilor (uneori numit şi codare a imaginilor) se referă la o clasă largă de tehnici şi metode al căror scop este reprezentarea unui imagini date cu un număr cât mai mic de biţi (mai mic decât numărul de biţi al reprezentării iniţiale). Necesitatea reducerii cantităţii de informaţie necesară reprezentării este evidentă dacă considerăm cazul memorării imaginilor radiografice (4000 x 2500 pixeli, cu 4096 nivele de gri, deci 14,3 MB) sau al transmisiei de televizune alb-negru (625 x 625 pixeli cu 256 nivele de gri, de 50 de ori pe secundă, deci un flux de 18.6 MB/ secundă) [9]. Procesul de recompunere a imaginii iniţiale din reprezentarea restrânsă se numeşte decompresie sau decodare; este evident că prin decodare trebuie să se obţină o imagine cât mai apropiată de imaginea orginală. Există două categorii fundamentale de tehnici de compresie (codare): codarea fără pierderi (în care imaginea decodată este identică cu imaginea iniţială) şi codarea cu pierderi, în care se admit mici diferenţe faţă de original. Calitatea unui procedeu de compresie (pentru o imagine dată) se măsoară prin factorul de calitate (raportul semnal zgomot (3.4) dintre imaginea originală f şi imaginea decodată g) şi factorul (raportul) de compresie. Factorul de compresie C este raportul dintre cantitatea de informaţie necesară reprezentării imaginii iniţiale şi cantitatea de informaţie necesară reprezentării imaginii codate; evident compresia are loc dacă factorul de compresie este supraunitar ($C > 1$). Uneori, factorului de compresie i se asociază (sau este înlocuit de) rata de compresie: cantitatea de informaţie necesară reprezentării comprimate a fiecărui pixel al imaginii; rata de compresie se măsoară în biţi per pixel (bpp).

O altă clasificare posibilă a tehnicilor de compresie se poate face după tipul imaginii căreia i se aplică: vom face astfel distincţia între compresia imaginilor binare şi compresia imaginilor cu nivele de gri. Se impune totuşi o observaţie: metodele de codare ce se vor prezenta în cadrul tehnicilor speifice imaginilor binare pot fi folosite pentru compresia oricărei succesiuni de valori binare, indiferent de semnificaţia acestora (ceea ce înseamnă că ar putea fi folosite şi pentru compresia imaginilor cu nivele de gri) şi sunt metode de

compresie fără pierderi.

7.1 Compresia imaginilor binare

Putem considera că singura categorie de imagini binare de interes sunt imaginile în alb-negru (sau monocrome); valorile punctelor acestora sunt fie 0 (reprezentând fundalul de culoare albă), fie 1 (reprezentând punctele de interes, de culoare neagră)[1]. Cele două clase de metode de codare pe care le avem în vedere sunt codarea entropică (metoda de codare Huffman) şi metodele de codare *on-line* (pe flux de biţi); deosebirea dintre aceste metode (la un nivel al implementării) este că pentru codarea entropică este necesară parcurgerea şi stocarea intermediară a întregii imagini.

7.1.1 Codarea entropică (Huffman)

Codarea entropică (Huffman) este metoda optimală de codare a unei surse de informaţie. Codarea sursei de informaţie S ale cărei mesaje sunt $\{s_1, s_2, ..., s_N\}$, de probabilităţi $\{p(s_1), p(s_2), ..., p(s_N)\}$ prin alfabetul X cu D simboluri înseamnă a asocia fiecărui mesaj s_i al sursei primare de informaţie un şir de simboluri din alfabetul codului. Lungimea medie a cuvintelor de cod este dată de raportul dintre entropia sursei primare şi entropia alfabetului codului:

$$\bar{l} = \frac{H(S)}{H(X)} \tag{7.1}$$

Se doreşte obţinerea unei lungimi medii minime a cuvintelor de cod, şi deci, echivalent, mărirea entropiei alfabetului codului; la limită, lungimea media minimă posibilă de obţinut este:

$$\bar{l}_{\min} = \frac{H(S)}{\log D} \tag{7.2}$$

Procedeul practic prin care se realizează alocarea simbolurilor din alfabetul codului astfel încât entropia acestuia să fie maximizată (metoda Huffman) se bazează pe reducerea iterativă a numărului de simboluri ale sursei de codat şi construirea unei surse restrânse. La fiecare etapă cele D simboluri cele mai puţin probabile ale sursei de informaţie sunt reunite într-un nou simbol; procedeul de restrângere continuă până când se obţine o sursă redusă cu exact D simboluri. Apoi, pentru fiecare reunire de simboluri, fiecare mesaj individual va primi codul cuvântului reunit ca prefix, urmat de câte un simbol din alfabetul codului. Vom considera următorul exemplu.

O sursă S generează 6 simboluri, de probabilităţi descrise de vectorul $P = [0.3; 0.25; 0.2; 0.1; 0.1; 0.05]$. *Sursa este codată optimal, simbol cu simbol, cu cuvinte de cod generate cu*

[1] Deci convenţia de reprezentare prin culori este modificată faţă de convenţia generală utilizată – 0 nu mai este negru, ci alb.

simboluri dintr-un alfabet ternar. Să se calculeze cuvintele de cod, arborele de codare şi eficienţa codării.

Codarea optimală a unei surse se realizează conform algoritmului Huffman. Se ştie că numărul de simboluri ale sursei ce se codează trebuie să îndeplinească o anume relaţie $\left(\frac{N-D}{D-1} \in \mathbf{N}\right)$; în acest caz, $N = 6$, $D = 3$ şi deci

$$\frac{N-D}{D-1} = \frac{6-3}{3-1} = \frac{3}{2} \notin \mathbf{N}$$

Completarea se face adăugând sursei simboluri de probabilitate nulă; în acest caz, cu un singur astfel de simbol adăugat obţinem $N = 7$ şi

$$\frac{N-D}{D-1} = \frac{7-3}{3-1} = \frac{4}{2} \in \mathbf{N}$$

Entropia sursei extinse S este dată de:

$$H(S) = -\sum_{i=1}^{7} p(s_i) \log p(s_i) = -\left(0.3 \log 0.3 + 0.25 \log 0.25 + 0.2 \log 0.2 + 2 \cdot 0.1 \log 0.1\right) -$$

$$-0.05 \log 0.05 - 0 \log 0 = 2.366 \text{ bit/simbol}$$

Atunci, conform (7.2), lungimea medie minimă a unui cuvânt de cod este:

$$\bar{l}_{\min} = \frac{H(S)}{\log D} = \frac{2.366}{\log 3} = 1.493 \text{ bit}$$

Codarea Huffman este realizată conform tabelului 7.1.

Sursă primară	$p(s_i)$	Cod	Sursă restrânsă	$p(r_{ji})$	Cod	Sursă restrânsă	$p(r_{ji})$	Cod
						r_{21}	0.45	0
s_1	0.3		r_{11}	0.3		r_{22}	0.3	1
s_2	0.25		r_{12}	0.25		r_{23}	0.25	2
s_3	0.2		r_{13}	0.2	00			
			r_{14}	0.15	01			
s_4	0.1		r_{15}	0.1	02			
s_5	0.1	010						
s_6	0.05	011						
s_7	0	012						

Tabel 7.1: Codare Huffman

Cuvintele de cod sunt grupate în tabelul 7.2; pe baza lor putem calcula lungimea medie a cuvintelor de cod:

$$\overline{l} = \sum_{i=1}^{7} l_i p(s_i) = 1 \cdot 0.3 + 1 \cdot 0.25 + 2 \cdot 0.2 + 2 \cdot 0.1 + 3 \cdot 0.1 + 3 \cdot 0.05 + 3 \cdot 0 = 1.6$$

Sursă primară	$p(s_i)$	Cod	Lungime
s_1	0.3	1	1
s_2	0.25	2	1
s_3	0.2	00	2
s_4	0.1	02	2
s_5	0.1	010	3
s_6	0.05	011	3
s_7	0	012	3

Tabel 7.2: Coduri asociate simbolurilor sursei şi lungimile lor

În cazul imaginilor (sau al şirurilor binare) mesajele sursei primare sunt formate din grupuri de câte B biţi succesivi (astfel, sursa primară are 2^B mesaje, ale căror probabilităţi de apariţie sunt specifice imaginii considerate). Cu cât B este mai mare, cu atât mai mare va fi eficienţa codării. Tabelul de codare (echivalenţa între mesajele sursei primare şi şirurile de simboluri de cod) este specific fiecărei imagini şi trebuie să însoţească codul. Codarea clasică este binară ($D = 2$, $X = \{0, 1\}$).

7.1.2 Codarea pe flux de biţi

Metodele de codare pe flux de biţi (*on-line*) se regăsesc la primul nivel de codare al transmisiilor de fax (RLE, WBS) sau ca tehnici generale de compresie a datelor, înglobate atât în formate de fişiere grafice (TIFF) cât şi în arhivatoare uzuale (ZIP) – metoda Ziv-Lempel. Aplicarea codării la imagini presupune în primul rând transformarea imaginii (structurii de matrice a acesteia) într-un vector (şir) prin concatenarea liniilor acesteia.

Codarea RLE

Principiul codării RLE (*Run Length Encoding*) este acela de a memora numărul de simboluri succesive ale şirului binar ce au aceeaşi valoare. Odată ce o secvenţă de simboluri succesive de aceeaşi valoare (*run length*) se încheie, simbolul următor nu poate avea decât

valoarea complementară celei iniţiale, prin alternanţă. Şirul codat trebuie să înceapă cu valoarea primului simbol.

Conform acestei codări, şirul de valori binare $0,1,1,1,1,0,0,0,1,0,1,1,1,0,1,1,1$ este codat RLE ca $0,4,3,1,1,3,1,3$, iar şirul binar $1,1,1,1,1,1,1,0,1,0,0,0,0,0,0$ este codat RLE ca $1,7,1,1,5$. Şirul cuvintelor de cod trebuie deci să fie compus din codurile ce corespund lungimilor secvenţelor de valori identice succesive; aceasta înseamnă că numărul de biţi al reprezentării binare al cuvintelor de cod trebuie să permită reprezentarea lungimii celei mai mari secvenţe. Cum determinarea de fiecare dată a acestei lungimi maximale şi varierea lungimii cuvântului de cod nu este o soluţie corespunzătoare, se preferă fixarea unei lungimi maximale admise a secvenţelor. Orice secvenţă mai lungă decât secvenţa maximă este despărţită prin intercalarea fictivă a unor simboluri complementare. Să considerăm o codare RLE cu lungimea cuvântului de cod de 2 biţi; aceasta înseamnă că lungimea secvenţei maxime este 3 (codul 0 trebui rezervat pentru secvenţa de lungime nulă). Dacă şirul binar este $1,1,1,1,1,1,1,0,1,0,0,0,0,0$ atunci şirul codat va fi 1, 3, 0, 3, 0, 1, 1, 1, 3, 0, 2 (unde fiecare cifră va fi reprezentată pe 2 biţi).

O codare mai bună se obţine dacă lungimile secvenţelor de simboluri identice succesive sunt codate (într-o fază ulterioară) entropic, cu un algoritm de tip Huffman. Standardul CCITT de transmisie fax recomandă folosirea unor codări Huffman truncheate: astfel, dacă o lungime de secvenţă este mai mică decât 64 este codată direct, cuvântul de cod respectiv numindu-se terminator; dacă o lungime de secvenţă este cuprinsă în gama [64;1791] se codează separat numărul de pixeli ce formează un multiplu de 64 (formând un cod mască, *make-up code*) şi restul împărţirii lungimii secvenţei la 64, cu un cod terminator. În plus, există un cod special de sfârşit de linie (EOL). Tabelele de codare sunt standardizate (se pot găsi de exemplu în [9, pp. 544-545]). Pe lângă transmisia de fax, codarea RLE mai este utilizată în diferite formate de fişiere imagine (BMP, PCX, TGA).

Porţiunea următoare de program C realizează codarea RLE a şirul de intrare *in*, de dimensiune *DIM_IN*, cu cuvinte de cod ce pot reprezenta cel mult *MAX_RUN* simboluri identice succesive; cuvintele de cod sunt scrise în şirul *out*. Poziţiile curente din şirurile de intrare şi ieşire sunt determinate de variabilele *pos_in* şi respectiv *pos_out*. Implementarea presupune că tipul de date al şirului *in* permite aplicarea operatorului de negaţie (*!*).

```
pos_in=0;
out[0]=in[0];
pos_out=1;
crt_value=in[0];
while (pos_in<DIM_IN) {
  if (in[pos_in]==crt_value) then
    if (run_length<MAX_RUN) then
      run_length++;
```

```
    else {
      out[pos_out]=MAX_RUN;
      pos_out++;
      out[pos_out]=0;
      run_length=1; }
  else {
    out[pos_out]=run_length;
    pos_out++;
    run_length=1;
    crt_value=!crt_value; }
  pos_in++; }
```

Codarea WBS

Prima etapă a codării WBS (*White Block Skipping*) presupune împărţirea şirului binar în grupe de câte D simboluri succesive. Principiul codării este simplu: un bloc nul este înlocuit cu un singur simbol de 0, un bloc nenul este prefixat de un simbol de 1 şi copiat. Conform acestei codări (cu grupe de $D = 3$ simboluri), şirul de valori binare $0,0,0,0,1,0,0,0,1,1,1,0,0,0,0$ este împărţit în grupele $(0,0,0)$, $(0,1,0)$, $(0,0,1)$, $(1,1,0)$, $(0,0,0)$ şi codat ca (0), $(1,0,1,0)$, $(1,0,0,1)$, $(1,1,1,0)$, (0), transformându-se în şirul $0,1,0,1,0,1,0,0,1,1,1,1,0,0$.

Dacă şirul dat binar de n grupe de simboluri conţine n_0 grupe nule, în urma compresiei mai rămân $(n-n_0)(D+1)+n_0$ simboluri şi factorul de compresie este atunci:

$$C = \frac{nD}{(n-n_0)(D+1)+n_0} \tag{7.3}$$

Pentru ca să existe compresie trebuie ca $C > 1$ şi deci atunci

$$n_0 D > n \Longleftrightarrow \frac{n_0}{n} > \frac{1}{D} \tag{7.4}$$

adică proporţia de grupe nule din şir trebuie să fie mai mare decât inversul numărului de simboluri dintr-o grupă.

Pentru o largă categorie de imagini, lungimea blocului de codare $D = 10$ poate fi folosită cu rezultate bune [9]. O îmbunătăţire a performanţei se poate obţine dacă se introduce o adaptare a codării, prin detecţia liniilor albe (reprezentate doar cu valori 0) şi codarea unei linii întregi cu un singur 0 (desigur că în acest caz la codurile tuturor blocurilor se mai adaugă un prefix de valoare 1). O altă variantă a metodei exploatează structura bidimensională a imaginii, codând blocuri pătrate din imagine (deci imaginea nu mai este vectorizată).

Porţiunea următoare de program C realizează o variantă de codare WBS cu blocuri de dimensiune D a şirului de intrare *in* (ale cărui valori sunt 0 şi 1) de dimensiune DIM_IN;

codul WBS este scris în şirul de ieşire *out*. Verificarea dacă blocul curent este nul se face prin sumarea valorilor acestuia. Poziţiile curente în şirul de intrare şi şirul de ieşire sunt memorate în *pos_ in* şi respectiv *pos_ out*.

```
pos_in=0;
pos_out=0;
while (pos_in<DIM_IN){
s=0;
for (i=0;i<D;i++)
  s=s+in[pos_in+i];
if (s>0) then {
  out[pos_out]=1;
  pos_out++;
  for (i=0;i<D;i++)
    out[pos_out+i]=in[pos_out+i];
  pos_out=pos_out+D; }
else {
  out[pos_out]=0;
  pos_out++; }
pos_in=pos_in+D; }
```

Codarea Ziv-Lempel

Codarea Ziv-Lempel nu are ca origine ideea explicită de a mări entropia alfabetului codului (precum codarea Huffman), ci se bazează pe tehnica dicţionar LUT (folosită şi la reprezentarea imaginilor indexate): cuvintele de cod asociate unor şiruri de simboluri ale sursei de intrare sunt indicii (numerele de ordine) poziţiilor la care se găsesc respectivele şiruri de simboluri în dicţionarul codului (tabela de codare). Tabela de codare (dicţionarul) este creată pe măsura parcurgerii şirului de simboluri de intrare, şi, pentru decodare, tabela se crează din cuvintele de cod deja transmise (ceea ce înseamnă că nu este necesară memorarea sau transmiterea explicită a tabelului de codare[2]).

Dicţionarul iniţial are două cuvinte: cuvântul 0, având indicele 0 şi cuvântul 1, având indicele 1. Din şirul de intrare (presupus binar) se extrage câte un bit şi se verifică dacă şirul de biţi deja extras se regăseşte sau nu în dicţionar. Dacă şirul se regăseşte în dicţionar, se mai adaugă un bit din şirul de intrare. Dacă şirul nu se regăseşte în dicţionar, atunci şirul este trecut în dicţionar, în şirul codat se scrie codul (indicele) prefixului şirului nou adăugat (deci poziţia la care se găseşte în dicţionar cuvântul la care

[2]Această caracteristică este probabil unică codului Ziv-Lempel; pentru toate celelalte aplicări ale tehnicii LUT, tabela de codare este transmisă sau memorată împreună cu cuvintele de cod sau se presupune existenţa unei tabele de codare implicite, cunoscute (ca de exemplu tabela de culoare cu nivele de gri).

adăugând ultimul bit al şirului de intrare se obţine un cuvânt nou) şi ultimul bit din şir devine primul bit al următorului şir de intrare. Procedeul continuă până când şirul de intrare nu mai are simboluri necodate. Dacă, înaintea terminării simbolurilor din şirul de codat, tabela de codare (cu număr fixat de intrări) se umple, există două variante de continuare: fie restul şirului de intrare se codează conform tabelei de codare existente prin căutarea celor mai lungi cuvinte de cod, fie tabela de codare este "golită" şi se continuă cu procedeul iniţial.

Să considerăm şirul de intrare 1, 0, 0, 0, 0, 1, 0, 1, 1, 1; dicţionarul cuvintelor de cod conţine cuvântul 0 (cu indice 0) şi cuvântul 1 cu indice 1. Simbolul curent este 1 (pe prima poziţie a şirului de intrare) şi formează şirul de biţi extras; acest şir se regăseşte în dicţionar (cu indicele 1) şi atunci se mai extrage un bit din şir; şirul de biţi extraşi devine 10. Cuvântul 10 nu există în dicţionar, şi atunci: în şirul de ieşire se scrie indicele celui mai lung cuvânt din dicţionar la care adăugând un bit obţinem noul cuvânt (noul cuvânt este 10, cuvântul prefix este 1, iar indicele scris la ieşire este 1), în dicţionar se adaugă cuvântul 10 (care va avea indicele 2), iar noua poziţie curentă din şirul de intrare este ultimul bit al şirului deja extras, deci poziţia 2, bitul 0. Simbolul curent este 0; acest şir se regăseşte în dicţionar (cu indicele 0) şi atunci se mai extrage un bit din şir; şirul de biţi extraşi devine 00. Cuvântul 00 nu există în dicţionar, şi atunci: în şirul de ieşire se scrie indicele prefixului noului cuvânt (deci 0), în dicţionar se adaugă cuvântul 00 (care va avea indicele 2), iar noua poziţie curentă din şirul de intrare este ultimul bit al şirului deja extras, deci poziţia 3, bitul 0. Simbolul curent este 0; acest şir se regăseşte în dicţionar (cu indicele 0) şi atunci se mai extrage un bit din şir; şirul de biţi extraşi devine 00. Şirul 00 se regăseşte în dicţionar (indicele 3) şi atunci se mai extrage un bit din şir; şirul de biţi extraşi devine 000. Cuvântul 000 nu există în dicţionar, şi atunci: în şirul de ieşire se scrie indicele prefixului noului cuvânt (deci 3), în dicţionar se adaugă cuvântul 000 (care va avea indicele 4), iar noua poziţie curentă din şirul de intrare este ultimul bit al şirului deja extras, deci poziţia 5, bitul 0. Procedeul continuă în mod analog.

Pentru decodare, fiecare nou cuvânt de cod implică scrierea unei noi intrări în tabelul de codare (dicţionar) în care şirul de simboluri este format din prefix (şirul de simboluri care se găseşte în dicţionar la intrarea precizată de cuvântul de cod) şi o terminaţie de 1 bit, a cărei valoare rezultă la primirea cuvântului de cod următor (să nu uităm că şirurile succesive de simboluri ce se codează au în comun ultimul, respectiv primul bit).

Metoda Ziv-Lempel a pus bazele unei clase mai largi de metode de compresie, incluzând printre altele şi varianta LZW (Lempel-Ziv-Walsh) folosită de utilitarul de compresie ZIP.

7.2 Compresia imaginilor cu nivele de gri

Cea mai imediată metodă de codare a unei imagini cu nivele de gri este de a o considera ca un şir de biţi şi de aplica metodele de codare pentru imagini binare: fie pentru

fiecare plan de bit al reprezentării binare a nivelor de gri, fie pentru succesiunea de biţi a reprezentărilor nivelelor de gri. Asemenea abordări produc codări fără pierderi a imaginilor şi nu produc întotdeauna rezultate spectaculoase. O mult mai mare amploare a căpătat clasa de metode de compresie cu pierderi [controlabile].

7.2.1 Codarea predictivă

Codarea predictivă se bazează pe existenţa unei importante corelaţii spaţiale între valorile pixelilor unei imagini (deci valorile pixelilor vecini sunt asemănătoare). Această corelaţie poate implica, de exemplu, că valoarea unui pixel poate fi aproximată cu o combinaţie a valorilor unora dintre vecinii săi. Dacă se stabileşte o ordine de parcurgere a pixelilor ce formează imaginea (deci dacă se stabileşte o ordine de baleiere a imaginii) şi pentru aproximarea valorii pixelului curent se folosesc pixeli vecini spaţial lui, parcurşi anterior, spunem că prezicem valoarea pixelului curent. Predicţia se realizează printr-o funcţie, care, cunoscută la nivelul întregii imagini permite determinarea unei variante aproximative a imaginii date cunoscând doar un număr mic de "pixeli de start". Pentru a avea o codare cât mai precisă, se folosesc şi erorile dintre valorile prezise şi cele reale; codarea asociată imaginii iniţiale va conţine deci funcţia de predicţie, valorile de start şi erorile de aproximare ale fiecărui punct. Dacă predictorul este determinat în mod corect, atunci eroarea de predicţie $e(n)$ este mică şi reprezentarea ei necesită mult mai puţini biţi decât reprezentarea valorii originale $u(n)$. Schema de codare cu predicţie este reprezentată în figura 7.1.

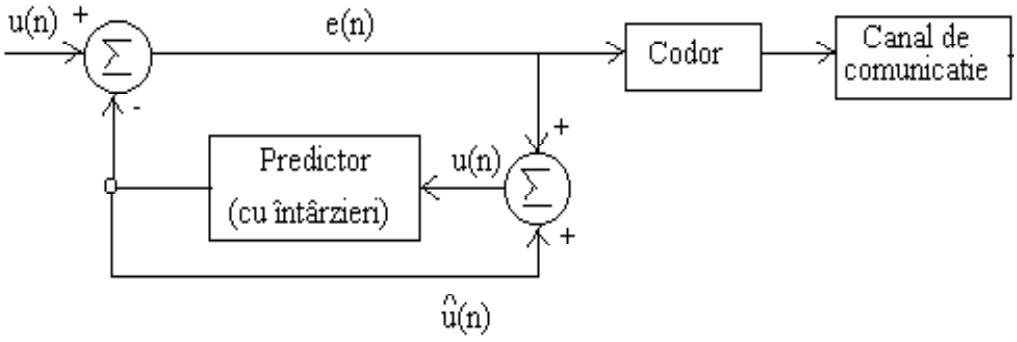


Fig. 7.1: Schema de codare cu predicţie.

Ecuaţiile ce descriu procesul sunt ecuaţia erorii (7.5) (care exprimă eroarea de aproximare $e(n)$ ca diferenţă între valoarea corectă $u(n)$ şi valoarea prezisă $\widehat{u}(n)$) şi ecuaţia de predicţie (7.6) (ce exprimă modul în care se determină valoarea aproximativă $\widehat{u}(n)$ din valorile anterioare $u(n-k_1), u(n-k_2), ...$ pe baza predictorului *pred*):

$$e(n) = u(n) - \widehat{u}(n) \tag{7.5}$$

$$\widehat{u}(n) = pred\,(u(n-k_1), u(n-k_2), ...) \tag{7.6}$$

Procesul de decodare este reprezentat schematic în figura 7.2. Eroarea de predicţie $e_q(n)$ (cuantizată) este adunată la valoarea aproximativă $\widehat{u'}(n)$, determinată cu acelaşi predictor *pred* din valorile $u'(n)$ deja calculate.

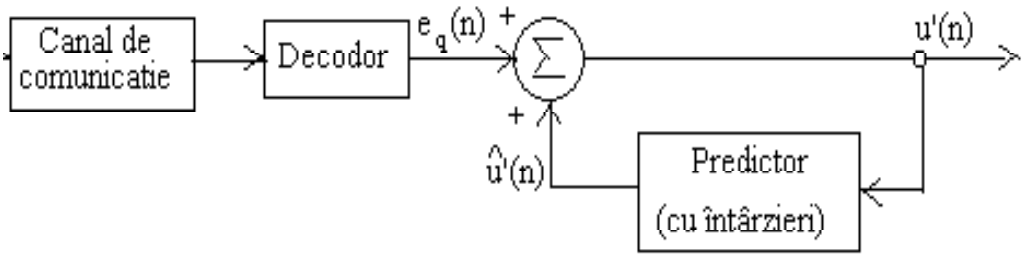


Fig. 7.2: Schema de decodare cu predicţie.

Predictorii cei mai simpli sunt liniari şi sunt invariaţi la schimbarea punctului curent. Predictorii pe linii sau coloană calculează aproximarea în punctul curent $u(m,n)$ al imaginii ca valoarea anterioară de pe aceeaşi linie $\widehat{u}(m,n) = u(m,n-1)$ sau de pe aceeaşi coloană $\widehat{u}(m,n) = u(m-1,n)$ (dacă ordinea de baleiaj este cea uzuală). Se mai pot folosi predictori de tip valoare medie:

$$\widehat{u}(m,n) = \frac{1}{2}\left(u(m-1,n) + u(m,n-1)\right) \tag{7.7}$$

$$\widehat{u}(m,n) = \frac{1}{4}\left(u(m-1,n) + u(m,n-1) + u(m-1,n-1) + u(m-1,n+1)\right) \tag{7.8}$$

Un caz particular de codare cu predicţie este modulaţia delta, caracterizată de cuantizarea erorii de predicţie (aproximare) $e(n)$ cu un singur bit (bit de semn).

7.2.2 Compresia imaginilor cu transformate

Principiul compresiei cu transformate a imaginilor se bazează pe proprietăţile de compactare a energiei şi decorelare a componentelor spectrale pe care le prezintă majoritatea transformărilor integrale unitare (discutate în secţiunea 4.2). Atâta vreme cât cea mai mare parte a energiei este distribuită în câteva componente spectrale (de joasă frecvenţă), toate celelalte pot fi ignorate; astfel memoria necesară reprezentării este mult mai mică. Este evident că o asemenea metodă de compresie este cu pierderi.

Aplicarea practică a compresiei cu transformate trebuie să aibă în vedere trei aspecte: alegerea transformatei, stabilirea frecvenţei limită de la care începe ignorarea valorilor şi, în fine, cuantizarea componentelor spectrale păstrate.

Transformarea optimală trebuie să decoreleze complet componentele spectrale (asigurând astfel şi compactarea maximă a energiei şi cea mai bună eroare de aproximare prin truncherea frecvenţelor înalte). Decorelarea completă este dependentă de proprietăţile statistice ale imaginii (matrice de covariaţie), deci, teoretic, pentru fiecare imagine în parte, trebuie găsită transformarea optimală. Această transformare este transformata Karhunen-Loeve: transformare integrală unitară a cărei matrice de transformare are pe coloane vectorii proprii normaţi ai matricii de covariaţie a imaginii. Cum această transformare este evident unică pentru o clasă de imagini, în practică se încearcă găsirea unei aproximaţii. În condiţiile în care majoritatea imaginilor naturale pot fi aproximate printr-un model Markov puternic corelat (exprimând dependenţa puternică a valorii pixelilor de valorile vecinilor lor imediaţi), transformata cosinus s-a dovedit o foarte bună alegere.

Cuantizarea componentelor spectrale poate integra şi selecţia componentelor cel mai importante: componentele de frecvenţă joasă sunt cuantizate o precizie mai mare, iar componentele de frecvenţă înaltă sunt cuantizate grosier (echivalent chiar cu eliminarea acestora). Numărul de nivele de cuantizare şi distribuţia acestora (diferenţele dintre nivelele vecine) este adaptate statisticii semnalului (cuantizarea optimă este cuantizarea Loyd-Max [9], [16]).

Exemplul cel mai des folosit de compresie cu transformate este standardul de compresie JPEG (fişiere imagine cu extensia .jpg). Imaginea este divizată în blocuri de 8 x 8 pixeli, care nu se suprapun. Fiecărui bloc i se aplică o transformată cosinus bidimensională, iar cei 64 de coeficienţi ai transformării sunt copiaţi într-un vector prin baleierea pe diagonală a blocului de 8 x 8 pixeli. Coeficienţii sunt cuantizaţi în conformitate cu un număr prestabilit de nivele de cuantizare (stabilit prin standard, şi proporţional cu factorul de calitate dorit pentru imaginea refăcută). Coeficienţii corespunzând frecvenţelor nule (valorile medii ale blocurilor) sunt codate predictiv printr-o tehnică de tip DPCM (*Differential Pulse Code Modulation*). Valorile celorlalţi coeficienţi sunt codaţi entropic (Huffman). Factorii de compresie ce rezultă sunt cuprinşi în mod tipic între 10 şi 100. Aspectul de compresie cu pierderi (diferenţele faţă de imaginea originală) se manifestă prin efectul de *blocking*: sublinierea frontierelor de separaţie a blocurilor de bază (efect observabil şi în figura 7.3).

7.2.3 Codarea cu arbori cuaternari

Un arbore cuaternar (numit în engleza *quadtree*) este un arbore în care fiecare nod neterminal are exact patru descendenţi.

Orice imagine pătrată, de dimensiune putere a lui 2 ($N = 2^K$) poate fi reprezentată

Fig. 7.3: Imagine decodată în urma unei compresii JPEG cu raport de compresie 23 (factor de calitate 90)

(într-o reprezentare de tip ierarhic) pe o structură de arbore cuaternar. Nodurile de pe fiecare nivel al arborelui corespund unei împărţiri a unei zone pătrate din imagine în patru "sferturi". Rădăcina arborelui este asociată întregii imagini (imaginii iniţiale), nodurile de pe primul nivel al arborelui corespund celor patru sferturi ale imaginii, nodurile de pe nivelul doi corespund sferturilor fiecărui sfert anterior determinat al imaginii, şi aşa mai departe. Împărţirea imaginii poate continua până când nodurile nivelului curent al arborelui corespund unor zone pătrate a căror dimensiune este de un pixel. Adâncimea arborelui astfel obţinut este K, şi fiecare pixel al imagini va corespunde unui nod terminal (frunză) de pe ultimul nivel al arborelui. Fiecare nod terminal conţine informaţia de valoare a pixelului la care este asociat.

Structura arborelui anterior poate fi simplificată prin introducerea în etapa de construcţie a unui test de uniformitate a regiunilor reprezentate de fiecare nod: dacă regiunea pătrată considerată nu este uniformă, atunci aceasta va fi descompusă prin tăiere în patru părţi egale şi nodul corespunzător va deveni neterminal (va "căpăta" cei patru descendenţi). Dacă regiunea pătrată considerată este uniformă (deci este compusă din pixeli de acelaşi fel), nodul respectiv devine un nod frunză (terminal) al arborelui (deci nu mai are descendenţi). Fiecare nod terminal conţine informaţia de valoare a zonei de imagine la care este asociat (vezi figura 7.4). O zonă este considerată uniformă dacă diferenţa maximă de nivel de gri a pixelilor ce o formează nu depăşeşte un anumit prag impus; valoarea zonei uniforme este media nivelelor de gri a pixelilor ce o compun.

Pentru refacerea imaginii inţiale din reprezentarea arborescentă este suficientă alegerea nodurilor terminale a căror valoare corespunde pixelilor de obiect. Adâncimea la care este plasată în arbore o frunză conţine informaţia de dimensiune a zonei pătrate corespunzătoare din imagine (o frunză situată la adâncimea h corespunde unei zone pătrate de latură 2^{K-h} pixeli). Poziţia frunzei faţă de nodurile de pe acelaşi nivel ce au acelaşi

predecesor este direct determinată de regula de alocare a sferturilor unei zone la nodurile descendente ale arborelui (regula de alocare trebuie să se păstreze pentru întregul arbore) (vezi figura 7.5).

Codarea imaginii (sau a arborelui cuaternar asociat) se face prin memorarea poziţiei în arbore a nodurilor terminale şi a valorilor acestora. Poziţia în arbore a unui nod se specifică prin descrierea căii prin care se ajunge la acesta, pornind de la rădăcina arborelui; această cale va conţine codurile de alocare a descendenţilor ce corespund avansului în adâncime în arbore.

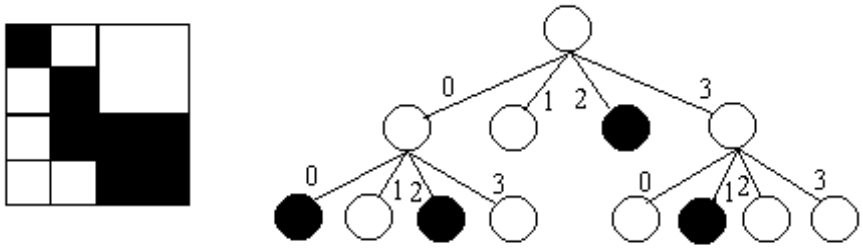


Fig. 7.4: Exemplu de reprezentare a unei imagini binare pe un arbore cuaternar complet

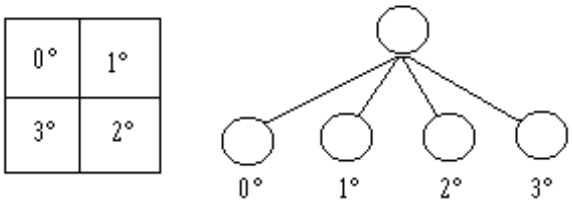


Fig. 7.5: Regula de alocare a descendenţilor

Principalul inconvenient al metodei de etichetare folosind arborele cuaternar este legat de complexitatea construirii acestuia prin abordarea ierarhică *top-down* (de sus în jos) prezentată; în particular, testul de uniformitate la nivelul fiecărui bloc presupune testarea valorilor tuturor pixelilor care compun blocul. Pe ansamblu, aceasta duce la parcurgerea fiecărui pixel din imagine de un număr de ori egal cu adâncimea în arborele cuaternar al blocului pătrat din care face parte.

Pentru a înlătura acest incovenient este suficient ca pixelii imaginii să nu mai fie parcurşi în ordinea tradiţională de baleiaj (pe linii, de la stânga la dreapta şi de sus în jos), ci

într-o altă ordine, care să îi prearanjaze pe grupuri ce corespund pătratelor de diviziune a imaginii. Un asemenea baleiaj este reprezentat de o curbă de umplere a spaţiului: un parcurs ce trece o singură dată prin fiecare pixel al imaginii, nu se autointersectează şi în care oricare doi pixeli parcurşi consecutiv sunt vecini spaţial în imagine (într-o vecinătate de tip V_4 sau V_8). Curbele de umplere a spaţiului sunt structuri fractale, definite prin repetarea la diferite nivele ierarhice a unei aceleiaşi structuri. Pentru baleiajul imaginilor s-au reţinut două astfel de curbe: curba Peano-Hilbert (numită şi curba Peano în U, după forma celulei sale de bază) (vezi figura 7.6) şi curba Morton (sau curba Peano în Z) (vezi figura 7.7).

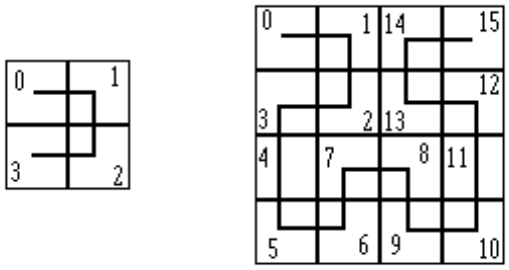


Fig. 7.6: Ordinea de parcurgere a pixelilor pentru curba Peano în U, la două nivele de rezoluţie

Dispunând de o astfel de ordine de baleiere, este evident că dacă se parcurge imaginea în acestă ordine, zonele pătrate uniforme (cu pixelii de aceeaşi valoare) sunt detectate într-o singură trecere şi astfel arborele cuaternar poate fi creat direct prin nodurile sale terminale. Pentru o implementare eficientă este însă necesară şi deducerea rapidă a indicelui pe curba de baleiere a pixelilor, pornind de la coordonatele lor în imagine. Doar curba Peano în Z are o asemenea relaţie rapidă de calcul, prin întreţeserea biţilor ce dau coordonatele în imagine a punctului. Cuvântul binar ce exprimă indicele pe curbă a oricărui punct este format din biţii din coordonata verticală, ce vor ocupa poziţiile de ordin par şi din biţii din coordonata orizontală ce vor ocupa poziţiile de ordin impar (păstrându-şi aceeaşi ordine de rang).

7.2.4 Cuantizarea vectorială

Cuantizarea vectorială este un algoritm de compresie a imaginilor ce se aplică asupra unor date vectoriale şi nu scalare, putând fi interpretat ca o extensie a conceptului de cuantizare scalară. Cuantizarea scalară asociază unei mulţimi mari de valori numere dintr-o mulţime mai mică (în mod tipic acestea din urmă fiind numere naturale); asocierea include (chiar dacă nu explicit) operaţii de tipul rotunjirii la cel mai apropiat întreg. Cuantizarea vectorială aproximează (sau rotunjeşte) un grup de numere deodată, nu doar unul singur. Aşadar, pentru a realiza o cuantizare vectorială sunt necesare un set

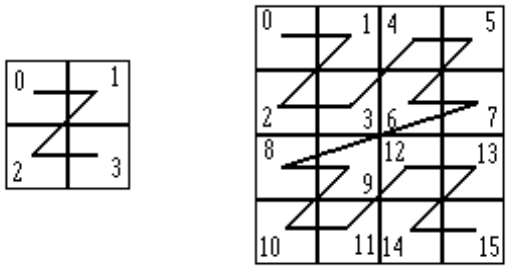


Fig. 7.7: Ordinea de parcurgere a pixelilor pentru curba Peano în Z, la două nivele de rezoluţie

de vectori de aproximare (inclusiv metoda prin care acestea pot fi deduse) şi o regulă de asociere a vectorilor de intrare cu vectorii de aproximare.

Să notăm cu $\mathbf{x}_i$ al i-lea vector de intrare şi cu $\boldsymbol{\mu}_j$ al j-lea vector de aproximare. Operaţia de cuantizare presupune înlocuirea vectorilor de intrare $\mathbf{x}_i$ cu vectori de aproximare $\boldsymbol{\mu}_j$, introducând deci erori; pentru ca erorile (măsurate de eroarea pătratică medie) să fie cât mai mici, este necesar ca pentru fiecare vector de intrare, aproximarea să se facă cu vectorul de aproximare cel mai apropiat (în sensul distanţei euclidiene). Aceasta este regula de asociere. Dacă există n vectori de aproximare ce pot fi folosiţi, aceştia se pot grupa într-un tabel de codare (existent şi la codare şi la decodare), iar fiecare vector de aproximare $\boldsymbol{\mu}_j$ va fi reprezentat doar prin indicele j (deci o altă aplicare a tehnicii LUT). Dacă vectorii de intrare au p componente, codate fiecare cu câte b biţi, iar numărul de vectori de aproximare poate fi reprezentat pe n_b biţi, atunci factorul de compresie realizat de cuantizarea vectorială este dat de[3]:

$$C = \frac{pb}{n_b} \tag{7.9}$$

Pentru cazul imaginilor, vectorii de intrare se aleg ca blocuri pătrate, nesuprapuse între ele, din imagine. Dimensiuni uzuale ale acestor blocuri sunt 4 x 4 şi 8 x 8 (rezultând deci vectori de intrare cu 16, respectiv 64 de componente). Dacă considerăm cazul imaginilor cu nivele de gri uzuale, reprezentate cu 256 nivele de gri ($b = 8$) şi dimensiuni ale tabelei de codare $n = 256$ (256 vectori de aproximare), atunci raportul de compresie este de 16, respectiv 64.

Construirea tabelei de codare (determinarea vectorilor de aproximare) se realizează în mod clasic prin algoritmi de clustering iterativ. În original (adică în limba engleză) termenul de "cluster" defineşte un grup, ciorchine, snop sau o clasă de unităţi, "asemănătoare". Asemănarea unităţilor este determinată în mod uzual prin asociere, similaritate sau distanţă între unităţi (vectori). Algoritmul de clustering este procesul prin care unei

[3]Acest mod de calcul al raportului de compresie nu ţine seama de necesitatea transmiterii sau memorării şi a tabelului de codare, de dimensiune npb biţi.

mulţimi de unităţi (entităţi) i se asociază, element cu element, o informaţie de apartenenţă la un anumit grup. Mai general, putem interpreta procesul de clustering ca un proces de partiţionare a unui set de unităţi într-un număr de submulţimi (numite clase sau clustere), pe baza unui anumit criteriu. Indiferent de criteriul folosit, se doreşte obţinerea unor clustere distincte, omogene şi bine separate. Numărul de clustere în care se face împărţirea setului de unităţi nu este în mod obligatoriu cunoscut apriori.

Metodele de clustering iterativ distribuie obiectele (vectorii) într-un număr predefinit de clase, repetând testarea unor condiţii pentru fiecare obiect al mulţimii; în funcţie de îndeplinirea sau nu a respectivelor condiţii, partiţia existentă la un moment dat este declarată corespunzătoare sau, în urma modificării alocării unor unităţi, procedeul de verificare se reia. Se poate considera că algoritmii iterativi fac mai multe "treceri" prin setul de obiecte de partiţionat, până la obţinerea stabilizării (convergenţei) valorii criteriului ce caracterizează calitatea partiţiei.

Calitatea partiţiei (a clasificării) este măsurată de suma varianţelor clusterelor (adică suma distanţelor de la fiecare vector la centrul clasei în care aparţine, ceea ce poate fi interpretat şi ca o eroare de aproximare a vectorilor din clase prin centrul respectivei clase). Deci funcţia criteriu ce trebuie minimizată este

$$J = \sum_{j=1}^{n} J_j = \sum_{j=1}^{n} \sum_{\mathbf{x}_i \in \omega_j} \left\| \mathbf{x}_i - \boldsymbol{\mu}_j \right\|^2 = \sum_{j=1}^{n} \sum_{i=1}^{N} u_{ij} \left\| \mathbf{x}_i - \boldsymbol{\mu}_j \right\|^2 \tag{7.10}$$

În urma minimizării lui J trebuie determinate valorile $\boldsymbol{\mu}_j$ (centrele claselor) şi valorile binare u_{ij}, coeficienţii de apartenenţa ai vectorilor i la clasele j, definiţi de:

$$u_{ij} = \begin{cases} 1, \text{ dacă } \mathbf{x}_i \in \omega_j \\ 0, \text{ dacă } \mathbf{x}_i \notin \omega_j \end{cases} \tag{7.11}$$

Determinarea centrelor claselor se poate face simplu, prin anularea derivatei în raport cu $\boldsymbol{\mu}_j$ a funcţiei criteriu J

$$\frac{\partial J}{\partial \boldsymbol{\mu}_j} = 2 \sum_{i=1}^{N} u_{ij} \left(\boldsymbol{\mu}_j - \mathbf{x}_i \right) = 0 \tag{7.12}$$

de unde rezultă că centrele claselor sunt mediile vectorilor ce aparţin acestora:

$$\boldsymbol{\mu}_j = \frac{\sum\limits_{i=1}^{N} u_{ij} \mathbf{x}_i}{\sum\limits_{i=1}^{N} u_{ij}} \tag{7.13}$$

Determinarea coeficienţilor de apartenenţă u_{ij} este însă o problemă de optimizare combinatorială, care nu poate fi rezolvată analitic. Pentru rezolvarea acestei probleme sunt

necesare metode iterative. Metoda imediată urmăreşte să determine, pentru fiecare vector al setului, dacă acesta poate fi mutat dintr-o clasă în alta, astfel ca suma varianţelor claselor să scadă. După fiecare asemenea mutare, este necesară actualizarea mediilor claselor între care s-a făcut schimbul. Iteraţiile se repetă până când nici un vector nu mai poate fi mutat. Algoritmul poate fi descris în etapele următoare:

1. se alege o partiţie aleatoare a setului de obiecte (vectori)
2. pentru fiecare vector $\mathbf{x}_i$ din set, dacă nu este unic în clasa sa ω_j, se calculează costul mutării în altă clasă, ω_k, $k \neq j$; acest cost este
$$c_k = \frac{n_k}{n_k + 1} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2 - \frac{n_j}{n_j - 1} \left\|\mathbf{x}_i - \boldsymbol{\mu}_j\right\|^2 \tag{7.14}$$
3. vectorul $\mathbf{x}_i$ este mutat în clasa pentru care costul c_k este minim; se recalculează mediile claselor implicate în schimbare (ω_j şi ω_k)
4. dacă cel puţin un vector a fost mutat între două clase, algoritmul se reia de la pasul 2.

Principalul dezavantaj al acestei abordări este faptul că mediile claselor sunt recalculate după fiecare schimbare ce implică fiecare vector al mulţimii considerate, ceea ce are ca efect un volum mare de calcule. O simplificare a metodei provine din observaţia intuitivă că este normal ca un obiect să fie alocat (să aparţină) clasei de care este cel mai apropiat (în sensul distanţei la media acesteia). Folosind acestă observaţie, realocarea se poate face pentru toate obiectele considerate fără a fi nevoie de recalcularea mediilor claselor pentru fiecare obiect; recalcularea mediilor se va face după fiecare parcurgere completă a mulţimii de obiecte de partiţionat. Acest algoritm este algoritmul "Basic ISODATA[4]" (cunoscut şi sub numele de *k-means* - "cele k medii"). Algoritmul poate fi descris de următoarele etape:

1. se alege o partiţie aleatoare a setului de obiecte (vectori)
2. pentru fiecare vector $\mathbf{x}_i$ din set, se calulează distanţele sale la mediile tuturor claselor,
$$d_k = \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2 \tag{7.15}$$
3. în iteraţia următoare, vectorul $\mathbf{x}_i$ va fi mutat în clasa la care distanţa d_k este minimă
4. după parcurgerea completă a setului de vectori, se reactualizează mediile claselor

[4]ISODATA este acronimul de la "Iterative Self Organizing Data Analysis Technique" - tehnică iterativă cu auto-organizare de analiză a datelor, apărut prin 1965.

5. dacă (faţă de iteraţia anterioară) nici un vector nu a fost mutat în altă clasă (sau media nici unei clase nu s-a modificat), algoritmul se încheie; dacă nu, se reia algoritmul de la pasul 2.

Pentru nici unul dintre algoritmii iterativi prezentaţi nu se poate preciza numărul de parcurgeri ale setului de vectori (obiecte) de partiţionat. Numărul de iteraţii este puternic dependent de alegerea partiţiei iniţiale a vectorilor, precum şi de organizarea intrinsecă a acestora.

Figura 7.8 prezintă o imagine refăcută după o compresie prin cuantizare vectorială, cu raport mare de compresie (128). Se remarcă slaba calitate a imaginii refăcute, cauza fiind numărul mic (16) de vectori de cod folosiţi. În imagine sunt foarte vizibile frontierele dintre blocurile de 8 x 8 folosite.

Fig. 7.8: Imagine refăcută după codarea prin cuantizare vectorială; codarea a fost realizată cu blocuri de 8 x 8 pixeli; tabelul vectorilor de cod are 16 intrări, deci se obţine o compresie de 128.

Capitolul 8

SEGMENTAREA IMAGINILOR

Segmentarea imaginilor se referă la descompunerea unei scene (imagini) în componentele sale [9]. În urma procesului de segmentare vor fi extrase din imagine obiecte distincte, regiuni ce satisfac anumite criterii de uniformitate, sau alte elemente.

În [19] se propune o definiţie matematizată a procesului de segmentare, şi anume segmentarea unei imagini f este definită ca partiţionarea [completă] a lui f (8.1) într-un ansamblu de mulţimi disjuncte nevide şi conexe (8.2), ce satisfac fiecare un anumit criteriu C (8.3), criteriu ce nu mai este respectat pentru reuniunea oricăror două elemente ale partiţiei.

$$f = \bigcup_{i=1}^{C} f_i \ , \quad f_i \text{ conexe} \tag{8.1}$$

$$f_i \bigcap f_j = \emptyset, \forall i \neq j \text{ şi } f_i \text{ este conexă}, \forall i \tag{8.2}$$

$$\mathsf{C}(f_i) = TRUE, \forall i \text{ şi } \mathsf{C}\left(f_i \bigcup f_j\right) = FALSE, \forall i \neq j \tag{8.3}$$

Alegerea unei tehnici specifice de segmentare (partiţionare a imaginii) este legată de mai multe aspecte caracteristice imaginii de analizat şi cerinţelor utilizatorului. După natura şi conţinutul imaginii, tehnicile de segmentare trebuie să ţină cont de prezenţa în imagine a diverse categorii de artefacte:

- reflexii, iluminare neomogenă
- zgomot suprapus infomaţiei utile

- zone texturate

După primitivele de extras, tehnicile de segmentare se împart în două categorii fundamentale: tehnicile de segmentare orientate pe regiuni şi tehnicile de segmentare orientate pe contur. Primitivele extrase din imagine sunt regiuni (forme) şi zone texturate pentru tehnicile orientate pe regiuni, sau entităţi de tip discontinuitate (frontiere, segmente de dreaptă, unghiuri) pentru tehnicile orientate pe contur. În cadrul segmentării orientate pe regiuni se disting câteva categorii principale de tehnici:

- etichetarea imaginilor binare
- segmentarea pe histogramă
- creşterea şi fuziunea regiunilor
- segmentarea texturilor
- segmentarea prin metode de clustering

Tehnicile principale de segmentare orientată pe contururi sunt:

- extragerea contururilor prin metode de gradient şi derivative
- extragerea contururilor prin metode neliniare
- extragerea contururilor prin metode liniare optimale
- extragerea contururilor prin modelare matematică

În cele ce urmează se prezintă doar o parte dintre aceste tehnici, pe care le considerăm cele mai semnificative.

8.1 Segmentarea orientată pe regiuni

8.1.1 Segmentarea bazată pe histogramă

În general, operaţia de segmentare orientată pe regiuni urmăreşte extragerea din imagine a zonelor (regiunilor) ocupate de diferitele obiecte prezente în scenă. Un obiect se defineşte ca o entitate caracterizată de un set de parametri ale căror valori nu se modifică în

diferitele puncte ce aparţin entităţii considerate. Mai simplu, putem spune că obiectul are proprietatea de uniformitate a parametrilor de definiţie.

Unul dintre cei mai simpli parametri de definiţie este nivelul de gri al punctului. Nivelul de gri corespunde în scenă unei proprietăţi fizice [9] (reflectanţă, transmitivitate, valoare tristimulus, etc.) ce este preluat de senzorul de imagine şi asociat luminanţei imaginii. În acest caz, histograma imaginii (funcţia de densitate de probabilitate a variabilei aleatoare discrete ale cărei realizări sunt nivelele de gri din imagine) reflectă distribuţia în scenă a proprietăţii fizice înregistrate. Pentru o imagine f de $M \times N$ pixeli şi L nivele de gri, histograma este definită (8.4) ca probabilitatea de apariţie în imagine a diferitelor nivele de gri posibile.

$$h(i) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \delta(i - f(m,n)) \quad , \quad i = 0, 1, ...L-1 \tag{8.4}$$

Dacă nivelul de gri (respectiv proprietatea fizică pe care acesta o reprezintă) caracterizează în mod suficient obiectele din scenă, histograma imaginii va prezenta o structură de moduri dominante - intervale de nivele de gri ce apar cu probabilitate mai mare. Fiecare asemenea mod (maxim al histogramei) va reprezenta o anumită categorie de obiecte.

Ca exemplu imediat se poate cita cazul imaginilor obţinute prin scanarea documentelor scrise şi a tipăriturilor sau imaginile în infraroşu (temperatura punctelor este asociată nivelelor de gri astfel încât mai fierbinte însemnă mai alb). Pentru toate aceste tipuri de imagini histograma este de tipul celei prezentate în figura 8.1.

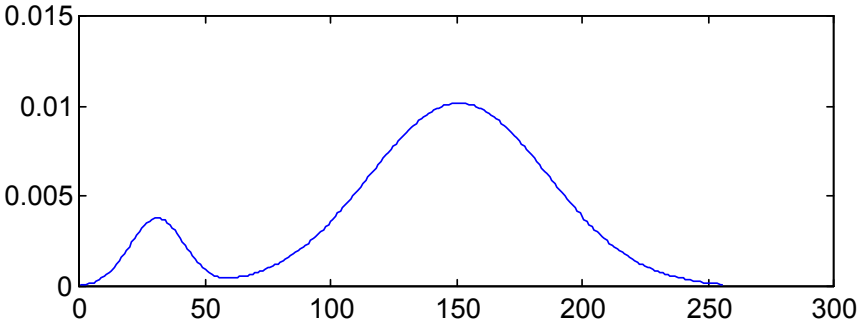


Fig. 8.1: Histogramă bimodală

Tehnici de prăguire (*thresholding*)

Separarea modurilor histogramei (şi deci identificarea obiectelor din imagine, respectiv caractere scrise / pagini albe şi obiecte fierbinţi / obiecte reci) se face prin alegerea unui

nivel de gri T, numit prag de segmentare. Acest prag de segmentare se alege pe minimul global al histogramei. Din imaginea iniţială f de nivele de gri se construieşte o imagine de etichete (imagine etichetată) g, conform transformării descrise de (8.5) (vezi figura 8.2).

$$g(m,n) = \begin{cases} E_0, 0 \leq f(m,n) < T \\ E_1, T \leq f(m,n) < L \end{cases} \tag{8.5}$$

Imaginea etichetată va fi descrisă de două etichete: E_0 pentru punctele al căror nivel de gri este mai mic decât pragul T şi E_1 pentru punctele al căror nivel de gri este mai mare decât pragul T. Etichetele E_0 şi E_1 pot fi valori numerice (0 şi 1, sau 0 şi 255) sau pot fi şiruri de simboluri sau alţi identificatori.

Fig. 8.2: Transformări punctuale de binarizare.

Transformarea (8.5) este o transformare punctuală (noua valoare din punctul (m,n) depinde doar de valoarea anterioară din punctul (m,n)) şi poartă numele de binarizare. Această denumire provine din faptul că rezultatul transformării (imaginea etichetată) este o imagine binară - deci o imagine caracterizată doar de două valori. Se poate remarca de asemenea faptul că binarizarea este un caz particular al transformării de modificare liniară a contrastului (2.2), în care limitele domeniilor de contrastare sunt egale ($T_1 = T_2$) şi contrastarea se face la valorile limită ale nivelelor de gri ($\alpha = 0$, $\beta = L - 1$)[1].

Segmentarea pe histogramă (numită şi prăguire sau *thresholding*) semnifică determinarea unor nivele de gri ce separă modurile histogramei. Tuturor punctelor din imagine al căror nivel de gri corespunde unui acelaşi mod, li se asociază o aceeaşi etichetă (număr, şir de simboluri), rezultând o imagine etichetată, ce pune în evidenţă diferitele obiecte ale scenei iniţiale.

[1] Există însă şi o variantă de binarizare cu două praguri (transformare punctuală numită decupare - "slicing"), (figura 8.2) definită de ecuaţia următoare:

$$g(m,n) = \begin{cases} E_0, \text{ dacă } f(m,n) < T_1 \text{ sau } f(m,n) > T_2 \\ E_1, \text{ în rest} \end{cases}$$

În cazul general al existenţei a mai multe praguri de segmentare T_k, transformarea de segmentare pe histogramă este descrisă de (8.6)

$$g(m,n) = E_k \quad \text{dacă} \quad T_k \leq f(m,n) < T_{k+1} \tag{8.6}$$

unde $T_0 = 0$, $T_C = L$, $k = 0, 1, ..., C-1$.

Pragurile T_k se aleg prin inspecţia histogramei, în minimele locale ale acesteia. Acest tip de segmentare multinivel este mai puţin eficient decât binarizarea, din cauza dificultăţii de stabilire a pragurilor care să izoleze eficient intervalele de interes din histogramă, mai ales atunci când numărul modurilor este mare. Trebuie de asemenea remarcat faptul că este necesară cunoaşterea numărului de tipuri de obiecte din imagine, pentru alegerea corespunzătoare a numărului de praguri de segmentare. În marea majoritate a cazurilor, segmentarea obţinută nu este corectă (există regiuni prost etichetate); ca o regulă generală de îmbunătăţire a performanţelor, se recomandă aplicarea, înaintea segmentării, a unor operaţii de filtrare (eliminare a zgomotului), contrastare, îmbunătăţire, netezire a histogramei - numite preprocesări.

În general, se admite clasificarea metodelor de segmentare pe histogramă [5] după atributele global, local şi dinamic. Aceste atribute se referă la modul de calcul al pragurilor de segmentare T_k în funcţie de nivelul de gri din fiecare punct al imaginii $f(m,n)$, coordonatele punctelor din imagine (m,n) şi o anumită proprietate locală $p(m,n)$ a punctului (m,n), conform (8.7):

$$T_k = T_k(f(m,n), p(m,n), (m,n)) \tag{8.7}$$

Segmentarea se numeşte globală dacă pragurile depind doar de nivelele de gri ale punctelor imaginii:

$$T_k = T_k(f(m,n)) \tag{8.8}$$

Segmentarea multinivel descrisă de (8.6) este în mod evident o metodă de tip global.

Segmentarea se numeşte locală dacă pragurile depind de nivelul de gri şi de anumite atribute locale calculate pentru vecinătăţi ale fiecărui punct:

$$T_k = T_k(f(m,n), p(m,n)) \tag{8.9}$$

Segmentarea se numeşte dinamică dacă pragurile depind de poziţionarea punctelor în imagine (forma cea mai generală a modului de deducere pragurilor) (8.7).

Determinarea automată a pragurilor: metoda Bhattacharya

Metoda Bhattacharyya se bazează pe descompunerea histogramei în moduri individuale Gaussiene, adică se încearcă exprimarea histogramei imaginii ca o sumă ponderată de funcţii de densitate de probabilitate de tip normal (Gaussian). Modelarea modurilor histogramei imaginilor prin distribuţii normale este o presupunere ce se întâlneşte în multe tehnici de prelucrare şi analiză şi pare a fi justificată de considerarea imaginii ca provenind dintr-o imagine ideală, în care fiecare tip de obiect este reprezentat de un unic nivel de gri, peste care s-a suprapus un zgomot alb, aditiv, gaussian. În acest mod, mediile modurilor din histogramă corespund nivelelor de gri ce caracterizează obiectele scenei, iar varianţele acestor moduri sunt determinate de zgomotul suprapus imaginii (care nu este obligatoriu să afecteze în acelaşi mod toate nivelele de gri).

Pentru segmentarea după metoda Bhattacharyya nu este necesară precizarea unui număr de clase (praguri de segmentare), acesta urmând a fi determinat în mod automat. Ideea de plecare a metodei este de a determina parametrii caracteristici ai unei distribuţii normale. Pentru o distribuţie normală

$$N(\mu_k, \sigma_k)(x) = \frac{1}{\sigma_k\sqrt{2\pi}} e^{-\frac{(x-\mu_k)^2}{2\sigma_k^2}}$$

derivata logaritmului este:

$$\frac{\delta \ln N(\mu_k, \sigma_k)(x)}{\delta x} = -\frac{x}{\sigma_k^2} + \frac{\mu_k}{\sigma_k^2} = m_k x + n_k \tag{8.10}$$

Se observă prin examinarea expresiei (8.10) că derivata logaritmului distribuţiei normale este o dreaptă de pantă negativă, din ai cărei parametri se pot deduce media şi varianţa distribuţiei. Parametrii statistici ai distributiei sunt daţi de ecuaţiile (8.11).

$$\sigma_k = \sqrt{\frac{1}{|m_k|}}, \text{şi } \mu_k = \frac{n_k}{|m_k|} \tag{8.11}$$

Această observaţie poate fi aplicată şi pentru o mixtură de distribuţii normale. Să considerăm că histograma h a imaginii este compusă prin superpoziţia aditivă a C moduri gaussiene $N(\mu_k, \sigma_k)$, adică:

$$h(x) = \sum_{k=1}^{C} w_k N(\mu_k, \sigma_k)(x)$$

Din parametrii dreptei se pot determina deci conform (8.11) parametrii statistici ai distribuţiei locale.

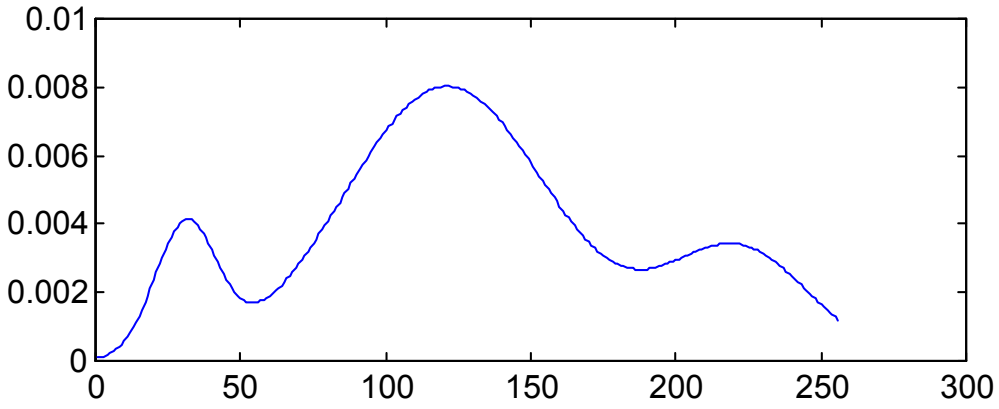


Fig. 8.3: Histogramă cu trei moduri normale

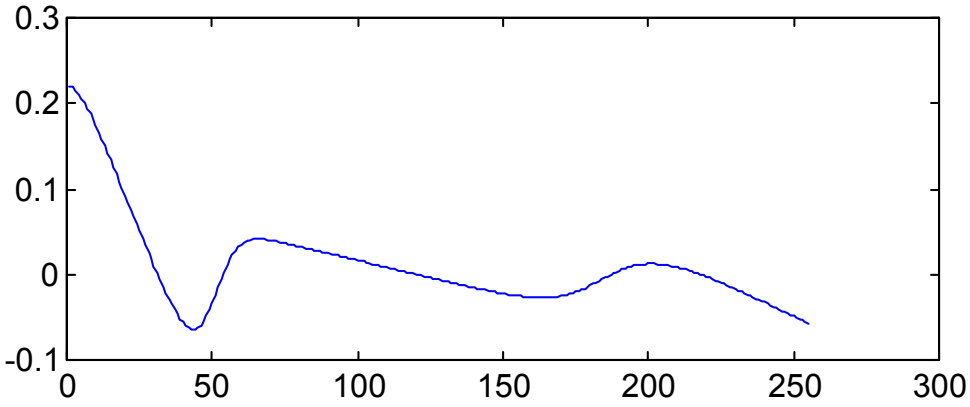


Fig. 8.4: Aplicarea metodei Bhattacharyya pentru histograma trimodală prezentată anterior; se pot observa intervalele pe care funcţia este liniară şi descrescătoare, ce corespund modurilor.

Aşadar, pentru aplicarea metodei la segmentarea pe histogramă a imaginilor, se va studia comportamentul derivatei logaritmului histogramei, adică a funcţiei $z(a)$:

$$z(a) = \ln \frac{h(a)}{h(a-1)}, a = \overline{1, L-1} \tag{8.12}$$

Pentru funcţia astfel construită, se determină intervalele pe care acesta este descrescătoare (vezi figura 8.4); limitele superioare ale acestor intervale sunt pragurile T_k de segmentare pe histogramă. Suplimentar, pe fiecare dintre aceste intervale se poate face o aproximare liniară a punctelor şi pe baza parametrilor deduşi pentru dreapta de aproximare se pot calcula, conform (8.11) parametrii statistici locali.

Principalele inconveniente ale metodei derivă din faptul că presupunerea alcătuirii histogramei imaginii numai din moduri gaussiene nu este întotdeauna adevărată. Ca rezultat, metoda Bhattacharrya va identifica un număr mai mare de praguri decât este necesar, producând fenomenul de suprasegmentare.

Segmentarea cu prag optim

Metoda de segmentare cu prag optim [5], [3], [19] face apel la teoria deciziilor (criteriul de decizie Bayes) pentru stabilirea valorii pragurilor de segmentare ce optimizează un anumit criteriu de eroare. Informaţiile apriori necesare pentru aplicarea unei asemenea tehnici sunt numărul de tipuri de obiecte din imagine, C, procentele de ocupare a imaginii de către fiecare tip de obiecte, P_i şi distribuţia nivelelor de gri ce caracterizează fiecare tip de obiect, $p_i(x)$. Atunci histograma imaginii va fi determinată de mixtura distribuţiilor tipurilor de obiecte:

$$h(x) = \sum_{i=1}^{C} P_i p_i(x), \sum_{i=1}^{C} P_i = 1 \tag{8.13}$$

Cazul cel mai simplu şi mai des folosit este cel al binarizării (8.5), în care trebuie determinat un unic prag T ce separă distribuţiile celor două tipuri de obiecte din imagine (în mod tipic, obiecte "utile" şi fundal). Criteriul ce se urmăreşte optimizat este eroarea de segmentare (clasificare) a punctelor din imagine, adică este dat de numărul de pixeli ce aparţin primului tip de obiect, dar au nivelul de gri mai mare ca pragul T (fiind deci alocaţi greşit celui de-al doilea tip de obiect) şi numărul de pixeli ce aparţin celui de-al doilea tip de obiect, dar au nivelul de gri mai mic decât pragul de segmentare T (fiind deci alocaţi greşit primului tip de obiect). Aşadar, eroarea de segmentare va fi dată de (8.14):

$$E(T) = P_1 \int_{T}^{+\infty} p_1(x)dx + P_2 \int_{-\infty}^{T} p_2(x)dx \tag{8.14}$$

Pragul optim va minimiza eroarea de segmentare a pixelilor. Minimizarea erorii (8.14) conduce la rezolvarea ecuaţiei (8.15), în necunoscuta T.

$$\frac{\partial E(T)}{\partial T} = 0 \tag{8.15}$$

Derivând (8.14) se obţine forma echivalentă a ecuaţiei (8.15):

$$P_1 p_1(T) = P_2 p_2(T) \tag{8.16}$$

După cum am menţionat şi în secţiunea dedicată tehnicilor de segmentare ce nu folosesc informaţii apriori despre imagine (metoda Bhattacharyya), presupunerea că distribuţia nivelelor de gri a diferitelor tipuri de obiecte este de tip normal (Gaussian) este relativ des întâlnită. În aceste condiţii, distribuţiile $p_1(x)$ şi $p_2(x)$ sunt distribuţii normale, $N_1(\mu_1, \sigma_1)(x)$ şi $N_2(\mu_2, \sigma_2)(x)$, iar ecuaţia (8.16) devine:

$$P_1 \frac{1}{\sigma_1\sqrt{2\pi}} e^{-\frac{(T-\mu_1)^2}{2\sigma_1^2}} = P_2 \frac{1}{\sigma_2\sqrt{2\pi}} e^{-\frac{(T-\mu_2)^2}{2\sigma_2^2}}$$

Prin logaritmare, se obţine urmatoarea ecuaţie de gradul 2 în necunoscuta T:

$$T^2\left(\frac{1}{\sigma_1^2} - \frac{1}{\sigma_2^2}\right) - 2T\left(\frac{\mu_1}{\sigma_1^2} - \frac{\mu_2}{\sigma_2^2}\right) + \left(\frac{\mu_1^2}{\sigma_1^2} - \frac{\mu_2^2}{\sigma_2^2}\right) - 2\ln\frac{P_1}{P_2}\frac{\sigma_2}{\sigma_1} = 0$$

Una dintre simplificările uzuale este presupunerea că $\sigma_1 = \sigma_2 = \sigma$; această presupunere implică modelarea imaginii în nivele de gri ca o imagine cu doar două nivele de gri μ_1 şi μ_2, afectată de un zgomot Gaussian aditiv, având varianţa σ^2. În aceste condiţii, ecuaţia de gradul 2 devine o ecuaţie liniară, a carei soluţie este:

$$T = \frac{\mu_1 + \mu_2}{2} - \frac{\sigma^2}{\mu_1 - \mu_2}\ln\frac{P_1}{P_2}$$

Metoda se poate extinde şi pentru imagini ce conţin mai mult de două tipuri de obiecte; în acest caz este însă necesară presupunerea suplimentară de localizare a modurilor, astfel încât să se poată considera, ca şi în cazul metodei Bhattacharyya, că influenţa fiecărui mod este limitată la intervale nesuprapuse de nivele de gri.

8.1.2 Creşterea şi fuziunea regiunilor

Pentru aplicarea cu succes a tehnicilor de segmentare pe histogramă prezentate anterior trebuiesc îndeplinite neapărat câteva condiţii (deja enunţate). Aplicarea tehnicilor de segmentare pe histogramă este condiţionată în primul rând de reprezentarea diferitelor clase de obiecte din imagine pe intervale de nivele de gri diferite care nu se suprapun (sau se suprapun parţial pe porţiuni foarte mici); apoi este necesară cunoaşterea numărului de tipuri de obiecte diferite. În fine, se presupune că valorile prag corespunzătoare se pot determina cu o precizie corespunzătoare.

Chiar în cazurile în care toate aceste condiţii enunţate sunt îndeplinite, nu se poate garanta condiţia de conexitate a regiunilor obţinute în urma segmentării (8.2). Acest lucru este evident, atât timp cât două obiecte de acelaşi tip, neconexe, primesc prin segmentarea pe histogramă o aceeaşi etichetă, şi formează în imaginea de etichete o regiune neconexă. O metodă care respectă toate condiţiile impuse de definiţia metematică a segmentării, şi anume (8.1), (8.2) şi (8.3), este creşterea regiunilor.

Creşterea regiunilor

Principiul pe care se bazează creşterea regiunilor este simplu: se aleg în imagine puncte reprezentative pentru fiecare obiect individual şi categorie de obiecte, pe baza cărora are loc un proces de aglomerare a pixelilor vecini acestora, ce au aceleaşi proprietăţi (în particular acelaşi nivel de gri). În urma acestui proces de aglomerare (adăugare de puncte) se obţin zone (regiuni) de pixeli cu aceleaşi caracteristici, deci obiecte individuale. Procesul se opreşte în momentul în care fiecare punct al imaginii a fost alocat unei regiuni. Evident, metoda astfel descrisă pe scurt, are două etape esenţiale: alegerea punctelor de start (puncte iniţiale), numite germeni sau seminţe, şi creşterea propriu-zisă a regiunilor [19], [2].

Numărul final de regiuni rezultate este egal cu numărul de germeni aleşi iniţial pentru creştere. În principiu, este de dorit ca fiecare obiect individual aflat în imagine să fie marcat de câte un germene. Dacă în interiorul unui aceluiaşi obiect se găsesc mai mulţi germeni, pentru fiecare dintre ei va fi crescută o regiune; acesta face ca obiectul iniţial să fie împărţit artificial prin segmentare în mai multe regiuni. Parţial, acest neajuns se poate corecta printr-o etapă ce urmează creşterii regiunilor, şi anume fuziunea regiunilor adiacente ce au proprietăţi asemănătoare. Dacă în interiorul unui obiect nu este ales nici un germene, obiectul respectiv va fi înglobat de regiunile ce cresc pornind de la germeni din vecinătatea sa spaţială; astfel, respectivul obiect nu apare ca o regiune distinctă şi este pierdut, rezultând o eroare gravă de segmentare.

Pentru a preveni efectul unor neuniformităţi de iluminare pe suprafaţa imaginii, acesta este împărţită în ferestre nesuprapuse; în fiecare astfel de fereastră se alege un număr de germeni, al căror plasament spaţial este aleator (germenii se distribuie uniform pe

suprafaţa imaginii). Germenii se aleg astfel încât nivelul lor de gri să fie reprezentativ pentru obiectele prezente local (deci nivelul de gri al germenilor trebuie să corespundă unor maxime ale histogramei locale). În plus, trebuie verificat ca plasamentul spaţial al germenilor să se facă în interiorul regiunilor şi nu pe frontiera acestora. Verificarea se poate face simplu pe baza calculului unui operator derivativ local, ca de exemplu laplacianul (37); dacă valoarea acestuia nu depăşeşte un anumit procent prestabilit (10% - 20%) din diferenţa maximă de nivele de gri a ferestrei, punctul ales este considerat ca plasat corect.

O verificare suplimentară încearcă să prevină o eventuală suprasegmentare[2] (împărţirea artificială a unui acelaşi obiect în mai multe regiuni), eliminând germenii plasaţi în interiorul aceluiaşi obiect. Verificarea se face pe baza calculului variaţiei nivelelor de gri de-a lungul drumurilor[3] arbitrare ce unesc perechi de germeni. Dacă există o cale ce uneşte doi germeni de-a lungul căreia nivelul de gri nu variază cu mai mult de 20% - 30% din diferenţa maximă a nivelelor de gri din fereastră, cei doi germeni sunt plasaţi în interiorul unei zone de nivele de gri uniforme, deci în interiorul unui acelaşi obiect. În aceste condiţii unul dintre cei doi germeni ai perechii este eliminat, deoarece este redundant. Dacă de-a lungul tuturor căilor ce unesc perechea de germeni nivelul de gri variază mai mult decât pragul ales, atunci se consideră că cei doi germeni sunt plasaţi în interiorul unor obiecte diferite (deoarece căile ce unesc germenii traversează regiuni de frontieră). În practică, examinarea tuturor drumurilor (căilor) ce unesc perechi de germeni este extrem de costisitoare din punctul de vedere al timpului de calcul. De aceea se verifică doar căile formate din segmente verticale şi orizontale, şi eventual, dreapta ce uneşte cele două puncte (dacă această dreaptă poate fi reprezentată de o secvenţă de puncte conexe) (vezi figura 8.5).

Valorile procentuale ale pragurilor de comparaţie, precum şi numărul de germeni distincţi ce rămân după procesul de reducere, nu trebuie considerate ca fixe; nu există valori standardizate şi alegerea acestora se face pe baza condiţiilor particulare (legate de conţinutul imaginii) şi a experienţei utilizatorului.

Pornind de la germenii aleşi, regiunile sunt obţinute printr-un proces de creştere aproape simultană, început de la aceştia, până când toţi pixelii imaginii sunt repartizaţi unei regiuni. Cvasi-simultaneitatea creşterii poate fi realizată cu un algoritm serial, prin alocarea pixelilor ce sunt adiacenţi (vecini) zonelor deja segmentate. Acestă alocare trebuie să ţină seama de criteriul ca regiunile crescute să fie uniforme: nivelul de gri al pixelului ce se adaugă nu trebuie să difere cu mai mult de un prag prestabilit faţă de nivelul de gri al germenului regiunii la care se alocă. În acelaşi timp, la o singură trecere, numărul de puncte ce se adaugă unei regiuni nu poate depăşi un număr prestabilit (condiţia încearcă

[2] În general, prin suprasegmentare se înţelege partiţionarea imaginii într-un număr de regiuni mai mare decât numărul de obiecte. Există şi noţiunea reciprocă de subsegmentare: împărţirea imaginii într-un număr de regiuni mai mic ca numărul de obiecte.

[3] Un drum între două puncte ale imaginii este o secvenţă ordonată de puncte ale imaginii, vecine două câte două (relativ la un anumit tip de conexitate, V_4 sau V_8), care are drept capete punctele considerate.

Fig. 8.5: Reducerea numărului de germeni: germenii 1 şi 2 sunt uniţi de o cale cu segmente paralele cu orizontala şi verticala de intensitate constantă, deci sunt redundanţi; germenii 3 şi 4 sunt uniţi de o cale dreaptă de aceeşi intensitate, deci sunt redundanţi; orice cale ce uneşte germenii 1 şi 3 are o diferenţă mare de intensitate.

să asigure creşterea relativ uniformă şi izotropă a tuturor regiunilor).

Dacă adăugarea de noi pixeli se blochează (criteriul de uniformitate nu mai este respectat), diferenţa maxim admisă pentru nivelul de gri poate fi crescută în etape, până la epuizarea pixelilor imaginii.

Avantajele pe care le are o asemenea tehnică de creştere a regiunilor sunt acelea că nu mai este necesară nici o informaţie privind conţinutul imaginii, regiunile crescute sunt conexe şi nu există puncte neetichetate (nealocate vreunei regiuni) şi poziţia frontierelor dintre diferitele regiuni corespunde poziţiei frontierelor percepute subiectiv în imagine.

Fuziunea regiunilor

O extindere a principiului utilizat în creşterea regiunilor, şi anume adăugarea la o regiune a unor entităţi (pixeli în acest caz) a căror proprietăţi sunt similare cu cele ale unui obiectului de bază (regiunea), se află la baza tehnicilor de fuziune a regiunilor [9]. Fuziunea regiunilor constă în reunirea iterativă a regiunilor adiacente (începând de la nivelul unor entităţi atomice ale imaginii - deci pixelii) până când regiunile adiacente devin suficient de diferite. Procesul de fuziune a regiunilor poate fi aplicat şi în urma unei creşteri a regiunilor, pentru a înlătura efectele unei eventuale suprasegmentări. Există mai multe criterii de fuziune a regiunilor adiacente, a căror acţiune de verificare a deosebirii între regiuni se face fie prin inspecţia frontierei comune, fie prin caracterizarea interiorului regiunii.

Pentru două regiuni adiacente R_i şi R_j, al căror perimetru este $Perim(R_i)$ şi $Perim(R_j)$, putem determina $P_m = \min(Perim(R_i), Perim(R_j))$ şi P lungimea frontierei comune[4].

[4]Lungimea frontierei comune poate fi măsurată fie ca perimetru, fie ca număr de puncte ce o compun.

Pe această frontieră comună se disting puncte *slabe* (în număr de n_s) şi puncte *tari* (în număr de n_t). Un punct slab este acel punct pentru care diferenţa nivelelor de gri între vecinii din regiunile adiacente este foarte mică (mai mică decât un anumit prag fixat). Un punct tare este acel punct pentru care diferenţa de nivele de gri între vecinii din regiunile adiacente este foarte mare (mai mare ca un anumit prag fixat). Cu aceste notaţii, criteriile de fuziune a regiunilor R_i şi R_j sunt:

- dacă numărul de puncte slabe raportat la perimetrul minim este important, $\frac{n_s}{P_m} > \theta_1$
- dacă numărul de puncte slabe de pe frontiera comună este mare, $\frac{n_s}{P} > \theta_2$
- dacă numărul de puncte tari de pe frontiera comună este mic, $\frac{n_t}{P} < \theta_3$.

Parametrul θ_1 controlează dimensiunea regiunilor ce se unesc şi se alege în general cu valoarea 0.5 (de exemplu o valoare apropiată de 1 implică unirea a două regiuni numai dacă una dintre ele este aproape înconjurată de cealaltă). Valori tipice pentru parametrii θ_2 şi θ_3 sunt 0.75 şi 0.2.

Abordarea fuziunii pe baza caracterizării interiorului regiunilor necesită definirea a două componente: o modalitate de caracterizare a proprietăţilor regiunilor şi o modalitate de a defini "apropierea" sau similaritatea dintre trăsături în termeni numerici.

Vectorul de trăsături ce caracterizează o regiune se compune din momente statistice ale variabilei aleatoare ale cărei realizări particulare sunt nivelele de gri din regiune repectivă; nu pot lipsi din acest vector nivelul de gri mediu al regiunii şi varianţa acestuia.

În [9] se propun patru funcţii de măsură a asemănării între perechi de vectori; pentru doi vectori $\mathbf{x}_i$ şi $\mathbf{x}_j$, având aceeaşi dimensiune, acestea se definesc ca:

$$F_1(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle \text{ (produsul scalar dintre vectori)}$$

$$F_2(\mathbf{x}_i, \mathbf{x}_j) = \frac{\langle \mathbf{x}_i, \mathbf{x}_j \rangle}{\langle \mathbf{x}_i, \mathbf{x}_i \rangle + \langle \mathbf{x}_j, \mathbf{x}_j \rangle - \langle \mathbf{x}_i, \mathbf{x}_j \rangle} \text{ (similaritatea dintre vectori)}$$

$$F_3(\mathbf{x}_i, \mathbf{x}_j) = \frac{\langle \mathbf{x}_i, \mathbf{x}_j \rangle}{\sqrt{\langle \mathbf{x}_i, \mathbf{x}_i \rangle \langle \mathbf{x}_j, \mathbf{x}_j \rangle}} \text{ (corelaţia normalizată dintre vectori)}$$

$$F_4(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j) \cdot A \cdot (\mathbf{x}_i - \mathbf{x}_j)^T$$

(distanţa generalizată dintre vectori, unde A este o matrice pozitiv definită)

Pentru primele trei funcţii, o valoare mai mare corespunde unei asemănări mai mari între vectori (valorile maxime pentru $F_2(\mathbf{x}_i, \mathbf{x}_j)$ şi $F_3(\mathbf{x}_i, \mathbf{x}_j)$ sunt 1). Pentru funcţia de similaritate bazată pe distanţa generalizată, o valoare mai mică corepunde unei asemănări mai puternice între vectori. Prin particularizarea matricii A se pot obţine diferite distanţe, ca distanţa Euclidiană obişnuită (A fiind matricea unitate, $A = I$), distanţe Euclidiene ponderate (dacă A este o matrice diagonală), sau distanţa Mahalanobis (dacă A este o matrice de covariaţie a componentelor).

8.2 Segmentarea orientată pe contururi

Într-o imagine, variaţiile de valoare ale pixelilor reprezintă schimbări ale proprietăţilor fizice sau geometrice ale scenei sau ale obiectului observat. Aceste schimbări pot corespunde fizic la variaţiile iluminării, schimbările de orientare sau de distanţă faţă de observator, schimbări de reflectanţă ale suprafeţelor, variaţii de absorbţie a radiaţiei. Într-un număr mare de cazuri, aceste variaţii de intensitate sunt informaţii importante pentru operaţiile ce urmează segmentării, informaţii ce corespund frontierelor regiunilor determinate de obiectele scenei.

8.2.1 Metode derivative

Principiul acestei metode constă în definirea punctelor de contur ca fiind acei pixeli ai imaginii în care apar schimbări importante (abrupte) ale nivelului de gri. Deci, măsurarea acestei variaţii se va face prin operatori derivativi de tip gradient.

Pentru o imagine cu suport spaţial continuu, pe direcţia unei muchii, derivata va fi maximă. Derivata imaginii pe direcţia r, ce face unghiul θ cu orizontala, este dată de combinaţia liniară a derivatelor parţiale pe direcţiile orizontală şi verticală (8.17):

$$\frac{\partial f}{\partial r} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial r} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial r} = \frac{\partial f}{\partial x}\cos\theta + \frac{\partial f}{\partial y}\sin\theta$$

$$\frac{\partial f}{\partial r} = f_x \cos\theta + f_y \sin\theta \tag{8.17}$$

Valoarea maximă a acestei derivate, calculate după unghiul θ este determinată de ecuaţia

$$\frac{\partial}{\partial \theta}\left(\frac{\partial f}{\partial r}\right) = -f_x \sin\theta + f_y \cos\theta = 0$$

ce are soluţia evidentă:

$$\theta_0 = \arctan\left(\frac{f_y}{f_x}\right) \tag{8.18}$$

Pe această direcţie, modulul gradientului este:

$$\left(\frac{\partial f}{\partial r}\right)_{\max} = \sqrt{f_x^2 + f_y^2} \tag{8.19}$$

Din punct de vedere practic, implementarea acestei metode implică atunci calcularea, pentru fiecare punct al imaginii, a derivatelor parţiale f_x şi f_y, calcularea modulului gradientului maxim (8.19) şi a direcţiei acestuia (8.18). Valoarea gradientului maxim din fiecare punct al imaginii este apoi comparată cu un prag fixat: dacă pragul este depăşit

(deci gradientul maxim în pixelul respectiv este suficient de important) atunci pixelul testat este pixel de contur.

Realizarea derivatelor parţiale după direcţiile orizontală şi verticală implică translaţia în discret a lui f_x şi f_y:

$$f_x = \frac{\partial f}{\partial x} = \frac{\Delta f(m,n)}{\Delta m}$$
$$f_y = \frac{\partial f}{\partial y} = \frac{\Delta f(m,n)}{\Delta n}$$

Aceste derivate parţiale discrete pot avea mai multe implementări:

$$f_x = f(m,n) - f(m+1,n),\ f_y = f(m,n) - f(m,n+1) \tag{8.20}$$

$$f_x = f(m-1,n) - f(m,n),\ f_y = f(m,n-1) - f(m,n) \tag{8.21}$$

$$f_x = f(m-1,n) - f(m+1,n),\ f_y = f(m,n-1) - f(m,n+1) \tag{8.22}$$

Toate expresiile date de (8.20), (8.21), (8.22) sunt combinaţii liniare ale valorilor unor pixeli din imagine, situaţi în vecinătatea pixelului curent din poziţia (m,n). Deci toate aceste operaţii se pot realiza prin filtrări liniare cu măşti potrivite: (8.23) pentru (8.20), (8.24) pentru (8.21), (8.25) pentru (8.22).

$$W_x = \begin{pmatrix} 1 & -1 \end{pmatrix},\ W_y = \begin{pmatrix} 1 \\ -1 \end{pmatrix} \tag{8.23}$$

$$W_x = \begin{pmatrix} 1 & -1 \end{pmatrix},\ W_y = \begin{pmatrix} 1 \\ -1 \end{pmatrix} \tag{8.24}$$

$$W_x = \begin{pmatrix} 1 & 0 & -1 \end{pmatrix},\ W_y = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix} \tag{8.25}$$

Schema bloc a extragerii de contururi este reprezentată în figura 8.6.

Harta de orientări este o imagine care conţine, pentru fiecare pixel, orientarea gradientului de modul maxim în punctul respectiv, şi este în general folosită la prelucrarea suplimentară a contururilor (conectare de contururi, extragere direcţională de contururi). Harta de contururi este o imagine binară în care punctele marcate (puncte-obiect) corespund poziţiei punctelor de contur (puncte cu gradient de modul mare). O simplificare uzuală practicată este înlocuirea normei L2 din calculul modulului maxim al gradientului (8.19) cu norma L1, ceea ce conduce la aproximarea:

$$\left(\frac{\partial f}{\partial r}\right)_{\max} \approx |f_x| + |f_y|$$

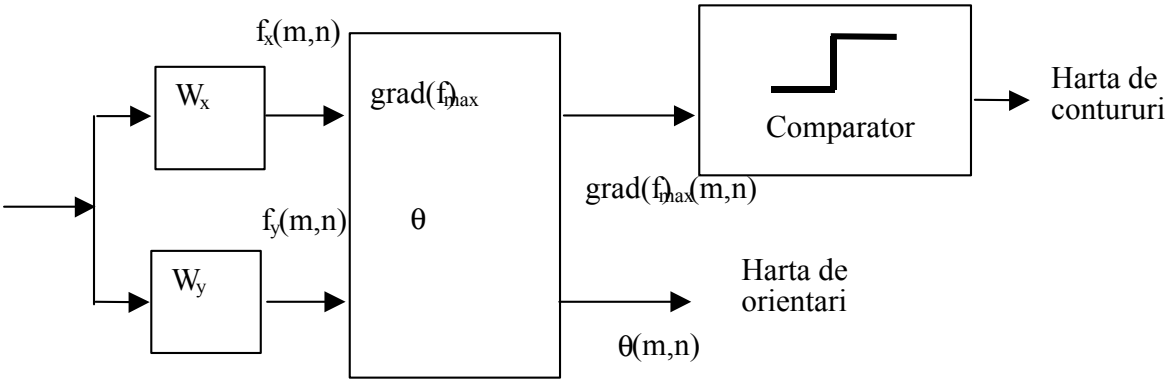


Fig. 8.6: Schema bloc a extractorului de contururi bazat pe metoda de gradient.

Folosirea măştilor de derivare pe verticală şi orizontală prezentate are însă serioase neajunsuri: dimensiunea lor mică face ca rezultatele să fie extrem de sensibile în prezenţa zgomotului. În aceste condiţii a apărut naturală ideea de a combina filtrarea de derivare cu o filtrare de netezire, care să mai reducă efectele zgomotului. Considerând zgomotul de tip gaussian, aditiv, filtrarea de netezire are ca efect secundar micşorarea contrastului frontierelor obiectelor din imagine (efectul de înceţoşare, sau *blur*). Pentru ca în aceste condiţii detecţia contururilor să nu fie afectată, trebuie ca operaţia de mediere prin care se realizează netezirea să se facă pe o direcţie perpendiculară direcţiei contururilor căutate [3]. Atunci derivarea pe verticală se combină cu o operaţie de netezire cu mască orizontală $\begin{pmatrix} 1/3 & \boxed{1/3} & 1/3 \end{pmatrix}$ şi derivarea pe orizontală se combină cu o operaţie de netezire cu mască verticală $\begin{pmatrix} 1/3 \\ \boxed{1/3} \\ 1/3 \end{pmatrix}$. Dacă folosim pentru derivare masca W_y din (8.23), masca de filtrare rezultantă va fi $\begin{pmatrix} 1/3 & \boxed{1/3} & 1/3 \\ -1/3 & -1/3 & -1/3 \end{pmatrix}$. În cazul general se pot folosi însă pentru netezire medieri ponderate (şi nu neapărat medieri aritmetice), care să acorde o mai mare importanţă pixelului curent prelucrat, ca de exemplu $\frac{1}{c+2}\begin{pmatrix} 1 & \boxed{c} & 1 \end{pmatrix}$ şi se preferă folosirea operatorilor de derivare simetrici, de tipul (8.25). Ceea ce rezultă pentru operatorii de derivare orizontală şi verticală sunt măştile:

$$W_x = \begin{pmatrix} 1 & 0 & -1 \\ c & \boxed{0} & -c \\ 1 & 0 & -1 \end{pmatrix}, \; W_y = \begin{pmatrix} 1 & c & 1 \\ 0 & \boxed{0} & 0 \\ -1 & -c & -1 \end{pmatrix} \tag{8.26}$$

Prin particularizarea valorilor constantei de ponderare c se pot obţine diferite tipuri de operatori de extragere de contur clasici: Prewitt ($c = 1$), Izotrop ($c = \sqrt{2}$), Sobel ($c = 2$) [9]. Se remarcă faptul că constanta de ponderare globală a măştii de filtare este neesenţială, întrucât condiţia de normare ce trebuie îndeplinită este cea pentru filtre de contrastare (derivare) (3.9): suma coeficienţilor măştii să fie nulă. Figura 8.7 prezintă

harta de intensitate a tranziţiilor (modulul maxim al gradientului, $\left(\frac{\partial f}{\partial r}\right)_{\max}$) iar figura 8.8 prezintă harta binară de contururi extrasă prin compararea hărţii de intensităţii cu un prag fixat (binarizarea hărţii de intensitate).

Fig. 8.7: Harta de intensitate a contururilor (modulul maxim al gradientului) calculat cu măşti Prewitt pentru imaginea "lena".

Fig. 8.8: Harta binară de contururi extrasă din harta de intensităţi precedentă.

Informaţia de orientare este în general folosită în etape următoare ale prelucrării; unghiurile determinate după (8.18) oferă un unghi "exact" al direcţiei conturului în punctul curent, calculat cu un efort semnificativ de calcul (împărţire şi calcul de arctangentă). În practică, această informaţie este prea exactă: pe grila pătrată de eşantionare nu se pot reprezenta cu uşurinţă drepte continue după orice direcţie[5]; câteva direcţii sunt fa-

[5]Problema trasării figurilor geometrice oarecari (inclusiv a dreptelor) este rezolvată de grafica pe

vorizate şi uşor de utilizat (vertical, orizontal, cele două diagonale). În acest caz se poate măsura în fiecare modulul gradientului după aceste câteva direcţii importante, şi apoi se poate alege direcţia după care acest modul este maxim. Acesta este principul operatorilor compas.

Un operator compas este definit de un număr de măşti de derivare (corespunzătoare în continuare unor filtrări liniare) pe direcţiile principale (vertical, orizontal, cele două diagonale), în cele două sensuri. Compasul clasic are $D = 8$ măşti de filtrare (identice două câte două, mai puţin semnul), fiecare dintre ele realizând o derivare după o direcţie multiplu de 45°. Schema bloc a unui operator compas este prezentată în figura 8.9; se remarcă faptul că, odată determinată valoarea maximă a modulului gradientului în pixelul curent (m, n), obţinerea hărţii de contururi se face ca şi la un operator de gradient clasic.

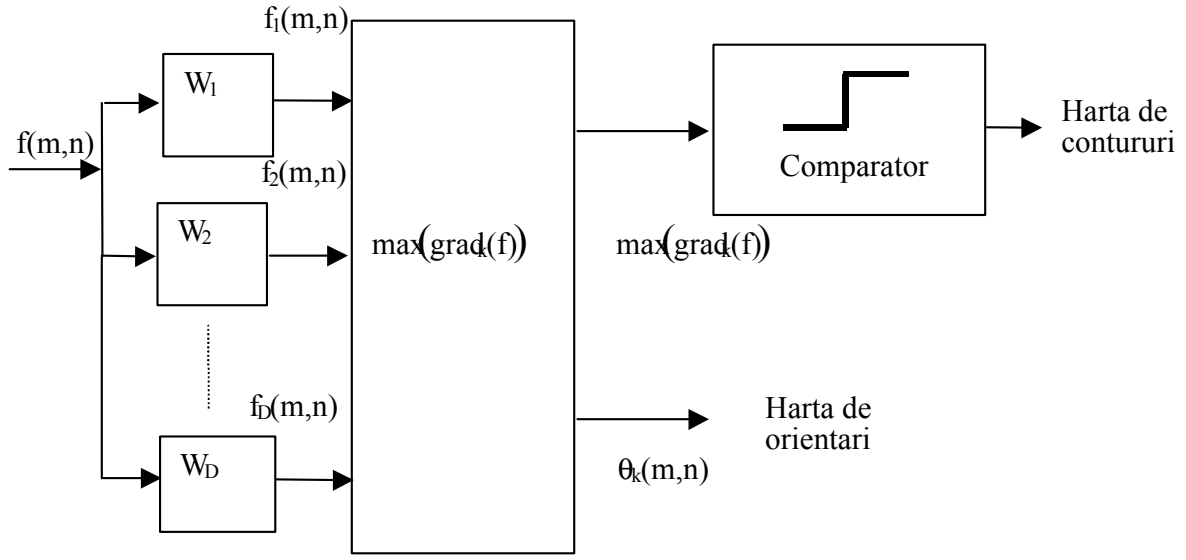


Fig. 8.9: Schema bloc a unui operator compas de extragere a contururilor.

Un exemplu de măşti de derivare direcţională sunt măştile următoare (indexate după direcţia geografică pe care calculează derivata): $W_N = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$, $W_{NV} = \begin{pmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$, $W_V = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$, $W_{SV} = \begin{pmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{pmatrix}$, $W_S = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}$, $W_{SE} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -1 \end{pmatrix}$, $W_E = \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix}$, $W_{NE} = \begin{pmatrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{pmatrix}$. După cum

calculator, prin algoritmi de rendering.

se remarcă, familia de măşti se poate genera pornind de la una dintre măştile Prewitt, prin translaţii circulare cu o poziţie a frontierei măştii în jurul centrului ei; în mod analog se pot obţine operatori compas bazaţi pe masca Sobel sau pe gradientul izotrop sau pe masca Kirsch $\begin{pmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{pmatrix}$. Precizia unghiulară a operatorilor compas este deci determinată de numărul de orientări diferite pe care se calculează derivatele, şi deci de numărul de translaţii ale frontierei măştii; pentru o mască pătrată de bază de dimensiune N, precizia unghiulară a operatorului compas este de $90°/(N-1)$.

Unul dintre principalele dezavantaje ale metodelor de gradient este precizia slabă de localizare a conturului (a centrului tranziţiei) în condiţiile unei pante puţin abrupte a acestuia (tranziţii slabe, graduale). Derivata a doua poate fi însă folosită pentru a determina capetele tranziţiei (cele două extreme), sau pentru a marca centrul tranziţiei (trecerea sa prin zero); figura 8.10 ilustrează această comportare pentru cazul unidimensional.

Operatorul bazat pe trecerea prin zero a derivatei secunde este operatorul "zero-crossing" [9]. În cazul imaginilor (semnale cu suport bidimensional) trebuie luată în considerare derivata secundă după ambele direcţii, combinate în laplacian:

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

În cazul discret, măşti ce implementează laplacianul sunt măştile $W_5 - W_7$, prezentate la capitolul de îmbunătăţire a contrastului imaginilor (pag. 37). Precizia sporită a operatorilor laplacieni conduce însă la o sensibilitate crescută în prezenţa zgomotelor (mai mare decât a operatorilor de gradient). Mai mult, laplacianul nu mai conţine informaţie relativă la direcţia tranziţiei.

8.2.2 Alte metode

O clasă importantă de operaţii neliniare de extragere a contururilor sunt cele bazate pe morfologia matematică. În secţiunea 6.2.3 am prezentat operatori morfologici de extragere a contururilor. Principiul acestora este de a măsura diferenţele dintre valorile extreme (minim şi maxim) ale vecinătăţii punctului curent; dacă diferenţa dintre aceste valori este suficient de mare înseamnă că punctul curent este un punct de contur, aflându-se într-o zonă de tranziţie a valorilor pixelilor. Variante ale acestei tehnici de bază se pot obţine prin considerarea a mai multe elemente structurante, având diferite forme şi dimensiuni.

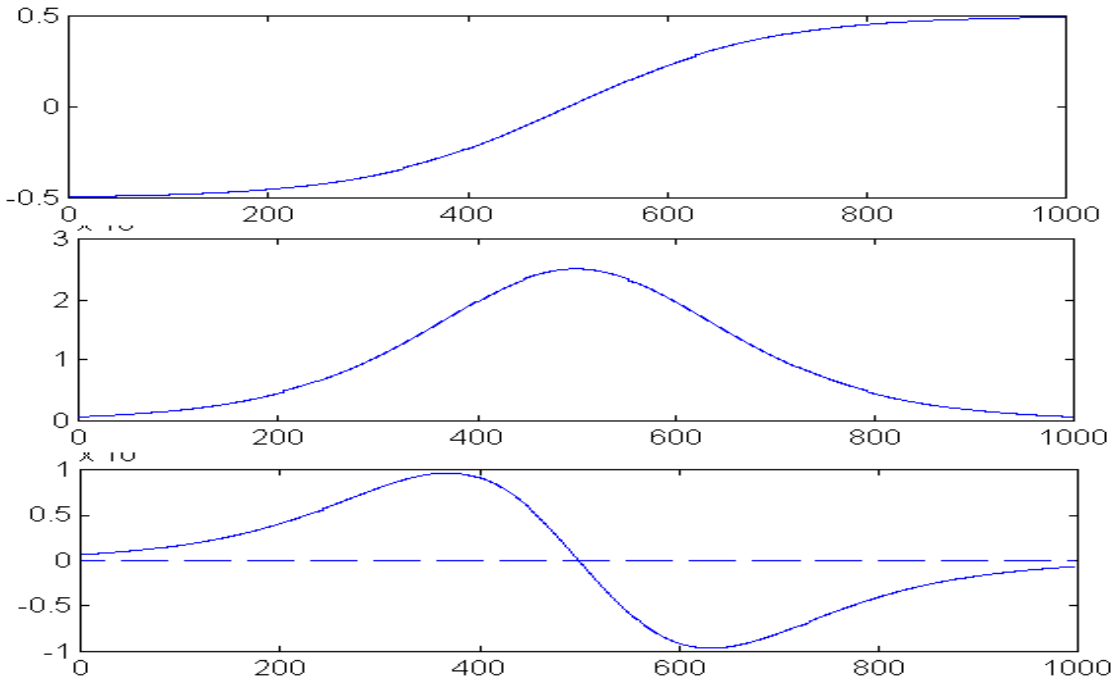


Fig. 8.10: Profil de tranzitie graduală; maximul primei derivate nu poate marca cu precizie centrul tranziţiei; derivata secundă trece prin zero la mijlocul conturului.

Capitolul 9

PARAMETRI DE FORMĂ

Prin parametri de formă înţelegem în general orice scalar sau funcţie (cu suport unidimensional sau bidimensional) asociate unei forme plane pe care o caracterizează; forme asemănătoare sunt caracterizate de parametri de formă de valori apropiate; formele diferite prezintă diferenţe mari între parametrii de formă ce le sunt asociaţi. Parametrii de formă compun un fel de fişă de identitate a formei respective, pe baza cărei această formă poate fi recunoscută în mod unic. În mod ideal, aceşti parametri trebuie să fie invarianţi la translaţie, rotaţie şi scalare. Tehnicile de recunoaştere a formelor sau de clasificare sunt precedate întotdeauna de o etapă de extragere a parametrilor de formă (sau a caracteristicilor formei).

Pentru analiza imaginilor, o formă este o funcţie de două variabile, cu suport compact $f(x, y) : K \rightarrow \mathbf{R}$; în general valorile acestei funcţii sunt binare (0 sau 1), descriind deci o parte a unei imagini binare (zona din imagine în care se află obiectul de interes). Atunci funcţia poate fi văzută ca o funcţie caracteristică a formei, asemănătoare funcţiei caracteristice a unei mulţimi.

În cele ce urmează vom prezenta câţiva parametri de formă clasici.

9.1 Parametri geometrici

Această categorie de parametri se bazează pe măsura unor atribute geometrice simple: arie (S), perimetru (P), număr de găuri, numărul lui Euler (numărul de regiuni conexe – numărul de găuri). Cum nu toate aceste numere sunt invariante şi caracteristice unic unei anume forme, au apărut combinaţii de tip raport.

Raportul de compacitate (numit şi factor de formă [19]) este raportul dintre pătratul

perimetrului şi suprafaţa formei:

$$\kappa = \frac{P^2}{4\pi S} \tag{9.1}$$

Pentru o formă circulară raportul este unitar; cu cât numărul κ este mai apropiat de această valoare, cu atât mai mult forma seamănă cu un disc (pătratul are un raport de compacitate $\kappa = 1.273$). Există însă forme diferite caracterizate de aceeaşi valoarea a parametrului dat de (9.1).

Excentricitatea sau circularitatea formei (măsura în care forma dată se deosebeşte de disc) poate fi definită şi ca un raport al razelor cercurilor circumscrise (R) şi înscrise (r) formei:

$$c = \frac{R}{r} \tag{9.2}$$

Acest raport este evident unitar în cazul discului; pentru pătrat valoarea sa este de $c = 1.412$.

9.2 Momente statistice şi invarianţi

Interpretând funcţia caracteristică a formei ca pe o funcţie de densitate de probabilitate bidimensională, putem defini momentele statistice asociate celor două variabile aleatoare (ce sunt coordonatele punctelor formei) [17]:

$$m_{pq} = \iint\limits_K f(x,y) x^p y^q dx dy,\ p,q = 0,1,2,... \tag{9.3}$$

Scalarul m_{pq} (momentul de ordin p,q sau $p+q$) este proiecţia funcţiei $f(x,y)$ pe polinoamele x^p şi y^q ale bazei complete de polinoame. Teorema reprezentării cu momente afirmă că mulţimea infinită de momente m_{pq} determină în mod unic $f(x,y)$ şi reciproc.

În cazul imaginilor binare, coordonatele sunt discrete şi funcţia este o funcţie caracteristică; formula momentelor (9.3) devine

$$m_{pq} = \sum_{f(x,y)\neq 0} \sum x^p y^q \tag{9.4}$$

Cum caracterizarea unei forme printr-o serie infinită de numere (aşa cum cere teorema reprezentării cu momente) nu este posibilă, în practică se folosesc serii de momente truncheate până la un ordin maxim fixat N ($p+q \leqslant N$). Acestea însă caracterizează o altă funcţie, $g(x,y)$, o aproximare a lui $f(x,y)$. Această aproximare este dată de o combinaţie liniară a polinoamelor bazei, ponderate cu scalarii necunoascuţi g_{pq}:

$$g(x,y) = \sum_{p+q\leqslant N} \sum g_{pq} x^p y^q \tag{9.5}$$

Găsirea acestor scalari se face prin egalarea momentelor cunoscute ale lui $f(x,y)$ cu momentele lui $g(x,y)$ dată de expresia (9.5). Rezolvând sistemul de ecuaţii cuplate ce se formează (a se vedea [9]), se pot obţine relaţiile căutate; calculul trebuie însă refăcut, din cauza cuplării ecuaţiilor, ori de câte ori se doreşte trecerea la o aproximare mai bună a formei f , mărind valoarea lui N. Această cuplare provine din cauza folosirii unei baze neortogonale (aşa cum este familia de polinoame $x^p y^q$) pentru calculul momentelor; problema a fost rezolvată prin folosirea proiecţiilor pe baza de polinoame Legendre [9].

Trebuie însă remarcat că folosirea momentelor statistice pentru caracterizarea unei forme nu asigură îndeplinirea a nici unuia dintre principiile de invarianţă căutate; de aceea au fost introduse momente statistice invariante [19], [9].

Momentele statistice invariante la translaţie sunt momentele statistice centrate:

$$\mu_{pq} = \sum_{f(x,y)\neq 0} \sum (x-\overline{x})^p (y-\overline{y})^q \tag{9.6}$$

Momentele statistice invariante la translaţie şi scalare sunt definite de:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}},\ \gamma = 1 + \frac{p+q}{2} \tag{9.7}$$

Invarianţii la translaţie, scalare şi rotaţie ai unei forme, obţinuţi în condiţiile folosirii unor momente statistice de ordin cel mult 3 ($N = 3$), sunt în număr de 7 şi sunt exprimaţi de:

$$\Phi_1 = \eta_{20} + \eta_{02} \tag{9.8}$$

$$\Phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \tag{9.9}$$

$$\Phi_3 = (\eta_{30} - 3\eta_{12})^2 + (\eta_{03} - 3\eta_{21})^2 \tag{9.10}$$

$$\Phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{03} + \eta_{21})^2 \tag{9.11}$$

$$\begin{aligned}\Phi_5 = {} & (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12}) \left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{03} + \eta_{21})^2\right] + \\ & + (\eta_{03} - 3\eta_{21})(\eta_{03} + \eta_{21}) \left[(\eta_{03} + \eta_{21})^2 - 3(\eta_{30} + \eta_{12})^2\right]\end{aligned} \tag{9.12}$$

$$\Phi_6 = (\eta_{20} - \eta_{02}) \left[(\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2\right] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{03} + \eta_{21}) \tag{9.13}$$

$$\begin{aligned}\Phi_7 = {} & (\eta_{30} - 3\eta_{21})(\eta_{03} + \eta_{21}) \left[(\eta_{03} + \eta_{21})^2 - 3(\eta_{30} + \eta_{12})^2\right] - \\ & - (\eta_{03} - 3\eta_{21})(\eta_{30} + \eta_{12}) \left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{03} + \eta_{21})^2\right]\end{aligned} \tag{9.14}$$

Iniţial (mijlocul anilor '60) aceşti invarianţi au fost folosiţi pentru recunoaşterea caracterelor mari de tipar, cu rezultate modeste. Eficienţa lor constă însă în modul rapid de calcul şi posibilitatea de a le utiliza cu succes pentru recunoaşterea formelor geometrice convexe.

Folosind momentele invariante, se mai pot deduce alte atribute: excentricitatea suprafeţei (9.15), care măsoară gradul de uniformitate al distribuţiei punctelor formei în jurul centrului de greutate şi orientarea suprafeţei, caracterizată de unghiul θ faţă de orizontală al axei faţă de care momentul inerţie al formei este minim (9.16).

$$\varepsilon = \frac{\Phi_2}{\mu_{00}} \tag{9.15}$$

$$\theta = \frac{1}{2} \arctan \frac{2\mu_{11}}{\mu_{20} - \mu_{02}} \tag{9.16}$$

9.3 Semnătura formei

Semnătura unei forme este o funcţie scalară de o variabilă, asociată unei forma plane. Semnătura este definită de distanţa de la un punct de referinţă fixat $\mathbf{x}$ (în general centrul de greutate al formei) la fiecare punct de pe conturul (frontiera) formei. Această distanţă este exprimată (sau măsurată) în funcţie de unghiul la centru θ realizat de punctul curent de pe contur cu axa orizontală de referinţă, $d_{\mathbf{x}}(\theta)$ sau în funcţie de abscisa curbilinie ρ (lungimea conturului cuprins între punctul curent şi punctul în care axa de referinţă intersectează conturul), $d_{\mathbf{x}}(\rho)$. Semnătura formei este o reprezentare reversibilă (cunoscând semnătura se poate reconstrui conturul obiectului). Figurile 9.1 şi 9.2 prezintă semnăturile unor forme poligonale.

Fig. 9.1: Semnătura unui pătrat ca funcţie de unghiul la centru; punctul de referinţă este centrul de greutate, axa de referinţă este orizontala.

Este posibil ca, pentru formele concave, semnătura în funcţie de unghiul la centru să nu poată fi calculată, deoarce, pentru anumite unghiuri θ, intersecţia dreptei de direcţie θ cu conturul să fie formată din mai multe puncte. Această problemă nu apare în cazul semnăturii în funcţie de abscisa curbilinie. Ca o restricţie generală, în cazul folosirii semnăturii bazate pe unghi, trebuie ca punctul de referinţă $\mathbf{x}$ să aparţină nucleului formei. Nucleul formei K, $Ker(K)$ este definit prin:

$$Ker(K) = \{\mathbf{x} | \forall \mathbf{y} \in K, [\mathbf{xy}] \subseteq K\} \tag{9.17}$$

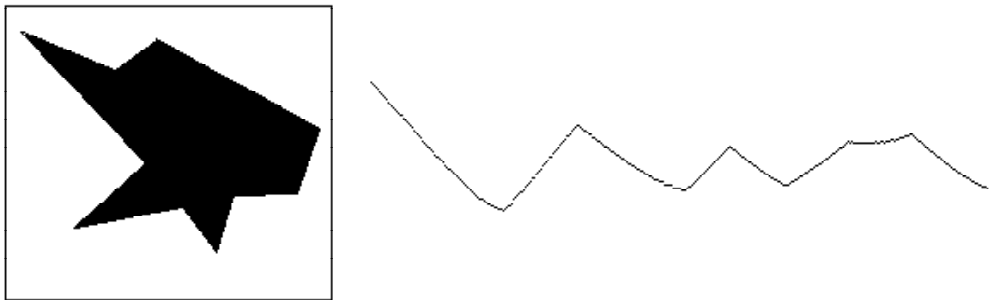


Fig. 9.2: Semnătura unei forme poligonale în funcţie de abscisa curbilinie; punctul de referinţă este centrul de greutate, axa de referinţă este orizontala.

unde $[\mathbf{xy}]$ semnifică segmentul de dreaptă definit de punctele $\mathbf{x}$ şi $\mathbf{y}$. În cazul formelor concave, nucleul formei este o mulţime vidă.

Folosind semnătura formei, se pot defini parametri de tip geometric. Raportul de simetrie este definit ca

$$Y(K) = \sup_{\mathbf{x}\in\mathbf{K}} \inf_{\theta\in[0;\pi]} \frac{d_{\mathbf{x}}(\theta)}{d_{\mathbf{x}}(\theta+\pi)} \tag{9.18}$$

Se observă că nu s-a făcut nici o presupunere privind convexitatea formei K, dar unicitatea distanţelor $d_{\mathbf{x}}(\theta)$ şi $d_{\mathbf{x}}(\theta+\pi)$ implică calcularea raportului de simetrie după punctele ce aparţin nucleului formei.

Raportul de circularitate este definit ca:

$$C(K) = \frac{\sup_\theta \left(d_{\mathbf{x}}(\theta) + d_{\mathbf{x}}(\theta+\pi)\right)}{\inf_\theta \left(d_{\mathbf{x}}(\theta) + d_{\mathbf{x}}(\theta+\pi)\right)} \tag{9.19}$$

Funcţia $h_K(\theta) = d_{\mathbf{x}}(\theta) + d_{\mathbf{x}}(\theta+\pi)$ se numeşte suportul formei, şi este diametrul formei pe direcţia θ.

În [5] s-a propus aproximarea formelor prin dezvoltarea în serie Fourier a semnăturii acestora şi truncherea reprezentării (tehnică utilizată şi într-o aplicaţie clasică de recunoaştere a conturului unor tipuri de avioane). O altă posibilă aplicaţie a semnăturii pleacă de la observaţia că, pentru o formă poligonală, în semnătură, poziţia vârfurilor este marcată de puncte unghiulare (a se urmări figurile 9.1 şi 9.2). Atunci aceasta poate fi o metodă de aproximare poligonală a unei forme oarecari (găsirea vârfurilor unui poligon ce o aproximează cât mai bine).

9.4 Skeletoane morfologice şi generalizate

Skeletonul este o reprezentare bidimensională simplificată, echivalentă, a unei forme. Pentru o formă A oarecare se defineşte discul maximal în A, de centru $\mathbf{x}$ şi rază r, $B_{\mathbf{x}}(r)$ ca fiind discul caracterizat de:

$$B_{\mathbf{x}}(r) \subseteq A \tag{9.20}$$

$$B_{\mathbf{x}}(r) \subseteq B_{\mathbf{x}'}(r') \subseteq A \Leftrightarrow \begin{cases} r = r' \\ \mathbf{x} = \mathbf{x}' \end{cases} \tag{9.21}$$

Deci discul maximal trebuie să fie inclus în formă (9.20) şi nici un alt disc inclus în formă să nu îl includă, sau, dacă îl include, să fie identic cu acesta (9.21). O formă poate avea mai multe discuri maximale.

Skeletonul unei forme este mulţimea centrelor discurilor maximale ale formei. Ca exemple simple, skeletonul unui disc este centrul său, skeletonul unui pătrat este reuniunea diagonalelor sale (figura 9.3).

Fig. 9.3: Exemple de skeletoane ale unor forme continue simple.

9.4.1 Skeletonul morfologic

Calculul skeletonului unei forme reprezentate în spaţiul discret se poate face prin formula Lantuejoul [15]; skeletonul formei A, $SK(A)$, este format din reuniunea unui număr finit de seturi skeleton:

$$SK(A) = \bigcup_{n=0}^{N_{\max}} S_n(A) \tag{9.22}$$

$$S_n(A) = (A \ominus nB) - (A \ominus nB) \circ B \tag{9.23}$$

Ordinul $N_{\max}$ corespunde momentului în care toate seturile skeleton succesive devin nule, moment marcat de $A \ominus N_{\max}B = \emptyset$; elementul structurant nB este iterarea de n ori a elementului structurant B, $nB = B \oplus ... \oplus B$ (de n ori). Elementul structurant folosit

este în general o expresie discretă a discului unitar (deci element structurant V_4 sau V_8). Figura 9.4 prezintă skeletonul unei forme discrete oarecare.

Fig. 9.4: Skeleton al unei forme discrete, reprezentat prin reuniunea seturilor skeleton şi informaţia de apartenenţă a fiecărui punct la un anumit set skeleton.

Reconstrucţia formei din skeleton se face după formula

$$A = \bigcup_{n=0}^{N_{\max}} S_n(A) \oplus nB \tag{9.24}$$

Se pot realiza şi reconstrucţii parţiale (aproximări ale mulţimii A) prin neglijarea seturilor skeleton ce reprezintă detaliile (acestea sunt seturile skeleton de ordin mic); aproximarea de ordin k a formei înseamnă deci ignorarea primelor k seturi skeleton din reconstrucţie:

$$\widetilde{A_k} = \bigcup_{n=k}^{N_{\max}} S_n(A) \oplus nB \tag{9.25}$$

Skeletonul este invariant la translaţie, nu este conex (chiar dacă forma A este conexă; a se vedea figura 9.5), nu este comutativ cu operaţia de reuniune a formelor (a se vedea figura 9.6). Transformarea skeleton este idempotentă ($SK(SK(A)) = SK(A)$) şi antiextensivă ($SK(A) \subseteq A$). Seturile skeleton sunt disjuncte două câte două ($S_i(A) \cap S_j(A) = \emptyset$, $\forall i \neq j$).

Folosirea skeletonului morfologic pentru recunoaşterea fomelor este restricţionată de puternica sa sensibilitate la zgomote (o mică schimbare a formei duce la o modificare semnificativă a skeletonului[1]), şi de variaţia la rotaţia şi scalarea obiectelor (pentru reprezentări în spaţiul discret). În acelaşi timp, folosirea elementului structurant de tip disc unitar aduce o puternică dependenţă de metrica folosită în definirea sa (să nu uităm că într-un

[1] Exemplul clasic este de a considera un disc fără centru; în acest caz skeletonul este o coroană circulară situată la jumătatea razei.

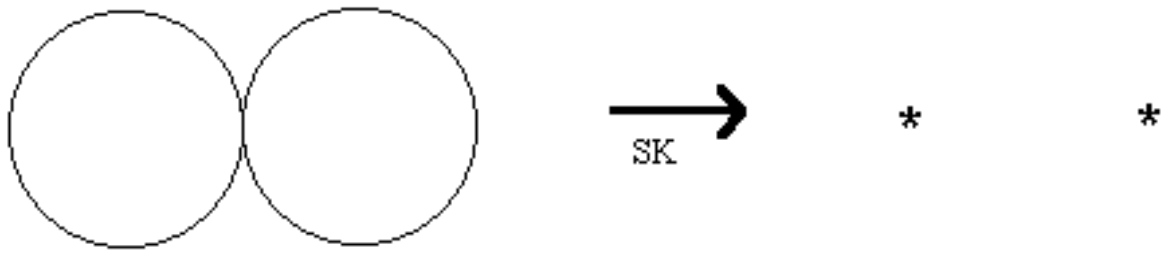


Fig. 9.5: Skeletonul nu este conex.

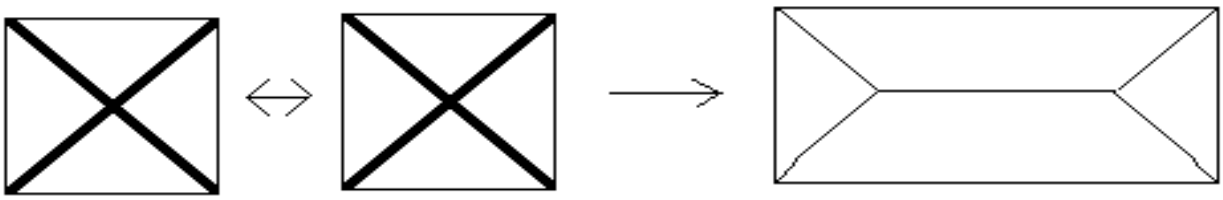


Fig. 9.6: Transformata skeleton nu este comutativă cu reuniunea.

spaţiu discret, metrica Euclidiană nu este cea mai favorabilă) şi produce elemente structurante pătrate (V_8) sau în cruce (V_4), ce pot fi cu greu interpretate ca discuri. Aceasta a dus la folosirea unei clase de elemente structurante care să nu provină din noţiunea de metrică - elementele structurante generalizate.

9.4.2 Skeletonul generalizat

Fie $\{G_i\}$ un set de mulţimi, numit set generator. Elementele structurante generalizate sunt definite recurent prin:

$$B_i = B_{i-1} \oplus G_i,\ i = 1, 2, ... \tag{9.26}$$

$$B_0 = 0_n$$

În general se folosesc seturi generatoare având o anumită periodicitate T ($G_{i+kT} = G_i,\ \forall i, k \in \mathbf{Z}$). De exemplu, figura 9.7 prezintă construcţia setului de elemente structurante generalizate rezultat dintr-un set generator cu perioadă 1 (deci compus dintr-o singură mulţime), $E_1 = \{(-1,-1), (-1,0), (0,-1), (0,0)\}$.

Fiecărui punct al formei A i se va asocia ordinul (indicele) elementului structurant generalizat maximal centrat cu originea în punctul respectiv, construind astfel o "hartă de

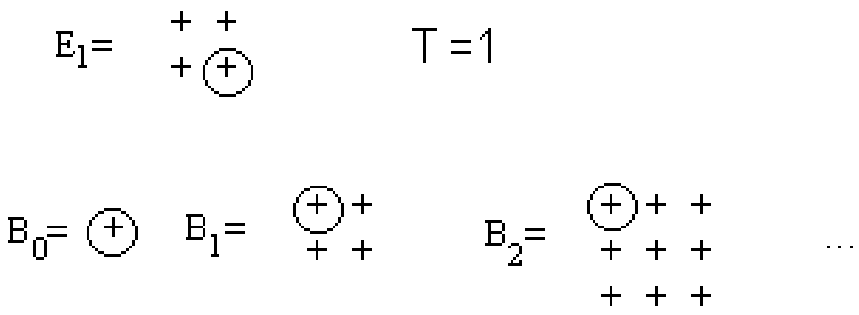


Fig. 9.7: Construcţia unui set de elemente structurante generalizate.

distanţe" (a se vedea şi figura 9.8):

$$D(\mathbf{x}) = \begin{cases} 0, \text{ dacă } \mathbf{x} \notin A \\ n, \text{ dacă } (B_{n-1})_{\mathbf{x}} \subseteq A \text{ şi } (B_n)_{\mathbf{x}} \nsubseteq A \end{cases} \tag{9.27}$$

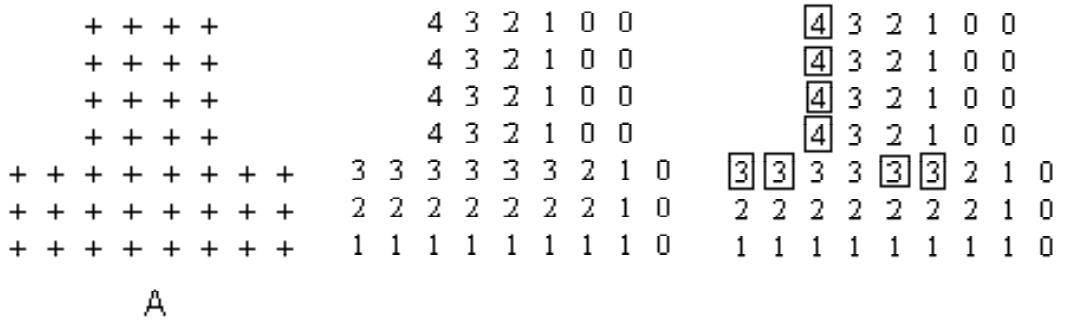


Fig. 9.8: Harta de distanţe a unui obiect construită pe baza setului de elemente structurante generalizate prezentat anterior şi skeletonul generalizat corespunzător.

Skeletonul generalizat al unei forme este mulţimea originilor elementelor structurante generalizate maximale în formă[2]:

$$GSK(A) = \left\{ \mathbf{x} \in A | \left(B_{D(\mathbf{x})-1}\right)_{\mathbf{x}} \nsubseteq \left(B_{D(\mathbf{y})-1}\right)_{\mathbf{y}}, \forall \mathbf{x} \neq \mathbf{y} \right\} \tag{9.28}$$

Folosind harta de distanţă anterioară, skeletonul generalizat obţinut este prezentat în figura 9.8.

[2]Se poate remarca similitudinea cu definiţia skeletonului, în care s-au înlocuit noţiunile: centru al discului cu origine a elementului structurant, disc cu element structurant generalizat, rază a discului cu indice (ordin) al elementului structurant generalizat.

Capitolul 10

PRINCIPII DE IMPLEMENTARE SOFTWARE ŞI HARDWARE

Principiile esenţiale legate de implementările practice ale sistemelor de prelucrarea şi analiza imaginilor urmăresc două direcţii, cu dezvoltare corelată: implementările software şi dispozitivele hardware (de accelerare). După cum am arătat în capitolul introductiv, la descrierea structurii tipice a unui sistem de prelucrarea şi analiza imaginilor, marea majoritate a implementărilor folosesc ca suport fizic pentru unitatea centrală de prelucrare un calculator obişnuit (compatibil PC); ceea ce îl particularizează este pachetul de programe rulate. Putem distinge două categorii fundamentale de astfel de programe: programe strict dependente de aplicaţie şi programe de uz general.

Un program dependent de aplicaţie realizează doar algoritmii specifici operaţiei pe care o execută sau supraveghează. Interacţiunea cu operatorul uman este minimă şi calificarea acestuia nu este necesar să o depăşească pe cea a unui tehnician [11]. Adeseori programul trebuie să fie de timp real. Aceste variante de implementare sunt potrivite pentru procese caracterizate de parametri stabili şi care se desfăşoară în condiţii ambiante (iluminare, poluare vizibilă – particule, fum) relativ constante şi nu sunt portabile (fiind în general optimizate pentru o anumită structură hardware).

Programele de uz general permit efectuarea unui mare număr de operaţii de prelucrarea şi analiza imaginilor, cu numeroşi parametri reglabili. Interacţiunea cu operatorul uman este mare şi acesta trebuie să aibă o calificare superioară. Programele de acest tip nu sunt de timp real şi în general sunt cuplate *off-line* cu instalaţiile tehnologice propriuzise, făcând parte mai ales din dotarea laboratoarelor de cercetare şi de analiza calităţii şi conformităţii. Interfaţa utilizator este cea care crează diferenţa între două categorii de programe, în ceea ce priveşte modalitatea în care operatorul specifică succesiunea de operaţii de executat. Din acest punct de vedere vom face distincţia între *menu-driven* şi *flow-chart driven* (deci programe controlate prin meniu sau prin graf de flux).

Implementările de tip *menu-driven* aplică câte un unic pas de prelucrare asupra imaginii din fereastra activă; rezultatul va fi prezentat într-o nouă fereastră de afişare. Tipurile de operaţii se aleg din meniuri sau bara de butoane. Asemenea soluţii corespund majorităţii sistemelor software comerciale de grafică şi *imaging* (de tipul Adobe Photoshop, Corel Draw, Paint Shop Pro). Operaţiile realizate sunt orientate mai ales către aspectul grafic (publicistic) al imaginilor, referindu-se în special la operaţii de filtrare, modificare a contrastului, pseudocolorare, decupare de regiuni. Programele cu comandă *menu-driven*, cu rezultatele intermediare parcurgând etapele operaţiilor din fereastră în fereastră, sunt direct derivate din proiectarea obiectelor de interfaţă în sistemele de tip Windows.

Implementările de tip *flow-chart* permit construirea grafică interactivă a unui lanţ de operaţii aplicate unei imagini iniţiale. Fiecare operaţie este un bloc funcţional caracterizat de intrări, ieşiri şi parametri de control; blocurile funcţionale sunt interconectate, implementând fluxul de operaţii pe care îl va parcurge imaginea. Procesul de comandă înseamnă selectarea imaginii iniţiale şi acţionarea unui buton de start. Asemenea programe sunt tipice sistemelor de calcul mari (staţii de lucru), pentru care produsul Khoros este un standard de fapt; un produs similar pentru PC este programul AdOculos (figura 10.1).

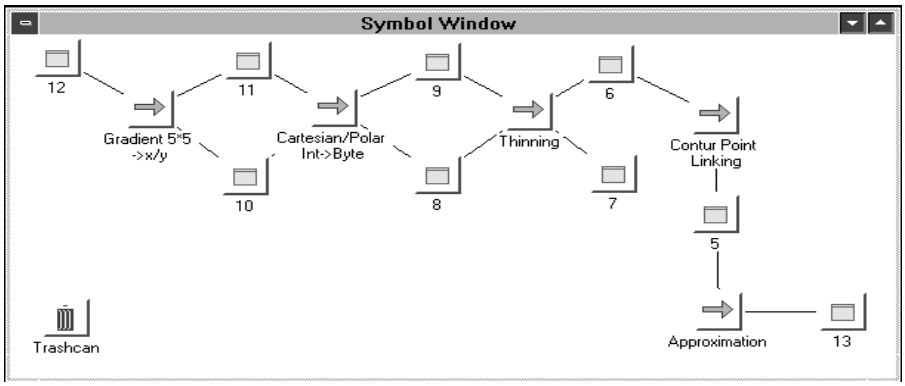


Fig. 10.1: Specificarea fluxului de operaţii în fereastra de comandă a programului AdOculos.

Ca o categorie intermediară între comenzile meniu şi diagrama de flux, putem considera programele cu comandă de tip macro (sau *batch*); programul (sau limbajul) Matlab poate fi încadrat într-o asemenea categorie.

Multe dintre operţiile iniţial realizate prin software-ul de aplicaţie au migrat către nivelul hardware. Exemplul cel mai tipic de asemenea comportament este dat de rezolvare problemei de dithering (aproximarea culorilor reale dintr-o imagine. folosind un număr mic de culori disponibile la dispozitivul de afişaj). După ce, iniţial, problema era rezolvată

de programele aplicaţie, gestiunea culorilor a trecut în grija sistemului de operare, pentru ca, recent, să apară plăcile grafice inteligente. Aceasta este de altfel una dintre tendinţele actuale: evoluţia accesoriilor (plăci specializate, dispozitive de achiziţie) inteligente, deci cu facilităţi de prelucrare integrate. Senzorii de imagine CCD sunt una dintre ţintele predilecte ale acestei dezvoltări, înglobând cipuri de compresie şi transmisie a imaginilor, comandă de urmărire automată a ţintei, operaţii de recunoaştere.

Deşi programarea orientată pe obiecte este un concept destul de nou, evoluţia sa a fost deosebit de spectaculoasă. Probabil că principalul atu al acestei evoluţii a fost apariţia sistemului de operare Windows (cu toate derivatele sale 3.1, 3.11, '95, '98, etc.) a cărui structură este în întregime bazată pe conceptul de obiecte. Programarea unei aplicaţii Windows va face în mod clar apel la structuri predefinite - clasele de bază aplicaţie, fereastră, buton, ş.a.m.d. Atunci devine destul de normal să fie proiectate şi datele specifice ale aplicaţiei particulare tot ca structuri de tip obiecte.

Alegerea unui limbaj specific de programare nu mai este totuşi o problemă esenţială; majoritatea limbajelor de uz general ingineresc (tehnic) provin din trunchiul comun C / Pascal. Desigur că există nenumărate variante ale acestor limbaje de bază (C++ Builder, Visual C++, Delphi, Borland C++ şi Borland Pascal cu obiecte) şi chiar apar limbaje noi (Java, care, deşi se revendică ca o aplicaţie pur distribuită pentru folosirea reţelelor şi în special a Internetului, nu este cu mult diferită ca structură de un C cu clase). Principala evoluţie a mediilor de programare nu a fost însă legată de modificarea limbajului în sine (şi deci a caracteristicilor de compilare a codului) ci modificarea stilului în care se crează aplicaţia specifică, şi mai precis, aspectul de interfaţă.

Proiectarea unei interfeţe pentru Windows înseamnă definirea unor acţiuni asociate evenimentelor produse în sistem (clicuri de mouse, apăsări de taste) şi construirea elementelor grafice specifice (ferestre de afişare şi de dialog, meniuri, introducere date, etc.). Acestă activitate a fost simplificată la maximum prin apariţia constructoarelor vizuale de aplicaţii, în care interfaţa grafică este construită prin alipirea unor elemente de bază (butoane, ferestre, zone de text) pe cadre fixe (fereastră), numai cu mouse-ul (prin tehnică *drag and drop*). Compilatorul generează automat codul sursă aferent construcţiei; tot ceea ce trebuie să facă programatorul este să lege sursa ce descrie aplicaţia şi acţiunea specifică la codul care interpretează evenimentul de apel.

Ceea ce trebuie subliniat (spre a evita anumite interpretări uzuale, dar eronate) este că aceste medii de programare avansate (gen Builder, Visual, etc..) nu preiau şi atribuţia de a scrie algoritmii specifici; uşurinţa construcţiei vizuale este legată strict de construcţia interfeţei aplicaţiei, deci aspect grafic şi eventual partea de preluare a datelor. Programatorul va dezvolta restul aplicaţiei ca pentru un compilator clasic.

Concluzia ce rezultă din această scurtă discuţie este aceea că, pentru orice aplicaţie, *trebuie separată partea de interfaţă de partea de calcul specific.* În cazul unui program de prelucrarea şi analiza imaginilor acesta însemnă că trebuie făcută o proiectare la nivelul structurii de date (matricea ce conţine valorile pixelilor din imagine şi prelucrările specifice

asupra acestora) şi o proiectare la nivelul interfeţei (care să specifice cum se va face afişarea imaginilor pe ecran, cum vor fi scrise pe disc, cum se va face un eventual transfer dinamic al datelor de la sau către alte aplicaţii). Majoritatea covârşitoare a sarcinilor legate de interfaţă pot fi rezolvate fără a cunoaşte nimic despre operaţiile specifice prelucrării imaginilor şi despre modul în care o imagine este reprezentată în memoria de lucru a calculatorului. Să considerăm de exemplu problema afişării unei imagini într-o fereastră. Fără îndoială că cel mai simplu mod de afişare este folosirea imaginii ca un *canvas*, înglobat în obiectul de tip fereastră, şi de a cărui afişare se ocupă sistemul Windows. Această abordare exclude însă accesul la conţinutul imaginii; aceasta trebuie deci separată de fereastra de afişare. Este poate deci preferabilă memorarea imaginii ca o matrice şi transformarea acesteia într-o structură *bitmap* ataşată ferestrei, la fiecare cerere de reafişare a acesteia.

Cuvântul de ordine actual în implementările software este orientarea-obiect (limbajele C++, Delphi, Java), insistând mai ales pe aspectele de polimorfism şi moştenire pe care le aduce acest stil de programare. Moştenirea corespunde cazului unor imagini cu structură din ce în ce mai elaborată, pentru care se pot face tot mai multe operaţii (de tip analiză de formă şi clasificare, de exemplu). Polimorfismul corespunde realizării unor operaţii cu nume (sau metode de implementare) identice care să aibă comportări diferite, în funcţie de tipul datelor cărora li se aplică (funcţia de histogramă poate produce, de exemplu, funcţia de densitate de probabilitate a câmpului aleator imagine, în cazul imaginilor cu nivele de gri, aria obiectelor, în cazul imaginilor binare şi numărul de componente conexe, în cazul imaginilor binare etichetate). De asemenea, aspectul legat de refolosirea codului nu este neglijabil (să nu uităm, că, de exemplu, pentru filtrarea în domeniul spaţial, un numitor comun al diverselor tipuri de filtre este tehnica de tip fereastră glisantă).

Tehnologia Java, aduce, pa lângă orientarea obiect, ideea de folosire a resurselor distribuite (fie pe Internet, fie ca resurse de multiprocesare). Astfel a devenit posibilă crearea de depozite de software specializat, universal executabil (prin mecanismul de *applet*) pe orice maşină pe care este disponibil un *browser* Internet. Marile avantaje anunţate pentru Java (cod portabil, execuţie paralelă, protecţie la manipularea defectuoasă a zonelor de memorie, protecţia datelor) sunt contrabalansate de codul executabil relativ lent şi de dimensiunile mari ale codului sursă.

Java este însă interesantă pentru prelucrarea şi analiza imaginilor prin perspectiva deschisă de *multi-threding* – existenţa şi gestionarea de către o aplicaţie oarecare a mai multe fire de execuţie ce rulează concomitent din punct de vedere logic (sau aproape concomitent din punctul de vedere fizic al calculatorului cu unic procesor); astfel devine posibilă rularea paralelă de aplicaţii pe sisteme cu unic procesor, fără sisteme de operare *multi-tasking*. De altfel, puţine sisteme fizice multiprocesor au fost realizate pentru prelucrarea de imagini; sistemele de prelucrarea şi analiza imaginilor au utilizat în general maşini cu paralelism masiv (SIMD/ SPMD, *transputer farm*) sau grupuri de calculatoare (*cluster processing*).

Exploatarea paralelismului este un merit pe care prelucrarea imaginilor şi l-a arogat încă de la începuturi; operaţiile tipice de prelucrarea imaginilor sunt operaţii relativ simple, cu un număr mare de instanţe (pentru fiecare pixel al imaginii se face ceva), folosind date puţin redundante (suprapunerea între poziţiile alăturate ale ferestrelor de filtrare este în general mică) şi, uneori, informaţie globală (cazul histogramei sau a filtrărilor în domeniul de frecvenţă). Se pot distinge astfel două nivele de paralelism: un paralelism masiv, intrinsec structurii imaginii, legat de nivelul pixel, şi un paralelism ascuns, specific operaţiilor de prelucrare (ca de exemplu realizarea operaţiilor morfologice prin (6.3) şi (6.6) sau implementarea paralelă a FFT).

În concluzie, putem afirma că prelucrarea şi analiza imaginilor a reuşit să câştige teren ca urmare a realizărilor spectaculoase ale tehnologiei electronice şi informaticii. Creşterea continuă a puterii de calcul disponibile în unităţile de prelucrare ale calculatoarelor va transforma probabil în viitorul apropiat prelucrarea şi analiza imaginilor dintr-o anexă nebuloasă şi exotică a aplicaţiilor speciale într-o soluţie fiabilă de larg consum industrial.

Bibliografie

[1] Buzuloiu, V.: *Prelucrarea imaginilor: note de curs*, Universitatea "Politehnica" Bucureşti, 1998

[2] Castleman, K. R.: *Digital Image Processing*, Prentice Hall, Englewood Cliffs, NJ, 1996

[3] Cocquerez, J. P., Philipp, S. (coord.): *Analyse d'images: filtrage et segmentation*, Masson, Paris, 1995

[4] Dougherty E. R., Giardina, C. R.: *Image Processing - Continous to Discrete*, vol. 1, *Geometric, Transform and Statistical Methods*, Prentice Hall Inc., Englewood Cliffs, 1987

[5] Gonzales, R. C., Woods, R. E.: *Digital Image Processing*, Addison Wesley, Reading MA, 1992

[6] Haralick, R. M., Shapiro, L. G.: *"Glossary of Computer Vision Terms"*, în Pattern Recognition, vol. 24, no. 1, pag. 69-93, 1991

[7] Haralick, R. M., Sternberg, S. R., Zhuang, X.: "*Image Analysis using Mathematical Morphology*", în IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 9, no. 4, Iulie 1987, pag. 532-549

[8] Jähne, B.: *Practical Handbook on Image Processing for Scientific Applications*, CRC Press, 1997

[9] Jain, A. K.: *Fundamentals of Digital Image Processing*, Prentice Hall, Englewood Cliffs NJ, 1989

[10] Ministerium für Wirtschaft, Mittelstand und Technologie des Landes Nordrhein-Westfalen: *Stand und Trends der Bidverarbeitung in NRW*, Düsseldorf, 1995

[11] Ministerium für Wirtschaft, Mittelstand und Technologie des Landes Nordrhein-Westfalen: *Produkte und Dienstleitungen für die Bildverarbeitung. Stand und Trends*, Düsseldorf, 1996

[12] Pitas, I., Venetsanopoulos, A. N.: *Nonlinear Digital Filters – Principles and Applications*, Kluwer Academic Publ., Norwell MA, 1990

[13] Press, W. H., Flannery, B. P., Teukolsky, W. T., Vetterling, W. T.: *Numerical Recipes in C. The art of scientific computing*, Cambridge University Press, 1988

[14] Serra, J.: *Image Analysis and Mathematical Morphology*, Academic Press, London, 1982

[15] Schmitt, M., Mattioli, J.: "*Reconnaissance de formes planaires par morphologie mathematique et reseaux de neurones*", în Revue Technique Thomson CSF, vol. 22, no. 4, Decembrie 1990, pag. 573-609, Ed. Gauthiers-Villars: Paris

[16] Spătaru, A.: *Teoria Transmisiunii Informaţiei*, Ed. Didactică şi Pedagogică, Bucureşti, 1984

[17] Vertan, C., Gavăt, I., Stoian, R.: *Variabile aleatoare: principii şi aplicaţii*, Editura Printech, Bucureşti, 1999

[18] Zamperoni, P.: "*Image Enhancement*", Advances in Imaging and Electron Physics, vol. 92, pp. 1-77, Academic Press, 1995

[19] Wahl, F. M.: *Digital Image Signal Processing*, Artech House, Boston, 1987