

# PRELUCRAREA ȘI ANALIZA IMAGINILOR

Constantin VERTAN

{1999}

# Cuprins

<b>1</b>	<b>INTRODUCERE</b>	<b>6</b>
1.1	Imagini digitale . . . . .	7
1.2	Structura unui sistem de prelucrarea și analiza imaginilor . . . . .	9
1.3	Stocarea imaginilor . . . . .	12
1.3.1	Stocarea imaginilor în memorie . . . . .	13
1.3.2	Stocarea imaginilor în fișiere . . . . .	14
<b>2</b>	<b>TEHNICI DE ÎMBUNĂTĂȚIRE A IMAGINILOR</b>	<b>17</b>
2.1	Operații punctuale de modificare a contrastului . . . . .	18
2.1.1	Modificarea liniară a contrastului . . . . .	19
2.1.2	Modificarea neliniară a contrastului . . . . .	23
2.2	Pseudocolorarea . . . . .	25
2.3	Operații de contrastare bazate pe histograma imaginii . . . . .	26
<b>3</b>	<b>FILTRAREA LINIARĂ A IMAGINILOR</b>	<b>31</b>
3.1	Filtrarea liniară de netezire . . . . .	34
3.2	Filtrarea liniară de contrastare . . . . .	36
3.3	Filtrarea liniară adaptivă . . . . .	38
<b>4</b>	<b>TRANSFORMĂRI INTEGRALE UNITARE DISCRETE</b>	<b>42</b>

4.1	Generalități . . . . .	42
4.2	Proprietățile transformatelor unitare unidimensionale . . . . .	44
4.3	Transformata Fourier discretă . . . . .	45
4.3.1	Proprietățile fundamentale ale transformatei Fourier . . . . .	46
4.3.2	Transformata Fourier rapidă . . . . .	49
4.4	Alte transformări . . . . .	53
4.4.1	Transformata cosinus . . . . .	53
4.4.2	Transformata sinus . . . . .	54
<b>5</b>	<b>FILTRAREA NELINIARĂ A IMAGINILOR</b>	<b>56</b>
5.1	Filtrarea de ordine . . . . .	57
5.1.1	Filtrul median . . . . .	59
5.1.2	Filtrele de ordine ponderate și structurile multietaj . . . . .	62
5.2	Filtre de ordine de domeniu . . . . .	65
5.3	L-filtre . . . . .	65
5.4	Aspecte de implementare . . . . .	66
<b>6</b>	<b>ELEMENTE DE MORFOLOGIE MATEMATICĂ</b>	<b>71</b>
6.1	Transformări morfologice de bază . . . . .	72
6.1.1	Transformarea Hit or Miss . . . . .	72
6.1.2	Erodarea morfologică . . . . .	74
6.1.3	Dilatarea morfologică . . . . .	75
6.1.4	Proprietățile erodării și dilatării . . . . .	77
6.1.5	Aspecte de implementare . . . . .	84
6.2	Transformări morfologice derivate . . . . .	85
6.2.1	Deschiderea și închiderea . . . . .	85

6.2.2	Filtrele alternate secvențial . . . . .	88
6.2.3	Operatori morfologici de contur . . . . .	88
6.3	Extinderea morfologiei matematice la nivele de gri . . . . .	89
<b>7</b>	<b>METODE DE COMPRESIE A IMAGINILOR</b>	<b>92</b>
7.1	Compresia imaginilor binare . . . . .	93
7.1.1	Codarea entropică (Huffman) . . . . .	93
7.1.2	Codarea pe flux de biți . . . . .	95
7.2	Compresia imaginilor cu nivele de gri . . . . .	99
7.2.1	Codarea predictivă . . . . .	100
7.2.2	Compresia imaginilor cu transformate . . . . .	101
7.2.3	Codarea cu arbori cuaternari . . . . .	102
7.2.4	Cuantizarea vectorială . . . . .	105
<b>8</b>	<b>SEGMENTAREA IMAGINILOR</b>	<b>110</b>
8.1	Segmentarea orientată pe regiuni . . . . .	111
8.1.1	Segmentarea bazată pe histogramă . . . . .	111
8.1.2	Creșterea și fuziunea regiunilor . . . . .	119
8.2	Segmentarea orientată pe contururi . . . . .	123
8.2.1	Metode derivative . . . . .	123
8.2.2	Alte metode . . . . .	128
<b>9</b>	<b>PARAMETRI DE FORMĂ</b>	<b>130</b>
9.1	Parametri geometrici . . . . .	130
9.2	Momente statistice și invarianți . . . . .	131
9.3	Semnătura formei . . . . .	133

9.4	Skeletoane morfologice și generalizate . . . . .	135
9.4.1	Skeletonul morfologic . . . . .	135
9.4.2	Skeletonul generalizat . . . . .	137
<b>10</b>	<b>PRINCIPII DE IMPLEMENTARE SOFTWARE ȘI HARDWARE</b>	<b>139</b>

# Capitolul 1

## INTRODUCERE

Prelucrarea și analiza imaginilor (numită adeseori prescurtat doar prelucrarea imaginilor) s-a născut datorită ideii și necesității de a înlocui observatorul uman printr-o mașină. Este important de precizat că analiza imaginilor a mers mai departe decât simpla înlocuire a observatorului uman, deoarece au apărut soluții 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 mașina totală de viziune, capabilă să realizeze funcțiile vizuale ale oricărei viețuitoare (desigur, după realizarea a importante dezvoltări teoretice și 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 înțelege prelucrarea pe un calculator digital a unor date bidimensionale (imagini). Termenul cheie este cuvântul *digital*, înlocuit adesea în mod eronat în multe traduceri românești cu termenul de numeric. După cum o arată dicționarul limbii române moderne, definiția cuvântului *numeric* este aceea de

“care aparține numerelor, privitor la numere, exprimat prin numere”.

Rezultatul oricărui calcul este numeric. Termenul *digital* înseamnă însă

“ceea ce este referitor la reprezentarea informației discrete în calculatoare”

Deci atâta vreme cât acceptăm ideea că unealta de lucru în prelucrarea imaginilor este calculatorul, și acesta la rândul său este digital, atunci și prelucrarea este la rândul ei digitală, ca un caz particular al oricărei prelucrări numerice. Desigur că există însă și prelucrări de imagini care sunt analogice - așa cum sunt toate prelucrările ce au loc în cadrul lanțului de transmisie și recepție a imaginii standard de televiziune.

## 1.1 Imagini digitale

La început, imaginile sunt semnale, dar nu funcții temporale, ci funcții definite pe un domeniu spațial. Orice imagine este o structură bidimensională (tablou, matrice) de date. Un element al imaginii se numește *pixel* (cuvânt preluat din engleză, unde provine de la **p**icture **e**lement). Aceste date pot fi numere naturale, reale sau complexe, reprezentate însă pe un număr finit de biți. După tipul datelor din această structură bidimensională, imaginile prelucrate pot fi împărțite 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 conținut binar al imaginii, în general alb-negru) și imaginile cu nivele de gri (de tipul imaginii de luminanță 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 constituenți 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 frecvență, așa cum sunt imaginile de teledetecție ale sateliților, imaginile de termidetecție în benzile de infraroșu,...). Tot în categoria imaginilor vectoriale intră însă și imaginile stereo (o pereche de imagini ale aceleiași scene, luate din unghiuri diferite) și secvențele de imagini.

Conform datelor prezentate în [11], dintre imaginile prelucrate în aplicații funcționale, 20 % sunt alb-negru, 32 % sunt cu nivele de gri, 20 % sunt color, 10 % sunt imagini stereoscopice și 18 % sunt secvențe de imagini.

În mod clasic, valoarea unui element al unei imaginii este o măsură a intensității 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 și imagini vizibile [2]. Imaginile abstracte sau modelele sunt de fapt funcții [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 achiziții 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 distribuții de intensitate luminoasă (așa ca hologramele, imaginile de interferență și difracție) sau imagini propriu-zise (de luminanță - în sensul curent al termenului, ce se referă la fotografii, desene, picturi, schițe, scheme și altele din aceeași categorie).

O altă împărțire a imaginilor scalare se poate face după semnificația ce se dă valorii numerice a pixelilor. Vom distinge astfel imagini de intensitate și imagini indexate. O imagine de intensitate este o imagine în care valoarea fiecărui pixel este o măsură directă a intensității 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 regăsește informația de culoare asociată pixelului respectiv. Deci, pentru afișarea sau reprezentarea unei imagini indexate este necesară o informație suplimentară, de asociere între indici și culori. Această asociere se face prin intermediul tabelii de culoare. Tabela de culoare este o matrice în care fiecare linie conține descrierea unei culori (deci cele trei componente ce definesc culoarea - în mod tipic intensitățile relative de roșu, verde și albastru ce compun culoarea dată printr-un amestec aditiv). Deci tabela de culoare are trei coloane; numărul de linii al tabelii de culoare este egal cu numărul de culori din imaginea reprezentată și 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 găsește 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 și în grafica pe calculator. Afișarea imaginilor pe ecranul monitorului se face corespunzător unui anumit mod grafic, determinat din placa video a calculatorului. Un mod video definește numărul maxim de culori ce pot fi utilizate simultan și dimensiunile ecranului (în pixeli de afișaj). Culorile utilizate la un moment dat sunt grupate într-o paletă de culori de afișare. Paleta de afișare este o structură logică definită în BGI<sup>1</sup> (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);
```

---

<sup>1</sup>Exemplele de cod C prezentate în lucrare corespund mediilor integrate Borland (Borland C++ 3.1, Turbo C 2.0)



Afișarea unui pixel cu o anumită culoare se face cu:

```
putpixel(int pozx, int pozy, int index_culoare);
```

Sub Windows, este suportată și specificarea directă a culorii de afișat (sub forma unui triplet RGB<sup>2</sup>), sistemul de operare aproximând culoarea respectivă cu cea mai apropiată culoare disponibilă din paleta de lucru curentă (în acest fel, utilizatorul poate neglija existența acesteia).

Pentru o imagine cu nivele de gri, componentele de roșu, verde și albastru ale fiecărei culori din paleta de culoare sunt identice. Dacă specificarea componentelor de culoare se face prin numere de 8 biți (deci între 0 și 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 indecși între 0 și 255, alocați conform convenției 0 - negru, 255 - alb. În acest fel, pentru o imagine indexată cu nivele de gri, nu mai este necesară specificarea tabelii de culoare; culorii reprezentate de indexul  $i$  îi corespunde nivelul de gri  $i$ , adică tripletul RGB  $(i, i, i)$ .

Modelul imaginii indexate este un caz particular de folosire a tehnicii dicționar (sau tehnicii tabelului de echivalență - *Look Up Table* - LUT): o tehnică de regăsire a unei cantități de informație folosind asocierea unei chei de căutare mult mai mici.

## 1.2 Structura unui sistem de prelucrarea și analiza imaginilor

Structura tipică a unui sistem de prelucrarea (evident digitală) și analiza imaginilor este alcătuită din punct de vedere funcțional 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 radiația electromagnetică a obiectului studiat pentru a forma imaginea trăsăturilor de interes
- convertorul de radiație: convertește radiația electromagnetică din planul imaginii într-un semnal electric.

Sistemul de formare a imaginii și convertorul de radiație formează senzorul; acesta realizează o proiecție plană (bidimensională) a scenei reale (care este în general tridimensională). Un studiu realizat în Germania în anul 1996 [11] prin inventarierea sistemelor de

---

<sup>2</sup>Red Green Blue - Roșu, Verde, Albastru: sistemul primar de reprezentare a culorilor.

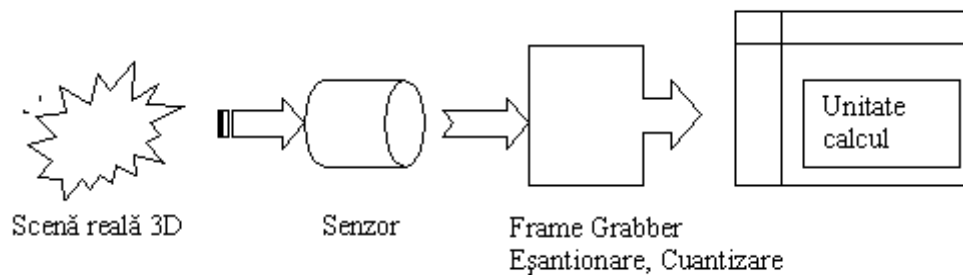


Fig. 1.1: Schema generală a unui sistem de analiza și prelucrarea imaginilor

preluare a imaginilor folosite în industrie indică o distribuție a tipurilor de senzori după gama de radiație captată conform tabelului 1.1:

Domeniu electromagnetic	Infraroșu depărtat	Infraroșu apropiat	Vizibil	Ultraviolet	Radar (microunde)	Radiație X
Procent	5 %	25 %	40 %	10 %	10 %	10 %

Tabel 1.1: Tipuri de senzori folosiți în prelucrarea imaginilor

- sistemul de achiziție (echivalent unui *frame-grabber* sau *video-blaster*): convertește semnalul electric al senzorului într-o imagine digitală, pe care o stochează; acesta nu este altceva decât un dispozitiv de eșantionare (discretizare a suportului imaginii) și cuantizare (discretizare a domeniului de valori a imaginii).
- sistemul de prelucrare (în mod tipic un calculator, fie el PC sau stație de lucru); în această categorie se încadrează însă și mașinile specializate de calcul, calculatoarele de proces, ...
- software-ul specializat: implementează algoritmi de prelucrare și analiză

În [11] se arată că unitatea de prelucrarea hardware (deci calculatorul) folosită la aplicațiile de prelucrarea imaginilor funcționale la acea dată este în cea mai mare majoritate a cazurilor un PC obișnuit, cu performanțe standard; datele sunt sintetizate în tabelul 1.2:

Platformă hardware	Procent din piață
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 %
Stații de lucru (workstations)	10 %
Mașini specializate	10 %
Calculatoare Macintosh, calculatoare paralele (transputere), altele	10 %

Tabel 1.2: Unități 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 și nu neapărat prezente împreună: îmbunătățirea, restaurarea, compresia, segmentarea și analiza [9].

Blocul de îmbunătățire a imaginilor are ca scop accentuarea anumitor trăsături [ale obiectelor conținute în imagine] pentru ușurarea unor sarcini ulterioare de analiză automată sau interpretare prin afișare. Asemenea metode pot fi utile la extragerea trăsăturilor caracteristice ale obiectelor, eliminarea zgomotelor suprapuse imaginii, mărirea confortului vizual. Acești algoritmi nu măresc conținutul informațional al imaginii și sunt în general interactivi și puternic dependenți de aplicație.

Restaurarea imaginilor se referă la eliminarea sau minimizarea efectelor unor perturbații și a unor degradări. Perturbațiile reprezintă în general zgomotele (modelate ca procese aleatoare) ce se suprapun în cursul achiziției imaginii (din cauza senzorului și a lanțului de transmisiune și captare); degradările sunt cauzate de imperfecțiunile și limitările deterministe ale senzorului (efecte de apertură, timp de expunere, deficiențe geometrice ale sistemului de lentile, ...).

Compresia imaginilor se referă la reducerea volumului de date (numărului de biți) cu care este reprezentată imaginea, printr-o transformare reversibilă - imaginea trebuie să poată să fie recuperată integral (sau cu diferențe 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 algoritmi 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 și analiza imaginilor este deosebit de largă; practic, în orice domeniu de activitate se pot găsi numeroase aplicații. Această clasă de aplicații extrem de specifice a fost caracterizată drept “consumul imaginii” [1] (imaginea folosită în vederea analizei, deci a luării unor decizii). Imaginile preluate de către sateliți pot fi folosite la descoperirea resurselor terestre, cartografiere geografică, predicția recoltelor, urmărirea dezvoltării urbane, urmărirea vremii, controlul și prevenirea incendi-

ilor și inundațiilor, precum și alte aplicații ecologice și militare. Aplicațiile transmisiei și compresiei imaginilor se regăsesc în televiziunea digitală, sistemele de teleconferință, transmisiile fax, birotică, comunicația pe rețele distribuite, sisteme de securitate cu circuit închis, aplicații militare. În aplicațiile medicale sunt prelucrate radiografiile cu raze X, angiogramele, echografiile, tomografiile, imaginile de rezonanță magnetică nucleară. Acestea pot fi utilizate pentru monitorizarea pacienților și pentru descoperirea și identificarea de boli și tumori.

O largă clasă de aplicații sunt cele industriale, în care componentele de prelucrarea și analiza imaginilor sunt folosite în sisteme mai mari de asigurare a calității produselor (metrologie, controlul calității - inclusiv defectoscopie nedistructivă). Soluțiile sunt extrem de specifice, puternic legate de procesul de fabricație urmărit și tind să devină din ce în ce mai utilizate odată cu impunerea normelor de “calitate totală” ale standardului ISO9000 (se poate urmări [10] pentru aplicații specifice ale diferitelor firme germane). Din acest punct de vedere este interesantă comparația între câteva caracteristici ale sistemului vizual și de prelucrare uman și un sistem de prelucrare și analiza imaginilor [8], folosite pentru aplicații 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
Rezistență la oboseală	MICĂ	MARE
Iluzie optică	DA	NU
Prelucrare statistică	Greu realizabil	DA
Reproductibilitate	Greu realizabil	DA
Măsurare geometrică	Cu instrumente auxiliare	DA
Recunoaștere de forme	DA	DA

Tabel 1.3: Comparația între caracteristici esențiale ale sistemului vizual uman și sistemele de prelucrarea și analiza imaginilor

## 1.3 Stocarea imaginilor

Se poate considera că există două moduri de stocare a imaginilor: stocarea în memoria de lucru a unității de prelucrare a imaginii de lucru (care este o stocare de scurtă durată - doar pe durata prelucrării efective) și stocarea de lungă durată imaginilor, în fișiere, pe suporturi externe de memorie (benzi, discuri, etc.). Diferența esențială 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 informația fiecărui pixel.

### 1.3.1 Stocarea imaginilor în memorie

Principalul limbaj de programare utilizat pentru aplicații 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, și deci este necesară o declarare dinamică. Particularitatea este dată de memorarea separată a fiecărei linii (sau coloane) a matricii într-un vector alocat dinamic, și gruparea adreselor de început a acestora într-un vector de pointeri, la care se va reține adresa de început (deci un alt pointer). Dacă considerăm un tip generic de date pentru componentele matricii (caracter, sau întreg, sau real), atunci o secvență C de declarare a unui imaginii poate fi:

```
tip ** imagine;
unsigned int contor;
imagine=(tip**) malloc(nr_linii*sizeof(typ*));
for (contor=0;contor<nr_linii;contor++)
    imagine[contor]=(tip*) malloc(nr_coloane*sizeof(typ));
```

Se remarcă folosirea constantelor *nr\_linii* și *nr\_coloane* (cu semnificație evidentă) și a tipului generic *tip* pentru valoarea pixelilor. Linia a 3-a alocă spațiul pentru un masiv de pointeri la date de tip pointer; spațiul de memorie necesar (argumentul funcției *malloc*) este dat de numărul de pointeri la liniile imaginii ce înmulțește dimensiunea unui astfel de pointer (*sizeof(typ\*)*). Valoarea *imagine[contor]* este adresa de început a spațiului de memorie la care încep valorile pixelilor de pe linia *contor*; aceștia sunt stocați într-un vector declarat de *malloc(nr\_coloane\*sizeof(typ))*. Trebuie remarcată conversia de tip (cast) obligatorie ce însoțește fiecare alocare de memorie (se știe că funcția *malloc* întoarce un pointer la *void*). De asemenea se observă că secvența anterioară nu face nici un fel de verificare a succesului operației de alocare de memorie (verificarea faptului că valoarea returnată de funcția *malloc* nu este *NULL*). În mod implicit, la compilare, toți pixelii (toate valorile matricii *imagine*) sunt inițializați cu 0.

Spre deosebire de C, limbajul Matlab<sup>3</sup> aduce mari simplificări. Există un singur tip de date, reprezentate pe 8 octeți (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 și o coloană). Funcțiile returnează matrici. Secvența C anterioară este echivalentă cu:

```
imagine=zeros(nr_linii,nr_coloane);
```

---

<sup>3</sup>Codurile Matlab prezentate în lucrare corespund versiunii Matlab 4.2c.

### 1.3.2 Stocarea imaginilor în fișiere

Un fișier este entitatea logică de organizare a informației înscrise pe mediile magnetice de stocare și se compune dintr-un șir de octeți. Pentru stocarea imaginii este necesar ca acești octeți să conțină informația aferentă pixelilor precum și informație despre tipul imaginii: dimensiunile acesteia, dacă este sau nu indexată, dacă are sau nu o tabelă de culoare atașată, dacă este sau nu comprimată și după ce metodă. Anumite structuri de fișiere 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 fișierelor ce conțin imaginile stocate după formatul respectiv: BMP, TIF, GIF, PCX, JPG ... . În cele ce urmează ne vom referi la formatele RAW(cunoscut și ca IMG), unul dintre cele mai rudimentare formate de fișiere imagine, și Windows Bitmap -BMP al firmei Microsoft, care este unul dintre cele mai larg recunoscute formate de fișiere.

Un fișier RAW conține imagini indexate cu nivele de gri, de formă pătrată. Fișierul nu are antet (dimensiunile imaginii fiind deduse din dimensiunea fișierului ce o conține) și nu conține 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 biți (4, 8, etc.); imaginea este baleiată în ordinea normală (începând cu prima linie a imaginii, de la stânga la dreapta).

Un fișier BMP<sup>4</sup> are trei componente consecutive: un antet de fișier (BITMAPFILE-HEADER), o structură de informație a imaginii(BITMAPINFO) și codarea pixelilor. Antetul de fișier (BITMAPFILEHEADER) conține informații asupra tipului, dimensiunii și reprezentării fișierului Bitmap independent de dispozitiv (DIB - Device Independent Bitmap); semnificațiile componentelor sunt date în tabelul 1.4.

```
typedef struct tagBITMAPFILEHEADER{  
WORD bfType;  
DWORD bfSize;  
WORD bfReserved1;  
WORD bfReserved2;  
DWORD bfOffBits;  
}BITMAPFILEHEADER;
```

Structura de informație a imaginii (BITMAPINFO) conține informații asupra dimensiunilor și culorilor unui DIB, și este alcătuită din două componente: antetul structurii de informații (BITMAPINFOHEADER), a cărui componente sunt descrise în tabelul 1.5 și tabelul de culoare, format din structuri RGBQUAD.

---

<sup>4</sup>Denumirile componentelor logice ale fișierului sunt cele standardizate de Microsoft.

Câmp	Descriere
bfType	Specifică tipul de fișier; trebuie să conțină caracterele BM
bfSize	Specifică lungimea fișierului în DWORD
bfReserved1	Câmp rezervat, valoare 0
bfReserved2	Câmp rezervat, valoare 0
bfOffBits	Specifică deplasamentul în octeți de la sfârștul structurii BITMAPFILEHEADER până la zona din fișier ce conține pixelii codați

Tabel 1.4: Descrierea câmpurilor structurii BITMAPFILEHEADER

Câmp	Descriere
biSize	Numărul de octeți ai structurii BITMAPINFOHEADER
biWidth	Lățimea imaginii, în pixeli
biHeight	Înălțimea imaginii, în pixeli
biPlanes	Numărul de plane de culoare ale dispozitivului de afișaj (1)
biBitCount	Numărul de biți 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 biți
biSizeImage	Dimensiunea imaginii în octeți
biXPelsPerMeter	Rezoluția pe orizontală a dispozitivului țintă (în pixeli pe metru)
biYPelsPerMeter	Rezoluția pe verticală a dispozitivului țintă (în pixeli pe metru)
biClrUsed	Numărul de culori utilizate în imagine; dacă este 0, imaginea folosește 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 roșu, verde și albastru, și 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 biți; dacă biBitCount este 1, 4 sau 8, imaginea va fi indexată și fișierul conține tabela de culoare asociată imaginii. Codurile pixelilor se grupează pe octeți (deci pentru o codare de 4 biți per pixel, fiecare octet de cod va corespunde la doi pixeli alăturați). Dacă biBitCount este 24, pentru fiecare pixel se asociază direct trei octeți, ce reprezintă componentele de roșu, verde și albastru ale culorii respective; această imagine se numește *True Color* și 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 ( $2^{24}$ ) depășește limita sensibilității umane de discernere a culorilor.

Codarea se face independent pe fiecare linie orizontală a imaginii. Codurile (indexurile) tuturor pixelilor unei linii sunt concatenate; șirul rezultat trebuie să fie multiplu de 32 de biți (sau de 4 octeți, sau să conțină un număr întreg de DWORD's). Dacă această constrângere nu este respectată, linia respectivă se completează cu numărul necesar de biți (care, în mod evident, nu vor fi utilizați la citirea imaginii din fișier). Codarea imaginii începe cu ultima linie, și pentru fiecare linie baleiajul este normal (de la stânga la dreapta).



## Capitolul 2

# TEHNICI DE ÎMBUNĂTĂȚIRE A IMAGINILOR

Îmbunătățirea imaginilor este o sintagmă generală ce se referă la o clasă largă de operații al căror scop este mărirea detectabilității componentelor imaginii. Detectabilitatea componentelor este legată mai mult de percepția vizuală a unui observator uman decât de o analiză automată cantitativă. Percepția vizuală de referință este cea a unui expert uman în domeniul aplicației din care provine imaginea.

Așadar criteriile de evaluare ale calității unei imagini sunt subiective și specifice aplicației. În [2] se face analogia operațiilor de îmbunătățire a imaginilor cu reglajul tonalității muzicii ascultate; în funcție de ascultător, se vor favoriza componentele înalte sau joase, sau nici unele. Ca o consecință, procesul de îmbunătățire va fi interactiv, transformările efectuate trebuind să fie validate (cel puțin în etapa de proiectare sau probă) de către un utilizator uman.

Principiul (aproape unanim acceptat) este că îmbunătățirea calității unei imagini se face fără a lua în considerare nici o informație asupra imaginii originale sau asupra procesului de degradare (prin care imaginea nu este suficient de bună). Conform acestui punct de vedere chiar și o imagine originală (nedeградată) poate fi îmbunătățită, obținând 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) și pe baza netezimii în regiunile uniforme.

Creșterea uniformității regiunilor este însă asimilată eliminării unui eventual zgomot suprapus imaginii, operație 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 operații de îmbunătățire:

- operații punctuale, prin care se realizează o corespondență de tip “unu la unu” între vechea valoare a nivelului de gri și noua valoare a acestuia, pentru fiecare pixel al imaginii. Tot în această categorie vom include și operațiile de pseudocolorare, care se referă la afișarea imaginii folosind o paletă de culoare modificată.
- operații locale (sau de vecinătate), prin care noua valoare a nivelului de gri într-un pixel este obținută din vechea valoare a pixelului respectiv și din valorile unor pixeli vecini pixelului considerat.
- operații integrale, în care noua valoare a unui pixel este dependentă de valorile tuturor pixelilor imaginii

În acest capitol vom studia doar operațiile punctuale și de pseudocolorare.

Prin îmbunătățire, unei imagini nu i se adaugă nici o informație nouă față de cea ce exista inițial [9] (deci nu se adaugă nimic imaginii), ci doar este prezentat altfel conținutul inițial al acesteia. Deși la o examinare superficială afirmația este corectă, putem găsi măcar două obiecții (sau contraexemple) la această formulare:

- din punctul de vedere al utilizatorului, informația, chiar dacă există, nu poate fi folosită, deci este asimilabil nulă. Acesta este cazul imaginilor obținute în condiții extreme de iluminare, ce prezintă un contrast foarte slab (imagini subexpuse sau supraexpuse) [5].
- din punctul de vedere al teoriei informației, informația 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 distribuția nivelelor de gri din imagine.

## 2.1 Operații punctuale de modificare a contrastului

Operațiile punctuale de modificare a contrastului (numite și 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 funcție:

$$v = T(u), u \in [0; L - 1] \quad (2.1)$$

În [5] se stabilesc ca necesare condițiile ca:

- transformarea  $T$  să păstreze gama admisibilă de valori ale imaginii (dacă nivelele de gri au fost reprezentate pe  $L$  nivele de cuantizare, atunci  $0 \leq T(u) \leq L - 1$ ,  $\forall u \in [0; L - 1]$ )

- 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 porțiuni, dată de:

$$v = \begin{cases} \frac{\alpha}{T_1}u, & 0 \leq u < T_1 \\ \alpha + \frac{\beta-\alpha}{T_2-T_1}(u-T_1), & T_1 \leq u < T_2 \\ \beta + \frac{L-1-\beta}{L-1-T_2}(u-T_2), & T_2 \leq u < L \end{cases} \quad (2.2)$$

În formula anterioară, parametrii de control sunt  $T_1$ ,  $T_2$ ,  $\alpha$  și  $\beta$ ; aceștia sunt grupați câte doi, definind punctele  $(T_1, \alpha)$  și  $(T_2, \beta)$ . Aceste două puncte de control, împreună cu punctele fixe  $(0, 0)$  și  $(L-1, L-1)$  vor defini cele trei segmente de dreaptă ce apar în formula (2.2). Rezultatul aplicării unei asemenea operații punctuale se obține modificând valoarea (nivelul de gri) fiecărui pixel al imaginii inițiale,  $u$ , conform (2.2), obținând 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 și  $L-1$ ) și 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 și au fost eliminate structurile condiționale *if* - impuse de definiția 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 obținute), iar alăturat imaginea obținută cu o modificare liniară pe porțiuni

a contrastului, determinată de parametrii  $T_1 = 30$ ,  $T_2 = 100$ ,  $\alpha = 20$ ,  $\beta = 200$ . În figura 2.2 este prezentată funcția de transformare a nivelelor de gri ( $T(u)$ ).



Fig. 2.1: Imagine originală și imagine cu contrastul modificat

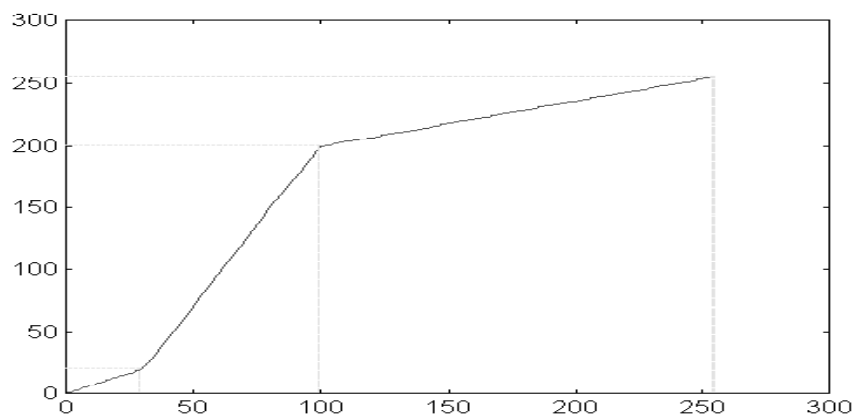


Fig. 2.2: Îmbunătățire liniară pe porțiuni, definită de punctele  $(0,0)$ ,  $(30,20)$ ,  $(100,200)$ ,  $(255,255)$

Vizibilitatea componentelor scenei este în general determinată în cea mai mare parte de contrastul zonei din imagine; contrastul este o măsură proporțională cu diferența dintre luminozitatea anumitor pixeli (nivelul lor de gri). Pentru a putea prevedea deci efectele unei operații de îmbunătățire de tipul prezentat asupra contrastului este deci suficientă studierea diferențelor de nivele de gri între o aceeași pereche de pixeli înainte și după efectuarea transformării. La limită, este posibil ca pixelii să aibă nivelul de gri original diferit cu doar o unitate (cuanta minimă), și atunci modificarea contrastului va fi dată

de diferența valorilor transformate, adică de derivata funcției de transformare:

$$C = \frac{\Delta v}{\Delta u} = \frac{T(u_2) - T(u_1)}{u_2 - u_1} = \frac{dT(u)}{du} = T'(u) \quad (2.3)$$

Pentru funcția liniară pe porțiuni este evident că derivata va fi constantă pe aceleași intervale, având valoarea egală cu panta segmentului de dreaptă.

$$C = \begin{cases} \frac{\alpha}{T_1}, & \text{dacă } u \in [0; T_1] \\ \frac{\beta - \alpha}{T_2 - T_1}, & \text{dacă } u \in [T_1; T_2] \\ \frac{L - 1 - \beta}{L - 1 - T_2}, & \text{dacă } u \in [T_2; L - 1] \end{cases} \quad (2.4)$$

Dacă pe un interval această pantă este subunitară, atunci diferența între nivelele alăturate de gri se micșorează și deci contrastul scade; dacă din contră, panta dreptei este supraunitară, diferența dintre nivelele de gri alăturate se mărește și contrastul va crește. Spre exemplu, în transformarea din figura 2.2, pe intervalul  $[30, 100]$  contrastul crește, iar pe intervalele  $[0, 30]$  și  $[100, 255]$  contrastul scade. Aceste efecte sunt ușor 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 dispariția detaliilor luminoase din imagine (panglica lată de la pălărie); creșterea contrastului pe intervalul  $[30, 100]$  (deci griuri închise) este vizibil în zona penei de pălărie, ale cărei detalii sunt acum mult mai sesizabile.

În funcție de alegerea celor patru parametri, se pot obține câteva cazuri particulare de interes ce poartă denumiri specifice.



Fig. 2.3: Imagine originală și imagine binarizată cu pragul 125

Dacă  $T_1 = T_2$  și  $\alpha = 0$ ,  $\beta = L - 1$ , se obține prăguirea sau binarizarea (“thresholding”) (vezi figura 2.4); în imaginea rezultată nu există decât alb și negru (figura 2.3); toate

nivelele de gri inițiale a căror valoare era mai mică decât  $T_1$  fiind negre și toate nivelele de gri inițiale mai mari ca  $T_1$  devenind albe. După cum se va vedea la capitolul de segmentare orientată pe regiuni (capitolul 8), aceasta este și una dintre tehnicile cele mai simple de segmentare. În urma acestei transformări, contrastul este maximizat la nivelul întregii imagini.

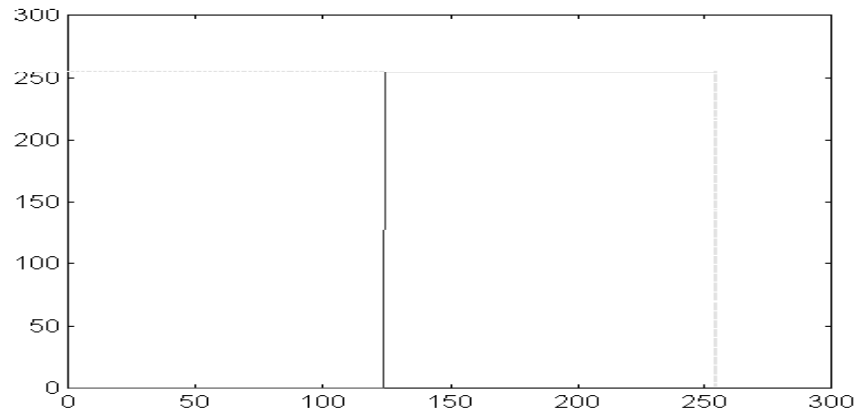


Fig. 2.4: Transformarea de binarizare

Dacă  $\alpha = 0$  și  $\beta = L - 1$  se obține operația de întindere maximă a contrastului (*contrast stretching*) (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.

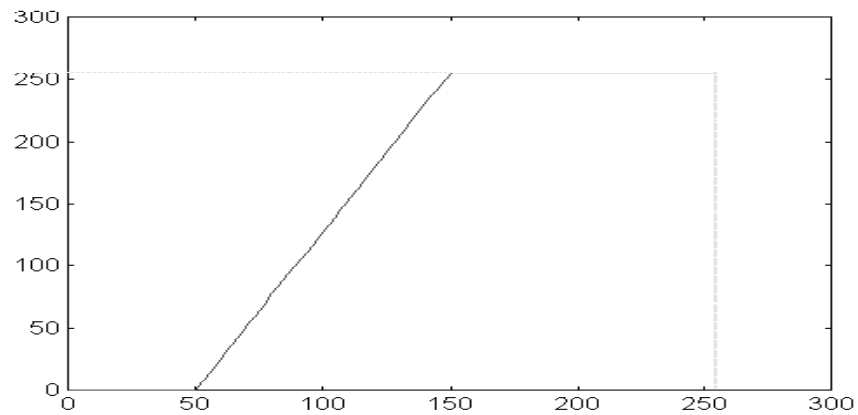


Fig. 2.5: Transformarea de întindere maximă a contrastului

### 2.1.2 Modificarea neliniară a contrastului

Principalul dezavantaj al tehnicii liniare pe porțiuni prezentate este faptul că modificarea contrastului este aceeași pe un întreg interval de nivele de gri, și 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 proprietăți.

O primă variantă este compandarea domeniului [9], definită de o curbă logaritmică și cu punctele fixe  $(0, 0)$  și  $(L - 1, L - 1)$ :

$$v = T(u) = \frac{L - 1}{\lg L} \lg(1 + u) \quad (2.5)$$

Contrastul va varia neuniform de-a lungul scalei de gri, mărindu-se la capătul inferior (negru) și micșorându-se la capătul superior (alb). În mod reciproc se poate defini expandarea domeniului, ca transformare inversă celei de compandare, și deci având o alură exponențială:

$$v = T(u) = (L - 1) \frac{e^u - 1}{e^{L-1} - 1} \quad (2.6)$$

Contrastul va varia neuniform de-a lungul scalei de gri, mărindu-se la capătul superior (alb) și micșorându-se la capătul inferior (negru). Termenii de compandare și de expandare au fost dați prin asemănare cu transformările folosite în teoria codării și cuantizării (ce intervin în metodele de cuantizare a semnalului vocal pentru telefonie digitală, cunoscute sub numele de legea A în Europa și legea  $\mu$  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 obținute prin folosirea unor funcții de tip putere; și acestea au nivelele de gri extreme ca puncte fixe  $((0, 0)$  și  $(L - 1, L - 1))$ . O primă variantă este funcția putere:

$$v = T(u) = (L - 1) \left( \frac{u}{L - 1} \right)^r \quad (2.7)$$

După valorile parametrului-putere  $r$  se pot obține două comportări diferite: pentru  $r < 1$  comportarea este de același tip cu al funcției de compandare logaritmice, iar pentru  $r > 1$  comportarea este de tipul funcției de expandare. Trebuie remarcat că legile de variație ale contrastului vor fi însă diferite.

Există însă și o variantă la care se mai adaugă un punct fix  $(T, T)$ , funcția devenind cu două intervale de definiție:

$$v = T(u) = \begin{cases} T \left( \frac{u}{T} \right)^r, & \text{dacă } u \in [0; T] \\ L - 1 - (L - 1 - T) \left( \frac{L-1-u}{L-1-T} \right)^r, & \text{dacă } u \in [T, L - 1] \end{cases} \quad (2.8)$$

Funcția are o alură de tipul celei prezentate în figura 2.6.

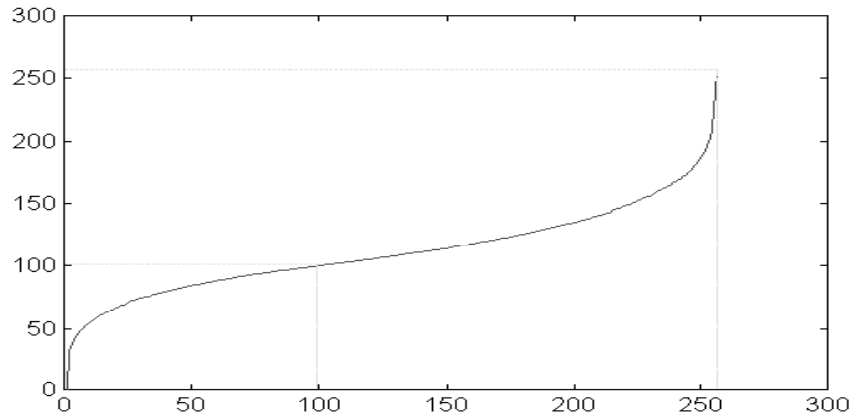


Fig. 2.6: Modificare neliniară a contrastului, cu trei puncte fixe

În [9], în cadrul operațiilor punctuale de îmbunătățire a imaginilor sunt prezentate și operații aritmetice simple, ca de exemplu negativarea. Negativarea este descrisă de:

$$v = T(u) = L - 1 - u \quad (2.9)$$

Efectul acesteia de modificare a contrastului se bazează doar pe caracteristicile sistemului vizual uman, pentru care contrastul depinde de diferența de luminozitate între pixeli aparținând unui obiect, respectiv fundalului, raportată la luminanța medie a fundalului. O categorie aparte de aplicații în care sunt utile asemenea inversiuni este analiza imaginilor medicale, care pentru o analiză automată trebuie inversate; un astfel de exemplu este imaginea angiografică din figura 2.7.

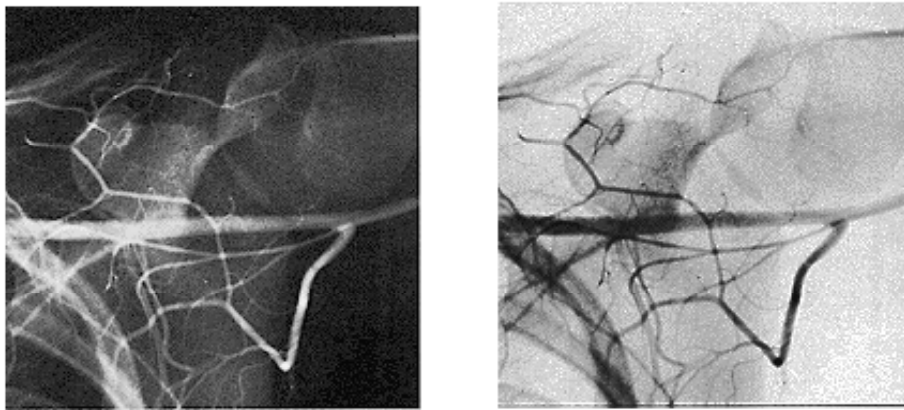


Fig. 2.7: Imagine originală și negativată (dintr-o aplicație medicală)

Alte operații posibile (tratate în [9], dar de mai mică semnificație 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 și le anulează pe celelalte - imaginea rezultată conținând obiectele de interes pe fond negru) și transformarea de tăiere *slicing* (care transformă nivelele de gri dintr-un interval fixat în alb și tot restul în negru; nu este altceva decât un alt tip de binarizare).

## 2.2 Pseudocolorarea

Pseudocolorarea este o tehnică de îmbunătățire a vizibilității anumitor componente ale imaginii (sau a imaginii în ansamblu) prin modificarea paletei de culoare cu care imaginea este afișată (reprezentată). Aceasta înseamnă că pentru anumite nivele de gri, afișarea nu se va mai face cu culoarea a cărei componente sunt toate egale cu indexul (nivelul de gri), ci cu o altă culoare. Această definiție acoperă însă și cazul operațiilor de modificare a contrastului prezentate anterior; funcția  $v = T(u)$  nu este altceva decât o funcție de construcție a unei noi paleti de culoare pentru aceeași 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. Remarcați cast-ul (schimbarea de tip) la tipul *color*; acesta este inserat mai mult ca o măsură de atenționare: culoarea (deci variabila de tip *color*) trebuie obținută 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 evidență zone de interes din imagini cu nivele de gri (există și varianta colorării false - *false coloring*, care transformă o imagine color într-o altă imagine color, dar cu un contrast mult mai pronunțat și artificial între elementele sale). Această idee este normală dacă se are în vedere faptul că ochiul (sistemul vizual) uman distinge ceva mai puțin de 256 nuanțe de gri, deși diferențiază câteva milioane de culori [9].

O aplicație interesantă a pseudocolorării este prezentată în [2]. La NASA, la începuturile erei digitale în tehnicile de achiziție 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 și precis care era iluminarea cea mai potrivită (care nici nu producea suprailuminare și nici nu lăsa umbre mai mari decât era necesar), la afișarea în timp real a imaginii achiziționate pe monitoarele de control, paleta de griuri a fost modificată pe ultima poziție (alb), unde s-a inserat roșu. Atunci instrucțiunile pentru operator erau: crește curentul prin lampă (iluminarea) până când în imagine apare roșu, după care redu curentul până când culoarea roșie dispăre. Rezultatul acestei tehnici simple au fost mii de imagini digitizate corect.

În final merită poate amintită remarca (destul de acidă) din [2]:

“Deși prin natura sa este un detaliu al tehnicilor de afișare, 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 demonstrațiile produselor [software], deoarece poate stârni interesul în ochii clienților mult mai repede decât orice altă metodă de afișare cunoscută. Cercetările mele au adus la lumină o listă dureros de scurtă a aplicațiilor demonstabile productive a pseudocolorării”

## 2.3 Operații de contrastare bazate pe histograma imaginii

Pentru o imagine  $f$  de  $M \times N$  pixeli și  $L$  nivele de gri, histograma este definită (2.10) ca probabilitatea (frecvență relativă) de apariție în imagine a diferitelor nivele de gri posibile.

$$h(i) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \delta(i - f(m, n)) \quad , \quad i = 0, 1, \dots, L-1 \quad (2.10)$$

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 funcția de densitate de probabilitate a acestei variabile aleatoare. Fiind o funcție de densitate de probabilitate, histograma oricărei imagini verifică condiția 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 și contorizarea numărului de nivele de gri întâlnite. Presupunând  $L$  nivele de gri posibile în imaginea de dimensiuni  $NRLIN$  și  $NRCOL$ , codul

următor alocă pentru fiecare nivel de gri posibil câte un *unsigned int*, ce va contoriza numărul de apariții ale nivelului de gri respectiv. Pentru fiecare punct al imaginii se incrementează poziția 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 obținute pot fi normate pentru a obține o funcție de densitate de probabilitate prin împărțirea cu  $NRLIN*NRCOL$  și 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 funcții 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
```

Funcția *find* (funcție standard a pachetului Matlab) returnează pozițiile (indicii) la care este găsită valoarea  $i$  în matricea *imagine* (adică întoarce pozițiile punctelor ce au valoarea  $i$ ); numărând aceste puncte (deci calculând lungimea vectorului în care sunt stocate) se găsește numărul de puncte ce au respectivul nivel de gri. Normarea histogramei se poate face fără a modifica declarațiile de definire a structurilor (pentru Matlab este indiferent dacă se stochează întregi sau numere cu parte zecimală). Această structură este mai rapidă decât cea prin care se parcurgea toată imaginea deoarece se folosește o singură buclă de lungime  $L$  (și nu două bucle imbricate, de lungime totală  $NRLIN*NRCOL$ ), iar funcția *find* este rapidă, fiind o funcție precompilată.

Histograma imaginii oferă informații asupra plasamentului în “nuanță” a conținutului imaginii (vezi figura 2.8). La majoritatea imaginilor există o distribuție neuniformă a nivelelor de gri; există nivele de gri predominante și există nivele de gri folosite puțin sau

deloc. Operațiile de îmbunătățire a imaginilor (pentru îmbunătățirea percepției vizuale) au ca scop redistribuirea nivelelor de gri, astfel ca acestea să ocupe întreaga gamă de variație disponibilă, în mod uniform: aceasta este egalizarea de histogramă [9], [18].

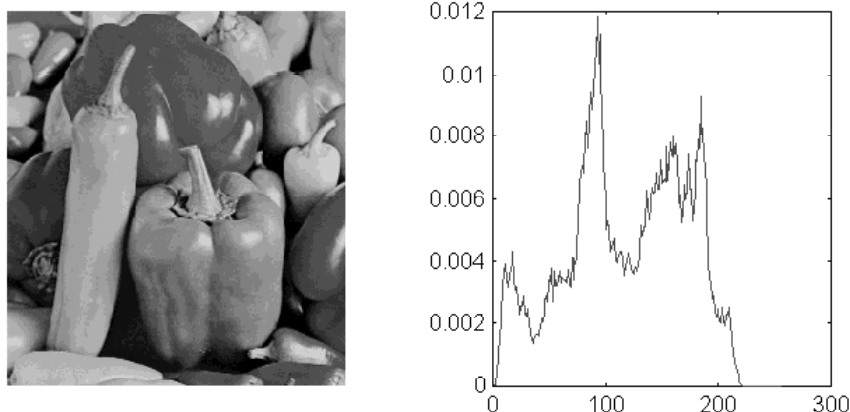


Fig. 2.8: Imagine cu nivele de gri și histograma acesteia

Scopul egalizării de histogramă este deci obținerea unei distribuții uniforme a nivelelor de gri; imaginea rezultată va prezenta cea mai mare îmbunătățire a contrastului, distribuit regulat în întreaga gamă dinamică a nivelelor de gri. Din punct de vedere matematic, egalizarea de histogramă înseamnă transformarea unei distribuții oarecare (descrisă de histograma imaginii inițiale) într-o distribuție uniformă. Considerând variabilele aleatoare  $X(\xi, x)$  și  $Y(\xi, y)$  legate prin  $Y = g(X)$ , atunci între funcțiile de densitate de probabilitate a celor două variabile aleatoare există relația [16]:

$$f_Y(y) = f_X(x) \frac{1}{|(g^{-1}(y))'|} \Big|_{x=g^{-1}(y)}$$

Dacă dorim ca funcția de densitate de probabilitate  $f_Y(y)$  să fie uniformă (în condițiile în care funcția de densitate de probabilitate  $f_X(x)$  este dată), atunci înseamnă că vom avea  $|(g^{-1}(y))'| = \frac{1}{k} f_X(g^{-1}(y))$ . Rezolvarea acestei ecuații produce soluția  $y = g(x) = \int_{-\infty}^x f_X(t) dt$ .

Pentru cazul particular al imaginilor, variabila aleatoare  $X$  ia valori naturale - nivelele de gri. Funcția de densitate de probabilitate  $f_X(x)$  este histograma [normată] a imaginii iar funcția de transformare devine  $y = g(x) = \int_0^x f_X(t) dt$ . Ținând seama că valorile de gri sunt discrete, integrala se transformă în sumă și această formă nu este altceva decât

histograma cumulativă a imaginii; dacă  $h$  este histograma imaginii, atunci

$$g(x) = H(x) = \sum_{i=0}^x h(i) \quad (2.11)$$

Valorile funcției trebuie însă redistribuite în intervalul permis de valori de gri, ceea ce duce la deducerea formulei care exprimă noile valori de gri:

$$v = \left[ \frac{H(u) - H(0)}{MN - H(0)}(L - 1) + 0.5 \right] \quad (2.12)$$

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 folosește 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”) și rezultatul egalizării de histogramă, precum și histogramele originale și egalizate (figura 2.9). Ceea ce se remarcă cu ușurință este diferența dintre histograma obținută și 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

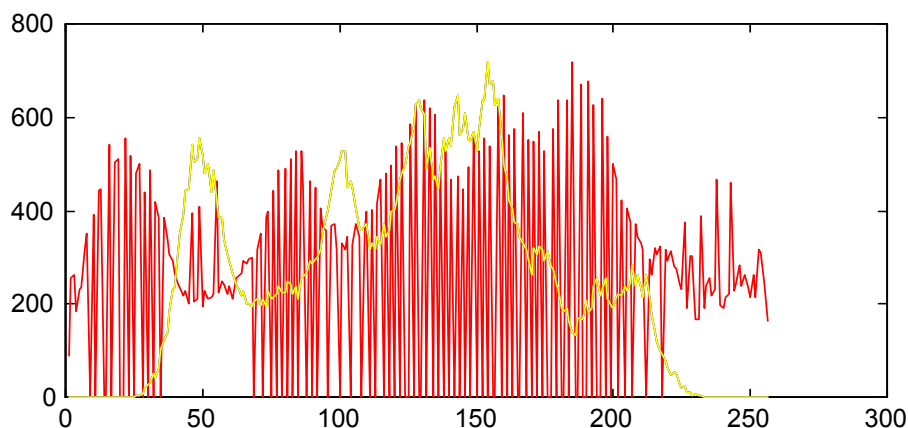


Fig. 2.9: Histograma imaginii “lena”, înainte și după egalizare

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 histogramme 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 îmbunătățire a histogramei în categoria de operații punctuale. Majoritatea autorilor consideră că orice operație care este echivalentă cu modificarea paletei de culoare a imaginii este o operație punctuală. În același timp însă, noile valori de gri ale fiecărui punct sunt calculate pe baza histogrammei 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 operațiilor integrale.

## Capitolul 3

# FILTRAREA LINIARĂ A IMAGINILOR

Odată cu operațiile de filtrare liniară, se începe studiul operatorilor de vecinătate: operatorii al căror rezultat depinde atât de valoarea punctului în care au fost aplicați, cât și de valorile unui număr de alte puncte vecine (nu neapărat topologic) punctului curent de calcul. Filtrarea se cheamă liniară pentru că operația verifică principiul superpoziției (liniarității): pentru două imagini  $f_1$  și  $f_2$ , doi scalari  $a_1$  și  $a_2$  și operatorul liniar  $L$  avem

$$L(a_1 f_1 + a_2 f_2) = a_1 L(f_1) + a_2 L(f_2) \quad (3.1)$$

Operația de filtrarea liniară calculează noua valoare a unui pixel al imaginii (din poziția  $(m, n)$ ) ca o combinație liniară (medie ponderată) a unui număr de valori din imaginea originală:

$$v(m, n) = \sum_{(k, l) \in W} w_{kl} f(m - k, n - l) \quad (3.2)$$

$W$  este vecinătatea punctului curent în care se face calculul;  $W$  este numită mască sau fereastră de filtrare și este o formă plană, descrisă ca o mulțime de puncte din spațiul cartezian, în coordonate relative (deci în alt sistem de coordonate decât sistemul de coordonate a imaginii). Scalarii  $w_{kl}$  sunt atașați pozițiilor  $(k, l)$  din fereastră de filtrare și poartă numele de coeficienți ai filtrului. Spre exemplu, o mască simplă de mediere este media aritmetică a valorilor dintr-o vecinătate pătrată de  $3 \times 3$  pixeli, centrată în pixelul curent; pentru acest caz, toți coeficienții vor avea valoarea  $1/9$  și 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:

$$v(m, n) = \sum_{k=-1}^1 \sum_{l=-1}^1 \frac{f(m - k, n - l)}{9} \quad (3.3)$$

adică, desfășurat,  $v(m, n) = \frac{f(m+1, n-1)}{9} + \frac{f(m+1, n)}{9} + \frac{f(m+1, n+1)}{9} + \frac{f(m, n)}{9} + \frac{f(m, n-1)}{9} + \frac{f(m, n+1)}{9} + \frac{f(m+1, n-1)}{9} + \frac{f(m+1, n)}{9} + \frac{f(m+1, n+1)}{9}$ . Același lucru se poate exprima și prin specificarea matricială a măștii de filtrare și 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 definiție (3.2) nu este altceva decât suma produselor punct cu punct a coeficienților măștii și 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 operații de convoluție bidimensională, în care, prin convenție, masca de filtrare este considerată nucleul de convoluție. Deplasarea măștii (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 și se selectează punctele imaginii situate în interiorul ferestrei (putem imagina fereastra de filtrare ca fiind o apertură într-o placă opacă; într-o anumită poziție a acesteia valorile selectate din imagine sunt valorile punctelor ce se “văd” prin apertură).
- pentru fiecare poziție se face suma produselor punct cu punct coeficient mască - valoare pixel.

Fereastra de filtrare este deci definită de formă (mulțimea  $W$ ) și valori (coeficienții  $w_{kl}$ ). 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 și nuclee mici, făcând referință la operația de convoluție). Codul care urmează exemplifică această cea mai simplă filtrare, cu un nucleu de dimensiune 3 x 3. Se remarcă alocarea statică a coeficienților  $w$  ai filtrului (numere reale) și alocarea dinamică a imaginii rezultat (considerată aici ca având valori întregi). Calculul valorilor pixelilor din imaginea filtrată s-a făcut decupând câte un rând și o coloană de la extremitățile 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 și o coloană cu valori nule. De asemenea, se poate constata că în expresia de calcul efectiv indicii tabelului de coeficienți ai măștii au fost deplasați, astfel încât indicele minim să fie 0, și nu negativ.

```
real w[3][3];
img_out=(int**)malloc(NRLIN*sizeof(int*));
for (i=0;i<NRLIN;i++)
```



```

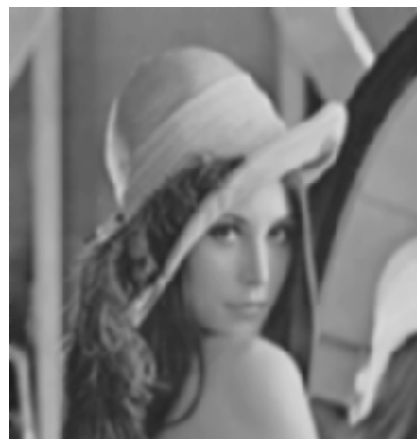
    img_out[i]=(int*)malloc(NRCOL*sizeof(int));
for (i=0;i<NRCOL;i++)
{ img_out[0][i]=img[0][i];
  img_out[NRLIN-1][i]=img[NRLIN-1][i];}
for (i=0;i<NRLIN;i++)
{ img_out[i][0]=img[i][0];
  img_out[i][NRCOL-1]=img[i][NRCOL-1];}
for (i=1;i<NRLIN-1;i++)
  for (j=1;j<NRCOL-1;j++)
    img_out[i][j]=round(w[1][1]*img[i][j]+w[1][0]*img[i][j-1]+
      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  $N$  și o mască de filtrare pătrată de dimensiune  $n$ , numărul de operații necesare unei filtrări liniare este de  $N^2 n^2$  înmulțiri și  $N^2(n^2 - 1)$  adunări, adică complexitatea de calcul este  $O[N^2]$ , atât față de dimensiunile imaginii cât și față de dimensiunile ferestrei de filtrare. Aceasta duce la constrângerea practică de a folosi ferestre de filtrare cât mai mici (și de a încerca descompunerea ferestrelor mari de filtrare într-o succesiune de ferestre mai mici - spre exemplu o filtrare cu un nucleu de  $5 \times 5$ , care ar necesita 25 de operații pentru fiecare pixel, poate fi echivalată cu două filtrări succesive cu nuclee de  $3 \times 3$  pixeli, care necesită doar 18 operații pentru fiecare pixel). În figurile următoare se prezintă efectul unei operații de mediere cu un nucleu de  $7 \times 7$ : se remarcă faptul că imaginea rezultat este mai neclară și cețoasă (efect de *blur*) și că toate contururile au devenit mai vagi.



Imagine originală



Rezultatul unei filtrări de mediere

### 3.1 Filtrarea liniară de netezire

Efectul de încetșare a unei imagini poate fi considerat și ca un efect de îmbunătățire a uniformității regiunilor [18]. Aceasta înseamnă că se elimină micile diferențe dintre valorile pixelilor aparținând unei aceleiași 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 toți pixelii ce o formează au aceeași valoare) produce variația valorilor din interiorul acesteia, deci micșorează uniformitatea. Variațiile introduse sunt cu atât mai mari cu cât puterea zgomotului este mai mare; pentru un zgomot de medie nulă (așa cum este zgomotul alb gaussian aditiv) puterea este identică cu varianța. Teoria proceselor aleatoare [16] arată că, pentru o combinație liniară de realizări ale unei variabile aleatoare  $y = \sum_{i=1}^n a_i x_i$ , varianța noii variabile aleatoare este proporțională cu varianța

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, varianța 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) și  $g$  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) și eroarea medie absolută (Mean Absolute Error - MAE) (3.7).

$$SNR = 10 \log \frac{\sum_{n=1}^N \sum_{m=1}^M f^2(n, m)}{\sum_{n=1}^N \sum_{m=1}^M (g(m, n) - f(n, m))^2} \text{ dB} \quad (3.4)$$

$$PSNR = 10 \log \frac{NM \left( \max_{n, m} f(n, m) \right)^2}{\sum_{n=1}^N \sum_{m=1}^M (g(m, n) - f(n, m))^2} \text{ dB} \quad (3.5)$$

$$MSE = \frac{1}{NM} \sum_{n=1}^N \sum_{m=1}^M (g(m, n) - f(n, m))^2 \quad (3.6)$$

$$MAE = \frac{1}{NM} \sum_{n=1}^N \sum_{m=1}^M |g(m, n) - f(n, m)| \quad (3.7)$$



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) și 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 și factori de calitate mai buni (SNR și PSNR mai mari, MAE mai mic).

Un caz particular de interes este considerarea regiunilor constante (absolut uniforme) ale imaginii; în aceste regiuni toți pixelii vor avea aceeași valoare, și deci uniformitatea nu mai poate fi îmbunătățită. În același timp, operația de filtrare de netezire trebuie să păstreze această valoare constantă a pixelilor (pe care o vom nota cu  $\mu$ ); înlocuind în formula de

definiție a filtrării liniare (3.2), vom obține că

$$\mu = \sum_{(k,l) \in W} w_{kl} \mu$$

adică:

$$\sum_{(k,l) \in W} w_{kl} = 1 \quad (3.8)$$

Această condiție (suma coeficienților măștii de filtrare să fie unitară) se numește condiția de normare a nucleelor de filtrare de netezire și definește 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. Măștile următoare rea-

lizează 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 mulțimii  $W$  (3.2)); în același timp însă, practica a demonstrat suficiența considerării unor forme regulate (pătrate, centrate) pentru mască. Orice mască poate fi extinsă cu puncte ce au atașat un coeficient nul, pentru a rezulta o mască pătrată centrată; spre exemplu, masca  $W_4$  poate fi scrisă și 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 și cu măștile  $W_2$  și  $W_3$ . Trebuie remarcat

că toate aceste măști respectă condiția de normare a unui nucleu de netezire (3.8).

## 3.2 Filtrarea liniară de contrastare

Contrastarea unei imagini are ca obiectiv îmbunătățirea percepției vizuale a conturilor obiectelor (îmbunătățirea detectabilității componentelor scenei de-a lungul frontierelor acestora). În principiu, acest deziderat se poate realiza prin modificarea valorilor pixelilor aflați de o parte și de alta a unei frontiere comune.

O experiență de percepție vizuală subiectivă (“benzile lui Mach”) a pus în evidență faptul că sistemul vizual uman are tendința de a adânci profilul zonelor de tranziție 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 informația vizuală; efectul global poate fi descris ca scăderea din semnalul original a unei derivate secunde a acestuia, ponderată corespunzător (așa cum prezintă figurile 3.3 și 3.4, pentru cazul unidimensional - secțiunea unei frontiere între regiunile imaginii).

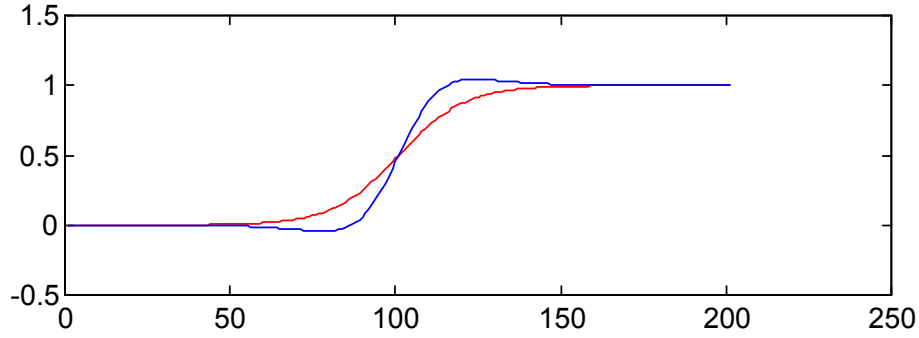


Fig. 3.3: Profilul original de tranziție și profilul adâncit prin scăderea derivatei secunde

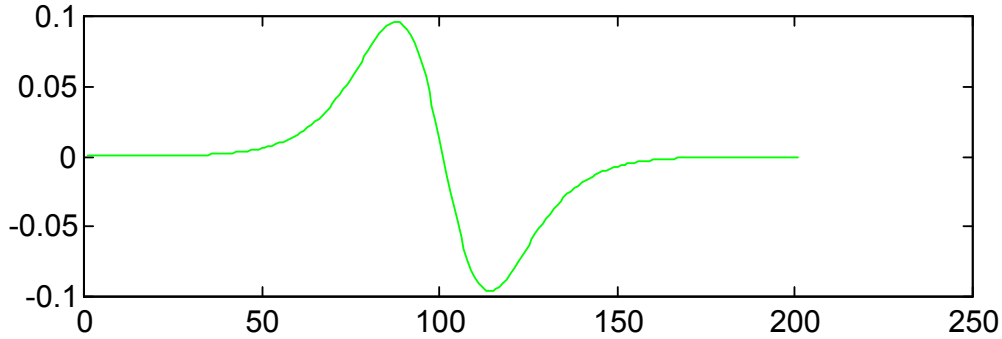


Fig. 3.4: Derivata secundă a profilului original prezentat anterior

Implementarea unei derivate secunde pentru cazul discret va trebui să ia în considerare existența a două direcții de derivare de bază și modul de definire a derivatei în cazul discret (de exemplu prima derivată pe direcția 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  $f'(m, n) = f(m, n+1) - f(m, n-1)$ ). Măști obișnuite de implementare a unei derivate secunde bidirecționale (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 esențial ca răspunsul său pentru pixeli din interiorul unei regiuni absolut uniforme (de valoare  $\mu$ ) să fie nul (derivata unei constante este nulă); pentru aceasta, din formula de definiție a filtrării liniare (3.2) avem că  $0 =$

$\sum_{(k,l) \in W} w_{kl} \mu$ , adică:

$$\sum_{(k,l) \in W} w_{kl} = 0 \quad (3.9)$$

Aceasta este condiția de normare a unui nucleu de filtrare derivativă, pentru care suma coeficienților măștii trebuie deci să fie nulă. Se remarcă faptul că nucleele de derivată secundă (operatori laplacieni) prezentate anterior verifică această condiție de normare (3.9). Figurile următoare prezintă o imagine slab contrastată și varianta sa îmbunătățită prin contrastare cu nucleul laplacian  $W_5$ .



Imagine originală



Contrast îmbunătățit

### 3.3 Filtrarea liniară adaptivă

Termenul de adaptiv se referă la capacitatea filtrului de a-și ajusta comportarea (care este determinată de caracteristicile sale de definiție) în funcție 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 și 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 puțin teoretic) în fiecare punct al imaginii operația este diferită (deoarece se face cu un filtru diferit), filtrarea globală nu mai este liniară (nu mai respectă principiul superpoziției (3.1)).

Adaptarea filtrelor liniare nu poate urmări decât două variante: modificarea formei ferestrei de filtrare, sau modificarea coeficienților unei ferestre de filtrare de formă fixată.

Exemplul cel mai folosit de modificare a formei ferestrei de filtrare este filtrul de netezire direcțională adaptivă [9]. Termenul de filtrare direcțională 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 direcție). Pentru fiecare punct al imaginii se pot considera mai multe orientări (deci mai multe direcții) posibile pentru masca de mediere. Pentru fiecare dintre direcțiile alese se obține în urma filtrării o valoare. În mod ideal, dorim ca valoarea obținută prin filtrare să nu difere în mod semnificativ de valoarea inițială din pixelul considerat. O diferență semnificativă poate însemna că punctul curent este situat pe un contur, și acceptarea acestei valori mult diferite de cea inițială produce efectul de încetșoare a imaginii (caracteristic oricărei filtrări de netezire). Soluția adusă de filtrul adaptiv este de a alege din setul de valori obținute pentru diferitele ferestre de filtrare pe aceea care este cea mai apropiată de valoarea inițială din punctul curent. Fragmentul de cod următor prezintă un caz particular de filtrare direcțională adaptivă: ferestrele direcționale au trei puncte și sunt alese după cele patru direcții principale (orizontal, vertical și cele două diagonale, adică direcțiile orientate la  $0^\circ$ ,  $45^\circ$ ,  $90^\circ$  și  $135^\circ$  față 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 ieșire *img\_out* și nici partea de copiere a valorii marginilor imaginii; se remarcă folosirea a patru variabile suplimentare *out0*, *out45*, *out90*, *out135* care primesc valorile obținute prin filtrarea direcțională cu masca orientată la  $0^\circ$ ,  $45^\circ$ ,  $90^\circ$  și  $135^\circ$ . Variabilele *val\_temp1* și *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 această valoare este suficient de apropiată de valoarea originală se poate aplica și pentru filtrarea de netezire cu fereastră și coeficienți fiși [5]. În acest caz rezultatul netezirii într-un punct va fi acceptat doar dacă diferența față de valoarea originală este mai mică decât un prag impus:

$$g(n, m) = \begin{cases} \sum_{(k,l) \in W} w_{kl} f(m-k, n-l), & \text{dacă } \left| f(m, n) - \sum_{(k,l) \in W} w_{kl} f(m-k, n-l) \right| < T \\ f(m, n), & \text{în rest} \end{cases}$$

O variantă clasică de filtrare liniară adaptivă, realizată prin recalcularea coeficienților măștii de filtrare în fiecare punct al imaginii este prezentată în [18]. Acest filtru este o combinație liniară convexă a imaginii zgomotoase și a imaginii obținute din aceasta printr-o filtrare liniară de mediere (cu fereastră constantă), adică:  $\hat{f} = \alpha f + \beta \bar{f}$ . O combinație liniară convexă se obține doar dacă  $\alpha + \beta = 1$ . Deci

$$\hat{f} = \alpha f + (1 - \alpha) \bar{f} \quad (3.10)$$

Imaginea zgomotoasă  $f$  provine dintr-o imagine corectă  $g$ , la care s-a adăugat un zgomot alb aditiv de medie nulă  $n$  ( $\bar{n} = 0$ ) și necorelat cu imaginea ( $\overline{ng} = \bar{n}\bar{g} = 0$ ):  $f = g + n$ . Atunci

$$\hat{f} = \alpha(g + n) + (1 - \alpha)\overline{(g + n)} = \alpha g + \alpha n + (1 - \alpha)\bar{g} \quad (3.11)$$

Ceea ce definește filtrul adaptiv este coeficientul  $\alpha$ ; acesta este calculat local impunând optimizarea unei măsuri de calitate obiective - eroarea pătratică medie între imaginea originală  $g$  și rezultatul filtrării  $\hat{f}$ .

$$\begin{aligned} \varepsilon^2 &= (\hat{f} - g)^2 = (\alpha n + (\alpha - 1)(g - \bar{g}))^2 = \alpha^2 n^2 + 2\alpha(\alpha - 1)ng - 2\alpha(\alpha - 1)n\bar{g} + (\alpha - 1)^2(g - \bar{g})^2 \\ \overline{\varepsilon^2} &= \overline{\alpha^2 n^2 + 2\alpha(\alpha - 1)ng - 2\alpha(\alpha - 1)n\bar{g} + (\alpha - 1)^2(g - \bar{g})^2} = \alpha^2 \overline{n^2} + (\alpha - 1)^2 \overline{(g - \bar{g})^2} \\ \overline{\varepsilon^2} &= \alpha^2 \sigma_n^2 + (\alpha - 1)^2 \sigma_g^2 \end{aligned} \quad (3.12)$$

Formula (3.12) arată că eroarea pătratică medie este suma ponderată dintre varianța (puterea) zgomotului  $\sigma_n^2$  și varianța imaginii originale  $\sigma_g^2$ . Găsirea coeficientului  $\alpha$  optim înseamnă găsirea minimumului lui  $\overline{\varepsilon^2}$ , deci anularea derivatei acestuia.

$$\begin{aligned} \frac{d\overline{\varepsilon^2}}{d\alpha} &= 2\alpha\sigma_n^2 + 2(\alpha - 1)\sigma_g^2 = 0 \\ \alpha &= \frac{\sigma_g^2}{\sigma_g^2 + \sigma_n^2} = 1 - \frac{\sigma_n^2}{\sigma_f^2} \end{aligned} \quad (3.13)$$

De aici rezultă expresia finală a filtrului:

$$\hat{f} = \left(1 - \frac{\sigma_n^2}{\sigma_f^2}\right) f + \frac{\sigma_n^2}{\sigma_f^2} \bar{f} \quad (3.14)$$



Se observă că trebuie cunoscute puterea de zgomot și varianța locală (în fiecare punct) al imaginii zgomotoase. Adaptarea se face între două cazuri limită: dacă puterea de zgomot este mult mai mare decât varianța valorilor imaginii originale ( $\sigma_n^2 \gg \sigma_g^2$ ) atunci, din prima parte a formulei (3.13) rezultă  $\alpha = 0$  și deci filtrul optim este un simplu filtru de mediere; dacă varianța 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  $\alpha = 1$  și filtrul optim este un filtru identitate (trece tot). Prezintă în continuare codul Matlab ce implementează această 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 funcțiile Matlab *mean* (pentru a calcula media valorilor selectate de fereastra de filtrare pătrată de  $3 \times 3$ ) și *std* (pentru a calcula varianța 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 frecvențelor spațiale (o extensie directă a spectrului semnalelor unidimensionale). De asemenea vom arăta că rămâne valabilă teorema convoluției, prin care vom introduce și o nouă modalitate de implementare a filtrelor liniare: filtrarea în domeniul de frecvență.

## Capitolul 4

# TRANSFORMĂRI INTEGRALE UNITARE DISCRETE

### 4.1 Generalități

Această categorie de prelucrări include operații de tip integral – totalitatea pixelilor imaginii inițiale contribuie la obținerea 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

$$\mathbf{U} = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} \mathbf{A}_{kl}^* V(k, l) \quad (4.1)$$

unde  $\mathbf{A}_{kl}^*$  sunt matricile de bază<sup>2</sup> (de dimensiuni  $N \times N$ ) iar  $V(k, l)$  sunt coeficienții dezvoltării în serie. Exprimând relația (4.1) la nivelul fiecărui pixel al imaginii  $\mathbf{U}$  obținem

$$U(m, n) = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} a_{kl}^*(m, n) V(k, l) \quad (4.2)$$

Calculul coeficienților  $V(k, l)$  ai transformării se poate face în condițiile impuse de orto-

---

<sup>1</sup>În acest capitol vom folosi notațiile matriciale și vectoriale clasice: litere mari, drepte și groase pentru matrici ( $\mathbf{A}$  este o matrice), litere mici, drepte și groase pentru vectori ( $\mathbf{v}$  este un vector) și litere înclinate pentru elementele vectorilor și matricilor ( $A(i, j)$ ,  $v(m)$ ).

<sup>2</sup>Numite uneori și imagini de bază.

gonalitate și completitudine a matricilor de bază  $\mathbf{A}_{kl}^*$  prin:

$$V(k, l) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} a_{kl}(m, n) U(m, n) \quad (4.3)$$

Mulțimea coeficienților transformării  $V(k, l)$  formează matricea  $\mathbf{V}$ , transformata imaginii. Atunci relația (4.2) este transformarea directă a imaginii, iar relația (4.3), prin care se obține imaginea din transformata sa, este transformata inversă. Se poate remarca faptul că ambele transformări (directă și inversă) necesită  $N^4$  operații de înmulțire, și deci o complexitate  $O(N^4)$ .

Condiția de ortonormalitate a matricilor de bază se exprimă ca

$$\sum_{m=0}^{N-1} \sum_{n=0}^{N-1} a_{kl}(m, n) a_{k'l'}^*(m, n) = \delta(k - k', l - l'), \forall k, l, k', l' \quad (4.4)$$

și asigură faptul că orice dezvoltare în serie trunchiată minimizează eroarea pătratică de aproximare.

Condiția de completitudine a matricilor de bază se exprimă ca

$$\sum_{k=0}^{N-1} \sum_{l=0}^{N-1} a_{kl}(m, n) a_{kl}^*(m', n') = \delta(m - m', n - n'), \forall m, n, m', n' \quad (4.5)$$

și asigură faptul că baza de matrici folosită este completă.

O transformare de tip (4.2) este deci caracterizată de cei  $N^4$  coeficienți  $a_{kl}(m, n)$ , ce pot fi interpretați ca valori ai unei funcții de patru variabile  $(k, l, m, n)$ , câte două pentru fiecare imagine (inițială și transformată). Această funcție se numește nucleul transformării. Prin definiție, o transformare se zice separabilă dacă nucleul ei este separabil după perechi de variabile corespondente:

$$a_{kl}(m, n) = a_k(m) b_l(n) = a(k, m) b(l, n) \quad (4.6)$$

Impunând condițiile (4.4) și (4.5) acestei noi forme a coeficienților transformării, rezultă că matricile  $\mathbf{A} = \{a(k, m)\}$  și  $\mathbf{B} = \{b(l, n)\}$  trebuie să fie matrici unitare:

$$\mathbf{A} \mathbf{A}^{*T} = \mathbf{A}^T \mathbf{A}^* = \mathbf{I}_N \quad (4.7)$$

$$\mathbf{B} \mathbf{B}^{*T} = \mathbf{B}^T \mathbf{B}^* = \mathbf{I}_N$$

Pentru o transformare separabilă, relațiile de definiție (4.2) și (4.3) pot fi rescrise sub formă matricială ca:

$$\mathbf{V} = \mathbf{A} \mathbf{U} \mathbf{B}^T \quad (4.8)$$

$$\mathbf{U} = \mathbf{A}^{*T} \mathbf{V} \mathbf{B}^* \quad (4.9)$$

Numărul de înmulțiri 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 complexității calculului cu un ordin de mărime; în practică se folosesc numai transformări separabile, pentru care, în plus,  $\mathbf{A} = \mathbf{B}$ . În acest caz particular, relația (4.8) se poate scrie ca

$$\mathbf{V} = \mathbf{A}\mathbf{U}\mathbf{A}^T = \mathbf{A}(\mathbf{A}\mathbf{U}^T)^T = (\mathbf{U}^T\mathbf{A}^T)^T\mathbf{A}^T$$

ceea ce înseamnă că se poate face o transformare unidimensională pe fiecare linie sau coloană a lui  $\mathbf{U}$  urmată de aceeași transformare pe fiecare coloană sau linie a rezultatului. Astfel, transformările practic utilizate în prelucrarea imaginilor (matricilor) sunt (paradoxal ?) unidimensionale.

Aceste observații ne conduc la concluzia că pentru acoperirea cazurilor utilizate în mod curent este suficientă studierea proprietăților transformărilor integrale unitare unidimensionale.

## 4.2 Proprietățile transformărilor unitare unidimensionale

În cele ce urmează vom considera semnalul de intrare  $\mathbf{u} = (u(0), u(1), \dots, u(N-1))$  și transformata sa  $\mathbf{v}$ , obținută prin folosirea matricii unitare  $\mathbf{A}$ . Relațiile de transformare (directă și inversă) sunt  $\mathbf{v} = \mathbf{A}\mathbf{u}$  și  $\mathbf{u} = \mathbf{A}^{*T}\mathbf{v}$ .

1. Energia semnalului se conservă printr-o transformare unitară.

$$E_{\mathbf{v}} = \|\mathbf{v}\|^2 = \mathbf{v}^{*T}\mathbf{v} = (\mathbf{A}\mathbf{u})^{*T}\mathbf{A}\mathbf{u} = \mathbf{u}^{*T}\mathbf{A}^{*T}\mathbf{A}\mathbf{u} = \mathbf{u}^{*T}\mathbf{u} = \|\mathbf{u}\|^2 = E_{\mathbf{u}} \quad (4.10)$$

Energia vectorului semnal este de fapt lungimea (Euclidiană) a acestuia în spațiul  $N$ -dimensional de reprezentare. Conservarea energiei este deci echivalentă cu conservarea lungimii vectorului, deci transformarea unitară este o rotație în spațiul semnalului. Aceasta înseamnă că baza de reprezentare a lui  $\mathbf{u}$  este rotită, iar  $\mathbf{v}$  este proiecția lui  $\mathbf{u}$  pe noua bază.

2. Energia medie a semnalului se conservă printr-o transformare unitară.

$$\overline{\mathbf{v}} = \overline{\mathbf{A}\mathbf{u}} = \mathbf{A}\overline{\mathbf{u}} \quad (4.11)$$

$$\overline{E_{\mathbf{v}}} = \overline{\mathbf{v}^{*T}\mathbf{v}} = (\mathbf{A}\overline{\mathbf{u}})^{*T}\mathbf{A}\overline{\mathbf{u}} = \overline{\mathbf{u}}^{*T}\mathbf{A}^{*T}\mathbf{A}\overline{\mathbf{u}} = \overline{\mathbf{u}}^{*T}\overline{\mathbf{u}} = \overline{E_{\mathbf{u}}} \quad (4.12)$$

Corelației componentelor semnalului se modifică; dacă notăm cu  $\mathbf{C}$  matricea de covariație atunci:

$$\mathbf{C}_{\mathbf{u}} = \overline{(\mathbf{u} - \overline{\mathbf{u}})(\mathbf{u} - \overline{\mathbf{u}})^{*T}} \quad (4.13)$$

$$\mathbf{C}_{\mathbf{v}} = \overline{(\mathbf{v} - \bar{\mathbf{v}})(\mathbf{v} - \bar{\mathbf{v}})^*}^T = \overline{\mathbf{A}(\mathbf{u} - \bar{\mathbf{u}})(\mathbf{u} - \bar{\mathbf{u}})^*}^T \mathbf{A}^{*T} = \mathbf{A} \mathbf{C}_{\mathbf{u}} \mathbf{A}^{*T} \quad (4.14)$$

Majoritatea transformărilor unitare au tendința de a aglomera o mare parte a energiei medii a semnalului în relativ puțini coeficienți ai transformării. Deoarece energia totală se conservă prin transformare, mulți coeficienți ai transformării vor conține foarte puțină energie. Energia medie a coeficienților transformării va avea o distribuție neuniformă, chiar dacă în secvența de intrare aceasta era uniform distribuită.

Dacă componentele lui  $\mathbf{u}$  sunt puternic corelate, coeficienții transformării vor fi decorelați (termenii matricii de covariație care nu sunt pe diagonala principală vor avea valori mici comparativ cu valorile de pe diagonală).

3. Entropia unui vector cu componente aleatoare se conservă printr-o transformare unitară

$$H(\mathbf{u}) = \frac{N}{2} \log_2(2\pi e |\mathbf{C}_{\mathbf{u}}|^{1/N}) = \frac{N}{2} \log_2(2\pi e |\mathbf{C}_{\mathbf{v}}|^{1/N}) = H(\mathbf{v}) \quad (4.15)$$

Deoarece entropia este o măsură a cantității de informație (informația medie) înseamnă ca transformările unitare păstrează informația conținută în semnal.

### 4.3 Transformata Fourier discretă

Transformata Fourier este poate cea mai importantă transformare integrală unitară, ce asigură trecerea între spațiul semnalului și spațiul de frecvențe ale semnalului. După cum semnalul este “clasic” (temporal, și deci unidimensional) sau cu suport spațial bidimensional (image), spațiul de frecvență marchează frecvențe propriu-zise sau frecvențe spațiale.

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 exponențialele complexe:

$$F(k, n) = \frac{1}{\sqrt{N}} e^{-j \frac{2\pi}{N} kn} = \frac{1}{\sqrt{N}} w_N^{kn} \quad (4.16)$$

Separabilitatea ne permite deci să studiem majoritatea proprietăților transformării pe cazul unidimensional, urmând ca rezultatele să fie ușor extinse pentru cazul bidimensional. Folosind notațiile introduse în secțiunea 4.2, putem scrie deci transformata Fourier unidimensională directă ca:

$$v(k) = \sum_{n=0}^{N-1} u(n) w_N^{kn}, \quad k = \overline{0, N-1} \quad (4.17)$$

iar transformarea inversă ca

$$u(n) = \frac{1}{N} \sum_{k=0}^{N-1} v(k) w_N^{-kn}, \quad n = \overline{0, N-1} \quad (4.18)$$

Relațiile se extind imediat la cazul bidimensional:

$$v(k, l) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} u(m, n) w_N^{kn+ml}, \quad k, l = \overline{0, N-1} \quad (4.19)$$

$$u(m, n) = \frac{1}{N^2} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} v(k, l) w_N^{-(kn+ml)}, \quad m, n = \overline{0, N-1} \quad (4.20)$$

Trebuie totuși remarcat că formele prezentate ale transformărilor nu mai sunt unitare (apare o asimetrie între transformarea directă și cea inversă prin factorul de scalare de tip  $1/N$ ). Matematic, definiția (4.16) este corectă, dar în practică se folosesc formele neunitare prezentate în (4.17), (4.18) și (4.19), (4.20).

### 4.3.1 Proprietățile fundamentale ale transformatei Fourier

Dintre numeroasele proprietăți ale transformatei Fourier [9], [19] ne vom referi doar la două proprietăți fundamentale: simetria centrală a spectrului de frecvență (având drept consecință importantă faptul că este necesar același spațiu de memorie pentru a reprezenta fie spectrul unei imagini, fie imaginea propriu-zisă) și teorema convoluției circulare (care face legătura între filtrarea în domeniul spațial și în domeniul de frecvență).

Transformata Fourier a unei secvențe (matrici) reale este complex conjugată față de mijlocul său [9]. Aceasta înseamnă că

$$v\left(\frac{N}{2} - k\right) = v^*\left(\frac{N}{2} + k\right), \quad k = \overline{0, \frac{N}{2}} \quad (4.21)$$

$$v\left(\frac{N}{2} - k, \frac{N}{2} - l\right) = v^*\left(\frac{N}{2} + k, \frac{N}{2} + l\right), \quad k, l = \overline{0, \frac{N}{2}}$$

Relațiile arată că există o simetrie centrală a spectrelor de frecvență, în centru aflându-se o valoare reală. Dar singura valoare reală din spectre este cea ce corespunde componentei de frecvență nulă (componenta medie), și deci este necesară o interschimbare a jumătăților de spectru (în cazul secvențelor unidimensionale) sau a sferturilor de spectru (în cazul imaginilor), astfel încât componenta de frecvență nulă să fie în centru. În figura 4.1 este reprezentat modulul spectrului de frecvență al imaginii “lena”, reprezentat cu 256 nivele de gri (valorile au fost scalate după o reprezentare logaritmică) astfel încât valorile mai mari corespund unei nuanțe mai deschise. Se observă în partea dreaptă spectrul central simetrizat, în care valorile maxime (corespunzând frecvențelor cele mai mici, inclusiv

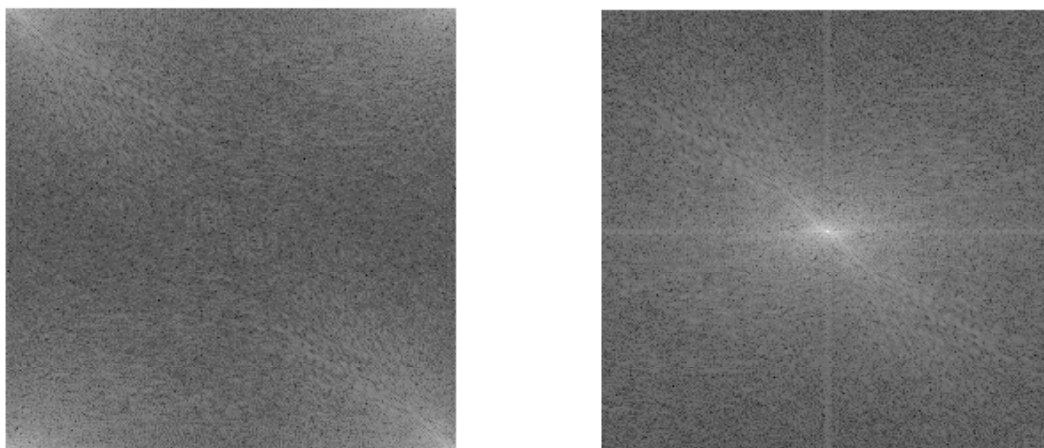


Fig. 4.1: Spectrele de frecvență (Fourier) ale imaginii “lena”, înainte și după aranjarea cu simetrie centrală.

zero) se găsesc în centrul imaginii, iar în partea stângă spectrul original obținut în urma transformării Fourier, în care maximele se găsesc pe cele patru colțuri.

Ca și în cazul unidimensional, și pentru imagini legătura dintre frecvențele spațiale și obiectele din domeniul spațial (imagine) se păstrează: frecvențele spațiale ridicate corespund detaliilor, obiectelor mici, conturilor, zgomotului. În figura 4.2 este prezentat spectrul de frecvență al imaginii “lena” degradată de zgomot impulsiv de tip sare și piper (figura 5.1); zgomotul impulsiv corespunde apariției a numeroase detalii (obiecte mici – punctele de zgomot) și variații ale nivelului de gri. Efectul evident este acela de mărire a valorilor din spectru corespunzătoare frecvențelor înalte și, corespunzător, diminuarea importanței relative a frecvențelor joase.

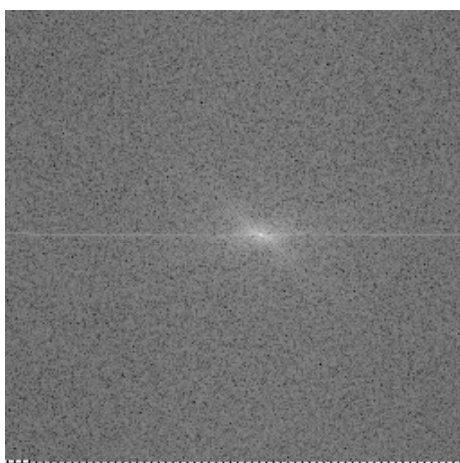


Fig. 4.2: Spectrul de frecvență al imaginii “lena” degradată de zgomot impulsiv.

Prelucrarea imaginilor (fie ele scalare sau vectoriale) presupune executarea, prin intermediul unui bloc funcțional ce poate fi presupus inițial de tip “cutie neagră”, a unei operații de tip *Image In, Image Out* [2] (spre deosebire de tehnicile de analiză ce pot fi caracterizate ca operații *Image In, Description Out*). Transformarea imaginii de intrare în imaginea rezultat de ieșire (ce prezintă aceleași caracteristici majore, simbolice și perceptuale<sup>3</sup>) se numește în mod generic filtrare<sup>4</sup>. Funcția unui filtru este de a transforma un semnal dat într-un alt semnal, mai potrivit unei anume aplicații date.

Definițiile filtrării (sau ale dispozitivului care o realizează, filtrul) ce se regăsesc în dicționarele lingvistice de uz general fac apel la noțiunea de semnal electric și de frecvențe asociate componentelor acestuia. Descrierea acțiunii filtrului este descrierea unei operații liniare efectuate în domeniul frecvențelor. Extinderea acestor definiții la cazul imaginilor presupune în primul rând definirea spectrului de frecvență asociat unui semnal cu suport [spațial] multidimensional și, implicit, introducerea noțiunii de frecvență spațială. Utilizarea unei prelucrări în domeniul frecvențelor presupune apoi identificarea benzilor de frecvență corepunzătoare entităților ce se doresc păstrate, respectiv eliminate, asociere ce nu este întotdeauna simplă. În practică, aceasta înseamnă determinarea unui spectru de frecvențe transformat, care apoi este transformat prin transformata Fourier inversă în imaginea filtrată – aceasta este filtrarea în domeniul frecvență.

Filtrarea liniară a imaginilor se bazează pe convoluția [circulară] între imaginea de prelucrat și un nucleu de filtrare (a se vedea capitolul 3); teoria transformatelor unitare [9] arată că o asemenea operație este echivalentă unui produs între spectrul Fourier al imaginii și spectrul Fourier al nucleului de filtrare; aceasta este teorema convoluției. În cazul unidimensional, dacă secvențele  $\mathbf{x}$  și  $\mathbf{y}$  au aceeași lungime, și  $\otimes$  este operatorul de convoluție circulară (definit în (4.23)) atunci:

$$\text{Fourier}(\mathbf{x} \otimes \mathbf{y}) = \text{Fourier}(\mathbf{x}) \text{Fourier}(\mathbf{y}) \quad (4.22)$$

$$\mathbf{x} \otimes \mathbf{y}(n) = \sum_{k=0}^{N-1} x((n-k) \bmod N) y(k), n = \overline{0, N-1} \quad (4.23)$$

Teorema convoluției oferă deci instrumentul prin care se pot echivala operațiile de filtrare realizate în domeniile spațial (filtrarea liniară, prezentată în capitolul 3) și de frecvență. Aceasta înseamnă că un nucleu mic de convoluție (mască de filtrare) este bordat cu zerouri până la obținerea dimensiunii imaginii (teorema convoluției circulare se referă la

---

<sup>3</sup>Prin prisma acestei definiții, transformările unitare ale imaginilor (deci reprezentarea într-un spațiu spectral) nu sunt filtrări; spectrul unei imagini, deși este echivalent acesteia din punct de vedere informațional, nu are aceleași caracteristici perceptuale.

<sup>4</sup>Filtrul este un sistem de circuite electrice, sonore, etc. cu care se selectează dintr-un complex de oscilații cu frecvențe diferite, oscilațiile cu frecvențele cuprinse între anumite limite. (Dicționarul Explicativ al Limbii Române, 1995)

Filtrul este un dispozitiv ce amortizează selectiv oscilațiile cu anumite frecvențe și nu afectează oscilațiile având alte frecvențe. (Webster Encyclopedic Unabridged Dictionary, 1995)

Filtrul este un dispozitiv destinat favorizării sau inhibării trecerii anumitor componente de frecvență a unui semnal electric. (Larousse, 1994)



secvențe de aceeași lungime) și apoi transformat Fourier; ceea ce se obține este răspunsul în frecvență al filtrului. Figura 4.3 prezintă răspunsul în frecvență a filtrului de mediere aritmetică realizat cu un nucleu pătrat, centrat de  $3 \times 3$ ; se remarcă comportamentul de tip filtru trece jos. Acest comportament este leagat de efectul de “blur” al filtrului de mediere, pus în evidență prin reducerea vizibilității conturilor imaginii, și deci înlăturarea frecvențelor înalte ce le sunt asociate.

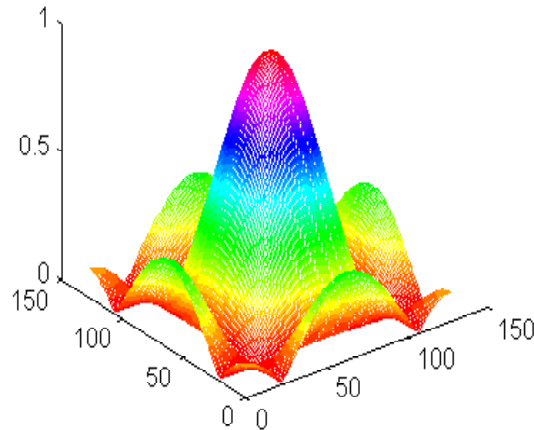


Fig. 4.3: Răspunsul în frecvență a unui nucleu  $3 \times 3$  de mediere aritmetică.

### 4.3.2 Transformata Fourier rapidă

Implementarea directă a relației de definiție a transformării Fourier discrete unidimensionale (4.17) necesită evident  $N^2$  înmulțiri complexe și  $N(N-1)$  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 inițială a fost divizarea în timp: cele  $N$  eșantioane ale secvenței se împart în două, după indicii pari și respectiv impari:

$$\begin{aligned}
 v(k) &= \sum_{n=0}^{N-1} u(n) \exp(-j \frac{2\pi}{N} kn) = \\
 &= \sum_{n=0}^{N/2-1} u(2n) \exp(-j \frac{2\pi}{N} k \cdot 2n) + \sum_{n=0}^{N/2-1} u(2n+1) \exp(-j \frac{2\pi}{N} k(2n+1)) = \\
 &= \sum_{n=0}^{N/2-1} u(2n) \exp(-j \frac{2\pi k}{N/2} 2n) \exp(j \frac{2\pi k}{N} 2n) + \exp(-j \frac{2\pi k}{N}) \sum_{n=0}^{N/2-1} u(2n+1) \exp(-j \frac{2\pi k}{N/2} 2n) =
 \end{aligned}$$

$$\begin{aligned}
&= \sum_{n=0}^{N/2-1} u_{par}(n) \exp(-j \frac{2\pi k}{N/2} n) + \exp(-j \frac{2\pi k}{N}) \sum_{n=0}^{N/2-1} u_{impar}(n) \exp(-j \frac{2\pi k}{N/2} n) = \\
&= v_{par}(k) + \exp(-j \frac{2\pi k}{N}) v_{impar}(k)
\end{aligned} \tag{4.24}$$

În (4.24) am notat cu  $\mathbf{u}_{par}$  secvența obținută din toate valorile pare ale secvenței inițiale, iar cu  $\mathbf{u}_{impar}$  secvența obținută din toate valorile impare ale secvenței inițiale. Această relație exprimă posibilitatea construirii transformării Fourier a secvenței  $\mathbf{u}$  de lungime  $N$  prin transformările Fourier ale unor jumătăți de secvență (de lungime  $N/2$ ) –  $\mathbf{v}_{par}$  și  $\mathbf{v}_{impar}$ .

Dacă notăm cu  $T(N)$  timpul de calcul al transformatei Fourier a secvenței 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 secvenței de lungime 1 este identică cu secvența); dezvoltând ecuația recurentă din (4.25) obținem:

$$\begin{aligned}
T(N) &= 2T\left(\frac{N}{2}\right) + N = 2\left(2T\left(\frac{N}{4}\right) + \frac{N}{2}\right) + N = 4T\left(\frac{N}{4}\right) + 2N = \\
&= \dots = 2^i T\left(\frac{N}{2^i}\right) + iN
\end{aligned} \tag{4.26}$$

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 obținut prin FFT față de implementarea clasică este deci proporțional cu  $N / \log_2 N$  (deci aproape un ordin de mărime pentru o lungime tipică de secvență  $N = 256$ ).

Relația de implementare de bază (4.24) mai poate fi scrisă și ca:

$$\begin{aligned}
v(k) &= v_{par}(k) + w_N^k v_{impar}(k), \quad k = 0, \dots, \overline{\frac{N}{2} - 1} \\
v(k) &= v_{par}(k) - w_N^k v_{impar}(k), \quad k = \overline{\frac{N}{2}, \dots, N - 1}
\end{aligned} \tag{4.27}$$

ceea ce semnifică faptul că două valori ale transformărilor Fourier ale secvențelor pe jumătate sunt combinate liniar pentru a genera două valori, simetrice față de centru, ale secvenței originale. Blocul ce realizează aceste prelucrări este numit celulă “*butterfly*”, caracterizat de o operație de înmulțire-adunare<sup>5</sup> (figura 4.4).

---

<sup>5</sup>Operația de înmulțire-adunare este una dintre operațiile de bază (cablate) din setul de instrucțiuni al procesoarelor de semnal (DSP).

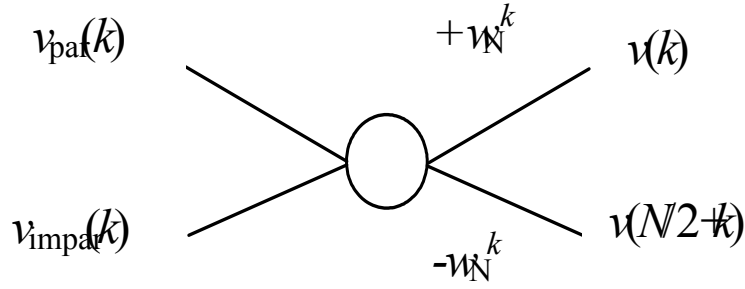


Fig. 4.4: Celula de bază butterfly

Construind etapele succesive de înjumătățire a lungimii secvenței, pentru o secvență de lungime  $N = 8$  se ajunge la schema din figura 4.5. Se remarcă reordonarea eșantioanelor secvenței de intrare după ordinea de bit inversă *bit-reverse* (forma binară a noului indice este forma binară a vechiului indice reflectată).

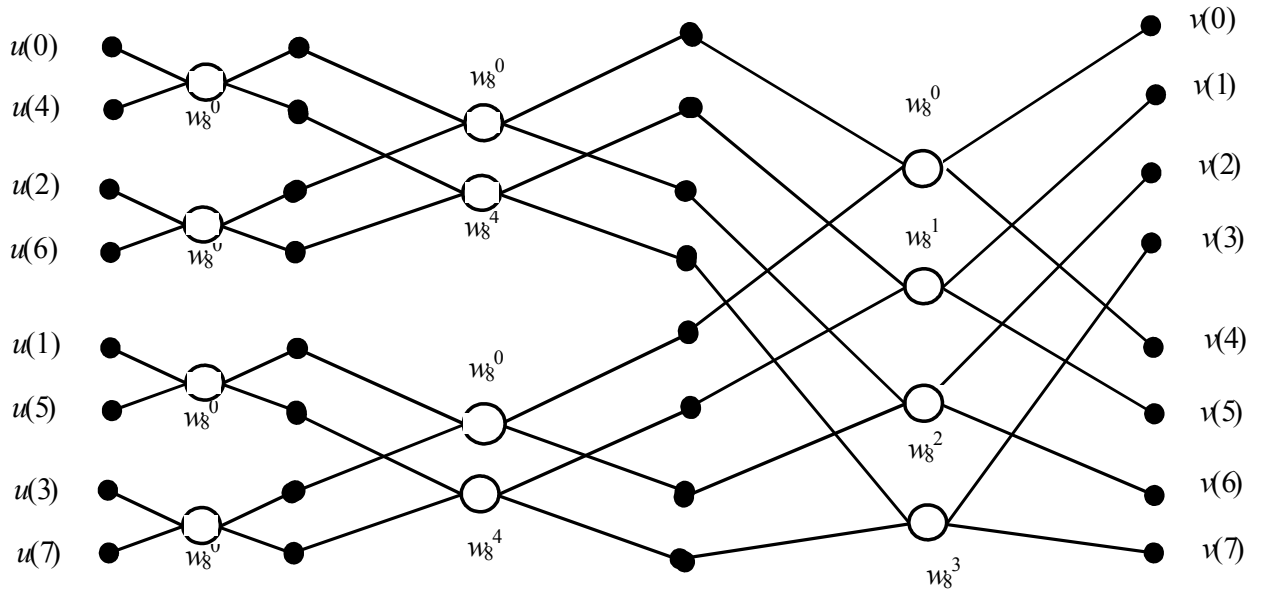


Fig. 4.5: Schema FFT pentru o secvență de lungime 8.

Implementările clasice ale FFT folosesc o reordonare *in-place* a secvenței de intrare (în același spațiu de memorie, prin interschimbări) pentru numere complexe (scrise în ordinea parte reală, parte imaginară). Același cod este folosit atât pentru transformata directă cât și pentru transformata inversă, doar prin modificarea unui parametru de semn ce intervine la calculul coeficienților Fourier  $w_N$ . Rezultatul (transformata) ia locul datelor

de intrare, în același spațiu 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 secvenței în ordinea inversă de bit. În partea a doua a codului se realizează calculul efectiv al celulelor *butterfly*, corespunzând transformărilor unor secvențe de dimensiune *mmax*, și deci preluării unor date situate la

distanța  $mmax$  între indici (vizibil și în schema din figura 4.5).

Transformata rapidă bidimensională implică realizarea câte unei transformări unidimensionale pentru fiecare linie și coloană a imaginii (conform principiului separabilității); 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; apariția acestora s-a datorat adaptării mai bune decât transformata Fourier la compactarea energiei și 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

$$C(k, n) = \frac{1}{\sqrt{N}} \begin{cases} 1, \text{ dacă } k = 0 \\ \sqrt{2} \cos \frac{(2n+1)\pi k}{2N}, \text{ dacă } k = \overline{1, N-1} \end{cases}, n = \overline{0, N-1} \quad (4.28)$$

Transformările directă și inversă a unei secvențe  $\mathbf{u}$  sunt definite ca:

$$v(k) = \alpha(k) \sum_{n=0}^{N-1} u(n) \cos \frac{(2n+1)\pi k}{2N}, k = \overline{0, N-1} \quad (4.29)$$

$$u(n) = \sum_{k=0}^{N-1} \alpha(k) v(k) \cos \frac{(2n+1)\pi k}{2N}, n = \overline{0, N-1} \quad (4.30)$$

unde

$$\alpha(k) = \frac{1}{\sqrt{N}} \begin{cases} 1, k = 0 \\ \sqrt{2}, k \neq 0 \end{cases} \quad (4.31)$$

Una dintre proprietățile importante ale transformatei cosinus se referă la legătura acesteia cu transformata Fourier: transformata cosinus a unei secvențe nu este partea reală a transformatei Fourier a aceleiași secvențe. Transformata cosinus se poate obține prin transformarea Fourier a unei secvențe rearanjate sau a unei secvențe dublate și simetrizate central. Fie  $N$  par; și definim

$$\begin{aligned} \tilde{u}_1(n) &= u(2n) \\ \tilde{u}_1(N - n - 1) &= u(2n + 1), n = \overline{0, \frac{N}{2} - 1} \end{aligned} \quad (4.32)$$

$$\tilde{u}_2(n) = \begin{cases} u(N - n - 1), & 0 \leq n < N \\ u(n - N), & N \leq n < 2N \end{cases} \quad (4.33)$$

Aceasta înseamnă că secvența  $\widetilde{\mathbf{u}}_1$  este  $u(0), u(2), u(4), \dots, u(N-2), u(N-1), u(n-3), \dots, u(3), u(1)$ , iar secvența  $\widetilde{\mathbf{u}}_2$  este  $u(N-1), u(N-2), \dots, u(1), u(0), u(0), u(1), \dots, u(N-2), u(n-1)$ . Atunci:

$$COS(\mathbf{u}) = \text{Re} \left[ \alpha(k) \exp(-\frac{j\pi k}{2N}) \text{Fourier}(\widetilde{\mathbf{u}}_1) \right] \quad (4.34)$$

$$COS(\mathbf{u}) = \text{Fourier}(\widetilde{\mathbf{u}}_2) \exp(-\frac{\pi k}{2N}), \quad k = \overline{0, \dots, N-1} \quad (4.35)$$

O consecință imediată a acestor relații 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 aceea că vectorii bazei cosinus (coloanele matricii  $\mathbf{C}$ ) sunt vectori proprii pentru orice matrice simetrică tridiagonală de forma:

$$\mathbf{Q} = \begin{pmatrix} 1-\rho & -\rho & 0 & \dots & \dots & 0 \\ -\rho & 1-\rho & -\rho & 0 & \dots & 0 \\ 0 & -\rho & 1-\rho & -\rho & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & -\rho & 1-\rho & -\rho \\ 0 & \dots & \dots & 0 & -\rho & 1-\rho \end{pmatrix} \quad (4.36)$$

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 transformata cosinus se comportă precum transformata Karhunen-Loeve pentru orice clasă de semnale a căror matrice de covarianță este de forma (4.36). Dar această formă este tipică secvențelor staționar Markov de ordinul 1, cu parametru de corelație mare ( $\rho \rightarrow 1$ ), 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 și la aplicațiile ei practice se găsesc în secțiunea 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

$$S(k, n) = \sqrt{\frac{2}{N+1}} \sin \frac{\pi(k+1)(n+1)}{N+1}, \quad k, n = \overline{0, N-1} \quad (4.37)$$

Transformările directă și inversă a unei secvențe  $\mathbf{u}$  sunt definite ca:

$$v(k) = \sqrt{\frac{2}{N+1}} \sum_{n=0}^{N-1} u(n) \sin \frac{\pi(k+1)(n+1)}{N+1}, \quad k = \overline{0, N-1} \quad (4.38)$$

$$u(n) = \sqrt{\frac{2}{N+1}} \sum_{k=0}^{N-1} v(k) \sin \frac{\pi(k+1)(n+1)}{N+1}, \quad n = \overline{0, N-1} \quad (4.39)$$

Una dintre proprietățile importante ale transformatei sinus se referă la legătura acesteia cu transformata Fourier: transformata sinus a unei secvențe nu este partea imaginară a transformatei Fourier a aceleiași secvențe. Transformata sinus se poate obține prin transformarea Fourier a extensiei antisimetrice a secvenței. Fie  $N$  par; și definim

$$\begin{aligned} \tilde{u}(0) &= \tilde{u}(N+1) = 0 \\ \tilde{u}(n) &= -u(N-n), \quad n = \overline{1, N} \\ \tilde{u}(N+2+n) &= u(n) \end{aligned} \quad (4.40)$$

Aceasta înseamnă că secvența  $\tilde{\mathbf{u}}$  este  $0, -u(N-1), -u(N-2), \dots, -u(1), -u(0), 0, u(0), u(1), \dots, u(N-1)$ . Atunci

$$SIN(\mathbf{u}) = \text{Im} [-j(-1)^k \text{Fourier}(\tilde{\mathbf{u}})] \quad (4.41)$$

O consecință imediată a acestei relații 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:

$$\mathbf{Q} = \begin{pmatrix} 1 & -\rho & 0 & \dots & 0 \\ -\rho & 1 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & 1 & -\rho \\ 0 & \dots & 0 & -\rho & 1 \end{pmatrix} \quad (4.42)$$

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 transformata sinus se comportă precum transformata Karhunen-Loeve pentru orice clasă de semnale a căror matrice de covariație este de forma (4.42). Dar această formă este tipică secvențelor staționar Markov de ordinul 1, cu parametru de corelație  $\rho \in [-0.5; 0.5]$ .

## Capitolul 5

# FILTRAREA NELINIARĂ A IMAGINILOR

Operațiile de filtrare studiate până în prezent au fost caracterizate esențial (din punct de vedere matematic) de către principiul liniarității (sau superpoziției). Istoric, acestea au fost primele operații utilizate, având un suport teoretic solid și desigur, având mai multe caracteristici utile. O mare parte a aplicațiilor curente de prelucrare a imaginilor probabil că se pot rezolva suficient de precis folosind doar operații liniare. Limitările operațiilor liniare de filtrare apar însă în momentul în care se consideră zgomote ce nu au distribuție normală, și, mai mult, nu mai sunt aditive.

Exemplul cel mai simplu de astfel de zgomot este zgomotul impulsiv, caracterizat de impulsuri pozitive și 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) și foarte deschise (albe), rezultatul fiind ceea ce se numește zgomot de tip sare și piper (vezi figura 5.1). Asemenea erori apar în porțiunile de achiziție și transmisie ale sistemului de prelucrare și analiza imaginilor (din cauza erorilor în convertorul analog-digital, din cauza erorilor de înregistrare pe mediile magnetice de stocare sau de pe canalul de transmisie); este posibil ca asemenea valori să apară și ca urmare a unui zgomot aditiv caracterizat de o distribuție “cu coadă lungă” - deci care asigură probabilități nenule de apariție 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 asemenea 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 și 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 și compuneri neaditive ale zgomotului cu imaginea originală (multiplicativă, convolutivă) [9].

Este deci evident că se impune considerarea și a altor metode de filtrare, care nu mai respectă principiul superpoziției, și deci sunt neliniare. O clasă esențială de astfel de metode de filtrare sunt cele bazate pe ordonare - rank order filtering [12]. Ideea esențială este aceea că operația de ordonare plasează valorile aberante (extremale) la capetele șirului de valori (deci într-o zonă bine definită), permițând astfel identificarea și 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 același model de prelucrare al ferestrei glisante, întâlnit și la filtrările liniare în domeniul spațial). Valorile selectate de fereastra de filtrare sunt apoi ordonate crescător; dacă valorile selectate de o fereastră cu  $n$  poziții sunt  $\{x_1, x_2, \dots, x_n\}$ , atunci același set de valori, ordonate crescător este  $\{x_{(1)}, x_{(2)}, \dots, x_{(n)}\}$ . Valorile  $x_{(i)}$  se numesc statistici de ordine de ordinul  $i$  și au proprietatea

$$x_{(1)} \leq x_{(2)} \leq \dots \leq x_{(n)} \quad (5.1)$$

În mod evident statistica de ordinul 1,  $x_{(1)}$ , este valoarea minimă, iar statistica de ordinul  $n$ ,  $x_{(n)}$ , este valoarea maximă.

Ieșirea filtrului de ordine de ordin  $k$  este statistica de ordinul  $k$ , cu  $k \in [1; n]$ :

$$\text{rank}_k\{x_1, x_2, \dots, x_n\} = x_{(k)} \quad (5.2)$$

În acest fel, ieșirea unui filtru de ordine este una dintre valorile selectate de fereastra de filtrare, și deci nu se crează la ieșire valori noi (așa cum se întâmplă la filtrarea liniară); acesta este un avantaj atât în ceea ce privește păstrarea valorilor originale din imagine, cât și din punctul de vedere al simplificării calculelor (nu mai sunt necesare operații cu numere reale, ce trebuie apoi rotunjite sau truncheate la numere naturale).

Filtrele de ordine nu sunt liniare din cauza operației de ordonare (ce nu respectă principiul superpoziției); în general putem scrie:

$$\text{rank}_k\{x_1 + y_1, x_2 + y_2, \dots, x_n + y_n\} \neq \text{rank}_k\{x_1, x_2, \dots, x_n\} + \text{rank}_k\{y_1, y_2, \dots, y_n\} \quad (5.3)$$

Filtrele de ordine comută însă cu operațiile de modificare liniară a valorilor:

$$\text{rank}_k\{\alpha x_1 + \beta, \alpha x_2 + \beta, \dots, \alpha x_n + \beta\} = \alpha \text{rank}_k\{x_1, x_2, \dots, x_n\} + \beta \quad (5.4)$$

Din punct de vedere statistic, interesează funcțiile de distribuție ale ieșirii filtrului (funcție de densitate de probabilitate, funcție de repartiție) atunci când funcțiile de distribuție ale valorilor de intrare sunt presupuse cunoscute. Modelul folosit în general se bazează pe ipoteza de independență și distribuție identică a valorilor selectate de fereastra filtrului (ceea ce conduce, implicit, la ipoteza că valorile pixelilor imaginii sunt variabile aleatoare independente, identic distribuite<sup>1</sup>). Dacă notăm cu  $f(x)$  și  $F(x)$  funcțiile de densitate de probabilitate și respectiv repartiție 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 funcția de densitate de probabilitate a ieșirii filtrului de ordine,  $f_k(x)$ .

Fie  $t$  o valoare oarecare din domeniul de variație al variabilelor aleatoare; atunci probabilitatea evenimentului ca ieșirea filtrului de ordine să se găsească în intervalul  $[t; t + dt]$  este  $f_k(t)dt$ . Evenimentul de interes este deci

$$t \leq \text{rank}_k\{x_1, x_2, \dots, x_n\} \leq t + dt \quad (5.5)$$

adică

$$t \leq x_{(k)} \leq t + dt \quad (5.6)$$

Statisticile de ordine sunt obținute prin ordonarea crescătoare a valorilor  $x_i$ , ordonare crescătoare care poate fi interpretată ca o permutare oarecare a indicilor valorilor ( $\{1, 2, \dots, n\} \rightarrow \{i_1, i_2, \dots, i_n\}$ ). În acest caz, evenimentul de interes exprimat de (5.6) este realizat prin mai multe evenimente elementare de tipul:

$$x_{i_1}, x_{i_2}, \dots, x_{i_{k-1}} \leq t \leq x_{i_k} \leq t + dt \leq x_{i_{k+1}}, \dots, x_{i_n} \quad (5.7)$$

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

---

<sup>1</sup>Această ipoteză a mai fost făcută și pentru deducerea transformării de egalizare a histogramei imaginilor.

ca  $t + dt$  (deci  $(1 - F(t))^{n-k}$ ) și 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:

$$\Pr(t \leq x_{(k)} \leq t + dt) = nC_{n-1}^{k-1}F^{k-1}(t)(1 - F(t))^{n-k}f(t)dt \quad (5.8)$$

și deci funcția de densitate de probabilitate a ieșirii filtrului de ordine de ordin  $k$  este:

$$f_k(x) = nC_{n-1}^{k-1}F^{k-1}(x)(1 - F(x))^{n-k}f(x) \quad (5.9)$$

În [12] sunt prezentate diferite utilizări ale funcției de densitate de probabilitate a ieșirii filtrelor de ordine, mai ales pentru determinarea comportării asimptotice ( $n \rightarrow \infty$ ) a filtrelor.

Proprietățile deterministe ale filtrelor de ordine se referă la comportamentul acestora în zonele tipice ale unei imagini: porțiuni netede și contururi. Filtrele de ordine nu afectează contururile (zonele de pantă abruptă) și păstrează valoarea zonelor uniforme. Aceste comportări au fost extinse la investigarea clasei mai generale de semnale rădăcină - semnale ce nu sunt modificate prin filtrare.

Orice secvență  $\{x_i\}$  monotonă (crescătoare sau descrescătoare) este un semnal rădăcină pentru filtrele de ordine. În plus, filtrele de ordine comută cu orice funcție monotonă  $g$ :

$$\text{rank}_k(g(x_i)) = g(\text{rank}_k(x_i))$$

Pe lângă metodele analitice de construire a semnalelor rădăcină [12], în practică, semnalele rădăcină se obțin prin filtrarea repetată a unui semnal oarecare, până la obținerea invariantei. Să considerăm spre exemplu secvența 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 obține secvența  $x_1 = \{0, 1, 1, 1, 3, 2, 3, 3, 3, 2, 1, 1, 0\}$ , iar după a doua filtrare se obține secvența  $x_2 = \{0, 1, 1, 1, 2, 3, 3, 3, 3, 2, 1, 1, 0\}$ . Această secvență este invariantă la filtrările următoare și este deci un semnal rădăcină. Figura 5.4 prezintă secvențele  $x$ ,  $x_1$ ,  $x_2$ .

### 5.1.1 Filtrul median

Filtrul median este un filtru de ordine a cărui ieșire este statistica de ordine de ordin central a setului de valori selectate de fereastră de filtrare.

$$\text{median}\{x_1, x_2, \dots, x_n\} = \begin{cases} x_{(\frac{n+1}{2})} & \text{dacă } n \text{ este impar} \\ \frac{1}{2}x_{(\frac{n}{2})} + \frac{1}{2}x_{(\frac{n}{2}+1)} & \text{dacă } n \text{ este par} \end{cases} \quad (5.10)$$

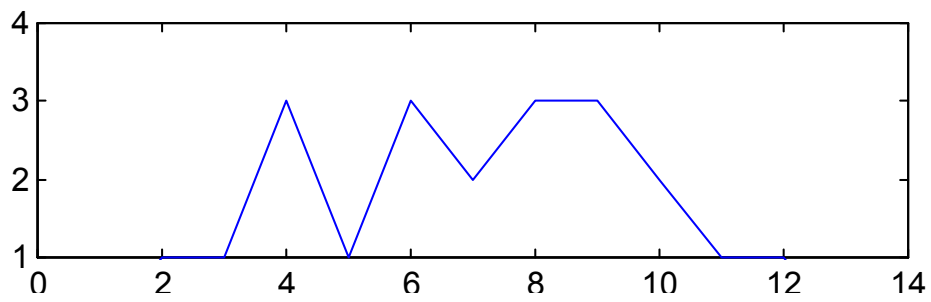


Fig. 5.2: Semnal inițial oarecare; prin filtrări de ordine repetate acesta se va transforma într-un semnal rădăcină.

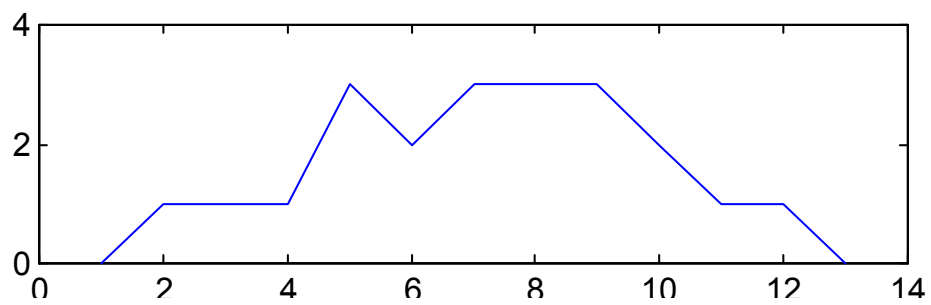


Fig. 5.3: Rezultatul primei filtrări a secvenței inițiale.

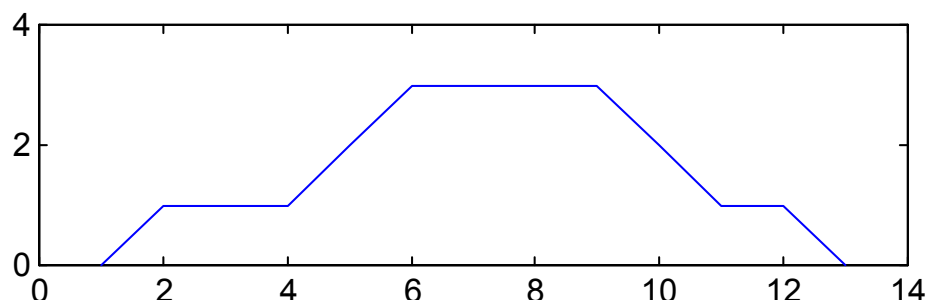


Fig. 5.4: Semnal rădăcină al filtrului de ordine considerat, obținut după două iterații ale filtrării.

Ieșirea filtrului median este deci valoarea din centrul secvenței ordonate; în cazul secvențelor (ferestrelor de filtrare) de dimensiune pară, această poziție centrală nu există, și atunci este definită ca media aritmetică a valorilor vecine centrului imaginar. În practică, filtrul median se folosește doar cu ferestre de filtrare de dimensiune impară, ceea ce conduce la posibilitatea de a scrie:

$$\text{median}\{x_1, x_2, \dots, x_n\} = \text{rank}_{\frac{n+1}{2}}\{x_1, x_2, \dots, x_n\} \quad (5.11)$$

Aceasta înseamnă că dacă fereastra de filtrare are 3 puncte, medianul este statistica de ordinul 2<sup>2</sup>, 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 și piper) sunt evidente; valorile punctelor de zgomot sunt 0 sau 255 și deci, după ordonare se vor afla la “capetele” șirului de valori; ieșirea filtrului median, fiind statistica de ordin central, este situată departe de valorile extreme. În imaginea filtrată median (figura 5.5) se observă totuși existența unor puncte albe și negre (puncte de zgomot) ce nu au putut fi eliminate prin filtrare; spunem că în acele poziții filtrul a fost străpuns de impulsuri (care deci au ajuns la ieșirea filtrului). Probabilitatea de străpungere a unui filtru median este dependentă de procentul de puncte de zgomot din imagine și 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 aceeași valoare.

O variantă a filtrului median cu fereastră de filtrare pătrată este filtrul median separabil. Separabilitatea semnifică (ca și în cazul transformărilor unitare) realizarea operației întâi

---

<sup>2</sup> Aceasta înseamnă deci că exemplul de construire a semnalului rădăcină prezentat anterior este realizat pentru un filtru median.

pe linii, și apoi pe coloanele imaginii, pe baza unor ferestre de filtrare liniare. Este evident că rezultatul acestei operații de filtrare nu coincide cu rezultatul filtrării mediane cu fereastră pătrată<sup>3</sup> (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, deși prezintă avantaje din punctul de vedere al timpului de calcul necesar, se străpunge mai ușor decât filtrul median cu fereastră pătrată din care a derivat.

### 5.1.2 Filtrele de ordine ponderate și structurile multietaj

Atât filtrele liniare cât și filtrele de ordine se bazează pe același principiu al ferestrei glisante. Prelucrările realizate asupra valorilor selectate de această fereastră de filtrare sunt evident diferite, dar se poate totuși remarca faptul că structura de filtrare liniară permite o flexibilitate mai mare (se pot realiza un număr infinit de filtre liniare pentru o aceeași fereastră de filtrare, prin varierea coeficienților de ponderare). Singurul factor de reglaj al filtrelor de ordine este ordinul statisticii selectate la ieșire<sup>4</sup>.

Este evident că ponderarea filtrelor de ordine nu se poate face prin multiplicarea valorilor selectate de fereastra de filtrare cu diferiți coeficienți, așa ca în cazul filtrelor liniare.

<sup>3</sup>Filtrul de minim (filtrul de ordine de ordin 1) și filtrul de maxim (filtrul de ordine de ordin  $n$ ) sunt singurele filtre de ordine separabile.

<sup>4</sup>Zamperoni a propus adaptarea ordinului statisticii folosite ca ieșire a filtrului de ordine prin  $k = \left\lceil \frac{1}{2} + \sum_{i=1}^n \frac{x_{(i)} - x_{(1)}}{x_{(n)} - x_{(1)}} \right\rceil$ .

Ponderarea are ca scop întărirea influenței anumitor pixeli din fereastra de filtrare (de exemplu pixelul central - curent prelucrat - care ar fi de așteptat ca să aibă o influență 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 influența unei anume valori este de a o repeta de mai multe ori, crescând astfel probabilitatea de a o regăsi ca statistica de ordin dorit. Așadar, ponderarea filtrelor de ordine se referă la repetarea de un număr fixat de ori a valorilor selectate de fereastra de filtrare. Mulțimea de valori ce rezultă se numește multiset. Ponderile  $w_i$  atașate fiecărei poziții de filtrare vor fi numere naturale nenule, ce exprimă numărul de repetiții al valorii corespunzătoare în multiset.

Spre exemplu să considerăm porțiunea 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 situații 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 rezultat 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 (și 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 pozițiile vecinilor imediați (orizontali și verticali) sunt repetate de câte 2 ori, iar valorile de pe pozițiile 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 conține  $\sum_{j=1}^n w_i$  termeni, și 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 toți coeficienții de ponderare (repetiție) sunt 1, cu excepția 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 și valoarea curentă), reprezentat în figura 5.8 și max/min-median (maxim sau minim din medianele pe verticală, orizontală și cele două diagonale ale ferestrei de filtrare pătrate centrate în punctul curent prelucrat), reprezentat în figura 5.7.

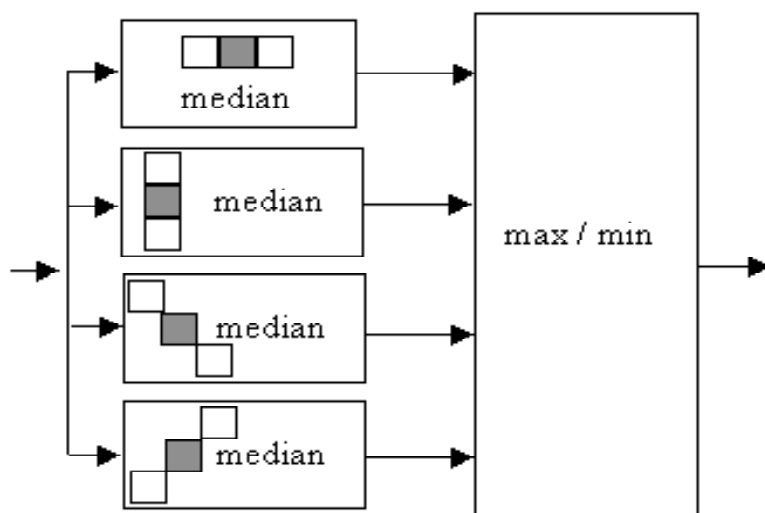


Fig. 5.7: Structură multietaj de tip max / min - median, bazată pe o fereastră de filtrare de dimensiune 3.

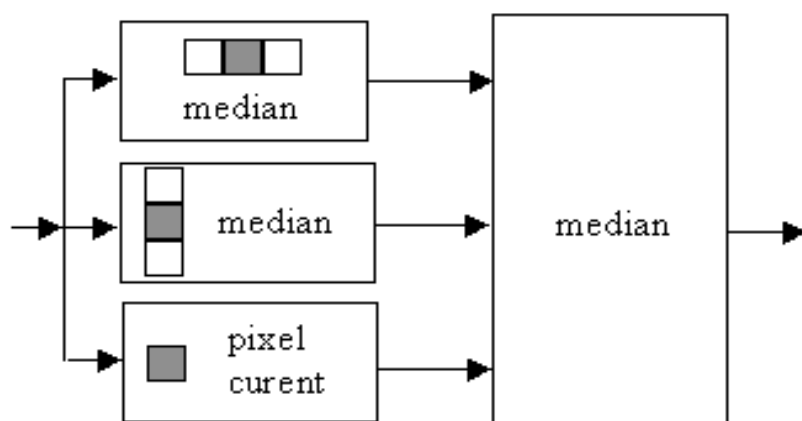


Fig. 5.8: Structură de filtrare de tip median - median, bazată pe o fereastră de filtrare pătrată de dimensiune 3.



## 5.2 Filtre de ordine de domeniu

Filtrele de ordine de domeniu (filtre LUM – *Lower-Upper-Middle* [18]) introduc o posibilitate de adaptare și reglare a filtrelor de ordine.

Valoarea  $y$  de ieșire a filtrului LUM de netezire este definită ca:

$$y = \begin{cases} x_{(k)}, & \text{dacă } x^* < x_{(k)} \\ x_{(n-k+1)}, & \text{dacă } x^* > x_{(n-k+1)} \\ x^*, & \text{în rest} \end{cases} \quad (5.12)$$

unde  $x^*$  este valoarea ce corespunde originii ferestrei de filtrare cu  $n$  poziții, 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 ieșire a filtrului LUM de conturare este definită ca:

$$y = \begin{cases} x_{(l)}, & \text{dacă } x_{(l)} < x^* < \frac{x_{(l)} + x_{(n-l+1)}}{2} \\ x_{(n-l+1)}, & \text{dacă } \frac{x_{(l)} + x_{(n-l+1)}}{2} < x^* < x_{(n-l+1)} \\ x^*, & \text{în rest} \end{cases} \quad (5.13)$$

unde  $x^*$  este valoarea ce corespunde originii ferestrei de filtrare cu  $n$  poziții, 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 dorința de a oferi o mai mare flexibilitate operațiilor de filtrare neliniară bazate pe ordonare; pasul firesc de dezvoltare a fost combinarea tuturor statisticilor de ordine în valoarea de la ieșirea filtrului. Ieșirea unui L-filtru este o combinație liniară a statisticilor de ordine:

$$L(\{x_1, x_2, \dots, x_n\}) = \sum_{i=1}^n w_i x_{(i)} \quad (5.14)$$

Este evident că prin varierea coeficienților  $w_i$  se pot obține diferite filtre, inclusiv filtrele de ordine (un singur coeficient egal cu 1, iar restul nuli). Coeficienții celor mai uzuale L-filtre sunt prezentați în tabelul 5.1

Filtru	Coeficienți
Filtru de ordine de ordin $k$ , $\text{rank}_k$	$a_k = 1, a_i = 0, i \neq k$
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 $\alpha$ -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: Coeficienții celor mai uzuale L-filtre.

Se poate demonstra cu ușurință că coeficienții L-filtrelor trebuie să îndeplinească aceleași condiții de normare ca și coeficienții filtrelor liniare: să aibă suma 1 pentru un filtru de netezire (conform (3.8)) și 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 uniformității 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 și piper)	filtru de ordine (median, maxim, minim)
gaussian	filtru de mediere aritmetică
impulsiv + gaussian	filtru de tip medie $\alpha$ -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 – diferența între valoarea maximă și minimă din fereastra de filtrare curentă:

$$\text{grad}\{x_1, x_2, \dots, x_n\} = x_{(n)} - x_{(1)}$$

Folosirea unui asemenea operator produce o imagine în care valoarea fiecărui pixel este invers proporțională cu gradul de uniformitate (netezime) al vecinătății acestuia (vezi figura 5.9).

## 5.4 Aspecte de implementare

Implementarea filtrelor neliniare de ordine sau a L-filtrelor urmează aceleași principii ca și oricare tehnică de tip fereastră glisantă (ca filtrarea liniară) în ceea ce privește



Fig. 5.9: Operatorul gradient morfologic aplicat imaginii filtrate din figura 5.5.

problemele legate de realizarea formei ferestrei de filtrare și efectele de margine. Ceea ce este particular filtrelor din această categorie este etapa de ordonare a valorilor selectate de fereastra de filtrare în fiecare poziție.

În teoria algoritmilor au fost dezvoltati numeroși 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 inserție) până la  $O(n)$  (pentru determinare doar a extremelor și 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 operații necesare, în condițiile în care dimensiunea caracteristică  $n$  a setului de date ce se prelucrează este foarte mare (la limită  $n \rightarrow \infty$ ). Sortările folosite în implementarea filtrelor neliniare de ordine prelucrează seturi mici de date (să nu uităm că rareori o fereastră de filtrare depășește dimensiunea de 25 de pixeli), și 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  $3 \times 3$ , centrată; ponderile filtrului sunt stocate în vectorul  $w$  (1 linie, 9 coloane, prima poziție corespunzând coeficientului statisticii minime). L-filtrul este aplicat imaginii  $in$  (de dimensiuni  $nrlin$  și  $nrcol$ ); efectele de margine sunt evitate prin neprelucrarea primei și 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 funcției standard *sort* ce realizează operația de ordonare crescătoare a valorilor; ieșirea L-filtrului este produsul scalar al vectorului de coeficienți ai filtrului cu vectorul statisticilor de ordine.

În cele ce urmează vom considera un exemplu de implementare a unui filtru cu ordonare după rang realizat cu o fereastră centrată de 3 x 3 pixeli, pentru filtrarea unei imagini cu *NRLIN* linii și *NRCOL* coloane; prima și ultima linie și coloană sunt identice cu cele din imaginea ce se filtrează (remarcați buclele *for* de alocare a spațiului pentru imaginea filtrată și inițializarea 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 poziții *temp*. Citirea acestor valori se face printr-un ciclu, bazându-se pe introducerea unei relații de echivalență între ordinea de baleiaj a ferestrei și poziția valorilor în vector (economia nu este de timp de execuție 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 poziție a vectorului ce conține valorile extrase de fereastra de filtrare este comparată cu poziția următoare, iar dacă ordinea crescătoare nu este respectată, cele două valori sunt interschimbate; indicatorul *sortat* indică efectuarea a măcar unei interschimbări, și 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 poziție a ferestrei de filtrare; o serioasă economie de timp se poate realiza dacă se ține 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 părți 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 și este echivalentă cu funcția de densitate de probabilitate a unei variabile aleatoare. Pentru aceeași variabilă aleatoare există însă și funcția de repartiție – primitiva funcției de densitate de probabilitate; așadar putem asocia oricărei histogramă  $h$  o histogramă cumulativă (5.15):

$$H(i) = \sum_{j=0}^i h(j), \text{ cu } i = 0, 1, \dots, L-1 \quad (5.15)$$

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  $h = (5 \ 7 \ 3 \ 2 \ 1 \ 2 \ 3 \ 2)$  iar histograma cumulativă este  $H = (5 \ 12 \ 15 \ 17 \ 18 \ 20 \ 23 \ 25)$ . 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  $j = 0, 1, 2, \dots, L - 1$  și  $H(-1) = 0$ .

Deci, pentru o statistică oarecare de ordin  $r$ , trebuie identificat numărul  $j$  astfel ca:

$$H(j - 1) + 1 \leq r \leq H(j) \quad (5.16)$$

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 proporțional cu numărul de nivele de gri din imagine (și deci lungimea vectorului histogramă sau histogramă cumulativă). Se poate însă observa că histograma într-o fereastră de filtrare va selecta relativ puține valori diferite, și atunci histograma va fi un vector rar (*sparse*), cu multe intrări nule, iar histograma cumulativă va avea, corespunzător, multe porțiuni constante. În aceste condiții, o reprezentare mai eficientă (atât ca memorie ocupată, cât și ca timp de căutare) este memorarea doar a tranzițiilor (poziție, valoare) din histograma cumulativă, adică doar memorarea histogramei.

## Capitolul 6

# ELEMENTE DE MORFOLOGIE MATEMATICĂ

În mod tradițional, prelucrarea semnalelor multidimensionale (și a imaginilor în particular) a fost bazată pe exploatarea conceptelor și teoriei sistemelor liniare și a transformatei Fourier ([9], [2]). Deși aceste abordări clasice sunt justificate și 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, după cum indică și numele (*morphos* – formă, *logos* – știință, deci știința formelor), realizează o abordare axată pe formă a prelucrării imaginilor. Folosită corespunzător, morfologia matematică conduce la prelucrări ce simplifică structura imaginii, păstrând caracteristicile esențiale de formă și eliminând irelevanțele.

Ideea de bază a oricărei prelucrări morfologice este considerarea imaginii ca un ansamblu (mulțime, reuniune de părți) asupra căruia se aplică transformări a căror esență este comparația cu mulțimi (ansambluri) mai simple, numite *elemente structurante*. Scopul acestor transformări este extragerea de forme mai simple din formele inițiale (complexe) ale imaginii.

Morfologia matematică este utilizată ca o abordare naturală a proceselor de vedere artificială, deoarece trăsăturile și respectiv identificarea obiectelor sunt corelate cu forma. Principalele aplicații sunt în domeniile roboticii, microscopiei electronice, imagisticii biomedicale, telemetriei, inspecției automate a produselor, analizei de scenă. Aplicațiile industriale sunt impulsionate și de continua dezvoltare și îmbunătățire 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] și reprezintă poate cea mai primară și 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 poziții; a recunoaște forma implică identificarea locală a părților sale componente și deci o operație simplă de potrivire de măști (*pattern matching*).

Rezultatul aplicării acestei transformări de identificare (numită uneori și transformarea “totul sau nimic”) este o mulțime ale cărei puncte satisfac criteriul de identificare a unei vecinătăți cu masca aplicată.

Transformarea *Hit or Miss* se definește pe baza unei partiții  $(B_1, B_2)$  a elementului structurant  $B$ :  $B = B_1 \cup B_2$  și  $B_1 \cap B_2 = \emptyset$  ca:

$$A \otimes B = \{\mathbf{x} | (B_1)_{\mathbf{x}} \subseteq A\} \cap \{\mathbf{x} | (B_2)_{\mathbf{x}} \subseteq A^C\} \quad (6.1)$$

În (6.1)  $A$  este mulțimea căreia i se aplică transformarea,  $A^C$  este complementara mulțimii  $A$ ,  $B$  este elementul structurant și  $(B_i)_{\mathbf{x}}$  este mulțimea  $B_i$  traslatată cu vectorul  $\mathbf{x}$ . Trebuie remarcat faptul că oricărui element structurant trebuie să i se atașeze o origine. Figura 6.1 prezintă o exemplificare a transformării *Hit or Miss*.

Se poate observa că această transformare produce ca rezultat o mulțime de puncte ce satisfac concomitent un set de condiții de tipul  $(B_i)_{\mathbf{x}} \subseteq A_i$ , unde  $\{A_i\}$  formează o partiție a spațiului.

Este evident că  $\{A, A^C\}$  formează o partiție, dar aceasta nu este singura posibilă. Anumite aplicații practice pot impune însă situații mai complexe, a căror rezolvare depășește cadrul morfologiei binare [14]. Astfel se pot cita așa numitele “cazuri”: cazul petrografic (provenit din analiza hidrocarburilor și a mineralelor), în care  $p$  componente împart exact spațiul (fără a lăsa locuri libere) și cazul geografic, în care  $p$  componente nu ocupă întreg spațiul și 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 construcției morfologiei matematice. Se va arăta însă că este posibilă exprimarea acesteia și în funcție de transformările morfologice fundamentale larg utilizate (erodarea și dilatarea).



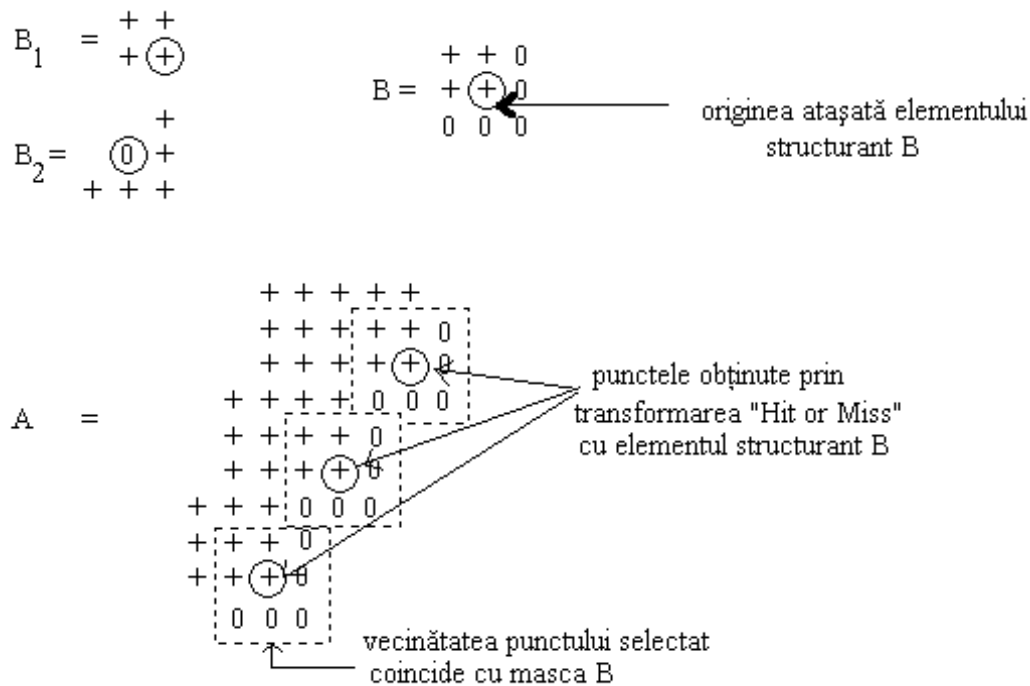


Fig. 6.1: Exemplificare a transformării de identificare a configurațiilor locale (Hit or Miss); în particular exemplul prezintă determinarea punctelor de tip “colț dreapta-jos”.

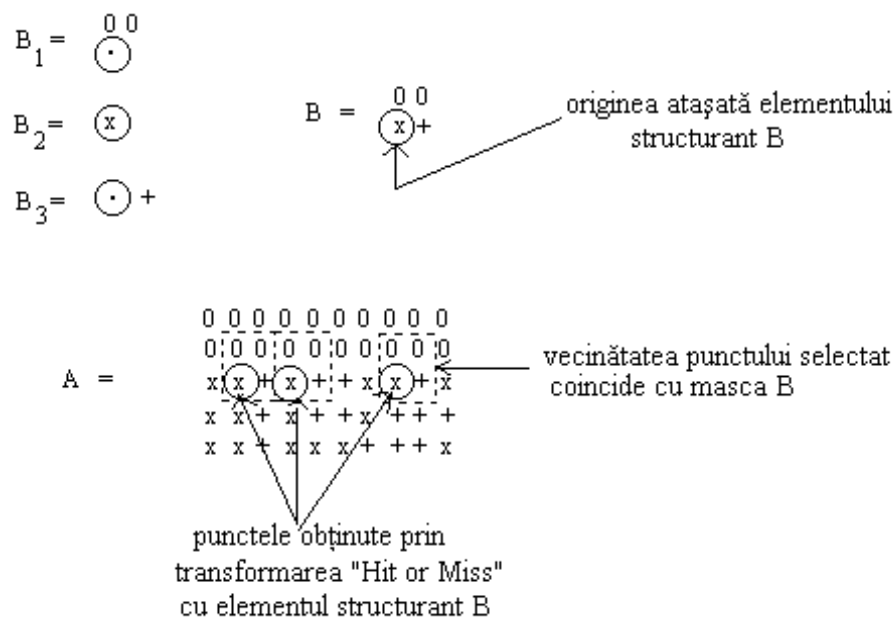


Fig. 6.2: Exemplu de transformare Hit or Miss extinsă pentru o partiție cu trei mulțimi a spațiului imaginii.

### 6.1.2 Erodarea morfologică

Erodarea morfologică a mulțimii  $A$  prin elementul structurant  $B$  se definește ca mulțimea punctelor (elementelor) cu care se poate transla elementul structurant astfel încât acesta să fie inclus în mulțimea de prelucrat  $A$ :

$$A \ominus B = \{\mathbf{x} | B_{\mathbf{x}} \subseteq A\} \quad (6.2)$$

Erodare morfologică a mulțimii  $A$  este transformata Hit or Miss a mulțimii 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 relație de definiție se mai poate exprima ca:

$$\begin{aligned} A \ominus B &= \{\mathbf{x} | B_{\mathbf{x}} \subseteq A\} = \{\mathbf{x} | \forall \mathbf{b} \in B, \exists \mathbf{a} \in A \text{ astfel încât } \mathbf{b} + \mathbf{x} = \mathbf{a}\} = \quad (6.3) \\ &= \{\mathbf{x} | \forall \mathbf{b} \in B, \exists \mathbf{a} \in A \text{ astfel încât } \mathbf{x} = \mathbf{a} - \mathbf{b}\} = \bigcap_{\mathbf{b} \in B} A_{-\mathbf{b}} = \bigcap_{\mathbf{b} \in B^S} A_{\mathbf{b}} \quad (6.4) \end{aligned}$$

Figura 6.3 prezintă rezultatul erodării unor mulțimi discrete, iar figura 6.4 prezintă rezultatul erodării unor mulțimi continue.

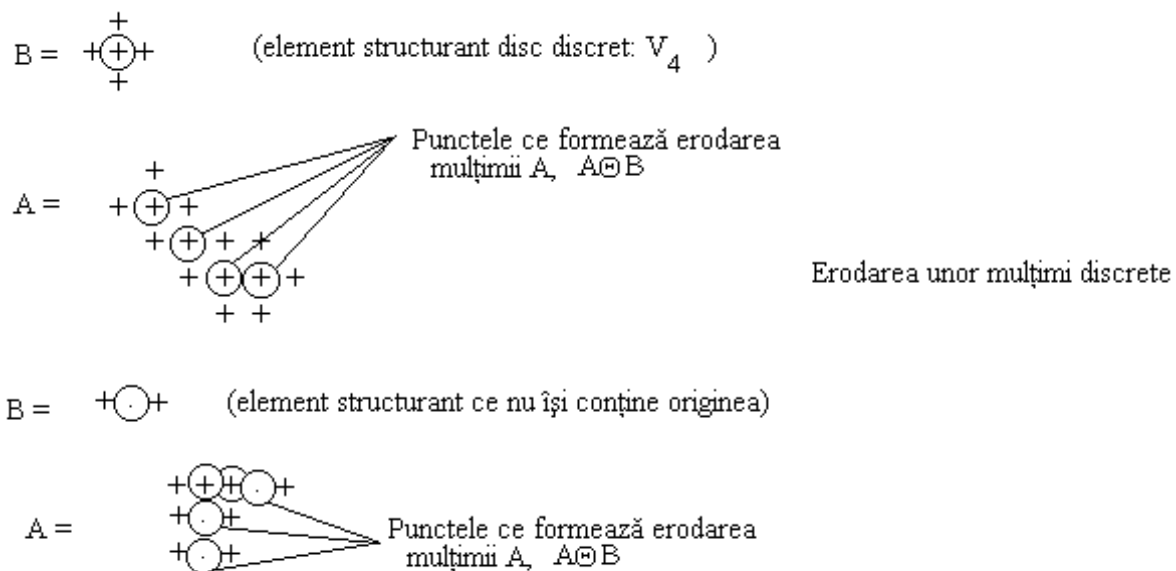


Fig. 6.3: Exemple de erodare a unor mulțimi discrete cu diferite elemente structurante, ce își conțin (sau nu) originea.

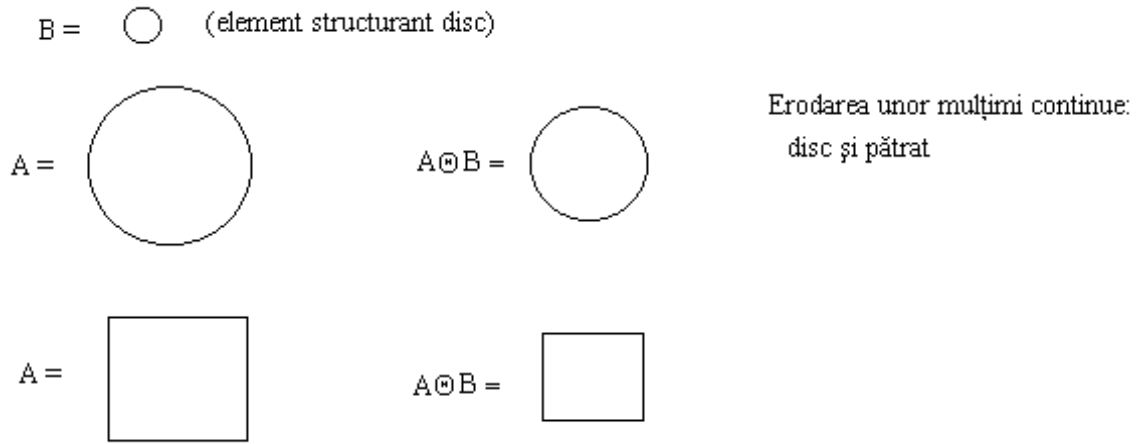


Fig. 6.4: Rezultatul erodării unor mulțimi continue cu un element structurant de tip disc (originea elementului structurant este centrul discului).

### 6.1.3 Dilatarea morfologică

Dilatarea morfologică a mulțimii  $A$  cu elementul structurant  $B$  se definește ca mulțimea punctelor (elementelor) cu care se poate transla elementul structurant astfel încât acesta să aibă puncte comune cu mulțimea de prelucrat  $A$ :

$$A \oplus B = \{\mathbf{x} | B_{\mathbf{x}} \cap A \neq \emptyset\} \quad (6.5)$$

Erodare morfologică a mulțimii  $A$  este complementara transformatei Hit or Miss a mulțimii cu un element structurant  $B = B_2$  ( $B_1 = \emptyset$ ):

$$(A \oplus B)^C = \{\mathbf{x} | (B_1)_{\mathbf{x}} \subseteq A\} \cap \{\mathbf{x} | (B_2)_{\mathbf{x}} \subseteq A^C\} = \{\mathbf{x} | (\emptyset)_{\mathbf{x}} \subseteq A\} \cap \{\mathbf{x} | (B_2)_{\mathbf{x}} \subseteq A^C\} = \{\mathbf{x} | (B)_{\mathbf{x}} \subseteq A^C\}$$

$$A \oplus B = \{\mathbf{x} | (B)_{\mathbf{x}} \subseteq A^C\}^C = \{\mathbf{x} | (B)_{\mathbf{x}} \not\subseteq A^C\} = \{\mathbf{x} | B_{\mathbf{x}} \cap A \neq \emptyset\}$$

Relația de definiție mai poate fi exprimată și ca:

$$A \oplus B = \{\mathbf{x} | B_{\mathbf{x}} \cap A \neq \emptyset\} = \{\mathbf{x} | \exists \mathbf{b} \in B, \exists \mathbf{a} \in A \text{ astfel încât } \mathbf{b} + \mathbf{x} = \mathbf{a}\} = \quad (6.6)$$

$$= \{\mathbf{x} | \exists \mathbf{b} \in B, \exists \mathbf{a} \in A \text{ astfel încât } \mathbf{x} = \mathbf{a} - \mathbf{b}\} = \bigcup_{\mathbf{b} \in B} A_{-\mathbf{b}} = \bigcup_{\mathbf{b} \in B^S} A_{\mathbf{b}} \quad (6.7)$$

Figura 6.5 prezintă rezultatul erodării unor mulțimi discrete, iar figura 6.6 prezintă rezultatul dilatării unor mulțimi continue.

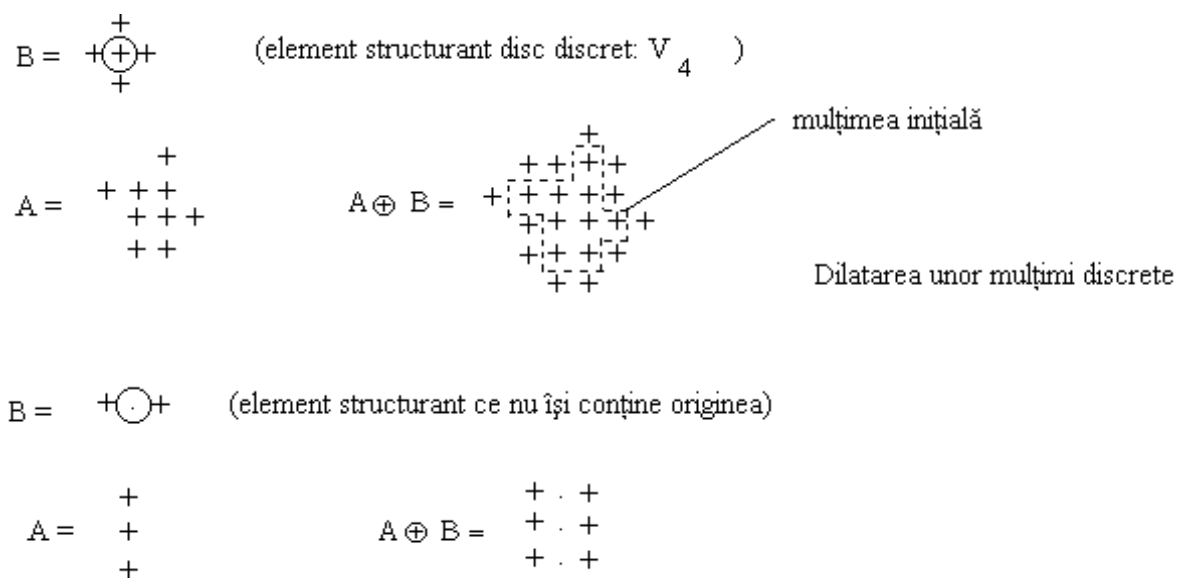


Fig. 6.5: Exemple de dilatare a unor mulțimi discrete cu diferite elemente structurante, ce își conțin (sau nu) originea.

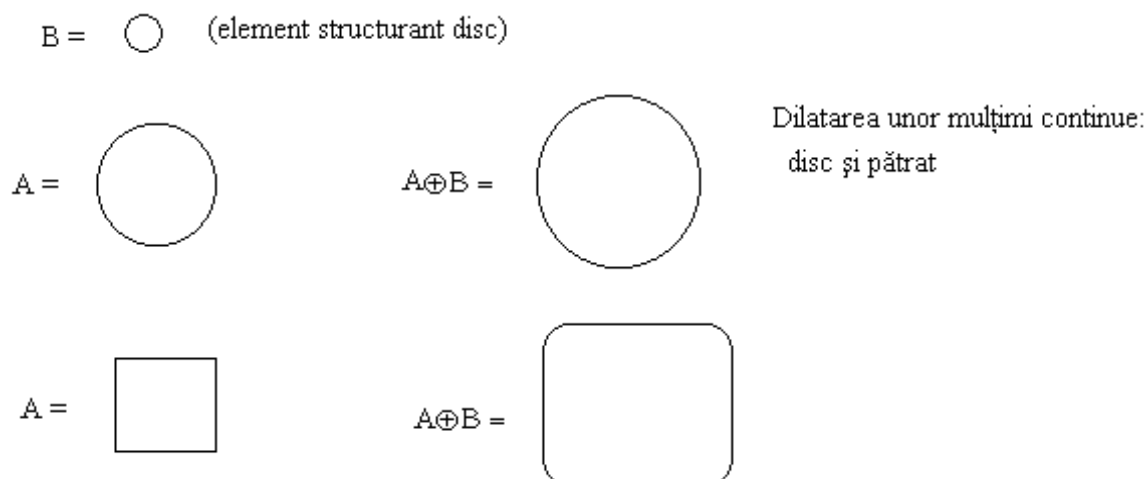


Fig. 6.6: Rezultatul dilatării unor mulțimi continue cu un element structurant de tip disc (originea elementului structurant este centrul discului).

### 6.1.4 Proprietățile erodării și dilatării

Se observă că, în general, efectul operației de dilatare este de a mări obiectul; acesta crește, “se umflă”, corespunzător formei și dimensiunii elementului structurant. Efectul erodării este, după cum am văzut, o micșorare 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 direcția elementului structurant și în sens contrar acestuia. Efectele erodării și dilatării vor fi discutate în continuare, pe baza proprietăților matematice ale acestora.

Dilatarea și erodarea formează o pereche de operații duale:

$$(A \oplus B)^C = A^C \ominus B \text{ și } (A \ominus B)^C = A^C \oplus B \quad (6.8)$$

Demonstrația este imediată:

$A \oplus B = \{\mathbf{x} | B_{\mathbf{x}} \cap A \neq \emptyset\}$ , de unde rezultă

$$(A^C \oplus B)^C = \{\mathbf{x} | B_{\mathbf{x}} \cap A^C \neq \emptyset\}^C = \{\mathbf{x} | B_{\mathbf{x}} \cap A^C = \emptyset\} = \{\mathbf{x} | B_{\mathbf{x}} \subseteq A\} = A \ominus B \quad (6.9)$$

și  $A \ominus B = \{\mathbf{x} | B_{\mathbf{x}} \subseteq A\}$  de unde rezultă

$$(A^C \ominus B)^C = \{\mathbf{x} | B_{\mathbf{x}} \subseteq A^C\}^C = \{\mathbf{x} | B_{\mathbf{x}} \not\subseteq A^C\} = \{\mathbf{x} | B_{\mathbf{x}} \cap A \neq \emptyset\} = A \oplus B \quad (6.10)$$

Interpretarea dualității ca obținerea acelorași efecte, dar asupra unor mulțimi complementare, este corectă: dacă dilatarea va mări obiectul (mulțimea), aceasta înseamnă că va micșora în același timp fundalul (complementara mulțimii), deci va fi echivalentă cu o erodare a fundalului. Dacă erodarea micșorează obiectul (mulțimea), aceasta înseamnă o mărire simultană a fundalului (complementara mulțimii), deci o dilatare a acestuia.

Un caz particular ce poate apare este acela al elementelor structurante ce nu conțin originea. Folosind un asemenea element structurant, rezultatele operațiilor morfologice vor fi “translatate” față de poziția la care ar fi fost de așteptat să apară.

Legătura între translația elementului structurant (faptul că acesta nu conține originea poate fi interpretat ca o deplasare) sau a mulțimii (obiectului) care se prelucrează și deplasarea rezultatului operațiilor morfologice este dat de proprietățile de invarianță la translație:

$$A_t \oplus B = (A \oplus B)_t \text{ și } A_t \ominus B = (A \ominus B)_t \quad (6.11)$$

$$A \oplus B_t = (A \oplus B)_{-t} \text{ și } A \ominus B_t = (A \ominus B)_{-t} \quad (6.12)$$

Aceste proprietăți de invarianță la translație (translația obiectului și translația elementului structurant) pot avea mai multe interpretări. În primul rând, invarianța la translație asigură faptul că forma rezultată prin dilatarea sau erodarea unei mulțimi este aceeași, indiferent de poziția mulțimii prelucrate. În al doilea rând, relațiile enunțate asigură suficiența considerării doar a elementelor structurante ce conțin originea; un element structurant ce nu conține originea poate fi obținut prin translație dintr-un element structurant ce conține originea; mulțimea (forma) rezultată în urma prelucrării va trebui translatată în sens opus elementului structurant.

În același grup de proprietăți de invarianță a operațiilor morfologice se încadrează și invarianța la scalare (omotetie). Dacă  $\lambda$  este parametrul nemul de scalare, atunci:

$$\frac{1}{\lambda}(\lambda A \oplus B) = A \oplus \frac{1}{\lambda}B \text{ și } \frac{1}{\lambda}(\lambda A \ominus B) = A \ominus \frac{1}{\lambda}B \quad (6.13)$$

Într-adevăr,

$$\begin{aligned} \frac{1}{\lambda}(\lambda A \oplus B) &= \frac{1}{\lambda}(\lambda A + B^S) = \frac{1}{\lambda}\{\mathbf{x} | \mathbf{x} = \lambda \mathbf{a} - \mathbf{b}, \mathbf{a} \in A, \mathbf{b} \in B\} = \{\mathbf{x} | \mathbf{x} = \mathbf{a} - \frac{1}{\lambda}\mathbf{b}, \mathbf{a} \in A, \mathbf{b} \in B\} = \\ &= A + \frac{1}{\lambda}B^S = A \oplus \frac{1}{\lambda}B \\ \frac{1}{\lambda}(\lambda A \ominus B) &= \frac{1}{\lambda}(\lambda A^C \oplus B)^C = (\frac{1}{\lambda}(\lambda A^C \oplus B))^C = (A^C \oplus \frac{1}{\lambda}B)^C = A \ominus \frac{1}{\lambda}B \end{aligned}$$

Relațiile de invarianță la scalare afirmă că rezultatul unei transformări morfologice aplicate unei versiuni scalate a formei  $A$  este identic cu aceeași scalare aplicată rezultatului transformării morfologice a formei  $A$  prin același element structurant scalat invers.

Proprietățile de monotonie a unei transformări nu pot fi neglijate la descrierea acestora. Atât dilatarea cât și erodarea sunt crescătoare față de mulțimea ce se prelucurează (forma): dacă  $A_1 \subseteq A_2$ , atunci

$$A_1 \oplus B \subseteq A_2 \oplus B \text{ și } A_1 \ominus B \subseteq A_2 \ominus B \quad (6.14)$$

Monotonia față de elementul structurant folosit este diferită: dilatarea este crescătoare, iar erodarea este descrescătoare: dacă  $B_1 \subseteq B_2$ , atunci:

$$A \oplus B_1 \subseteq A \oplus B_2 \text{ și } A \ominus B_2 \subseteq A \ominus B_1 \quad (6.15)$$

Aceste proprietăți subliniază corectitudinea interpretării dilatării ca o adăugare, ca o îngroșare, iar a erodării ca o subțiere; 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 micșora mai mult în urma erodării.

În general, dilatarea este extensivă, adică:

$$A \subseteq A \oplus B. \quad (6.16)$$

O condiție suficientă pentru ca această proprietate să fie adevărată este ca elementul structurant să conțină originea,  $\{0_n\} \in B$ . Atunci

$$A \oplus B = \bigcup_{\mathbf{b} \in B^S} A_{\mathbf{b}} = A \cup \left( \bigcup_{\mathbf{b} \in B^S - \{0_n\}} A_{\mathbf{b}} \right) \supseteq A \quad (6.17)$$

Condiția ca elementul structurant să conțină originea nu este însă și o condiție necesară (a se vedea figura 6.7).

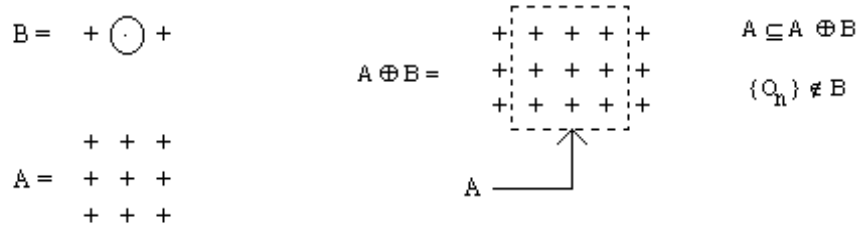


Fig. 6.7: Deși elementul structurant nu își conține originea, rezultatul dilatării include forma originală.

În general erodarea este antiextensivă, adică:

$$A \ominus B \subseteq A \quad (6.18)$$

O condiție suficientă pentru ca această proprietate să fie adevărată este ca elementul structurant să conțină originea,  $\{0_n\} \in B$ . Atunci

$$A \ominus B = \bigcap_{\mathbf{b} \in B^S} A_{\mathbf{b}} = A \cap \left( \bigcap_{\mathbf{b} \in B^S - \{0_n\}} A_{\mathbf{b}} \right) \subseteq A \quad (6.19)$$

Condiția ca elementul structurant să conțină originea nu este însă și o condiție necesară (a se vedea figura 6.8).

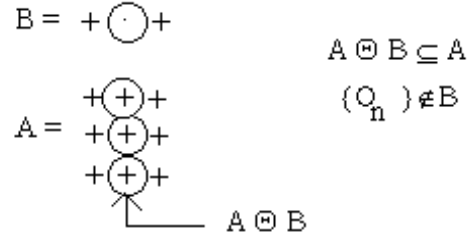


Fig. 6.8: Deși elementul structurant nu își conține originea, rezultatul erodării este inclus în mulțimea inițială.

Dilatarea este pseudocomutativă, proprietate care se exprimă ca

$$A \oplus B = (B \oplus A)^S \quad (6.20)$$

$$A \oplus B = A + B^S = (A^S + B)^S = (B + A^S) = (B \oplus A)^S \quad (6.21)$$

Proprietatea de asociativitate a dilatării se scrie ca

$$A \oplus (B \oplus C) = (A \oplus B) \oplus C^S \quad (6.22)$$

Într-adevăr:

$$A \oplus (B \oplus C) = A + (B \oplus C)^S = A + (B + C^S)^S = A + B^S + C = (A \oplus B) + C = (A \oplus B) \oplus C^S \quad (6.23)$$

Erodarea nu este nici comutativă (sau pseudocomutativă) și nici asociativă. Mai mult, “asociativitatea” erodării se scrie ca:

$$(A \ominus B) \ominus C = A \ominus (B \oplus C) \quad (6.24)$$

Într-adevăr:

$$(A \ominus B) \ominus C = ((A \ominus B)^C \oplus C)^C = ((A^C \oplus B) \oplus C)^C = (A^C \oplus (B \oplus C))^C = A \ominus (B \oplus C) \quad (6.25)$$

Proprietățile de asociativitate ale dilatării și erodării pot fi interpretate și ca o “descompunere” a elementului structurant; dacă un element structurant poate fi considerat ca descompus prin dilatare în  $X = B \oplus C$ , atunci operația morfologică efectuată prin elementul structurant respectiv este echivalentă cu operațiile morfologice prin “părțile” descompunerii elementului structurant, efectuate succesiv, adică:



$$A \oplus X = (A \oplus B) \oplus C^S \text{ și } A \ominus X = (A \ominus B) \ominus C$$

Dilatarea și erodarea sunt transformări (operații) ce nu păstrează numărul de conexități (numărul de obiecte). Rezultatul unei erodări poate fi o mulțime vidă (dacă elementul structurant nu poate fi inclus în formă pentru nici o poziție) sau mai multe forme (componente conexe). Rezultatul dilatării unei mulțimi formate din mai multe componente conexe poate fi o singură componentă conexă; “găurile” conținute într-o formă pot fi umplute.

Aceste proprietăți ale erodării și dilatării sunt fundamentul folosirii practice a acestora: prin erodare se pot elimina dintr-o mulțime componentele conexe ce sunt mai mici decât elementul structurant folosit (sau sunt mult diferite de acesta din punctul de vedere al formei) - aplicațiile ce ar putea folosi o asemenea comportare sunt separarea obiectelor după formă și dimensiune și eliminarea zgomotului suprapus scenei. Prin dilatare se pot grupa într-o singură entitate obiecte apropiate și se pot umple golurile înglobate în obiecte.

Dilatarea și erodarea nu sunt operații inversabile (nu admit o transformare inversă). Cu atât mai mult, dilatarea nu este inversa erodării și erodarea nu este inversa dilatării; exemplele prezentate în figurile 6.9 și 6.10 ilustrează această afirmație.

$$(A \ominus B) \oplus B \neq A \text{ și } (A \oplus B) \ominus B \neq A \quad (6.26)$$

O altă categorie de proprietăți se referă la distributivitatea operațiilor morfologice față de operațiile cu mulțimi. Pentru dilatare avem:

$$(A \cup B) \oplus C = (A \oplus C) \cup (B \oplus C) \quad (6.27)$$

$$(A \oplus C) \cup (B \oplus C) = \left( \bigcup_{\mathbf{c} \in C^S} A_{\mathbf{c}} \right) \cup \left( \bigcup_{\mathbf{c} \in C^S} B_{\mathbf{c}} \right) = \left( \bigcup_{\mathbf{c} \in C^S} A_{\mathbf{c}} \cup B_{\mathbf{c}} \right) = \bigcup_{\mathbf{c} \in C^S} (A \cup B)_{\mathbf{c}} = (A \cup B) \oplus C$$

$$A \oplus (B \cup C) = (A \oplus B) \cup (A \oplus C) \quad (6.28)$$

$$\begin{aligned} (A \oplus B) \cup (A \oplus C) &= (A + B^S) \cup (A + C^S) = (B^S + A) \cup (C^S + A) = \left( \bigcup_{\mathbf{a} \in A} B_{\mathbf{a}}^S \right) \cup \left( \bigcup_{\mathbf{a} \in A} C_{\mathbf{a}}^S \right) = \\ &= \bigcup_{\mathbf{a} \in A} (B_{\mathbf{a}}^S \cup C_{\mathbf{a}}^S) = \bigcup_{\mathbf{a} \in A} (B \cup C)_{\mathbf{a}}^S = (B \cup C)^S + A = A + (B \cup C)^S = A \oplus (B \cup C) \end{aligned}$$

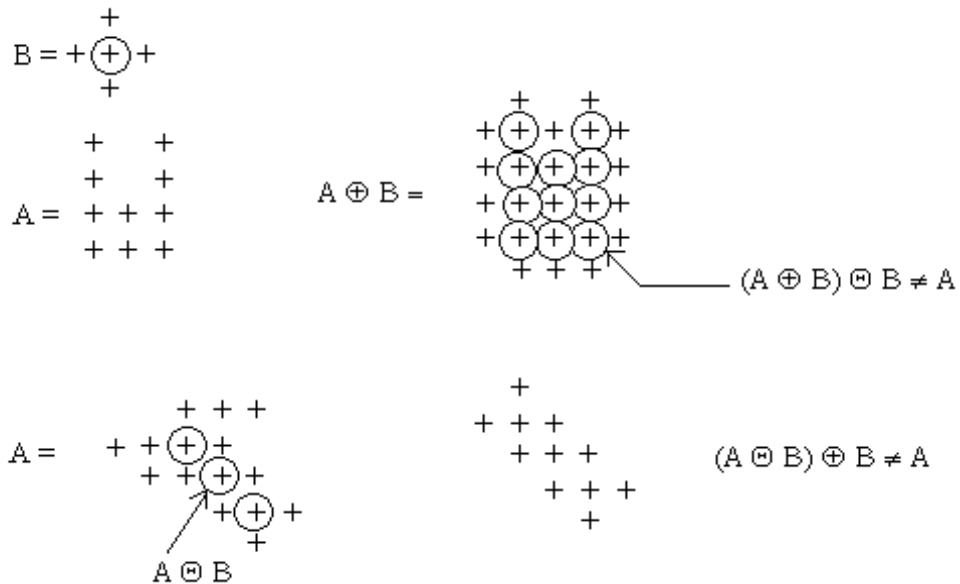


Fig. 6.9: Dilatarea și erodarea nu sunt transformări inverse una alteia; ilustrare pentru mulțimi discrete.

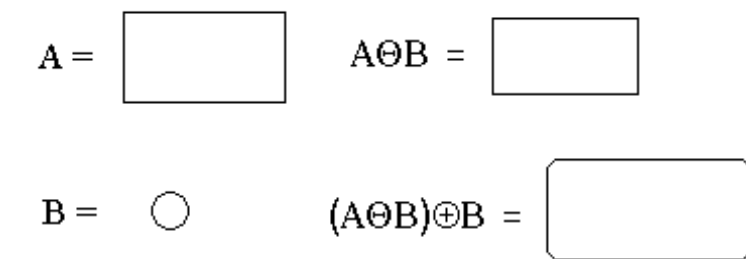


Fig. 6.10: Dilatarea și erodarea nu sunt transformări inverse una alteia; ilustrare pentru mulțimi continue.

Această din urmă relație poate fi interpretată și ca o descompunere prin reuniune a elementului structurant  $X = B \cup C$ , caz în care operația morfologică efectuată prin elementul structurant respectiv este echivalentă cu operațiile morfologice prin “părțile” descompunerii elementului structurant, efectuate succesiv, adică:

$$A \oplus X = (A \oplus B) \cup (A \oplus C) \quad (6.29)$$

Erodarea are proprietăți asemănătoare dilatării față de intersecția mulțimilor:

$$A \ominus (B \cup C) = (A \ominus B) \cap (A \ominus C) \quad (6.30)$$

$$(A \ominus B) \cap (A \ominus C) = \left( \bigcap_{\mathbf{t} \in B^S} A_{\mathbf{t}} \right) \cap \left( \bigcap_{\mathbf{t} \in C^S} A_{\mathbf{t}} \right) = \bigcap_{\mathbf{t} \in B^S \cup C^S} A_{\mathbf{t}} = \bigcap_{\mathbf{t} \in (B \cup C)^S} A_{\mathbf{t}} = A \ominus (B \cup C)$$

$$(A \cap B) \ominus C = (A \ominus C) \cap (B \ominus C)$$

$$(A \ominus C) \cap (B \ominus C) = \left( \bigcap_{\mathbf{c} \in C^S} A_{\mathbf{c}} \right) \cap \left( \bigcap_{\mathbf{c} \in C^S} B_{\mathbf{c}} \right) = \bigcap_{\mathbf{c} \in C^S} (A_{\mathbf{c}} \cap B_{\mathbf{c}}) = \bigcap_{\mathbf{c} \in C^S} (A \cap B)_{\mathbf{c}} = (A \cap B) \ominus C$$

Înrudite cu proprietățile de distributivitate deja enunțate, există o serie de inegalități (relații de incluziune) de aceeași natură:

$$A \oplus (B \cap C) \subseteq (A \oplus B) \cap (A \oplus C) \text{ și } (B \cap C) \oplus A \subseteq (B \oplus A) \cap (C \oplus A)$$

$$A \ominus (B \cap C) \supseteq (A \ominus B) \cup (A \ominus C) \text{ și } (B \cap C) \ominus A \supseteq (B \ominus A) \cap (C \ominus A)$$

Această tratare [aproape] exhaustivă a proprietăților operațiilor morfologice de bază (dilatate, erodare) se regăsește în majoritatea monografiilor de morfologie matematică, ca de exemplu în [14], [7]. Deși, în esență, proprietățile tratate sunt aceleași, se impun câteva precizări.

Notățiile introduse în acest text sunt diferite pentru operațiile Minkowski (adunare  $+$  și scădere  $-$ ) și pentru operațiile morfologice (dilatate  $\oplus$  și erodare  $\ominus$ ). Legătura dintre aceste operații este dată de:

$$A \oplus B = A + B^S \text{ și } A \ominus B = A - B^S \quad (6.31)$$

În majoritatea lucrărilor de morfologie însă, nu se face nici o distincție de notație între aceste operații, ceea ce poate genera confuzii. Chiar în condițiile păstrării aceluiași notații, există abordări diferite privind definirea operațiilor morfologice. În [7] dilatarea și erodarea sunt identice cu adunarea, respectiv scăderea Minkowski (deci fără simetrizarea elementului structurant). Avantajul unei asemenea formalizări este existența proprietăților de comutativitate și asociativitate a dilatării (determinate direct de comutativitatea și asociativitatea adunării Minkowski). În plus, rezultatul dilatării este o “creștere” a formei pe direcția și în sensul elementului structurant folosit.

Dezavantajul acestei abordări este acela că trebuie redefinit conceptul de dualitate 6.8 (dacă se dorește păstrarea proprietății fundamentale de dualitate a dilatării și erodării), după cum urmează:

$$Dual(Op(A, B))|_{B \text{ parametru}} = (Op(A^C, B^S))^C \quad (6.32)$$

În [14] este preferată definirea dilatării unei mulțimi ca adunarea Minkowski a mulțimii 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 operație 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). Convenția de reprezentare a imaginilor binare este de a avea asociate valori nule punctelor de fundal și valori pozitive (1 sau 255) punctelor obiect<sup>1</sup>. În aceste condiții, putem găsi o echivalență intuitivă simplă operațiilor morfologice de erodare și dilatare. Rezultatul operației de erodare într-un pixel este nenul dacă și numai dacă elementul structurant plasat cu originea în acel punct este inclus în forma de prelucrat, și deci, dacă și numai dacă toate valorile selectate de elementul structurant sunt nule; aceasta înseamnă că putem implementa operația de erodare printr-o operație de minim. Rezultatul operației de dilatare într-un pixel este nul dacă și numai dacă elementul structurant plasat cu originea în acel punct nu are nici un punct comun cu forma de prelucrat, și deci, dacă și numai dacă toate valorile selectate de elementul structurant sunt nule; aceasta înseamnă că putem implementa operația de dilatare printr-o operație de maxim. În cele ce urmează prezentăm codul pentru operația de erodare.

```
# define SIZE_MAX_ES=16;
```

---

<sup>1</sup>Reprezentarea grafică a acestei imagini, realizată cu tabela de nivele de gri normală va afișa 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 și 2 coloane. Fiecare linie a matricii corespunde unui punct din elementul structurant, reprezentat prin coordonatele față de originea elementului structurant:  $es[k][1]$  și  $es[k][2]$ . 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 operație (transformare) morfologică derivată operația morfologică construită ca o combinație de transformări de bază: erodări, dilatări și operații clasice ansambliste.

### 6.2.1 Deschiderea și închiderea

După cum s-a arătat, erodarea și dilatarea nu sunt transformări inverse una alteia (deci alternarea lor va produce un rezultat diferit de mulțimea originală ce a fost prelucrată). Această observație conduce la ideea utilizării unor iterații ale operațiilor morfologice de

bază, obținând astfel operații mai complexe, ale căror proprietăți le fac mai adecvate utilizării în scopuri practice.

Deschiderea morfologică a mulțimii  $A$  prin elementul structurant  $B$  se definește ca erodarea mulțimii cu elementul structurant respectiv, urmată de dilatarea cu elementul structurant simetrizat:

$$A \circ B = (A \ominus B) \oplus B^S \quad (6.33)$$

Închiderea morfologică a mulțimii  $A$  prin elementul structurant  $B$  se definește ca dilatarea mulțimii cu elementul structurant respectiv, urmată de erodarea cu elementul structurant simetrizat:

$$A \bullet B = (A \oplus B) \ominus B^S \quad (6.34)$$

Proprietatea de bază a deschiderii și închiderii morfologice este aceea că sunt transformări duale una alteia (proprietate ce derivă din dualitatea blocurilor constitutive de bază, dilatarea și erodarea). Această proprietate permite interpretarea rezultatelor unei operații și deducerea proprietăților acesteia pe baza caracteristicilor dualei sale.

$$(A \circ B)^C = A^C \bullet B \text{ și } (A \bullet B)^C = A^C \circ B \quad (6.35)$$

Demonstrația acestor proprietăți este imediată:

$$\begin{aligned} (A^C \circ B)^C &= ((A^C \ominus B) \oplus B^S)^C = ((A \oplus B)^C \oplus B^S)^C = (A \oplus B) \ominus B^S = A \bullet B \\ (A^C \bullet B)^C &= ((A^C \oplus B) \ominus B^S)^C = ((A \ominus B)^C \ominus B^S)^C = (A \ominus B) \oplus B^S = A \circ B \end{aligned}$$

În mod evident, rezultatul unei deschideri sau al unei închideri este diferit de mulțimea ce a fost prelucrată. Relația dintre rezultatul prelucrării și mulțimea inițială este dată de proprietățile de extensivitate ale transformărilor.

Deschiderea morfologică este antiextensivă, adică  $A \circ B \subseteq A$

Închiderea morfologică este extensivă, adică  $A \subseteq A \bullet B$

Deci, pentru a sintetiza, putem afirma că:

$$A \circ B \subseteq A \subseteq A \bullet B \quad (6.36)$$

Relațiile pot fi interpretate ca o modificare sigură a mulțimii: prin deschidere se vor elimina o parte dintre elementele mulțimii ce se prelucrează, iar prin închidere se adaugă elemente noi mulțimii.

Proprietatea de idempotență a operațiilor introduce o limitare a modificărilor: mulțimea obținută după o deschidere sau închidere este invariantă la repetarea operației:

$$(A \bullet B) \bullet B = A \bullet B \text{ și } (A \circ B) \circ B = A \circ B \quad (6.37)$$

Relațiile se demonstrează folosind proprietățile deja enunțate ale deschiderii și închiderii (extensivitate) și proprietățile de monotonie ale operațiilor morfologice de bază.

Închiderea și deschiderea moștenesc o parte dintre proprietățile operațiilor morfologice de bază: invarianța la translație și la scalare, monotonia față de mulțimea prelucrată și față de elementul structurant folosit. Din punctul de vedere al acestor proprietăți, deschiderea se comportă analog erodării iar închiderea are o comportare analoagă dilatării.

$$A_t \circ B = (A \circ B)_t \text{ și } A_t \bullet B = (A \bullet B)_t$$

$$\frac{1}{\lambda}(\lambda A \circ B) = A \circ \frac{1}{\lambda}B \text{ și } \frac{1}{\lambda}(\lambda A \bullet B) = A \bullet \frac{1}{\lambda}B$$

$$\text{Dacă } A_1 \subseteq A_2 : A_1 \circ B \subseteq A_2 \circ B \text{ și } A_1 \bullet B \subseteq A_2 \bullet B.$$

$$\text{Dacă } B_1 \subseteq B_2 : A \circ B_2 \subseteq A \circ B_1 \text{ și } A \bullet B_1 \subseteq A \bullet B_2$$

În ceea ce privește proprietățile legate de comportarea față de translație a operatorilor de deschidere și închidere, merită subliniat faptul că proprietatea este identică cu cea a erodării și dilatării doar la translația mulțimii prelucrate (rezultatul unei deschideri sau închideri a unei mulțimi este același, indiferent de poziția spațială a mulțimii). În cazul translatării elementului structurant, rezultatul operației este același, invariant la translație (ca rezultat a iterării erodării și dilatării cu elemente structurante simetrice).

Pentru realizarea efectivă a operațiilor de deschidere și închidere este importantă exprimarea acestora ca operații la nivelul elementelor mulțimilor ce se prelucrează (și nu ca o iterare de operații morfologice de bază). Deschiderea mai poate fi exprimată și ca mulțimea elementelor structurante translatate ce sunt incluse în mulțimea de prelucrat:

$$A \circ B = \bigcup_{B_x \subseteq A} B_x \quad (6.38)$$

Închiderea mai poate fi exprimată și ca mulțimea punctelor pentru care toate elementele structurante translatate ce le conțin au puncte comune cu mulțimea de prelucrat:

$$A \bullet B = \bigcap_{B_x \cap A \neq \emptyset} B_x \quad (6.39)$$

Pe baza acestor exprimări se pot deduce și efectele practice ale deschiderii și închiderii asupra formelor (mulțimilor). Prin deschidere, formele mai mici ca elementul structurant folosit vor fi eliminate, se largesc golurile înglobate în obiecte, contururile sunt netezite prin teșirea convexităților iar obiectele unite prin istmuri sunt separate. Datorită proprietății de dualitate, închiderea va avea aceleași efecte asupra fundalului (complementarei obiectelor) pe care le are deschiderea asupra mulțimilor. Închiderea umple golurile înglobate în obiecte (dacă aceste găuri sunt mai mici decât elementul structurant folosit), netezește contururile formelor prin umplerea concavităților și unește obiectele foarte apropiate (umple “strâmtorile”).

Efectele operațiilor de deschidere și închidere pot fi considerate ca analoage efectelor unei filtrări de netezire a formelor și eliminare a zgomotului (zgomot interpretat ca obiecte și găuri de mici dimensiuni). În [14], în cadrul teoriei algebrice a morfologiei matematice, un filtru este definit ca o operație crescătoare și idempotentă. Trebuie făcută deci distincția între filtrul algebric și filtrul obișnuit, în sensul teoriei prelucrării semnalelor.

### 6.2.2 Filtrele alternate secvențial

Filtrele morfologice sunt transformări neliniare ale semnalului, care modifică local caracteristici geometrice (de formă). Teoria algebrică generală a filtrelor [14] definește o transformare ca filtru dacă aceasta este crescătoare și idempotentă, ceea ce înseamnă deci că deschiderea și închiderea sunt filtre, în timp ce erodarea și dilatarea nu sunt filtre.

Dacă o imagine (mulțime) 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. Totuși, se pot construi filtre ce acționează asupra obiectelor și detaliilor în mod gradat, pe măsura iterării: aceste sunt filtrele alternate secvențial, ce alternează deschideri și închideri după elemente structurante de dimensiune crescătoare:

$$FAS(A) = (((((A \circ B) \bullet B) \circ 2B) \bullet 2B) \circ 3B) \bullet \dots \quad (6.40)$$

sau

$$FAS(A) = (((((A \bullet B) \circ B) \bullet 2B) \circ 2B) \bullet 3B) \circ \dots \quad (6.41)$$

### 6.2.3 Operatori morfologici de contur

O problemă curentă a prelucrărilor de imagini este extragerea punctelor de contur. Cazul în care imaginea este binară constituie o simplificare a metodelor și calculelor. Păstrând



interpretarea pixelilor ca fiind puncte ale obiectului sau ale fundalului, punctele de contur sunt acele puncte de obiect ce au cel puțin 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) și gradientul morfologic (6.44).

$$\delta A = A - A \ominus B \quad (6.42)$$

$$\Delta A = A \oplus B - A \quad (6.43)$$

$$\text{grad } A = A \oplus B - A \ominus B \quad (6.44)$$

### 6.3 Extinderea morfologiei matematice la nivele de gri

Pâna în acest moment s-au prezentat operațiile morfologice clasice, adică aplicate asupra unor seturi (mulțimi). Acestea prezintă limitarea implicită în aplicarea numai asupra unei categorii particulare de imagini, și anume cele a căror structură poate fi ușor și imediat asociată unor mulțimi, adică imaginile binare. Pentru extinderea operațiilor morfologice la funcții se vor construi transformări ce permit trecerea de la o reprezentare de tip funcție la o reprezentare de tip mulțime.

În cele ce urmează ne vom referi la o submulțime  $A$  a spațiului discret  $n$ -dimensional, adică  $A \subseteq \mathbf{Z}^n$ . Elementele mulțimii  $A$  sunt puncte, reprezentate prin vectorul  $n$ -dimensional al coordonatelor,  $\mathbf{x} \in A$ , cu  $\mathbf{x} = (x_1, x_2, \dots, x_n)$ . Ordinea coordonatelor este arbitrară.

Vom numi primele  $n - 1$  coordonate ale unui punct  $\mathbf{x}$  *domeniu spațial* al mulțimii și cea de-a  $n$ -a coordonată, *suprafața mulțimii*. Vom nota aceste “părți” ale mulțimii  $A$  cu  $A_{(1,n-1)}$  și respectiv  $A_{(n,n)}$ . Putem avea  $n$  moduri diferite de a alege aceste partiții ale coordonatelor (corespunzând modului în care se poate alege o coordonată dintr-un total de  $n$ ).

*Vârful* (top) sau *suprafața de vârf* a unei mulțimi  $A$  este funcția  $T(A) : A_{(1,n-1)} \longrightarrow A_{(n,n)}$  definită de:

$$T(A)(\mathbf{z}) = \max\{y \mid (\mathbf{z}, y) \in A, \forall \mathbf{z} \in A_{(1,n-1)}\} \quad (6.45)$$

În acest mod se introduce o funcție, pornind de la o mulțime; funcția este cu valori întregi (funcție scalară) de argument vectorial  $n - 1$ -dimensional. Pentru exemplificare considerăm cazul în care  $n = 2$ , deci fiecare punct al mulțimii  $A$  este un vector bidimensional,  $\mathbf{x} = (i, j)$ . Considerăm coordonata  $i$  ca fiind domeniul spațial și coordonata  $j$  ca

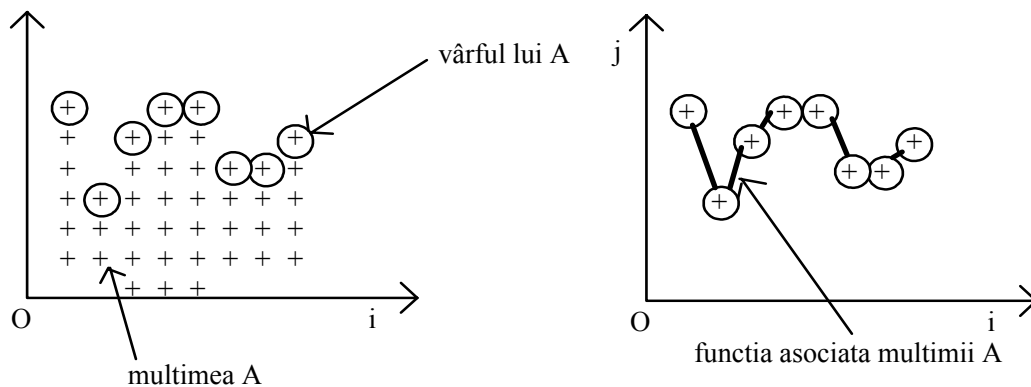


Fig. 6.11: Exemplificare pentru funcția vârf a unei mulțimi.

fiind suprafața mulțimii. Atunci, prin transformarea vârf, asociem mulțimii  $A$  o funcție definită pe domeniul său spațial și cu valori în suprafața sa.

Umbra unei funcții oarecare  $f$  este definită ca transformarea  $f : B \subseteq \mathbf{Z}^{n-1} \longrightarrow C \subseteq \mathbf{Z}$  dată de:

$$U(f) = \{(\mathbf{z}, y) \mid \mathbf{z} \in B, y \in \mathbf{Z}, y \leq f(\mathbf{z})\} \quad (6.46)$$

În acest mod, plecând de la o funcție de variabilă vectorială  $n - 1$ -dimensională, cu valori scalare, se obține o mulțime ale cărei elemente sunt vectori  $n$ -dimensionali. Mulțimea obținută este semideschisă (nemarginită).

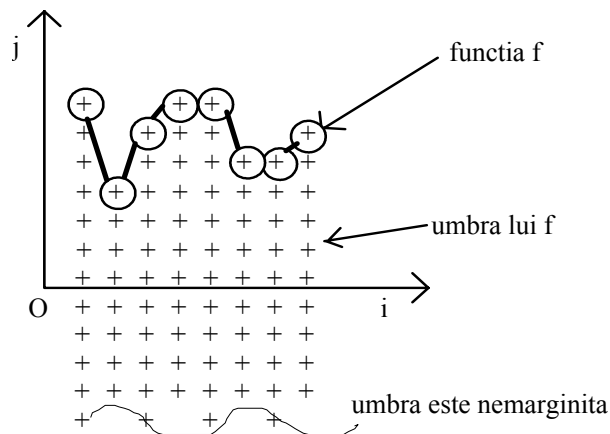


Fig. 6.12: Determinarea umbrei unei mulțimi.

În mod evident, vom avea următoarele relații între cele două transformări mulțime-funcție (vârful și umbra):

$$T(U(F)) = f \text{ și } A \subseteq U(T(A)) \quad (6.47)$$

Operațiile morfologice pentru funcții se definesc prin intermediul transformării acestora în mulțimi (reducând astfel operațiile pe funcții la operații pe mulțimi, așa cum au fost prezentate la morfologia binară):

$$f \ominus g = T(U(f) \ominus U(g)) \text{ și } f \oplus g = T(U(f) \oplus U(g)) \quad (6.48)$$

Ca și în cazul morfologiei pe mulțimi,  $g$  se numește element structurant (funcție structurantă). Se numește 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 mulțimi.

Forma echivalentă a definițiilor erodării și dilatării (folosite în mod efectiv în practică) este, pentru  $\forall \mathbf{x} \in Supp(f)$  :

$$f \ominus g = \min_{\mathbf{y} \in Supp(g)} \{f(\mathbf{x} - \mathbf{y}) - g(\mathbf{y})\}, \quad f \oplus g = \max_{\mathbf{y} \in Supp(g)} \{f(\mathbf{x} - \mathbf{y}) + g(\mathbf{y})\} \quad (6.49)$$

Dacă se utilizează elemente structurante plate (deci caracterizate doar de forma suportului, și nu și de valori asociate acestora), cele două relații de definiție devin identice cu filtrările de ordine de rang minim și respectiv maxim:

$$f \ominus g = \min_{\mathbf{y} \in Supp(g)} \{f(\mathbf{x} - \mathbf{y})\}, \quad f \oplus g = \max_{\mathbf{y} \in Supp(g)} \{f(\mathbf{x} - \mathbf{y})\} \quad (6.50)$$

## Capitolul 7

# METODE DE COMPRESIE A IMAGINILOR

Termenul de compresie a imaginilor (uneori numit și codare a imaginilor) se referă la o clasă largă de tehnici și metode al căror scop este reprezentarea unui imagini date cu un număr cât mai mic de biți (mai mic decât numărul de biți al reprezentării inițiale). Necesitatea reducerii cantității de informație 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 televiziune 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 inițiale din reprezentarea restrânsă se numește decompresie sau decodare; este evident că prin decodare trebuie să se obțină o imagine cât mai apropiată de imaginea originală. Există două categorii fundamentale de tehnici de compresie (codare): codarea fără pierderi (în care imaginea decodată este identică cu imaginea inițială) și codarea cu pierderi, în care se admit mici diferențe față 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  $f$  și imaginea decodată  $g$ ) și factorul (raportul) de compresie. Factorul de compresie  $C$  este raportul dintre cantitatea de informație necesară reprezentării imaginii inițiale și cantitatea de informație 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 informație necesară reprezentării comprimate a fiecărui pixel al imaginii; rata de compresie se măsoară în biți per pixel (bpp).

O altă clasificare posibilă a tehnicilor de compresie se poate face după tipul imaginii căreia i se aplică: vom face astfel distincția între compresia imaginilor binare și compresia imaginilor cu nivele de gri. Se impune totuși o observație: metodele de codare ce se vor prezenta în cadrul tehnicilor specifice imaginilor binare pot fi folosite pentru compresia oricărei succesiuni de valori binare, indiferent de semnificația acestora (ceea ce înseamnă că ar putea fi folosite și pentru compresia imaginilor cu nivele de gri) și 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)<sup>1</sup>. Cele două clase de metode de codare pe care le avem în vedere sunt codarea entropică (metoda de codare Huffman) și metodele de codare *on-line* (pe flux de biți); deosebirea dintre aceste metode (la un nivel al implementării) este că pentru codarea entropică este necesară parcurgerea și stocarea intermediară a întregii imagini.

### 7.1.1 Codarea entropică (Huffman)

Codarea entropică (Huffman) este metoda optimală de codare a unei surse de informație. Codarea sursei de informație  $S$  ale cărei mesaje sunt  $\{s_1, s_2, \dots, s_N\}$ , de probabilități  $\{p(s_1), p(s_2), \dots, p(s_N)\}$  prin alfabetul  $X$  cu  $D$  simboluri înseamnă a asocia fiecărui mesaj  $s_i$  al sursei primare de informație un șir de simboluri din alfabetul codului. Lungimea medie a cuvintelor de cod este dată de raportul dintre entropia sursei primare și entropia alfabetului codului:

$$\bar{l} = \frac{H(S)}{H(X)} \quad (7.1)$$

Se dorește obținerea unei lungimi medii minime a cuvintelor de cod, și deci, echivalent, mărirea entropiei alfabetului codului; la limită, lungimea media minimă posibilă de obținut este:

$$\bar{l}_{\min} = \frac{H(S)}{\log D} \quad (7.2)$$

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 și construirea unei surse restrânse. La fiecare etapă cele  $D$  simboluri cele mai puțin probabile ale sursei de informație sunt reunite într-un nou simbol; procedeul de restrângere continuă până când se obține 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 probabilități 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*

---

<sup>1</sup>Deci convenția de reprezentare prin culori este modificată față de convenția generală utilizată – 0 nu mai este negru, ci alb.

simboluri dintr-un alfabet ternar. Să se calculeze cuvintele de cod, arborele de codare și eficiența codării.

Codarea optimală a unei surse se realizează conform algoritmului Huffman. Se știe că numărul de simboluri ale sursei ce se codează trebuie să îndeplinească o anumită relație ( $\frac{N-D}{D-1} \in \mathbf{N}$ ); în acest caz,  $N = 6$ ,  $D = 3$  și 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 obținem  $N = 7$  și

$$\frac{N-D}{D-1} = \frac{7-3}{3-1} = \frac{4}{2} \in \mathbf{N}$$

Entropia sursei extinse  $S$  este dată de:

$$H(S) = - \sum_{i=1}^7 p(s_i) \log p(s_i) = - (0.3 \log 0.3 + 0.25 \log 0.25 + 0.2 \log 0.2 + 2 \cdot 0.1 \log 0.1) - \\ - 0.05 \log 0.05 - 0 \log 0 = 2.366 \text{ bit/simbol}$$

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
$s_1$	0.3		$r_{11}$	0.3		$r_{21}$	0.45	0
$s_2$	0.25		$r_{12}$	0.25		$r_{22}$	0.3	1
$s_3$	0.2		$r_{13}$	0.2	00	$r_{23}$	0.25	2
$s_4$	0.1		$r_{14}$	0.15	01			
$s_5$	0.1	010	$r_{15}$	0.1	02			
$s_6$	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:

$$\bar{l} = \sum_{i=1}^7 l_i p(s_i) = 1 \cdot 0.3 + 1 \cdot 0.25 + 2 \cdot 0.2 + 2 \cdot 0.1 + 3 \cdot 0.1 + 3 \cdot 0.05 + 3 \cdot 0 = 1.6$$

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 și lungimile lor

În cazul imaginilor (sau al șirurilor binare) mesajele sursei primare sunt formate din grupuri de câte  $B$  biți succesivi (astfel, sursa primară are  $2^B$  mesaje, ale căror probabilități de apariție sunt specifice imaginii considerate). Cu cât  $B$  este mai mare, cu atât mai mare va fi eficiența codării. Tabelul de codare (echivalența între mesajele sursei primare și șirurile de simboluri de cod) este specific fiecărei imagini și trebuie să însoțească codul. Codarea clasică este binară ( $D = 2$ ,  $X = \{0, 1\}$ ).

### 7.1.2 Codarea pe flux de biți

Metodele de codare pe flux de biți (*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 fișiere grafice (TIFF) cât și 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 (șir) prin concatenarea liniilor acesteia.

#### Codarea RLE

Principiul codării RLE (*Run Length Encoding*) este acela de a memora numărul de simboluri succesive ale șirului binar ce au aceeași valoare. Odată ce o secvență de simboluri succesive de aceeași valoare (*run length*) se încheie, simbolul următor nu poate avea decât

valoarea complementară celei inițiale, prin alternanță. Șirul codat trebuie să înceapă cu valoarea primului simbol.

Conform acestei codări, șirul 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 șirul binar 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0 este codat RLE ca 1, 7, 1, 1, 5. Șirul cuvintelor de cod trebuie deci să fie compus din codurile ce corespund lungimilor secvențelor de valori identice succesive; aceasta înseamnă că numărul de biți al reprezentării binare al cuvintelor de cod trebuie să permită reprezentarea lungimii celei mai mari secvențe. Cum determinarea de fiecare dată a acestei lungimi maximale și varierea lungimii cuvântului de cod nu este o soluție corespunzătoare, se preferă fixarea unei lungimi maximale admise a secvențelor. Orice secvență mai lungă decât secvența maximă este despărțită prin intercalarea fictivă a unor simboluri complementare. Să considerăm o codare RLE cu lungimea cuvântului de cod de 2 biți; aceasta înseamnă că lungimea secvenței maxime este 3 (codul 0 trebui rezervat pentru secvența de lungime nulă). Dacă șirul binar este 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0 atunci șirul codat va fi 1, 3, 0, 3, 0, 1, 1, 1, 3, 0, 2 (unde fiecare cifră va fi reprezentată pe 2 biți).

O codare mai bună se obține dacă lungimile secvențelor 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 secvență este mai mică decât 64 este codată direct, cuvântul de cod respectiv numindu-se terminator; dacă o lungime de secvență 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*) și restul împărțirii lungimii secvenței la 64, cu un cod terminator. În plus, există un cod special de sfârșit 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 fișiere imagine (BMP, PCX, TGA).

Porțiunea următoare de program C realizează codarea RLE a șirul 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 șirul *out*. Pozițiile curente din șirurile de intrare și ieșire sunt determinate de variabilele *pos\_in* și respectiv *pos\_out*. Implementarea presupune că tipul de date al șirului *in* permite aplicarea operatorului de negație (!).

```
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 împărțirea șirului 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 și copiat. Conform acestei codări (cu grupe de  $D = 3$  simboluri), șirul de valori binare 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0 este împărțit în grupele (0, 0, 0), (0, 1, 0), (0, 0, 1), (1, 1, 0), (0, 0, 0) și codat ca (0), (1, 0, 1, 0), (1, 0, 0, 1), (1, 1, 1, 0), (0), transformându-se în șirul 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0.

Dacă șirul dat binar de  $n$  grupe de simboluri conține  $n_0$  grupe nule, în urma compresiei mai rămân  $(n - n_0)(D + 1) + n_0$  simboluri și factorul de compresie este atunci:

$$C = \frac{nD}{(n - n_0)(D + 1) + n_0} \quad (7.3)$$

Pentru ca să existe compresie trebuie ca  $C > 1$  și deci atunci

$$n_0 D > n \iff \frac{n_0}{n} > \frac{1}{D} \quad (7.4)$$

adică proporția de grupe nule din șir 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 îmbunătățire a performanței se poate obține dacă se introduce o adaptare a codării, prin detecția liniilor albe (reprezentate doar cu valori 0) și 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).

Porțiunea următoare de program C realizează o variantă de codare WBS cu blocuri de dimensiune  $D$  a șirului de intrare *in* (ale cărui valori sunt 0 și 1) de dimensiune *DIM\_IN*;

codul WBS este scris în șirul de ieșire *out*. Verificarea dacă blocul curent este nul se face prin sumarea valorilor acestuia. Pozițiile curente în șirul de intrare și șirul de ieșire sunt memorate în *pos\_in* și 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 dicționar LUT (folosită și la reprezentarea imaginilor indexate): cuvintele de cod asociate unor șiruri de simboluri ale sursei de intrare sunt indicii (numerele de ordine) pozițiilor la care se găsesc respectivele șiruri de simboluri în dicționarul codului (tabela de codare). Tabela de codare (dicționarul) este creată pe măsura parcurgerii șirului de simboluri de intrare, și, 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<sup>2</sup>).

Dicționarul inițial are două cuvinte: cuvântul 0, având indicele 0 și cuvântul 1, având indicele 1. Din șirul de intrare (presupus binar) se extrage câte un bit și se verifică dacă șirul de biți deja extras se regăsește sau nu în dicționar. Dacă șirul se regăsește în dicționar, se mai adaugă un bit din șirul de intrare. Dacă șirul nu se regăsește în dicționar, atunci șirul este trecut în dicționar, în șirul codat se scrie codul (indicele) prefixului șirului nou adăugat (deci poziția la care se găsește în dicționar cuvântul la care

---

<sup>2</sup>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 existența unei tabele de codare implicite, cunoscute (ca de exemplu tabela de culoare cu nivele de gri).

adăugând ultimul bit al șirului de intrare se obține un cuvânt nou) și ultimul bit din șir devine primul bit al următorului șir de intrare. Procedul continuă până când șirul de intrare nu mai are simboluri necodate. Dacă, înaintea terminării simbolurilor din șirul de codat, tabela de codare (cu număr fixat de intrări) se umple, există două variante de continuare: fie restul șirului de intrare se codează conform tabelii de codare existente prin căutarea celor mai lungi cuvinte de cod, fie tabela de codare este “golită” și se continuă cu procedul inițial.

Să considerăm șirul de intrare 1, 0, 0, 0, 0, 1, 0, 1, 1, 1; dicționarul cuvintelor de cod conține cuvântul 0 (cu indice 0) și cuvântul 1 cu indice 1. Simbolul curent este 1 (pe prima poziție a șirului de intrare) și formează șirul de biți extras; acest șir se regăsește în dicționar (cu indicele 1) și atunci se mai extrage un bit din șir; șirul de biți extrași devine 10. Cuvântul 10 nu există în dicționar, și atunci: în șirul de ieșire se scrie indicele celui mai lung cuvânt din dicționar la care adăugând un bit obținem noul cuvânt (noul cuvânt este 10, cuvântul prefix este 1, iar indicele scris la ieșire este 1), în dicționar se adaugă cuvântul 10 (care va avea indicele 2), iar noua poziție curentă din șirul de intrare este ultimul bit al șirului deja extras, deci poziția 2, bitul 0. Simbolul curent este 0; acest șir se regăsește în dicționar (cu indicele 0) și atunci se mai extrage un bit din șir; șirul de biți extrași devine 00. Cuvântul 00 nu există în dicționar, și atunci: în șirul de ieșire se scrie indicele prefixului noului cuvânt (deci 0), în dicționar se adaugă cuvântul 00 (care va avea indicele 2), iar noua poziție curentă din șirul de intrare este ultimul bit al șirului deja extras, deci poziția 3, bitul 0. Simbolul curent este 0; acest șir se regăsește în dicționar (cu indicele 0) și atunci se mai extrage un bit din șir; șirul de biți extrași devine 00. Șirul 00 se regăsește în dicționar (indicele 3) și atunci se mai extrage un bit din șir; șirul de biți extrași devine 000. Cuvântul 000 nu există în dicționar, și atunci: în șirul de ieșire se scrie indicele prefixului noului cuvânt (deci 3), în dicționar se adaugă cuvântul 000 (care va avea indicele 4), iar noua poziție curentă din șirul de intrare este ultimul bit al șirului deja extras, deci poziția 5, bitul 0. Procedul continuă în mod analog.

Pentru decodare, fiecare nou cuvânt de cod implică scrierea unei noi intrări în tabelul de codare (dicționar) în care șirul de simboluri este format din prefix (șirul de simboluri care se găsește în dicționar la intrarea precizată de cuvântul de cod) și o terminație de 1 bit, a cărei valoare rezultă la primirea cuvântului de cod următor (să nu uităm că șirurile 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 și 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 șir de biți și 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 biți a reprezentărilor nivelelor de gri. Asemenea abordări produc codări fără pierderi a imaginilor și 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 existența unei importante corelații spațiale între valorile pixelilor unei imagini (deci valorile pixelilor vecini sunt asemănătoare). Această corelație poate implica, de exemplu, că valoarea unui pixel poate fi aproximată cu o combinație a valorilor unora dintre vecinii săi. Dacă se stabilește o ordine de parcurgere a pixelilor ce formează imaginea (deci dacă se stabilește o ordine de baleiere a imaginii) și pentru aproximarea valorii pixelului curent se folosesc pixeli vecini spațial lui, parcurși anterior, spunem că prezicem valoarea pixelului curent. Predicția se realizează printr-o funcție, 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 și erorile dintre valorile prezise și cele reale; codarea asociată imaginii inițiale va conține deci funcția de predicție, valorile de start și erorile de aproximare ale fiecărui punct. Dacă predictorul este determinat în mod corect, atunci eroarea de predicție  $e(n)$  este mică și reprezentarea ei necesită mult mai puțini biți decât reprezentarea valorii originale  $u(n)$ . Schema de codare cu predicție este reprezentată în figura 7.1.

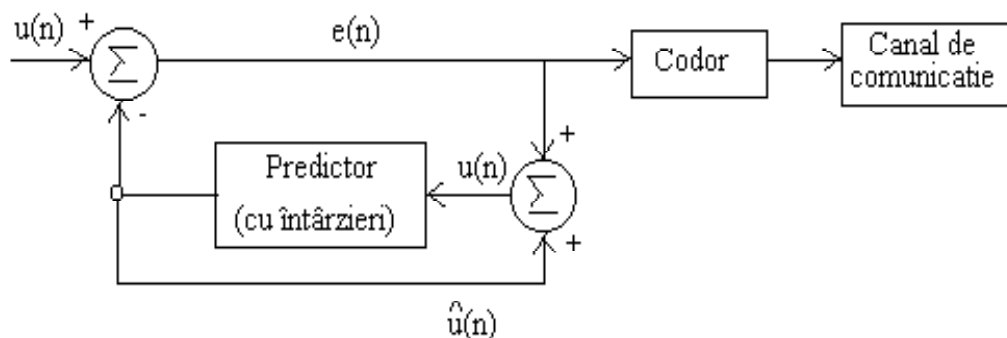


Fig. 7.1: Schema de codare cu predicție.

Ecuatiile ce descriu procesul sunt ecuația erorii (7.5) (care exprimă eroarea de aproximare  $e(n)$  ca diferență între valoarea corectă  $u(n)$  și valoarea prezisă  $\hat{u}(n)$ ) și ecuația de predicție (7.6) (ce exprimă modul în care se determină valoarea aproximativă  $\hat{u}(n)$  din valorile anterioare  $u(n - k_1), u(n - k_2), \dots$  pe baza predictorului  $pred$ ):

$$e(n) = u(n) - \hat{u}(n) \quad (7.5)$$

$$\hat{u}(n) = \text{pred}(u(n - k_1), u(n - k_2), \dots) \quad (7.6)$$

Procesul de decodare este reprezentat schematic în figura 7.2. Eroarea de predicție  $e_q(n)$  (cuantizată) este adunată la valoarea aproximativă  $\hat{u}'(n)$ , determinată cu același predictor  $\text{pred}$  din valorile  $u'(n)$  deja calculate.

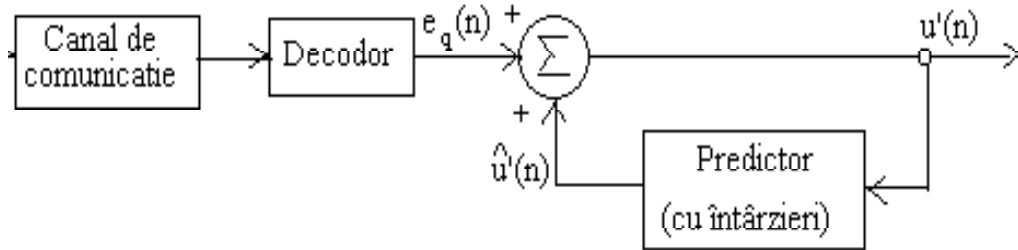


Fig. 7.2: Schema de decodare cu predicție.

Predictorii cei mai simpli sunt liniari și sunt invariați la schimbarea punctului curent. Predictorii pe linii sau coloană calculează aproximarea în punctul curent  $u(m, n)$  al imaginii ca valoarea anterioară de pe aceeași linie  $\hat{u}(m, n) = u(m, n - 1)$  sau de pe aceeași coloană  $\hat{u}(m, n) = u(m - 1, n)$  (dacă ordinea de baleiaj este cea uzuală). Se mai pot folosi predictorii de tip valoare medie:

$$\hat{u}(m, n) = \frac{1}{2} (u(m - 1, n) + u(m, n - 1)) \quad (7.7)$$

$$\hat{u}(m, n) = \frac{1}{4} (u(m - 1, n) + u(m, n - 1) + u(m - 1, n - 1) + u(m - 1, n + 1)) \quad (7.8)$$

Un caz particular de codare cu predicție este modulația delta, caracterizată de cuantizarea erorii de predicție (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 proprietățile de compactare a energiei și decorelare a componentelor spectrale pe care le prezintă majoritatea transformărilor integrale unitare (discutate în secțiunea 4.2). Atâta vreme cât cea mai mare parte a energiei este distribuită în câteva componente spectrale (de joasă frecvență), 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 frecvenței limită de la care începe ignorarea valorilor și, în fine, cuantizarea componentelor spectrale păstrate.

Transformarea optimală trebuie să decoreleze complet componentele spectrale (asigurând astfel și compactarea maximă a energiei și cea mai bună eroare de aproximare prin truncherea frecvențelor înalte). Decorelarea completă este dependentă de proprietățile statistice ale imaginii (matrice de covariație), 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 normați ai matricii de covariație a imaginii. Cum această transformare este evident unică pentru o clasă de imagini, în practică se încearcă găsirea unei aproximații. În condițiile în care majoritatea imaginilor naturale pot fi approximate printr-un model Markov puternic corelat (exprimând dependența puternică a valorii pixelilor de valorile vecinilor lor imediați), transformata cosinus s-a dovedit o foarte bună alegere.

Cuantizarea componentelor spectrale poate integra și selecția componentelor cel mai importante: componentele de frecvență joasă sunt cuantizate o precizie mai mare, iar componentele de frecvență înaltă sunt cuantizate grosier (echivalent chiar cu eliminarea acestora). Numărul de nivele de cuantizare și distribuția acestora (diferențele dintre nivelele vecine) este adaptate statisticii semnalului (cuantizarea optimă este cuantizarea Lloyd-Max [9], [16]).

Exemplul cel mai des folosit de compresie cu transformate este standardul de compresie JPEG (fișiere 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 coeficienți ai transformării sunt copiați într-un vector prin baleierea pe diagonală a blocului de 8 x 8 pixeli. Coeficienții sunt cuantizați în conformitate cu un număr prestabilit de nivele de cuantizare (stabilit prin standard, și proporțional cu factorul de calitate dorit pentru imaginea refăcută). Coeficienții corespunzând frecvențelor nule (valorile medii ale blocurilor) sunt codate predictiv printr-o tehnică de tip DPCM (*Differential Pulse Code Modulation*). Valorile celorlalți coeficienți sunt codati entropic (Huffman). Factorii de compresie ce rezultă sunt cuprinși în mod tipic între 10 și 100. Aspectul de compresie cu pierderi (diferențele față de imaginea originală) se manifestă prin efectul de *blocking*: sublinierea frontierelor de separație a blocurilor de bază (efect observabil și 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 descendenți.

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 împărțiri a unei zone pătrate din imagine în patru “sferturi”. Rădăcina arborelui este asociată întregii imagini (imaginii inițiale), 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, și așa mai departe. Împărțirea imaginii poate continua până când nodurile nivelului curent al arborelui corespund unor zone pătrate a căror dimensiune este de un pixel. Adâncimea arborelui astfel obținut este  $K$ , și fiecare pixel al imaginii va corespunde unui nod terminal (frunză) de pe ultimul nivel al arborelui. Fiecare nod terminal conține informația de valoare a pixelului la care este asociat.

Structura arborelui anterior poate fi simplificată prin introducerea în etapa de construcție 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 părți egale și nodul corespunzător va deveni neterminal (va “căpăta” cei patru descendenți). Dacă regiunea pătrată considerată este uniformă (deci este compusă din pixeli de același fel), nodul respectiv devine un nod frunză (terminal) al arborelui (deci nu mai are descendenți). Fiecare nod terminal conține informația de valoare a zonei de imagine la care este asociat (vezi figura 7.4). O zonă este considerată uniformă dacă diferența maximă de nivel de gri a pixelilor ce o formează nu depășește un anumit prag impus; valoarea zonei uniforme este media nivelelor de gri a pixelilor ce o compun.

Pentru refacerea imaginii inițiale din reprezentarea arborescentă este suficientă alegerea nodurilor terminale a căror valoare corespunde pixelilor de obiect. Adâncimea la care este plasată în arbore o frunză conține informația de dimensiune a zonei pătrate corespunzătoare din imagine (o frunză situată la adâncimea  $h$  corespunde unei zone pătrate de latură  $2^{K-h}$  pixeli). Poziția frunzei față de nodurile de pe același nivel ce au același

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 poziției în arbore a nodurilor terminale și a valorilor acestora. Poziția în arbore a unui nod se specifică prin descrierea căii prin care se ajunge la acesta, pornind de la rădăcina arborelui; această cale va conține codurile de alocare a descendenților ce corespund avansului în adâncime în arbore.

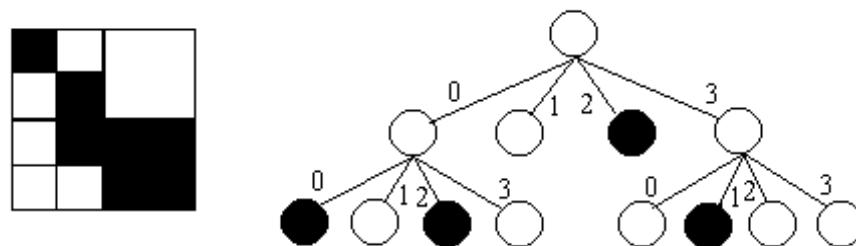


Fig. 7.4: Exemplu de reprezentare a unei imagini binare pe un arbore cuaternar complet

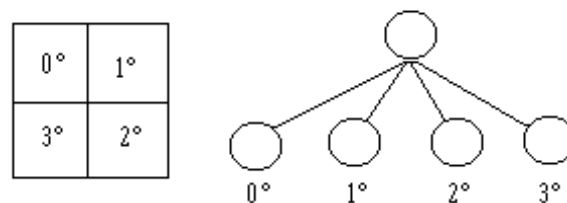


Fig. 7.5: Regula de alocare a descendenților

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 inconvenient este suficient ca pixelii imaginii să nu mai fie parcurși în ordinea tradițională de baleiaj (pe linii, de la stânga la dreapta și 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 spațiului: un parcurs ce trece o singură dată prin fiecare pixel al imaginii, nu se autointersectează și în care oricare doi pixeli parcurși consecutiv sunt vecini spațial în imagine (într-o vecinătate de tip  $V_4$  sau  $V_8$ ). Curbele de umplere a spațiului sunt structuri fractale, definite prin repetarea la diferite nivele ierarhice a unei aceleiași structuri. Pentru baleiajul imaginilor s-au reținut două astfel de curbe: curba Peano-Hilbert (numită și curba Peano în U, după forma celei sale de bază) (vezi figura 7.6) și curba Morton (sau curba Peano în Z) (vezi figura 7.7).

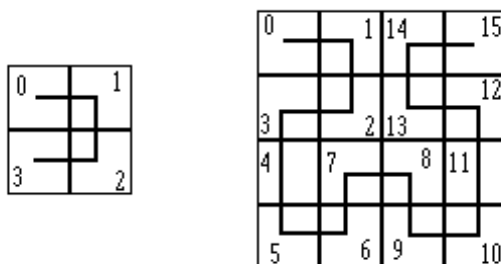


Fig. 7.6: Ordinea de parcurgere a pixelilor pentru curba Peano în U, la două nivele de rezoluție

Disponând de o astfel de ordine de baleiere, este evident că dacă se parcurge imaginea în această ordine, zonele pătrate uniforme (cu pixelii de aceeași valoare) sunt detectate într-o singură trecere și astfel arborele cuaternar poate fi creat direct prin nodurile sale terminale. Pentru o implementare eficientă este însă necesară și 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 relație rapidă de calcul, prin întreteserea biților ce dau coordonatele în imagine a punctului. Cuvântul binar ce exprimă indicele pe curbă a oricărui punct este format din biții din coordonata verticală, ce vor ocupa pozițiile de ordin par și din biții din coordonata orizontală ce vor ocupa pozițiile de ordin impar (păstrându-și aceeași ordine de rang).

## 7.2.4 Cuantizarea vectorială

Cuantizarea vectorială este un algoritm de compresie a imaginilor ce se aplică asupra unor date vectoriale și nu scalare, putând fi interpretat ca o extensie a conceptului de cuantizare scalară. Cuantizarea scalară asociază unei mulțimi mari de valori numere dintr-o mulțime mai mică (în mod tipic acestea din urmă fiind numere naturale); asocierea include (chiar dacă nu explicit) operații de tipul rotunjirii la cel mai apropiat întreg. Cuantizarea vectorială aproximează (sau rotunjește) un grup de numere deodată, nu doar unul singur. Așadar, pentru a realiza o cuantizare vectorială sunt necesare un set

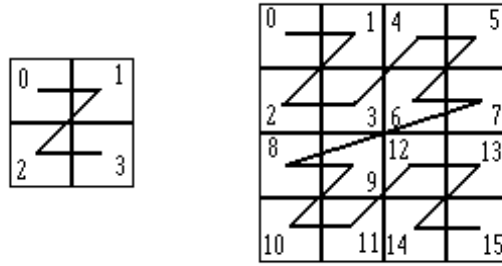


Fig. 7.7: Ordinea de parcurgere a pixelilor pentru curba Peano în  $Z$ , la două nivele de rezoluție

de vectori de aproximare (inclusiv metoda prin care acestea pot fi deduse) și o regulă de asociere a vectorilor de intrare cu vectorii de aproximare.

Să notăm cu  $\mathbf{x}_i$  al  $i$ -lea vector de intrare și cu  $\boldsymbol{\mu}_j$  al  $j$ -lea vector de aproximare. Operația 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 distanței euclidiene). Aceasta este regula de asociere. Dacă există  $n$  vectori de aproximare ce pot fi folosiți, aceștia se pot grupa într-un tabel de codare (existent și la codare și 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  $b$  biți, iar numărul de vectori de aproximare poate fi reprezentat pe  $n_b$  biți, atunci factorul de compresie realizat de cuantizarea vectorială este dat de<sup>3</sup>:

$$C = \frac{pb}{n_b} \quad (7.9)$$

Pentru cazul imaginilor, vectorii de intrare se aleg ca blocuri pătrate, nesuprapuse între ele, din imagine. Dimensiuni uzuale ale acestor blocuri sunt  $4 \times 4$  și  $8 \times 8$  (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 ( $b = 8$ ) și dimensiuni ale tabelii de codare  $n = 256$  (256 vectori de aproximare), atunci raportul de compresie este de 16, respectiv 64.

Construirea tabelii 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” definește un grup, ciorchine, snop sau o clasă de unități, “asemănătoare”. Asemănarea unităților este determinată în mod uzual prin asociere, similaritate sau distanță între unități (vectori). Algoritmul de clustering este procesul prin care unei

<sup>3</sup>Acest mod de calcul al raportului de compresie nu ține seama de necesitatea transmiterii sau memorării și a tabelului de codare, de dimensiune  $n p b$  biți.

mulțimi de unități (entități)  $i$  se asociază, element cu element, o informație de apartenență la un anumit grup. Mai general, putem interpreta procesul de clustering ca un proces de partiționare a unui set de unități într-un număr de submulțimi (numite clase sau clustere), pe baza unui anumit criteriu. Indiferent de criteriul folosit, se dorește obținerea unor clustere distincte, omogene și bine separate. Numărul de clustere în care se face împărțirea setului de unități nu este în mod obligatoriu cunoscut apriori.

Metodele de clustering iterativ distribuie obiectele (vectorii) într-un număr predefinit de clase, repetând testarea unor condiții pentru fiecare obiect al mulțimii; în funcție de îndeplinirea sau nu a respectivelor condiții, partiția existentă la un moment dat este declarată corespunzătoare sau, în urma modificării alocării unor unități, procedeul de verificare se reia. Se poate considera că algoritmii iterativi fac mai multe “tregeri” prin setul de obiecte de partiționat, până la obținerea stabilizării (convergenței) valorii criteriului ce caracterizează calitatea partiției.

Calitatea partiției (a clasificării) este măsurată de suma varianțelor clusterelor (adică suma distanțelor de la fiecare vector la centrul clasei în care aparține, ceea ce poate fi interpretat și ca o eroare de aproximare a vectorilor din clase prin centrul respectivei clase). Deci funcția criteriu ce trebuie minimizată este

$$J = \sum_{j=1}^n J_j = \sum_{j=1}^n \sum_{\mathbf{x}_i \in \omega_j} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2 = \sum_{j=1}^n \sum_{i=1}^N u_{ij} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2 \quad (7.10)$$

În urma minimizării lui  $J$  trebuie determinate valorile  $\boldsymbol{\mu}_j$  (centrele claselor) și valorile binare  $u_{ij}$ , coeficienții de apartenență ai vectorilor  $i$  la clasele  $j$ , definiți de:

$$u_{ij} = \begin{cases} 1, & \text{dacă } \mathbf{x}_i \in \omega_j \\ 0, & \text{dacă } \mathbf{x}_i \notin \omega_j \end{cases} \quad (7.11)$$

Determinarea centrelor claselor se poate face simplu, prin anularea derivatei în raport cu  $\boldsymbol{\mu}_j$  a funcției criteriu  $J$

$$\frac{\partial J}{\partial \boldsymbol{\mu}_j} = 2 \sum_{i=1}^N u_{ij} (\boldsymbol{\mu}_j - \mathbf{x}_i) = 0 \quad (7.12)$$

de unde rezultă că centrele claselor sunt mediile vectorilor ce aparțin acestora:

$$\boldsymbol{\mu}_j = \frac{\sum_{i=1}^N u_{ij} \mathbf{x}_i}{\sum_{i=1}^N u_{ij}} \quad (7.13)$$

Determinarea coeficienților de apartenență  $u_{ij}$  este însă o problemă de optimizare combinatorială, care nu poate fi rezolvată analitic. Pentru rezolvarea acestei probleme sunt

necesare metode iterative. Metoda imediată urmărește să determine, pentru fiecare vector al setului, dacă acesta poate fi mutat dintr-o clasă în alta, astfel ca suma varianțelor claselor să scadă. După fiecare asemenea mutare, este necesară actualizarea mediilor claselor între care s-a făcut schimbul. Iterațiile se repetă până când nici un vector nu mai poate fi mutat. Algoritmul poate fi descris în etapele următoare:

1. se alege o partiție aleatoare a setului de obiecte (vectori)
2. pentru fiecare vector  $\mathbf{x}_i$  din set, dacă nu este unic în clasa sa  $\omega_j$ , se calculează costul mutării în altă clasă,  $\omega_k$ ,  $k \neq j$ ; acest cost este

$$c_k = \frac{n_k}{n_k + 1} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2 - \frac{n_j}{n_j - 1} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2 \quad (7.14)$$

3. vectorul  $\mathbf{x}_i$  este mutat în clasa pentru care costul  $c_k$  este minim; se recalculează mediile claselor implicate în schimbare ( $\omega_j$  și  $\omega_k$ )
4. dacă cel puțin 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 recalculat după fiecare schimbare ce implică fiecare vector al mulțimii considerate, ceea ce are ca efect un volum mare de calcule. O simplificare a metodei provine din observația intuitivă că este normal ca un obiect să fie alocat (să aparțină) clasei de care este cel mai apropiat (în sensul distanței la media acesteia). Folosind această observație, 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 mulțimii de obiecte de partiționat. Acest algoritm este algoritmul “Basic ISODATA”<sup>4</sup> (cunoscut și sub numele de *k-means* - “cele k medii”). Algoritmul poate fi descris de următoarele etape:

1. se alege o partiție aleatoare a setului de obiecte (vectori)
2. pentru fiecare vector  $\mathbf{x}_i$  din set, se calculează distanțele sale la mediile tuturor claselor,

$$d_k = \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2 \quad (7.15)$$

3. în iterația următoare, vectorul  $\mathbf{x}_i$  va fi mutat în clasa la care distanța  $d_k$  este minimă
4. după parcurgerea completă a setului de vectori, se reactualizează mediile claselor

---

<sup>4</sup>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 (față de iterația 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 algoritmi iterativi prezentați nu se poate preciza numărul de parcurgeri ale setului de vectori (obiecte) de partiționat. Numărul de iterații este puternic dependent de alegerea partiției inițiale a vectorilor, precum și 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 folosiți. În imagine sunt foarte vizibile frontierele dintre blocurile de 8 x 8 folosite.

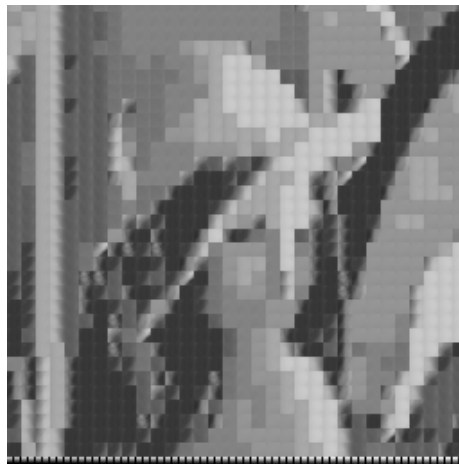


Fig. 7.8: Imagine refăcută după codarea prin cuantizare vectorială; codarea a fost realizată cu blocuri de 8 x 8 pixeli; tabelul vectorilor de cod are 16 intrări, deci se obține o compresie de 128.

## Capitolul 8

# SEGMENTAREA IMAGINIILOR

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 definiție matematizată a procesului de segmentare, și anume segmentarea unei imagini  $f$  este definită ca partiționarea [completă] a lui  $f$  (8.1) într-un ansamblu de mulțimi disjuncte nevide și conexe (8.2), ce satisfac fiecare un anumit criteriu  $\mathcal{C}$  (8.3), criteriu ce nu mai este respectat pentru reuniunea oricăror două elemente ale partiției.

$$f = \bigcup_{i=1}^C f_i, \quad f_i \text{ conexe} \quad (8.1)$$

$$f_i \cap f_j = \emptyset, \forall i \neq j \text{ și } f_i \text{ este conexă}, \forall i \quad (8.2)$$

$$\mathcal{C}(f_i) = TRUE, \forall i \text{ și } \mathcal{C}\left(f_i \cup f_j\right) = FALSE, \forall i \neq j \quad (8.3)$$

Alegerea unei tehnici specifice de segmentare (partiționare a imaginii) este legată de mai multe aspecte caracteristice imaginii de analizat și cerințelor utilizatorului. După natura și conținutul imaginii, tehnicile de segmentare trebuie să țină cont de prezența în imagine a diverse categorii de artefacte:

- reflexii, iluminare neomogenă
- zgomot suprapus informației utile

- zone texturate

După primitivele de extras, tehnicile de segmentare se împart în două categorii fundamentale: tehnicile de segmentare orientate pe regiuni și tehnicile de segmentare orientate pe contur. Primitivele extrase din imagine sunt regiuni (forme) și zone texturate pentru tehnicile orientate pe regiuni, sau entități 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
- creșterea și fuziunea regiunilor
- segmentarea texturilor
- segmentarea prin metode de clustering

Tehnicile principale de segmentare orientată pe contururi sunt:

- extragerea conturilor prin metode de gradient și derivative
- extragerea conturilor prin metode neliniare
- extragerea conturilor prin metode liniare optimale
- extragerea conturilor 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, operația de segmentare orientată pe regiuni urmărește extragerea din imagine a zonelor (regiunilor) ocupate de diferitele obiecte prezente în scenă. Un obiect se definește ca o entitate caracterizată de un set de parametri ale căror valori nu se modifică în

diferitele puncte ce aparțin entității considerate. Mai simplu, putem spune că obiectul are proprietatea de uniformitate a parametrilor de definiție.

Unul dintre cei mai simpli parametri de definiție este nivelul de gri al punctului. Nivelul de gri corespunde în scenă unei proprietăți fizice [9] (reflectanță, transmitivitate, valoare tristimulus, etc.) ce este preluat de senzorul de imagine și asociat luminanței imaginii. În acest caz, histograma imaginii (funcția de densitate de probabilitate a variabilei aleatoare discrete ale cărei realizări sunt nivelele de gri din imagine) reflectă distribuția în scenă a proprietății fizice înregistrate. Pentru o imagine  $f$  de  $M \times N$  pixeli și  $L$  nivele de gri, histograma este definită (8.4) ca probabilitatea de apariție în imagine a diferitelor nivele de gri posibile.

$$h(i) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \delta(i - f(m, n)) \quad , \quad i = 0, 1, \dots, L-1 \quad (8.4)$$

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 obținute prin scanarea documentelor scrise și a tipăriturilor sau imaginile în infraroșu (temperatura punctelor este asociată nivelelor de gri astfel încât mai fierbinte înseamnă mai alb). Pentru toate aceste tipuri de imagini histograma este de tipul celei prezentate în figura 8.1.

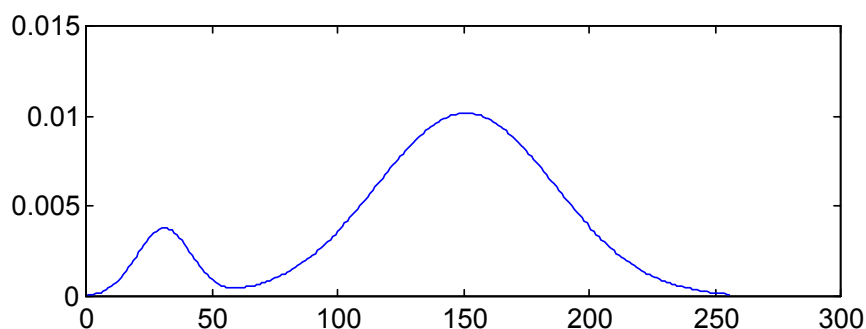


Fig. 8.1: Histogramă bimodală

### Tehnici de prăguire (*thresholding*)

Separarea modurilor histogramei (și deci identificarea obiectelor din imagine, respectiv caractere scrise / pagini albe și obiecte fierbinți / 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 inițială  $f$  de nivele de gri se construiește o imagine de etichete (imagine etichetată)  $g$ , conform transformării descrise de (8.5) (vezi figura 8.2).

$$g(m, n) = \begin{cases} E_0, 0 \leq f(m, n) < T \\ E_1, T \leq f(m, n) < L \end{cases} \quad (8.5)$$

Imaginea etichetată va fi descrisă de două etichete:  $E_0$  pentru punctele al căror nivel de gri este mai mic decât pragul  $T$  și  $E_1$  pentru punctele al căror nivel de gri este mai mare decât pragul  $T$ . Etichetele  $E_0$  și  $E_1$  pot fi valori numerice (0 și 1, sau 0 și 255) sau pot fi șiruri de simboluri sau alți 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  $(m, n)$ ) și 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 ( $T_1 = T_2$ ) și contrastarea se face la valorile limită ale nivelelor de gri ( $\alpha = 0, \beta = L - 1$ )<sup>1</sup>.

Segmentarea pe histogramă (numită și prăguire sau *thresholding*) semnifică determinarea unor nivele de gri ce separă modurile histogramei. Tuturor punctelor din imagine al căror nivel de gri corespunde unui același mod, li se asociază o aceeași etichetă (număr, șir de simboluri), rezultând o imagine etichetată, ce pune în evidență diferitele obiecte ale scenei inițiale.

---

<sup>1</sup>Există însă și o variantă de binarizare cu două praguri (transformare punctuală numită decupare - "slicing"), (figura 8.2) definită de ecuația 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 existenței a mai multe praguri de segmentare  $T_k$ , transformarea de segmentare pe histogramă este descrisă de (8.6)

$$g(m, n) = E_k \quad \text{dacă} \quad T_k \leq f(m, n) < T_{k+1} \quad (8.6)$$

unde  $T_0 = 0$ ,  $T_C = L$ ,  $k = 0, 1, \dots, C - 1$ .

Pragurile  $T_k$  se aleg prin inspectia histogramei, în minimele locale ale acesteia. Acest tip de segmentare multinivel este mai puțin eficient decât binarizarea, din cauza dificultății 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 cunoașterea numărului de tipuri de obiecte din imagine, pentru alegerea corespunzătoare a numărului de praguri de segmentare. În marea majoritate a cazurilor, segmentarea obținută nu este corectă (există regiuni prost etichetate); ca o regulă generală de îmbunătățire a performanțelor, se recomandă aplicarea, înaintea segmentării, a unor operații de filtrare (eliminarea zgomotului), contrastare, îmbunătățire, netezire a histogramei - numite preprocesări.

În general, se admite clasificarea metodelor de segmentare pe histogramă [5] după atributele global, local și dinamic. Aceste atribute se referă la modul de calcul al pragurilor de segmentare  $T_k$  în funcție de nivelul de gri din fiecare punct al imaginii  $f(m, n)$ , coordonatele punctelor din imagine  $(m, n)$  și o anumită proprietate locală  $p(m, n)$  a punctului  $(m, n)$ , conform (8.7):

$$T_k = T_k(f(m, n), p(m, n), (m, n)) \quad (8.7)$$

Segmentarea se numește globală dacă pragurile depind doar de nivelele de gri ale punctelor imaginii:

$$T_k = T_k(f(m, n)) \quad (8.8)$$

Segmentarea multinivel descrisă de (8.6) este în mod evident o metodă de tip global.

Segmentarea se numește locală dacă pragurile depind de nivelul de gri și de anumite atribute locale calculate pentru vecinătăți ale fiecărui punct:

$$T_k = T_k(f(m, n), p(m, n)) \quad (8.9)$$

Segmentarea se numește dinamică dacă pragurile depind de poziționarea punctelor în imagine (forma cea mai generală a modului de deducere pragurilor) (8.7).

## Determinarea automată a pragurilor: metoda Bhattacharyya

Metoda Bhattacharyya se bazează pe descompunerea histogramei în moduri individuale Gaussiene, adică se încearcă exprimarea histogramei imaginii ca o sumă ponderată de funcții de densitate de probabilitate de tip normal (Gaussian). Modelarea modurilor histogramei imaginilor prin distribuții normale este o presupunere ce se întâlnește în multe tehnici de prelucrare și analiză și 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 varianțele acestor moduri sunt determinate de zgomotul suprapus imaginii (care nu este obligatoriu să afecteze în același 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 distribuții normale. Pentru o distribuție 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:

$$\frac{\delta \ln N(\mu_k, \sigma_k)(x)}{\delta x} = -\frac{x}{\sigma_k^2} + \frac{\mu_k}{\sigma_k^2} = m_k x + n_k \quad (8.10)$$

Se observă prin examinarea expresiei (8.10) că derivata logaritmului distribuției normale este o dreaptă de pantă negativă, din ai cărei parametri se pot deduce media și varianța distribuției. Parametrii statistici ai distribuției sunt dați de ecuațiile (8.11).

$$\sigma_k = \sqrt{\frac{1}{|m_k|}}, \text{ și } \mu_k = \frac{n_k}{|m_k|} \quad (8.11)$$

Această observație poate fi aplicată și pentru o mixtură de distribuții normale. Să considerăm că histograma  $h$  a imaginii este compusă prin superpoziția 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 drepte se pot determina deci conform (8.11) parametrii statistici ai distribuției locale.

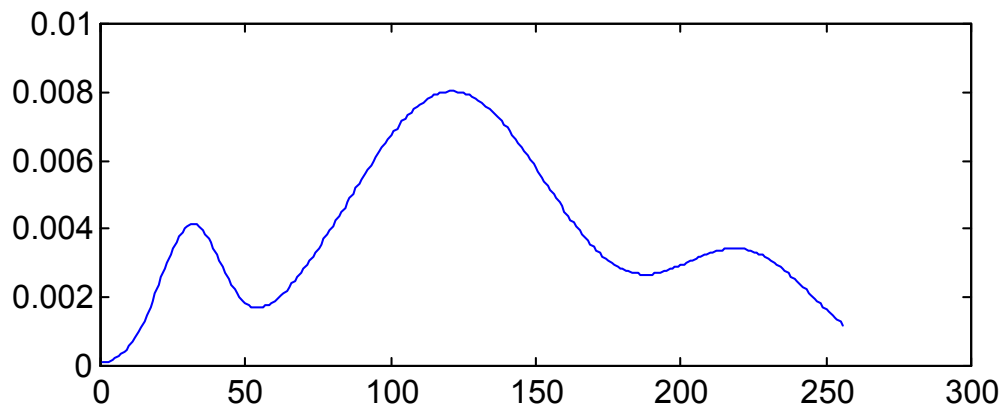


Fig. 8.3: Histogramă cu trei moduri normale

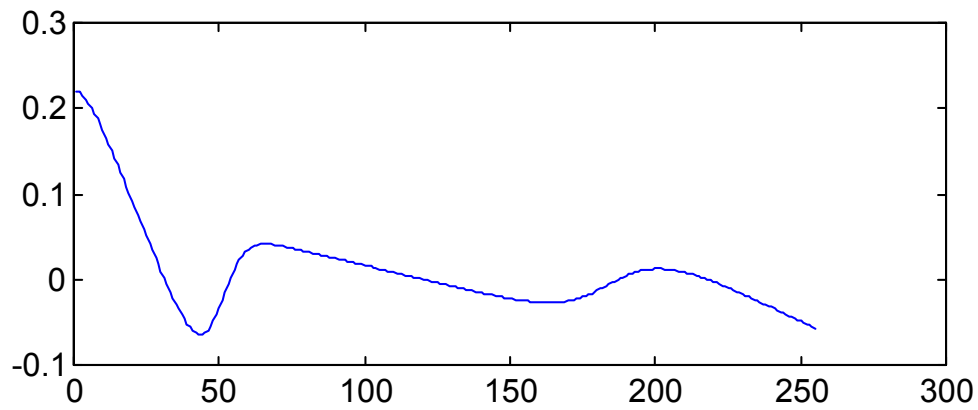


Fig. 8.4: Aplicarea metodei Bhattacharyya pentru histograma trimodală prezentată anterior; se pot observa intervalele pe care funcția este liniară și descrescătoare, ce corespund modurilor.

Așadar, pentru aplicarea metodei la segmentarea pe histogramă a imaginilor, se va studia comportamentul derivatei logaritmului histogramei, adică a funcției  $z(a)$ :

$$z(a) = \ln \frac{h(a)}{h(a-1)}, a = \overline{1, L-1} \quad (8.12)$$

Pentru funcția 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 și pe baza parametrilor deduși 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. Informațiile apriori necesare pentru aplicarea unei asemenea tehnici sunt numărul de tipuri de obiecte din imagine,  $C$ , procente de ocupare a imaginii de către fiecare tip de obiecte,  $P_i$  și distribuția nivelelor de gri ce caracterizează fiecare tip de obiect,  $p_i(x)$ . Atunci histograma imaginii va fi determinată de mixtura distribuțiilor tipurilor de obiecte:

$$h(x) = \sum_{i=1}^C P_i p_i(x), \sum_{i=1}^C P_i = 1 \quad (8.13)$$

Cazul cel mai simplu și mai des folosit este cel al binarizării (8.5), în care trebuie determinat un unic prag  $T$  ce separă distribuțiile celor două tipuri de obiecte din imagine (în mod tipic, obiecte “utile” și fundal). Criteriul ce se urmărește optimizat este eroarea de segmentare (clasificare) a punctelor din imagine, adică este dat de numărul de pixeli ce aparțin primului tip de obiect, dar au nivelul de gri mai mare ca pragul  $T$  (fiind deci alocați greșit celui de-al doilea tip de obiect) și numărul de pixeli ce aparțin celui de-al doilea tip de obiect, dar au nivelul de gri mai mic decât pragul de segmentare  $T$  (fiind deci alocați greșit primului tip de obiect). Așadar, eroarea de segmentare va fi dată de (8.14):

$$E(T) = P_1 \int_T^{+\infty} p_1(x) dx + P_2 \int_{-\infty}^T p_2(x) dx \quad (8.14)$$

Pragul optim va minimiza eroarea de segmentare a pixelilor. Minimizarea erorii (8.14) conduce la rezolvarea ecuației (8.15), în necunoscuta  $T$ .

$$\frac{\partial E(T)}{\partial T} = 0 \quad (8.15)$$

Derivând (8.14) se obține forma echivalentă a ecuației (8.15):

$$P_1 p_1(T) = P_2 p_2(T) \quad (8.16)$$

După cum am menționat și în secțiunea dedicată tehnicilor de segmentare ce nu folosesc informații apriori despre imagine (metoda Bhattacharyya), presupunerea că distribuția nivelelor de gri a diferitelor tipuri de obiecte este de tip normal (Gaussian) este relativ des întâlnită. În aceste condiții, distribuțiile  $p_1(x)$  și  $p_2(x)$  sunt distribuții normale,  $N_1(\mu_1, \sigma_1)(x)$  și  $N_2(\mu_2, \sigma_2)(x)$ , iar ecuația (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 obține următoarea ecuație de gradul 2 în necunoscuta  $T$ :

$$T^2 \left( \frac{1}{\sigma_1^2} - \frac{1}{\sigma_2^2} \right) - 2T \left( \frac{\mu_1}{\sigma_1^2} - \frac{\mu_2}{\sigma_2^2} \right) + \left( \frac{\mu_1^2}{\sigma_1^2} - \frac{\mu_2^2}{\sigma_2^2} \right) - 2 \ln \frac{P_1 \sigma_2}{P_2 \sigma_1} = 0$$

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  $\mu_1$  și  $\mu_2$ , afectată de un zgomot Gaussian aditiv, având varianța  $\sigma^2$ . În aceste condiții, ecuația de gradul 2 devine o ecuație liniară, a cărei soluție 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 și pentru imagini ce conțin 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 și în cazul metodei Bhattacharyya, că influența fiecărui mod este limitată la intervale nesuprapuse de nivele de gri.

### 8.1.2 Creșterea și fuziunea regiunilor

Pentru aplicarea cu succes a tehnicilor de segmentare pe histogramă prezentate anterior trebuie să fie îndeplinite neapărat câteva condiții (deja enunțate). Aplicarea tehnicilor de segmentare pe histogramă este condiționată 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 parțial pe porțiuni foarte mici); apoi este necesară cunoașterea 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 condiții enunțate sunt îndeplinite, nu se poate garanta condiția de conexitate a regiunilor obținute în urma segmentării (8.2). Acest lucru este evident, atât timp cât două obiecte de același tip, neconexe, primesc prin segmentarea pe histogramă o aceeași etichetă, și formează în imaginea de etichete o regiune neconexă. O metodă care respectă toate condițiile impuse de definiția metematică a segmentării, și anume (8.1), (8.2) și (8.3), este creșterea regiunilor.

#### Creșterea regiunilor

Principiul pe care se bazează creșterea regiunilor este simplu: se aleg în imagine puncte reprezentative pentru fiecare obiect individual și categorie de obiecte, pe baza cărora are loc un proces de aglomerare a pixelilor vecini acestora, ce au aceleași proprietăți (în particular același nivel de gri). În urma acestui proces de aglomerare (adăugare de puncte) se obțin zone (regiuni) de pixeli cu aceleași caracteristici, deci obiecte individuale. Procesul se oprește în momentul în care fiecare punct al imaginii a fost alocat unei regiuni. Evident, metoda astfel descrisă pe scurt, are două etape esențiale: alegerea punctelor de start (puncte inițiale), numite germeni sau semințe, și creșterea propriu-zisă a regiunilor [19], [2].

Numărul final de regiuni rezultate este egal cu numărul de germeni aleși inițial pentru creștere. În principiu, este de dorit ca fiecare obiect individual aflat în imagine să fie marcat de câte un germen. Dacă în interiorul unui aceluiasi obiect se găsesc mai mulți germeni, pentru fiecare dintre ei va fi crescută o regiune; acesta face ca obiectul inițial să fie împărțit artificial prin segmentare în mai multe regiuni. Parțial, acest neajuns se poate corecta printr-o etapă ce urmează creșterii regiunilor, și anume fuziunea regiunilor adiacente ce au proprietăți asemănătoare. Dacă în interiorul unui obiect nu este ales nici un germen, obiectul respectiv va fi înglobat de regiunile ce cresc pornind de la germeni din vecinătatea sa spațială; astfel, respectivul obiect nu apare ca o regiune distinctă și este pierdut, rezultând o eroare gravă de segmentare.

Pentru a preveni efectul unor neuniformități de iluminare pe suprafața imaginii, acesta este împărțită în ferestre nesuprapuse; în fiecare astfel de fereastră se alege un număr de germeni, al căror plasament spațial este aleator (germenii se distribuie uniform pe

suprafața 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 spațial al germenilor să se facă în interiorul regiunilor și 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 depășește un anumit procent prestabilit (10% - 20%) din diferența maximă de nivele de gri a ferestrei, punctul ales este considerat ca plasat corect.

O verificare suplimentară încearcă să prevină o eventuală suprasegmentare<sup>2</sup> (împărțirea artificială a unui același obiect în mai multe regiuni), eliminând germenii plasați în interiorul aceluiași obiect. Verificarea se face pe baza calculului variației nivelelor de gri de-a lungul drumurilor<sup>3</sup> arbitrare ce unesc perechi de germenii. Dacă există o cale ce unește doi germenii de-a lungul căreia nivelul de gri nu variază cu mai mult de 20% - 30% din diferența maximă a nivelelor de gri din fereastră, cei doi germenii sunt plasați în interiorul unei zone de nivele de gri uniforme, deci în interiorul unui același obiect. În aceste condiții unul dintre cei doi germenii ai perechii este eliminat, deoarece este redundant. Dacă de-a lungul tuturor căilor ce unesc perechea de germenii nivelul de gri variază mai mult decât pragul ales, atunci se consideră că cei doi germenii sunt plasați în interiorul unor obiecte diferite (deoarece căile ce unesc germenii traversează regiuni de frontieră). În practică, examinarea tuturor drumurilor (căilor) ce unesc perechi de germenii este extrem de costisitoare din punctul de vedere al timpului de calcul. De aceea se verifică doar căile formate din segmente verticale și orizontale, și eventual, dreapta ce unește cele două puncte (dacă această dreaptă poate fi reprezentată de o secvență de puncte conexe) (vezi figura 8.5).

Valorile procentuale ale pragurilor de comparație, precum și numărul de germenii distincți ce rămân după procesul de reducere, nu trebuie considerate ca fixe; nu există valori standardizate și alegerea acestora se face pe baza condițiilor particulare (legate de conținutul imaginii) și a experienței utilizatorului.

Pornind de la germenii aleși, regiunile sunt obținute printr-un proces de creștere aproape simultană, început de la aceștia, până când toți pixelii imaginii sunt repartizați unei regiuni. Cvasi-simultaneitatea creșterii poate fi realizată cu un algoritm serial, prin alocarea pixelilor ce sunt adiacenți (vecini) zonelor deja segmentate. Această alocare trebuie să țină 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 față de nivelul de gri al germenului regiunii la care se alocă. În același timp, la o singură trecere, numărul de puncte ce se adaugă unei regiuni nu poate depăși un număr prestabilit (condiția încearcă

<sup>2</sup>În general, prin suprasegmentare se înțelege partiționarea imaginii într-un număr de regiuni mai mare decât numărul de obiecte. Există și noțiunea reciprocă de subsegmentare: împărțirea imaginii într-un număr de regiuni mai mic ca numărul de obiecte.

<sup>3</sup>Un drum între două puncte ale imaginii este o secvență 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 și 2 sunt uniți de o cale cu segmente paralele cu orizontala și verticala de intensitate constantă, deci sunt redundanți; germenii 3 și 4 sunt uniți de o cale dreaptă de aceeași intensitate, deci sunt redundanți; orice cale ce unește germenii 1 și 3 are o diferență mare de intensitate.

să asigure creșterea relativ uniformă și izotropă a tuturor regiunilor).

Dacă adăugarea de noi pixeli se blochează (criteriul de uniformitate nu mai este respectat), diferența 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 creștere a regiunilor sunt acelea că nu mai este necesară nici o informație privind conținutul imaginii, regiunile crescute sunt conexe și nu există puncte neetichetate (nealocate vreunei regiuni) și poziția frontierelor dintre diferitele regiuni corespunde poziției frontierelor percepute subiectiv în imagine.

## Fuziunea regiunilor

O extindere a principiului utilizat în creșterea regiunilor, și anume adăugarea la o regiune a unor entități (pixeli în acest caz) a căror proprietăți 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 entități atomice ale imaginii - deci pixelii) până când regiunile adiacente devin suficient de diferite. Procesul de fuziune a regiunilor poate fi aplicat și în urma unei creșteri a regiunilor, pentru a înlătura efectele unei eventuale suprasegmentări. Există mai multe criterii de fuziune a regiunilor adiacente, a căror acțiune de verificare a deosebirii între regiuni se face fie prin inspecția frontierei comune, fie prin caracterizarea interiorului regiunii.

Pentru două regiuni adiacente  $R_i$  și  $R_j$ , al căror perimetru este  $Perim(R_i)$  și  $Perim(R_j)$ , putem determina  $P_m = \min(Perim(R_i), Perim(R_j))$  și  $P$  lungimea frontierei comune<sup>4</sup>.

---

<sup>4</sup>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  $n_s$ ) și puncte *tari* (în număr de  $n_t$ ). Un punct slab este acel punct pentru care diferența 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 diferența de nivele de gri între vecinii din regiunile adiacente este foarte mare (mai mare ca un anumit prag fixat). Cu aceste notații, criteriile de fuziune a regiunilor  $R_i$  și  $R_j$  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  $\theta_1$  controlează dimensiunea regiunilor ce se unesc și 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  $\theta_2$  și  $\theta_3$  sunt 0.75 și 0.2.

Abordarea fuziunii pe baza caracterizării interiorului regiunilor necesită definirea a două componente: o modalitate de caracterizare a proprietăților regiunilor și 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 respectivă; nu pot lipsi din acest vector nivelul de gri mediu al regiunii și varianța acestuia.

În [9] se propun patru funcții de măsură a asemănării între perechi de vectori; pentru doi vectori  $\mathbf{x}_i$  și  $\mathbf{x}_j$ , având aceeași dimensiune, acestea se definesc ca:

$$F_1(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle \text{ (produsul scalar dintre vectori)}$$

$$F_2(\mathbf{x}_i, \mathbf{x}_j) = \frac{\langle \mathbf{x}_i, \mathbf{x}_j \rangle}{\langle \mathbf{x}_i, \mathbf{x}_i \rangle + \langle \mathbf{x}_j, \mathbf{x}_j \rangle - \langle \mathbf{x}_i, \mathbf{x}_j \rangle} \text{ (similaritatea dintre vectori)}$$

$$F_3(\mathbf{x}_i, \mathbf{x}_j) = \frac{\langle \mathbf{x}_i, \mathbf{x}_j \rangle}{\sqrt{\langle \mathbf{x}_i, \mathbf{x}_i \rangle \langle \mathbf{x}_j, \mathbf{x}_j \rangle}} \text{ (corelația normalizată dintre vectori)}$$

$$F_4(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j) \cdot A \cdot (\mathbf{x}_i - \mathbf{x}_j)^T$$

(distanța generalizată dintre vectori, unde  $A$  este o matrice pozitiv definită)

Pentru primele trei funcții, o valoare mai mare corespunde unei asemănări mai mari între vectori (valorile maxime pentru  $F_2(\mathbf{x}_i, \mathbf{x}_j)$  și  $F_3(\mathbf{x}_i, \mathbf{x}_j)$  sunt 1). Pentru funcția de similaritate bazată pe distanța generalizată, o valoare mai mică corepunde unei asemănări mai puternice între vectori. Prin particularizarea matricii  $A$  se pot obține diferite distanțe, ca distanța Euclidiană obișnuită ( $A$  fiind matricea unitate,  $A = I$ ), distanțe Euclidiene ponderate (dacă  $A$  este o matrice diagonală), sau distanța Mahalanobis (dacă  $A$  este o matrice de covariație a componentelor).

## 8.2 Segmentarea orientată pe contururi

Într-o imagine, variațiile de valoare ale pixelilor reprezintă schimbări ale proprietăților fizice sau geometrice ale scenei sau ale obiectului observat. Aceste schimbări pot corespunde fizic la variațiile iluminării, schimbările de orientare sau de distanță față de observator, schimbări de reflectanță ale suprafețelor, variații de absorbție a radiației. Într-un număr mare de cazuri, aceste variații de intensitate sunt informații importante pentru operațiile ce urmează segmentării, informații 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 variații se va face prin operatori derivativi de tip gradient.

Pentru o imagine cu suport spațial continuu, pe direcția unei muchii, derivata va fi maximă. Derivata imaginii pe direcția  $r$ , ce face unghiul  $\theta$  cu orizontala, este dată de combinația liniară a derivatelor parțiale pe direcțiile orizontală și verticală (8.17):

$$\begin{aligned}\frac{\partial f}{\partial r} &= \frac{\partial f}{\partial x} \frac{\partial x}{\partial r} + \frac{\partial f}{\partial y} \frac{\partial y}{\partial r} = \frac{\partial f}{\partial x} \cos \theta + \frac{\partial f}{\partial y} \sin \theta \\ \frac{\partial f}{\partial r} &= f_x \cos \theta + f_y \sin \theta\end{aligned}\tag{8.17}$$

Valoarea maximă a acestei derivate, calculate după unghiul  $\theta$  este determinată de ecuația

$$\frac{\partial}{\partial \theta} \left( \frac{\partial f}{\partial r} \right) = -f_x \sin \theta + f_y \cos \theta = 0$$

ce are soluția evidentă:

$$\theta_0 = \arctan \left( \frac{f_y}{f_x} \right)\tag{8.18}$$

Pe această direcție, 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 parțiale  $f_x$  și  $f_y$ , calcularea modulului gradientului maxim (8.19) și a direcției acestuia (8.18). Valoarea gradientului maxim din fiecare punct al imaginii este apoi comparată cu un prag fixat: dacă pragul este depășit

(deci gradientul maxim în pixelul respectiv este suficient de important) atunci pixelul testat este pixel de contur.

Realizarea derivatelor parțiale după direcțiile orizontală și verticală implică translația în discret a lui  $f_x$  și  $f_y$ :

$$f_x = \frac{\partial f}{\partial x} = \frac{\Delta f(m, n)}{\Delta m}$$

$$f_y = \frac{\partial f}{\partial y} = \frac{\Delta f(m, n)}{\Delta n}$$

Aceste derivate parțiale discrete pot avea mai multe implementări:

$$f_x = f(m, n) - f(m + 1, n), f_y = f(m, n) - f(m, n + 1) \quad (8.20)$$

$$f_x = f(m - 1, n) - f(m, n), f_y = f(m, n - 1) - f(m, n) \quad (8.21)$$

$$f_x = f(m - 1, n) - f(m + 1, n), f_y = f(m, n - 1) - f(m, n + 1) \quad (8.22)$$

Toate expresiile date de (8.20), (8.21), (8.22) sunt combinații liniare ale valorilor unor pixeli din imagine, situați în vecinătatea pixelului curent din poziția  $(m, n)$ . Deci toate aceste operații se pot realiza prin filtrări liniare cu măști potrivite: (8.23) pentru (8.20), (8.24) pentru (8.21), (8.25) pentru (8.22).

$$W_x = \begin{pmatrix} \blacksquare & -1 \end{pmatrix}, W_y = \begin{pmatrix} \blacksquare \\ -1 \end{pmatrix} \quad (8.23)$$

$$W_x = \begin{pmatrix} 1 & \blacksquare \end{pmatrix}, W_y = \begin{pmatrix} 1 \\ \blacksquare \end{pmatrix} \quad (8.24)$$

$$W_x = \begin{pmatrix} 1 & \blacksquare & -1 \end{pmatrix}, W_y = \begin{pmatrix} 1 \\ \blacksquare \\ -1 \end{pmatrix} \quad (8.25)$$

Schema bloc a extragerii de contururi este reprezentată în figura 8.6.

Harta de orientări este o imagine care conține, pentru fiecare pixel, orientarea gradientului de modul maxim în punctul respectiv, și este în general folosită la prelucrarea suplimentară a contururilor (conectare de contururi, extragere direcțională de contururi). Harta de contururi este o imagine binară în care punctele marcate (puncte-obiect) corespund poziției punctelor de contur (puncte cu gradient de modul mare). O simplificare uzuală practică este înlocuirea normei L2 din calculul modului maxim al gradientului (8.19) cu norma L1, ceea ce conduce la aproximarea:

$$\left( \frac{\partial f}{\partial r} \right)_{\max} \approx |f_x| + |f_y|$$

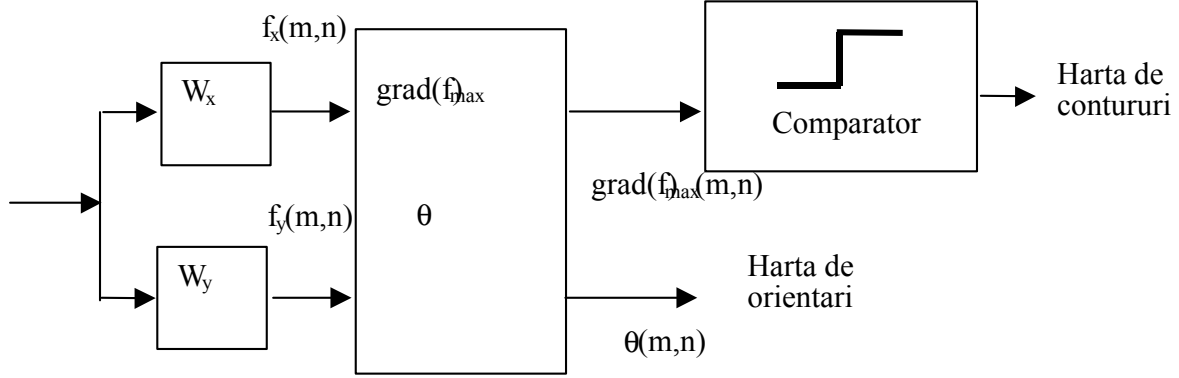


Fig. 8.6: Schema bloc a extractorului de contururi bazat pe metoda de gradient.

Folosirea măștilor de derivare pe verticală și orizontală prezentate are însă serioase neajunsuri: dimensiunea lor mică face ca rezultatele să fie extrem de sensibile în prezența zgomotului. În aceste condiții 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 micșorarea contrastului frontierelor obiectelor din imagine (efectul de încetșoare, sau *blur*). Pentru ca în aceste condiții detecția contururilor să nu fie afectată, trebuie ca operația de mediere prin care se realizează netezirea să se facă pe o direcție perpendiculară direcției contururilor căutate [3]. Atunci derivarea pe verticală se combină cu o operație de netezire cu mască orizontală  $\begin{pmatrix} 1/3 & \boxed{1/3} & 1/3 \end{pmatrix}$  și derivarea pe orizontală se combină cu o operație 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 (și nu neapărat medieri aritmetice), care să acorde o mai mare importanță pixelului curent prelucrat, ca de exemplu  $\frac{1}{c+2} \begin{pmatrix} 1 & \boxed{c} & 1 \end{pmatrix}$  și se preferă folosirea operatorilor de derivare simetrici, de tipul (8.25). Ceea ce rezultă pentru operatorii de derivare orizontală și verticală sunt măștile:

$$W_x = \begin{pmatrix} 1 & 0 & -1 \\ c & \boxed{0} & -c \\ 1 & 0 & -1 \end{pmatrix}, W_y = \begin{pmatrix} 1 & c & 1 \\ 0 & \boxed{0} & 0 \\ -1 & -c & -1 \end{pmatrix} \quad (8.26)$$

Prin particularizarea valorilor constantei de ponderare  $c$  se pot obține 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 măștii de filtrare este neesențială, întrucât condiția de normare ce trebuie îndeplinită este cea pentru filtre de contrastare (derivare) (3.9): suma coeficienților măștii să fie nulă. Figura 8.7 prezintă

harta de intensitate a tranzițiilor (modulul maxim al gradientului,  $(\frac{\partial f}{\partial r})_{\max}$ ) iar figura 8.8 prezintă harta binară de contururi extrasă prin compararea hărții de intensități cu un prag fixat (binarizarea hărții de intensitate).



Fig. 8.7: Harta de intensitate a contururilor (modulul maxim al gradientului) calculat cu măști Prewitt pentru imaginea “lena”.



Fig. 8.8: Harta binară de contururi extrasă din harta de intensități precedentă.

Informația de orientare este în general folosită în etape următoare ale prelucrării; unghiurile determinate după (8.18) oferă un unghi “exact” al direcției conturului în punctul curent, calculat cu un efort semnificativ de calcul (împărțire și calcul de arctangentă). În practică, această informație este prea exactă: pe grila pătrată de eșantionare nu se pot reprezenta cu ușurință drepte continue după orice direcție<sup>5</sup>; câteva direcții sunt fa-

<sup>5</sup>Problema trasării figurilor geometrice oarecari (inclusiv a dreptelor) este rezolvată de grafica pe

vorizate și ușor de utilizat (vertical, orizontal, cele două diagonale). În acest caz se poate măsura în fiecare modulul gradientului după aceste câteva direcții importante, și apoi se poate alege direcția după care acest modul este maxim. Acesta este principul operatorilor compas.

Un operator compas este definit de un număr de măști de derivare (corespunzătoare în continuare unor filtrări liniare) pe direcțiile principale (vertical, orizontal, cele două diagonale), în cele două sensuri. Kompasul clasic are  $D = 8$  măști de filtrare (identice două câte două, mai puțin semnul), fiecare dintre ele realizând o derivare după o direcție multiplu de  $45^\circ$ . Schema bloc a unui operator compas este prezentată în figura 8.9; se remarcă faptul că, odată determinată valoarea maximă a modului gradientului în pixelul curent  $(m, n)$ , obținerea hărții de contururi se face ca și la un operator de gradient clasic.

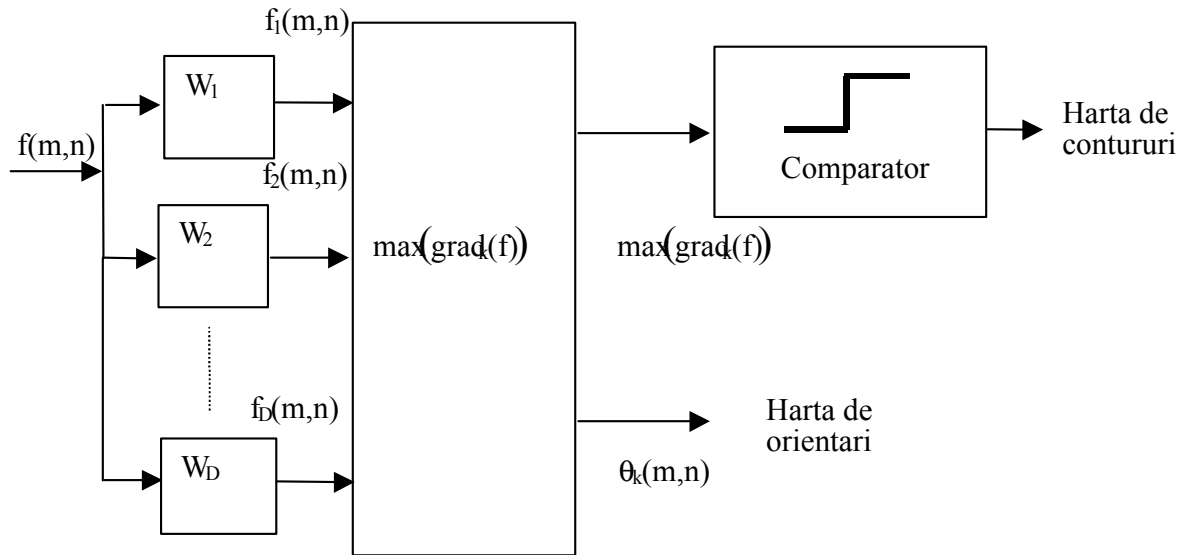


Fig. 8.9: Schema bloc a unui operator compas de extragere a conturilor.

Un exemplu de măști de derivare direcțională sunt măștile următoare (indexate după

direcția 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 măști se poate genera pornind de la una dintre măștile Prewitt, prin translații circulare cu o poziție a frontierei măștii în jurul centrului ei; în mod analog se pot obține operatori compas bazați 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, și deci de numărul de translații ale frontierei măștii; pentru o mască pătrată de bază de dimensiune  $N$ , precizia unghiulară a operatorului compas este de  $90^\circ/(N - 1)$ .

Unul dintre principalele dezavantaje ale metodelor de gradient este precizia slabă de localizare a conturului (a centrului tranziției) în condițiile unei pante puțin abrupte a acestuia (tranziții slabe, graduale). Derivata a doua poate fi însă folosită pentru a determina capetele tranziției (cele două extreme), sau pentru a marca centrul tranziției (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 direcții, combinate în laplacian:

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

În cazul discret, măști ce implementează laplacianul sunt măștile  $W_5 - W_7$ , prezentate la capitolul de îmbunătățire a contrastului imaginilor (pag. 37). Precizia sporită a operatorilor laplacieni conduce însă la o sensibilitate crescută în prezența zgomotelor (mai mare decât a operatorilor de gradient). Mai mult, laplacianul nu mai conține informație relativă la direcția tranziției.

## 8.2.2 Alte metode

O clasă importantă de operații neliniare de extragere a conturilor sunt cele bazate pe morfologia matematică. În secțiunea 6.2.3 am prezentat operatori morfologici de extragere a conturilor. Principiul acestora este de a măsura diferențele dintre valorile extreme (minim și maxim) ale vecinătății punctului curent; dacă diferența dintre aceste valori este suficient de mare înseamnă că punctul curent este un punct de contur, aflându-se într-o zonă de tranziție a valorilor pixelilor. Variante ale acestei tehnici de bază se pot obține prin considerarea a mai multe elemente structurante, având diferite forme și dimensiuni.



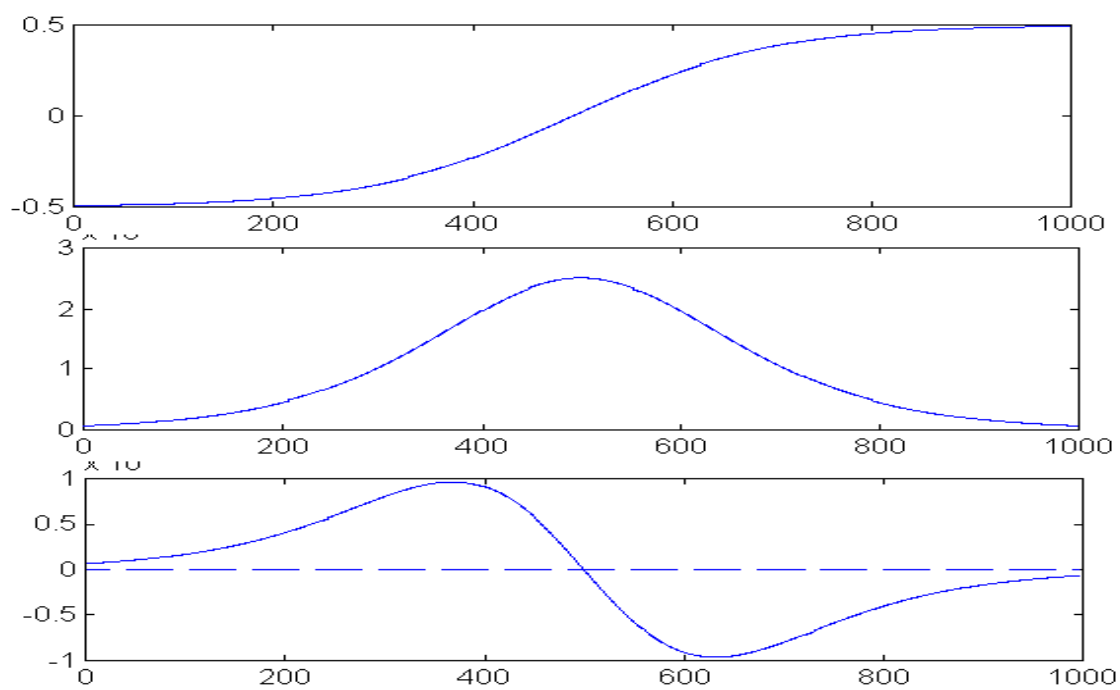


Fig. 8.10: Profil de tranziție graduală; maximul primei derivate nu poate marca cu precizie centrul tranziției; derivata secundă trece prin zero la mijlocul conturului.

## Capitolul 9

# PARAMETRI DE FORMĂ

Prin parametri de formă înțelegem în general orice scalar sau funcție (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 diferențe mari între parametrii de formă ce le sunt asociați. Parametrii de formă compun un fel de fișă de identitate a formei respective, pe baza căreia această formă poate fi recunoscută în mod unic. În mod ideal, acești parametri trebuie să fie invarianți la translație, rotație și scalare. Tehnicile de recunoaștere 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 funcție de două variabile, cu suport compact  $f(x, y) : K \rightarrow \mathbf{R}$ ; în general valorile acestei funcții sunt binare (0 sau 1), descriind deci o parte a unei imagini binare (zona din imagine în care se află obiectul de interes). Atunci funcția poate fi văzută ca o funcție caracteristică a formei, asemănătoare funcției caracteristice a unei mulțimi.

În cele ce urmează vom prezenta câțiva 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 și caracteristice unic unei anume forme, au apărut combinații de tip raport.

Raportul de compacitate (numit și factor de formă [19]) este raportul dintre pătratul

perimetrului și suprafața formei:

$$\kappa = \frac{P^2}{4\pi S} \quad (9.1)$$

Pentru o formă circulară raportul este unitar; cu cât numărul  $\kappa$  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 aceeași valoare a parametrului dat de (9.1).

Excentricitatea sau circularitatea formei (măsura în care forma dată se deosebește de disc) poate fi definită și ca un raport al razelor cercurilor circumscrise ( $R$ ) și înscrise ( $r$ ) formei:

$$c = \frac{R}{r} \quad (9.2)$$

Acest raport este evident unitar în cazul discului; pentru pătrat valoarea sa este de  $c = 1.412$ .

## 9.2 Momente statistice și invarianți

Interpretând funcția caracteristică a formei ca pe o funcție de densitate de probabilitate bidimensională, putem defini momentele statistice asociate celor două variabile aleatoare (ce sunt coordonatele punctelor formei) [17]:

$$m_{pq} = \iint_K f(x, y) x^p y^q dx dy, \quad p, q = 0, 1, 2, \dots \quad (9.3)$$

Scalarul  $m_{pq}$  (momentul de ordin  $p, q$  sau  $p + q$ ) este proiecția funcției  $f(x, y)$  pe polinoamele  $x^p$  și  $y^q$  ale bazei complete de polinoame. Teorema reprezentării cu momente afirmă că mulțimea infinită de momente  $m_{pq}$  determină în mod unic  $f(x, y)$  și reciproc.

În cazul imaginilor binare, coordonatele sunt discrete și funcția este o funcție caracteristică; formula momentelor (9.3) devine

$$m_{pq} = \sum_{f(x,y) \neq 0} x^p y^q \quad (9.4)$$

Cum caracterizarea unei forme printr-o serie infinită de numere (așa cum cere teorema reprezentării cu momente) nu este posibilă, în practică se folosesc serii de momente truncheate până la un ordin maxim fixat  $N$  ( $p + q \leq N$ ). Acestea însă caracterizează o altă funcție,  $g(x, y)$ , o aproximare a lui  $f(x, y)$ . Această aproximare este dată de o combinație liniară a polinoamelor bazei, ponderate cu scalarii necunoascuți  $g_{pq}$ :

$$g(x, y) = \sum_{p+q \leq N} \sum g_{pq} x^p y^q \quad (9.5)$$

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 ecuații cuplate ce se formează (a se vedea [9]), se pot obține relațiile căutate; calculul trebuie însă refăcut, din cauza cuplării ecuațiilor, ori de câte ori se dorește trecerea la o aproximare mai bună a formei  $f$ , măbind valoarea lui  $N$ . Această cuplare provine din cauza folosirii unei baze neortogonale (așa cum este familia de polinoame  $x^p y^q$ ) pentru calculul momentelor; problema a fost rezolvată prin folosirea proiecțiilor 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 invarianță căutate; de aceea au fost introduse momente statistice invariante [19], [9].

Momentele statistice invariante la translație sunt momentele statistice centrate:

$$\mu_{pq} = \sum_{f(x,y) \neq 0} (x - \bar{x})^p (y - \bar{y})^q \quad (9.6)$$

Momentele statistice invariante la translație și scalare sunt definite de:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^\gamma}, \quad \gamma = 1 + \frac{p+q}{2} \quad (9.7)$$

Invarianții la translație, scalare și rotație ai unei forme, obținuți în condițiile folosirii unor momente statistice de ordin cel mult 3 ( $N = 3$ ), sunt în număr de 7 și sunt exprimați de:

$$\Phi_1 = \eta_{20} + \eta_{02} \quad (9.8)$$

$$\Phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \quad (9.9)$$

$$\Phi_3 = (\eta_{30} - 3\eta_{12})^2 + (\eta_{03} - 3\eta_{21})^2 \quad (9.10)$$

$$\Phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{03} + \eta_{21})^2 \quad (9.11)$$

$$\Phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12}) [(\eta_{30} + \eta_{12})^2 - 3(\eta_{03} + \eta_{21})^2] + \quad (9.12)$$

$$+ (\eta_{03} - 3\eta_{21})(\eta_{03} + \eta_{21}) [(\eta_{03} + \eta_{21})^2 - 3(\eta_{30} + \eta_{12})^2]$$

$$\Phi_6 = (\eta_{20} - \eta_{02}) [(\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{03} + \eta_{21}) \quad (9.13)$$

$$\Phi_7 = (\eta_{30} - 3\eta_{21})(\eta_{03} + \eta_{21}) [(\eta_{03} + \eta_{21})^2 - 3(\eta_{30} + \eta_{12})^2] - \quad (9.14)$$

$$- (\eta_{03} - 3\eta_{21})(\eta_{30} + \eta_{12}) [(\eta_{30} + \eta_{12})^2 - 3(\eta_{03} + \eta_{21})^2]$$

Inițial (mijlocul anilor '60) acești invarianți au fost folosiți pentru recunoașterea caracterelor mari de tipar, cu rezultate modeste. Eficiența lor constă însă în modul rapid de calcul și posibilitatea de a le utiliza cu succes pentru recunoașterea formelor geometrice convexe.

Folosind momentele invariante, se mai pot deduce alte atribute: excentricitatea suprafeței (9.15), care măsoară gradul de uniformitate al distribuției punctelor forme în jurul centrului de greutate și orientarea suprafeței, caracterizată de unghiul  $\theta$  față de orizontală al axei față de care momentul inerției al forme este minim (9.16).

$$\varepsilon = \frac{\Phi_2}{\mu_{00}} \quad (9.15)$$

$$\theta = \frac{1}{2} \arctan \frac{2\mu_{11}}{\mu_{20} - \mu_{02}} \quad (9.16)$$

### 9.3 Semnătura forme

Semnătura unei forme este o funcție scalară de o variabilă, asociată unei forma plane. Semnătura este definită de distanța de la un punct de referință fixat  $\mathbf{x}$  (în general centrul de greutate al forme) la fiecare punct de pe conturul (frontiera) forme. Această distanță este exprimată (sau măsurată) în funcție de unghiul la centru  $\theta$  realizat de punctul curent de pe contur cu axa orizontală de referință,  $d_{\mathbf{x}}(\theta)$  sau în funcție de abscisa curbilinie  $\rho$  (lungimea conturului cuprins între punctul curent și punctul în care axa de referință intersectează conturul),  $d_{\mathbf{x}}(\rho)$ . Semnătura forme este o reprezentare reversibilă (cunoscând semnătura se poate reconstrui conturul obiectului). Figurile 9.1 și 9.2 prezintă semnăturile unor forme poligonale.



Fig. 9.1: Semnătura unui pătrat ca funcție de unghiul la centru; punctul de referință este centrul de greutate, axa de referință este orizontală.

Este posibil ca, pentru formele concave, semnătura în funcție de unghiul la centru să nu poată fi calculată, deoarece, pentru anumite unghiuri  $\theta$ , intersecția dreptei de direcție  $\theta$  cu conturul să fie formată din mai multe puncte. Această problemă nu apare în cazul semnăturii în funcție de abscisa curbilinie. Ca o restricție generală, în cazul folosirii semnăturii bazate pe unghi, trebuie ca punctul de referință  $\mathbf{x}$  să aparțină nucleului forme. Nucleul forme  $K$ ,  $Ker(K)$  este definit prin:

$$Ker(K) = \{\mathbf{x} | \forall \mathbf{y} \in K, [\mathbf{xy}] \subseteq K\} \quad (9.17)$$

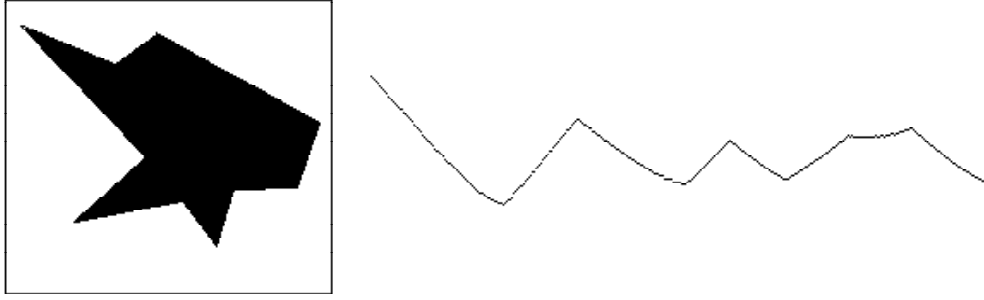


Fig. 9.2: Semnătura unei forme poligonale în funcție de abscisa curbilinie; punctul de referință este centrul de greutate, axa de referință este orizontală.

unde  $[\mathbf{xy}]$  semnifică segmentul de dreaptă definit de punctele  $\mathbf{x}$  și  $\mathbf{y}$ . În cazul formelor concave, nucleul formeii este o mulțime vidă.

Folosind semnătura formeii, se pot defini parametri de tip geometric. Raportul de simetrie este definit ca

$$Y(K) = \sup_{\mathbf{x} \in \mathbf{K}} \inf_{\theta \in [0; \pi]} \frac{d_{\mathbf{x}}(\theta)}{d_{\mathbf{x}}(\theta + \pi)} \quad (9.18)$$

Se observă că nu s-a făcut nici o presupunere privind convexitatea formeii  $K$ , dar unicitatea distanțelor  $d_{\mathbf{x}}(\theta)$  și  $d_{\mathbf{x}}(\theta + \pi)$  implică calcularea raportului de simetrie după punctele ce aparțin nucleului formeii.

Raportul de circularitate este definit ca:

$$C(K) = \frac{\sup_{\theta} (d_{\mathbf{x}}(\theta) + d_{\mathbf{x}}(\theta + \pi))}{\inf_{\theta} (d_{\mathbf{x}}(\theta) + d_{\mathbf{x}}(\theta + \pi))} \quad (9.19)$$

Funcția  $h_K(\theta) = d_{\mathbf{x}}(\theta) + d_{\mathbf{x}}(\theta + \pi)$  se numește suportul formeii, și este diametrul formeii pe direcția  $\theta$ .

În [5] s-a propus aproximarea formelor prin dezvoltarea în serie Fourier a semnăturii acestora și truncherea reprezentării (tehnică utilizată și într-o aplicație clasică de recunoaștere a conturului unor tipuri de avioane). O altă posibilă aplicație a semnăturii pleacă de la observația că, pentru o formă poligonală, în semnătură, poziția vârfurilor este marcată de puncte unghiulare (a se urmări figurile 9.1 și 9.2). Atunci aceasta poate fi o metodă de aproximare poligonală a unei forme oarecare (găsirea vârfurilor unui poligon ce o aproximează cât mai bine).

## 9.4 Skeletoane morfologice și generalizate

Skeletonul este o reprezentare bidimensională simplificată, echivalentă, a unei forme. Pentru o formă  $A$  oarecare se definește discul maximal în  $A$ , de centru  $\mathbf{x}$  și rază  $r$ ,  $B_{\mathbf{x}}(r)$  ca fiind discul caracterizat de:

$$B_{\mathbf{x}}(r) \subseteq A \quad (9.20)$$

$$B_{\mathbf{x}}(r) \subseteq B_{\mathbf{x}'}(r') \subseteq A \Leftrightarrow \begin{cases} r = r' \\ \mathbf{x} = \mathbf{x}' \end{cases} \quad (9.21)$$

Deci discul maximal trebuie să fie inclus în formă (9.20) și 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 mulțimea 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 spațiul discret se poate face prin formula Lantuejoul [15]; skeletonul formei  $A$ ,  $SK(A)$ , este format din reuniunea unui număr finit de seturi skeleton:

$$SK(A) = \bigcup_{n=0}^{N_{\max}} S_n(A) \quad (9.22)$$

$$S_n(A) = (A \ominus nB) - (A \ominus nB) \circ B \quad (9.23)$$

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  $B$ ,  $nB = B \oplus \dots \oplus B$  (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 și informația de apartenență a fiecărui punct la un anumit set skeleton.

Reconstrucția formei din skeleton se face după formula

$$A = \bigcup_{n=0}^{N_{\max}} S_n(A) \oplus nB \quad (9.24)$$

Se pot realiza și reconstrucții parțiale (aproximări ale mulțimii  $A$ ) 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 reconstrucție:

$$\widetilde{A}_k = \bigcup_{n=k}^{N_{\max}} S_n(A) \oplus nB \quad (9.25)$$

Skeletonul este invariant la translație, nu este conex (chiar dacă forma  $A$  este conexă; a se vedea figura 9.5), nu este comutativ cu operația de reuniune a formelor (a se vedea figura 9.6). Transformarea skeleton este idempotentă ( $SK(SK(A)) = SK(A)$ ) și 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 recunoașterea formelor este restricționată de puterea sa sensibilitate la zgomote (o mică schimbare a formei duce la o modificare semnificativă a skeletonului<sup>1</sup>), și de variația la rotația și scalarea obiectelor (pentru reprezentări în spațiul discret). În același timp, folosirea elementului structurant de tip disc unitar aduce o puternică dependență de metrica folosită în definirea sa (să nu uităm că într-un

<sup>1</sup>Exemplul clasic este de a considera un disc fără centru; în acest caz skeletonul este o coroană circulară situată la jumătatea razei.



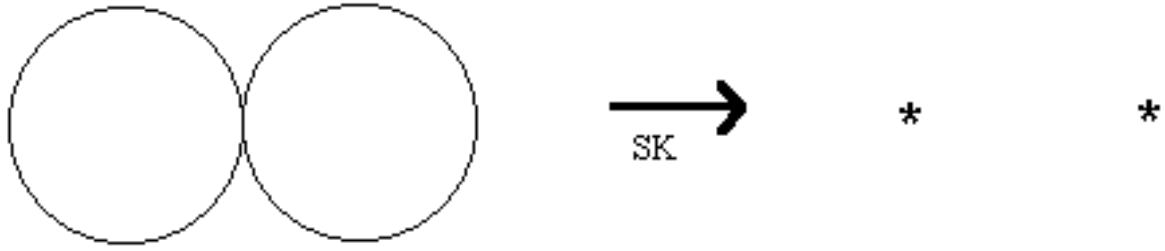


Fig. 9.5: Skeletonul nu este conex.

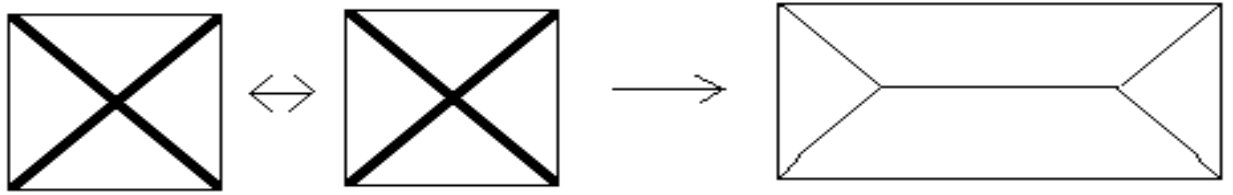


Fig. 9.6: Transformata skeleton nu este comutativă cu reuniunea.

spațiu discret, metrica Euclidiană nu este cea mai favorabilă) și 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 noțiunea de metrică - elementele structurante generalizate.

### 9.4.2 Skeletonul generalizat

Fie  $\{G_i\}$  un set de mulțimi, numit set generator. Elementele structurante generalizate sunt definite recurent prin:

$$B_i = B_{i-1} \oplus G_i, i = 1, 2, \dots \quad (9.26)$$

$$B_0 = 0_n$$

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 construcția setului de elemente structurante generalizate rezultat dintr-un set generator cu perioadă 1 (deci compus dintr-o singură mulțime),  $E_1 = \{(-1, -1), (-1, 0), (0, -1), (0, 0)\}$ .

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

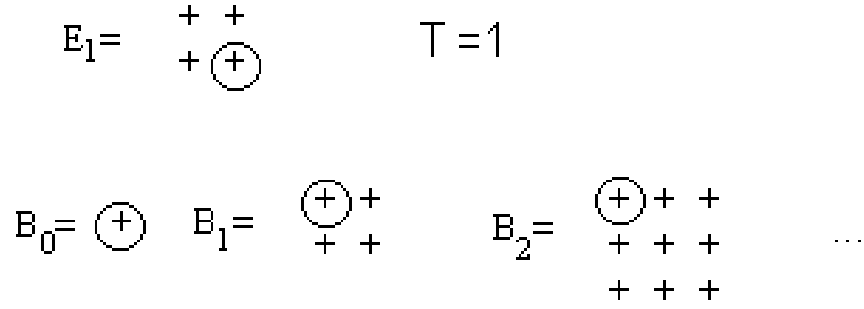


Fig. 9.7: Construcția unui set de elemente structurante generalizate.

distanțe” (a se vedea și figura 9.8):

$$D(\mathbf{x}) = \begin{cases} 0, & \text{dacă } \mathbf{x} \notin A \\ n, & \text{dacă } (B_{n-1})_{\mathbf{x}} \subseteq A \text{ și } (B_n)_{\mathbf{x}} \not\subseteq A \end{cases} \quad (9.27)$$

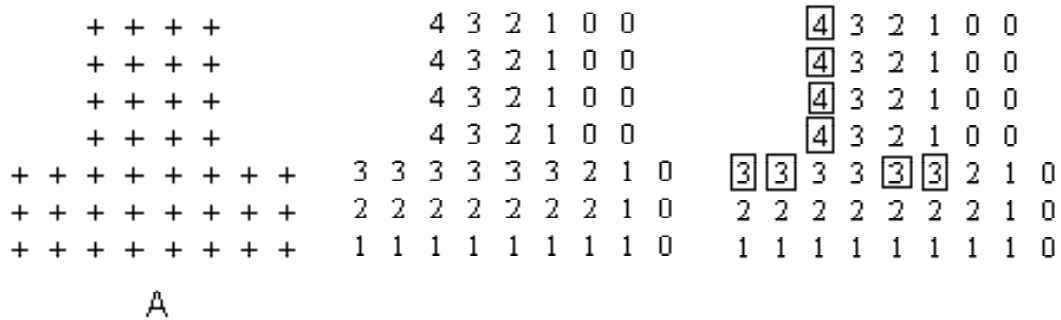


Fig. 9.8: Harta de distanțe a unui obiect construită pe baza setului de elemente structurante generalizate prezentat anterior și skeletonul generalizat corespunzător.

Skeletonul generalizat al unei forme este mulțimea originilor elementelor structurante generalizate maximale în formă<sup>2</sup>:

$$GSK(A) = \left\{ \mathbf{x} \in A \mid (B_{D(\mathbf{x})-1})_{\mathbf{x}} \not\subseteq (B_{D(\mathbf{y})-1})_{\mathbf{y}}, \forall \mathbf{x} \neq \mathbf{y} \right\} \quad (9.28)$$

Folosind harta de distanță anterioară, skeletonul generalizat obținut este prezentat în figura 9.8.

<sup>2</sup>Se poate remarca similitudinea cu definiția skeletonului, în care s-au înlocuit noțiunile: 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 ȘI HARDWARE

Principiile esențiale legate de implementările practice ale sistemelor de prelucrarea și analiza imaginilor urmăresc două direcții, cu dezvoltare corelată: implementările software și dispozitivele hardware (de accelerare). După cum am arătat în capitolul introductiv, la descrierea structurii tipice a unui sistem de prelucrarea și analiza imaginilor, marea majoritate a implementărilor folosesc ca suport fizic pentru unitatea centrală de prelucrare un calculator obișnuit (compatibil PC); ceea ce îl particularizează este pachetul de programe rulate. Putem distinge două categorii fundamentale de astfel de programe: programe strict dependente de aplicație și programe de uz general.

Un program dependent de aplicație realizează doar algoritmi specifici operației pe care o execută sau supraveghează. Interacțiunea cu operatorul uman este minimă și calificarea acestuia nu este necesar să o depășească 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 și care se desfășoară în condiții ambiante (iluminare, poluare vizibilă – particule, fum) relativ constante și nu sunt portabile (fiind în general optimizate pentru o anumită structură hardware).

Programele de uz general permit efectuarea unui mare număr de operații de prelucrarea și analiza imaginilor, cu numeroși parametri reglabili. Interacțiunea cu operatorul uman este mare și acesta trebuie să aibă o calificare superioară. Programele de acest tip nu sunt de timp real și în general sunt cuplate *off-line* cu instalațiile tehnologice propriu-zise, făcând parte mai ales din dotarea laboratoarelor de cercetare și de analiza calității și conformității. Interfața utilizator este cea care crează diferența între două categorii de programe, în ceea ce privește modalitatea în care operatorul specifică succesiunea de operații de executat. Din acest punct de vedere vom face distincția între *menu-driven* și *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 afișare. Tipurile de operații se aleg din meniuri sau bara de butoane. Asemenea soluții corespund majorității sistemelor software comerciale de grafică și *imaging* (de tipul Adobe Photoshop, Corel Draw, Paint Shop Pro). Operațiile realizate sunt orientate mai ales către aspectul grafic (publicistic) al imaginilor, referindu-se în special la operații de filtrare, modificare a contrastului, pseudocolorare, decupare de regiuni. Programele cu comandă *menu-driven*, cu rezultatele intermediare parcurgând etapele operațiilor din fereastră în fereastră, sunt direct derivate din proiectarea obiectelor de interfață în sistemele de tip Windows.

Implementările de tip *flow-chart* permit construirea grafică interactivă a unui lanț de operații aplicate unei imaginii inițiale. Fiecare operație este un bloc funcțional caracterizat de intrări, ieșiri și parametri de control; blocurile funcționale sunt interconectate, implementând fluxul de operații pe care îl va parcurge imaginea. Procesul de comandă înseamnă selectarea imaginii inițiale și acționarea unui buton de start. Asemenea programe sunt tipice sistemelor de calcul mari (stații de lucru), pentru care produsul Khoros este un standard de fapt; un produs similar pentru PC este programul AdOculus (figura 10.1).

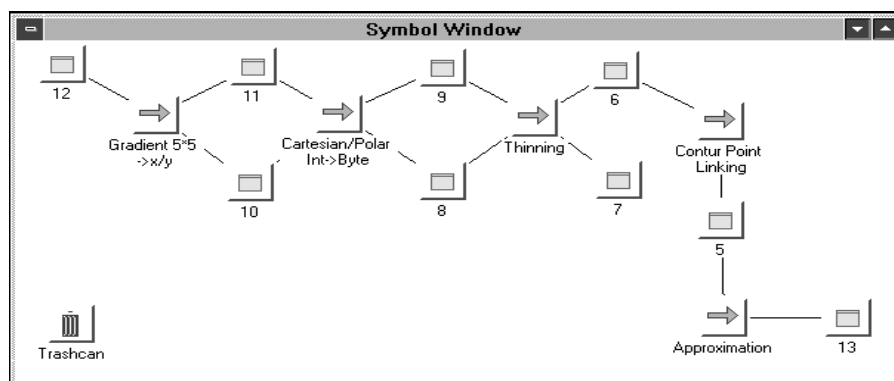


Fig. 10.1: Specificarea fluxului de operații în fereastra de comandă a programului AdOculus.

Ca o categorie intermediară între comenzile meniu și 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 operațiile inițial realizate prin software-ul de aplicație 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 afișaj). După ce, inițial, problema era rezolvată

de programele aplicație, gestiunea culorilor a trecut în grija sistemului de operare, pentru ca, recent, să apară plăcile grafice inteligente. Aceasta este de altfel una dintre tendințele actuale: evoluția accesoriilor (plăci specializate, dispozitive de achiziție) inteligente, deci cu facilități de prelucrare integrate. Senzorii de imagine CCD sunt una dintre țintele predilecte ale acestei dezvoltări, înglobând cipuri de compresie și transmisie a imaginilor, comandă de urmărire automată a țintei, operații de recunoaștere.

Deși programarea orientată pe obiecte este un concept destul de nou, evoluția sa a fost deosebit de spectaculoasă. Probabil că principalul atu al acestei evoluții a fost apariția 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 aplicații Windows va face în mod clar apel la structuri predefinite - clasele de bază aplicație, fereastră, buton, ș.a.m.d. Atunci devine destul de normal să fie proiectate și datele specifice ale aplicației particulare tot ca structuri de tip obiecte.

Alegerea unui limbaj specific de programare nu mai este totuși o problemă esențială; 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++ și Borland Pascal cu obiecte) și chiar apar limbaje noi (Java, care, deși se revendică ca o aplicație pur distribuită pentru folosirea rețelelor și în special a Internetului, nu este cu mult diferită ca structură de un C cu clase). Principala evoluție a mediilor de programare nu a fost însă legată de modificarea limbajului în sine (și deci a caracteristicilor de compilare a codului) ci modificarea stilului în care se crează aplicația specifică, și mai precis, aspectul de interfață.

Proiectarea unei interfețe pentru Windows înseamnă definirea unor acțiuni asociate evenimentelor produse în sistem (clicuri de mouse, apăsări de taste) și construirea elementelor grafice specifice (ferestre de afișare și de dialog, meniuri, introducere date, etc.). Această activitate a fost simplificată la maximum prin apariția constructoarelor vizuale de aplicații, în care interfața 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 construcției; tot ceea ce trebuie să facă programatorul este să lege sursa ce descrie aplicația și acțiunea 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 și atribuția de a scrie algoritmi specifici; ușurința construcției vizuale este legată strict de construcția interfeței aplicației, deci aspect grafic și eventual partea de preluare a datelor. Programatorul va dezvolta restul aplicației ca pentru un compilator clasic.

Concluzia ce rezultă din această scurtă discuție este aceea că, pentru orice aplicație, *trebuie separată partea de interfață de partea de calcul specific*. În cazul unui program de prelucrarea și analiza imaginilor acesta înseamnă că trebuie făcută o proiectare la nivelul structurii de date (matricea ce conține valorile pixelilor din imagine și prelucrările specifice

asupra acestora) și o proiectare la nivelul interfeței (care să specifice cum se va face afișarea imaginilor pe ecran, cum vor fi scrise pe disc, cum se va face un eventual transfer dinamic al datelor de la sau către alte aplicații). Majoritatea covârșitoare a sarcinilor legate de interfață pot fi rezolvate fără a cunoaște nimic despre operațiile specifice prelucrării imaginilor și despre modul în care o imagine este reprezentată în memoria de lucru a calculatorului. Să considerăm de exemplu problema afișării unei imagini într-o fereastră. Fără îndoială că cel mai simplu mod de afișare este folosirea imaginii ca un *canvas*, înglobat în obiectul de tip fereastră, și de a cărui afișare se ocupă sistemul Windows. Această abordare exclude însă accesul la conținutul imaginii; aceasta trebuie deci separată de fereastra de afișare. Este poate deci preferabilă memorarea imaginii ca o matrice și transformarea acesteia într-o structură *bitmap* atașată 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 și moștenire pe care le aduce acest stil de programare. Moștenirea corespunde cazului unor imagini cu structură din ce în ce mai elaborată, pentru care se pot face tot mai multe operații (de tip analiză de formă și clasificare, de exemplu). Polimorfismul corespunde realizării unor operații cu nume (sau metode de implementare) identice care să aibă comportări diferite, în funcție de tipul datelor cărora li se aplică (funcția de histogramă poate produce, de exemplu, funcția de densitate de probabilitate a câmpului aleator imagine, în cazul imaginilor cu nivele de gri, aria obiectelor, în cazul imaginilor binare și numărul de componente conexe, în cazul imaginilor binare etichetate). De asemenea, aspectul legat de re folosirea codului nu este neglijabil (să nu uităm, că, de exemplu, pentru filtrarea în domeniul spațial, un numitor comun al diverselor tipuri de filtre este tehnica de tip fereastră glisantă).

Tehnologia Java, aduce, pe 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 mașină pe care este disponibil un *browser* Internet. Marile avantaje anunțate pentru Java (cod portabil, execuție paralelă, protecție la manipularea defectuoasă a zonelor de memorie, protecția datelor) sunt contrabalansate de codul executabil relativ lent și de dimensiunile mari ale codului sursă.

Java este însă interesantă pentru prelucrarea și analiza imaginilor prin perspectiva deschisă de *multi-threading* – existența și gestionarea de către o aplicație oarecare a mai multe fire de execuție 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 aplicații pe sisteme cu unic procesor, fără sisteme de operare *multi-tasking*. De altfel, puține sisteme fizice multiprocesor au fost realizate pentru prelucrarea de imagini; sistemele de prelucrarea și analiza imaginilor au utilizat în general mașini cu paralelism masiv (SIMD/ SPMD, *transputer farm*) sau grupuri de calculatoare (*cluster processing*).

Exploatarea paralelismului este un merit pe care prelucrarea imaginilor și l-a arogat încă de la începuturi; operațiile tipice de prelucrarea imaginilor sunt operații relativ simple, cu un număr mare de instanțe (pentru fiecare pixel al imaginii se face ceva), folosind date puțin redundante (suprapunerea între pozițiile alăturate ale ferestrelor de filtrare este în general mică) și, uneori, informație globală (cazul histogramei sau a filtrărilor în domeniul de frecvență). Se pot distinge astfel două nivele de paralelism: un paralelism masiv, intrinsec structurii imaginii, legat de nivelul pixel, și un paralelism ascuns, specific operațiilor de prelucrare (ca de exemplu realizarea operațiilor morfologice prin (6.3) și (6.6) sau implementarea paralelă a FFT).

În concluzie, putem afirma că prelucrarea și analiza imaginilor a reușit să câștige teren ca urmare a realizărilor spectaculoase ale tehnologiei electronice și informaticii. Creșterea continuă a puterii de calcul disponibile în unitățile de prelucrare ale calculatoarelor va transforma probabil în viitorul apropiat prelucrarea și analiza imaginilor dintr-o anexă nebuloasă și exotică a aplicațiilor speciale într-o soluție fiabilă de larg consum industrial.

# Bibliografie

- [1] Buzuloiu, V.: *Prelucrarea imaginilor: note de curs*, Universitatea "Politehnica" București, 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 - Continuous 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 Bildverarbeitung in NRW*, Düsseldorf, 1995
- [11] Ministerium für Wirtschaft, Mittelstand und Technologie des Landes Nordrhein-Westfalen: *Produkte und Dienstleistungen 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 mathématique et réseaux 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 Informației*, Ed. Didactică și Pedagogică, București, 1984
- [17] Vertan, C., Gavăt, I., Stoian, R.: *Variabile aleatoare: principii și aplicații*, Editura Printech, București, 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