ANALIZA IMAGINILOR: ÎNDRUMAR DE LABORATOR

Constantin VERTAN, Mihai CIUC, Marta ZAMFIR

2001

Cuprins

Capitolul 1

Introducere

Odată cu importantele dezvoltări tehnologice din ultimii ani (tehnologie electronică, informatică şi a comunicaţiilor), dispozitivele de achiziţie a imaginilor şi sistemele de calcul de uz general au devenit tot mai răspândite şi mai la îndemână. În mod firesc, dorinţa de a dispune de sisteme autonome înzestrate cu vedere artificială, care să realizeze în mod independent diferite sarcini specifice a revenit în actualitate, limitată însă la obiective mai simple şi mai pragmatice faţă de declaraţiile oarecum bombastice ale deceniului trecut ("Prelucrarea imaginilor permite dezvoltarea maşinii totale de vedere artificială, capabilă să reproducă fuuncţiile vizuale ale oricărei vieţuitoare." [7]).

Prelucrarea şi analiza imaginilor cuprinde ansamblul de tehnici şi metode de achiziţie, stocare, afişare, modificare şi exploatare a informaţiei vizuale cuprinsă în imagini. În particular, analiza imaginilor se referă la capacitatea de a descrie, înţelege şi recunoaşte scene, obiecte din scene şi legăturile dintre acestea. Din punct de vedere funcţional, analiza imaginilor transformă o imagine de intrare într-o descriere.

În mod esenţial, analiza imaginilor statice înseamnă descompunerea acestora în elementele constitutive (prin segmentare) şi apoi caracterizarea elementelor individuale prin parametri de formă. Atât segmentarea cât şi descrierea pot fi realizate interpretând obiectele fie ca regiuni compacte, fie numai ca frontiere (contururi). Organizarea logică a prezentării domeniului de analiză a imaginilor va urmări deci aceste patru direcţii esenţiale.

Lucrarea de faţă este un ghid pentru realizarea experimentelor de laborator de analiză a imaginilor. Capitolele lucrării tratează câte o problemă bine definită: segmentarea pe regiuni fără folosirea de informaţii apriori, segmentarea pe regiuni cu folosirea de modele, creşterea regiunilor, etichetarea imaginilor binare, segmentarea prin extragerea contururilor, descrierea texturilor, a regiunilor compacte şi a contururilor. Anexele oferă informaţii asupra modului de utilizare a suportului software folosit.

Fiecare capitol este compus din trei părţi: un breviar teoretic ce recapitulează noţiunile fundamentale, un ghid de desfăşurare a lucrării, cuprinzând indicaţii pentru realizarea experimentelor concludente pentru înţelegerea şi ilustrarea teoriei şi un set de probleme deschise, a căror rezolvare nu este întotdeauna imediată, ce necesită o atenţie şi un interes suplimentar.

Experimentele propuse se bazează pe folosirea produsului software Matlab, pentru care au fost dezvoltate o serie de funcţii suplimentare (explicitate şi listate în Anexa A) şi pe utilizarea produsului software SAIN de analiză a formelor plane (al cărui ghid de utilizare este prezentat în Anexa B). Funcţiile Matlab prezentate funcţionează pentru versiuni Matlab începând cu 4.0; s-a evitat folosirea funcţiilor extrem de specializate din toolbox-ul de prelucrare a imaginilor sau din versiunile Matlab mai noi (5.3 sau 6.0), în încercarea de a menţine o abordare didactică şi de a oferi posibilitatea de experimentare personală şi celor ce nu dispun de calculatoare foarte performante. Tocmai în ideea de a oferi posibilitatea de realizare particulară a experimentelor, imaginile de test folosite (în majoritate de dimensiune 128 x 128) şi programul SAIN sunt disponibile şi prin Internet la adresa http://alpha.imag.pub.ro/release.

Acest îndrumar de laborator se adresează în primul rând (dar nu în mod exclusiv) studenţilor de la specializările IFIA (Imagini, Forme şi Inteligenţă Artificială) şi II (Ingineria Informaţiei) de la Catedra de Electronică Aplicată şi Ingineria Informaţiei, Facultatea de Electronică şi Telecomunicaţii a Universităţii Politehnica Bucureşti. Lucrarea de faţă însoţeşte cursul de Analiza Imaginilor de la secţia IFIA şi a doua jumătate a cursului de Prelucrarea şi Analiza Imaginilor de la secţia II. Sperăm că acest îndrumar îşi va dovedi utilitatea şi va constitui o completare utilă a cursurilor.

În final autorii doresc să mulţumească în mod deosebit Domnului Profesor Vasile Buzuloiu pentru sprijinul, sfatul şi încurajările acordate.

Capitolul 2

Segmentarea pe histogramă: generalităţi

Scopul acestei lucrări este familiarizarea cu tehnicile de segmentare pe histogramă a imaginilor, şi anume segmentarea cu mai multe praguri, prin alegerea acestora pe minimele histogramei. De asemenea, se urmăreşte folosirea histogramelor ponderate pentru accentuarea separaţiei între modurile histogramei.

2.1 Histograma unei imagini

Pentru o imagine f de $M \times N$ pixeli şi L nivele de gri, histograma este definită (2.1) ca probabilitatea de apariţie în imagine a diferitelor nivele de gri posibile.

$$h(i) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \delta(i - f(m,n)) \quad , \quad i = 0, 1, ...L-1 \tag{2.1}$$

Fiind o funcţie de densitate de probabilitate, histograma oricărei imagini verifică condiţia de normare:

$$\sum_{i=0}^{L-1} h(i) = 1$$

Histograma cumulativă este funcţia de repartiţie a variabilei aleatoare ce reprezintă nivelul de gri al imaginii, şi deci este probabilitatea ca un pixel din imagine să aibă nivelul de gri mai mic sau egal ca un prag fixat:

$$H(i) = \sum_{a=0}^{i} h(a),\ i = 0, 1, ...L-1 \tag{2.2}$$

Orice imagine este o structură bidimensională de date, deci o matrice. Imaginile sunt stocate în fişiere; adeseori fişierele ce conţin imagini au o organizare (structură) specială, descrisă de aşa numitele formate grafice. Fişierele grafice (ce conţin imagini) sunt identificate prin extensiile speciale pe care le au numele acestora. În laboratoarele de analiza imaginilor se vor folosi fişiere având formatele IMG, MFI şi BMP.

În Matlab citirea unei imagini stocate în asemenea fişiere se face cu ajutorul funcţiilor **imgread** (dacă fişierul este de tip IMG, deci are extensia .img), **mfiread** (dacă fişierul este de tip MFI, deci are extensia .mfi) şi **bmp8rd** (dacă fişierul este de tip BMP, deci are extensia .bmp). Modul de folosire a acestor funcţii este:

≫x=imgread('nume fişier imagine', dimensiune);

≫x=mfiread('nume fişier imagine');

≫x=bmp8rd('nume fişier imagine');

Rezultatul execuţiei comenzilor anterioare este acela că se creează (în memoria calculatorului sau în aşa numitul spaţiu de lucru Matlab) matricea x, ce corespunde imaginii stocate în fişierul cu numele 'nume fişier imagine' (apostrofurile precizează faptul că acesta este un şir de caractere). Pentru citirea fişierelor în format IMG este necesară şi precizarea dimensiunii imaginii, în pixeli (imaginea fiind presupusă pătrată).

Observaţie 1 *Pentru fişierele în format IMG disponibile pentru laboratoare, ultimele trei cifre din numele fişierului au semnificaţia dimensiunii (în pixeli) a imaginii conţinute; deci un fişier numit lena_128.img semnifică faptul că este de tip IMG şi conţine o imagine pătrată cu dimensiunea de 128 pixeli.*

Imaginile cu care se lucrează în mod curent sunt imagini cu nivele de gri, reprezentate pe 256 de nivele de cuantizare (deci $L = 256$). Valoarea fiecărui pixel al imaginii (nivelul de gri al fiecărui pixel) este măsura luminanţei punctului respectiv; 0 corespunde negrului şi 255 corespunde albului. Din acest punct de vedere, imaginile sunt imagini de intensitate (valoarea fiecărui punct este proporţinală cu intensitatea luminoasă din punctul considerat). În acelaşi timp însă, valorile întregi ale punctelor se folosesc la afişarea imaginii pentru a recupera culoarea corespunzătoare dintr-un tabel de culoare (tabel de culoare ce conţine cele 256 de nivele de gri folosite), şi deci există şi o a doua interpretare a imaginilor ca imagini indexate. Afişarea unei imagini cu L nivele de gri, reprezentată de matricea x în fereastra curentă, se face cu comanda Matlab:

≫image(x),colormap(gray(L))

Observaţie 2 *Evident, în comanda de mai sus, L trebuie înlocuit cu valoarea lui numerică particulară, ca de exemplu* 256*. De asemenea, trebuie avut în vedere că Matlab nu poate reprezenta imagini cu mai mult de* 256 *de culori (nivele de gri) diferite.*

Pentru calcularea histogramei (2.1) şi a histogramei cumulative (2.2) a unei imagini x cu nivele de gri se foloseşte funcţia **histo**:

≫[h, H]=histo(x);

Dacă nu este necesară şi histograma cumulativă, funcţia se poate apela doar prin:

≫h=histo(x);

Codul Matlab al acestei funcţii se regăseşte în fişierul **histo.m**; calculul efectiv este realizat de:

function [h,hcum]=histo(in,sw)	definiţia funcţiei histo (sw este parametru nefolosit)
[hh,w]=size(in);	se preiau dimensiunile imaginii
h=zeros(1,256);	se iniţializează histograma
for i=1:256	pentru toate nivelele de gri
h(i)=prod(size(find(in==i)));	se numără pixelii ce au un nivel de gri dat
end	
h=h/(w*hh);	se normează histograma
hcum=cumsum(h);	se calculează histograma cumulativă

Vizualizarea histogramei în figura curentă se face prin trasarea graficului definit de vectorul h:

≫plot(h)

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.

2.2 Histograma ponderată a unei imagini

Construirea histogramei ponderate are ca scop accentuarea minimelor locale, pentru o alegere mai uşoară a pragurilor de segmentare (separaţie inter-mod). Histograma pon-

derată este construită prin definirea unei funcţii de ponderare $w()$, care transformă o anumită proprietate locală $p(m,n)$, conform relaţiei (2.3):

$$h_w(i) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} w\left(p(m,n)\right) \delta(i - f(m,n)) \quad , \quad i = 0, 1, ...L-1 \tag{2.3}$$

În această histogramă ponderată contribuţia pixelilor nu mai este unitară, ci dependentă de proprietăţile vecinătăţii acestora, măsurate prin $p(m,n)$. Proprietatea locală cea mai des utilizată este aceea de neuniformitate (de variaţie a nivelelor de gri în vecinătatea fiecărui pixel), măsurată prin laplacianul local $l(m,n)$. Pixelii care se află în zonele de tranziţie (frontierele obiectelor din imagine) au o vecinătate neuniformă, şi deci un laplacian mare în valoare absolută, iar pixelii care se află în interiorul unor regiuni uniforme (interiorul obiectelor) au o vecinătate mai uniformă şi deci un laplacian mai apropiat de valoarea nulă. Pentru a accentua pe histogramă zonele de separaţie inter-mod este necesar ca pixelii ce se află în interiorul unor regiuni uniforme (obiecte) să aibă o contribuţie mai importantă decât contribuţia pixelilor aflaţi în zonele de frontieră. În aceste condiţii, funcţia de ponderare trebuie să fie invers proporţională cu laplacianul local; forma generală a funcţiei de ponderare este dată de relaţia (2.4):

$$w\left(p(m,n)\right) = \frac{1}{\left(1 + |l(m,n)|\right)^k} \tag{2.4}$$

Exponentul k reprezintă un factor suplimentar de reglaj, permiţând obţinerea de diferite grade de reliefare a minimelor histogramei (cu cât k este mai mare, cu atât minimele sunt mai bine puse în evidenţă).

Există un mod de segmentare alternativă, folosind maximele unei histograme ponderate. În acest caz este necesar ca funcţia de ponderare $w()$ să transforme minimele histogramei normale în maxime ale histogramei ponderate. Dacă se foloseşte în continuare laplacianul ca proprietate locală şi o funcţie de ponderare proporţională cu acesta, de exemplu (2.5), histograma ponderată va lua în calcul doar pixelii din imagine a căror vecinătate prezintă variaţii puternice ale nivelelor de gri, în timp ce pixelii aflaţi în interiorul unor regiuni omogene nu vor avea nici o contribuţie. Histograma ponderată astfel construită va prezenta maxime pe nivelele de gri ce corespund zonelor de tranziţie între modurile histogramei normale.

$$w\left(p(m,n)\right) = |l(m,n)| \tag{2.5}$$

Funcţia Matlab coresunzătoare construirii acestei histograme ponderate (şi a histogramei cumulative asociate) este **whisto**:

≫wh=whisto(x);

Codul corespunzător prelucrării de bază este:

function [h,hcum]=histo(in,p,mask)	definiţie funcţie
[hh,w]=size(in);	dimensiuni imagine
h=zeros(1,256);	iniţializare histogramă
in=[zeros(1,w+2);zeros(h,1) in zeros(h,1);zeros(1,w+2)];	bordare imagine cu zerouri
for i=2:hh+1	se filtrează liniar imaginea
for j=2:w+1	iniţială (fără margini)
temp=in(i-1:i+1,j-1:j+1);	vecinătatea fiecărui pixel
la=(temp(:))'*mask(:);	determină o pondere
h(in(i,j))=h(in(i,j))+(1+abs(la))^p;	se actualizează histograma
end	
end	

2.3 Segmentarea pe histogramă

Dacă histograma are doar două moduri dominante, separarea acestora (şi deci identificarea obiectelor din imagine) se face prin alegerea unui nivel de gri T, numit prag de segmentare. Acest prag de segmentare se alege pe minimul global al histogramei. Din imaginea iniţială f de nivele de gri se construieşte o imagine de etichete (imagine etichetată) g, conform transformării descrise de (2.6) (vezi Figura 2.1 a)).

$$g(m,n) = \begin{cases} E_0, 0 \leq f(m,n) < T \\ E_1, T \leq f(m,n) < L \end{cases} \tag{2.6}$$

Imaginea etichetată va fi descrisă de două etichete: E_0 pentru punctele al căror nivel de gri este mai mic decât pragul T şi E_1 pentru punctele al căror nivel de gri este mai mare decât pragul T. Etichetele E_0 şi E_1 pot fi valori numerice (0 şi 1, sau 0 şi 255) sau pot fi şiruri de simboluri sau alţi identificatori.

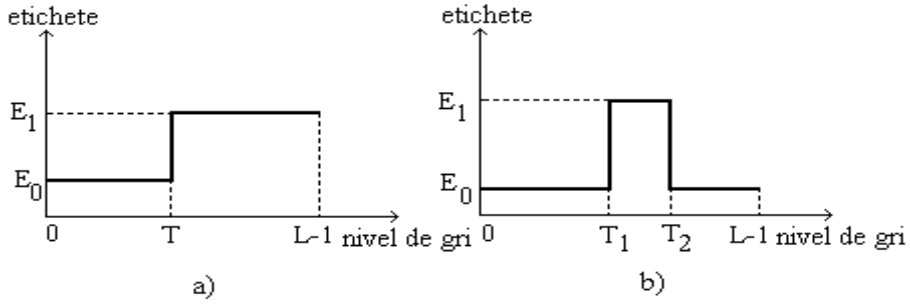

Fig. 2.1: Transformări punctuale de binarizare.

Transformarea (2.6) este o transformare punctuală (noua valoare din punctul (m,n) de-

pinde doar de valoarea anterioară din punctul (m, n)) şi poartă numele de binarizare. Această denumire provine din faptul că rezultatul transformării (imaginea etichetată) este o imagine binară - deci o imagine caracterizată doar de două valori.

Există însă şi o variantă de binarizare cu două praguri (transformare punctuală numită decupare - "slicing"), definită de ecuaţia următoare (2.7) (vezi Figura 2.1 b)):

$$g(m,n) = \begin{cases} E_0, \text{ dacă } f(m,n) < T_1 \text{ sau } f(m,n) > T_2 \\ E_1, \text{ în rest} \end{cases} \tag{2.7}$$

În cazul general al existenţei a mai multe praguri de segmentare T_k, transformarea de segmentare pe histogramă este descrisă de (2.8)

$$g(m,n) = E_k \quad \text{dacă} \quad T_k \leq f(m,n) < T_{k+1} \tag{2.8}$$

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

Pragurile T_k se aleg prin inspecţia histogramei, în minimele locale ale acesteia. Acest tip de segmentare multinivel este mai puţin eficient decât binarizarea, din cauza dificultăţii de stabilire a pragurilor care să izoleze eficient intervalele de interes din histogramă, mai ales atunci când numărul modurilor este mare. Trebuie de asemenea remarcat faptul că este necesară cunoaşterea numărului de tipuri de obiecte din imagine, pentru alegerea corespunzătoare a numărului de praguri de segmentare.

Pentru segmentarea unei imagini este disponibilă funcţia **thresh** (de la termenul "thresholding"); folosirea cea mai simplă este:

≫y=thresh(x, prag);

Comanda anterioară segmentează prin thresholding imaginea x după pragurile indicate de vectorul *prag*; imaginea de etichete este y. Pragurile sunt, evident, valorile nivelelor de gri ce corespund minimelor histogramei.

Observaţie 3 *Etichetele alocate sunt numerice, încep cu valoarea 1 şi sunt incrementate unitar. Vizualizarea unei imagini de etichete se poate face cu:*

≫image(y),colormap(hsv(max(y(:))))

Nucleul funcţiei de segmentare este realizat de:

function out=thresh(in,prag,t)	declaraţia funcţiei (cu parametri nefolosiţi)
out=ones(size(in));	se iniţializează imaginea de etichete
prag=[prag_min prag prag_max];	se completează pragurile cu valorile extreme
np=max(size(prag));	se determină numărul de etichete
for i=1:np-1	pentru fiecare etichetă
p=find((in≥prag(i))&(in<prag(i+1)));	se determină pixelii ce vor primi eticheta
val=i+1;	se incrementează valoarea etichetei
out(p)=val*ones(size(p));	se înscriu etichetele în imaginea de etichete
end	

2.4 Segmentarea pe histograma cumulativă

Histograma cumulativă (2.2) este funcţia de repartiţie a variabilei aleatoare ce reprezintă nivelul de gri al imaginii, şi deci este probabilitatea ca un pixel din imagine să aibă nivelul de gri mai mic ca un prag fixat.

Exemplul clasic de folosire a tehnicii de segmentare cu folosirea histogramei cumulative este acela de binarizare a unei imagini ce conţine obiecte de interes şi fundal. În general se consideră că obiectele de interes sunt caracterizate de un nivel de gri mai mic ca al fundalului, şi că obiectele ocupă un procent P din suprafaţa (numărul de pixeli) imaginii. În aceste condiţii, pragul de segmentare T se va alege astfel încât:

$$H(T) \widetilde{=} P \tag{2.9}$$

2.5 Desfăşurarea lucrării

1. Se vizualizează şi înregistrează (în memorie şi notiţe) întreaga informaţie despre folosirea funcţiilor de citire a unor imagini din fişiere cu formatele IMG şi BMP (**imgread**, **bmp8rd**), calcularea histogramei şi histogramei ponderate a unei imagini (**histo**, **whisto**) şi segmentarea imaginilor cu praguri (**thresh**).
   ≫*help imgread*
   ≫*help bmp8rd*
   ≫*help histo*
   ≫*help whisto*
   ≫*help thresh*

2. Urmăriţi modul în care au fost scrise funcţiile folosite (vizualizaţi codul complet al fiecărei funcţii).
   ≫*type imgread*

≫*type bmp8rd*
≫*type hist*

≫*type whisto*
≫*type thresh*

1. Se vor încărca imaginile de test disponibile: test1_128.img până la test5_128.img şi se vor afişa. Pentru fiecare dintre acestea se vor calcula histograma şi diferite histograme ponderate (pentru puterile -3,-2,-1,0,1,2). Încercaţi să deduceţi pragurile de segmentare.
   ≫*x=imgread('test1_128', 128);*
   ≫*figure(1), image(x), colormap(gray(256))*
   ≫*h=histo(x);*
   ≫*figure(2), plot(h)*
   ≫*wh=whisto(x, putere);*
   ≫*figure(3), plot(wh)*

2. Segmentaţi imaginile de test folosind pragurile deduse anterior. Caracterizaţi rezultatele acestei segmentări (în funcţie de numărul şi poziţia pragurilor).
   ≫*y=thresh(x, [prag1 prag2 prag3 ...]);*
   ≫*figure(4), image(y), colormap(hsv(max(y(:)))), colorbar*

3. Încărcaţi şi imagini "naturale" existente (lena, peppers, camera, cactus) şi repetaţi etapele precedente. Ce se poate spune despre rezultatele segmentării ?

4. Adăugaţi zgomot imaginilor şi reluaţi calculul histogramelor, al pragurilor şi al segmentării. Adăugarea zgomotului cu distribuţie normală, medie nulă şi dispersie *pzg* (având valorile 5, 10, 20, 30, 40, 50, 60) peste o imagine originală x se poate face cu următoarea secvenţă de comenzi Matlab.
   ≫*xzg=x+fix(pzg*randn(size(x)));*
   ≫*p=find(xzg<1);*
   ≫*xzg(p)=ones(size(p));*
   ≫*p=find(xzg>256);*
   ≫*xzg(p)=256*ones(size(p));*
   ≫*figure(5), image(xzg), colormap(gray(256))*

   Acelaşi rezultat poate fi obţinut şi prin secvenţa mai compactă:

   ≫*xzg=x+fix(pzg*randn(size(x)));*
   ≫*xzg=round(max(1, min(256, xzg)));*
   ≫*figure(5), image(xzg), colormap(gray(256))*

5. Urmăriţi dacă o eventuală preprocesare (o filtrare a imaginii zgomotoase înaintea segmentării) poate îmbunătăţi rezultatele segmentării. Reducerea zgomotului poate

fi încercată fie cu un filtru de mediere (de medie aritmetică), fie cu un filtru median. Ambele tipuri de filtrări se pot realiza cu funcţia **lfilter**.
≫help lfilter
≫type lfilter
≫xzg_medie=lfilter(xzg, ones(1,9)/9);
≫xzg_median=lfilter(xzg, [0 0 0 0 1 0 0 0 0]);

6. Urmăriţi rezultatele unei segmentări pe histograma cumulativă pentru una dintre imaginile intrinsec binare aflate la dispoziţie (reprezentând porţiuni din documente scanate) presupunând diferite procente de ocupare a suprafeţei imaginii cu obiectele de interes.

   ≫x=imgread('imagine_test', 128);
   ≫figure, image(x), colormap(gray(256))
   ≫h=histo(x);
   ≫ prag=cumthresh(h,procent,0);
   ≫y=thresh(x, [prag]);
   ≫figure, image(y), colormap(hsv(max(y(:)))), colorbar

2.6 Probleme şi întrebări

1. Urmăriţi cele două moduri de calculare a histogramei unei imagini (funcţia **histo**), determinate de alegerea unor valori diferite ale parametrului *sw*; care este diferenţa dintre acestea şi care este mai rapid ? Măsuraţi timpul de rulare al celor două variante. Timpul de rulare a unei comenzi se poate afla prin comenzile **tic** şi **toc**:
   ≫tic, y=histo(x, prag, 1); toc
   ≫tic, y=histo(x, prag, 0); toc
   Ce concluzii se pot deduce în ceea ce priveşte eficienţa temporală a structurilor de ciclare (**for**) în Matlab ?

2. Urmăriţi cele două variante de etichetare pe care le poate produce funcţia **thresh** (cu etichete ce încep cu 1 şi se incrementează şi cu nivelul de gri mediu al fiecărui interval). Reluaţi segmentarea unor imagini naturale folosind etichetarea cu nivelul de gri mediu; ceea ce rezultă este o aproximare a imaginii originale, dar cu un număr mai mic de nivele de gri (egal cu numărul de clase). Măsuraţi calitatea acestei aproximări (se foloseşte funcţia **graymas**).

3. Scrieţi o funcţie Matlab care să determine în mod automat pragurile de segmentare (minimele histogramei) şi testaţi rezultatele acesteia.

4. Scrieţi o funcţie Matlab care să realizeze calculul histogramei ponderate a unei imagini şi care să fie apelată cu matricea ce corespunde imaginii pentru care se face

calculul şi o matrice care să conţină parametrul de ponderare pentru fiecare pixel. Comparaţi timpul de rulare a acestei funcţii cu timpul de rulare a funcţiei **whisto**.

5. Scrieţi o funcţie Matlab care să evalueze cantitativ calitatea unei binarizări prin evaluarea matricii de confuzie (matricea ce conţine procentul de pixeli de obiect şi fundal etichetaţi corect şi procentele de pixeli clasificaţi eronat). Testaţi funcţia evaluând rezultatele segmentării imaginilor intrinsec binare (folosite la segmentarea pe histograma cumulativă) pentru care sunt disponibile rezultatele corecte ale segmentării.

Capitolul 3

Segmentarea pe histogramă: modelarea histogramei

Lucrarea urmăreşte modul în care o histogramă oarecare poate fi modelată ca o mixtură de distribuţii, în particular Gaussiene (normale).

3.1 Mixturi de distribuţii şi separabilitatea modurilor

O mixtură de distribuţii este o combinaţie liniară (ponderată) convexă a unor funcţii de distribuţie (funcţii de densitate de probabilitate). Cazul cel mai des folosit este acela al unei mixturi de distribuţii Gaussiene.

$$N(\mu_k, \sigma_k)(x) = \frac{1}{\sigma_k\sqrt{2\pi}} e^{-\frac{(x-\mu_k)^2}{2\sigma_k^2}}$$

Dacă histograma h a imaginii este compusă prin superpoziţia aditivă a C moduri gaussiene $N(\mu_k, \sigma_k)$, atunci:

$$h(x) = \sum_{k=1}^{C} w_k N(\mu_k, \sigma_k)(x)$$

$$\sum_{k=1}^{C} w_k = 1$$

Funcţia Matlab **gausmixt** construieşte o histogramă ca mixtură a unui număr fixat de moduri Gaussiene.

function [fct]=gausmixt(params,p,range)	declaraţia funcţiei
mu=params(:,1);	mediile distribuţiilor
sigma=params(:,2);	dispersiile distribuţiilor
for i=1:no	pentru toate punctele de calcul
e=-((x-mu(i))/sigma(i)).^2;	
P(i,:)=(p(i)/(sigma(i)*sqrt(2*pi)))*exp(e);	se calculează fiecare distribuţie
end	
fct=sum(P)/sum(sum(P));	se calculează suma distribuţiilor

Separabilitatea modurilor înseamnă verificarea de către modurile vecine spaţial a unei serii de condiţii, ce garantează că modurile sunt discernabile. Un asemenea set de condiţii a fost introdus de Nakagawa:

$$\begin{cases} \mu_2 - \mu_1 > 4 \\ 0.1 < \frac{\sigma_2}{\sigma_1} < 10 \\ \min\limits_{x \in [\mu_1; \mu_2]} f(x) < 0.8 \cdot \min\left(f(\mu_1), f(\mu_2)\right) \end{cases} \tag{3.1}$$

Funcţia **nakagawa** face aceste verificări pentru modurile unei mixturi oarecare caracterizate de funcţia de densitate de probabilitate h şi matricea de parametri statistici (medii şi dispersii) *param*. Lista *list* conţine numărul de ordine al modurilor vecine ce nu verifică condiţiile din (3.1) şi tipul condiţiei ce nu este verificată (medii prea apropiate, dispersii prea diferite, separaţie neclară)
$\gg$*list=nakagawa(h, param);*

3.2 Metoda Bhattacharyya

Metoda Bhattacharyya se bazează pe descompunerea histogramei în moduri individuale gaussiene, adică se încearcă exprimarea histogramei imaginii ca o sumă ponderată de funcţii de densitate de probabilitate de tip normal (Gaussian). Modelarea modurilor histogramei imaginilor prin distribuţii normale este o presupunere ce se întâlneşte în multe tehnici de prelucrare şi analiză şi pare a fi justificată de considerarea imaginii ca provenind dintr-o imagine ideală, în care fiecare tip de obiect este reprezentat de un unic nivel de gri, peste care s-a suprapus un zgomot alb, aditiv, Gaussian. În acest mod, mediile modurilor din histogramă corespund nivelelor de gri ce caracterizează obiectele scenei, iar varianţele acestor moduri sunt determinate de zgomotul suprapus imaginii (care nu este obligatoriu să afecteze în acelaşi mod toate nivelele de gri).

Pentru segmentarea după metoda Bhattacharyya nu este necesară precizarea unui număr de clase (praguri de segmentare), acesta urmând a fi determinat în mod automat. Ideea de plecare a metodei este de a determina parametrii caracteristici ai unei distribuţii normale. Pentru o distribuţie normală derivata logaritmului este:

$$\frac{\delta \ln N(\mu_k, \sigma_k)(x)}{\delta x} = -\frac{x}{\sigma_k^2} + \frac{\mu_k}{\sigma_k^2} = m_k x + n_k \tag{3.2}$$

Se observă prin examinarea expresiei (3.2) că derivata logaritmului distribuţiei normale este o dreaptă de pantă negativă, din ai cărei parametri se pot deduce media şi varianţa distribuţiei. Parametrii statistici ai distributiei sunt daţi de ecuaţiile (3.3).

$$\sigma_k = \sqrt{\frac{1}{|m_k|}}, \text{şi } \mu_k = \frac{n_k}{|m_k|} \tag{3.3}$$

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

$$z(a) = \ln \frac{h(a)}{h(a-1)}, a = \overline{1, L-1} \tag{3.4}$$

Pentru funcţia astfel construită, se determină intervalele pe care acesta este descrescătoare (vezi Figurile 3.1 şi 3.2); limitele superioare ale acestor intervale sunt pragurile T_k de segmentare pe histogramă. Suplimentar, pe fiecare dintre aceste intervale se poate face o aproximare liniară a punctelor şi pe baza parametrilor deduşi pentru dreapta de aproximare se pot calcula, conform (3.3) parametrii statistici locali.

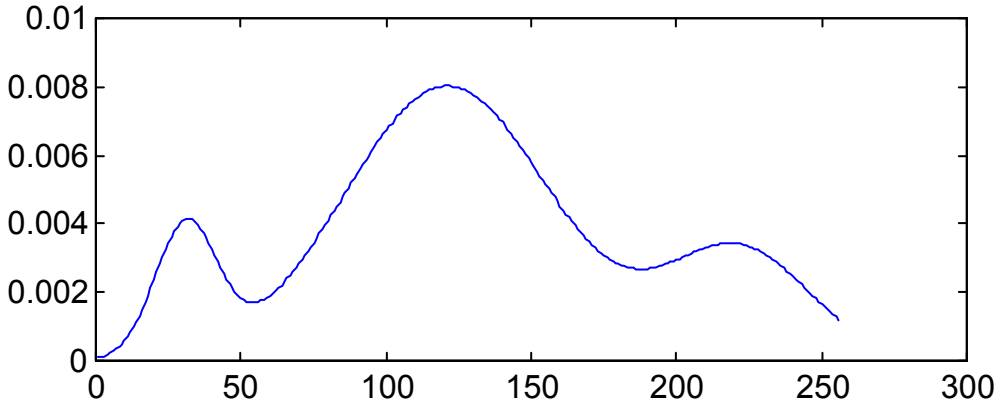

Fig. 3.1: Histogramă cu trei moduri normale

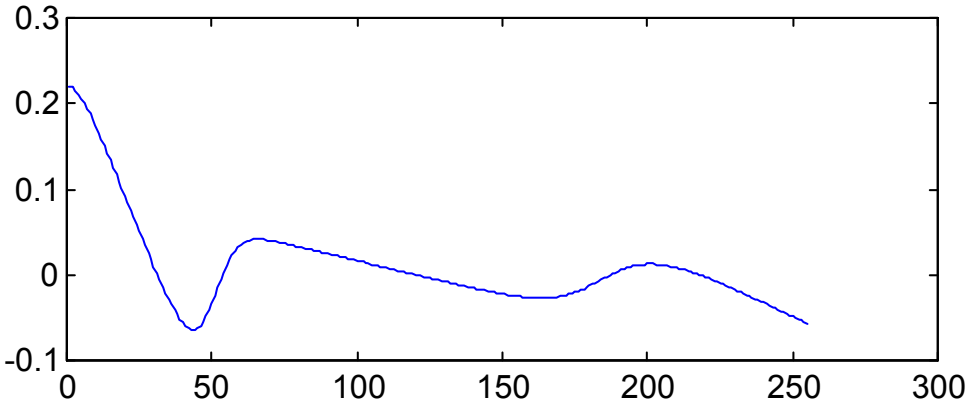

Fig. 3.2: Aplicarea metodei Bhattacharyya pentru histograma trimodală prezentată anterior; se pot observa intervalele pe care funcţia este liniară şi descrescătoare, ce corespund modurilor.

3.3 Segmentarea cu prag optim

Metoda de segmentare cu prag optim face apel la teoria deciziilor (criteriul de decizie Bayes) pentru stabilirea valorii pragurilor de segmentare ce optimizează un anumit criteriu de eroare. Informaţiile apriori necesare pentru aplicarea unei asemenea tehnici sunt numărul de tipuri de obiecte din imagine, C, procentele de ocupare a imaginii de către fiecare tip de obiecte, P_i şi distribuţia nivelelor de gri ce caracterizează fiecare tip de obiect, $p_i(x)$. Atunci histograma imaginii va fi determinată de mixtura distribuţiilor tipurilor de obiecte:

$$h(x) = \sum_{i=1}^{C} P_i p_i(x) \tag{3.5}$$

$$\sum_{i=1}^{C} P_i = 1$$

Criteriul ce se urmăreşte optimizat este eroarea de segmentare (clasificare) a punctelor din imagine, adică este dat de numărul de pixeli ce aparţin primului tip de obiect, dar au nivelul de gri mai mare ca pragul T (fiind deci alocaţi greşit celui de-al doilea tip de obiect) şi numărul de pixeli ce aparţin celui de-al doilea tip de obiect, dar au nivelul de gri mai mic decât pragul de segmentare T (fiind deci alocaţi greşit primului tip de obiect). Aşadar, eroarea de segmentare va fi dată de (3.6):

$$E(T) = P_1 \int_{T}^{+\infty} p_1(x)dx + P_2 \int_{-\infty}^{T} p_2(x)dx \tag{3.6}$$

Pentru ipoteza uzuală de modelare a histogramei imaginii ca o mixtură de moduri gaussiene se obţine:

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

Una dintre simplificările uzuale este presupunerea că $\sigma_1 = \sigma_2 = \sigma$; acestă presupunerea implică modelarea imaginii în nivele de gri ca o imagine cu doar două nivele de gri μ_1 şi μ_2, afectată de un zgomot Gaussian aditiv, având varianţa σ^2. În aceste condiţii, ecuaţia de gradul 2 devine o ecuaţie liniară, a carei soluţie este:

$$T = \frac{\mu_1 + \mu_2}{2} - \frac{\sigma^2}{\mu_1 - \mu_2}\ln\frac{P_1}{P_2} \tag{3.7}$$

3.4 Desfăşurarea lucrării

1. Urmăriţi modul complet de utilizare a fucţiei **gausmixt** şi modul în care a fost scrisă.
   ≫help gausmixt
   ≫type gausmixt

2. Construiţi mai multe mixturi de distribuţii gaussiene, cu diferite medii, varianţe şi factori proporţionali şi afişaţi-le. Generaţi mixturi cu 2, 3, 4, 5 şi 10 moduri, ale căror medii sunt relativ apropiate sau distanţate.
   ≫h=gausmixt([75 25;150 50]);
   ≫plot(h)

3. Verificaţi pentru fiecare mixtură generată anterior dacă modurile ce o formează sunt bine separate (folosiţi funcţia **nakagawa**):
   ≫help nakagawa
   ≫type nakagawa

4. Pentru fiecare dintre mixturile generate anterior determinaţi parametrii statistici individuali ai modurilor prin metoda Bhattacharyya, pentru diferite dimensiuni *dim* ale intervalului pe care se impune ca funcţia z (3.4) să fie descrescătoare (1, 2, 3, etc.):

≫help bhat
≫[prag, z, interval, param]=bhat(h, dim);

5. Comparaţi parametrii determinaţi anterior cu parametrii reali ai modurilor mixturii şi trageţi concluzii privind eficienţa şi precizia metodei Bhattacharyya. Intervalul "interval" identifică corect intervalele pe care le ocupă modurile reale ?

6. Construiţi histograme bimodale cu parametri diferiţi pentru cele două moduri (folosind funcţia **gausmixt**). Calculaţi pragul de binarizare asociat respectivelor histograme prin diferitele metode, construiţi un tabel care să conţină aceste valori şi valoarea intuitivă. Reprezentaţi grafic histograma şi vizualizaţi pragurile. Comentaţi rezultatele.

3.5 Probleme şi întrebări

1. Se poate aproxima un mod cu distribuţie uniformă plecând de la un mod Gaussian de dispersie foarte mare ?

2. Scrieţi o funcţie Matlab care să construiască o mixtură de moduri Gaussiene, uniforme, exponenţiale şi Rayleigh.

3. Generaţi o mixtură de moduri exponenţiale şi o mixtură de moduri uniforme. Aplicaţi metoda Bhattacharyya pentru respectivele distribuţii şi comentaţi rezultatele.

4. Segmentaţi imagini naturale folosind etichetarea cu nivelul de gri mediu (funcţia **thresh** cu parametru de tip diferit de 1) şi pragurile determinate de metoda Bhattacharyya aplicată histogramei imaginii; ceea ce rezultă este o aproximare a imaginii originale, dar cu un număr mai mic de nivele de gri (egal cu numărul de clase). Măsuraţi calitatea acestei aproximări (se foloseşte funcţia **graymas**) şi compresia realizată (reducerea numărului de nivele de gri diferite din imagine).

5. Să se scrie o funcţie Matlab care să implementeze binarizarea cu prag optim, în ipoteza modurilor Gaussiene.

Capitolul 4

Creşterea regiunilor

Pentru aplicarea cu succes a tehnicilor de segmentare pe histogramă prezentate anterior trebuiesc îndeplinite neapărat câteva condiţii (deja enunţate). Aplicarea tehnicilor de segmentare pe histogramă este condiţionată în primul rând de reprezentarea diferitelor clase de obiecte din imagine pe intervale de nivele de gri diferite care nu se suprapun (sau se suprapun parţial pe porţiuni foarte mici); apoi este necesară cunoaşterea numărului de tipuri de obiecte diferite. În fine, se presupune că valorile prag corespunzătoare se pot determina cu o precizie corespunzătoare.

Chiar în cazurile în care toate aceste condiţii enunţate sunt îndeplinite, nu se poate garanta condiţia de conexitate a regiunilor obţinute în urma segmentării. Acest lucru este evident, atât timp cât două obiecte de acelaşi tip, neconexe, primesc prin segmentarea pe histogramă o aceeaşi etichetă, şi formează în imaginea de etichete o regiune neconexă. Creşterea regiunilor respectă toate condiţiile impuse de definiţia matematică a segmentării.

Principiul pe care se bazează creşterea regiunilor este simplu: se aleg în imagine puncte reprezentative pentru fiecare obiect individual şi categorie de obiecte, pe baza cărora are loc un proces de aglomerare a pixelilor vecini acestora, ce au aceleaşi proprietăţi (în particular acelaşi nivel de gri). În urma acestui proces de aglomerare (adăugare de puncte) se obţin zone (regiuni) de pixeli cu aceleaşi caracteristici, deci obiecte individuale. Procesul se opreşte în momentul în care fiecare punct al imaginii a fost alocat unei regiuni. Evident, metoda astfel descrisă pe scurt, are două etape esenţiale: alegerea punctelor de start (puncte iniţiale), numite germeni sau seminţe, şi creşterea propriu-zisă a regiunilor.

Numărul final de regiuni rezultate este egal cu numărul de germeni aleşi iniţial pentru creştere. În principiu, este de dorit ca fiecare obiect individual aflat în imagine să fie marcat de câte un germene. Dacă în interiorul aceluiaşi obiect se găsesc mai mulţi germeni, pentru fiecare dintre ei va fi crescută o regiune; aceasta face ca obiectul iniţial

să fie împărţit artificial prin segmentare în mai multe regiuni. Parţial, acest neajuns se poate corecta printr-o etapă ce urmează creşterii regiunilor, şi anume fuziunea regiunilor adiacente ce au proprietăţi asemănătoare. Dacă în interiorul unui obiect nu este ales nici un germene, obiectul respectiv va fi înglobat de regiunile ce cresc pornind de la germeni din vecinătatea sa spaţială; astfel, respectivul obiect nu apare ca o regiune distinctă şi este pierdut, rezultând o eroare gravă de segmentare.

Pentru a preveni efectul unor neuniformităţi de iluminare pe suprafaţa imaginii, aceasta este împărţită în ferestre nesuprapuse; în fiecare astfel de fereastră se alege un număr de germeni, al căror plasament spaţial este aleator (germenii se distribuie uniform pe suprafaţa imaginii). Germenii se aleg astfel încât nivelul lor de gri să fie reprezentativ pentru obiectele prezente local (deci nivelul de gri al germenilor trebuie să corespundă unor maxime ale histogramei locale). În plus, trebuie verificat ca plasamentul spaţial al germenilor să se facă în interiorul regiunilor şi nu pe frontiera acestora. Verificarea se poate face simplu pe baza calculului unui operator derivativ local, ca de exemplu laplacianul; dacă valoarea acestuia nu depăşeşte un anumit procent prestabilit (10% - 20%) din diferenţa maximă de nivele de gri a ferestrei, punctul ales este considerat ca plasat corect.

Valorile procentuale ale pragurilor de comparaţie, precum şi numărul de germeni distincţi ce rămân după procesul de reducere, nu trebuie considerate ca fixe; nu există valori standardizate şi alegerea acestora se face pe baza condiţiilor particulare (legate de conţinutul imaginii) şi a experienţei utilizatorului.

Pornind de la germenii aleşi, regiunile sunt obţinute printr-un proces de creştere aproape simultană, început de la aceştia, până când toţi pixelii imaginii sunt repartizaţi unei regiuni. Cvasi-simultaneitatea creşterii poate fi realizată cu un algoritm serial, prin alocarea pixelilor ce sunt adiacenţi (vecini) zonelor deja segmentate. Acestă alocare trebuie să ţină seama de criteriul ca regiunile crescute să fie uniforme: nivelul de gri al pixelului ce se adaugă nu trebuie să difere cu mai mult de un prag prestabilit faţă de nivelul de gri al germenului regiunii la care se alocă. În acelaşi timp, la o singură trecere, numărul de puncte ce se adaugă unei regiuni nu poate depăşi un număr prestabilit (condiţia încearcă să asigure creşterea relativ uniformă şi izotropă a tuturor regiunilor).

Dacă adăugarea de noi pixeli se blochează (criteriul de uniformitate nu mai este respectat), diferenţa maxim admisă pentru nivelul de gri poate fi crescută în etape, până la epuizarea pixelilor imaginii.

Avantajele pe care le are o asemenea tehnică de creştere a regiunilor sunt acelea că nu mai este necesară nici o informaţie privind conţinutul imaginii, regiunile crescute sunt conexe şi nu există puncte neetichetate (nealocate vreunei regiuni), iar poziţia frontierelor dintre diferitele regiuni corespunde poziţiei frontierelor percepute subiectiv în imagine.

4.1 Desfăşurarea lucrării

1. Urmăriţi modul de folosire şi modul de scriere a funcţiei de creştere a regiunilor, **grow_reg** (folosiţi comenzile Matlab **help** şi **type**).

2. Pentru imaginile disponibile urmăriţi rezultatele procesului de creştere a regiunilor folosind diferiţi parametri de reglaj (toleranţa la modificarea nivelelor de gri, dimensiune maximă a regiunii) şi diferiţi germeni. De exemplu:

   ≫x=imgread('lena128', 128);

   ≫[pozitii, valori]=grow_reg(x, 100, 100, 20, 100);

   Pentru a vizualiza regiunea astfel crescută, executaţi
   ≫h=histo(x);

   ≫rez=find(h==0);

   ≫map=gray(256); map(rez(1), 1)=1; map(rez(1), 2)=0; map(rez(1), 3)=0;

   *≫y=x; [t,tt]=size(pozitii); y(pozitii)=rez(1)*ones(t);*

   ≫figure, image(y), colormap(map), colorbar

3. Modificaţi comenzile anterioare astfel încât regiunea crescută să fie reprezentată în nivelul său mediu de gri.

4.2 Probleme şi întrebări

1. Găsiţi o legătură între pragul de toleranţă la diferenţa de nivele de gri pentru creşterea unei regiuni şi eroarea de aproximare a zonei respective din imagine prin nivelul de gri mediu al regiunii crescute.

2. Determinaţi o modalitate automată de determinare a pragului de toleranţă la diferenţa de nivele de gri.

3. Scrieţi o funcţie Matlab care să realizeze iniţializarea germenilor pentru segmentarea unei imagini prin creştere regiunilor.

Capitolul 5

Etichetarea imaginilor binare

Scopul acestei lucrări este studiul unor metode de etichetare a imaginilor binare - alocarea la toate punctele aceluiaşi obiect din imagine o aceeaşi etichetă şi eventual extragerea obiectelor individuale din imagine.

5.1 Etichetarea secvenţială iterativă

Etichetarea secvenţială iterativă se bazează pe perechi de baleiaje ale imaginii (baleiaj direct, de la stânga la dreapta şi de sus în jos şi baleiaj invers, de la dreapta la stânga şi de jos în sus).

Un pseudocod ce implementează algoritmul propus este prezentat în continuare:

repetă
baleiaj direct:
dacă punctul curent este punct de obiect, **atunci**
dacă punctul curent are predecesori etichetaţi, **atunci**
punctul curent ia eticheta minimă a predecesorilor
altfel
dacă acesta este primul baleiaj direct, **atunci**
punctul curent ia o etichetă nouă
baleiaj invers:
dacă punctul curent este punct de obiect, **atunci**
dacă punctul curent are predecesori etichetaţi, **atunci**
punctul curent ia eticheta minimă a predecesorilor
pâna când etichetele nu se mai modifică

Observaţie 4 *Atenţie: toate funcţiile de citire a imaginilor indexate din fişiere (**imgread**, **bmp8rd**, etc.) generează imagini cu indici ce încep cu valoarea 1 şi nu 0. În aceste condiţii, punctele de fundal ale imaginilor binare nu mai sunt descrise de valoarea 0 ci de valoarea 1. De asemenea, este posibil ca punctele de obiect să fie descrise la rândul lor fie de valoarea 2 (ce provine din 1 incrementat), fie de valoarea 256 (ce provine din 255 incrementat).*

Etichetele vor fi numerice, începând cu valoarea 1.

5.2 Etichetarea cu corespondenţă între etichete

Dezavantajul principal al algoritmului anterior este că nu se reţin echivalenţele determinate pentru etichete, ci modificările se efectuează punctual. O îmbunătăţire a performanţelor acestui algoritm se realizează prin gestionarea unui tabel de echivalenţă (corespondenţă) a etichetelor.

Fie T tabelul de corespondenţă între etichete; o intrare a tabelului $T(j) = v$ înseamnă că eticheta j este echivalentă etichetei v; la pornirea algoritmului, fiecare intrare a tabelului este iniţializată cu valoarea indicelui acesteia ($T(j) = j$). Actualizarea tabelului se face în două situaţii:

- dacă punctul curent nu are nici un predecesor punct de obiect, atunci acesta trebuie să primească o nouă etichetă, care se adaugă la "coada" tabelului de echivalenţă ca $T(j_{\max}) = j_{\max}$
- dacă punctul curent are predecesori etichetaţi, atunci acesta va primi eticheta minimă a acestora şi în tabelul de corespondenţă trebuie înscris faptul că la toate etichetele punctelor ce au intervenit în calcul (predecesorii punctului curent) va corespunde o aceeaşi etichetă, şi anume eticheta de valoare minimă a acestora. Dacă punctul curent are predecesorii P_1, P_2, P_3, P_4 (cel mult patru predecesori, în cazul folosirii conexităţii V_8), atunci fie $m = min(P_i)$. Varianta pseudocod a prelucrării este prezentată în continuare.

pentru toţi predecesorii P
  cât timp $T(P_i) \neq m$ **execută**
    $temp = T(P_i)$
    $T(P_i) = m$
    $P_i = temp$

După primul baleiaj al imaginii, tabelul de corespondenţă este actualizat astfel încât la fiecare indice să corespundă eticheta finală a obiectului; etichetele atribuite obiectelor nu vor fi neapărat în ordine. Pseudocodul corespunzător acestei prelucrări este:

pentru j **de la** 1 **la** $j_{\max}$
    $k = j$
    cât timp $T(k) \neq k$ **execută**
        $k = T(k)$
    $T(j) = k$

Ultima parte a algoritmului constă în construirea imaginii finale de etichete prin înlocuirea în imaginea intermediară de etichete a fiecărei valori j cu $T(j)$.

5.3 Etichetarea prin propagarea etichetelor

O metodă alternativă de etichetare este extragerea fiecărui obiect individual în parte. Pentru aceasta trebuie întâi detectat un punct al obiectului (de exemplu primul punct de obiect întâlnit la baleiajul direct al imaginii), începând de la care se "creşte" obiectul, lipind punctele de obiect care sunt vecine, până în momentul în care nu mai există puncte de obiect vecine. Toate punctele astfel determinate formează un unic obiect şi capătă deci o unică etichetă; procedeul se reia pentru celelalte obiecte din imagine. Codul de bază care realizează acestă operaţie este dat în continuare.

function out=etichete(in,val_obj)	definiţia funcţiei
out=ones(size(in));	iniţializare imagine de etichete
[h,w]=size(in);	dimensiuni imagine
val=2;	prima etichetă obiect
p=find(in==val_obj);	primul punct al primului obiect
while (p~=[])	pentru toate obiectele imaginii
seed_i=rem(p(1),h);	coordonatele primului
seed_j=fix(p(1)/h)+1;	punct al obiectului curent
sir=grow_reg(in,seed_i,seed_j,0);	se creşte regiunea
p=(sir(:,2)-1)*h+sir(:,1);	coordonatele punctelor regiunii
out(p)=val*ones(size(p));	capătă eticheta curentă
in(p)=zeros(size(p));	şi nu se mai ia în calcul
val=val+1;	eticheta următoare
p=find(in==val_obj);	obiectul următor
end	gata

De exemplu, pentru a eticheta o imagine binară *in* ce are fundalul de valoare 1 şi obiectele de valoare 256 se foloseşte:

≫out=etichete(in, 256);

Afişarea confortabilă (afişarea fiecărui obiect cu o altă culoare) a hărţii de etichete se poate face cu una dintre secvenţele:

≫image(out),colormap(rand(max(out(:)),3)),colorbar

≫image(out),colormap(hsv(max(out(:)))),colorbar

5.4 Desfăşurarea lucrării

1. Se încarcă imaginile binare disponibile şi se etichetează, folosind funcţia **etichete**. Se măsoară de fiecare dată timpul necesar etichetării (folosind comenzile **tic** şi **toc** ale Matlab) şi numărul de parcurgeri ale imaginii de către funcţie. Aceste rezultate se trec într-un tabel.

2. Se repetă calculele folosind funcţia Matlab care realizează etichetarea imaginilor binare prin baleiaj cu tabel de corespondenţă între etichete (funcţia **etic1**).

5.5 Întrebări şi probleme

1. Din ce cauză, la vizualizarea imaginii etichetate, este posibil să apară uneori două culori diferite pentru un acelaşi obiect, sau pot apărea obiecte diferite de aceeaşi culoare ?

2. Se scrie o funcţie Matlab care să realizeze etichetarea imaginilor binare prin baleiaj secvenţial iterativ. Să se testeze funcţia pe imaginile binare disponibile. Să se măsoare timpul de rulare şi numărul de baleiaje ale imaginii.

3. Să se compare cele trei metode de etichetare pe baza timpului de rulare şi a numărului de parcurgeri a imaginilor.

Capitolul 6

Extragerea contururilor

Într-o imagine, variaţiile de valoare ale pixelilor reprezintă schimbări ale proprietăţilor fizice sau geometrice ale scenei sau ale obiectului observat. Aceste schimbări pot corespunde fizic la variaţiile iluminării, schimbările de orientare sau de distanţă faţă de observator, schimbări de reflectanţă ale suprafeţelor, variaţii de absorbţie a radiaţiei. Într-un număr mare de cazuri, aceste variaţii de intensitate sunt informaţii importante pentru operaţiile ce urmează segmentării, informaţii ce corespund frontierelor regiunilor determinate de obiectele scenei.

6.1 Metoda de gradient

Principiul acestei metode constă în definirea punctelor de contur ca fiind acei pixeli ai imaginii în care apar schimbări importante (abrupte) ale nivelului de gri. Deci, măsurarea acestei variaţii se va face prin operatori derivativi de tip gradient. Pentru o imagine cu suport spaţial continuu, pe direcţia unei muchii, derivata va fi maximă. Derivata imaginii pe direcţia r, ce face unghiul θ cu orizontala, este dată de combinaţia liniară a derivatelor parţiale pe direcţiile orizontală şi verticală (6.1):

$$\frac{\partial f}{\partial r} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial r} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial r} = \frac{\partial f}{\partial x}\cos\theta + \frac{\partial f}{\partial y}\sin\theta$$

$$\frac{\partial f}{\partial r} = f_x \cos\theta + f_y \sin\theta \tag{6.1}$$

Valoarea maximă a acestei derivate, calculate după unghiul θ este dată de ecuaţia

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

ce are soluţia evidentă:

$$\theta_0 = \arctan\left(\frac{f_y}{f_x}\right) \tag{6.2}$$

Pe această direcţie, modulul gradientului este:

$$\left(\frac{\partial f}{\partial r}\right)_{\max} = \sqrt{f_x^2 + f_y^2} \tag{6.3}$$

Din punct de vedere practic, implementarea acestei metode impune calcularea, pentru fiecare punct al imaginii, a derivatelor parţiale f_x şi f_y, calcularea modulului gradientului maxim (6.3) şi a direcţiei acestuia (6.2). Valoarea gradientului maxim din fiecare punct al imaginii este apoi comparată cu un prag fixat: dacă pragul este depăşit (deci gradientul maxim în pixelul respectiv este suficient de important) atunci pixelul testat este pixel de contur.

Realizarea derivatelor parţiale după direcţiile orizontală şi verticală implică translaţia în discret a lui f_x şi f_y:

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

Aceste derivate parţiale discrete pot avea mai multe implementări:

$$f_x = f(m,n) - f(m+1,n),\ f_y = f(m,n) - f(m,n+1) \tag{6.4}$$

$$f_x = f(m-1,n) - f(m,n),\ f_y = f(m,n-1) - f(m,n) \tag{6.5}$$

$$f_x = f(m-1,n) - f(m+1,n),\ f_y = f(m,n-1) - f(m,n+1) \tag{6.6}$$

Toate expresiile date de (6.4), (6.5), (6.6) sunt combinaţii liniare ale valorilor unor pixeli din imagine, situaţi în vecinătatea pixelului curent din poziţia (m,n). Deci toate aceste operaţii se pot realiza prin filtrări liniare cu măşti potrivite: (6.7) pentru (6.4), (6.8) pentru (6.5), (6.9) pentru (6.6).

$$W_x = \begin{pmatrix} 1 & -1 \end{pmatrix},\ W_y = \begin{pmatrix} 1 \\ -1 \end{pmatrix} \tag{6.7}$$

$$W_x = \begin{pmatrix} 1 & -1 \end{pmatrix},\ W_y = \begin{pmatrix} 1 \\ -1 \end{pmatrix} \tag{6.8}$$

$$W_x = \begin{pmatrix} 1 & 0 & -1 \end{pmatrix},\ W_y = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix} \tag{6.9}$$

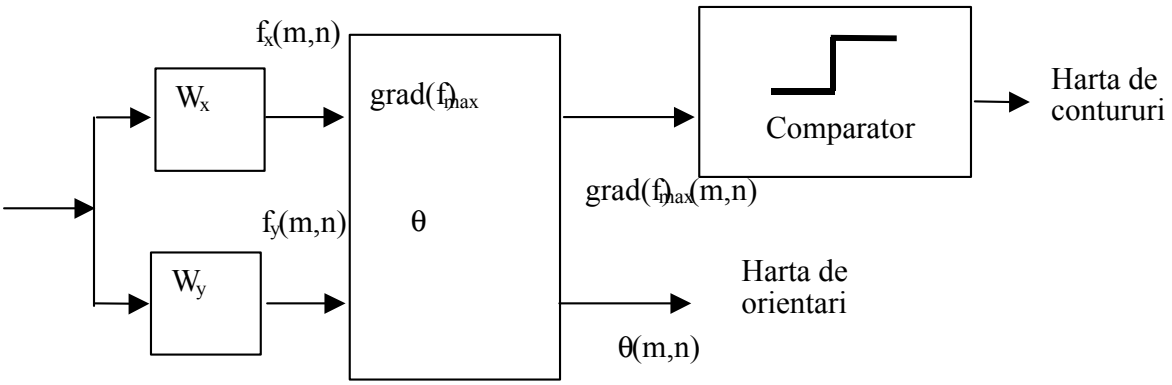

Fig. 6.1: Schema bloc a extractorului de contururi bazat pe metoda de gradient.

Schema bloc a extragerii de contururi este reprezentată în figura 6.1.

Harta de orientări este o imagine care conţine, pentru fiecare pixel, orientarea gradientului de modul maxim în punctul respectiv, şi este în general folosită la prelucrarea suplimentară a contururilor (conectare de contururi, extragere direcţională de contururi). Harta de contururi este o imagine binară în care punctele marcate (puncte-obiect) corespund poziţiei punctelor de contur (puncte cu gradient de modul mare). O simplificare uzuală practicată este înlocuirea normei L2 din calculul modulului maxim al gradientului (6.3) cu norma L1, ceea ce conduce la aproximarea:

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

Folosirea măştilor de derivare pe verticală şi orizontală prezentate are însă serioase neajunsuri: dimensiunea lor mică face ca rezultatele să fie extrem de sensibile la zgomot. În aceste condiţii a apărut naturală ideea de a combina filtrarea de derivare cu o filtrare de netezire, care să mai reducă efectele zgomotului. Considerând zgomotul de tip gaussian, aditiv, filtrarea de netezire are ca efect secundar micşorarea contrastului frontierelor obiectelor din imagine (efectul de înceţoşare, sau *blur*). Pentru ca în aceste condiţii detecţia contururilor să nu fie afectată, trebuie ca operaţia de mediere prin care se realizează netezirea să se facă pe o direcţie perpendiculară direcţiei contururilor căutate. Atunci derivarea pe verticală se combină cu o operaţie de netezire cu mască orizontală $\begin{pmatrix} 1/3 & \boxed{1/3} & 1/3 \end{pmatrix}$ şi derivarea pe orizontală se combină cu o operaţie de netezire cu mască verticală $\begin{pmatrix} 1/3 \\ \boxed{1/3} \\ 1/3 \end{pmatrix}$. Dacă folosim pentru derivare masca W_y din (6.7), masca de filtrare rezultantă va fi $\begin{pmatrix} 1/3 & \boxed{1/3} & 1/3 \\ -1/3 & -1/3 & -1/3 \end{pmatrix}$. În cazul general se pot folosi însă

pentru netezire medieri ponderate (şi nu neapărat medieri aritmetice), care să acorde o mai mare importanţă pixelului curent prelucrat, ca de exemplu $\frac{1}{c+2}\begin{pmatrix} 1 & c & 1 \end{pmatrix}$ şi se preferă folosirea operatorilor de derivare simetrici, de tipul (6.9). Ceea ce rezultă pentru operatorii de derivare orizontală şi verticală sunt măştile:

$$W_x = \begin{pmatrix} 1 & 0 & -1 \\ c & 0 & -c \\ 1 & 0 & -1 \end{pmatrix}, W_y = \begin{pmatrix} 1 & c & 1 \\ 0 & 0 & 0 \\ -1 & -c & -1 \end{pmatrix} \tag{6.10}$$

Prin particularizarea valorilor constantei de ponderare c se pot obţine diferite tipuri de operatori de extragere de contur clasici: Prewitt ($c = 1$), Izotrop ($c = \sqrt{2}$), Sobel ($c = 2$). Se remarcă faptul că constanta de ponderare globală a măştii de filtare este neesenţială, întrucât condiţia de normare ce trebuie îndeplinită este cea pentru filtre de contrastare (derivare): suma coeficienţilor măştii să fie nulă.

6.2 Operatori compas

Informaţia de orientare este în general folosită în etape următoare ale prelucrării; unghiurile determinate după (6.2) oferă un unghi "exact" al direcţiei conturului în punctul curent, calculat cu un efort semnificativ de calcul (împărţire şi calcul de arctangentă). În practică, această informaţie este prea exactă: pe grila pătrată de eşantionare nu se pot reprezenta cu uşurinţă drepte continue după orice direcţie; câteva direcţii sunt favorizate şi uşor de utilizat (vertical, orizontal, cele două diagonale). În acest caz se poate măsura în fiecare punct modulul gradientului după aceste câteva direcţii importante, şi apoi se poate alege direcţia după care acest modul este maxim. Acesta este principul operatorilor compas.

Un operator compas este definit de un număr de măşti de derivare (corespunzătoare în continuare unor filtrări liniare) pe direcţiile principale (vertical, orizontal, cele două diagonale), în cele două sensuri. Compasul clasic are $D = 8$ măşti de filtrare (identice două câte două, mai puţin semnul), fiecare dintre ele realizând o derivare după o direcţie multiplu de 45°. Schema bloc a unui operator compas este prezentată în figura 6.2; se remarcă faptul că, odată determinată valoarea maximă a modulului gradientului în pixelul curent (m, n), obţinerea hărţii de contururi se face ca şi la un operator de gradient clasic.

Un exemplu de măşti de derivare direcţională sunt măştile următoare (indexate după direcţia geografică pe care calculează derivata): $W_N = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$, $W_{NV} =$

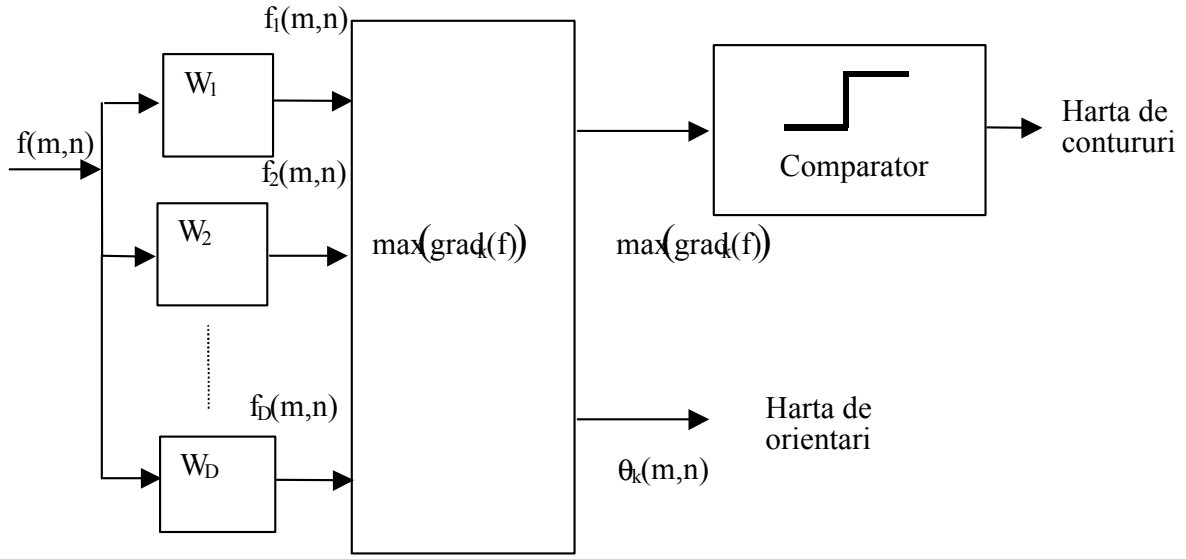

Fig. 6.2: Schema bloc a unui operator compas de extragere a contururilor.

$\begin{pmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}, W_V = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}, W_{SV} = \begin{pmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{pmatrix}, W_S = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix},$
$W_{SE} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -1 \end{pmatrix}, W_E = \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix}, W_{NE} = \begin{pmatrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{pmatrix}$. După cum se remarcă, familia de măşti se poate genera pornind de la una dintre măştile Prewitt, prin translaţii circulare cu o poziţie a frontierei măştii în jurul centrului ei; în mod analog se pot obţine operatori compas bazaţi pe masca Sobel sau pe gradientul izotrop sau pe masca Kirsch $\begin{pmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{pmatrix}$. Precizia unghiulară a operatorilor compas este deci determinată de numărul de orientări diferite pe care se calculează derivatele, şi deci de numărul de translaţii ale frontierei măştii; pentru o mască pătrată de bază de dimensiune N, precizia unghiulară a operatorului compas este de $90°/(N-1)$.

6.3 Operatori laplacieni

Unul dintre principalele dezavantaje ale metodelor de gradient este precizia slabă de localizare a conturului (a centrului tranziţiei) în condiţiile unei pante puţin abrupte a acestuia (tranziţii slabe, graduale). Derivata a doua poate fi însă folosită pentru a determina capetele tranziţiei (cele două extreme), sau pentru a marca centrul tranziţiei

(trecerea sa prin zero). Operatorul bazat pe trecerea prin zero a derivatei secunde este operatorul "zero-crossing". În cazul imaginilor (semnale cu suport bidimensional) trebuie luată în considerare derivata secundă după ambele direcţii, combinate în laplacian:

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

În cazul discret, măşti ce implementează laplacianul sunt măştile $WL_1 - WL_3$, $WL_1 = \begin{pmatrix} 0 & -1/4 & 0 \\ -1/4 & \boxed{1} & -1/4 \\ 0 & -1/4 & 0 \end{pmatrix}, WL_2 = \begin{pmatrix} 1/4 & -1/2 & 1/4 \\ -1/2 & \boxed{1} & -1/2 \\ 1/4 & -1/2 & 1/4 \end{pmatrix}, WL_3 = \begin{pmatrix} -1/8 & -1/8 & -1/8 \\ -1/8 & \boxed{1} & -1/8 \\ -1/8 & -1/8 & -1/8 \end{pmatrix}$.
Operatorii laplacieni au o sensibilitate crescută în prezenţa zgomotelor (mai mare decât a operatorilor de gradient) şi nu mai conţin informaţie relativă la direcţia tranziţiei.

6.4 Desfăşurarea lucrării

1. Iniţializaţi diferitele măşti ce vor fi folosite pentru calculul gradientului:

   ≫prewitt=[1 1 1;0 0 0;-1 -1 -1];

   ≫izotrop=[1 sqrt(2) 1;0 0 0;-1 -sqrt(2) -1];

   ≫sobel=[1 2 1;0 0 0;-1 -2 -1];

   ≫kirsch=[5 5 5;-3 0 -3;-3 -3 -3];

   Se încarcă una dintre imaginile disponibile:

   ≫x=imgread('nume_imagine', 128);

2. Se construieşte harta de intensităţi de tranziţie prin metoda de gradient cu diferitele măşti (*prewitt*, *izotrop*, *sobel*, *kirsch*) şi se extrage conturul folosind diferite valori pentru pragul de binarizare (decizie punct de contur sau nu, între 1 şi 256) şi se vizualizează rezultatele.

   ≫harta=gradient(x, masca);

   ≫contur=255(thresh(e,prag)-1)+1;*

   ≫figure; subplot(2,2,1), image(x), colormap(gray(256)

   ≫subplot(2,2,3), image(harta), subplot(2,2,4), image(contur)

3. Repetaţi calculele folosind un operator compas (se folosesc aceleaşi măşti, aceleaşi imagini, acelaşi mod de binarizare şi vizualizare a rezultatelor ca anterior):

   ≫harta=compas(x, masca);

4. Iniţializaţi diferitele măşti ce vor fi folosite pentru calculul operatorului zero-crossing (măşti de laplacian):

   ≫lapl1=[0 -0.25 0;-0.25 1 -0.25;0 -0.25 0];

   ≫lapl2=[0.25 -0.5 0.25;-0.5 1 -0.5;0.25 -0.5 0.25];

   ≫lapl3=[-1 -1 -1;-1 8 -1;-1 -1 -1]/8;

   Folosiţi aceste măşti pentru a calcula harta de intensităţi de tranziţie (cu funcţi a **gradient**) şi apoi extrageţi conturul prin binarizare.

5. Construiţi o hartă de intensităţi de tranziţie printr-un operator de tip gradient morfologic (dilatare - erodare, cu element structurant V_8) şi extrageţi apoi conturul prin binarizare:

   ≫harta=lfilter(x,[-1/2 0 0 0 0 0 0 0 1/2]);

6. Aplicaţi zgomot Gaussian, uniform şi impulsiv imaginii de test şi repetaţi secvenţa de măsurători pentru extragerea contururilor. Ce concluzii se pot trage cu privire la rezistenţa la zgomot a operatorilor de extragere a contururilor prezentaţi ? Imaginea degradată cu zgomot Gaussian (normal) de medie nulă şi dispersie 5 poate fi obţinută prin secvenţa de comenzi de mai jos; pentru un zgomot uniform se înlocuieşte funcţia **randn** cu **rand**.

   *≫zg=fix(5*randn(size(x))); xzg=x+zg;*

   ≫p=find(xzg<1); xzg(p)=ones(size(p));

   *≫p=find(xzg>256); xzg(p)=256*ones(size(p));*

7. Reluaţi extragerea contururilor pentru cazul imaginilor degradate de zgomot aplicând înaintea extractoarelor de contur o operaţie de filtrare care să reducă zgomotul.

6.5 Întrebări şi probleme

1. Implementaţi o funcţie Matlab care să realizeze detecţia punctelor în care laplacianul trece prin zero (deci îşi schimbă semnul).

2. Realizaţi un set de funcţii care să genereze o imagine de test compusă din două benzi uniform verticale, de nivel de gri specificat, şi frontiera corectă şi apoi să determine cantitativ precizia cu care a fost extras conturul din imaginea de test degradată cu un zgomot oarecare (de exemplu se poate determina matricea de confuzie: numărul de puncte de tip regiune şi respectiv contur corect identificate, număr de puncte de contur determinate eronat ca puncte de regiune şi număr de puncte de regiune determinate în mod eronat ca puncte de contur).

Capitolul 7

Caracterizarea texturilor

Dicţionarele lingvistice definesc textura ca o proprietate a constituţiei generale a unui corp solid (sau ca constituţia unei roci sau a unui aliaj din punctul de vedere al orientării în spaţiu a părţilor componente) sau ca un atribut caracteristic al unei stofe (sau material textil) care este urzit sau ţesut. Textura este observată în configuraţia structurală a obiectelor, a lemnului, a nisipului, a vegetaţiei, aglomerărilor de granule. Textura este descrisă în termeni lingvistici prin rugozitate, contrast, fineţe, regularitate, termeni a căror traducere matematică nu este clară. Câteva exemple de texturi sunt prezentate în Figura 7.1.

La o scară dată, orice textură are acelaşi aspect, indiferent de zona observată. Corespunzând diferitelor teorii relative la percepţia vizuală a texturii, au fost propuse două modele principale de descriere a texturilor:

- abordarea deterministă se referă la repetarea spaţială a unui motiv de bază, pe diferite direcţii. Acestă abordare corespunde unei viziuni macroscopice, întâlnită de altfel în cazul ţesăturilor, fragmentelor de piele de reptilă şi a modelelor de tip mozaic (mozaicul de pe pardoseală, parchetul). Elementul repetitiv de bază poartă numele de texon sau texel - "texture element" - (prin similaritate cu denumirea de "pixel").
- abordarea probabilistă (statistică) se referă la caracterizarea atributelor anarhice şi omogene în acelaşi timp, care nu ţin nici de un element de bază localizabil (motiv), nici de o frecvenţă principală de repetiţie.

În mod evident, mai există şi o abordare mixtă, care consideră textura ca o structură spaţială constituită din mai multe motive de bază, dipuse în mod aleator (în spaţiu şi în amplitudine). În cele ce urmează vom aborda doar metodele de caracterizare a texturilor

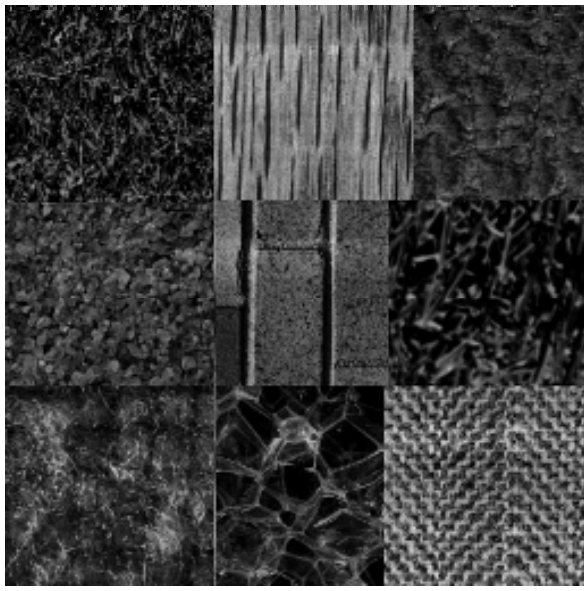

Fig. 7.1: Mozaic din diferite texturi

ce se bazează pe presupunerea caracterului predominant aleator al acestora; vom prezenta astfel caracterizarea texturilor prin matrici de coocurenţă, prin izosegmente şi prin tehnici frecvenţiale.

7.1 Caracterizarea texturilor prin matrici de coocurenţă

Matricea de coocurenţă este formată din medii spaţiale de ordinul doi (deci probabilităţi de apariţie în textură a unor perechi de nivele de gri). Pentru o regiune R a texturii studiate şi pentru un vector de translaţie spaţială plană $\mathbf{t}$ dat, componentele matricii de coocurenţă se definesc pentru toate perechile posibile de nivele de gri (a, b) ca (7.1):

$$M_{\mathbf{t}}(a,b) = Card\,\{(\mathbf{x}, \mathbf{x}+\mathbf{t}) \in R \times R \,|\, f(\mathbf{x}) = a \text{ si } f(\mathbf{x}+\mathbf{t}) = b\} \tag{7.1}$$

Deci $M_{\mathbf{t}}(a, b)$ este numărul de perechi de pixeli din regiunea R, separate de vectorul de translaţie $\mathbf{t}$, care au respectiv valorile nivelelor de gri a şi b. Pentru o imagine cu L nivele de gri, matricea de coocurenţă va fi o matrice pătrată de dimensiune L. În practică, pentru reducerea calculelor, numărul de nivele de gri este redus la 8 sau la 16, printr-o tehnică convenabilă. În concluzie, pentru o aceeaşi textură, matricea de coocurenţă depinde de regiunea considerată, vectorul de translaţie şi gradul (şi modul) de reducere a nivelelor de gri.

Distincţia între texturi diferite poate fi făcută în primul rând prin inspecţia (examinarea de ansamblu) a matricii de coocurenţă, printr-o vizualizare tridimensională a matricii. În plus, se pot defini o serie de indici de natură statistică ce caracterizează distribuţia componentelor matricii de coocurenţă. Aceste mărimi sunt:

Omogenitatea, definită de:

$$O = \frac{1}{N_{nz}} \sum_a \sum_b \left(M_{\mathbf{t}}(a,b)\right)^2 \tag{7.2}$$

N_{nz} este numărul de poziţii nenule din matricea de coocurenţă (deci numărul de perechi diferite de nivele de gri, aşezate la un deplasament $\mathbf{t}$, ce se regăsesc în regiunea considerată). Acest indice (7.2) este cu atât mai mare cu cât se regăseşte de mai multe ori o pereche de nivele de gri (deci unele poziţii ale matricii au valori mult mai mari decât celelalte), ceea ce se întâmplă în general atunci când există o periodicitate în sensul translaţiei.

Omogenitatea locală, definită de:

$$O_{loc} = \frac{1}{N_{nz}} \sum_a \sum_b \frac{1}{1+(a-b)^2} M_{\mathbf{t}}(a,b) \tag{7.3}$$

Contrastul, definit de:

$$C = \frac{1}{N_{nz}(L-1)^2} \sum_{k=0}^{L-1} k^2 \sum_{|a-b|=k} M_{\mathbf{t}}(a,b) \tag{7.4}$$

În această măsură fiecare termen al matricii de coocurenţă este ponderat cu distanţa la diagonală. Astfel se obţine un indice (7.4) ce corespunde interpretării uzuale a contrastului, şi anume are o valoare mare atunci când termenii depărtaţi de diagonala principală a matricii au valori mari (adică atunci când există numeroase treceri de la pixeli luminoşi la pixeli întunecaţi).

Entropia, definită de:

$$H = 1 - \frac{1}{N_{nz} \log N_{nz}} \sum_a \sum_b M_{\mathbf{t}}(a,b) \log M_{\mathbf{t}}(a,b) \delta\left(M_{\mathbf{t}}(a,b)\right) \tag{7.5}$$

Entropia este mică dacă o aceeaşi pereche de pixeli apare de multe ori şi este mare dacă

toate perechile de nivele de gri sunt slab reprezentate. Aceasta este un indicator al dezordinii ce caracterizează textura.

Uniformitatea este definită de:

$$U = \frac{1}{N_{nz}^2} \sum_a M_{\mathbf{t}}^2(a, a) \tag{7.6}$$

Directivitatea este definită de:

$$U = \frac{1}{N_{nz}} \sum_a M_{\mathbf{t}}(a, a) \tag{7.7}$$

Uniformitatea şi directivitatea sunt cu atât mai importante cu cât un unic nivel de gri apare de mai multe ori pe direcţia de translaţie.

Corelaţia este definită de:

$$B = \frac{1}{N_{nz}\sigma_x\sigma_y} \sum_a \sum_b (a - \mu_x)(b - \mu_y) M_{\mathbf{t}}(a, b) \tag{7.8}$$

În expresia (7.8) μ_x şi μ_y sunt mediile pe linii şi respectiv pe coloane ale componentelor matricii de coocurenţă, iar σ_x şi σ_y sunt dispersiile corespunzătoare (pe linii şi pe coloane).

Dintre mărimile definite, se consideră că cele mai importante pentru caracterizarea texturii sunt omogenitatea (7.2), entropia (7.5), contrastul (7.4) şi corelaţia (7.8).

7.2 Caracterizarea texturilor prin izosegmente

Un izosegment de nivele de gri (numit uneori şi plajă sau "runlength") este o mulţime liniară de pixeli consecutivi, având acelaşi nivel de gri, orientată pe o anumită direcţie. Lungimea unui izosegment (a unei plaje) este numărul de pixeli ce formează respectiva mulţime.

Pentru o orientare (direcţie) fixată θ, se poate determina o matrice de izosegmente, ale cărei elemente $M_\theta(a, b)$ reprezintă numărul de izosegmente de lungime b dată, formate din pixeli de un nivel de gri a dat şi orientate pe direcţia θ. Matricea rezultată are L linii (egal cu numărul de nivele de gri din imagine) şi un număr de coloane egal cu lungimea maximă a izosegmentelor pe direcţia dată (N_θ).

Aspectul acestei matrici este caracteristic unui anumit tip de textură; în plus, ca şi pentru matricea de coocurenţă, se pot defini diferite mărimi caracteristice:

- $N_{iz} = \sum_{a=0}^{L-1} \sum_{b=1}^{N_\theta} M_\theta(a,b)$ (numărul de izosegmente)
- $RF_1 = \frac{1}{N_{iz}} \sum_{a=0}^{L-1} \sum_{b=1}^{N_\theta} \frac{M_\theta(a,b)}{b^2}$ (proporţia plajelor mici)
- $RF_2 = \frac{1}{N_{iz}} \sum_{a=0}^{L-1} \sum_{b=1}^{N_\theta} b^2 M_\theta(a,b)$ (proporţia plajelor lungi)
- $RF_3 = \frac{1}{N_{iz}} \sum_{a=0}^{L-1} \left(\sum_{b=1}^{N_\theta} M_\theta(a,b) \right)^2$ (heterogenitatea nivelelor de gri; măsoară dispersia plajelor între nivelele de gri)
- $RF_4 = \frac{1}{N_{iz}} \sum_{b=1}^{N_\theta} \left(\sum_{a=0}^{L-1} M_\theta(a,b) \right)^2$ (heterogenitatea lungimilor plajelor; măsoară dispersia plajelor între lungimi)
- $RF_5 = \frac{N_{iz}}{N_{reg}}$ (procentajul plajelor; este raportul între numărul de izosegmente şi numărul de pixeli ai regiunii)

7.3 Caracterizarea spectrală a texturilor

Transformările [integale] ale imaginilor transpun informaţia acestora într-un domeniu spectral. Zonele de interes din spectru sunt apoi extrase cu ajutorul a diferite măşti (aperturi). Zonele de inters astfel extrase pun în evidenţă diferite caracteristici: frecvenţele [spaţiale] înalte pot fi folosite pentru detecţia contururilor, regiunile orientate sunt puse în evidenţă de existenţa unor direcţionări predominante.

Pentru texturi, se urmăreşte în principal distribuţia spectrală a energiei. Spectrul de energie $S(u,v)$ (7.10) este modulul pătrat al spectrului Fourier[1] $F(u,v)$ (7.9).

$$F(u,v) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) e^{-j2\pi\left(\frac{um}{M} + \frac{vn}{N}\right)},\ u = \overline{0, M-1} \text{ si } v = \overline{0, N-1} \tag{7.9}$$

$$S(u,v) = |F(u,v)|^2 \tag{7.10}$$

[1] Transformata Fourier face trecerea din domeniul spaţial (reprezentat de coordonatele spaţiale (m,n)) în domeniul frecvenţial (reprezentat de coordonatele de frecvenţe spaţiale (u,v)).

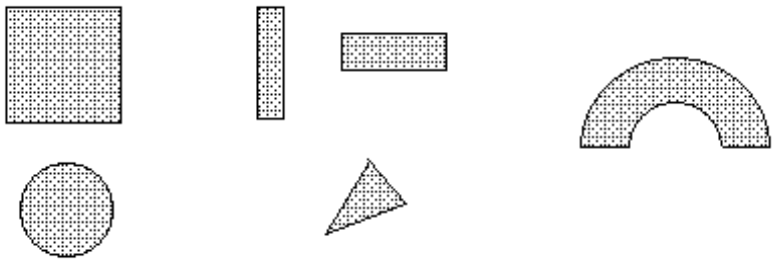

Fig. 7.2: Măşti pentru determinarea benzilor de măsurare a distribuţiei energetice spectrale a texturilor.

Caracteristicile energetice se pot măsura cu mai multe tipuri de măşti [7] (vezi Figura 7.2):

- măşti unghiulare (orientate), utile pentru detecţia caracteristicilor de orientare şi determinarea direcţiilor predominante
- măşti simetrice, de tip trece-jos sau trece-bandă, ce măsoară energia în benzi de frecvenţă

Una dintre metodele clasice de analiză energetică a texturilor calculează un vector caracteristic al entropiilor energetice pentru benzi de frecvenţă de tip trece jos, cu bandă de trecere din ce în ce mai mare. Pentru fiecare bandă de frecvenţă, energiile spectrale se normează la suma acestora şi se calculează entropia valorilor rezultate; la rândul ei, acestă entropie se normează la entropia maxim posibilă pentru o zonă spectrală de energie uniformă.

Analiza energetică spectrală a texturilor prin folosirea de măşti în spectrul energetic este o metodă valabilă însă numai în cazul în care textura ocupă o suprafaţă mare (şi deci întreaga imagine este o textură). În practică, se doreşte caracterizarea fiecărui pixel (de natură necunoscută) al imaginii şi încadrarea sa într-un anume tip de clasă texturală. Pentru aceasta se calculează doar spectrul unor vecinătăţi spaţiale centrate în pixelul de interes (deci ferestre de analiză de dimensiune impară) şi acest spectru este caracterizat de diferite mărimi statistice (momente de diferite ordine, centrate sau necentrate, amplitudini maxime / minime, variaţie a amplitudinii, etc.).

7.4 Desfăşurarea lucrării

1. Vizualizaţi cele 10 texturi disponibile, stocate în format IMG în fişierele de la

t0_256.img la t9_256.img. Texturile sunt imagini de 256 x 256 pixeli, cu 256 nivele de gri.

≫x=imgread('textura', 256);

≫image(x), colormap(gray(256))

2. Urmăriţi modul de folosire şi implementarea funcţiilor de calcul a matricilor de coocurenţă (**cooc**) şi izosegmente (**izoseg**). Urmăriţi modul de folosire şi implementarea funcţiilor de calcul al parametrilor texturali derivaţi din matricile de coocurenţă (**mas_cooc**) şi izosegmente (**mas_izo**).

3. Pentru caracterizarea texturilor prin matrice de coocurenţă, trebuie urmărite următoarele etape: pentru fiecare textură, se extrag mai multe regiuni diferite, după care numărul de nivele de gri se reduce la 16 prin trunchere (pentru uşurarea calculelor).

   ≫reg=x(minx:maxx, miny:maxy);

   ≫reg=fix((reg-1)/16)+1;

4. Pentru fiecare regiune extrasă se calculează matricile de coocurenţă ce corespund mai multor vectori de translaţie (de exemplu [1 0],[1 1], [0 1]) şi se vizualizează:

   ≫c10=cooc(reg,[1 0]);

   ≫c11=cooc(reg,[1 1]);

   ≫c01=cooc(reg,[0 1]);

   ≫figure;subplot(2,2,2),surf(c10),view(-20,45)

   ≫subplot(2,2,3),surf(c01),view(-20,45)

   ≫subplot(2,2,4),surf(c11),view(-20,45)

5. Pentru fiecare matrice de coocurenţă astfel calculată se calculează descriptorii texturali:

   ≫m=mas_cooc(cij)

6. Verificaţi dacă pentru aceeaşi textură măsurile calculate de baza matricii de coocurenţă şi aspectul grafic al matricii de coocurenţă sunt aceleaşi pentru zone diferite (verificarea ipotezei de staţionaritate a procesului aleator ce produce textura) şi dacă acestea sunt diferite pentru texturi diferite.

7. Pentru caracterizarea texturilor prin matrice de izosegmente se reia extragerea a mai multor regiuni diferite pentru fiecare textură, pentru care numărul de nivele de gri se reduce la 16 prin trunchere (pentru uşurarea calculelor).

   ≫reg=x(minx:maxx, miny:maxy);

   ≫reg=fix((reg-1)/16)+1;

8. Pentru fiecare regiune extrasă se calculează matricile de izosegmente ce corespund mai multor orientări (de exemplu 0°, 90°, 45° şi 135°) şi se vizualizează:

   ≫iz0=izoseg(reg);

   ≫iz90=izoseg(reg;

   ≫iz45=izoseg(rot45(reg));

   ≫iz135=izoseg(rot45(reg);

   ≫figure;subplot(2,2,1),surf(iz0),view(-20,45)

   ≫subplot(2,2,2),surf(iz90),view(-20,45)

   ≫subplot(2,2,3),surf(iz45),view(-20,45)

   ≫subplot(2,2,3),surf(iz135),view(-20,45)

9. Pentru fiecare matrice de izosegmente astfel calculată se calculează descriptorii texturali:

   ≫m=mas_ izo(izxxx)

10. Verificaţi dacă pentru aceeaşi textură măsurile calculate de baza matricii de izosegmente şi aspectul grafic al matricii de izosegmente sunt aceleaşi pentru zone diferite (verificarea ipotezei de staţionaritate a procesului aleator ce produce textura) şi dacă acestea sunt diferite pentru texturi diferite.

11. Urmăriţi modul de folosire a funcţiilor asociate calculului spectrului Fourier de amplitudine (**fft2**, **fftshift**) şi energie (**abs**).

12. Pentru fiecare textură disponibilă, calculaţi spectrul Fourier şi vizualizaţi-l cu compresie logaritmică a valorilor:

    ≫x=imgread(tx,256);

    ≫y=abs(fftshift(fft2(x)));

    ≫figure,image(log(abs+1)+1),colormap(gray(256))

13. Pentru fiecare textură calculaţi distribuţia spectrală de energie folosind funcţia **mas_four** şi diferite dimensiuni ale vectorului de caracteristici (16, 32, ...,128).

    ≫mas=mas_four(y, dim);

    ≫figure; plot(mas)

7.5 Întrebări şi probleme

1. Care este legătura dintre matricea de izosegmente pe o anumită direcţie, pentru o regiune R, şi histograma (funcţia de densitate de probabilitate a nivelelor de gri) zonei R ?

2. Care este legătura dintre matricea de coocurenţă pentru un vector de translaţie fixat, pentru o regiune R, şi histograma (funcţia de densitate de probabilitate a nivelelor de gri) zonei R ?

3. Cum depinde timpul de calcul al matricii de coocurenţă (sau al matricii de izosegmente) de numărul de nivele de gri pe care sunt reprezentaţi pizelii ? Efectuaţi măsurători ale timpului de rulare pentru diferite valori ale numărului maxim de nivele de gri (8, 16, 32, etc) folosind funcţiile Matlab **tic** şi **toc**.

4. Implementaţi o funcţie Matlab care să calculeze distribuţia spectrală de energie pentru măşti de tip sector circular ("felii de plăcintă").

5. Verificaţi invarianţa descriptorilor texturali folosiţi (din matricea de coocurenţă, din matricea de izosegmente) şi a distribuţiei spectrale de energie faţă de transformările punctuale de îmbunătăţire a imaginilor (modificare punctuală a nivelelor de gri).

Capitolul 8

Caracterizarea regiunilor

Prin parametru de formă înţelegem în general un scalar sau o funcţie (cu suport unidimensional sau bidimensional) asociate unei forme plane pe care o caracterizează; forme asemănătoare sunt caracterizate de parametri de formă de valori apropiate; formele diferite prezintă diferenţe mari între parametrii de formă asociaţi. Parametrii de formă compun un fel de fişă de identitate a formei respective, pe baza căreia această formă poate fi recunoscută în mod unic. În mod ideal, aceşti parametri trebuie să fie invarianţi la translaţie, rotaţie şi scalare. Tehnicile de recunoaştere a formelor sau de clasificare sunt precedate întotdeauna de o etapă de extragere a parametrilor de formă (sau a caracteristicilor formei).

Pentru analiza imaginilor, o formă este o funcţie de două variabile, cu suport compact $f(x, y) : K \rightarrow \mathbf{R}$; în general valorile acestei funcţii sunt binare (0 sau 1), descriind deci o parte a unei imagini binare (zona din imagine în care se află obiectul de interes). Atunci funcţia poate fi văzută ca o funcţie caracteristică a formei, asemănătoare funcţiei caracteristice a unei mulţimi.

În cele ce urmează vom prezenta câţiva parametri de formă clasici.

8.1 Parametri geometrici

Această categorie de parametri se bazează pe măsura unor atribute geometrice simple: arie (S), perimetru (P), număr de găuri, numărul lui Euler (numărul de regiuni conexe – numărul de găuri). Cum nu toate aceste numere sunt invariante şi caracteristice în mod unic unei anume forme, au apărut combinaţii de tip raport.

Raportul de compacitate (numit uneori şi factor de formă) este raportul dintre pătratul

perimetrului şi suprafaţa formei:

$$\kappa = \frac{P^2}{4\pi S} \tag{8.1}$$

Pentru o formă circulară raportul este unitar; cu cât numărul κ este mai apropiat de această valoare, cu atât mai mult forma seamănă cu un disc (pătratul are un raport de compacitate $\kappa = 1.273$).

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

$$c = \frac{R}{r} \tag{8.2}$$

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

8.2 Momente statistice şi invarianţi

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

$$m_{pq} = \iint\limits_K f(x,y) x^p y^q dx dy,\ p,q = 0,1,2,... \tag{8.3}$$

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

În cazul imaginilor binare, coordonatele sunt discrete şi funcţia este o funcţie caracteristică; formula momentelor (8.3) devine

$$m_{pq} = \sum_{f(x,y)\neq 0} \sum x^p y^q \tag{8.4}$$

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

$$g(x,y) = \sum_{p+q\leqslant N} \sum g_{pq} x^p y^q \tag{8.5}$$

Găsirea acestor scalari se face prin egalarea momentelor cunoscute ale lui $f(x,y)$ cu momentele lui $g(x,y)$ dată de expresia (8.5). Rezolvând sistemul de ecuaţii cuplate ce se formează, se pot obţine relaţiile căutate; calculul trebuie însă refăcut, din cauza cuplării ecuaţiilor, ori de câte ori se doreşte trecerea la o aproximare mai bună a formei f , mărind valoarea lui N. Această cuplare provine din cauza folosirii unei baze neortogonale (aşa cum este familia de polinoame $x^p y^q$) pentru calculul momentelor; problema a fost rezolvată prin folosirea proiecţiilor pe baza de polinoame Legendre.

Trebuie însă remarcat că folosirea momentelor statistice pentru caracterizarea unei forme nu asigură îndeplinirea a nici unuia dintre principiile de invarianţă căutate; de aceea au fost introduse momente statistice invariante.

Momentele statistice invariante la translaţie sunt momentele statistice centrate:

$$\mu_{pq} = \sum_{f(x,y)\neq 0} \sum (x-\overline{x})^p (y-\overline{y})^q \tag{8.6}$$

Momentele statistice invariante la translaţie şi scalare sunt definite de:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}},\ \gamma = 1 + \frac{p+q}{2} \tag{8.7}$$

Invarianţii la translaţie, scalare şi rotaţie ai unei forme, obţinuţi în condiţiile folosirii unor momente statistice de ordin cel mult 3 ($N = 3$), sunt în număr de 7 şi sunt exprimaţi de:

$$\Phi_1 = \eta_{20} + \eta_{02} \tag{8.8}$$

$$\Phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \tag{8.9}$$

$$\Phi_3 = (\eta_{30} - 3\eta_{12})^2 + (\eta_{03} - 3\eta_{21})^2 \tag{8.10}$$

$$\Phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{03} + \eta_{21})^2 \tag{8.11}$$

$$\begin{aligned}\Phi_5 = {} & (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12}) \left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{03} + \eta_{21})^2\right] + \\ & + (\eta_{03} - 3\eta_{21})(\eta_{03} + \eta_{21}) \left[(\eta_{03} + \eta_{21})^2 - 3(\eta_{30} + \eta_{12})^2\right]\end{aligned} \tag{8.12}$$

$$\Phi_6 = (\eta_{20} - \eta_{02}) \left[(\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2\right] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{03} + \eta_{21}) \tag{8.13}$$

$$\begin{aligned}\Phi_7 = {} & (\eta_{30} - 3\eta_{21})(\eta_{03} + \eta_{21}) \left[(\eta_{03} + \eta_{21})^2 - 3(\eta_{30} + \eta_{12})^2\right] - \\ & - (\eta_{03} - 3\eta_{21})(\eta_{30} + \eta_{12}) \left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{03} + \eta_{21})^2\right]\end{aligned} \tag{8.14}$$

Iniţial (mijlocul anilor '60) aceşti invarianţi au fost folosiţi pentru recunoaşterea caracterelor mari de tipar, cu rezultate modeste. Eficienţa lor constă însă în modul rapid de calcul şi posibilitatea de a le utiliza cu succes pentru recunoaşterea formelor geometrice convexe.

Folosind momentele invariante, se mai pot deduce alte atribute: excentricitatea suprafeţei (8.15), care măsoară gradul de uniformitate al distribuţiei punctelor formei în jurul centrului de greutate şi orientarea suprafeţei, caracterizată de unghiul θ faţă de orizontală al axei faţă de care momentul inerţie al formei este minim (8.16).

$$\varepsilon = \frac{\Phi_2}{\mu_{00}} \tag{8.15}$$

$$\theta = \frac{1}{2} \arctan \frac{2\mu_{11}}{\mu_{20} - \mu_{02}} \tag{8.16}$$

8.3 Skeletonul morfologic

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

$$B_{\mathbf{x}}(r) \subseteq A \tag{8.17}$$

$$B_{\mathbf{x}}(r) \subseteq B_{\mathbf{x}'}(r') \subseteq A \Leftrightarrow \begin{cases} r = r' \\ \mathbf{x} = \mathbf{x}' \end{cases} \tag{8.18}$$

Deci discul maximal trebuie să fie inclus în formă (8.17) şi nici un alt disc inclus în formă să nu îl includă, sau, dacă îl include, să fie identic cu acesta (8.18). O formă poate avea mai multe discuri maximale.

Skeletonul unei forme este mulţimea centrelor discurilor maximale ale formei. Ca exemple simple, skeletonul unui disc este centrul său, skeletonul unui pătrat este reuniunea diagonalelor sale.

Calculul skeletonului unei forme reprezentate în spaţiul discret se poate face prin formula Lantuejoul; skeletonul formei A, $SK(A)$, este format din reuniunea unui număr finit de seturi skeleton:

$$SK(A) = \bigcup_{n=0}^{N_{\max}} S_n(A) \tag{8.19}$$

$$S_n(A) = (A \ominus nB) - (A \ominus nB) \circ B \tag{8.20}$$

Ordinul $N_{\max}$ corespunde momentului în care toate seturile skeleton succesive devin nule, moment marcat de $A \ominus N_{\max}B = \emptyset$; elementul structurant nB este iterarea de n ori a elementului structurant B, $nB = B \oplus ... \oplus B$ (de n ori). Elementul structurant folosit este în general o expresie discretă a discului unitar (deci element structurant V_4 sau V_8). Figura 8.1 prezintă skeletonul unei forme discrete oarecare.

Fig. 8.1: Skeleton al unei forme discrete, reprezentat prin reuniunea seturilor skeleton şi informaţia de apartenenţă a fiecărui punct la un anumit set skeleton.

Reconstrucţia formei din skeleton se face după formula

$$A = \bigcup_{n=0}^{N_{\max}} S_n(A) \oplus nB \tag{8.21}$$

Se pot realiza şi reconstrucţii parţiale (aproximări ale mulţimii A) prin neglijarea seturilor skeleton ce reprezintă detaliile (acestea sunt seturile skeleton de ordin mic); aproximarea de ordin k a formei înseamnă deci ignorarea primelor k seturi skeleton din reconstrucţie:

$$\widetilde{A_k} = \bigcup_{n=k}^{N_{\max}} S_n(A) \oplus nB \tag{8.22}$$

Skeletonul este invariant la translaţie, nu este conex (chiar dacă forma A este conexă), nu este comutativ cu operaţia de reuniune a formelor. Transformarea skeleton este idempotentă ($SK(SK(A)) = SK(A)$) şi antiextensivă ($SK(A) \subseteq A$). Seturile skeleton sunt disjuncte două câte două ($S_i(A) \cap S_j(A) = \emptyset$, $\forall i \neq j$).

Folosirea skeletonului morfologic pentru recunoaşterea fomelor este restricţionată de puternica sa sensibilitate la zgomote (o mică schimbare a formei duce la o modificare semnificativă a skeletonului[1]), şi de variaţia la rotaţia şi scalarea obiectelor (pentru reprezentări în spaţiul discret). În acelaşi timp, folosirea elementului structurant de tip disc unitar aduce o puternică dependenţă de metrica folosită în definirea sa (să nu uităm că într-un spaţiu discret, metrica Euclidiană nu este cea mai favorabilă) şi produce elemente structurante pătrate (V_8) sau în cruce (V_4), ce pot fi cu greu interpretate ca discuri. Acestea au condus la folosirea unei clase de elemente structurante care să nu provină din noţiunea de metrică - elementele structurante generalizate.

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

8.4 Desfăşurarea lucrării

Se va utiliza programul SAIN - sistem de analiză a formelor binare plane, pentru a realiza prelucrări mai rapide decât cele posibile în Matlab. SAIN este comandat din meniurile de operaţii; fiecare operaţie este aplicată formei binare care se află în fereastra-document activă.

1. Se încarcă configuraţia de lucru pentru lucrarea curentă (*Options - Charger options - regiuni.cfg*).
2. Pentru fiecare formă disponibilă se vor calcula următorii parametri:

   aria (afişată în fereastră odată cu imaginea)

   perimetrul (afişat în fereastră odată cu imaginea)

   raportul de formă calculat conform (8.1)

   raportul de circularitate calculat după (8.2) (*Parametres - Circularites - Radiale*)

   invarianţii de formă (8.8 - 8.12) deduşi pe baza momentelor statistice (*Parametres - Moments statistiques*)

   Se va urmări ca pentru obiectele asemănătoare valorile parametrilor calculaţi să fie asemănătoare, iar pentru obiecte diferite valorile parametrilor să fie diferite. Parametrii pot fi notaţi şi apoi se pot calcula distanţe între obiecte pentru a verifica procentul de bună recunoaştere folosind o regulă de decizie simplă de tipul celui mai apropiat vecin (se poate folosi Matlab).
3. Pentru fiecare formă disponibilă se calculează skeletonul morfologic prin elementul structurant V_8(*Parametres - Morphologiques - Squelette*). Se va urmări numărul de puncte din care este alcătuit skeletonul şi modul în care acesta este "centrat" în obiect (se justifică deci denumirea de axă mediană ?). Se va urmări efectul micilor variaţii ale formei asupra skeletonului.

8.5 Întrebări şi probleme

1. Scrieţi o funcţie Matlab pentru calculul skeletonului morfologic al unei forme binare.
2. Scrieţi o funcţie Matlab pentru calculul orientării unei forme plane. Afişaţi apoi într-o aceeaşi figură forma plană şi axa având orientarea determinată. Comentaţi precizia metodei.
3. Scrieţi o funcţie Matlab pentru calculul raportului de compacitate a unei forme plane.

Capitolul 9

Caracterizarea contururilor

9.1 Semnătura formei

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

Fig. 9.1: Semnătura unui pătrat ca funcţie de unghiul la centru; punctul de referinţă este centrul de greutate, axa de referinţă este orizontala.

Este posibil ca, pentru formele concave, semnătura în funcţie de unghiul la centru să nu poată fi calculată, deoarce, pentru anumite unghiuri θ, intersecţia dreptei de direcţie θ

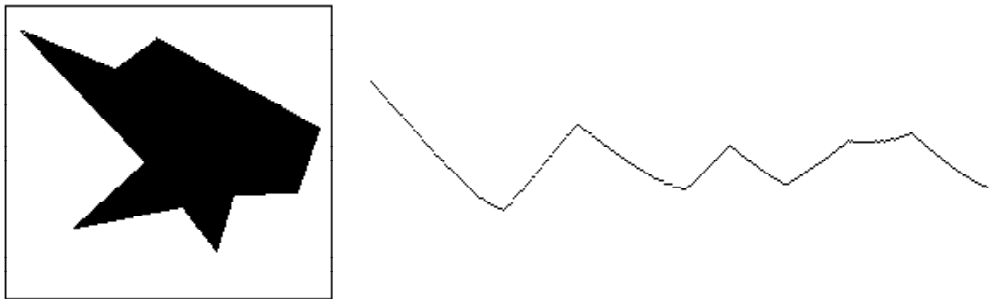

Fig. 9.2: Semnătura unei forme poligonale în funcţie de abscisa curbilinie; punctul de referinţă este centrul de greutate, axa de referinţă este orizontala.

cu conturul să fie formată din mai multe puncte. Această problemă nu apare în cazul semnăturii în funcţie de abscisa curbilinie. Ca o restricţie generală, în cazul folosirii semnăturii bazate pe unghi, trebuie ca punctul de referinţă $\mathbf{x}$ să aparţină nucleului formei. Nucleul formei K, $Ker(K)$ este definit prin:

$$Ker(K) = \{\mathbf{x} | \forall \mathbf{y} \in K, [\mathbf{xy}] \subseteq K\} \tag{9.1}$$

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

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

$$Y(K) = \sup_{\mathbf{x} \in \mathbf{K}} \inf_{\theta \in [0;\pi]} \frac{d_{\mathbf{x}}(\theta)}{d_{\mathbf{x}}(\theta + \pi)} \tag{9.2}$$

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

Raportul de circularitate este definit ca:

$$C(K) = \frac{\sup_\theta \left(d_{\mathbf{x}}(\theta) + d_{\mathbf{x}}(\theta + \pi)\right)}{\inf_\theta \left(d_{\mathbf{x}}(\theta) + d_{\mathbf{x}}(\theta + \pi)\right)} \tag{9.3}$$

Funcţia $h_K(\theta) = d_{\mathbf{x}}(\theta) + d_{\mathbf{x}}(\theta + \pi)$ se numeşte suportul formei, şi este diametrul formei pe direcţia θ.

9.2 Descriptori Fourier de contur

Frontiera unui obiect, odată ce a fost determinată, poate fi văzută ca o pereche de semnale unidimensionale $x(t)$ şi $y(t)$ în funcţie de abscisa curbilinie pe contur t (a se vedea figura

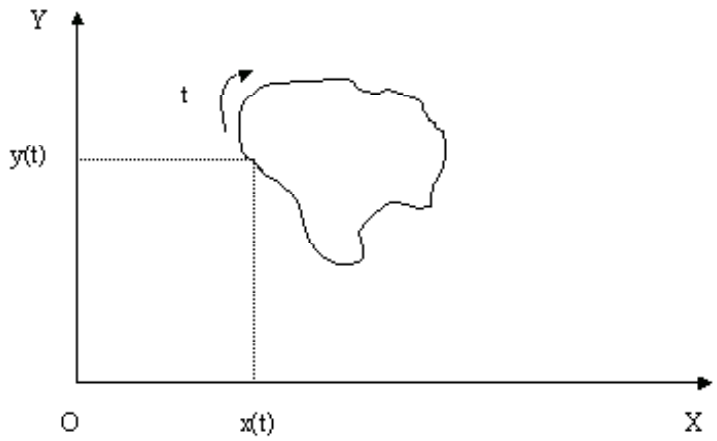

Fig. 9.3: Descriptorii Fourier ai frontierei unui obiect.

9.3). Astfel, frontiera respectivă poate fi reprezentată folosind tehnici uzuale pentru semnale unidimensionale. Pentru un contur discret, se poate defini semnalul complex:

$$u(n) = x(n) + jy(n), \qquad n = 0, 1, ..., N-1$$

care, pentru un contur închis, este un semnal periodic de perioadă N. Semnalul $u(n)$ poate fi caracterizat prin transformata sa Fourier discretă $a(k)$, definită ca:

$$a(k) = \sum_{n=0}^{N-1} u(n) \exp\left(-\frac{2\pi jkn}{N}\right)$$

Coeficienţii complecşi $a(k)$ se numesc *descriptorii Fourier* ai frontierei considerate. În figura 9.4 este prezentat un exemplu de descriptori Fourier ai unui contur.

Descriptorii Fourier ai unui contur prezintă anumite proprietăţi de invarianţă la operaţii geometrice de bază (scalare, rotaţie, translaţie) care îi fac să fie extrem de atractivi pentru recunoaşterea de forme. În cele ce urmează sunt prezentate câteva dintre proprietăţile cele mai importante ale descriptorilor Fourier, ale căror demonstraţii sunt lăsate ca exerciţiu.

9.2.1 Translaţie

Fie $u(n)$ semnalul complex asociat unei frontiere, şi fie $a(k)$ descriptorii Fourier ai acesteia. Fie $\tilde{u}(n)$ semnalul asociat frontierei obiectului translatat cu (x_0, y_0). Putem scrie $\tilde{u}(n) =$

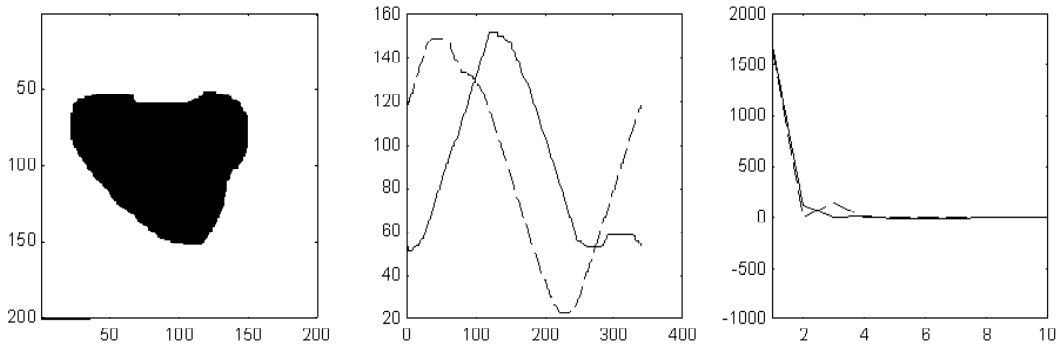

Fig. 9.4: Exemplu de descriptori Fourier: obiect original, func]ctia contur şi primii zece descriptori Fourier

$u(n) + u_0$, cu $u_0 = x_0 + jy_0$. Atunci, descriptorii Fourier ai obiectului translatat $\tilde{a}(k)$ se obţin în funcţie de cei ai obiectului original astfel:

$$\tilde{a}(k) = a(k) + u_0\delta(k).$$

Cu alte cuvinte, efectul unei translaţii asupra descriptorilor Fourier ai unui obiect este modificarea coeficientului de curent continuu.

9.2.2 Scalare

Fie $\tilde{u}(n)$ semnalul asociat frontierei obiectului scalat cu parametrul α: $\tilde{u}(n) = \alpha u(n)$. Atunci,

$$\tilde{a}(k) = \alpha a(k).$$

Altfel spus, o scalare a obiectului conduce la o scalare a descriptorilor săi Fourier.

9.2.3 Rotaţie

Fie $\tilde{u}(n)$ semnalul asociat frontierei obiectului rotit cu unghiul θ_0: $\tilde{u}(n) = u(n)\exp(j\theta_0)$. Descriptorii Fourier ai obiectului rotit devin:

$$\tilde{a}(k) = a(k)\exp(j\theta_0).$$

O rotaţie a obiectului induce, deci, o deviaţie constantă în faza descriptorilor săi Fourier.

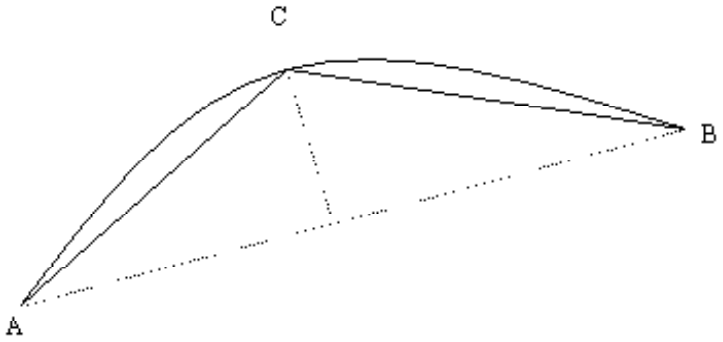

Fig. 9.5: Exemplu de aproximare poligonală a unei curbe

9.2.4 Schimbarea originii

În cazul în care se schimbă originea obiectului, semnalul care îi va descrie frontiera va fi $\tilde{u}(n) = u(n - n_0)$ cu n_0 constant, iar descriptorii săi Fourier pot fi scrişi:

$$\tilde{a}(k) = a(k) \exp\left(-\frac{2\pi j n_o k}{N}\right),$$

deci o schimbare a originii conduce la o modulaţie indusă în faza descriptorilor săi Fourier.

9.3 Poligonalizarea contururilor

Operaţia de poligonalizare a contururilor este utilă în vederea interpretării formelor. Un algoritm posibil de aproximare poligonală a unei curbe AB este următorul (v. figura 9.5): pentru fiecare punct D al curbei se calculează distanţa la segmentul de dreaptă AB. Se determină astfel punctul C caracterizat de distanţa la segmentul AB maximă. Fie d_C această distanţă. Dacă $d_C < T$, unde T este un prag ales de utilizator, atunci punctul C este reţinut şi se repetă procedura pentru cele două segmente rezultate AC şi CB. În caz contrar, curba se înlocuieşte cu segmentul AB. Pragul T controlează eroarea indusă de aproximarea poligonală. Alegându-se un T mare, contururile vor fi aproximate cu un număr redus de segmente, deci aproximarea este mai brutală. Pentru T mic, va rezulta un număr ridicat de segmente.

9.4 Desfăşurarea lucrării

1. Pentru calculul semnăturilor obiectelor şi a aproximărilor lor poligonale se va utiliza programul SAIN - sistem de analiză a formelor binare plane, pentru a realiza prelucrări mai rapide decât cele posibile în Matlab. Se încarcă configuraţia de lucru pentru lucrarea curentă (*Options - Charger options - contur.cfg*).

2. Pentru imaginile binare disponibile se calculează semnăturile faţă de centrul de greutate, indexate prin unghiul la centru sau prin abscisa curbilinie (*Parametres - Signatures*). Numărul de puncte pentru care se calculează semnăturile se poate controla prin meniu (*Options - Approximations*).

3. Pentru imaginile binare disponibile se va calcula poligonul de aproximare (*Operations - Polygonaliser*). Precizia de aproximare se poate controla prin meniu (*Options - Approximations*).

4. Se vor încărca şi vizualiza pe rând toate imaginile ce conţin obiectele binare ce vor fi utilizate pentru studiul descriptorilor Fourier:

   ≫[a, map]=bmp8rd('nume_fisier');

5. Pentru fiecare imagine, se va calcula funcţia contur a obiectului conţinut:

   ≫ [za, contura]=descfour(a);

6. Se va vizualiza conturul obiectului şi se vor afişa părţile reală şi imaginară ale funcţiei sale de contur:

   ≫figure; image(contura+1); colormap(gray(2))

   ≫figure; plot(real(za)); hold on; plot(imag(za),"–");

7. Se calculează şi se afişează descriptorii Fourier ai obiectului:

   ≫Za=fft(za);

   ≫figure; plot(real(Za)); hold on; plot(imag(Za),"–");

8. Pentru fiecare din cele 4 obiecte, se vor verifica proprietăţile de invarianţă a descriptorilor Fourier la operaţii geometrice.

9.5 Întrebări şi probleme

1. Să se scrie o funcţie Matlab care să compare două obiecte în funcţie de descriptorii lor Fourier.

2. Să se scrie o funcţie Matlab care realizează aproximarea poligonală a conturului unui obiect.

ANEXE

Anexa A

Ghid de utilizare a funcţiilor Matlab

Această anexă cuprinde întreaga documentaţie necesară implementării şi utilizării funcţiilor Matlab dezvoltate special pentru experimentele de analiza imaginilor. Unele funcţonalităţi ale acestor funcţii sunt acoperite şi de funcţii standard (incluse în distribuţia Matlab de bază) sau specializate (din diferite toolbox-uri) ale Matlab. Pentru fiecare funcţie este prezentat modul de utilizare (argumente de apel, valori returnate, valori implicite ale parametrilor) şi codul de implementare. Codul de implementare prezentat nu permite utilizarea sistemului de help on-line al Matlab. Deşi au fost făcute eforturi pentru a asigura o deplină funcţionalitate a codului prezentat, nu putem garanta în nici un fel faptul că acesta este total lipsit de erori. Autorii nu îşi asumă deci responsabilitatea pentru efectele nedorite rezultate ca urmare a utilizării acestuia.

A.1 Citire-scriere fişiere imagine

A.1.1 bmp8rd

Citirea fişierelor în format .bmp (Microsoft Windows Bitmap): imagini indexate cu nivele de gri sau color, cu tabelă de culoare, stocate în format de 8 biţi per pixel. Imaginea este citită într-o matrice şi la toate componentele este adăugată o unitate pentru a se permite afişarea imaginii cu funcţia **image** a Matlab. Utilizarea funcţiei:

[nume_matrice_imagine, nume_tabel_culoare]=bmp8rd('nume_fisier_imagine');

unde *'nume_fisier_imagine'* trebuie dat între apostrofuri şi este numele DOS complet şi valid al unui fişier de imagine; dacă extensia nu este precizată, se presupune implicit extensia .bmp. Imaginea de indecşi este plasată în matricea *nume_matrice_imagine*, iar

tabelul de culoare ce îi este asociat este plasat în matricea cu 3 coloane *nume_ tabel_ culoare*.
Cod de implementare:

```
function [x,map]=bmp8rd(filename);
if (nargin~=1)
  help bmp8rd
  error('Necesita un nume de fisier ca argument !');
end;
if (isstr(filename)~=1)
  help bmp8rd
  error('Argumentul trebuie sa fie un sir !');
end;
if (isempty(findstr(filename,'.'))==1)
  filename=[filename,'.bmp'];
end;
fid=fopen(filename,'rb','l');
if (fid==-1)
  error(['Error opening ',filename,' for input.']);
end;
bfType=fread(fid,2,'char');
setstr(bfType);
if (bfType~=['B';'M'])
  fclose(fid);
  error('Not a BMP file.');
end;
bfSize=fread(fid,1,'long');
bfReserved1=fread(fid,1,'short');
bfReserved2=fread(fid,1,'short');
bfOffBits=fread(fid,1,'long');
biSize=fread(fid,1,'long');
if (biSize~=40)
  fclose(fid);
  error('Not a MS Windows Device Independent Bitmap BMP file.');
end;
biWidth=fread(fid,1,'long');
biHeight=fread(fid,1,'long');
biPlanes=fread(fid,1,'short');
biBitCount=fread(fid,1,'short');
biCompression=fread(fid,1,'long');
biSizeImage=fread(fid,1,'long');
biXPels=fread(fid,1,'long');
biYPels=fread(fid,1,'long');
```

```
biClrUsed=fread(fid,1,'long');
biClrImportant=fread(fid,1,'long');
if (biClrUsed==0)
  nColors=2.^biBitCount;
else
  nColors=biClrUsed;
end;
if (biClrUsed>256)
  map=[];
else
  map=fread(fid,4*nColors,'uchar');
  map=reshape(map,4,nColors);
  map=map';
  map=map(:,1:3);
  map=fliplr(map);
end;
map=map./255;
rest=rem(biWidth,4);
if (rest~=0)
  rest=4-rest;
end
if (biBitCount==8)
  temp=fread(fid,[biWidth+rest,biHeight],'uchar');
  x=rot90(temp(1:biWidth,:))+1;
else
  help bmp8rd
  error('Acesta nu e un BMP cu 8 bpp !');
end;
fclose(fid);
```

A.1.2 bmp8wr

Scrierea fişierelor în format .bmp (Microsoft Windows Bitmap): imagini indexate cu nivele de gri sau color, cu tabelă de culoare, stocate în format de 8 biţi per pixel. Imaginea este citită dintr-o matrice şi din toate componentele este scăzută o unitate (deci se presupune că valorile indecşilor sunt în intervalul 1-256). Utilizarea funcţiei:

bmp8wr(nume_matrice_imagine, nume_tabel_culoare, 'nume_fisier_imagine');

unde *'nume_fisier_imagine'* trebuie dat între apostrofuri şi este numele DOS complet şi valid al unui fişier de imagine; dacă extensia nu este precizată, se presupune implicit

extensia .bmp. Imaginea de indecşi este preluată din matricea *nume_matrice_imagine*, iar tabelul de culoare ce îi este asociat este specificat de *nume_tabel_culoare*. Cod de implementare:

```
function bmp8wr(x,map,filename);
if (nargin~=3)
  help bmp8wr
  error('Necesita 3 argumente !');
end;
if (isstr(filename)~=1)
  help bmp3wr
  error('Al treilea argument trebuie sa fie sir !');
end;
if (isempty(findstr(filename,'.'))==1)
  filename=[filename,'.bmp'];
end;
fid=fopen(filename,'wb','l');
if (fid==-1)
  error(['Error opening ',filename,' for output.']);
end;
fwrite(fid,['B';'M'],'char');
[biHeight,biWidth]=size(x);
rest=4-rem(biWidth,4);
if (rest==4)
  rest=0;
end
bfSize=biHeight*(biWidth+rest)+54;
fwrite(fid,bfSize,'long');
bfReserved1=0;
fwrite(fid,bfReserved1,'short');
bfReserved2=0;
fwrite(fid,bfReserved2,'short');
bfOffBits=54;
fwrite(fid,bfOffBits,'long');
biSize=40;
fwrite(fid,biSize,'long');
fwrite(fid,biWidth,'long');
fwrite(fid,biHeight,'long');
biPlanes=1;
fwrite(fid,biPlanes,'short');
biBitCount=8;
fwrite(fid,biBitCount,'short');
```

```
biCompression=0;
fwrite(fid,biCompression,'long');
biSizeImage=0;
fwrite(fid,biSizeImage,'long');
biXPels=0;
fwrite(fid,biXPels,'long');
biYPels=0;
fwrite(fid,biYPels,'long');
biClrUsed=0;
fwrite(fid,biClrUsed,'long');
biClrImportant=0;
fwrite(fid,biClrImportant,'long');
[m,n]=size(map);
if (m~=256)
  map=[map;zeros(256-m,3)];
elseif (m>256)
  error('Colormap exceeds 256 colors! Only 8-bit BMP file output is supported.');
end;
map=[fliplr(map*255),zeros(256,1)]';
fwrite(fid,map(:),'uchar');
x=rot90(x,-1)-1;
temp=[x;zeros(rest,biHeight)];
fwrite(fid,temp,'uchar');
fclose(fid);
```

A.1.3 imgread

Citirea fişierelor în format .img: imagini cu nivele de gri, pătrate, fără antet, fără tabelă de culoare, stocate în format de 8 biţi per pixel în gama 0-255, în ordinea normală de baleiaj pe linii a imaginii. Imaginea este citită într-o matrice şi la toate valorile este adăugată o unitate pentru a permite afişarea imaginii cu funcţia **image** a Matlab. Utilizarea funcţiei:

nume_matrice_imagine=imgread('nume_fisier_imagine', dimensiune_imagine);

unde *'nume_fisier_imagine'* trebuie dat între apostrofuri şi este numele DOS complet şi valid al unui fişier de imagine; dacă extensia nu este precizată, se presupune implicit extensia .img; *dimensiune_imagine* este dimensiunea imaginii (dacă nu se precizează, se consideră implicit valoarea 256). Cod de implementare:

```
function [X]=imgread(filename,dim);
if exist('filename')==0
```

```
  help imgread
  error('Trebuie specificat numele fisierului !');
end;
if (isstr(filename)~=1)
  help imgread
  error('Numele fisierului trebuie dat ca sir !');
end;
if (isempty(findstr(filename,'.'))==1)
  filename=[filename,'.img'];
end;
fid=fopen(filename,'rb','l');
if (fid==-1)
  error(['Eroare la deschiderea fisierului ',filename]);
end;
if exist('dim')==0
  dim=256;
end
X=fread(fid,[dim,dim],'uchar');
X=X'+1;
fclose(fid);
```

A.1.4 imgwrite

Scrierea fişierelor în format .img: imagini cu nivele de gri, pătrate, fără antet, fără tabelă de culoare, stocate în format de 8 biţi per pixel în gama 0-255, în ordinea normală de baleiaj pe linii a imaginii. Imaginea este citită dintr-o matrice şi din toate valorile este scăzută o unitate pentru a permite compatibilitatea cu formatul unsigned char al valorilor scrise în fişier (deci se presupune că valorile indecşilor sunt în intervalul 1-256). Utilizarea funcţiei:

imgwrite(nume_matrice_imagine, 'nume_fisier_imagine');

unde *nume_matrice_imagine* este numele unei variabile de tip matrice din spaţiul de lucru Matlab, iar ume_fisier_imagine" trebuie dat între apostrofuri şi este numele DOS complet şi valid al unui fişir de imagine; dacă extensia nu este precizată, se presupune implicit extensia .img. Cod de implementare:

```
if (nargin~=2)
  help imgwrite
  error('Necesita doua argumente');
end;
```

```
if (isstr(filename)~=1)
  help imgwrite
  error('Al doilea argument trebuie sa fie un sir');
end;
if (isempty(findstr(filename,'.'))==1)
  filename=[filename,'.img'];
end;
fid=fopen(filename,'wb','l');
if (fid==-1)
  error(['Eroare la deschiderea ',filename,' ']);
end;
X=(X-1)';
fwrite(fid,X(:),'uchar');
fclose(fid);
```

A.1.5 mfiread

Citirea fişierelor în format .mfi: imagini cu nivele de gri, fără tabelă de culoare, stocate în format de 8 biţi per pixel în gama 0-255, în ordinea normală de baleiaj pe linii a imaginii. Imaginea este citită într-o matrice şi la toate componentele este adăugată o unitate pentru a permite afişarea imaginii cu funcţia **image** a Matlab. Utilizarea funcţiei:

nume_matrice_imagine=mfiread('nume_fisier_imagine');

unde *'nume_fisier_imagine'* trebuie dat între apostrofuri şi este numele DOS complet şi valid al unui fişier de imagine; dacă extensia nu este precizată, se presupune implicit extensia .mfi. Cod de implementare:

```
function [X]=mfiread(filename);
if (nargin~=1)
  help mfiread
  error('Trebuie dat numele fisierului !');
end;
if (isstr(filename)~=1)
  help mfiread
  error('Numele fisierului trebuie dat ca sir !');
end;
if (isempty(findstr(filename,'.'))==1)
  filename=[filename,'.mfi'];
end;
fid=fopen(filename,'rb','l');
```

```
if (fid==-1)
  error(['Eroare la deschiderea fisierului ',filename]);
end;
bfType=fread(fid,2,'char');
setstr(bfType);
if (bfType~=['M';'F'])
  fclose(fid);
  error('Acesta nu este un fisier MFI !');
end;
biWidth=fread(fid,1,'ushort');
biHeight=fread(fid,1,'ushort');
biCompression=fread(fid,1,'uchar');
biReserved=fread(fid,1,'uchar');
  X=fread(fid,[biHeight,biWidth],'uchar');
  X=X'+1;
fclose(fid);
```

A.1.6 mfiwrite

Scrierea fişierelor în format .mfi: imagini cu nivele de gri, fără tabelă de culoare, stocate în format de 8 biţi per pixel în gama 0-255, în ordinea normală de baleiaj pe linii a imaginii. Imaginea este citită dintr-o matrice şi din toate valorile este scăzută o unitate pentru a permite compatibilitatea cu formatul unsigned char al valorilor scrise în fişier (deci se presupune că valorile indecşilor sunt în intervalul 1-256). Utilizarea funcţiei:

mfiwrite(nume_matrice_imagine, 'nume_fisier_imagine');

unde *nume_matrice_imagine* este numele unei variabile de tip matrice din spaţiul de lucru Matlab, iar *'nume_fisier_imagine'* trebuie dat între apostrofuri şi este numele DOS complet şi valid al unui fişier de imagine; dacă extensia nu este precizată, se presupune implicit extensia .mfi. Cod de implementare:

```
if (nargin~=2)
  help mfiwrite
  error('Necesita doua argumente');
end;
if (isstr(filename)~=1)
  help mfiwrite
  error('Al doilea argument trebuie sa fie un sir');
end;
if (isempty(findstr(filename,'.'))==1)
```

```
  filename=[filename,'.mfi'];
end;
fid=fopen(filename,'wb','l');
if (fid==-1)
  error(['Eroare la deschiderea ',filename,' pentru scriere']);
end;
fwrite(fid,['M';'F'],'char');
[biHeight,biWidth]=size(X);
fwrite(fid,biHeight,'short');
fwrite(fid,biWidth,'short');
biReserved=0;
fwrite(fid,biReserved,'short');
biComp=0;
fwrite(fid,biComp,'short');
X=(X-1)';
fwrite(fid,X(:),'uchar');
fclose(fid);
```

A.2 Calcul caracteristici statistice

A.2.1 cooc

Calculează matricea de coocurenţă pentru o regiune de formă dreptunghiulară şi un vector de translaţie fixat. Funcţia se apelează cu:

matrice_ coocurenta = cooc(regiune, vector);

unde *regiune* este o matrice ce conţine valorile pentru care se calculează matricea de coocurenţă iar *vector* este vectorul de translaţie între poziţiile spaţiale ale pixelilor ce formează perechile a căror probabilitate de apariţie este calculată. Vectorul de translaţie este o matrice linie cu două coloane, ce reprezintă translaţia pe orizontală şi respectiv translaţia pe verticală. În mod implicit vectorul de translaţie este considerat nul, şi atunci matricea de coocurenţă este o matrice diagonală, având pe diagonala principală histograma valorilor din regiunea *regiune*. Este necesar ca matricea *regiune* să conţină doar valori naturale supraunitare; valorile sunt truncheate şi limitate inferior la 1. Rezultatul *matrice_ coocurenta* calculat este o matrice pătrată de dimensiune egală cu valoarea maximă întâlnită în matricea *regiune*. Cod de implementare:

```
function out=cooc(in,t)
if (exist('t')==0)
```

```
  t=[0 0];
end
in=fix(in);
if (min(in(:))<1)
  p=find(in<1);
  in(p)=ones(size(p));
end
[h,w]=size(in);
m=max(in(:));
out=zeros(m);
for i=max(1,1-t(1)):min(h,h-t(1))
  for j=max(1,1-t(2)):min(w,w-t(2))
    out(in(i,j),in(i+t(1),j+t(2)))=out(in(i,j),in(i+t(1),j+t(2)))+1;
  end
end
```

A.2.2 histo

Calculează histograma (probabilitatea de apariţie a valorilor) dintr-o matrice. Funcţia se apelează prin:

[h, H] = histo(matrice, switch, maxim);

unde h este histograma valorilor (corespunzătoare funcţiei de densitate de probabilitate) iar H este histograma cumulativă (funcţia de repartiţie) pentru valorile din matricea de intrare *matrice*. Valorile din matricea de intrare trebuie să fie numere naturale cuprinse între 1 şi *maxim* (în mod implicit valoarea pentru *maxim* este 256). Histograma h şi histograma cumulativă H sunt vectori linie de *maxim* componente. Parametrul *switch* permite două moduri de calcul: prin baleierea matricii element cu element (*switch*=1) sau prin căutarea numărului de elemente ce au fiecare valoare posibilă din intervalul [1, *maxim]* (*switch*=0). Valoarea implicită a parametrului este 0, având în vedere că metoda de calcul corespunzătoare este mult mai rapidă. Cod de implementare:

```
function [h,hcum]=histo(in,sw,M)
if (exist('sw')==0)
  sw=0;
end
if (exist('M')==0)
  M=256;
end
[hh,w]=size(in);
```

```
h=zeros(1,M);
if (sw==0)
for i=1:M
  h(i)=prod(size(find(in==i)));
end
end
if (sw==1)
  for i=1:hh
    for j=1:w
      h(in(i,j))=h(in(i,j))+1;
    end
  end
end
h=h/(w*hh);
hcum=cumsum(h);
```

A.2.3 izoseg

Calculează matricea de izosegmente pentru o regiune dată, pe direcţia verticală. Funcţia se apelează prin:

matrice_izosegmente = izoseg(regiune);

unde *regiune* este o matrice ce trebuie să conţină doar valori naturale supraunitare. Ieşirea *matrice_izosegmente* este o matrice având un număr de linii egal cu valoarea maximă din matricea *regiune*, iar un număr de coloane egal cu numărul maxim de elemente consecutive având aceeaşi valoare aflate pe coloanele matricii *regiune*. Cod de implementare:

```
function out=izoseg(in)
in=fix(in);
[h,w]=size(in);
m=max(in(:));
out=zeros(m,h);
for i=1:m
  p=find(in==i);
  l=max(size(p));
  count=1;
  for j=2:l
    if (p(j)==(p(j-1)+1))&(fix((p(j)-1)/h)==fix((p(j-1)-1)/h));
      count=count+1;
    else
```

```
      out(i,count)=out(i,count)+1;
      count=1;
    end
  end
  out(i,count)=out(i,count)+1;
end
[i,j]=find(out);
out=out(1:max(i),1:max(j));
```

A.2.4 mas_cooc

Calculează parametri statistici de interes pe baza unei matrici de coocurenţă. Funcţia se apelează prin:

param = mas_ cooc(matrice_ coocurenta);

unde *matrice_ coocurenta* este o matrice de coocurenţă a unei regiuni iar rezultatul *param* este un vector linie cu 7 elemente, ce reprezintă măsurile statistice de omogenitate, contrast, entropie, corelaţie, omogenitate locală, directivitate şi uniformitate. Cod de implementare:

```
function out=mas_cooc(in)
[h,w]=size(in);
nc=sum(sum(in));
out(1)=sum(sum(in.^2))/(nc*nc);
for i=1:w
  p(i)=sum(diag(in,i))+sum(diag(in,-i));
end
i=0:(w-1);
out(2)=sum(p*(i.^2)')/(nc*(w-1)*(w-1));
out(5)=sum(p./(1+i.^2))/nc;
p=find(in~=0);
p=in(p);
out(3)=1-sum(sum(p.*log(p)))/(nc*log(nc));
p=sum(in');
mx=p*i';
sx=p*((i-mx).^2)';
p=sum(in);
my=p*i';
sy=p*((i-my).^2)';
out(4)=abs(sum(sum(((i-mx)'*(i-my)).*in)))/(nc*sqrt(sx*sy));
```

```
out(6)=trace(in)/nc;
out(7)=sum(diag(in).^2)/(nc*nc);
```

A.2.5 mas_four

Calculează distribuţia spectrală a energiei unei imagini. Funcţia se apelează prin:

rezultat = mas_four(spectru, nr);

unde *rezultat* este un vector cu *nr* componente (în mod implicit *nr* este 64) şi *spectru* este o matrice a cărei dimensiune minimă este de cel puţin de două ori mai mare decât *nr* şi care reprezintă un spectru Fourier centrat de energie al unei imagini. Obligatoriu, valorile din matricea *spectru* sunt pozitive. Cod de implementare:

```
function out=mas_four(in,nr)
if (exist('nr')==0)
  nr=64;
end
[h,w]=size(in);
mid_h=floor(h/2);
mid_w=floor(w/2);
if (nr>min([mid_h mid_w]))
  nr=min([mid_h mid_w]);
end
pash=floor(mid_h/nr);
pasw=floor(mid_w/nr);
out=zeros(nr-1,1);
for k=1:nr-1
  kh=k*pash;
  kw=k*pasw;
  out(k)=sum(sum(in(mid_h-kh:mid_h+kh,mid_w-kw:mid_w+kw)));
end
out=out./sum(sum(in));
```

A.2.6 mas_izo

Calculează parametri statistici de interes pe baza unei matrici de izosegmente. Funcţia se apelează prin:

param = mas_izo(matrice_izosegmente);

unde *matrice_izosegmente* este o matrice de izosegmente a unei regiuni iar rezultatul *param* este un vector linie cu 5 elemente, ce reprezintă măsurile statistice de proporţie a plajelor scurte, proporţie a plajelor lungi, heterogenitate a nivelelor de gri, heterogenitate a lungimilor plajelor şi procentaj al plajelor. Cod de implementare:

```
function rez=mas_izo(in)
[h,w]=size(in);
i=1:w;
hist=in*i';
slp=sum(sum(in));
rez(5)=slp/sum(hist);
i=i.^2;
t=[];
for l=1:h
  t=[t;i];
end
rez(1)=sum(sum(in./t))/slp;
rez(2)=sum(sum(in.*t))/slp;
rez(3)=sum(sum(in').^2)/slp;
rez(4)=sum(sum(in).^2)/slp;
```

A.2.7 moments

Calculează momente statistice, momente statistice invariante la translaţie şi scalare şi invarianţi de formă. Funcţia se apelează prin:

[mom, mom_t, mom_st, fi] = moments(regiune, nmax)

unde *regiune* este o matrice ce conţine valorile de interes iar *nmax* este ordinul până la care se calculează momentele (în mod implicit acesta este 3). Rezultatele sunt *mom*, *mom_t* şi *mom_st* sunt matrici pătrate de dimensiune *nmax*+1 ce conţin momentele statistice, momentele statistice centrate (invariante la translaţie) şi momentele statistice invariante la translaţie şi scalare. Dacă toate cele patru argumente de ieşire sunt recuperate în variabile, *fi* este un vector linie conţinând primii 6 invarianţi de formă (caz în care *nmax* trebuie să fie cel puţin 3). Cod de implementare:

```
function [mom,mom_t,mom_st,fi]=moments(in,n)
if (exist('n')==0)
  n=3;
end
[i,j,v]=find(in);
```

```
im=i-mean(i);
jm=j-mean(j);
aria=sum(v);
for p=0:n
  for q=0:n
    mom(p+1,q+1)=sum(v.*(i.^p).*(j.^q));
    mom_t(p+1,q+1)=sum(v.*(im.^p).*(jm.^q));
    mom_st(p+1,q+1)=sum(v.*(im.^p).*(jm.^q))/(aria^((p+q+2)/2));
  end
end
if (nargout==4)
  fi(1)=mom_st(3,1)+mom_st(1,3);
  fi(2)=(mom_st(3,1)-mom_st(1,3))^2+4*mom_st(2,2)^2;
  fi(3)=(mom_st(4,1)-3*mom_st(2,3))^2+(mom_st(1,4)-3*mom_st(3,2))^2;
  fi(4)=(mom_st(4,1)+mom_st(2,3))^2+(mom_st(1,4)+mom_st(3,2))^2;
  fi(5)=(mom_st(4,1)-3*mom_st(2,3))*(mom_st(4,1)+mom_st(2,3))*
((mom_st(4,1)+mom_st(2,3))^2-3*(mom_st(3,2)+mom_st(1,4))^2);
  fi(5)=fi(5)+(mom_st(1,4)-3*mom_st(3,2))*(mom_st(1,4)+mom_st(3,2))*
((mom_st(1,4)+mom_st(3,2))^2-3*(mom_st(2,3)+mom_st(4,1))^2);
  fi(6)=(mom_st(3,1)-mom_st(1,3))*((mom_st(4,1)+mom_st(2,3))^2-
(mom_st(3,2)+mom_st(1,4))^2);
  fi(6)=fi(6)+4*mom_st(2,2)*(mom_st(4,1)+mom_st(2,3))*(mom_st(1,4)+mom_st(3,2));
end
```

A.2.8 whisto

Calculează histograma ponderată prin laplacian a valorilor dintr-o regiune. Funcţia se apelează prin:

[h, H] = whisto(regiune, param, masca);

unde h este histograma ponderată şi H este histograma cumulativă ce îi este asociată (vectori de dimensiune 256). Histogramele sunt calculate pentru valorile din matricea regiune şi ponderate cu puterea *param* a expresiei 1+modulul laplacianului din fiecare element al matricii *regiune*. În mod implicit, valoarea puterii *param* este -1, corespunzând neluării în considerare a elementelor ce au vecini de valori diferite. Laplacianul est calculat implicit dacă variabila *masca* nu este dată la apelul funcţiei; altfel, ponderarea este calculată prin filtrarea liniară a matricii *regiune* cu nucelul de filtrare 3 x 3 *masca*. Cod de implementare:

```
function [h,hcum]=whisto(in,p,mask)
```

```
if (exist('p')==0)
  p=-1;
end
if (exist('mask')==0)
  mask=[0 -1 0;-1 4 -1;0 -1 0];
end
[hh,w]=size(in);
h=zeros(1,256);
in=[zeros(1,w+2);zeros(h,1) in zeros(h,1);zeros(1,w+2)];
for i=2:hh+1
  for j=2:w+1
    temp=in(i-1:i+1,j-1:j+1);
    la=(temp(:))'*mask(:);
    h(in(i,j))=h(in(i,j))+(1+abs(la))^p;
  end
end
h=h/(w*hh);
hcum=cumsum(h);
```

A.3 Segmentare regiuni

A.3.1 bhat

Determinarea automată a pragurilor de segmentare prin metoda Bhattacharyya. Mod de utilizare:

[prag, z, interval, param]=bhat(histo, limit);

unde *histo* este histograma imaginii pentru care se calculează pragurile de segmentare, iar *limit* este lungimea minimă a unui interval de descreştere a funcţiei de decizie şi are implicit valoarea 1. Rezultatele întoarse de funcţie sunt: *prag*: vectorul de praguri de segmentare, *z*: funcţia de decizie, *interval*: matricea de intervale pe care funcţia *z* este descrescătoare şi *param*: matricea de parametri statistici (medie, dispersie) pentru modurile determinate în histogramă. Cod de implementare:

```
function [prg,z,interval,param]=bhat(h,limit);
if (exist('limit')==0)
  limit=1;
end
no=max(size(h));
```

```
z=log(h(2:no)+eps)-log(h(1:no-1)+eps);
prg=[];
prg_start=[];
desc=0;
k=1;
count=0;
start=1;
for i=2:max(size(z))
  if (z(i)<z(i-1))
    if (desc==0)
      start=i-1;
    end
    desc=1;
    count=count+1;
  else
    if (desc==1)&(count>limit)
      prg(k)=i;
      prg_start(k)=start;
      k=k+1;
    end
  desc=0;
  count=0;
  end
end
if (desc==1)&(count>limit)
  prg(k)=i;
  prg_start(k)=start;
end
for i=1:k
  xdata=prg_start(i):prg(i);
  ydata=z(prg_start(i):prg(i));
  poly=polyfit(xdata,ydata,1);
  param(i,:)=[poly(2)/abs(poly(1)) sqrt(1/abs(poly(1)))];
end
interval=[prg_start' prg'];
```

A.3.2 cumthresh

Segmentare pe histograma cumulativă. Mod de utilizare:

[prag1, prag2]=cumthresh(histo, procent, tip);

unde *prag1* şi *prag2* sunt nivelele de gri ce încadrează cel mai bine pragul de segmentare ideal pentru care suprafaţa ocupată în imagine de obiectele având nivelul de gri mai mic (*tip=0*), respectiv mai mare (*tip=1*) decât pragul de segmentare. Implicit *tip=1*. Suprafaţa relativă ocupată în imagine de obiectele de interes este specificată de *procent* (valoare implicită *0.1*). Imaginea de segmentat este descrisă de histograma sa, *histo*.Cod de implementare:

```
function [t1,t2]=cumthres(h,proc,type)
if (exist('proc')==0)|(proc=[])
  proc=.1;
end
if(exists('type')==0)
  type=0;
end
H(i)=cumsum(h);
if (type==0)
  p=find(H<=proc);
  t1=p(max(size(p)));
  p=find(H>proc);
  t2=p(1);
end
if (type==1)
  p=find(H<=1-proc);
  t1=p(max(size(p)));
  p=find(H>1-proc);
  t2=p(1);
end
```

A.3.3 etichete

Etichetează o imagine binară prin propagarea etichetelor în regiuni conexe. Mod de utilizare:

rezultat=etichete(intrare, val_obiecte, val_fundal);

unde *rezultat* este imaginea etichetată cu etichete numerice incrementate cu 1. Fundalul are eticheta 1, etichetele obiectelor încep de la valoarea 2. Imaginea binară iniţială este *intrare*, caracterizată de valorile *val_obiecte* pentru pixelii de obiect (valoare implicită 2) şi *val_fundal* pentru pixelii de fundal (valoare implicită 1). Etichetarea se bazează pe creşterea regiunilor. Cod de implementare:

```
function [out]=etichete(in,val_obj,val_bk)
if (exist('val_bk')==0)
  val_bk=1;
end
if (exist('val_obj')==0)
  val_obj=2;
end
if (val_bk==val_obj)
  error('Fundalul si obiectele nu pot avea aceeasi valoare !')
end
out=ones(size(in));
[h,w]=size(in);
val=2;
p=find(in==val_obj);
while (p~=[])
  seed_i=rem(p(1),h);
  seed_j=fix(p(1)/h)+1;
  sir=grow_reg(in,seed_i,seed_j,0);
  p=(sir(:,2)-1)*h+sir(:,1);
  out(p)=val*ones(size(p));
  in(p)=val_bk*ones(size(p));
  val=val+1;
  p=find(in==val_obj);
end
```

A.3.4 etic1

Etichetarea unei imagini binare prin metoda tabelului de corespondenţe între etichete. Mod de utilizare:

rezultat=etic1(intrare);

unde *rezultat* este imaginea etichetată cu etichete numerice incrementate cu 1. Fundalul are eticheta 1, etichetele obiectelor încep de la valoarea 2. Imaginea binară iniţială este *intrare*, caracterizată de valorile *2* pentru pixelii de obiect şi *1* pentru pixelii de fundal. Cod de implementare:

```
function [out,tab]=etic1(in)
vv=[-1 -1;-1 0;-1 1;0 -1];
[lv,cv]=size(vv);
[h,w]=size(in);
```

```
out=ones(h,w);
h1=h-1;
w1=w-1;
tab(1)=1;
k=1;
for i=2:h1
  for j=2:w1
    if(in(i,j)==2)
      for v=1:lv
        pp(v)=out(vv(v,1)+i,vv(v,2)+j);
      end
      ma=max(pp);
      m=min(pp(find(pp>1)));
      if ma==1
        k=k+1;
        tab(k)=k;
        out(i,j)=k;
      else
        out(i,j)=m;
        for p=1:lv
          pr=pp(p);
          if(pr~=1)
            while(tab(pr)~=m)
              temp=tab(pr);
              tab(pr)=m;
              pr=temp;
            end
          end
        end
      end
    end
  end
end
for j=1:k
  et=j;
  while (tab(et)~=et)
    et=tab(et);
  tab(j)=et;
  end
end
out=reshape(tab(out),h,w);
```

A.3.5 gausmixt

Generarea unei histograme ce corespunde unei mixturi de moduri gaussiene. Mod de utilizare:

histo=gausmixt(param, procente, interval);

unde *histo* este histograma (distribuţia) generată, formată din moduri gaussiene descrise de parametrii medie şi dispersie (pe liniile matricii *param*), mixate în proporţiile descrise de vectorul *procente*. Punctele de calcul ale funcţiilor generate sunt numerele întregi din intervalul a cărui limite de jos şi sus sunt specificate în vectorul *interval* (în mod implicit *interval=[0 255]*). Cod de implementare:

```
function [fct]=gausmixt(params,p,range)
if (exist('params')==0)
  error('Trebuiesc specificati parametrii modurilor')
end
[no,w]=size(params);
if (w~=2)
  error('Parametrii modurilor nu sunt specificati corect')
end
if (exist('range')==0)
  range=[0 255];
end
x=range(1):range(2);
if (exist('p')==0)
  p=ones(1,no)/no;
end
if (p==[])
  p=ones(1,no)/no;
end
p=p/sum(p);
mu=params(:,1);
sigma=params(:,2);
for i=1:no
  P(i,:)=(p(i)/(sigma(i)*sqrt(2*pi)))*exp(-((x-mu(i))/sigma(i)).^2);
end
fct=sum(P)/sum(sum(P));
```

A.3.6 grow_reg

Creşterea unei regiuni uniforme. Mod de utilizare:

[pozitii, valori]=grow_reg(imag, x_start, y_start, prag, numar);

unde *pozitii* este vectorul indicilor pixelilor din imaginea iniţială *imag* ce formează regiunea, iar *valori* este vectorul valorilor (nivelelor de gri) a pixelilor selectaţi în regiune. Sămânţa (germenele de creştere al regiunii) este pixelul din imagine având coodonatele *x_start* şi *y_start*. Regiunea este crescută având o toleranţă *prag* pentru diferenţa de nivele de gri a pixelilor din regiune (implicit *prag=15*) şi având cel mult *numar* pixeli (implicit *numar= 200*). Cod de implementare:

```
function [sir,sir_val]=grow_reg(img,i,j,prag,nrmax)
if (exist('prag')==0)
  prag=15;
end
if (prag==[])
  prag=15;
end
if (exist('nrmax')==0)
  nrmax=200;
end
[dimx,dimy]=size(img);
seed=img(i,j);
map=zeros(size(img));
map(i,j)=1;
sir=zeros(nrmax,2);
sir_val=zeros(nrmax,1);
sir(1,1)=i;
sir(1,2)=j;
ptr=2;
sir_val(1)=seed;
ptr1=1;
while (ptr1<ptr) & (ptr < nrmax),
  i=sir(ptr1,1);
  j=sir(ptr1,2);
  for m=-1:1,
    for n=-1:1,
      if i+m >=1 & i+m <= dimx & j+n >=1 & j+n <=dimy ,
        if ~map(i+m,j+n),
        if abs(img(i+m,j+n)-seed) <= prag,
            sir(ptr,:)=[i+m,j+n];
```

```
            sir_val(ptr)=img(i+m,j+n);
            map(i+m,j+n)=1;
            ptr=ptr+1;
          end
        end
      end
    end
  end
  ptr1=ptr1+1;
end
b=max(size(find(sir_val)));
sir=sir(1:b,:);
sir_val=sir_val(1:b);
```

A.3.7 nakagawa

Verificarea separabilităţii modurilor alăturate ale unei distribuţii. Mod de utilizare:

lista=nakagawa(histograma, param);

unde *lista* este un tabel în care sunt trecuţi indicii perechilor de moduri vecine ce nu satisfac una dintre condiţiile de separaţie, precum şi condiţia ce nu este îndeplinită. Mixtura de moduri este specifcată de distribuţia (histograma) *histograma* şi lista de parametri *param*. Lista de parametri este o matrice cu două coloane ce conţine pentru fiecare mod media şi dispersia. Cod de implementare:

```
function [bad_list]=nakagawa(h,params)
if (exist('params')==0)
  error('Parametrii modurilor lipsesc')
end
[no,w]=size(params);
if (w~=2)
  error('Parametrii modurilor nu sunt specificati corect')
end
if (no==1)
  error('Problema nu are sens pentru un singur mod')
end
for i=2:no
  dif=params(i,1)-params(i-1,1);
  rap=params(i,2)/params(i-1,2);
  rap2=min(h(params(i,1):params(i-1,1)))/min([params(i,1) params(i-1,1)]);
```

```
  if (dif<=4)
    bad_list=[bad_list;num2str(i) ' ' num2str(i-1)
' medii prea apropiate'];
  end
  if (rap>10)|(rap<.1)
    bad_list=[bad_list;num2str(i) ' ' num2str(i-1)
' dispersii prea diferite'];
  end
  if (rap2>.8)
    bad_list=[bad_list;num2str(i) ' ' num2str(i-1)
' separatie neclara'];
  end
end
```

A.3.8 thresh

Segmentare cu praguri pe histogramă (thresholding). Mod de utilizare:

rezultat=thresh(init, prag, switch, prag_min, prag_max);

unde imaginea de ieşire *rezultat* este obţinută fie ca o imagine de etichete (*switch=1*, valoare implicită), fie ca o aproximare prin nivelul de gri mediu (*switch=2*) a imaginii iniţiale *init*, prin segmentarea după histogramă a imaginii iniţiale *init* după vectorul de praguri *prag*. Dacă vectorul *prag* nu se specifică, se consideră implicit o segmentare în două clase, cu pragul de 128. Pragurile de segmentare minim şi respectiv maxim sunt *prag_min* şi *prag_max* şi au implicit valorile 1 şi 257 (acestea se adaugă automat la vectorul de praguri). Etichetele sunt alocate începând cu 1 şi incremetate cu 1; etichetele de valoare mică sunt alocate obiectelor cu nivele de gri mai mici. Cod de implementare:

```
function out=thresh(in,prag,t,prag_min,prag_max)
if (exist('prag')==0)|(prag==[])
  prag=[128];
end
if (exist('t')==0)
  t=1;
end
if (exist('prag_min')==0)
  prag_min=1;
end
if (exist('prag_max')==0)
  prag_max=257;
```

```
end
out=ones(size(in));
prag=[prag_min prag prag_max];
np=max(size(prag));
for i=1:np-1
  p=find((in>=prag(i))&(in<prag(i+1)));
  if (t==1)
    val=i;
  else
    temp=in(p);
    val=round(mean(temp(:)));
  end
  out(p)=val*ones(size(p));
end
```

A.4 Extragere contururi

A.4.1 compas

Operator compas pentru extragerea contururilor. Funcţia se apelează prin:

harta_intensitati=compas(imagine_init, masca_init);

unde rezultatul *harta_intensitati* este o imagine cu nivele de gri (rescalate între 1 şi 256), de aceleaşi dimensiuni cu imaginea de prelucrat *imagine_init*, ce reprezintă harta de intensităţi de tranziţie. Nucleul de filtrare *masca_init* este o matrice 3 x 3, având suma valorilor nulă, ce reprezintă nucleul derivativ pe direcţie orizontală. Cod de implementare:

```
function [cont]=compas(in,mask);
[h,w]=size(in);
H1=mask(:);
H2=rot90(mask,1);
H2=H2(:);
H3=rot90(mask,2);
H3=H3(:);
H4=rot90(mask,3);
H4=H4(:);
mask=[H1(2) H1(1) H1(4);H1(3) H1(5) H1(7);H1(6) H1(9) H1(8)];
H5=mask(:);
H6=rot90(mask,1);
```

```
H6=H6(:);
H7=rot90(mask,2);
H7=H7(:);
H8=rot90(mask,3);
H8=H8(:);
H=[H1';H2';H3';H4';H5';H6';H7';H8'];
cont=zeros(size(in));
for i=2:h-1
  for j=2:w-1
    t=in(i-1:i+1,j-1:j+1);
    cont(i,j)=max(abs(H*t(:)));
  end
end
cont=remap(cont);
```

A.4.2 gradient

Operator de gradient pentru extragerea contururilor. Funcţia se apelează prin:

harta_intensitati=gradient(imagine_init, masca_init);

unde rezultatul *harta_intensitati* este o imagine cu nivele de gri (rescalate între 1 şi 256), de aceleaşi dimensiuni cu imaginea de prelucrat *imagine_init*, ce reprezintă harta de intensităţi de tranziţie. Nucleul de filtrare *masca_init* este o matrice 3 x 3, având suma valorilor nulă, ce reprezintă nucleul derivativ pe direcţie orizontală. Cod de implementare:

```
function [cont]=gradient(in,mask);
[h,w]=size(in);
Hx=mask(:);
Hy=rot90(mask);
Hy=Hy(:);
cont=zeros(size(in));
for i=2:h-1
  for j=2:w-1
    t=in(i-1:i+1,j-1:j+1);
    cont(i,j)=abs(Hx'*t(:))+abs(Hy'*t(:));
  end
end
cont=remap(cont);
```

A.4.3 fourdesc

Extrage funcţia de contur a unei forme pentru calculul descriptorilor Fourier ai acesteia. Mod de utilizare:

[u, imcont] = fourdesc(im);

unde *im* este imaginea binară de intrare ce conţine un unic obiect ai cărui pixeli au valoarea 1, *u* este vectorul complex ce conţine coordonatele punctelor de contur în ordinea parcurgerii acestuia, iar *imcont* este imaginea binară ce conţine conturul. Cod de implementare:

```
function [z,b]=fourdesc(a)
[l,c]=size(a);
temp=a;
for i=2:l-1,
  for j=2:c-1,
    temp(i,j)=min(min(a(i-1:i+1,j-1:j+1)));
  end
end
a=a-temp;
for i=2:l-1,
  for j=2:c-1,
    if a(i,j),
      if (a(i,j-1)&a(i-1,j)&(~a(i+1,j+1))) | ...
         (a(i-1,j)&a(i,j+1)&(~a(i+1,j-1))) | ...
         (a(i,j+1)&a(i+1,j)&(~a(i-1,j-1))) | ...
         (a(i+1,j)&a(i,j-1)&(~a(i-1,j+1))),
         a(i,j)=0;
      end
    end
  end
end
for i=2:l-1,
  for j=2:c-1,
    if a(i,j),
      B=a(i-1:i+1,j-1:j+1);
      if (sum(abs(conv(B([1 2 3 6 9 8 7 4]),[-1 1]))) == 2)
        a(i,j)=0;
      end
    end
  end
end
```

```
b=a;
L=sum(a(:));
z=zeros(1,L);
temp=find(a);
i=floor(temp(1)/lin)+1;
j=temp(1)-(i-1)*lin;
z(1)=i+sqrt(-1)*j;
a(i,j)=0;
i=1;
while(1)
  c1=real(z(i));
  c2=imag(z(i));
  test=0;
  for k=-1:1,
    for l=-1:1,
      if a(c1+k,c2+l)==1,
        test=1;
        break;
      end
    end
    if test==1,
      break;
    end
  end
  if test==0
    break;
  end
  i=i+1;
  z(i)=c1+k+sqrt(-1)*(c2+l);
  a(c1+k,c2+l)=0;
end
```

A.5 Prelucrări generale

A.5.1 graymas

Calculează măsuri cantitative de calitate a unei imagini. Funcţia se apelează prin:

[snr, psnr, mae] = graymas(orig, distort, nr);

unde *orig* şi *distort* sunt matrici având aceleaşi dimensiuni, reprezentând imaginea originală (fără distorsiuni) şi respectiv imaginea degradată, a cărei calitate se doreşte măsurată. Parametrul *nr* (valoare implicită nulă) specifică numărul de linii şi de coloane de pe marginile imaginii ce sunt excluse de la comparaţie. Rezultatele sunt raportul semnal zgomot *snr* (în dB), raportul semnal de vârf zgomot *psnr* (în dB) şi eroarea absolută medie *mae*. Cod de implementare:

```
function [snr,psnr,mae]=graymas(x,y,win)
if (exist('win')==0)
  win=0;
end
[h,w]=size(x);
x=x(1+win:h-win,1+win:w-win);
y=y(1+win:h-win,1+win:w-win);
[h,w]=size(x);
dif=abs(x-y);
s=sum(dif(:).^2);
mae=mean(dif(:));
sorig=sum(sum(x.^2));
snr=10*log10(sorig/s);
psnr=10*log10((h-1)*(w-1)*(max(max(x))^2)/s);
```

A.5.2 lfilter

Calculează filtrarea neliniară de ordine printr-un L-filtru a unei imagini. Funcţia se apelează prin:

imagine_rezultat=lfilter(imagine_init, ponderi);

unde matricea *imagine_rezultat* conţine rezultatul filtrării matricii *imagine_init* prin L-filtrul definit de vectorul de coeficienţi *ponderi*. Coeficienţii sunt specificaţi ca un vector linie cu 9 componente, prima componentă corespunzând statisticii de ordin minim şi ultima componentă corespunzând statisticii de ordin maxim din fereastra de filtrare de 3 x 3, centrată. În mod implicit, filtrarea este de mediere aritmetică. Filtrele uzuale de ordonare ce se pot obţine sunt: filtrul median *ponderi=[0 0 0 0 1 0 0 0 0]*, erodarea morfologică *ponderi = [1 0 0 0 0 0 0 0 0]*, dilatarea morfologică *ponderi=[0 0 0 0 0 0 0 0 1]*, gradientul morfologic ponderi=*[-1 0 0 0 0 0 0 0 1]*, filtrul de mijloc *ponderi=[0.5 0 0 0 0 0 0 0 0.5]*. Cod de implementare:

```
function [out]=lfilter(in,ww)
if (exist('ww')==0)
```

```
  ww=ones(1,9)/9;
end
[h,w]=size(in);
out=in;
for i=2:h-1
  for j=2:w-1
    temp=in(i-1:i+1,j-1:j+1);
    temp=sort(temp(:));
    out(i,j)=fix(ww*temp);
  end
end
```

A.5.3 linfilt

Filtrarea liniară a unei imagini. Funcţia se apelează prin:

imagine_rezultat=linfilt(imagine_init, nucleu, switch);

unde matricea *imagine_rezultat* conţine rezultatul filtrării matricii *imagine_init* prin filtrul liniar definit de matricea de coeficienţi *nucleu*. Dimensiunile matricii *nucleu* trebuie să fie impare. Suma valorilor din *nucleu* trebuie să fie *1* pentru o filtrare de netezire sau *0* pentru o filtrare derivativă. Parametrul *switch* controlează modul de realizare a filtrării: în domeniul spaţial prin *switch=0* sau în domeniul de frecvenţă (alegere implicită) prin *switch=1* (această din urmă variantă este mult mai rapidă). Cod de implementare:

```
function [out]=linfilt(in,mask,sw)
if (exist('sw')==0)
  sw=1;
end
if nargin < 2,
  help linfilt
  error('Nu ati furnizat numarul necesar de parametri!');
end
[li ci]=size(in);
[ln cn]=size(mask);
if ~(ln-2*floor(ln/2) | cn-2*floor(cn/2)),
  help linfilt
  error('Dimensiunile nucleului de filtrare trebuie sa fie numere impare!');
end
tol=10^(-5);
sum_coef=sum(mask(:));
```

```
if round(sum_coef-1)>tol & round(sum_coef)>tol,
  error('Nucleul de filtrare nu corespunde');
end
ln=floor(ln/2);
cn=floor(cn/2);
if (sw==1)
  out=round(real(ifft2(fft2(in).*fft2(mask,li,ci))));
  out=max(1,min(256,out));
  out=[out out;out out];
  out=out(1+ln:li+ln,1+cn:ci+cn);
else
  in=[zeros(ln,ci+2*cn);zeros(li,cn) in zeros(li,cn);zeros(ln,ci+2*cn)];
  for i=1+ln:li+ln,
    for j=1+cn:ci+cn,
      a=in(i-ln:i+ln,j-cn:j+cn);
      out(i-ln,j-cn)=abs(round(a(:)'*mask(:)));
    end
  end
  out=max(1,min(256,out));
end
```

A.5.4 remap

Funcţie de scalare a valorilor unei matrici. Mod de utilizare:

rezultat=remap(intrare, val_jos, val_sus);

unde matricea de ieşire *rezultat* are aceleaşi dimensiuni cu matricea de iniţială *intrare*. Gama de valori a componentelor din matricea de ieşire este cuprinsă între valoarea minimă *val_jos* (valoare implicită 1) şi valoarea maximă *val_sus* (valoare implicită 256). Componentele matricii de ieşire sunt truncheate la numere întregi. Dacă matricea iniţială are toate componentele identice, matricea de ieşire va avea toate componentele identice, de valoare egală cu media aritmetică a *val_jos* şi *val_sus*. Cod de implementare:

```
function [X]=remap(IN,lo,hi)
if (exist('IN')==0)
  help remap
  error('Input data missing !')
end
if (exist('lo')==0)
  lo=1;
```

```
end
if (exist('hi')==0)
  hi=256;
end
m=min(min(IN));
M=max(max(IN));
if (M~=m)
  a=(hi-lo)/(M-m);
  b=lo-a*m;
  X=fix(a*IN+b);
else
  X=((hi+lo)/2)*ones(size(X));
end
```

A.5.5 rot45

Rotaţia unei matrici cu 45°. Mod de utilizare:

rezultat=rot45(intrare);

unde *rezultat* este matricea iniţială *intrare* rotită în sens orar cu 45°. Matricea de intrare trebuie să fie pătrată. Cod de implementare:

```
function out=rot45(in)
[h,w]=size(in);
dim=min(h,w);
in=in(1:dim,1:dim);
out=[];
for i=-(dim-1):(dim-1)
  temp=[diag(in,i);zeros(abs(i),1)];
  out=[out temp];
end
```

Anexa B

Ghid de utilizare a programului SAIN

Programul SAIN - Sistem de analiză a imaginilor binare (Système d'Analyse d'Images Normalisées) a fost conceput ca un instrument de test şi experimentare, atât în ceea ce priveşte prelucrarea imaginilor binare şi extragerea de parametri reprezentativi ai acestora, cât şi în domeniul realizării de programe educaţionale (structurare, implementare, ...). Ideea de plecare a fost realizarea unui program interactiv şi convivial, cu posibilităţi de integrare în sisteme de calcul de putere redusă (de tip calculator personal ieftin); pentru aceasta a fost aleasă o programare pentru integrarea sub Windows. Stilul de programare ales este stilul de programare orientat obiect OOP (Object Oriented Programming) într-un limbaj ce permite numeroase facilităţi, şi anume Borland C++ 3.1.

Se consideră ca date de intrare pentru program imagini binare, reprezentate sub forma unor matrici de dimensiune variabilă (dar limitată superior la o valoare maximă de 600 x 600 pixeli), codate cu convenţia de asociere a unei valori nule pentru fondul imaginii şi a unei valori nenule (1) pentru pixelii obiectelor. Imaginile se pot afla stocate în fişiere cu anumite formate (BITMAP sau MFI), sau pot fi create chiar în interiorul programului prin facilităţi rudimentare de editare de forme geometrice simple (dreptunghiuri şi elipse).

Programul permite exportul principalelor tipuri de imagini prelucrate sub formă de fişiere BITMAP: imaginile binare propriu-zise rezultate în urma diverselor prelucrări, imaginile diferitelor hărţi asociate unei imagini binare (hărţi de distanţe Danielsson, hărţi Voronoi) sau imaginile grafice ale diverselor semnături ale formelor. Semnătura unei forme poate fi reţinută sub forma specială a unui fişier semnătură, posibil de utilizat pentru prelucrări ulterioare specifice (extrageri de parametri, aproximaţii ...).

Operaţiile implementate urmăresc să pună la dispoziţia utilizatorului o gamă cât mai completă de posibilităţi de operare; se întâlnesc implementate în programul SAIN ope-

raţiile binare şi morfologice, operaţiile de extragere a hărţilor de distanţe Danielsson şi Voronoi, operaţiile de calcul a unor puncte caracteristice pentru o formă (centrul de greutate, centrele cercurilor înscris şi circumscris, punctele de contur, punctele skeletonului). Alte operaţii complementare (cum ar fi etichetare unei imagini sau aproximarea poligonal a unui contur) sunt de asemenea disponibile. De asemenea, se permite extragerea unui mare număr de parametrii de circularitate şi convexitate, momente statistice şi parametri de încadrare. La toate acestea se adaugă facilităţile mediului de operare Windows, cu posibilitatea de rulare a aplicaţiilor multiple, gestiunea documentelor multi fereastră, aspectul "standardizat" al interfeţei utilizator şi al comenzilor.

Fereastra principală a aplicaţiei pune la dispoziţia utilizatorului o bară de meniu cu meniuri derulante; principalele meniuri (şi submeniurilor lor) sunt :

Image : operaţii de intrare / ieşire la nivel fişier

Nouvelle : crearea unei noi imagini şi a unei noi ferestre asociate

Ouvrir : încărcarea unei imagini stocate în format MFI

Enregistrer : salvarea imaginii curente în formatul MFI, sub numele curent

Enregistrer sous : salvarea imaginii curente în format MFI sub un alt nume

Enregistrer signature : salvarea semnăturii în format SIG

Ouvrir Bitmap : încărcarea unei imagini stocate în format BITMAP

Enregistrer Bitmap : salvarea unei imagini (binare, hărţi, semntură) în format BITMAP

Quitter : părăsirea aplicaţiei

Operations : operaţii de prelucrare a imaginilor binare

Binaires : operaţiile binare AND, NOT, OR, XOR, DIFF

Morphologiques : operaţiile morfologice de dilatare, erodiune, închidere, deschidere

Extractions contour : extragerea de contur vertical, orizontal, complet, codare Freeman

Distances : calculul distanţelor între forme (Hausdorff, diferenţă simetrică)

Cartes distances : calculul hărţilor de distanţe (Danielsson, Voronoi)

Labelisation : etichetarea unei imagini

Polygonaliser : extragerea unui poligon de aproximare pentru o formă

Paramètres : operaţii de extragere a parametrilor

Centre de gravité : calculul centrului de greutate

Moments statistiques : calculul invarianţilor statistici ai formelor

Convexités : calculul parametrilor de convexitate ai unei forme

Circularités : calculul parametrilor de circularitate ai unei forme

Morphologiques : calculul skeletonului morfologic şi a centrelor cercurilor înscris şi circumscris

Signatures : calculul semnăturii unei forme

Edition : editarea de imagini de test conţinând corpuri geometrice simple

Rectangle : desenarea unui dreptunghi (laturi paralele cu axele de coordonate)

Ellipse : desenarea unei elipse (axe paralele cu axele de coordonate)

Effacer : ştergerea conţinutului imaginii curente

Options : configurarea programului

Enregistrement : configurarea salvării imaginilor

Distance dausdorff : configurarea modului de calcul al distanţei Hausdorff

Approximations : parametrii de aproximare poligonală şi ai semnăturii

Labelisation : configurarea etichetării

Cartes : configurarea extragerii hărţilor de distanţe

Affichage : configurarea modului de afişare a informaţiilor asociate unei imagini

Enregistrer options : înregistrarea opţiunilor

Charger options : încrcarea unui fişier de configurare

Fenêtres : meniu standard Windows pentru configurarea spaţiului de lucru

Cascade : aranjarea ferstrelor deschise în cascadă

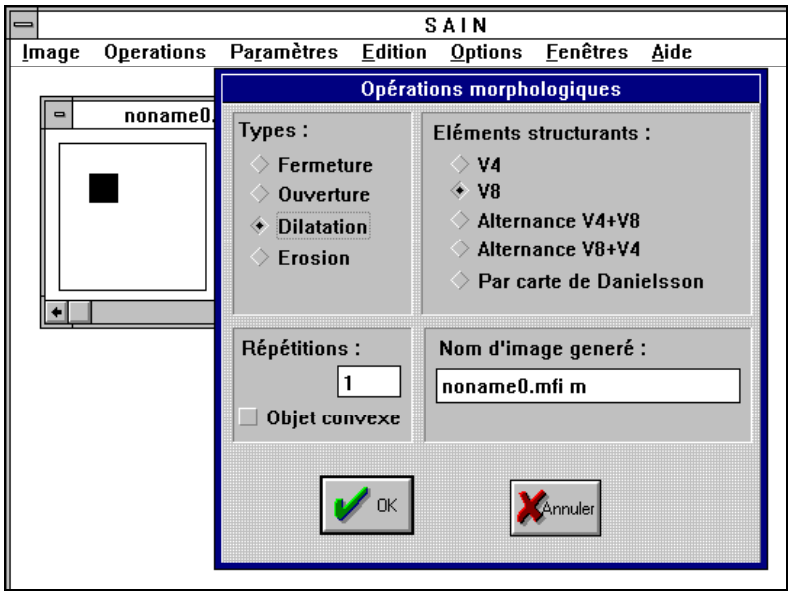

Fig. B.1: Fereastra principală a aplicaţiei SAIN, continând bara meniului principal; asupra imaginii conţinute într-o fereastră de tip imagine se alege efectuarea unei operaţii de tip morfologic, prin selecţia parametrilor şi comenzilor într-o fereastră de dialog.

Mosaique : aranjarea ferstrelor deschise în mod mozaic (pe tot ecranul)

Arranger icônes : aranjarea iconurilor ferestrelor minimizate la baza ecranului

Tout fermer : închiderea tuturor ferestrelor

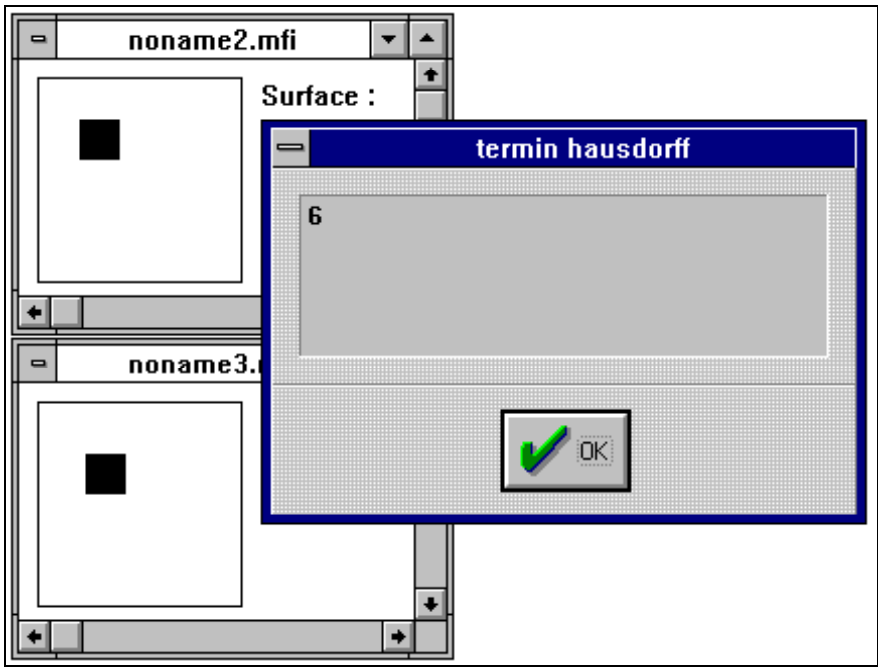

Fig. B.2: Rezultatul calculului distanţei Hausdorff între două forme identice, dar decalate poziţional.

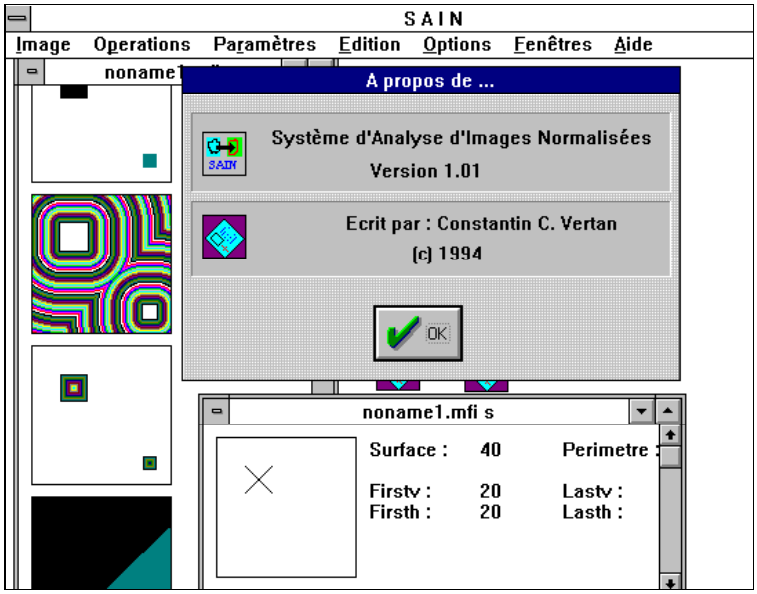

Fig. B.3: O fereastră de tip imagine conţine o imagine binară etichetată (diferenţele dintre obiecte sunt marcate prin culori) şi hărţile de distanţe ce îi sunt asociate (fereastra "noname1.mfi") ; fereastra de tip imagine "noname1.mfi s" conţine skeletonul unei forme pătrate. Fereastra de dialog "A propos de..." aduce informaţii asupra programului.

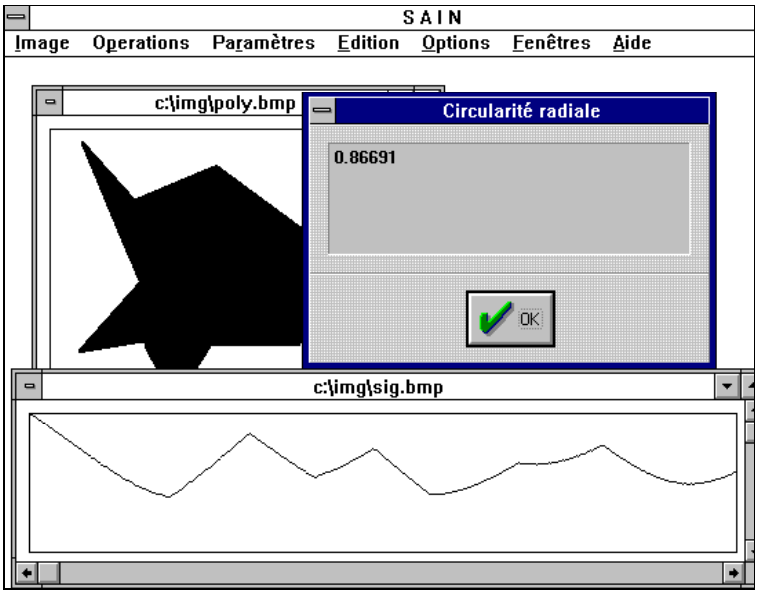

Fig. B.4: În două ferestre distincte se observă o formă iniţială ("poly.bmp") şi semnătura sa, extrasă în funcţie de abscisa curbilinie, cu un număr de 512 eşantioane; într-o fereastră de dialog apare rezultatul calculului parametrului de circularitate al formei ca raport al valorilor extreme ale semnăturii.

Anexa C

Lista imaginilor disponibile

C.1 Imagini binare pentru etichetare

binar1.bmp, binar2.bmp, binar3.bmp, binar4.bmp, binar5.bmp, obiect2.bmp, obiect4.bmp

C.2 Imagini binare pentru caracterizarea formelor

f0_0.mfi, f0_1.mfi, f0_2.mfi, f0_3.mfi, f0_4.mfi, f0_5.mfi, f0_6.mfi, f0_7.mfi

f1_0.mfi, f1_1.mfi, f1_2.mfi, f1_3.mfi, f1_4.mfi, f1_5.mfi, f1_6.mfi, f1_7.mfi

f2_0.mfi, f2_1.mfi, f2_2.mfi, f2_3.mfi, f2_4.mfi, f2_5.mfi, f2_6.mfi, f2_7.mfi

f3_0.mfi, f3_1.mfi, f3_2.mfi, f3_3.mfi, f3_4.mfi, f3_5.mfi, f3_6.mfi, f3_7.mfi

f4_0.mfi, f4_1.mfi, f4_2.mfi, f4_3.mfi, f4_4.mfi, f4_5.mfi, f4_6.mfi, f4_7.mfi

f5_0.mfi, f5_1.mfi, f5_2.mfi, f5_3.mfi, f5_4.mfi, f5_5.mfi, f5_6.mfi, f5_7.mfi

f6_0.mfi, f6_1.mfi, f6_2.mfi, f6_3.mfi, f6_4.mfi, f6_5.mfi, f6_6.mfi, f6_7.mfi

f7_0.mfi, f7_1.mfi, f7_2.mfi, f7_3.mfi, f7_4.mfi, f7_5.mfi, f7_6.mfi, f7_7.mfi

f8_0.mfi, f8_1.mfi, f8_2.mfi, f8_3.mfi, f8_4.mfi, f8_5.mfi, f8_6.mfi, f8_7.mfi

f9_0.mfi, f9_1.mfi, f9_2.mfi, f9_3.mfi, f9_4.mfi, f9_5.mfi, f9_6.mfi, f9_7.mfi

Sunt reprezentate 10 forme (de la f0 la f9), fiecare în 8 variante (de la _0 la _7) translatate, scalate, rotite şi cu mici variaţii. Fişierele există cu aceleaşi nume (şi extensia .bmp) şi în format BITMAP.

C.3 Imagini intrinsec binare

D128.bmp, M128.bmp, T128.bmp - forma corectă a documentelor

Draw128.bmp, Mail128.bmp, Text128.bmp - imagini cu nivele de gri (8 biţi pe pixel), documente scanate

C.4 Imagini de sinteză cu nivele de gri

tst1_128.img, tst2_128.img, tst3_128.img, tst4_128.img, tst5_128.img.

C.5 Texturi

t0_256.img, t1_256.img, t2_256.img, t3_256.img, t4_256.img, t5_256.img, t6_256.img, t7_256.img, t8_256.img, t9_256.img

C.6 Imagini naturale cu nivele de gri

cact_128.img, came_128.img, dune_128.img, face_128.img, flow_128.img, lena_128.img, pepp_128.img

Bibliografie

[1] Castleman, K. R.: *Digital Image Processing*, Prentice Hall, Englewood Cliffs, NJ, 1996

[2] Cocquerez, J. P., Philipp, S. (coord.): *Analyse d'images: filtrage et segmentation*, Masson, Paris, 1995

[3] Dougherty E. R., Giardina, C. R.: *Image Processing - Continous to Discrete*, vol. 1, *Geometric, Transform and Statistical Methods*, Prentice Hall Inc., Englewood Cliffs, 1987

[4] Gonzales, R. C., Woods, R. E.: *Digital Image Processing*, Addison Wesley, Reading MA, 1992

[5] Haralick, R. M., Shapiro, L. G.: *"Glossary of Computer Vision Terms"*, în Pattern Recognition, vol. 24, no. 1, pag. 69-93, 1991

[6] Jähne, B.: *Practical Handbook on Image Processing for Scientific Applications*, CRC Press, 1997

[7] Jain, A. K.: *Fundamentals of Digital Image Processing*, Prentice Hall, Englewood Cliffs NJ, 1989

[8] Pitas, I., Venetsanopoulos, A. N.: *Nonlinear Digital Filters – Principles and Applications*, Kluwer Academic Publ., Norwell MA, 1990

[9] Serra, J.: *Image Analysis and Mathematical Morphology*, Academic Press, London, 1982

[10] Spătaru, A.: *Teoria Transmisiunii Informaţiei*, Ed. Didactică şi Pedagogică, Bucureşti, 1984

[11] C. Vertan: "Prelucrarea şi Analiza Imaginilor", Ed. Printech, Bucureşti, 1999

[12] Wahl, F. M.: *Digital Image Signal Processing*, Artech House, Boston, 1987