6. ANALIZA ŞI PRELUCRAREA IMAGINILOR

Prelucrarea imaginilor înseamnă modificarea unei imagini în vederea facilitării extragerii informației, fie de către om, fie de un echipament. Din perspectivă umană se dorește ca în imagine să existe un contrast convenabil, imaginea să fie clară și să se poată distinge detaliile. Din punctul de vedere al echipamentelor, este indicat ca imaginea să fie cât mai simplă și fără paraziți.

Operațiile de prelucrare a imaginii, care să asigure o mai bună vizualizare de către om, pot fi:

- Îmbunătățire a muchiilor dintr-o imagine;

a. Imagine originală

b. Imagine după prelucrare

- Îndepărtarea zgomotului;

a. Imagine originală

b. Imagine după prelucrare

- Îndepărtarea neclarităților datorate mișcării.

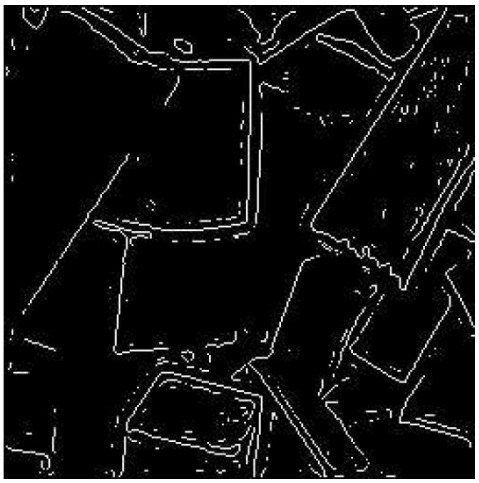
a. Imagine originală

b. Imagine după prelucrare

În cea de-a doua categorie se pot include operații în vederea:

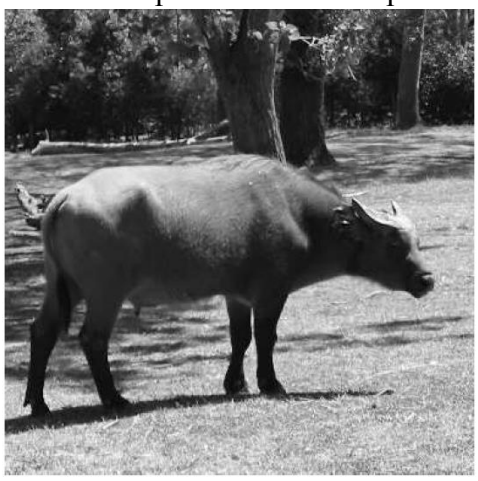
- Obținerii muchilor într-o imagine, în vederea determinării perimetrelor sau ariilor;

a. Imagine originală



b. Imagine după prelucrare

- Îndepărtarea detaliilor pentru a evidenția anumite elemente semnificative.



a. Imagine originală

b. Imagine după prelucrare

Între primele aplicaţii ale tehnicii de prelucrare a imaginilor a fost îmbunătăţirea transmisiei imaginii din ziar în formă digitală, prin intermediul cablului marin Londra – New York. În deceniile următoare metodele de prelucrare a imaginiilor s-au îmbunătăţit continuu, simultan cu apariţia a numeroase noi aplicaţii în diferite domenii: tehnica spaţială, medicină, studiul poluării etc. Multe din aceste domenii necesită îmbunătăţirea calităţii imaginii. Metodele de îmbunătăţire şi restaurare se folosesc şi la prelucrarea imaginiilor degradate ale unor obiecte irecuperabile (tablouri) sau în experimente prea costisitoare pentru a putea fi repetate.

Un alt domeniu major de aplicare a tehnicii de prelucrare a imaginii este

analiza ei cu calculatorul. Scopul acestei analize este extragerea informaţiei într-o formă cât mai adecvată prelucrării. Domeniile tipice de aplicaţii sunt: recunoaşterea caracterelor, vederea artificială, prelucrarea automată a amprentelor, analize biomedicale, analiza imaginiilor din satelit etc.

6.1. Achiziţia şi reprezentarea imaginilor

Prelucrarea imaginiilor digitale implică atât studierea stocării, cât şi manipularea imaginiilor cu ajutorul unui calculator sau un hardware specializat. Stocarea optimă a imaginii este importantă, deoarece abilitatea prelucrării eficiente depinde de modul de codificare folosit pentru reprezentarea imaginii originale. Imaginiile sunt transmise calculatorului într-o varietate de forme, depinzând de senzorul de imagine şi hardware-ul folosit pentru compactarea imaginii. În prezent cele mai uzuale tipuri de imagini sunt:

- imagini monocrome (cunoscute în literatură sub denumirea multi-level-gray-scale images);
- imagini binare, în care doar două niveluri de strălucire sunt permise;
- imagini color, ce pot fi reprezentate de trei componente monocrome separate;
- imagini multispectrale, cu mai mult de trei componente monocrome;
- perechi de imagini stereoscopice, din care rezultă vederea binoculară;
- imagini în mişcare (şiruri de imagini);
- imagini generalizate (o combinaţie a ultimelor trei tipuri).

Termenul de imagine monocromă sau imagine de niveluri de gri se referă la funcţia de intensitate bidimensională f(x,y), unde x şi y reprezintă coordonatele spaţiale, iar f este o funcţie ce ia valori proporţionale cu strălucirea sau nivelul de gri al imaginii în punctul (x, y).

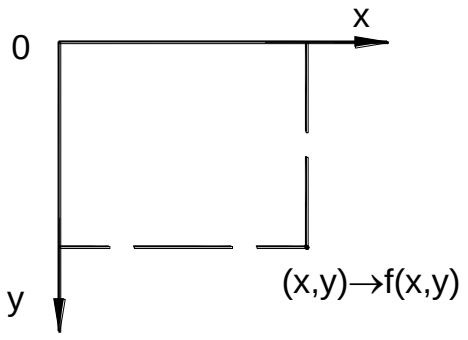


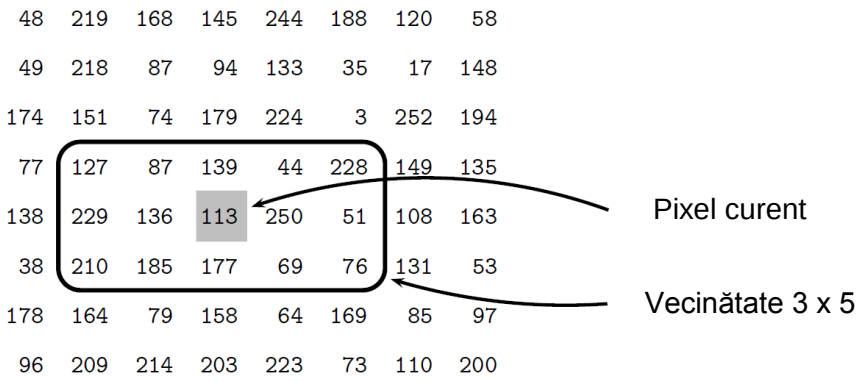
Fig. 6.1. Convenţia de axe

Convenţia de axe este cea prezentată în fig. 6.1, originea fiind plasată în colţul din stânga sus.

O imagine digitală este o imagine ce a fost digitizată atât în coordonate spaţiale, cât şi în strălucire. Ea poate fi considerată ca o matrice, la care indicii liniei şi coloanei identifică un punct din imagine, iar elementul corespunzător al matricei reprezintă nivelul de gri în punctul respectiv, ce poate fi un număr întreg din intervalul 0 – 255 sau un număr cuprins în intervalul [0, 1]. Elementele unei astfel de reţele digitizate se numesc elemente de imagine sau pixeli (picture element).

În diferite aplicaţii este avantajos ca matricea, ce stochează imaginea, să fie pătratică, iar numărul nivelurilor de gri să fie o putere întreagă a lui 2. Ex.: imaginea comparabilă cu imaginea TV alb-negru este 512 x 512 cu 128 niveluri de gri.

Pixelii ce înconjoară un anumit pixel constituie o vecinătate. Vecinătatea este caracterizată prin formă, ca în cadrul matricelor:3 x 5, 5 x 7 etc. De regulă, vecinătatea are un număr impar de linii şi coloane, ceea ce asigură faptul că pixelul curent este centrul vecinătății.



Există o serie de operații ce se efectuează în cadrul prelucrării imaginilor și acestea se pot clasifica în funcție de obiectivele propuse :

1. Îmbunătățirea imaginii – prelucrarea imaginii astfel încât rezultatul să fie convenabil unei anumite aplicații, cuprinde operații de mărirea contrastului sau luminozității, evidențierea muchiilor, înlăturarea zgomotului, îndepărtarea neclarităților (sharpening, deblurring) și încețoșerii datorate focusării necorespunzătoare.
2. Restaurarea imaginii sau eliminarea efectelor unei cauze cunoscute, cum ar fi: eliminarea neclarităților datorate mișcării, îndepărtarea distorsiunilor optice, îndepărtarea interferenței periodice.
3. Segmentarea imaginii, presupune împărțirea unei imagini în părți

constitutive sau izolarea anumitor caracteristici din imagine.

Aceste trei clase de operații nu sunt disjuncte, un anumit algoritm poate fi utilizat atât în cazul îmbunătățirii imaginii, cât și al restaurării.

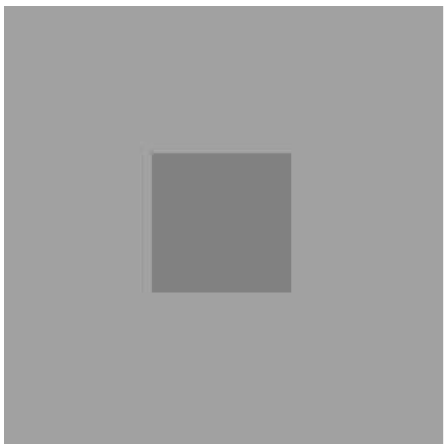
Pentru a înțelege mai bine modul cum se rezolvă o problemă de procesare a imaginii în condiții reale, se consideră operația de identificare automată a codului poștal de pe un plic.

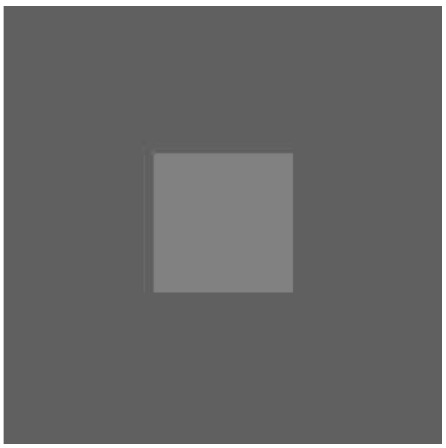
Etapele ce trebuie parcurse sunt:

- Achiziționarea imaginii – trebuie obținută imaginea digitală a plicului, se poate face cu o cameră CCD sau un scanner.
- Preprocesarea – conține operațiile efectuate înaintea realizării sarcinii principale. În acest caz poate înseamna mărirea contrastului,îndepărtarea zgomotului sau identificarea regiunii ce poate conține codul de interes.
- Segmentarea imaginii, prin care se extrage din imagine partea ce conține codul poștal.
- Reprezentarea și descrierea imaginii – se extrag anumite caracteristici ce permit diferențierea între obiecte. În acest caz se urmăresc curbe, găuri sau colțuri, care permit identificarea diferiților digiți.
- Recunoașterea și interpretarea – atribuirea unor etichete obiectelor, în funcție de descriptorii utilizați la pasul anterior și asocierea semnificației cu etichetele respective (se identifică cifrele componente și se formează șirul de șase caractere ce reprezintă codul poștal).

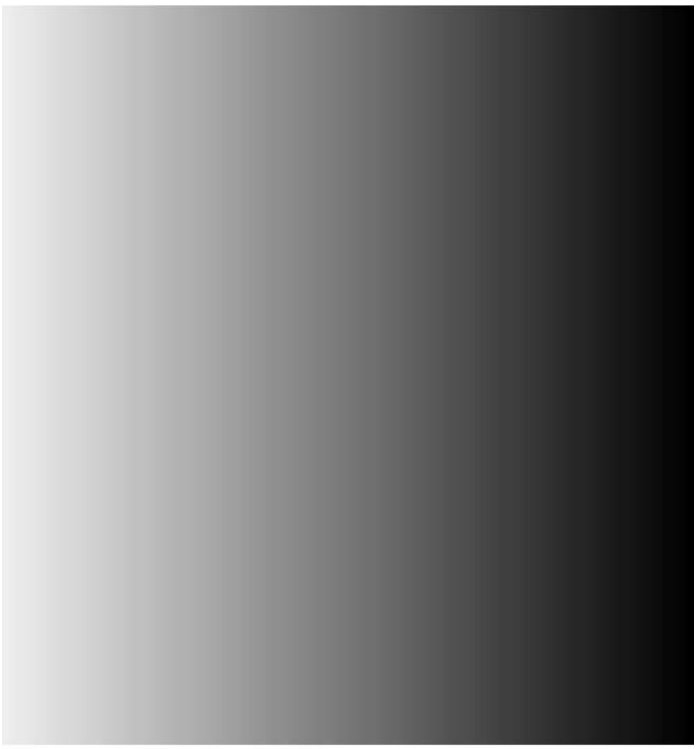
Legat de percepția umană a imaginilor trebuie menționate câteva aspecte:

- Intensitatea observată este influențată de fundal. Pătratul central este perceput mai închis la culoare, dacă este plasat pe un fundal deschis. Deși nunața de gri a pătratului central este aceeași, observatorul îl percepe diferit, deoarece percepția se face prin diferență față de mediul înconjurător.





- Se pot percepe linii inexistente într-o imagine ce conține nuanțe de gri ce variază continuu.



- Se supraestimează sau subestimează intensitatea luminoasă în apropierea frontierelor unui obiect de intensitate diferită față de mediul înconjurător. În cazul unei forme luminoase pe un fond întunecat, frontiera apare mai luminoasă decât restul, dacă privirea trece de la fundal spre obiect și invers, dacă privirea trece de la obiect spre fundal.

6.1.1. Reprezentarea și manevrarea imaginilor în Matlab

În Matlab matricile în care se stochează imagini pot fi de două tipuri:

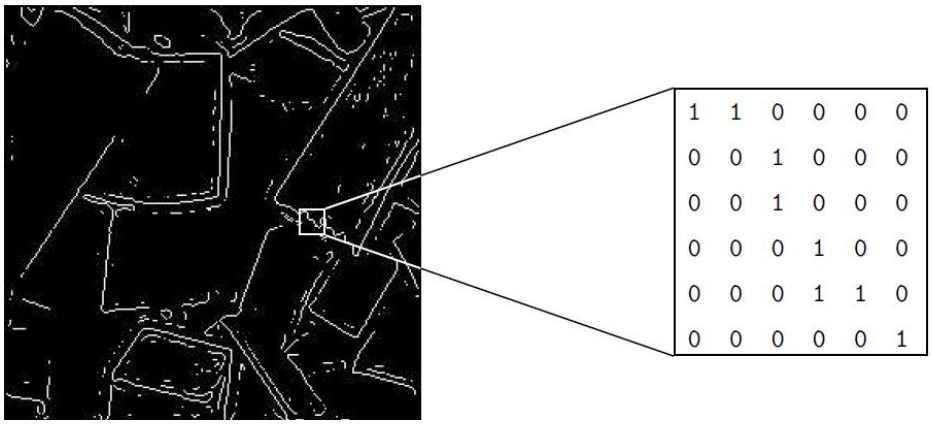
- matrici de clasă dublă, adică în dublă precizie, pe 64 de biţi, valorile fiind reprezentate în virgulă mobilă, ca numere aparţinând intervalului [0,1];
- matrici de clasă uint8, uint16, uint32 (uint = unsigned integer), conţinând valori întregi, fără semn, pe 8 biţi (între 0 și 255).

În mediul Matlab există patru tipuri de imagini:

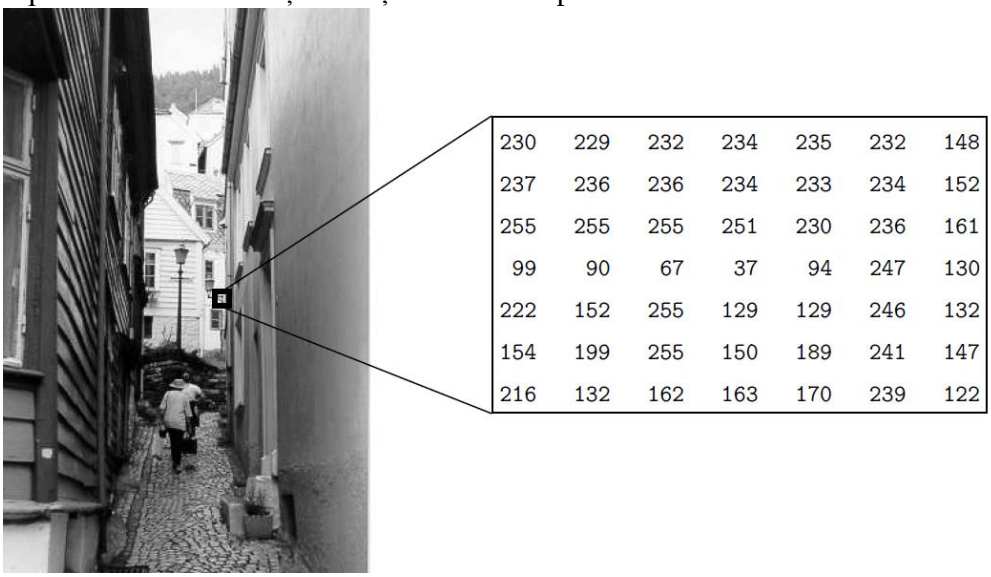
- binare;

- monocrome sau în nuanțe de gri;
- RGB – imagini color (Red, Green, Blue adică roşu, verde, albastru);
- indexate.

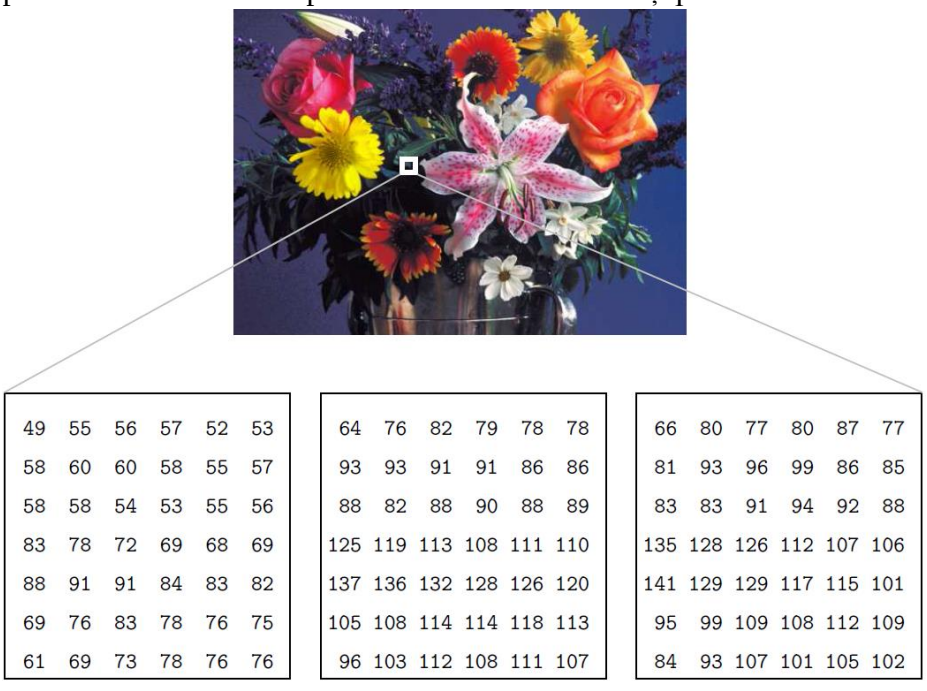
În cazul imaginilor binare, fiecare pixel poate avea una din cele două valori posibile, ex. 0 şi 1. Aceste imagini pot fi considerate ca un tip special de imagine de intensitate, ce conţine doar alb şi negru. Existând doar două posibilități pentru un pixel este suficient 1 bit pentru stocarea informației. Aceste imagini sunt foarte eficiente din punct de vedere al consumului de memorie.



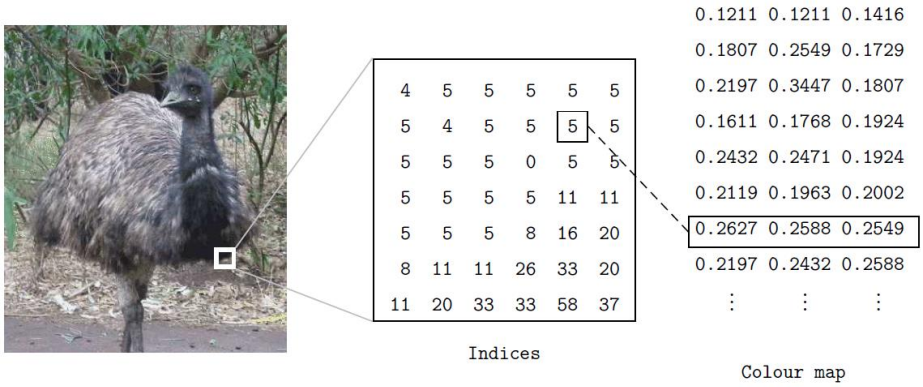
Imaginile de intensitate se reprezintă într-o singură matrice, fiecare element corespunzând unui pixel imagine. Fiecare valoare reprezintă intensitatea sau nivelul de gri (0 corespunde pentru negru, respectiv 255 reprezintă intensitatea maximă sau alb). În aceste condiții sunt necesari 8 biți pentru reprezentarea informației conținute într-un pixel.



În cazul imaginilor RGB fiecare pixel este reprezentat prin 3 valori – intensitatea celor 3 culori de bază, RGB. Spre deosebire de imaginile indexate, în acest caz există o singură matrice spaţială m x n x 3, m x n reprezintă pixelii imaginii şi există 3 plane m x n, câte unul pentru fiecare culoare. Dacă fiecare componentă are domeniul între 0 – 255, se pot obține 255^3 = 16.777.216 culori posibile. Pentru fiecare pixel sunt necesari 3 x 8 biți pentru stocarea informației.



Majoritatea imaginilor color utilizează doar un set restrâns din cele 16 milioane de culori posibile. Pentru a economisi spațiu de stocare și a manevra mai ușor fișierele, imaginea este asociată cu o hartă a culorii, ce este, de fapt, o listă a tuturor culorilor utilizate în imagine. Imaginile indexate se reprezintă în două matrici: una pentru imaginea propriu-zisă şi a doua pentru harta culorii. Pentru fiecare pixel imagine, prima matrice conţine o valoare ce reprezintă indexul pentru harta culorii, respectiv numărul liniei din matricea culorii. A doua matrice are trei coloane, fiecare coloană specifică culoarea prin cele trei componente fundamentale: roşu, verde, albastru. Valoriile din harta culorii sunt numere reale din domeniul [0, 1] (0 corespunde culorii negru şi 1 intensităţii maxime). Dacă o imagine conține 256 de culori sau mai puțin, memorarea indexului culorii necesită doar 8 biți. Anumite formate impun numărul de culori la 256, tocmai din acest motiv (GIF).



Pentru compararea necesarului de memorie se consideră o imagine ce conține 512 x 512 pixeli.

a. Imagine binară
   512 x512 x 1 = 262.144 biți = 32.768 bytes ≈ 0,033 Mb
b. Imagine indexată
   512 x512 x 8 = 262.144 bytes ≈ 0.266 Mb
c. Imagine color
   512 x 512 x 8 x 3 = 786.432 bytes ≈ 0.786 Mb

În Matlab se poate lucra cu imagini de tip: BMP, JPEG, TIFF, PNG, HDF, XWD.

Citirea imaginilor se face cu funcția imread, iar scrierea imaginii se face cu imwrite.

i=imread('peppers.png');

Valorile asociate unui (sau mai multor) pixeli se pot obține cu comanda impixel, ce returnează cele trei componenete de culoare asociate pixelului. Ea admite sintaxele:

- impixel(imagine, coloana, linie)

```
impixel(i,5,7)
ans =
    64    31    61
```

- impixel – pixelii de interes fiind selectați de utilizator cu ajutorul butonului dreapta al mouse-ului
- `[c,r,culoare]=impixel`

```
c =
   260
   394
r =
   210
   207
```

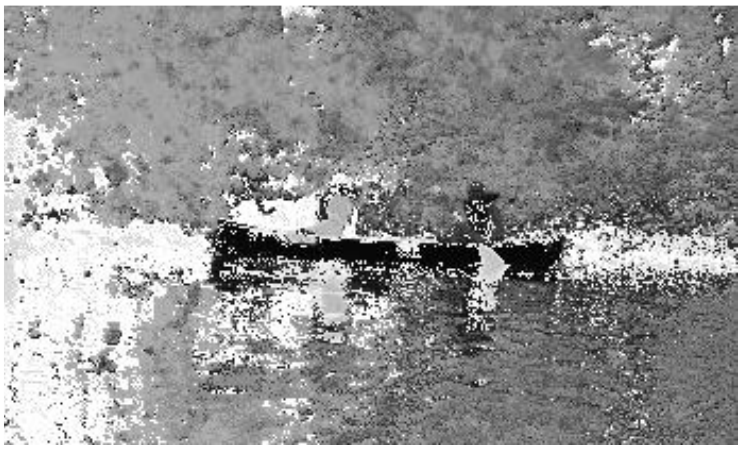
```
culoare =
   255   177     0
   191    21    35
```

Se observă că în ultima variantă, funcția returnează coordonatele pixelilor selectați și componenetele de culoare. Pixelul aflat în coloana 260 și linia 210 are R=255, G=177 și B=0.

De remarcat că adresarea comenzilor de prelucrare a imaginilor utilizează indicii sub forma (coloană, linie) spre deosebire de matrice, la care adresarea se face (linie, coloană).

În cazul imaginilor indexate, unde imaginea este formată din două matrici (harta de culoare și matricea de indici), se recomandă ca citirea să se facă împreună cu harta culorii. În caz contrar, fișierul este interpretat ca o imagine în nuanțe de gri.

```
>> [i1,i1map]=imread('canoe.tif');
>> figure,imshow(i1,i1map)
>> figure,imshow(i1)
```



O serie de informații despre fișierul de imagine pot fi obținute cu comanda **`imfinfo('nume_fisier.ext').`** Sunt numeroase informații ce nu prezintă interes, dar se poate vedea dimensiunea imaginii, mărimea fișierului în bytes, numărul de biți/pixel (BitDepth) și tipul imaginii (ColorType). În cazul imaginilor binare tipul imaginii este afișat 'grayscale', deoarece nu se face distincție între imaginile de niveluri de gri și cele alb-negru, dar se vede că numărul de biți/pixel este 1.

În anumite operaţii este util să se facă conversia între diferite tipuri de imagini. De ex., pentru filtrarea unei imagini color de tip indexat trebuie în prealabil să fie convertită imaginea în formatul RGB. Câteva din aceste funcţii sunt prezentate în Tabelul 6.1.

Tabelul 6.1

Funcția	Comandă	Scop
gray2ind	[y,map] = gray2ind(x) ;	Creează o imagine indexată din imagine de intensităţi de gri
ind2gray	y=ind2gray(x);	Creează o imagine de intensitate dintr-o imagine indexată
ind2rgb	y =rgb2ind(x);	Creează o imagine RGB dintr-o imagine indexată
rgb2gray	y = rgb2gray(x) ;	Creează o imagine de intensitate din imagine RGB
rgb2ind	[y,map] = rgb2ind(x);	Creează o imagine indexată din imagine RGB
mat2gray	y = mat2gray(x)	Creează o imagine de intensitate din datele unei matrici scalare
im2bw	BW = im2bw(I,LEVEL) BW = im2bw(X,MAP,LEVEL) BW = im2bw(RGB,LEVEL)	Creează o imagine binară din imagine de intensitate, indexată sau RGB, pe baza pragului de luminanţă

Matlab oferă un suport limitat pentru operaţii cu matrice de clasă uint8, în speţă nu admite operaţii matematice. Dacă se încearcă efectuarea unei operaţii matematice, se primeşte un mesaj de eroare. De ex.:

```
BW=BW1+BW2
??? Function '+' not defined for variables of class uint8
```

Din acest motiv pentru a efectua anumite operaţii sunt necesare conversii între tipurile de date. Ex.: **`BW=double(BW1)+double(BW2).`**

De remarcat că în urma conversiei se modifică modul în care Matlab interpretează datele din imagine. Pentru ca matricea să fie corect interpretată este necesară rescalarea sau decalarea datelor. Acest lucru trebuie făcut în matrici de clasă dublă, deoarece matricele de clasă uint8 nu suportă operaţii aritmetice. De aceea la conversia din clasă dublă în clasă uint8 trebuie să se execute toate operaţiile aritmetice înainte de apelarea funcţiei uint8; iar la conversia din uint8 în clasă dublă, trebuie să se efectueze toate operaţiile aritmetice după apelarea funcţiei double. În tabelul 6.2 se prezintă modalităţile de conversie pentru diferite tipuri de imagini.

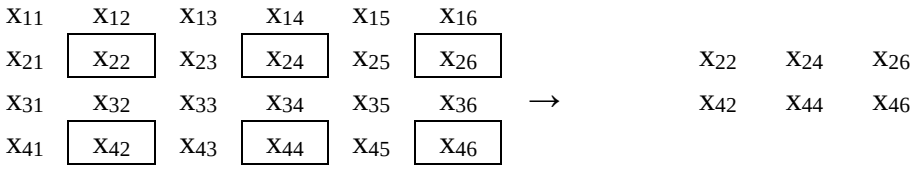
Tabelul 6.2

Tipul imaginii	Conversie uint8 → double	Conversie double → uint8
Indexată	B=double(A)-1	B=uint8(round(A-1))
Intensitate sau RGB	B=double(A)/255	B=uint8(round(A*255))
Binară	B=double(A)	B=logical(uint8(round(A)))

Pentru a evita posibile erori se recomandă efectuarea conversiei cu comenzile im2double(), respective im2uint8(), care fac automat scalarea sau decalarea datelor.

Calitatea unei imagini depinde și de rezoluția spațială sau densitatea de pixeli din imagine. Modificarea rezoluției se face cu funcția imresize(imagine, factor), unde factor este factorul de scalare ce se aplică imaginii.

Ex. imresize(I,0.5) va înjumătăți numărul de pixeli din imagine, prin eliminarea tuturor pixelilor ce nu au indicii pari, valorile effective fiind stabilite pe baza unor algoritmi.



În cazul când la imaginea rezultată se aplică modificarea rezoluției, dar cu factorul 2, imaginea revine la dimensiunea inițială, dar rezoluția se înjumătățește pe ambele direcții.

x_{22}	x_{22}	x_{24}	x_{24}	x_{26}	x_{26}
x_{22}	x_{22}	x_{24}	x_{24}	x_{26}	x_{26}
x_{42}	x_{42}	x_{44}	x_{44}	x_{46}	x_{46}
x_{42}	x_{42}	x_{44}	x_{44}	x_{46}	x_{46}

În Matlab, în afara sistemului de coordonate prezentat anterior, conform căruia un pixel este tratat ca o entitate discretă având coordonate numere întregi (numărul de linie, respectiv de coloană), se mai utilizează şi coordonatele spaţiale. În acest sistem, pixelul nu mai este tratat ca un punct, ci ca un pătrat având o arie. Coordonatele centrului fiecărui pixel (pătrat) sunt în corespondenţă cu reprezentarea în pixeli. Acest sistem de coordonate este continuu. Ordinea de indicare a coordonatelor este inversă (x, y), spre deosebire de sistemul pixel (coloană, linie – y, x), O altă diferenţă între cele două sisteme de coordonate este originea sistemului (colţul din stânga sus), care în cazul sistemului spaţial are coordonatele (0.5, 0.5), iar în sistemul de coordonate discret are coordonatele (1,1).

Afişarea imaginilor se face cu comanda imshow (nume.extensie), cu condiţia ca imaginea să fie introdusă în directorul de lucru curent al Matlab-ului sau dacă imaginea a fost citită în prealabil:

```
I=imread('puncte.jpg');
imshow(I)
```

Funcţia imshow produce doar afişarea unei imagini, fără a stoca fişierul respectiv în spaţiul de lucru. Pentru a putea prelucra imaginea, fişierul trebuie citit cu imread.

O altă posibilitate de vizualizare a unei imagini este cu comanda image, având sintaxa:

```
image(nume_imagine)
```

Imaginea va fi distorsionată şi pentru a păstra proporţiile trebuie trecute axele în modul imagine:

figura

```
axis image
```

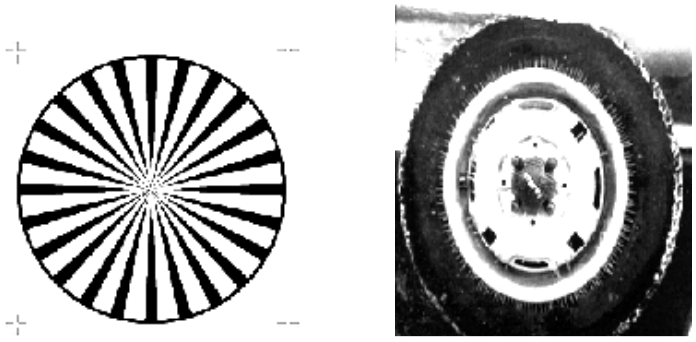
Având în vedere sistemul de axe asociat unei imagini (originea în colţul din stânga sus), în multe situaţii este avantajos să se transpună imaginea în coordinate carteziene, adică să se răstoarne imaginea cu comanda flipud (flip up down). Fie I o matrice reprezentând o imagine color, pentru a o transpune în coordinate carteziene se poate utiliza secvența:

```
for k=1:3
      I(:,:,k)=flipud(I(:,:,k)
end
```

Dacă se doreşte să se reprezinte imaginile într-un sistem de axe, având o anumită lungime a axelor, de ex. lx şi ly, pentru nu distorsiona imaginea trebuie păstrată proporţia laturilor imaginii. Raportul laturilor este utilizat pentru scalare. Comanda image creează un obiect imagine şi returnează pointerul pentru acesta. In pus, mai trebuie specificate câteva proprietăţi ale obiectului imagine:

- Proprietatea CData se setează egală cu valoarea pixelilor din imagine, adică se egalează cu variabila asociată imaginii;
- XData, YData – sunt vectori cu câte două valori, coordonatele x respectiv y ale colțului stânga jos şi dreapta sus al imaginii;
- Parent trebuie setată ca pointerul pentru sistemul de axe unde se afişează imaginea (gca – get current axis);
- AlphaData – este o proprietate prin care se poate specifica transparența imaginii. Este o matrice, de aceleaşi dimensiuni cu CData, dar valorile sunt cuprinse între 0 şi 1 (0 este transparent şi 1 opac).

```
%%demo_im.m
% demo afisare 2 imagini in spatiu lx x ly
%fiecare imagine avand alocata o lungime de 3/8*lx
%% date intrare
I1=imread('testpat1.png');
I2=imread('tire.tif');
lx=100;%lungimea ferestrei de vizualizare
ly=80; %inaltimea ferestrei de afisare
%% calcule
clf
axis([0,lx,0,ly]);
axis equal
[nx1,ny1]=size(I1);
[nx2,ny2]=size(I2);
raport1=ny1/nx1;
raport2=ny2/nx2;
inaltime1=3*lx/8*raport1;
inaltime2=3*lx/8*raport2;
x11=lx/16;x12=lx/2-lx/16;% coordonate x pt prima imagine, coltul
stg jos si dreapta sus
x21=lx/2+lx/16;x22=lx-lx/16;%coordonate x pt a doua  imagine,
coltul stg jos si dreapta sus
%% afisare imagini
hI1=image('Cdata',I1,'Parent',gca,'XData',[x11,x12],'YData',[0,i
naltime1]);
colormap(gray);
hI2=image('CData',I2,'Parent',gca,'XData',[x21,x22],'YData',[0,i
naltime2]);
axis off
```

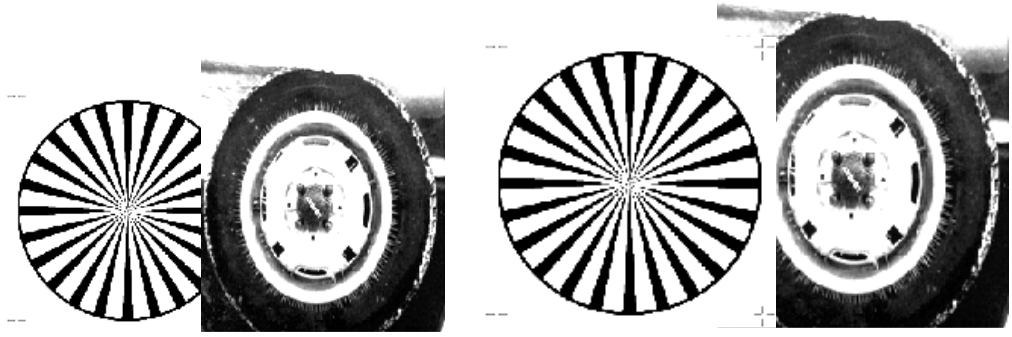


Una din imagini poate fi deplasată, putând să se suprapună parțial, prin actualizarea valorilor XData și YData într-o buclă. Obiectul imagine se manevrează prin pointerul său. Pentru aplicația anterioară, dacă se doește deplasarea imaginii I1 spre I2 se adaugă codul:

```
%% deplasare imagine I1 in k pasi
k=50;
x_depl=linspace(0,20,k);
for i=1:k
    set(hI1,'XData',[x11+x_depl(i),x12+x_depl(i)]);
    drawnow
end
```

Rezultatul se vizualizează în figură.Obiectul care este vizualizat deasupra este determinat de proprietatea children a sistemului de axe. Valoarea acestei proprietăți este un vector coloană, ce conține pointerul fiecărui obiect derivat (child). Ordinea poate fi modificată mutând celălalt obiect deasupra cu comanda:

```
set(gca,'children',flipud(get(gca,'children')));
```



Comanda surf poate fi utilizată la construirea unei suprafețe obiect pe care poate fi aplicată o imagine. Pentru a realiza acest lucru trebuie ca:

- proprietatea FaceColor să fie setată la valoarea ‘texturemap’;
- proprietatea CData să aibe valoarea corespunzătoare imaginii; nu trebuie

ca dimensiunea matricei imagine să corespundă discretizării de pe axa ox și oy;
- imaginea trebuie transformată în coordonate carteziene;
- se recomandă ca proprietatea EdgeColor să fie setată none (pentru a nu se vedea caroiajul).

```
%% demo_im_surf
% demonstreaza aplicarea unei imagini pe suprafata unui corp
(sfera)
%% date intrare
I1=imread('office_5.jpg');
for k=1:3
    I1(:,:,k)=flipud(I1(:,:,k));
end
%% construire sfera
[X,Y,Z]=sphere(30);
surf(X,Y,Z,'FaceColor','texturemap','CData',I1,'EdgeColor','n
one');
axis equal
%colormap(gray)
axis off
```

Dacă se dorește vizualizarea caroiajului, proprietatea EdgeColor trebuie setată la o culoare dorită, ex. [0.6,0.6,0.6].

6.1.2. Achiziţia imaginii

Achiziţia imaginii este o etapă esenţială deoarece condiţionează şi etapele ulterioare de prelucrare. Această etapă presupune prezenţa unui senzor de imagine şi capacitatea de digitizare a semnalului de ieşire. Senzorul de imagine este un dispozitiv sensibil într-o anumită gamă a spectrului energetic (gama corespunzătoare razelor X, UV, vizibil sau infraroşu). El produce la ieşire un semnal electric proporţional cu energia acestei radiaţii. Acest semnal este ulterior digitizat. De ex.: fie un sistem de achiziţie cu raze X; ieşirea unei surse de radiaţii este direcţionată spre un corp, iar un mediu sensibil la raze X se plasează în partea

opusă. Mediul va capta o imagine a ţesuturilor componente din care este constituit corpul respectiv, pe baza proprietăţii de absorbţie diferită a radiaţiei. Mediul în sine poate fi o peliculă sensibilă la raze X, o cameră video combinată cu un convertor de raze X.

În domeniul vizibil, senzorii de imagine uzuali sunt camera video sau dispozitivele videocaptoare integrate bazate pe transfer de sarcină, pe scurt CCD – charged coupled device.

Funcţionarea camerei video se bazează pe principiul fotoconductibilităţii: o imagine focalizată pe suprafaţa tubului produce un relief de conductibilităţi proporţionale cu distribuţia de străluciri din imaginea optică. Un fascicol de electroni, focalizaţi pe suprafaţa posterioară a ţintei fotosensibile a tubului, explorează punct cu punct această suprafaţă şi prin neutralizarea sarcinii electrice, creează o diferenţă de potenţial, ce produce pe un electrod colector un curent proporţional cu strălucirea de pe suprafaţa anterioară a ţintei. Imaginea obţinută este digitizată.

Dispozitivele videocaptoare integrate sunt compuse din mai mulţi senzori în stare solidă, configuraţi ca o matrice, de elemente de siliciu. Fiecare element generează un semnal electric de o amplitudine determinată de intensitatea luminii incidente. Această matrice poate acoperi o suprafaţă sau poate fi montată vectorial (de-a lungul unei linii). În cel de-al doilea caz, zona de interes se scanează linie cu linie, iar în primul caz scanarea se face în paralel.

Un aspect de maximă importanţă în procesarea imaginii este alegerea corectă a rezoluţiei (sau densităţii de eşantionare). Având în vedere că imaginea este un semnal în două dimensiuni, aceleaşi legi ce se aplică la procesarea semnalelor se aplică şi la imagini. Pentru a putea reprezenta o imagine în condiţii corespunzătoare (fără pierderi de detalii), imaginea trebuie eşantionată cu o frecvenţă (spaţială) care este dublul distanţei cele mai semnificative.

În vederea digitizării imaginii, la fel cum se face digitizarea spaţială şi intensitatea imaginii trebuie digitizată. Ieşirea unei camere video sau CCD este o tensiune analogică determinată de intensitatea luminii incidente. Domeniul de tensiuni (de la 0 la valoarea maximă obţinută pentru anumite intensităţi ale luminii incidente) se împarte într-un număr de intervale echidistante. Cu cât numărul de intervale creşte, cu atât creşte şi numărul nivelurilor de gri. În mod frecvent 256 de niveluri (ce pot fi reprezentate pe 8 biţi) se consideră suficient.

Ţinând cont de rezoluţia spaţială şi de nivelurile de gri (8 biţi), o imagine de 512 x 512 pixeli cu 256 niveluri de gri ocupă o memorie de cca 0,25 Mbyte. Pentru reprezentarea color, la aceeaşi rezoluţie spaţială se ajunge la 0,75 Mbyte (creşte de trei ori datorită reprezentării color în RGB).

6.2. Etape de prelucrare a imaginii

Diversitatea aplicaţiilor face dificilă încercarea de reprezentare unitară a etapelor de prelucrare şi a relaţiilor dintre acestea. Pe baza referinţelor bibliografice se propune o schemă bloc a procesului de prelucrare a imaginii, în fig. 6.3. Trebuie menţionat că nu toate aceste etape sunt prezente în orice prelucrare şi nu a fost inclusă etapa de comprimare a imaginii.

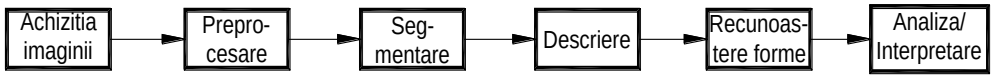


Fig. 6.2. Etapele prelucrării imaginilor

Etapa de preprocesare cuprinde o serie de operaţii ce au ca efect îmbunătăţirea imaginii. În timpul achiziţiei imaginii apar numeroase procese ce pot determina degradarea acesteia: focalizări necorespunzătoare, fie datorită mişcării, fie datorită ieşirii din câmp, distorsiuni geometrice etc.

Segmentarea imaginii are ca scop descompunerea ei în componente constitutive, eventual cu realizarea anumitor măsurări.

Orice operație de prelucrare a imaginii transformă valoarea pixelilor. În funcție de complexitatea operațiilor, ele se pot clasifica:

- operații la nivel de pixel,
- operații pe bază de vecinătăți, filtrări
- transformări sau operații globale.

Operațiile la nivel de pixel, se fac doar pe baza valorii pixelului respectiv, fără a ține cont de valorile pixelilor învecinați, filtrarea se face pe baza unei vecinătăți a pixelului, iar transformările se fac pe baza tuturor pixelilor din imagine. Deși operațiile la nivel de pixel sunt cele mai simple, constituie unele din cele mai puternice și larg utilizate operații în prelucrarea imaginilor. Se utilizează cel mai mult în faza de preprocesare, adică înaintea operației principale.

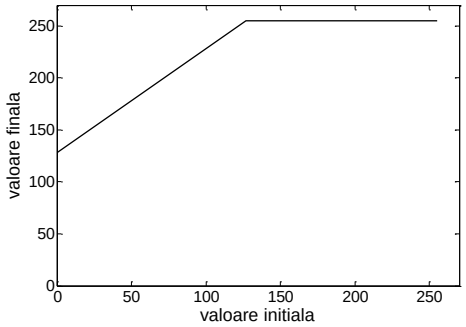
Operații la nivel de pixel

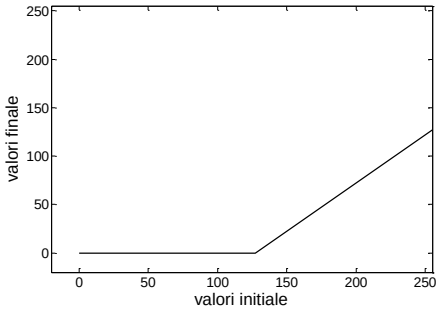
O primă categorie de operații la nivel de pixel sunt operațiile aritmetice, ce acționează prin aplicarea unei funcții simple, $y = f(x)$ asupra fiecărui nivel de gri din imagine, cu condiția ca domeniul de valori al funcției să rămână în domeniul [0, 255].

Astfel de operații sunt de tipul adunării sau scăderii unei constante din valoarea fiecărui pixel, $y = x \pm c$ sau multiplicarea fiecărei valori cu o constantă, $y = cx$.

În fiecare dintre cazuri trebuie pusă condiția ca rezultatul să rămână în domeniul [0, 255], prin mărginirea valorilor cu condiții de forma y = 255, dacă y>255, respectiv y = 0, dacă y<0.

Adunarea unei constante (128), respectiv scăderea uneia (128) se pot reprezenta prin transformările prezentate în figură.

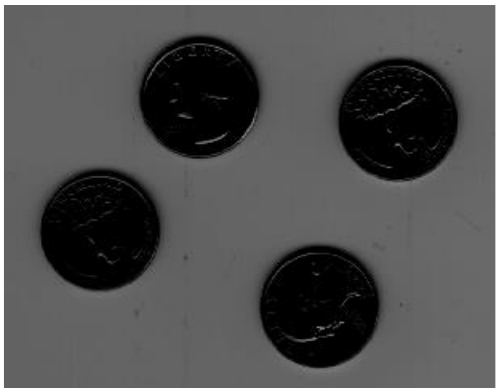




Imagine initiala

+128

-128



Se observă că adăugarea unei constante mărește luminozitatea, iar scăderea unei constante determină întunecarea imaginii.

Pentru a efectua aceste operații, matricea trebuie convertită în clasa double, efectuată operația și se revine la clasa uint8. O posibilitate mai elegantă este apelarea funcților imadd, respectiv imsubtract, care sunt special destinate imaginilor.

```
i=imread('eight.tif');
figure,imshow(i),title('Imagine initiala')
i1=imadd(i,128);
figure,imshow(i1),title('+128')
i2=imsubtract(i,128);
figure,imshow(i2),title('-128')
```

Multiplicare cu 0,5

Multiplicare cu 2

Luminozitatea se poate modifica și prin înmulțire cu o constantă. Înmulțirea este o operație care se aplică mai rar, în general se pierd multe detalii, deoarece valoriile pixelilor ajung să depășească limitele admise de reprezentarea imaginilor.

Complementara unei imagini este negativul fotografic al acesteia. Pentru imaginile în clasă dublă, negativul este 1 – I, iar dacă imaginea este binară, ~i. Pentru matricele de clasă uint8, funcția recomandată este imcomplement, care de fapt calculează 255-i.

Operaţiile de prelucrare de imagini prezentate până în acest moment lucrează asupra unei imagini şi produce o modificare. O altă clasă de operaţii utilizează 2 imagini pentru a produce o imagine nouă. Astfel de operaţii sunt descrise ca aritmetica imaginii, deoarece au la bază operaţii simple: adunare, scădere, eventual înmulţire şi împărţire. Ele se efectuează pixel cu pixel, astfel încât suma a două imagini va conţine valori obţinute prin însumarea intensităţilor pixelilor corespunzători din cele două imagini.

În cazul efectuării adunării, rezultatul poate depăşi domeniul de

reprezentare (dacă se adună 2 imagini având intensităţile cuprinse între 0 şi 255, rezultatul poate fi cuprins între 0 şi 510). Din acest motiv este necesar să se revină în domeniul admis. O variantă este împărţirea cu 2 a rezultatului. O altă variantă este depistarea domeniului de variaţie al intensităţilor (max – min) din imaginea rezultată şi efectuarea unei scalări dinamice cu expresia:

$$i' = 255 \cdot \frac{i_1 + i_2 - \min}{\max - \min}. \qquad (6.5)$$

Această variantă este mai avantajoasă deoarece prin împărţirea cu 2 se pierde partea fracţionară, fiindcă rezultatul trebuie să fie un număr întreg. Pe de altă parte, în cazul scalării dinamice este mai dificil să se compare imaginile, deoarece se modifică scala de intensitate. De asemenea, a doua variantă este mai laborioasă necesitând două parcurgeri ale imaginii, una pentru depistarea minimului şi maximului şi cea de-a doua pentru scalarea propriu-zisă.

Scăderea imaginii este în primul rând o metodă de depistare a diferenţelor între două imagini. În cazul scăderii, dacă imaginile iniţiale au valori cuprinse între 0 şi 255, rezultatul poate ajunge între -255 şi 255. Revenirea în domeniu de reprezentare se poate face prin adunare cu 255 şi împărţire la 2 sau cu scalarea dinamică prezentată anterior.

Prin scăderea a două imagini sunt eliminate toate trăsăturile comune şi scoase în evidenţă cele care se modifică. O utilizare majoră a acestei tehnici este în controlul calităţii. Se generează o imagine etalon, din care se scad imaginile obţinute în timpul procesului de fabricaţie. Orice abatere apare, ea se semnalează în noua imagine.

Se poate determina şi mişcarea obiectelor cu ajutorul scăderii, cu condiţia ca frecvenţa de eşantionare a imaginii să fie suficient de ridicată pentru ca imaginea obiectelor în mişcare să se suprapună parţial în cadre succesive. În acest mod, în imaginea finală apar zone luminoase. Prin împărţirea lungimii zonelor luminoase la intervalul de timp de eşantionare, se poate obţine viteza obiectului.

```
%demo determinare viteza de deplasare
clear
a=imread('circles.png');
[r,c]=find(a==1);
b=zeros(size(a));
[m,n]=size(a)
trans=[20,10];
r1=r+trans(1);c1=c+trans(2);
for i=1:length(r)
    if ((r1(i)<=n)&(c1(i)<=m))&((r1(i)>0)&(c1>0))
        b(r1(i),c1(i))=1;
    end
end
imshow(a)
```

```
a1=im2double(a);b1=im2double(b);
figure,imshow(b)
c=imsubtract(a1,b1);
figure,imshow(c)
x=regionprops(a,'Centroid');
x1=regionprops(b,'Centroid');
lx=round(x1.Centroid(1)-x.Centroid(1))
ly=round(x1.Centroid(2)-x.Centroid(2))
```

Înmulţirea şi împărţirea se folosesc foarte puţin ca operaţii matematice cu imagini. Este foarte ddificil să se facă rescalarea imaginii, deoarece produsul a două numere aparţinând domeniului [0, 255], generează un număr aparţinând intervalului [0,65.000]. Prin rescalarea rezultatului în domeniul admis se pierde multă informaţie. Prin înmulţire se poate aplica o textură pe un corp, iar prin împărţire se poate elimina fundalul [Russ].

Un calculator uzual poate afişa 2^8 sau 256 nuanţe de gri, respectiv poate genera culori cu acelaşi număr de nuanţe pentru roşu, verde şi albastru, ceea ce produce un total de 2^{24} sau 16 milioane de culori diferite. Această gamă este cunoscută sub denumirea "true color", deoarece poate reproduce majoritatea imaginilor din natură. Aceste posibilităţi ale calculatorului depăşesc posibilităţiile de percepţie umane, care se reduc la câteva zeci de nuanţe de gri şi sute de nuanţe cromatice. Pentru comparare, o fotografie de bună calitate poate reproduce 20 până la 30 de niveluri de gri.

Cu toate acestea, privind o imagine nu se pot distinge detaliile din zonele mai luminoase sau mai întunecate ale unei imagini în nuanţe de gri. Pentru a putea creşte vizibilitatea în anumite regiuni se poate modifica luminozitatea.
În fig. 6.6 se prezintă câteva funcţii de transfer (intensitatea de afişare în raport cu intensitatea imaginii stocate în memorie).

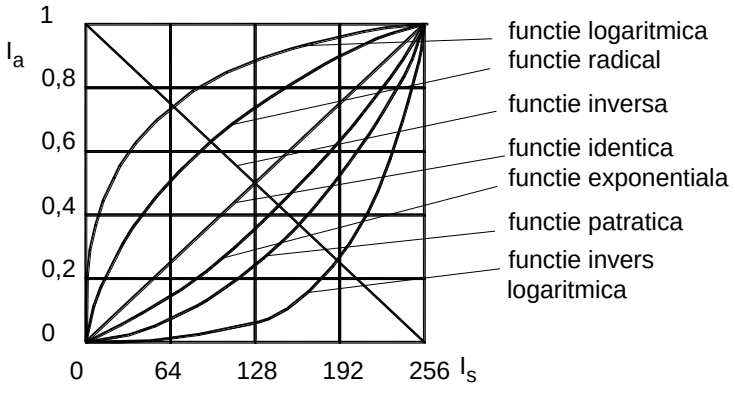
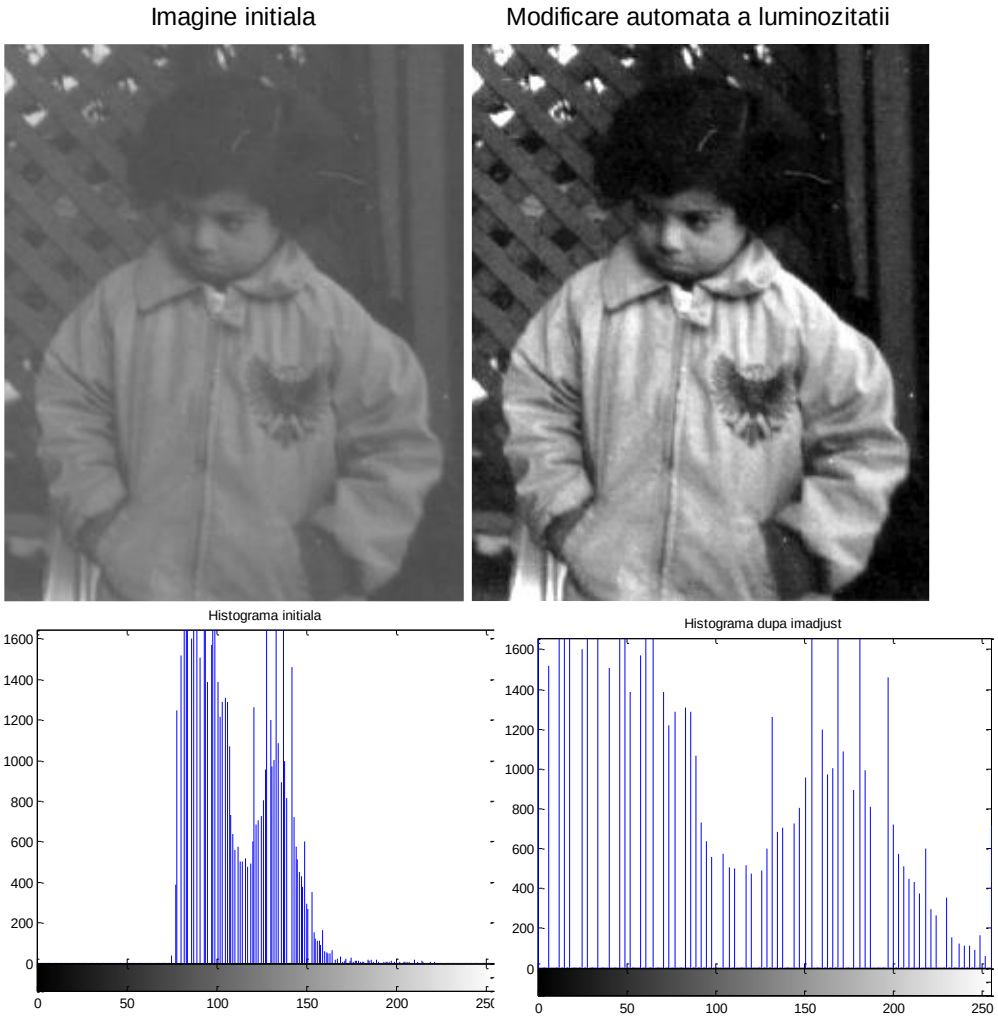


Fig. 6.6. Exemple de funcţii de transfer

O curbă logaritmică sau radical va micşora luminozitatea la maximul scalei şi o va mări în zonele întunecate. O curbă pătratică sau invers logaritmică va produce efectul contrar. Oricare din aceste funcţii se pot utiliza împreună cu creşterea contrastului pe întreg domeniul de afişare.

În Matlab, modificarea luminozităţii se face cu comanda imadjust, care lucrează cu imagini de niveluri de gri. Comanda se poate aplica în modul automat:

```
i=imread('pout.tif');
ia=imadjust(i);
figure,imshow(i),title('Imagine initiala')
figure,imshow(ia),title('Modificare automata a luminozitatii')
```



Rezultatul aplicării se vede în figură. Matlab calculează valorile intensității și le extinde pe domeniul admisibil sau se poate solicita de utilizator modul de aplicare:

```
imadjust(I,[inf sup],[jos sus])
```

unde I este matricea imagine, inf și sup sunt nivelurile intensităţii în imaginea inițială, care se modifcă la valorile jos și sus, valori cuprinse între 0 și 1. Pentru a obține mai multe informații despre imagine se poate construi histograma pentru intensitățiile din imagine, din care se poate constata:

- modul cum se distribuie intensitățile pe domeniul de valori;
- într-o imagine întunecată, histograma se va concentra spre valori mai mici;
- într-o imagine luminoasă, histograma va fi concentrată spre valori mai mari;
- pentru o imagine cu contrast bun, histograma va fi împrăștiată pe tot domeniul de intensități.

Vizualizarea histogramei se face cu funcția imhist(imagine). În acest mod se pot extrage valorile necesare pentru aplicarea funcției imadjust.

Dacă există anumite zone întunecate, în care se dorește vizualizarea unor detalii se poate mări luminozitatea considerabil, lărgind astfel domeniul pentru zonele întunecate, ca în exemplul următor. Domeniul [0 51] se modifică în [128 255]:

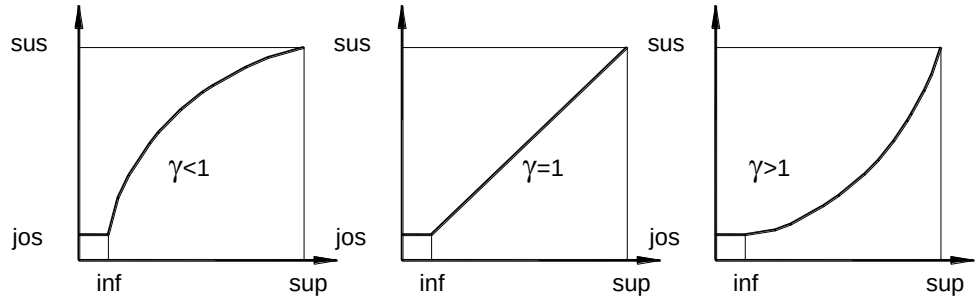
```
I=imread('cameraman.tif');
J=imadjust(I,[0 0.2],[0.5 1]);
imshow(I),figure,imshow(J)
```

Cele două imagini sunt prezentate în figura următoare. În urma acestei operaţii se pot distinge o serie de detalii, care nu se văd în imaginea inițială.

Funcția se poate aplica atât imaginilor în nuanțe de gri, cât și imaginilor color. În cazul imaginilor indexate se recomandă conversia în imagini RGB înainte de aplicarea funcției.

Funcţia imadjust scalează liniar valoarea intensităţii între limitele date. Ea admite şi un argument suplimentar, ce specifică factorul de corecţie γ. În funcţie de valoarea acestuia, scalarea poate fi neliniară. Factorul de corecţie admite valori între 0 şi ∞. Valoarea implicită este 1, cea corespunzătoare scalării liniare. În figura 6.9 sunt prezentate funcţiile de transfer pentru diferite valori ale lui γ.

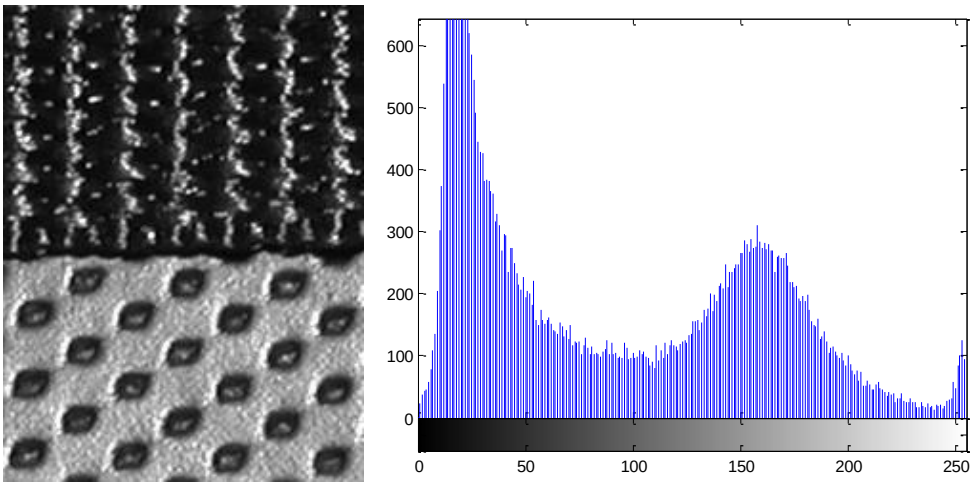
Dacă γ < 1, imaginea este mai luminoasă decât cea iniţială, iar dacă γ > 1, imaginea este mai întunecată. În fig. 6.10 se prezintă o imagine cu γ = 1, γ = 0.5 şi γ = 2.



Funcţiile de transfer pentru comanda imadjust

Modificarea luminozităţii cu funcţia imadjust

În anumite situaţii este avantajos să se construiască funcţia de transfer pentru o imagine specifică, în vederea obţinerii rezultatelor optime. Cea mai uzuală metodă este egalizarea histogramei. În histogramă se evidenţiază numărul de pixeli din imagine având cele 256 valori posibile ale intensităţii. Vârfurile indică intensităţile cele mai uzuale, iar văile indică valori ale intensităţii mai puţin folosite. Apariţia unor zone libere la extremităţi se datorează faptului că intensitatea nu acoperă întregul domeniu admis. Procesul de egalizare al histogramei, determină o împrăştiere a nivelurilor de gri dinspre vârfuri spre văi, astfel încât un acelaşi număr de pixeli să indice nivelurile de gri posibile. Funcţia de transfer, în acest caz, este histograma originală a imaginii redesenată ca un grafic cumulativ.



Egalizarea histogramei – imaginea iniţială cu histograma asociată şi imaginea finală cu

histograma egalizată

Vizualizarea unei histograme se face în Matlab cu funcţia imhist(I,n), unde I este imaginea, iar n este numărul de clase. În fig. 6.11 se prezintă o imagine cu histograma aferentă.

Egalizarea histogramei se face cu comanda histeq(I). rezultatul se observă în fig. 6.11.b, împreună cu noua histogramă.

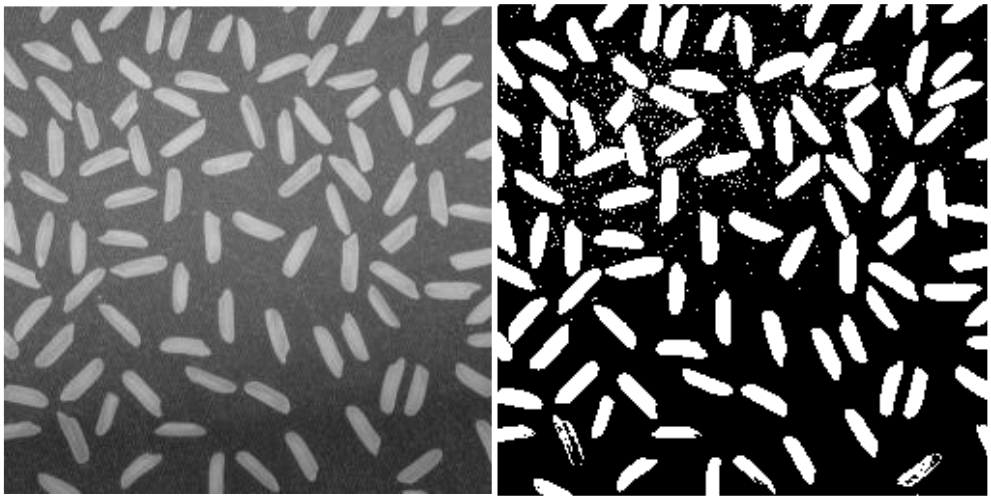
Egalizarea histogramei se poate face în anumite zone ale imaginii, pentru scoaterea în evidenţă a unor detalii. Acest lucru poate fi realizat cu o decuparea imaginii, cu comanda imcrop, iar ulterior noua imagine va fi supusă unui proces de egalizare a histogramei. Funcţia imcrop decupează un dreptunghi, ce poate fi specificat ca argument sau selecta cu mouse-ul. Sintaxa este:

```
I2 = imcrop(I,drept) ;
```

unde drept este un vector cu 4 elemente [xmin ymin laţime înălţime]. În cazul selectării dreptunghiului cu mouse-ul se pot obţine coordonatele pixelilor, dându-se comanda sub forma :

```
[I2,drept]=imcrop(I1);
```

unde dreptt are aceeaşi semnificaţie. Coordonatele dreptunghiului sunt exprimate în coordinate spațiale, deci trebuie rotunjite la valori întregi.



O a treia categorie de operații la nivel de pixel o constituie binarizarea unei imagini. O imagine în nuanțe de gri poate fi transformată în imagine binară alegând o anumită valoare de intensitate, T, din imaginea inițială și transformând în negru sau alb toți pixeli care au valoarea mai mare sau mai mică decât T.

Binarizarea este o parte importantă din segmentarea imaginilor, prin care se dorește izolarea unor obiecte față de fundal. Această binarizare se paote face

foarte simplu în MATLAB prin comanda x>T.

```
r=imread('rice.png');
figure,imshow(r)
figure,imshow(r>120)
```

Această comnadă se poate aplica imaginilor de niveluri de gri, dar există și funcția im2bw. Sintaxa este:

im2bw(imagine,nivel)

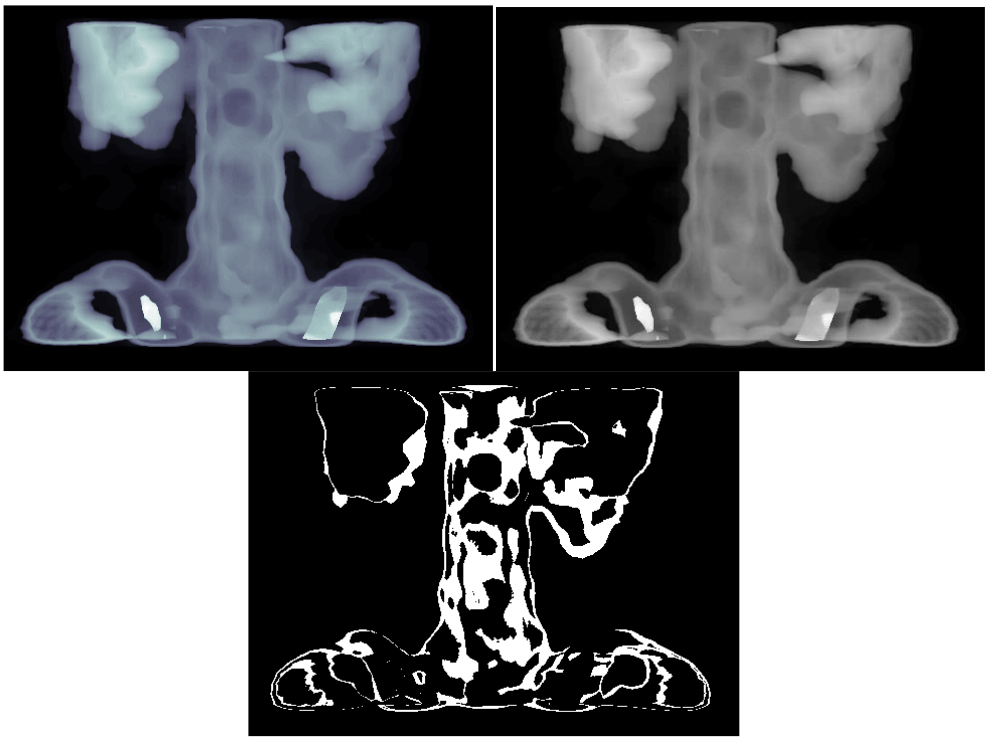
unde nivel este o valoare din [0, 1] ce indică fracția din intensitate imaginii care se transformă în alb. Comanda funcționează pentru orice tip de imagine, În nuanțe de gri, color sau imagini indexate, ce pot fi stocate în matrice de clasă double sau uint8.

Funcția calculează valoarea indicată prin parametrul nivel și o adaptează tipului de imagine la care este aplicată.

Se poate aplica și binarizare dublă, adică valoarea unui pixel va fi alb, dacă valoarea pixelului respectiv aparține unui anumit interval, toți ceilalți pixeli, care nu îndeplinesc condiția respectivă, devenind negri.

```
[x,map]=imread('spine.tif');
figure,imshow(x,map)
g=ind2gray(x,map);
figure,imshow(g)
figure,imshow(g>110&g<125)
```

În exemplul prezentat se citește o imagine indexată, se convertește în imagine în nuanțe de gri și se pune condiția ca valoarea pixelilor albi să fie cuprinsă între valorile 110 și 125.

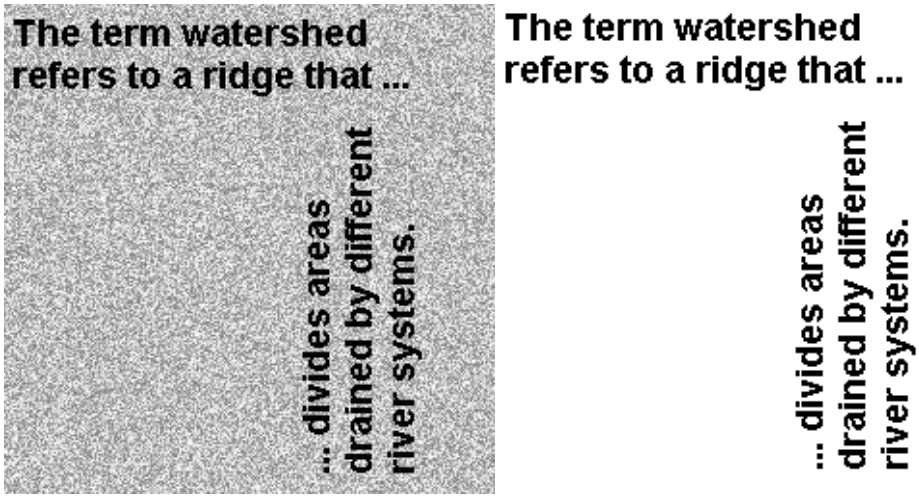


Binarizarea este utilă în numeroase situații:

- dacă se dorește eliminarea detaliilor inutile dintr-o imagine;
- pentru a scoate în evidență anumite detalii ascunse;
- când se dorește eliminarea fundalului pentru un desen sau text

Această ultimă utilizare se exemplifică în următorul cod. Se generează un fundal (256 x 256) ce conține valori între 127 și 255. Se citește imaginea textului, initial alb pe negru. Se inversează imaginea (not()) pentru a avea textul negru pe alb, se convertește în matrice de clasă double, pentru a putea efectua operațiile aritmetice, se aplică fundalul și se revine la clasa uint8.

```
fundal=rand(256)*128+127
text=imread('text.png');
tr=uint8(fundal.*double(not(text)))
figure,imshow(tr)
figure,imshow(tr>100)
```



6.2.1. Corectarea defectelor

O primă categorie de operaţii de prelucrare a imaginilor o constituie corectarea anumitor defecte ale imaginii achiziţionate, ce se pot datora erorilor de detecţie, iluminării neuniforme sau inadecvate etc. Trebuie remarcat faptul că aceste corecţii se aplică după achiziţionarea şi stocarea imaginii şi nu pot compensa în totalitate achiziţia incorectă, în sensul că rezultatul nu se poate ridica la valoarea unei imagini achiziţionate în condiţii optime. În anumite situaţii achiziţia optimă a unei imagini este impracticabilă: sursa de iluminare nu poate fi controlată pentru a fi perfect centrată şi normală la suprafaţa vizualizată în cazul când iluminarea se face de la soare; suprafţa vizualizată este curbă astfel încât apare o iluminare neuniformă ce trebuie corectată ulterior etc. Zgomotul din imagine poate să apară şi datorită instabilităţii sursei luminoase sau detectorilor din timpul scanării sau digitizării imaginii. Modelul acestui zgomot poate diferi mult de modelul normal sau Gauss, dar se manifestă ca o variaţie de intensitate. Metoda cea mai uzuală de minimizare a efectelor de fluctuaţie este utilizarea unui timp de scanare mult mai scurt sau mult mai lung decât timpul de fluctuaţie.

Se presupune că imaginea reprezintă cea mai bună calitate ce poate fi practic obţinută şi în aceste condiţii, în acest paragraf se vor prezenta metode de atenuare a zgomotului, ce permit îmbunătăţirea vizualizării şi demarcării detaliilor în vederea măsurării.

O altă ipoteză este că pixelii din imagine sunt mult mai mici decât orice detaliu semnificativ şi majoritatea pixelilor învecinaţi reprezintă aceeaşi structură. În aceste condiţii se pot aplica metode de mediere şi comparaţie pentru eliminarea zgomotului.

Filtrarea este o transformare a imaginii pe baza vecinătății pixelului. Ideea filtrării este mutarea unei măști sau nucleu de convoluție (un dreptunghi de dimensiuni impare sau alte forme) peste imaginea respectivă. Pe măsură ce se derulează procesul, se obține o nouă imagine, ai cărei pixeli au valoarea intensității calculate pe baza valorilor aflate sub mască. Combinația dintre mască și funcția de calcul este filtrul. În cazul când funcția este liniară, filtrul va fi liniar. Ex. un filtru liniar ce utilizează o mască 3 x 3 și calculează media celor 9 valori de intensitate, pe care o atribuie pixelului central, adică noua valoare a pixelului e va fi :

a	b	c
d	e	f
g	h	i

$$\rightarrow e' = \frac{1}{9}(a + b + c + d + e + f + g + h + i)$$

Este convenabil să se descrie filtrul liniar pe baza coeficienților cu care se multiplică nivelurile de gri din vecinătate. În acest caz filtrul poate fi descris prin matricea $\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$. Un filtru $\begin{bmatrix}1 & -2 & 1\\ -2 & 4 & -2\\ 1 & -2 & 1\end{bmatrix}$ va opera asupra imaginii din figura anterioară, rezultând noua valoare pentru pixelul central e = a - 2b + c -2d + 4e - 2f + g -2h + i

Cea mai simplă metodă de mediere spaţială este însumarea valorilor intensităţii mai multor pixeli din vecinătate, împărţirea rezultatului la numărul pixelilor şi construirea unei noi imagini. Acest lucru însă generează o estompare a contururilor precum şi detaliilor foarte fine. În cazul transformărilor locale, valoarea intensităţii unui punct în imaginea nouă este o funcţie de valoarea intensităţii pixelilor din vecinătatea celui iniţial. Acest domeniu este definit de un aşa numit “nucleu”.

Filtrarea liniară este implementată prin convoluţia bidimensională. În convoluţie, valoarea pixelului de ieşire este obţinută prin înmulţirea a două matrici şi însumarea elementelor. O matrice este prelevată din imaginea propriu-zisă, iar cea de-a doua este filtrul. Matricea ce reprezintă filtrul se mai numeşte nucleu de convoluţie, h.

De obicei, acest nucleu are forma unui pătrat format dintr-un număr impar de pixeli (3 x 3, 5 x 5, (2n+1) x (2n+1) pixeli). În cazul utilizării unei transformări liniare, fiecărui element al nucleului i se asociază un coeficient. Fiecare pixel este reprezentat prin intensitatea $p_{x,y}$. Intensitatea unui punct din imaginea rezultantă va fi o sumă ponderată a intensităţilor pixelilor din zonele vecine:

$$p^*_{x,y} = \frac{\sum_{i,j=-m}^{m} h_{i,j} \cdot p_{x+i,y+j}}{\sum_{i,j=-m}^{m} h_{i,j}}, \qquad (6.1)$$

unde $h_{i,j}$ sunt factorii de pondere. Cu ajutorul acestui nucleu se determină doar intensitatea centrală a nucleului (pixelul aflat la intersecţia celor două diagonale ale nucleului).

Operaţia de vecinătate se aplică simetric în jurul fiecărui pixel, ceea ce determină apariţia unei probleme pentru pixelii aflaţi în apropierea extremităţii imaginii la o distanţă mai mică decât jumătatea nucleului.

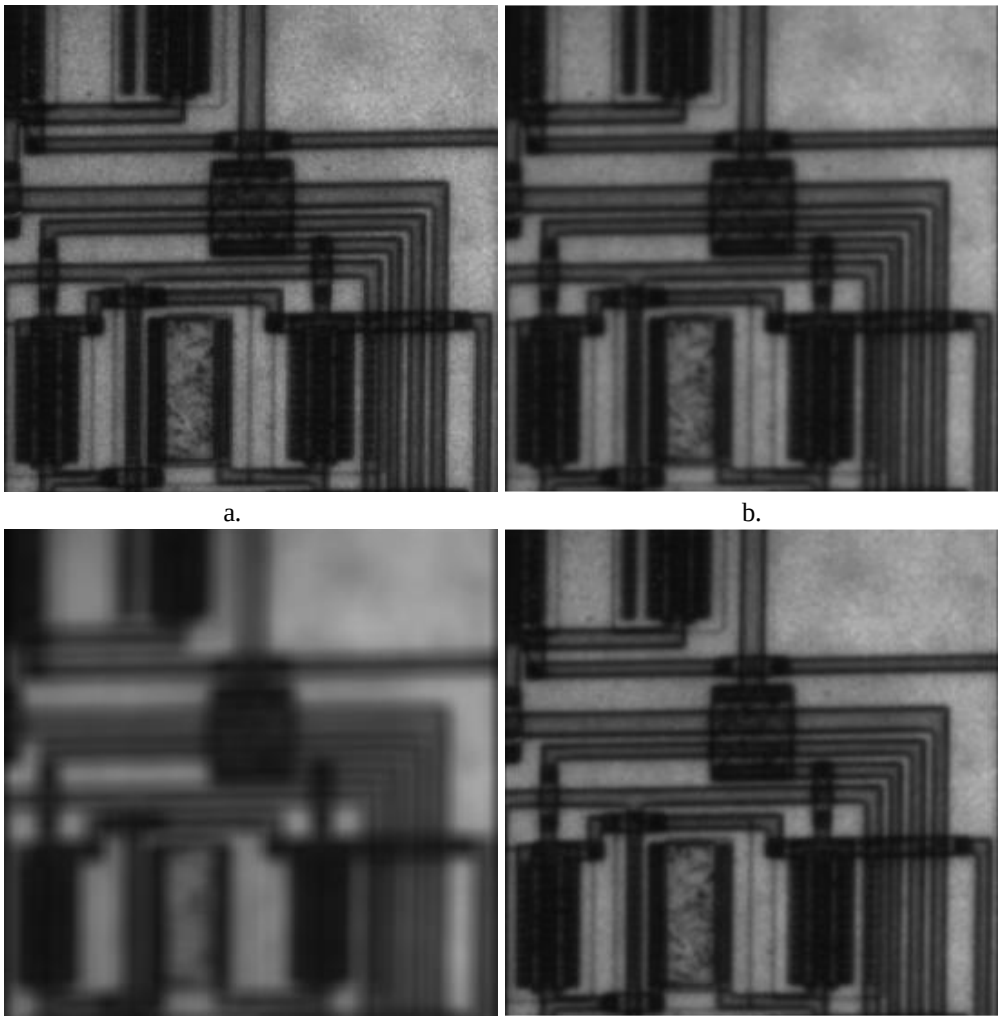
În fig. 6.3 se prezintă o imagine netezită cu un nucleu 3 x 3 şi 11 x 11, având factorii de pondere egali cu 1. Reducerea zgomotului este mai accentuată în cazul nucleului mai mare, dar difuzia contururilor este mult mai pronunţată.

Difuzia contururilor poate fi controlată prin utilizarea unor factori de pondere diferiţi. Un exemplu ar fi un nucleu de forma:

$$h = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix},$$

rezultatul netezirii fiind prezentat în fig. 6.3.c. Acest tip de nucleu prezintă câteva avantaje:

- factorul de pondere central 4 este cel mai mare, ceea ce determină un efect dominant al pixelului central, reducând difuzia;
- factorul de pondere 2 atribuit pe linie, respectiv coloană şi 1 pe diagonală sunt determinaţi de distanţa până la pixelul central, care este mai mare pe diagonală cu factorul $\sqrt{2}$.



a. b.

c. d.

Imaginea iniţială şi filtrări prin netezire cu diferite nuclee

Acest procedeu de filtrare prin mediere se poate face în Matlab cu comanda filter2, iar cu comanda fspecial se generează filtrul.

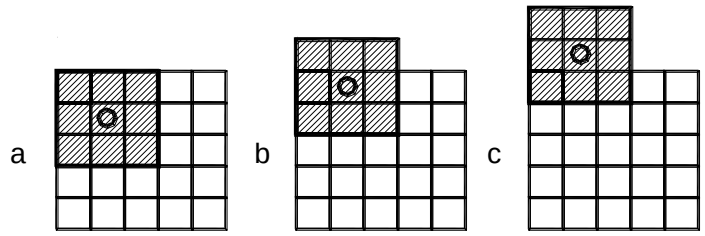
În primul caz sintaxa este:

`B=filter2(h,A)` sau **`B=filter2(h,A,parametru de formă),`**

unde h este nucleul de convoluţie, A este imaginea iniţială, iar parametrul de formă este un şir de caractere ce poate avea una din valorile: **`'full', 'same'`** sau **`'valid'.`**

Matricea A trebuie să fie de clasă dublă sau orice clasă întreagă, iar rezultatul, B, este o matrice de clasă dublă.

Parametrul de formă determină mărimea matricei de ieşire. La aplicarea nucleului de convoluţie, acesta poate fi aplicat numai în suprapunere completă cu imaginea iniţială (elementul central rămâne la o distanţă (h-1)/2 de margine) sau şi în afara imaginii (vezi figura)



Posibilităţi de amplasare a nucleului de convoluţie a.- valid,
b. – same, c. - full

În cazul când parametrul de formă are valoarea ‘valid’, nucleul de convoluţie se plasează numai în suprapunere completă cu imaginea, astfel încât imaginea rezultată va avea dimensiunea mai mică decât imaginea iniţială cu intervalul (h-1)/2 – fig. 6.4.a. Pentru valoare ‘same’ (cazul implicit) se obţine o imagine de aceeaşi dimensiune cu matricea iniţială, iar pixelii din margine se calculează atribuind valoarea 0 pixelilor ce nu aparţin imaginii. Pentru cazul ‘full’ (fig. 6.4.c) se obţine o imagine de dimensiuni mai mari, deoarece nucleul se aplică şi-n afara imaginii.

Nucleul de convoluţie se defineşte cu factorii de pondere doriţi. Ex.: h=[1 2 1;2 4 2;1 2 1];, dar ulterior nucleul trebuie normalizat (suma elementelor trebuie să fie egală cu 1).

Comanda fspecial conţine o serie de filtre predefinite. Sintaxa este:

`h=fspecial(tip)` sau **`h=fspecial(tip,parametru)`.**

Dacă parametru este un scalar, filtrul va fi o matrice pătratică sau poate fi un vector cu două dimensiuni. În varianta implicită este filtrul are dimensiunea 3 x 3.

În cazul filtrării prin mediere, tip este o variabilă de tip şir ce are valoare ‘average’, iar parametru este dimensiunea nucleului. Implicit este 3 x 3. Există și varianta

`h=fspecial('disk',raza),`

care generează un filtru de mediere de formă circulară, în varianta implicită raza fiind 5.

Pentru exemplificare se prezintă aplicaţia care a generat figura anterioară

```
%%demo_filtrare.m
% exemplificari pt diferite filtre
%% citire imagini
ii=imread('cameraman.tif');
i=im2double(ii);
ii1=imread('circuit.tif');
```

```
i1=im2double(ii1);
%% filtru average
f1=fspecial('average');
if1=filter2(f1,i);
i1f1=filter2(f1,i1);
figure,imshow(i),title('Imagine initiala')
f2=fspecial('average',9);
if2=filter2(f2,i);
f3=fspecial('disk');
if3=filter2(f3,i);

figure,imshow(if1),title('Filtru average')
figure,imshow(i1),title('Imagine initiala')
figure,imshow(i1f1),title('Filtru average')
figure,imshow(if2),title('Filtru average 9x9')
figure,imshow(if3),title('Filtru disk')
```

Filtrarea prin mediere este o operaţie liniară prin care nu se pierd informaţii din imaginea originală, dar se estompează muchile, efect ce se amplifică cu creșterea dimensiunilor nucleului.

Imagine initiala

Filtru average

Filtru average 9x9

Un aspect important într-o imagine este cât de rapid se modifică valorile intensităților în raport cu distanța, noțiune ce reprezintă frecvența unei imagini. Componentele de frecvență înaltă se caracterizează prin modificări mari de-a lungul unor distanțe mici, cum ar fi muchii sau zgomot. Componentele de frecvențe joase se caracterizează prin modificări mici, cum ar fi fundal, texturi omogene.

Pe baza acestor considerente se pot defini:

- filtru trece sus (high pass filter), care lasă componentele de frecvențe ridicate și elimină sau reduce componenetele joase;
- filtru trece jos – reduce sau elimină componentele de frecvență ridicată, lăsându-le pe cele de frecvență joasă.

Filtrul de mediere este un filtru trece jos, deoarece estompează muchiile.

Filtrul $\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$ este un filtru trece sus.

Un filtru trece sus are suma coeficienților (elementele din mască) egală cu 0. Într-o imagine cu frecvențe joase, adică cu valori similare ale pixelilor, rezultatul aplicării va fi aproape de 0.

Un alt filtru trece jos, predefinit în MATLAB este filtrul Gaussian, elementele nucleului de convoluție fiind generate pe baza funcției densitate de repartiție normală. Generarea unui astfel de filtru se poate face cu comanda

```
fspecial('gaussian',[a,b],sigma)
```

`fspecial('gaussian',a,sigma)`

unde a x b sau a x a este dimensiunea nucleului, iar sigma este abaterea standard a repartiției. Valorile implicite sunt 3 x 3 și sigma = 0.5. Efectul aplicării unui astfel de filtru poate fi vizualizat în figură. Se remarcă faptul că estomparea muchiilor se accentuează pe măsură ce crește valoarea abaterii standard.

Filtru Gaussian 5x5 sigma 0.5

Filtru Gaussian 5x5 sigma 2

Filtru Gaussian 11x11 sigma 1

Filtru Gaussian 11x11 sigma 10

În imagini. în multe situații, este prezent zgomot. Zgomotul este degradarea semnalului imagine. determinat de perturbații externe.Curățarea unei imagini afectate de zgomot face parte din categoria operațiilor de restaurare a imaginilor.

Cele mai frecvente tipuri de zgomote sunt: sare și piper, zgomot gaussian, zgomot tip speckle sau impuls și zgomotul periodic.

Zgomotul sare și piper se mai numește zgomot binar și este determinat de perturbații bruște ale semnalului, manifestându-se prin pixeli albi și negri, împrăștiați aleator în imagine.

Pentru a exemplifica aspectul acestui zgomot se poate utiliza comanda imnoise(imagine, tip, parametri). Rezultatul va fi o imagine, de același tip cu imaginea inițială, afectată de zgomot de tipul specificat. Parametri sunt opționali, având valori prestabilite asociate cu fiecare tip de zgomot.

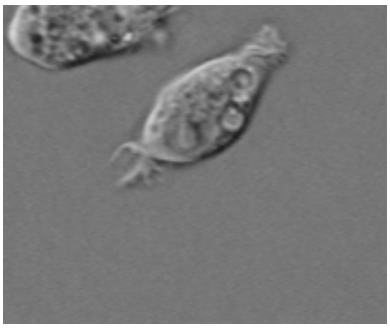
```
i2=imread('cell.tif');
i2_sp=imnoise(i2,'salt & pepper');
figure,imshow(i2),title('Imagine initiala')
figure,imshow(i2_sp),title('Zgomot salt & pepper')
```

În varianta implicită, cantitatea de zgomot aplicată este de 5%. Pentru a modifica această valoare se poate include un parametru, valoare subunitară, ce reprezintă fracția de pixeli corupți. Comanda:

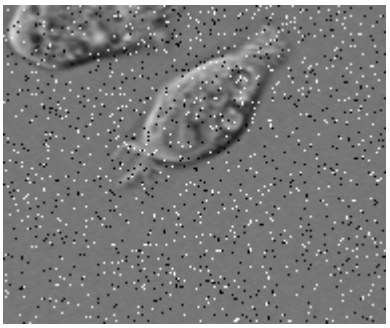
```
imnoise(i2,'salt & pepper',0.2);
```

determină zgomot ce afectează 20% din pixeli.

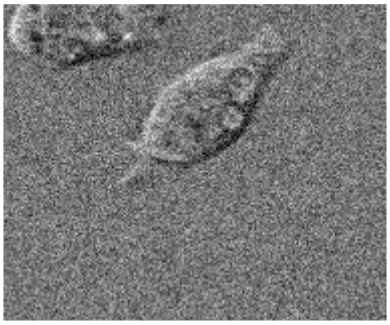
Imagine initiala



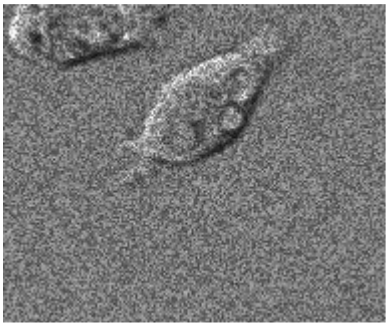
Zgomot salt & pepper



Zgomot gaussian



Zgomot tip speckle



Zgomotul gaussian este determinat de fluctuații aleatoare ale semnalului. Este un zgomot alb, distribuit normal. Efectul lui poate fi demonstrat cu :

```
i2_ga=imnoise(i2,'gaussian');
```

În variantă implicită zgomotul este distribuit normal, având media 0 și dispersia 0.01.

Zgomotul de tip speckle sau impuls este un zgomot multiplicativ, adică o valoare aleatore multiplică intensitatea pixelilor. Imaginea afectată de zgomot, I_spk, se obține:

I_spk = I + nI

unde I este imaginea inițială, iar n este o variabilă aleatoare distribuită uniform, cu medie 0 și varianță 0.04.

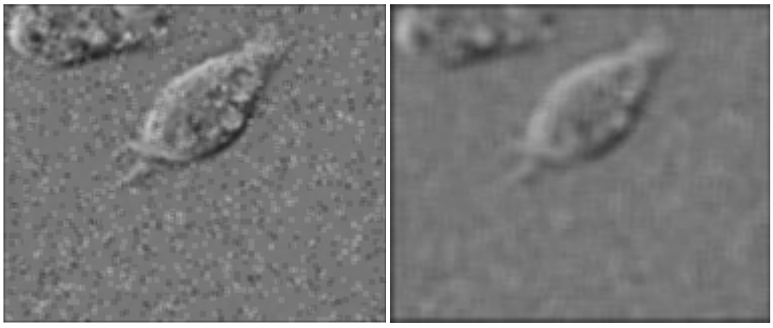
```
i2_spk=imnoise(i2,'speckle');
```

Deși, zgomotul gaussian și cel de tip speckle se manifestă în imagini într-un mod asemănător, ele se generează prin metode complet diferite și necesită metode diferite de eliminare.

Zgomotul periodic apare în cazul unei perturbații periodice și se manifestă prin apariția unor linii în imagine. Funcția imnoise nu generează un astfel de zgomot, dar el poate fi simulat prin adunarea unei matrice, care se obține pe baza unei funcții trigonometrice. Zgomotul de tip periodic se elimină din imagini prin transformări în domeniul frecvență.

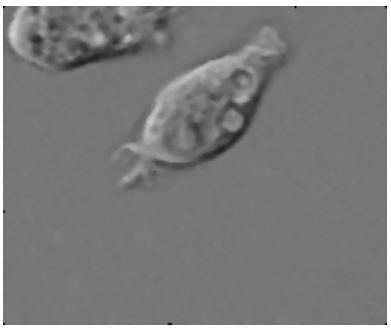
Celelalte tipuri de zgomote pot fi eliminate prin filtrare. Zgomotul de tip sare și piper, fiind componentă de frecvență înaltă, ar fi de așteptat să poată fi eliminat prin aplicarea unui filtru trece jos. Aplicarea unui filtru de mediere are efectul din fig. Se observă că zgomotul nu este îndepărtat, mai mult difuzează în vecinătăți. În cazul măririi filtrului, efectul se accentuează.

Filtru average 3x3 Filtru average 7x7



Un filtru mult mai potrivit este filtrul median. Mediana unui șir de valori este valoarea centrală a șirului de date ordonat. Mediana nu este afectată de valorile extreme ale șirului, care sunt, de fapt, pixelii afectați de zgomot. Deci, mediana va înlocui o valoare extremă (zgomot) cu o valoare apropiată de valorile înconjurătoare. Un filtru median se implementează în MATLAB cu funcția medfilt2(imagine, dimensiune nucleu).

Filtru median



Filtrul median lucrează bine și în cazul când imaginea este afectată de zgomot în procent mai mare. Pentru ca rezultatul să fie eficient, fie se mărește dimensiunea nucleului, fie se aplică de două ori succesiv.

Filtrul median este eficient în situaţia când un număr de pixeli lipsesc sau au valori extreme. Există două avantaje principale pe care le oferă filtrul median în comparaţie cu filtrarea medie cu factori de pondere [Russ]:

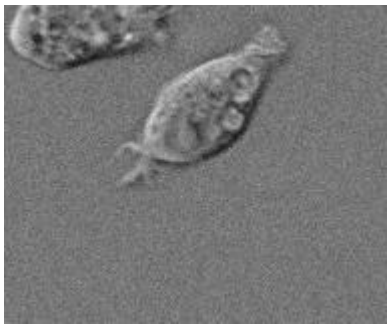
- metoda nu reduce diferenţele de intensitate ce pot apare;
- metoda nu deplasează muchiile.

Datorită acestor considerente, filtrarea mediană se preferă filtrării medii atât pentru inspecţii vizuale, cât şi pentru măsurări ale imaginii.

Zgomotul gaussian poate fi eliminat foarte simplu dacă există mai multe imagini, ale aceluiași obiect, afestate de acest tip de zgomot, ex. imagini prelevate din satelit, la mai multe treceri succesive sau în microscopie. În această situație medierea tuturor imaginilor se dovedește foarte eficientă. Pentru a exemplifica această metodă se generează 10 imagini afectate de zgomot gaussian, stocate într-o variabilă cu trei dimensiuni.

```
[n,m]=size(i2);
i_ga10=zeros(n,m,10);
for k=1:10;i_ga10(:,:,k)=imnoise(i2,'gaussian');end
i_ga_av=mean(i_ga10,3);%se face media dupa a treia dimensiune
figure, imshow(i_ga_av/255),title('Eliminare zgomot Gaussian
prin mediere')
```

Eliminare zgomot Gaussian prin mediere



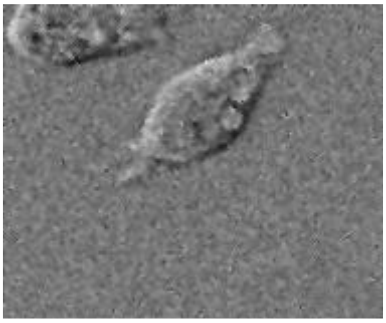
Eliminarea zgomotul gaussian se poate face și cu filtre Wiener. Ele se bazează pe metoda celor mai mici pătrate. Filtrul Wiener este un filtru adaptiv, el lucrând pe baza varianței locale din imagine. Dacă varianța este mare, filtrarea este slabă, iar zonele cu varianță mică sunt netezite mai puternic. În aceste condiții, acest filtru va menține muchiile și alte componente de frecvență ridicată, dar le estompează. Există mai multe variante de filtre Wiener, printre care este unul eficient pentru eliminarea zgomotului de tip gaussian. Filtrul lucrează plecând de la ipoteza unei reprtiții normale a zgomotului, cu medie 0. Valoarea de ieșire în urma filtrării se calculează:

$$g' = m_h + \frac{\sigma_h^2}{\sigma_h^2 + \sigma_g^2}(g - m_h)$$

unde m_h și σ^2_h este media și dispersia valorilor pixelilor aflate sub mască, σ^2_g este dispersia corespunzătoare întregii imagini, iar g este valoarea pixelului central. Aplicarea unui astfel de filtru se face cu funcția wiener2(imagine, dimensiune).

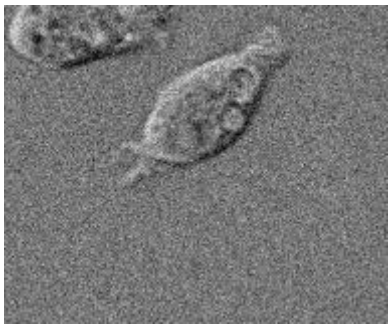
Filtrul este eficient mai ales dacă dispersia zgomotului este mică.

Filtru wiener 3x3

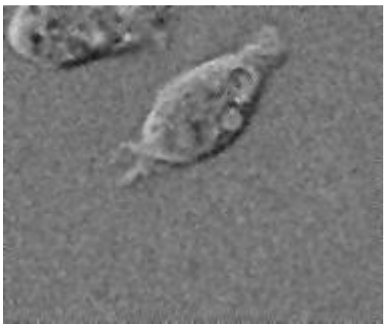


Filtru wiener 5x5

Zgomot gaussian cu sigma 0.005



Filtru wiener



5.2.2. Îmbunătăţirea imaginii

Prin îmbunătăţirea imaginii se urmăreşte accentuarea selectivă a unor caracteristici de interes, cum ar fi contrastul sau muchiile, simultan cu atenuarea unor caracteristici irelevante în raport cu scopul urmărit. Tehnicile de îmbunătăţire a imaginii nu conduc neaparat la o imagine mai fidelă, ci la una mai uşor de interpretat sau analizat. Principalele operaţii incluse în categoria tehnicilor de îmbunătăţire a imaginii sunt:

- modificarea contrastului şi a scării de gri;
- modificarea culorilor (şi pseudocromatizarea);
- eliminarea zgomotului;
- accentuarea contururilor;
- filtrarea imaginii.

Multe din tehnicile de îmbunătăţire a imaginii nu au la bază criterii riguros elaborate, alegerea lor bazându-se pe intuiţie şi experiment. Cu toate acestea este greu de conceput o aplicaţie în care acestea să fie complet eliminate.

Muchiile și contururile conțin informații importante despre imagini. Ele pot fi utilizate pentru măsurarea dimensiunilor unui obiect dintr-o imagine, pentru a izola un obiect de fundal, pentru a recunoaște sau clasifica un obiect. Există un număr mare de algoritmi pentru găsirea muchilor, din care se vor prezenta o parte.

Comanda generală pentru găsirea muchilor dintr-o imagine de intensitate sau binară este

edge(imagine, metoda, parametri)

une parametri depind de metoda utilizată, returnându-se o imagine binară cu valoarea 1 pentru muchii.

O muchie poate fi interpretată ca o succesiune de pixeli învecinați între care există diferențe semnificative.

50	49	51	54	50	53	148	151
53	49	53	61	51	54	150	160
49	51	52	57	52	53	151	172
55	50	49	55	51	52	149	158

Fie un grup de 4 x 4 pixeli conform fig. În blocul din dreapta se vede clar o diferență semnificativă între coloana a doua și a treia, deci apare o muchie verticală. Pentru a evidenția muchiile trebuie efectuate diferențe între valorile pixelilor învecinați, în acest fel se vor evidenția muchiile și se va estompa restul imaginii.

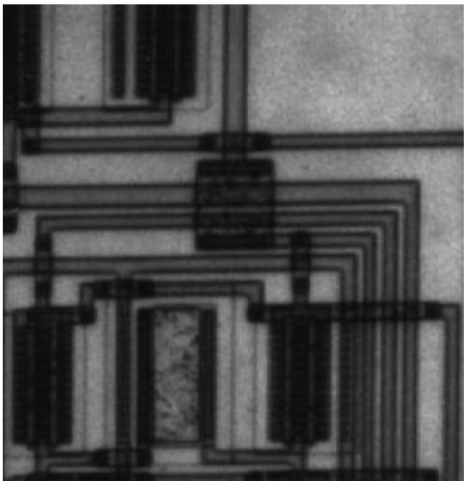
Pentru găsirea muchilor verticale se poate utiliza un filtru de forma p_x, iar pentru cele orizontale unul de forma p_y.

$$p_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

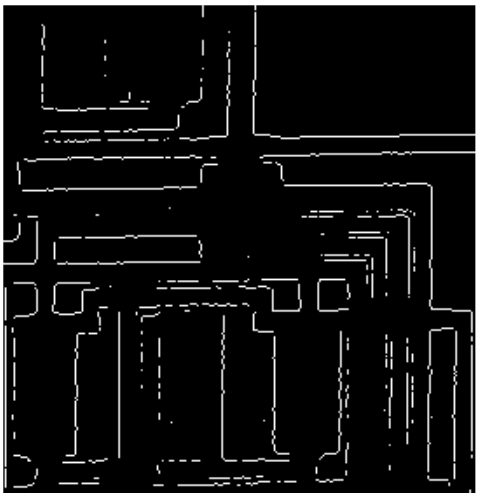
$$p_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Ele constituie filter Prewitt pentru detectarea muchiilor. Alte filtre pentru detectarea muchilor sunt Roberts, Sobel și Canny. În variant implicită se folosește metoda Sobel. Sobel, Prewitt și Roberts detectează muchiile pe baza gradientului maxim al imaginii, iar Canny caută maxim local pentru gradient.

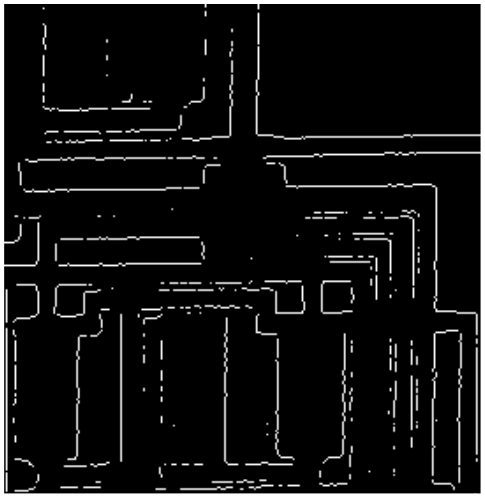
Imagine initiala



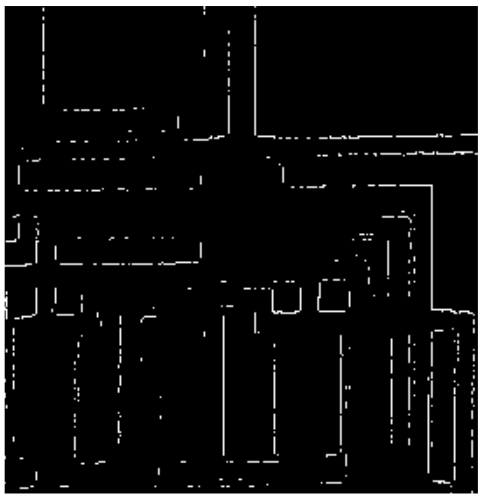
Filtru prewitt



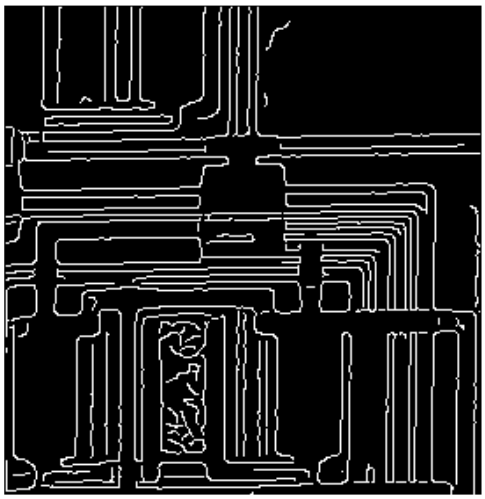
Filtru Sobel



Filtru Roberts



Filtru Canny



Cele mai eficiente s-au dovedit filtrul Sobel și Canny, ce lucrează corespunzător și în prezența zgomotului.

O a doua categorie de metode de detectare a muchilor sunt cele bazate pe diferențe de ordinul doi. Un filtru din această categorie este filtrul Laplacian. El este foarte sensibil la zgomot, dar detectează muchiile în orice direcție la fel de bine. Este un filtru trece sus, la fel cu filtrul Laplacian of Gaussian – log.

Imagine initiala

Filtru Laplacian

```
ii=imread('cameraman.tif');
i=im2double(ii);
fl=fspecial('laplacian');
ifl=filter2(fl,i);
figure,imshow(ifl),title('Filtru Laplacian')
```

Generarea filtrelor de tip Laplacian se fac în MATLAB cu funcția fspecial, având sintaxa:

`fspecial('laplacian',alfa),`

care va genera un nucleu de forma:

$$\frac{1}{\alpha+1}\begin{bmatrix} \alpha & 1-\alpha & \alpha \\ 1-\alpha & -4 & 1-\alpha \\ \alpha & 1-\alpha & \alpha \end{bmatrix}$$

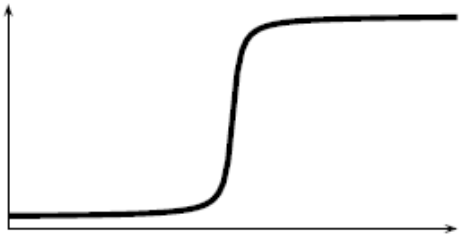
Valoarea implicită pentru alfa este 0.2.

După detectarea muchiilor, de multe ori este necesară o operație de îmbunătățire a acestora, adică de a le face cât mai clare și continue. Pentru accentuarea contururilor se poate scădea o imagine estompată și scalată a imaginii din imaginea inițială. Efectul acestei operații se vede în reprezentarea următoare.

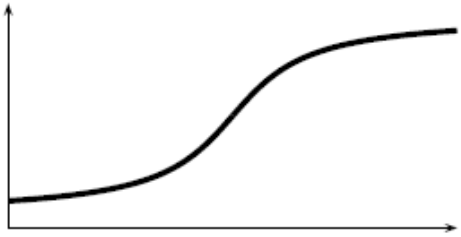
Un astfel de filtru se poate obține cu funcția fspecial. având parametrul 'unsharp'). Nucleul este de forma:

$$\frac{1}{\alpha+1}\begin{bmatrix} -\alpha & \alpha-1 & -\alpha \\ \alpha-1 & \alpha+5 & \alpha-1 \\ -\alpha & \alpha-1 & -\alpha \end{bmatrix}$$

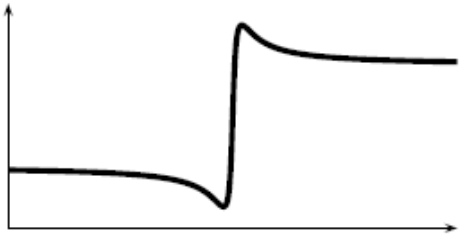
având valoarea implicită 0.2



(a) Pixel values over an edge



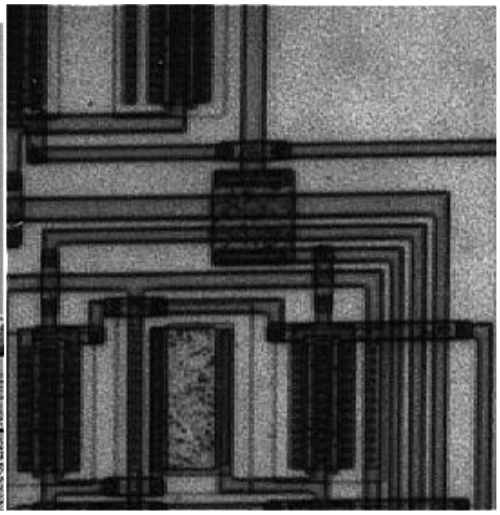
(b) The edge blurred



(c) (a) – k(b)

Filtru unsharp

Filtru unsharp



```
u=fspecial('unsharp');
iu=filter2(u,i);
figure,imshow(iu/255),title('Filtru unsharp')
j=imread('circuit.tif');
ju=filter2(u,j);
figure,imshow(ju/255),title('Filtru unsharp')
```

În concluzie se poate afirma că există numeroase metode de detectare a muchiilor. Alegerea metodei depimde de natura imaginii, zgomotul prezent și se utilizarea ulterioară a muchiilor.

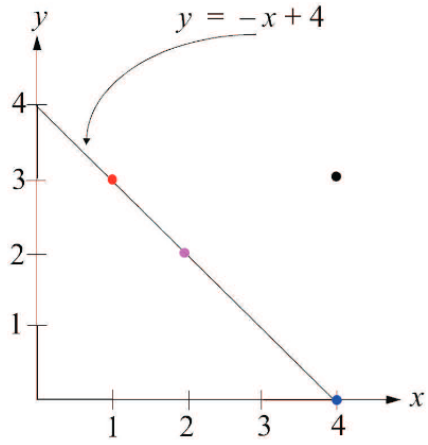
În condiții ideale, metodele prezentate anterior ar genera doar pixeli ce aparțin muchiilor. În practică, rezultatele obținute nu reprezintă complet muchiile datorită zgomotului, iluminării neuniforme și altor efecte perturbatoare, care determină apariția unor întreruperi în muchii, precum și apariția unor pixeli ce nu aparțin acestora. Din acest motiv algoritmii de detectare a muchilor sunt urmați de proceduri ce asigură unificarea muchiilor aparținătoare aceluiași obiect. O metodă ce permite unirea unor segmente aparținătoare unei muchii este transformata Hough. Transformata Hough permite detectarea de forme geometrice într-o imagine: linii, cercuri, elipse etc. Pentru imagini binare, în MATLAB, transformata se poate aplica

```
[H,ro,teta]=hough(BW);
```

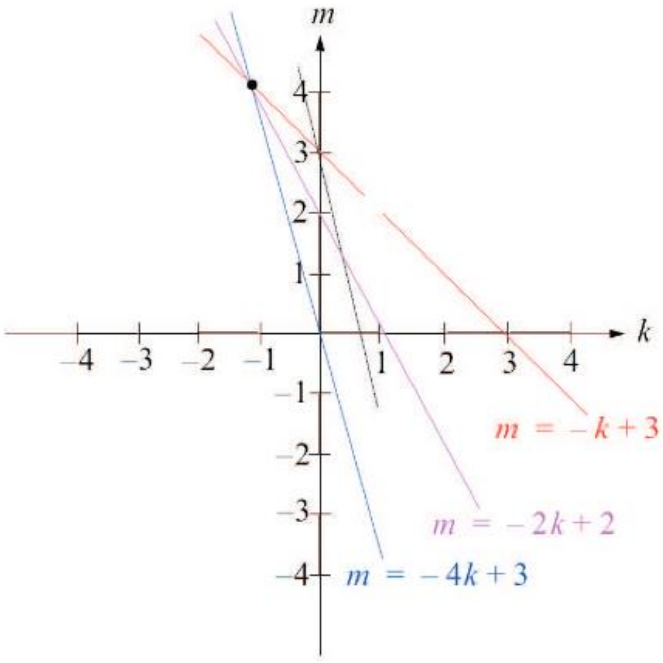
Transformarea utilizează reprezentarea parametrică a dreptei (ro=xcos(teta)+ysin(teta)), ro fiind distanța de la origine la dreaptă, iar teta este unghiul format intre axa 0x și normala la dreaptă prin origine.

Principiul transformatei Hough este unul foarte simplu. Fie un punct (x0, y0) într-o imagine binară. Prin acest punct trec o infinitate de drepte care îndeplinesc ecuația y0 = ax0 + b. În planul ab se poate reprezenta dreapta b = y0 – ax0, adică unui punct din planul xy îi corespunde o dreaptă în planul ab. Dacă se consideră al doilea punct (x1, y1) cu dreapta b = y1 - ax1, cele două drepte se intersectează în planul ab într-un punct de coordonate (a’, b’), a’ fiind panta dreptei ce unește punctele (x0, y0), respectiv intersecția în origine a adreptei din planul xy. Cu alte cuvinte, toate punctele coliniare cu dreapta ce trece prin punctele (x0, y0), (x1, y1) vor trece prin punctul (a’, b’) în planul ab. În aceste condiții, dacă se determină punctele cu număr mare de intersecții din planul ab, se pot depista dreptele din planul inițial.

Pentru exemplificare se ia un caz cu trei puncte coliniare și unul necoliniar.

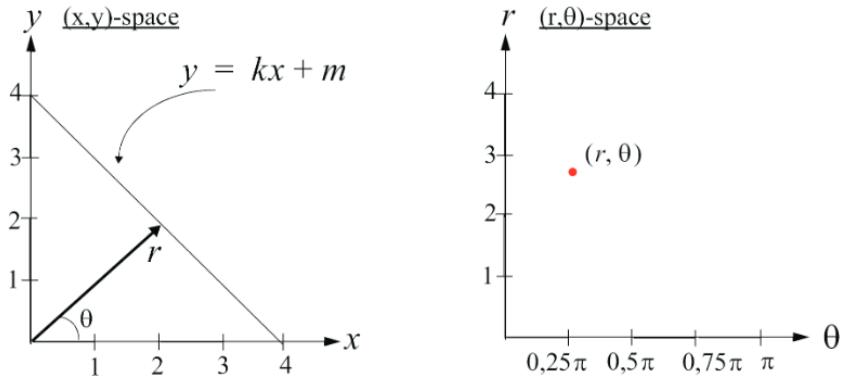


y	x
3	1
2	2
0	4
3	4



Pentru fiecare din punctele (1, 3), (2, 2), (4, 0) și (4, 3) se construiesc dreptele în planul parametrilor. Se constată că apar trei intersecții în punctul (-1, 4), ecuația dreptei din planul xy fiind y = -x + 4.

Această implementare a transformatei Hough ridică problema liniilor verticale, ce au panta infinită. Pentru a putea reprezenta și linile verticale se trece la reprezentarea dreptei în coordonate polare.



Unei dreapte din planul xy îi corespunde un punct în planul θr. Ecuația dreptei poate fi exprimată sub forma:

$xcos\theta + ysin\theta = r$

Transformata Hough poate fi implementată în felul următor, se alege un set de valori r și θ. Pentru fiecare pixel (x, y) din imagine se calculează expresia xcosθ +ysinθ =r, luând în calcul toate valorile lui θ. Valorile se stochează într-o matrice de acumulare. Din această matrice se extrag valorile mari, pe baza cărora se determină dreptele din imaginea inițială. Transformata Hough poate fi adaptată și la determinarea altor forme, de ex. cercuri sau elipse, dar cu un efort de calcul mai mare.

În multe aplicații de prelucrarea imaginilor este necesar să se calculeze distanța de la un pixel la o anumită regiune, R. Calcularea distanței euclidienese face cu formula uzuală:

$$d(i,j) = \min_{p,q \in R} \sqrt{(i-p)^2 + (j-q)^2}$$

Metoda este ineficientă, mai ales dacă regiunea R este mare. O posibilitate mult mai eficientă este utilizarea transformatei distanței. Transformata distanței pentru o matrice binară este o matrice, ce conține valori numerice ce reprezintă distanța minimă până la un pixel egal cu 1. Ilustrarea transformatei se face în figură, pentru o imagine binară de dimensiuni mici.

1	1	0	0	0	0	0	1.00	2.00	3.00
1	1	0	0	0	0	0	1.00	2.00	3.00
0	0	0	0	0	1.00	1.00	1.41	2.00	2.24
0	0	0	0	0	1.41	1.00	1.00	1.00	1.41
0	1	1	1	0	1.00	0	0	0	1.00

În MATLAB transformata distanței, pentru imagini binare, se calculează cu funcția bwdist(imagine). Există mai multe metode de calculare a distanței, varianta implicită fiind distanța euclideană

6.2.3. Prelucrarea imaginilor în domeniul frecvenţă

Prelucrarea imaginilor în domeniul frecvenţă se utilizează pentru măsurări şi anumite prelucrări. Multe dintre aceste operaţii pot fi realizate şi în domeniul spaţiu (cel original), dar presupun un efort de calcul considerabil mai mare.

Cea mai cunoscută metodă de transformare este transformarea Fourier, datorită algoritmului foarte eficient dezvoltat de Cooley şi Bracewell în 1965, cunoscut sub numele de transformarea Fourier rapidă (FFT).

Fie o funcţie f(x), x variabilă reală, reprezentând timpul sau distanţa pe o direcţie din imagine. Această funcţie se consideră o funcţie în domeniul timp sau în domeniul spaţiu, iar transformata F este funcţia în domeniul frecvenţă. Conform teoremei Fourier între cele două funcţii există relaţia :

$$\mathrm{F(u)} = \int_{-\infty}^{\infty} \mathrm{f(x)} e^{-2\pi \mathrm{iux}} \mathrm{dx}\,, \tag{6.6}$$

unde

$$e^{-2\pi \mathrm{iux}} = cos(2\pi \mathrm{ux}) - \mathrm{i}\, sin(2\pi \mathrm{ux}). \tag{6.7}$$

O caracteristică foarte importantă a acestei transformări este că se poate reconstitui funcţia f(x) din F(u):

$$f(x) = \int_{-\infty}^{\infty} F(u) e^{2\pi i u x} du\,. \tag{6.8}$$

Relaţia (6.6) şi (6.8) reprezintă transformata Fourier directă şi inversă. În general f(x) este o funcţie reală, de exemplu: variaţia tensiunii în raport cu timpul sau în raport cu spaţiul. Transformata F(u) este o funcţie complexă, ce se mai poate scrie sub forma:

$$F(u) = R(u) + iI(u)\,, \tag{6.9}$$

dar este mai convenebilă exprimarea în coordonate polare:

$$F(u) = |F(u)| e^{i\Phi(u)}\,, \tag{6.10}$$

unde |F| este amplitudinea şi Φ este faza. Pătratul amplitudinii, $P(u) = |F(u)|^2$ se numeşte densitate spectrală a lui f(x).

În practică, integrala pe tot domeniul real se înlocuieşte cu o sumă de termeni cu frecvenţă crescătoare, limitată de spaţiul finit al eşantionului de puncte din imagine. Transformata Fourier discretă se scrie:

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) e^{-i2\pi ux/N} , \qquad (6.11)$$

unde N este numărul de puncte spaţiate echidistant. Transformata inversă este similară:

$$f(x) = \sum_{u=0}^{N-1} F(u) e^{i2\pi ux/N} . \qquad (6.12)$$

Valorile u reprezintă frecvenţele componentelor discrete ce se însumează pentru reconstituirea funcţiei f(x). În mod normal însumarea se face pe o jumătate din dimensiunea imaginii (în pixeli), această limitare este descrisă ca frecvenţă Nyquist. Datorită faptului că însumarea are un număr de termeni egal cu jumătate din numărul punctelor din eşantion, dar fiecare termen are parte reală şi imaginară, numărul total de valori produse prin transformarea Fourier este egală cu numărul pixelilor din imagine sau cu numărul punctelor din eşantion.

Atât în cazul transformării continue, cât şi discrete, extinderea de la funcţii unidimensionale la bi- sau tridimensionale poate fi făcută prin înlocuirea lui f(x) cu f(x, y) şi F(u) cu F(u, v), iar suma, respectiv integrala se face după două variabile. Deoarece direcţiile x, y, z sunt ortogonale, la fel sunt şi dimensiunile u, v, w. Acest lucru înseamnă că transformarea poate fi realizată separat pe fiecare direcţie. De exemplu, la o imagine bidimensională se poate face o transformare pe fiecare linie, obţinând un rezultat intermediar cu valori complexe şi apoi, o a doua serie de transformări pe fiecare coloană, rezultând în final transformata bidimensională. Această aplicare a transformării unidimensionale pe linii, respectiv pe coloane, nu este cea mai rapidă metodă de calcul, dar este de departe cea mai simplă şi este utilizată de multe ori în aplicaţii.

Urmare a transformării imaginii originale în domeniul frecvenţă apar valori complexe şi rezultatul este dificil de reprezentat grafic într-o manieră intuitivă. În majoritatea cazurilor, reprezentarea se bazează pe amplitudinile valorilor şi se neglijează fazele. Prin reprezentarea densităţii spectrale se poate izola structura periodică a zgomotului.

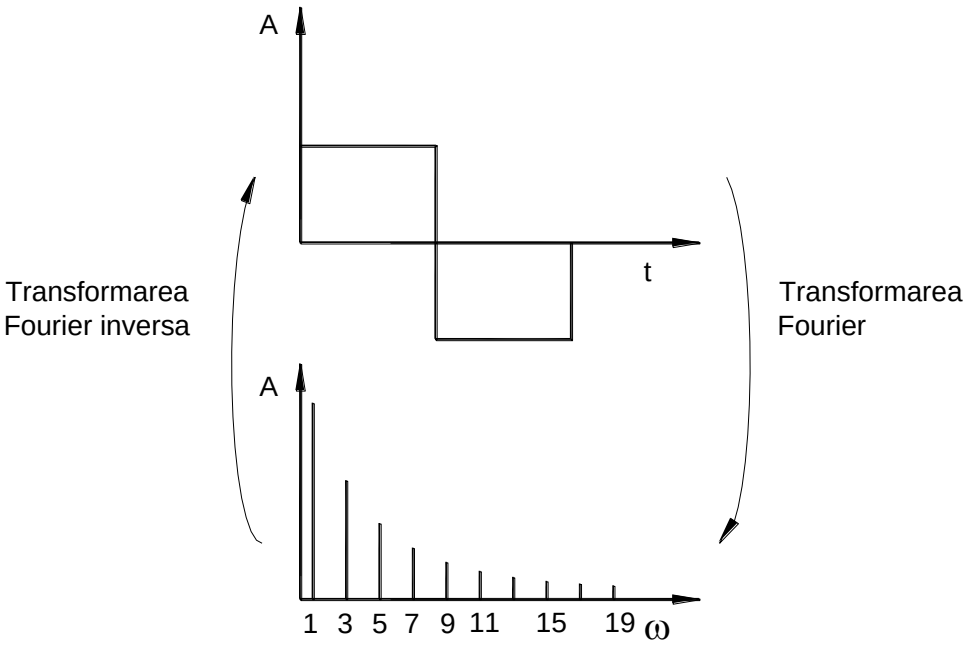


Fig. 6.15. Legătura dintre transformarea Fourier directă şi inversă

În fig. 6.15 se prezintă legătura dintre cele două domenii, spaţiu şi frecvenţă. Este important să se reţină faptul că termenii de frecvenţă joasă (în apropierea originii) generează forma generală a funcţiei, iar termenii de frecvenţă ridicată sunt necesari pentru obţinerea demarcaţilor clare şi a detaliilor fine.

O altă observaţie importantă este că termenii sunt independenţi unul faţă de celălalt, adică făcând transformarea pentru determinarea coeficienţilor de frecvenţe înalte, nu se modifică termenii anteriori.

În fig. 6.16 se prezintă două imagini în domeniul spaţiu, împreună cu spectrele de frecvenţă corespunzătoare. În a treia figură se prezintă imaginea obţinută prin însumarea celor două imagini iniţiale şi spectrul de frecvenţă obţinut prin însumarea spectrelor imaginilor iniţiale. Acest spectru este identic cu spectrul obţinut din cea de-a treia imagine. Se poate constata că transformarea în domeniul frecvenţă are o proprietate de separabilitate şi aditivitate. Adunarea celor două spectre de frecvenţă produce acelaşi rezultat cu transformata sumei imaginiilor.

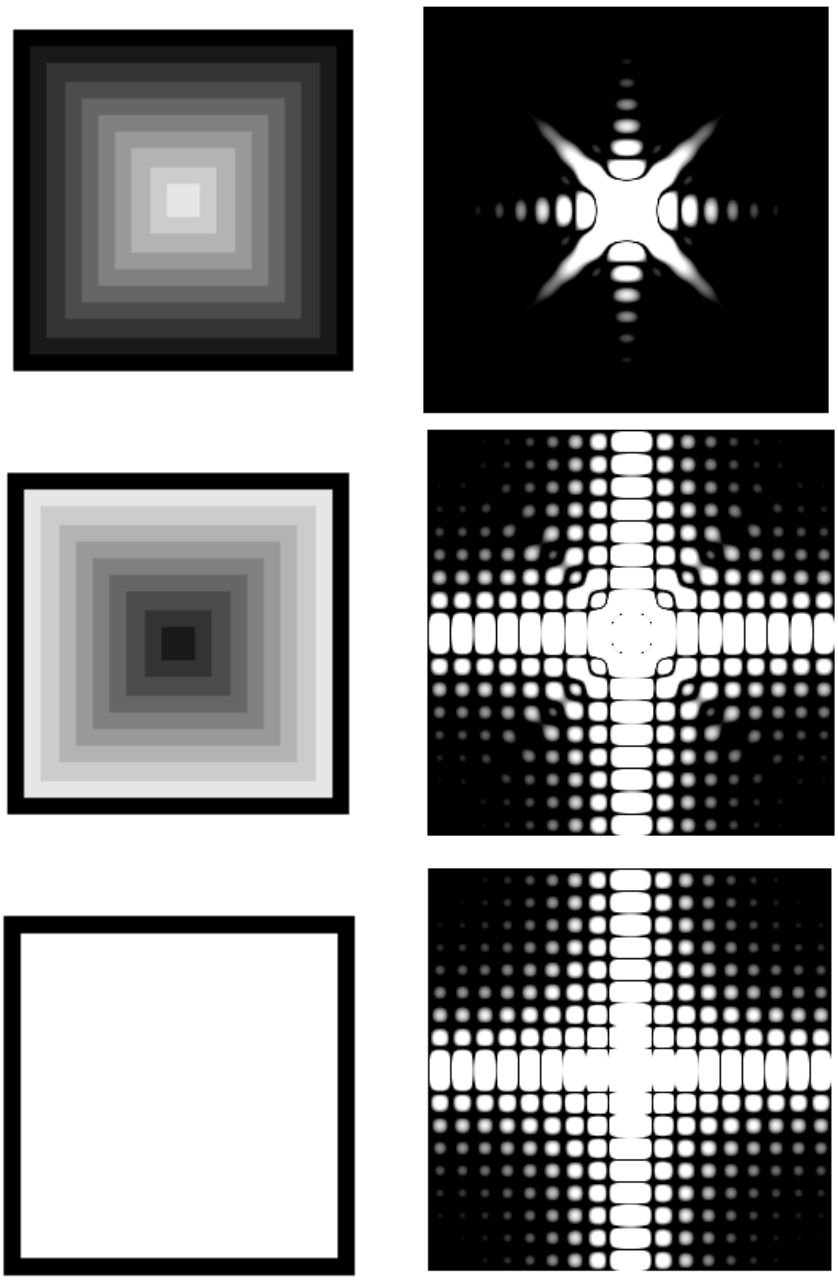


Fig. 6.16 Proprietatea de aditivitate a spectrelor de frecvenţă a şî b – imagini cu spectrele de frecvenţă asociate; c – imagine rezultată prin însumarea celor două imagini iniţiale, spectrul imaginii rezultante şi spectrul obţinut prin însumarea spectrelor din a şi b

Această observaţie generează ideea de a utiliza scăderea pentru a înlătura zgomotul periodic, ce poate fi introdus de echipamentele utilizate la înregistrarea

sau transmiterea imaginii. Dacă imaginea zgomotului sau transformata Fourier a acestuia poate fi determinată, prin scăderea transformatei din cea a imaginii iniţiale (cu zgomot) se obţine spectrul imaginii cu zgomotul înlăturat, iar apoi se poate reveni în domeniul spaţiu pentru obţinerea imaginii iniţiale.

Funcţiile din Matlab pentru calcularea transformatei Fourier sunt fft, fft2, fftn, care calculează transformata Fourier rapidă pentru funcţii uni-, bi- şi n dimensionale, iar funcţiile ifft, ifft2, ifftn calculează transformarea Fourier inversă.

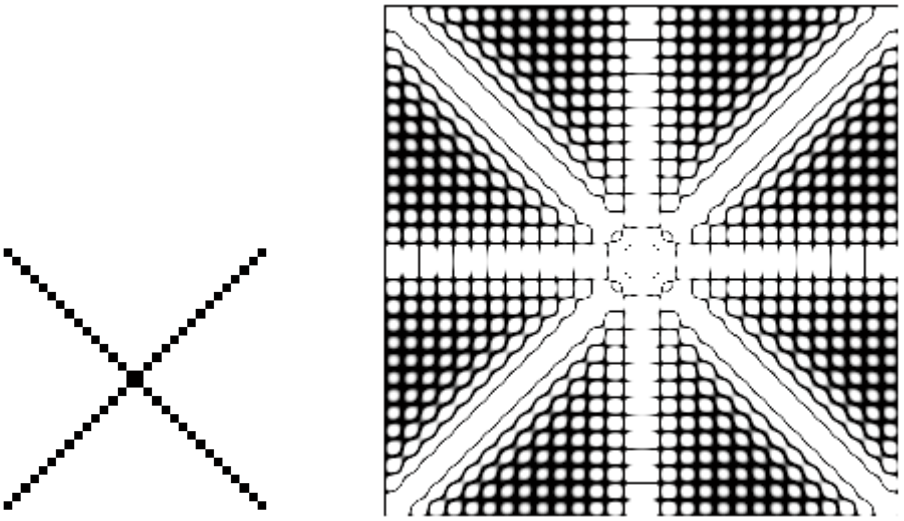


Fig. 6.17. Imagine cu spectrul de frecvenţe corespunzător

În continuare se prezintă un exemplu de aplicaţie care construieşte spectrul de frecvenţe al imaginii din fig. 6.17:

```
f=ones(30,30);
for ii=1:30
    f(ii,ii)=0;
    f(30+1-ii,ii)=0;
end
imshow(f,'truesize')
F=fft2(f,256,256);
F2=log(abs(F));
```

La vizualizarea coeficienţilor, coeficientul fundamental este reprezentat în colţul din stânga sus. Pentru a-l reprezenta în centru se introduce secvenţa:

```
F=fft2(f,256,256);
F2=fftshift(F);
figure,imshow(log(abs(F2)))
```

Prin aplicarea transformării Fourier răspunsului la un semnal impuls

aplicat unui filtru, se obţine răspunsul în frecvenţă al filtrului. Funcţia freqz2 calculează şi afişează răspunsul în frecvenţă. De exemplu, răspunsul în frecvenţă la un nucleu de convoluţie de tip Gauss scoate în evidenţă faptul că acest filtru permite trecerea frecvenţelor joase şi atenuează frecvenţele înalte (vezi fig. 6. 18).

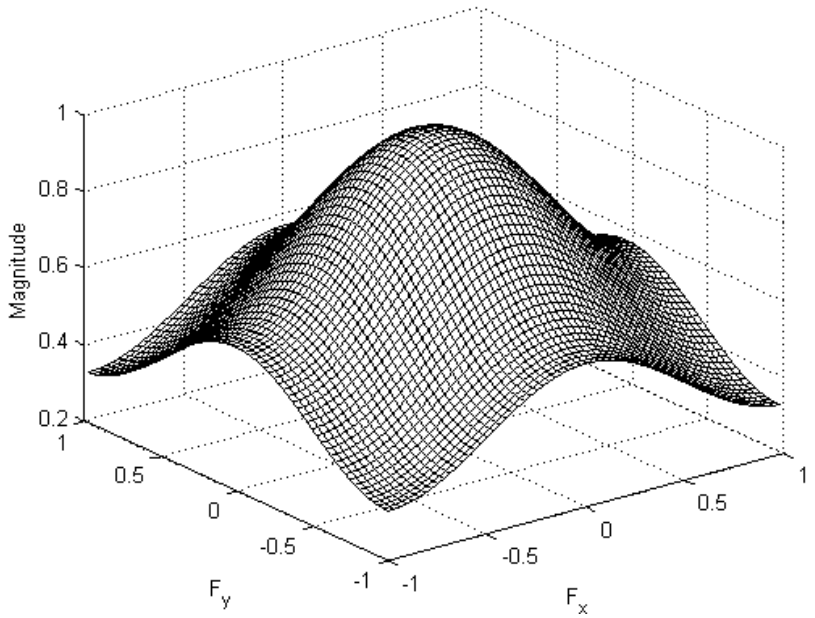


Fig. 6.18. Răspunsul în frecvenţă al unui filtru de tip Gauss

O proprietate foarte importantă a transformării Fourier este faptul că multiplicarea a două transformări corespunde convoluţiei funcţiilor spaţiale asociate. Din acest motiv este mai avantajos în cazul nucleelor de dimensiuni mari să se înlocuiască procesul de convoluţie cu prelucrarea în domeniul frecvenţă. Această echivalare a multiplicării din domeniul frecvenţă cu convoluţia în domeniul spaţiului este valabilă în cazul filtrelor liniare şi multiplicative, [Russ].

În domeniul frecvenţă pot fi atenuate o serie de defecte, cum ar fi neclarităţile din imagine datorate mişcării. Aceste probleme apar la achiziţionarea unor imagini cu niveluri de luminozitate scăzută, la care timpul de expunere trebuie să fie suficient de lung, astfel încât pot apărea deplasări ale camerei în raport cu zona vizualizată. În majoritatea cazurilor, mărimea şi orientarea vitezei este cunoscută. Acest lucru permite parcurgerea următoarelor etape:

- construirea transformatei Fourier a imaginii iniţiale;
- reprezentarea vectorului viteză şi a transformatei asociate;

- împărţirea celor două transformate (a imaginii la cea a vectorului);
- reconstruirea imaginii pe baza transformatei rezultante.

Prelucrarea imaginilor în domeniul frecvenţă este utilă la eliminarea anumitor tipuri de zgomote, aplicarea nucleelor de convoluţie mari etc. Majoritatea operaţiilor pot fi efectuate şi în domeniul spaţiu, dar presupun un efort de calcul mult mai mare.

6.2.4. Segmentarea imaginilor

O etapă de mare importanţă în procesul de obţinerea informaţiei din imagine este segmentarea, adică împărţirea imaginii în regiuni ce corespund anumitor entităţi structurale sau obiecte de interes. Segmentarea este deseori descrisă prin analogie cu procesul vizual de separare a fundalului de restul imaginii.

Algoritmii de segmentare pentru imagini monocrome, în general, se bazează pe două caracteristici ale intensității: discontinuități și similitudini. Discontinuitățiile permit determinarea contururilor diferitelor obiecte din imagine, iar similitudinile permit detectarea regiunilor aparţinătoare obiectelor.

Segmentarea este o operaţie dependentă de aplicaţie, segmentarea automată reprezentând una din problemele cele mai dificile în analiza imaginii. Fragmentele de imagine separate prin segmentare se denumesc regiuni. O regiune extrasă poate fi caracterizată, la modul general, prin omogenitatea unei anumite proprietăţi sau caracteristici în interiorul ei. În general, se consideră proprietăţi locale, evaluate în ferestre de anumite dimensiuni. O problemă specifică segmentării este aceea că ferestrele, în care se estimează proprietăţile ce stau la baza segmentării, pot include pixeli din regiuni diferite. Prin urmare, proprietăţile estimate sunt afectate de erori la graniţele dintre regiuni.

Optimizarea procesului de segmentare presupune găsirea unui compromis între două cerinţe contradictorii:

- regiunile generate trebuie să fie cât mai puţine şi cu forme cât mai simple;
- regiunile trebuie să fie cât mai omogene.

Specificarea regiunilor se poate face în mai multe moduri. Este frecvent utilizată etichetarea imaginii (fiecare pixel aparţinând unei anumite regiuni are un acelaşi număr de ordine, iar numărul maxim de ordine ce apare într-o imagine reprezintă numărul de regiuni identificate). Pentru afişarea rezultatelor segmentării este mai intuitivă reprezentarea fiecărei regiuni cu nivelul mediu de gri.

Definirea şi evaluarea obiectivă a unor proprietăţi reprezentative ale imaginii joacă un rol important în analiza imaginii şi este cu atât mai complicată

cu cât informaţiile apriorice referitoare la conţinutul iamginii sunt mai puţine. Proprietăţile imaginii pot fi clasificate, în funcţie de extinderea zonei din imagine folosită pentru evaluare, în trei categorii:

- proprietăţi locale;
- proprietăţi regionale;
- proprietăţi globale.

O proprietate locală se evaluează în vecinătatea unui pixel din imagine. Regiunea reprezintă o suprafaţă mult mai mică decât imaginea completă. O proprietate globală se evaluează cu ajutorul tuturor pixelilor din imagine. Proprietăţile regionale sunt măsurate în regiuni ale imaginii, extrase printr-un proces de segmentare şi sunt denumite curent descriptori de imagine. Proprietăţile locale şi globale sunt determinate, în general, în scopul segmentării imaginii. Ca o regulă generală, segmentarea se bazează pe proprietăţi locale, în vederea delimitării regiunilor omogene, iar proprietăţile globale sunt utilizate la adaptarea procesului de segmentare la caracteristicile imaginii.

Printre proprietăţile globale se menţionează histograma. Trebuie menţionat că histogramele sunt, uneori, evaluate local, pentru ferestre de diferite dimensiuni. Histograma poate fi utilizată la evaluarea unor parametri globali ai imaginii, cum ar fi media nivelurilor de gri, modulul, dispersia, momentele centrate de diferite ordine. O altă proprietate globală este proiecţia orizontală sau profilul vertical (media intensităţiilor pe linii):

$$p_m = \frac{1}{N}\sum_{n=1}^{N} f\left(m,n\right). \tag{6.13}$$

Similar se defineşte proiecţia verticală sau profilul orizontal (media intensităţiilor pe coloane):

$$p_n = \frac{1}{N}\sum_{m=1}^{N} f\left(m,n\right). \tag{6.14}$$

Proiecţia pe o direcţie oarecare, ce face unghiul θ cu orizontala, se obţine prin rotirea imaginii cu unghiul θ şi calcularea proiecţiei orizontale.

Printre proprietăţile locale cele mai reprezentative se numără nivelul de gri, muchiile şi texturile.

Selectarea caracteristicilor dintr-o imagine este o etapă importantă pentru majoritatea măsurărilor sau interpretării unei imagini. Cea mai simplă modalitate este definirea unui domeniu de intensităţi din imaginea originală, selectarea pixelilor aparţinători acestui domeniu ca reprezentând elementele imaginii şi eliminarea celorlalţi, care se consideră fundal. Această operaţie se numeşte fixarea pragului şi poate fi făcută pe baza histogramei imaginii. Vârfurile din histogramă, de multe ori, identifică zone omogene şi domeniul poate fi stabilit

între vârfuri.

În unele cazuri, segmentarea poate fi realizată utilizând mai multe imagini ale aceluiaşi plan obiect, de exemplu prelevate cu diferite lungimi de undă ale luminii incidente. În cazul imaginilor preluate din satelit, se pot selecta regiunile în funcţie de vegetaţie, tip de zăcăminte etc. Cu cât există mai multe benzi de culoare, cu atât mai uşoară este segmentarea, deoarece punctele ce nu pot fi distinse în una din imagini, devin vizibile în alta. Totuşi, în cazul imaginilor multispectrale poate deveni dificilă specificarea criteriului de selecţie. O variantă logică este limitarea luminozităţii prin specificarea domeniului culorilor fundamentale, R, G, B. Acest criteriu se combină printr-o funcţie Şi logic.

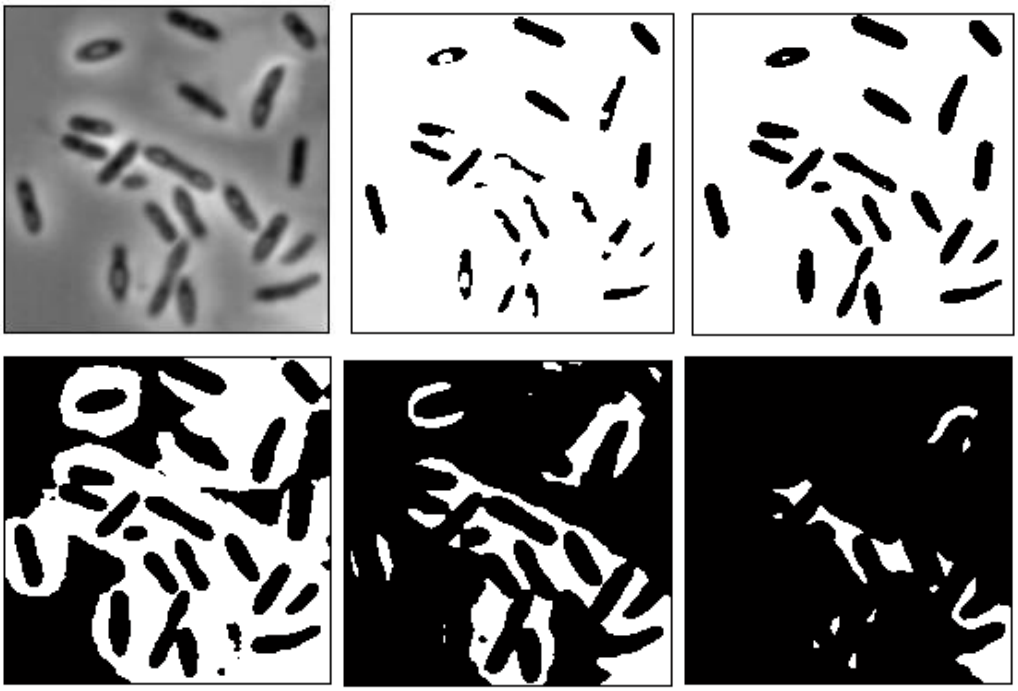


Fig. 6.19. Segmentarea unei imagini cu nivel de prag cuprins între 0,3 -0,7 cu pas 0,1

În cazul unei imagini monocrome, problema este mult mai simplă şi un exemplu de segmentare poate fi vizualizat în fig. 6.19. Imaginile sunt obţinute cu secvenţa de comenzi:

```
I=imread('bacteria.tif');
imshow(I)
for i=.3:.1:.7
    T=im2bw(I,i);
    figure,imshow(T)
end
```

Obţinerea imaginii segmentate se poate face mai simplu cu comanda:

`T=I>nivel`, în urma căreia se generează o matrice de aceleaşi dimensiuni cu matricea I, având valoarea logică 1 în toate locaţiile ce depăşesc valoarea variabilei **`nivel`**. O posibilitate automată de determinare a nivelului de gri corespunzător segmentării este apelarea comenzii **`nivel=graythresh(I).`** În acest caz valoarea variabilei nivel se determină astfel încât să se minimizeze varianţa între pixelii cu valoarea 1 şi 0.

Muchiile sunt o caracteristică locală importantă deoarece concentrează informaţia referitoare la forma obiectului. Muchiile sunt regiuni din imagine unde intensitatea se schimbă brusc. Trebuie făcută o distincţia între muchii şi contururi. Conturul desemnează punctele din imagine ce delimitează obiecte sau părţi de sine stătătoare ale unui obiect. Contururile sunt curbe ce pot fi reprezentate prin imagini binare, în timp ce muchiile sunt structuri bidimensionale.

Dificultatea detectării muchiilor se datorează următorilor factori: diversitatea tipurilor de muchii existente în imaginile reale, orientarea şi factorii de scară variabili, posibilitatea prezenţei unor structuri diverse în vecinătatea muchiilor, zgomotul etc.

Detecţia muchiilor poate fi privită ca o modalitate de segmentare bazată pe detecţia discontinuităţii din imagine. Un avantej al acestei metode este faptul că variaţiile globale lente ale intensităţii nu modifică rezultatul segmentării, iar ca dezavantaj trebuie menţionată sensibilitatea la zgomot şi faptul că metoda nu conduce totdeauna la detecţia unor contururi închise, pe baza cărora să fie posibilă delimitarea unor regiuni.

Un tip de linie continuă din imagine ce poate furniza informaţii este o linie de contur. Ea este analogă curbei de nivel dintr-o hartă topografică, cu observaţia că în imagine ea se caracterizează prin intensitate constantă. Contururile sunt curbe închise, ce se pot ramifica. Pentru o imagine, care este discretizată, valoarea luminozităţii poate să nu corespundă valorii unui anumit pixel, dar trebuie să existe o pereche de pixeli cu o valoare apropiată (unul cu o valoare mai mare şi celălalt cu cu o valoare mai mică) de intensitatea conturului, astfel încât conturul trece printre ei. În practică cea mai uzuală metodă de detecţie a conturului este marcarea pixelilor ce au valori ale intensităţilor apropiate de o valoare dată.

Există şi alte metode utilizate la segmentarea imaginilor în afara segmentării pe baza stabilirii nivelului de prag. Aceste metode se asociază cu sisteme de calcul mult mai puternice şi prelucrează imaginea în sensul vederii artificiale. Două din cele mai utilizate metode sunt creşterea regiunilor şi divizarea – şi – fuziunea regiunilor.

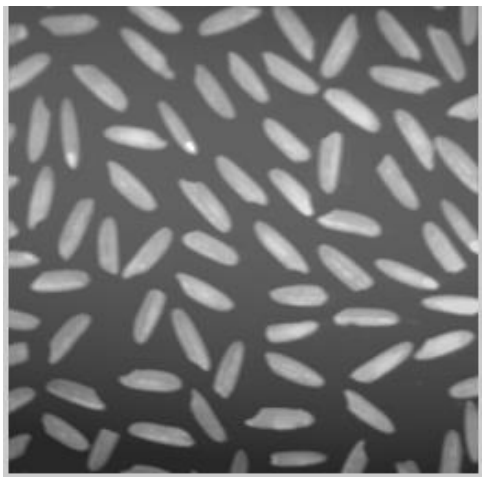
Divizarea şi fuziunea regiunilor este o metodă ce începe cu întreaga imagine. O anumită proprietate a imaginii este selectată drept criteriu de uniformitate, frecvent fiind utilizaţi anumiţi parametri statistici ai histogramei.

Dacă histograma este multimodală sau are deviaţia standard mare, atunci regiunea se presupune neuniformă şi este împărţită în patru blocuri. Fiecare bloc este examinat în acelaşi mod şi subîmpărţit dacă este necesar. Procedeul continuă până la împărţirea completă a imaginii.

Simpla divizare a regiunii nu creează o segmentare corespunzătoare. după fiecare iteraţie de divizare, noile regiuni sunt comparate cu cele adiacente. Dacă sunt similare, ele fuzionează. Decizia de stabilire a similitudinii se face în baza aceloraşi criteri utilizate la divizare.

Un avantaj al acestei metode este faptul că o segmentare completă este realizată cu un număr redus de iteraţii. De exemplu, o imagine având dimensiunea 512 x 512 necesită 9 iteraţii pentru a ajunge la nivelul unui pixel ($2^9 = 512$). Prin comparaţie, metoda de detecţie cu prag, în mod uzual, identifică un sigur tip de regiune. Ea trebuie aplicată de mai multe ori pentru a prelucra imagini conţinând mai multe clase de obiecte. Pe de lată parte, metoda divizării şi fuziunii regiunilor depinde de testarea neomogenităţilor unei regiuni. Subregiuni de dimensiuni mici, aparţinând unor regiuni de dimensiuni mari, pot să nu fie sesizate cu această metodă. Un alt avantaj al metodei de detecţie cu prag este faptul că pot fi identificate obiecte similare în diferite locaţii ale imaginii, ceea ce nu poate fi făcut cu metoda divizării şi fuziunii regiunilor.

Creşterea regiunilor este o metodă ce pleacă de la pixeli individuali. În general se pleacă de la o locaţie furnizată de utilizator şi sunt examinaţi pixelii din vecinătate pentru a vedea dacă pot fi ataşaţi regiunii. După ce regiunea nu mai poate fi mărită, procesul începe într-o altă locaţie. În cazul când criteriul de testare este acelaşi, prin aplicarea metodei creşterii regiunilor sau a metodei divizării şi fuziunii regiunilor, se obţine acelaşi rezultat. Dificultatea în metoda creşterii regiunii constă în faptul că punctul de start trebuie furnizat pentru fiecare regiune.



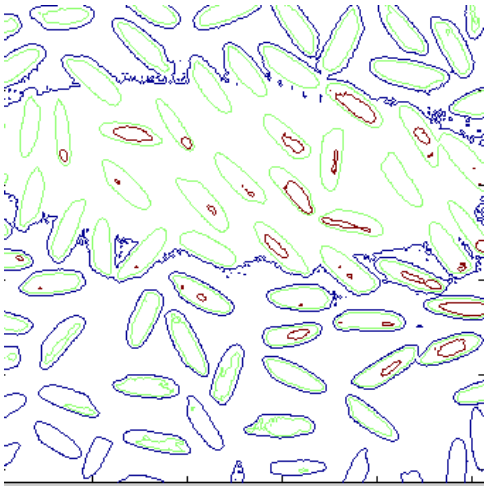


Fig. 6.21. Vizualizarea contururilor cu funcţia imcontour

Printre funcţiile definite în Matlab ce servesc scopului acestui paragraf se menţionează :

- Funcţia imcontour. Ea vizualizează conturul dintr-o imagine de intensitate. Sintaxa este **`imcontour(I)`** sau **`imcontour(I,n)`**, unde n este numărul de niveluri ale conturului egal distanţate. Un exemplu poate fi vizualizat în fig. 6.21.

Numărul de niveluri de contur poate fi specificat şi prin intermediul unui vector, v. Componentele vectorului reprezintă valorile de gri ale fiecărui contur, iar numărul de niveluri este egal cu length(v), adică lungimea vectorului v.

- Funcţia edge este utilizată la datecţia frontierelor regiunii. Pentru detectarea frontierelor se caută zone în care intensitatea se schimbă rapid. În acest scop se apelează unul din următoarele criterii: locuri unde prima derivata a intensităţii are amplitudinea mai mare decât un anumit prag sau locuri unde a doua derivată a intensităţii trece prin zero.

Funcţia oferă un număr de operatori de derivare, fiecare dintre ei implementând una din definiţiile anterioare. Pentru unii operatori se poate specifica dacă operaţia trebuie să fie sensibilă la margini orizontale sau verticale sau la ambele margini. Funcţia edge returnează o imagine binară ce conţine 1 în zonele unde este detectată frontiera şi 0 în rest. În continuare se prezintă un exemplu ce utilizează metoda Sobel pentru detectarea frontierelor. Operatorul este sensibil atât la marginile orizontale, cât şi la cele verticale.

```
A1=imread('blood1.tif') ;
A2=edge(A1,'sobel') ;
imshow(A1),figure,imshow(~A2)
```

Rezultatul este prezentat în fig. 6.22. Imaginea finală este reprezentată după inversarea culorilor.

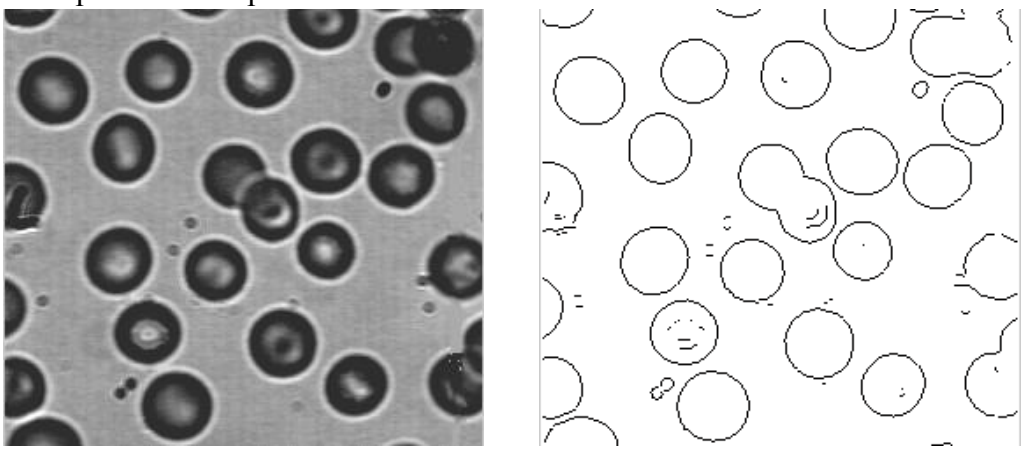


Fig. 6.22. Detectarea frontierelor cu funcţia edge

- Funcţia qtdecomp execută o descompunere a imaginii. Ea lucrează divizând imaginea în blocuri. Este obligatoriu ca imaginea iniţială să fie pătratică; în acest scop se pot adăuga zerouri în imaginea iniţială sau se aplică funcţia imresize. În cazul când criteriul de descompunere este domeniul maxim de variaţie a intensităţii să fie o valoare, a, specificată în domeniul [0, 1], sintaxa este:

```
S=qtdecomp(I,a).
```

În matricea S, elementele diferite de 0 reprezintă colţul stâng al fiecărui bloc, iar valoarea indică dimensiunea blocului. S are aceleaşi dimensiuni cu I. În fig. 6. 23 se prezintă un exemplu de descompunere în blocuri a unei imagini. Este vizualizată atât imaginea rezultată în urma aplicării funcţiei qtdecomp, cât şi o imagine în care se vizualizează blocurile.

Fig. 6.23. Descompunerea în blocuri cu funcţia qtdecomp

Secvenţa de program cu care s-au obţinut imaginile este prezentată în continuare.

```
I=imread('alumgrns.tif');
I1=imresize(I,[256,256],'nearest');
imshow(I1)
S=qtdecomp(I1,.2);
figure,imshow(S)
S1=zeros(256,256);
for i=1:256
    for j=1:256
        if S(i,j)~=0
            k=S(i,j)
            S1(i:i+k-1,j:j+k-1)=I1(i,j);
        end
    end
end
S2=double(S1)/255;
```

```
figure,imshow(S2)
```

6.2.5. Prelucrarea imaginilor binare

Imaginile binare conţin doar 2 niveluri de intensităţi, ce apar ca urmare a unei operaţii de selecţie în funcţie de anumite proprietăţi. În multe cazuri, acest lucru apare în urma unei segmentări pe baza unui prag de intensitate. Scopul binarizării este separarea caracteristicilor faţă de fundal, astfel încât numărarea, măsurarea sau anumite operaţii de comparaţie să poată fi realizate.

Operaţiile efectuate asupra imaginilor binare pot fi clasificate în două categorii:

- operaţii booleene pentru combinarea imaginilor;
- operaţii morfologice ce modifică valoarea unor pixeli din imagine.

În cadrul acestui paragraf se adoptă convenţia ca pixelii ON să fie aparţinători regiunii de interes, iar cei OFF aparţin fundalului.

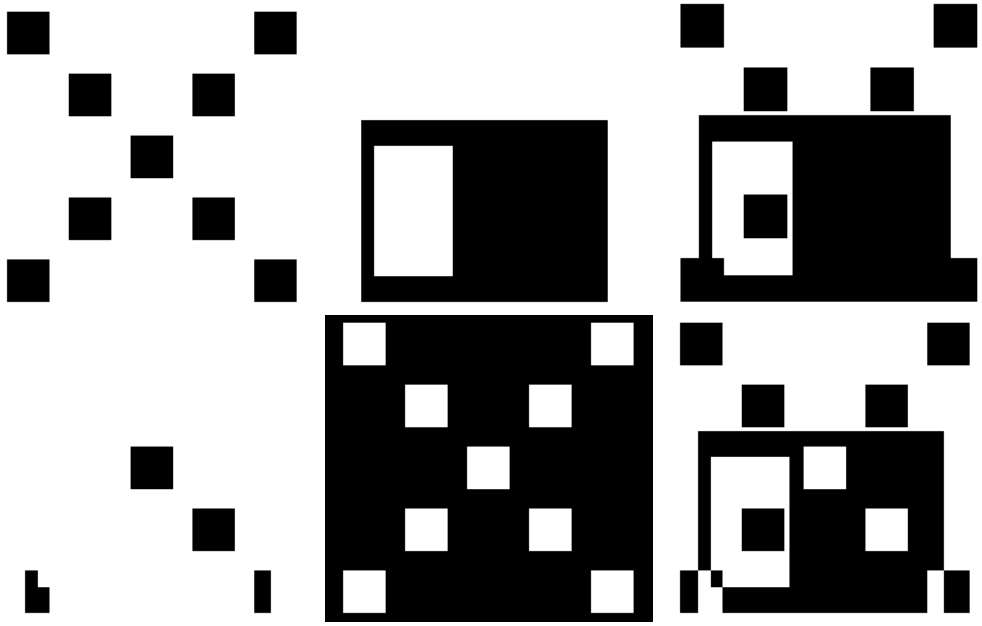


Fig. 6.24. Operaţii booleene simple: imagine a, imagine b, a SAU b, a ŞI b, NOT a, a SAU exclusiv b

Operaţiile booleene sunt: ŞI, SAU, SAU exclusiv, NOT. În fig. 6.24 sunt ilustrate câteva exemple. De fapt, doar trei operaţii sunt suficiente pentru obţinerea unui numit rezultat. De exemplu, A xor B ≡ (A AND NOT B) OR (NOT A AND B).

Este posibil să se utilizeze imagini binare ca măşti pentru a modifica o

imagine de niveluri de gri. Acest lucru se poate face pentru a scoate în evidenţă doar regiunile de interes sau pentru a selecta acele regiuni care au o anumită intensitate, pentru a fi măsurate.

O altă utilizare a măştilor şi operaţiilor booleene este introducerea unor etichete în imagini ce conţin zone albe şi negre de dimensiuni mici, unde introducerea unei etichete scrise cu alb sau negru face dificilă citirea. Se poate crea o mască din etichetă, care se dilată cu câte un pixel pe fiecare direcţie. Această mască este utilizată pentru ştergerea pixelilor din imaginea de intensitate şi apoi se scrie eticheta cu negru.

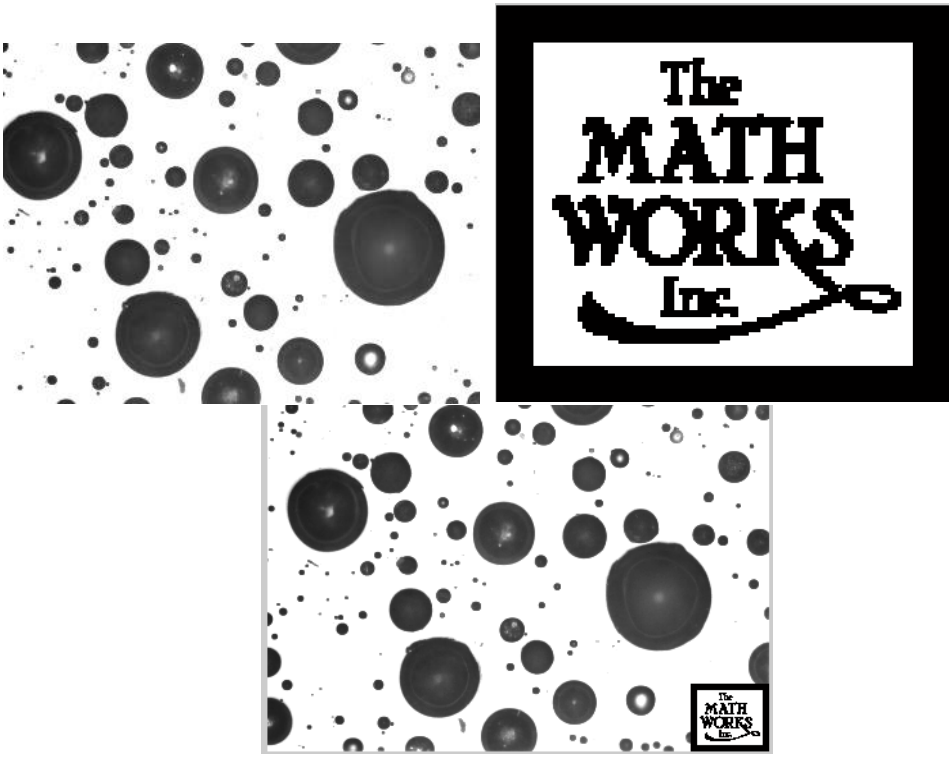
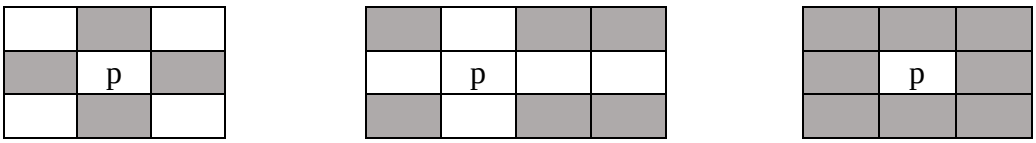


Fig. 6.25. Introducerea unei etichete în imagine

Un alt exemplu este combinarea a două imagini, operaţie utilizată mult în reclame şi grafică comercială. Se creează o mască pe baza unei imagini şi în zona respectivă informaţia se preia din imaginea dorită. Un exemplu se prezintă în fig. 6.25.

Pentru a putea eticheta obiectele dintr-o imagine trebuie stabilite clar condiţiile în care doi pixeli învecinaţi aparţin unui aceluiaşi obiect. Un pixel de coordonate p(x, y) are doi pixeli învecinați pe orizontală și doi pe verticală, (x-1, y), (x+1, y), respectiv (x, y-1) și (x, y+1). Ei reprezintă vecinătatrea ortogonală. De asemenea, există patru pixeli învecinați pe diagonală: (x-1, y-1),. (x+1, y-1),

(x-1, y+1) şi (x+1, y+1). Cei opt pixeli formează vecinătatea de 8.



Doi pixeli aparţin unui aceluiaşi obiect dacă sunt adiacenţi în vecinătatea V4, respectiv V8, deci obiectele dintr-o imagine se pot stabili fie pe baza vecinătăţii V4, fie pe baza V8, adică prin conectivitate de 4 sau conectivitate de 8.

1	1	1	0	0	0	0	0
1	1	1	0	1	1	0	0
1	1	1	0	1	1	0	0
1	1	1	0	0	0	1	0
1	1	1	0	0	0	1	0
1	1	1	0	0	0	1	0
1	1	1	0	0	1	0	0
1	1	1	0	0	0	0	0

1	1	1	0	0	0	0	0
1	1	1	0	2	2	0	0
1	1	1	0	2	2	0	0
1	1	1	0	0	0	3	0
1	1	1	0	0	0	3	0
1	1	1	0	0	0	3	0
1	1	1	0	0	4	0	0
1	1	1	0	0	0	0	0

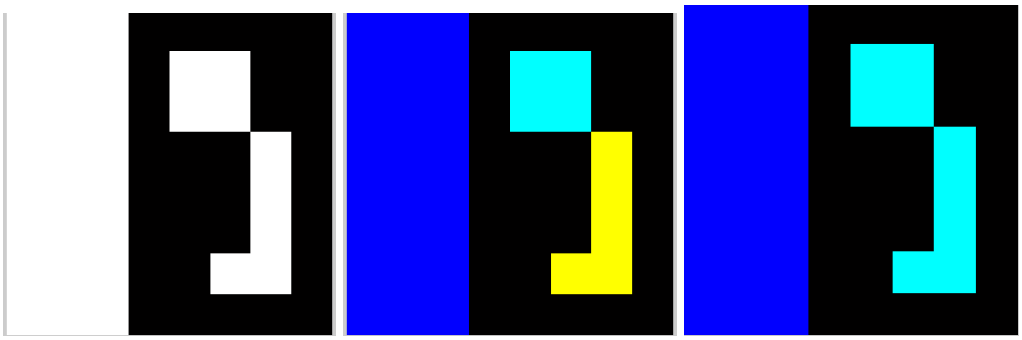
1	1	1	0	0	0	0	0
1	1	1	0	1	1	0	0
1	1	1	0	1	1	0	0
1	1	1	0	0	0	1	0
1	1	1	0	0	0	1	0
1	1	1	0	0	0	1	0
1	1	1	0	0	1	0	0
1	1	1	0	0	0	0	0

1	1	1	0	0	0	0	0
1	1	1	0	2	2	0	0
1	1	1	0	2	2	0	0
1	1	1	0	0	0	2	0
1	1	1	0	0	0	2	0
1	1	1	0	0	0	2	0
1	1	1	0	0	2	0	0
1	1	1	0	0	0	0	0

Pentru etichetarea componentelor conectate dintr-o imagine binară se utilizează comanda bwlabel. Ea returnează o matrice de aceleaşi dimensiuni cu matricea iniţială, iar obiectele diferite sunt identificate prin numere întregi succesive, iar pixelii aparţinători fundalului au valoarea 0. Sintaxa este **`bwlabel(I,vecinatate)`**. Rezultatul este o matrice de clasă dublă, nu o imagine binară. Pentru a putea vizualiza rezultatul se aplică o secvenţă de tipul :

```
X=bwlabel(I,4) ;
imX=mat2gray(X);
```

```
imagesc(imX,[0,1]);colormap(jet)
imshow(X+1,map) ;
```



Efectul tipului de vecinătate la aplicarea comenzii bwlabel

În figură se prezintă o imagine binară la care s-a aplicat comanda bwlabel cu vecinătate de 8, respectiv cu vecinătate de 4. Se remarcă faptul că numărul de componente determinate depinde de tipul vecinătății.

Comanda poate fi apelată şi sub forma :

```
[L,num]=bwlabel(imagine,conectivitate);
```

unde magine este o imagine binară, L este o matrice de aceleaşi dimensiuni cu imagine, iar num este numărul de obiecte din imagine, este un argument optional.

O altă comandă legată de etichetarea obiectelor din imaginile binare este bwselect, ce permite selectarea anumitor elemente dintr-o imagine. Sintaxa este bwselect(I,c,r,vecinatate), unde I este imaginea binară iniţială, c şi r sunt coordonatele pixelului (pixelilor) unde se face selectarea, iar vecinătate are aceeaşui semnificaţie ca în cazurile precedente. Coordonatele pixelilor se dau în ordinea coloană, linie. În cazul când c şi r sunt vectori având aceeaşi dimensiune, se vor selecta toate obiectele care se suprapun cu pixelii indicaţi. Rezultatul este o imagine binară, ce conţine doar obiectele respective. O ilustrare a acestei comenzi este prezentată în fig. 6.28. Pentru a obţine cea de-a doua imagine din prima s-a utilizat următoarea secvenţă :

```
c=[15 15 110 220];
 r=[15 220 15 220];
 bw=bwselect(a,c,r,4);
```

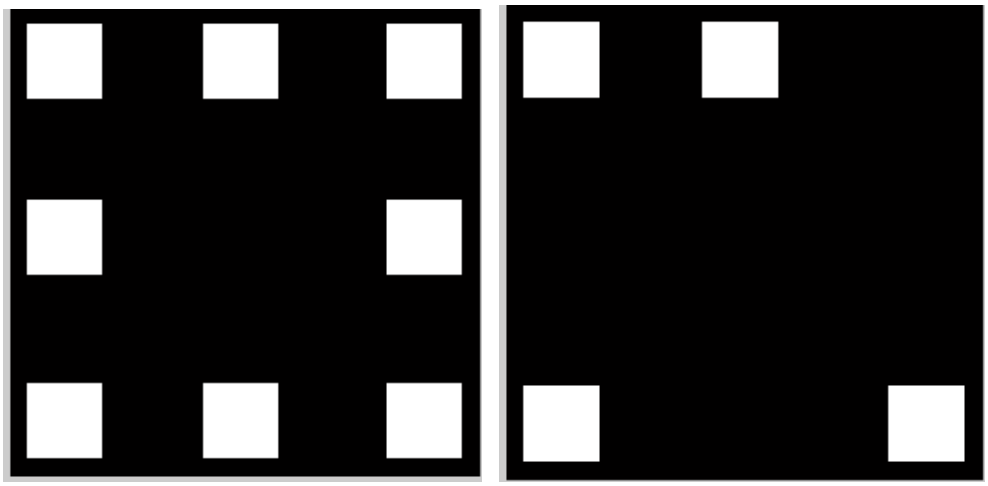
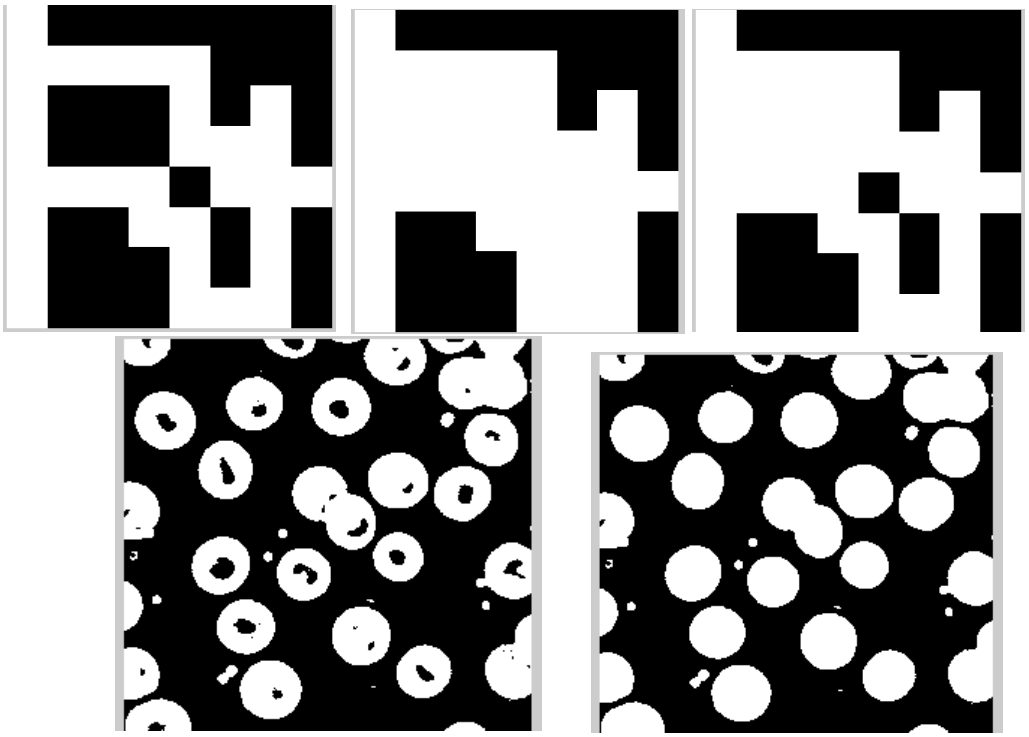


Fig. 6.28. Utilizarea comenzii bwselect

În anumite situaţii este necesar să se identifice pixelii ce formează o anumită regiune interioară închisă. La fel ca în cazul extinderii regiunii se poate pleca de la un pixel şi se investighează cei aflaţi în vecinătatea acestuia (fie vecinătăţi de 4, fie vecinătăti de 8), etichetând pixelii ON ca aparţinând unei anumite caracteristici. Procesul se continuă iterativ până la atribuirea etichetelor tuturor pixelilor din imagine. Când procesul de etichetare este încheiat se pot aplica o serie de operaţii booleene. Pentru găsirea regiunilor interioare şi umplerea acestora se caută pixelii ce sunt setaţi OFF (adică aparţin fundalului) şi sunt înconjuraţi de pixeli ON, iar în reprezentarea frontierelor ei nu trebuie să fie conectaţi cu marginea câmpului imagine. Dacă se găsesc etichete ce îndeplinesc aceste condiţii (se află în interiorul unei regiuni închise ce nu atinge marginea câmpului imagine) s-au identificat regiunile interioare închise. Printr-o operaţie SAU logic, aceşti pixeli se ataşează imaginii originale rezultând un procces de umplere.

În Matlab acest lucru se realizează cu comanda imfill. sintaxa este **`imfill(I,py,px,vecinatate)`**, unde py, px sunt coordonatele pixelului cu care se începe procesul de umplere aplicat imaginii I, iar **`vecinătate`** specifică tipul de conexiune utilizat (vecinătate de 4 sau de 8, implicit se consideră 8).
Operația se poate aplica și interactiv bw=imfill(I), utilizatorul selectând pixelii de unde începe procesul. Pentru imaginile de intensitate, imfill umple goluri, golul fiind un set de pixeli întunecați, înconjurați de pixeli luminoși.



Umplerea regiunilor cu comanda imfill

În fig. 6.26 se prezintă efectul aplicării funcţiei imfill. Prima imagine este cea iniţială, cea de-a doua este obţinută cu vecinătate de 8, iar a treia cu vecinătate de 4. Se remarcă menţinerea celor două dreptunghiuri negre din partea stângă a imaginii iniţiale. Următoarele două imagini exemplică umplerea golrilor dintr-o imagine binară. Este prezentată imaginea iniţială si imaginea obţinută după aplicarea comenzii **`imfill(I,'holes').`**

Operaţiile logice se utilizează frecvent în măsurări, în vederea eşantionării semnalului preluat din imagini. De ex. măsurarea grosimii stratului de acoperire pe un cablu sau o tablă. Imaginea poate fi binarizată în condiţii corespunzătoare, dar grosimea stratului de acoperire este neuniformă. Pentru a obţine o serie de valori discrete în vederea calculelor statistice se generează o machetă de măsurare liniar sau radial în funcţie de scopul măsurării, conform fig. 6.29. Cele două imagini se combină cu funcţia ŞI logic rezultând o serie de semnale discrete ce reprezintă eşantionul şi pot fi măsurate cu uşurinţă.

Machetă de măsurare liniară şi radială

Sunt numeroase situaţii când se construiesc diferite machete de măsurare. În cazul unor particule acoperite şi incluse într-o masă metalografică prezintă interes grosimea stratului de acoperire. În această situaţie, grosimile stratului trebuie interpretate stereologic, deoarece planul de lustruire al probei nu trece prin centrul particulelor şi acoperirea apare mai subţire decât este în realitate. Acest lucru se rezolvă prin utilizarea unei machete de măsurare formată din linii aleatoare. Prin prelucrarea lungimilor segmentelor generate prin intersectarea machetei cu imaginea iniţială se poate determina grosimea stratului de acoperire pe direcţie normală, [Russ].

Alegerea machetei corespunzătoare este esenţială pentru corectitudinea măsurării. Pentru o structură metalografică cristalină, unde interesează numărul de cristale şi nu se cunoaşte anterior o orientare preferenţială, se alege o machetă formată din cercuri, pentru a eşantiona uniform pe toate direcţiile. Combinând cecurile cu imaginea binară a structurii metalografice se obţin o serie de arce. Numărul de cristale (şi numărul de frontiere) se obţine pe baza numărului de arce, ce poate fi determinat prin numărare automată, [Russ]. Se mai poate determina, pe baza numărului şi lungimii arcelor), aria suprafeţei/unitatea de nolum şi mărimea cristalelor.

În cadrul operaţiilor booleene se pot efectua operaţii cu pixeli grupaţi conform anumitor caracteristici. Acest lucru presupune în prealabil o etichetare a regiunilor, deoarece operaţia se efectuează pe regiuni. În acest caz, operaţia ŞI logic va determina ca o anumită regiune dintr-o imagine să se transfere în întregime în imaginea finală atunci când cel puţin un pixel ON din imaginea iniţială aparţinător unei anumite caracteristici (regiuni) este în coincidenţă cu un pixel ON din cea de-a doua imagine.

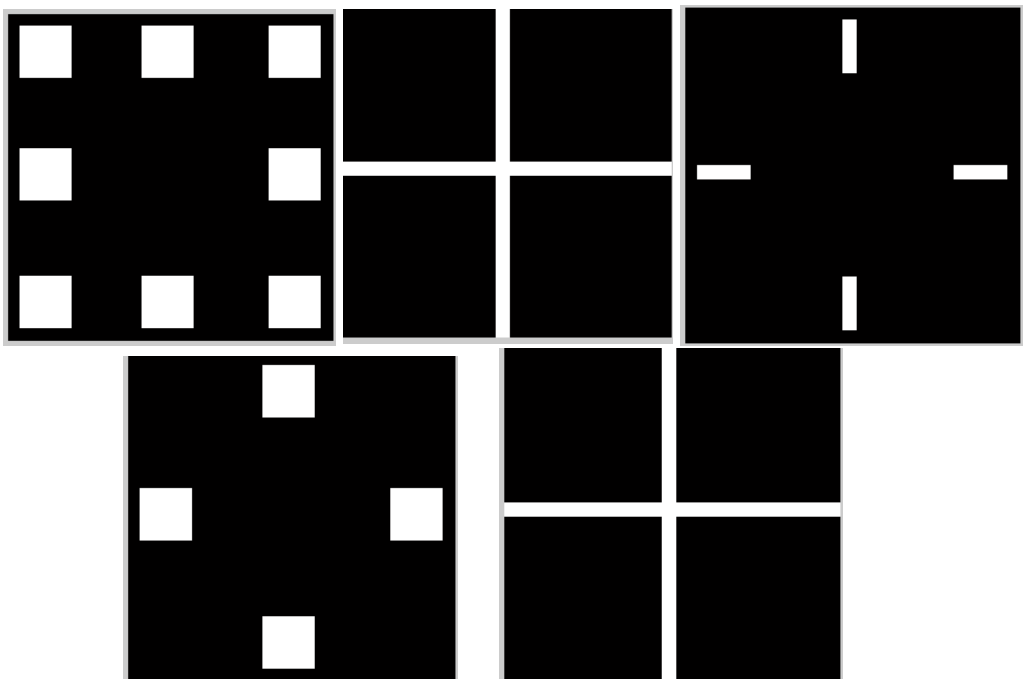


Fig. 6.30. Operaţii booleene cu regiuni: a, b, a ŞI b, a ŞI b cu regiuni, b ŞI a cu regiuni

În fig. 6.30 se poate vedea efectul acestei operaţii prin combinarea imaginii a şi b. Se observă că operaţia nu este comutativă. Imaginile sunt rezultatul următoarei aplicaţii:

```
a=zeros(256,256);b=a;
a(10:50,10:50)=1;a(10:50,206:246)=1;a(10:50,106:146)=1;
a(106:146,10:50)=1;a(206:246,10:50)=1;a(106:146,206:246)=1;
a(206:246,206:246)=1;a(206:246,106:146)=1;
imshow(a)
b(121:131,:)=1;b(:,121:131)=1;
figure,imshow(b)
log_and=a & b;
figure,imshow(log_and)
[r,c]=find(log_and);
feat_and=bwselect(a,r,c);
figure,imshow(feat_and)
feat_and1=bwselect(b,r,c);
figure,imshow(feat_and1)
```

Aceste operaţii pot fi utilizate la aplicarea markerilor într-o caracteristică pentru a face o selecţie. De ex. imaginea unor celule având în interiorul lor anumite elemente caracteristice. Se produc două imagini binare prin delimitare cu prag. În prima se evidenţiază toate celulele, iar în cea de-a doua doar markerii.

Prin combinarea celor două imagini cu o operaţie ŞI logic se vor selecta acele celule care au fost marcate într-un anumit mod.

Tot în baza acestui tip de operaţii se pot determina componentele aflate la o anumită distanţă de o frontieră neregulată. În acest scop se generează o imagine în care frontiera se îngroaşă controlat (printr-o operaţie de dilatare) şi se aplică o operaţie ŞI logic rezultând o imagine ce va conţine doar acele caracteristici ce se află la o distanţă controlată.

O altă posibilă aplicaţie este utilizarea operaţiei ŞI logic combinată cu două segmentări pe bază de nivel de prag. Imaginile pot fi urmărite în fig. 6.31. Fie imaginea unei structuri cristaline (a) în care prezintă interes frontierele ce apar între cristale. Se fac două segmentări cu niveluri diferite de gri, rezultând o imagine (b) în care contururile nu sunt închise şi o a doua (c) în care frontierele sunt mai îngroşate şi apar o serie de elemente în interiorul cristalelor. Prin combinarea cu ŞI logic a celor două imagini se obţine o imagine în care frontierele cristalelor sunt închise, dar mai îngroşate (d). Acest lucru se elimină prin scheletare (e) şi se elimină pixelii de capăt, ce nu aparţin frontierei (f). Asupra operaţiilor de scheletare şi eliminarea pixelilor de capăt se va reveni în continuarea paragrafului. În final se evidenţiază cristalele din structură (g).

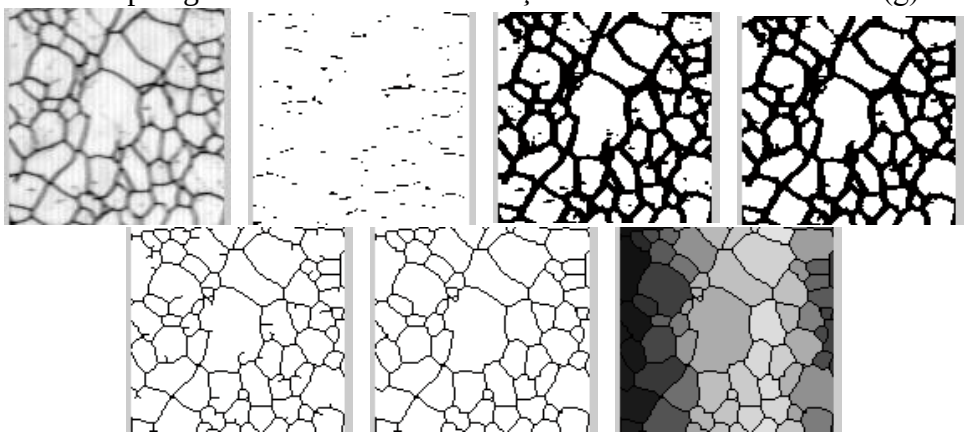


Fig. 6.31. Evidenţierea frontierelor cu ajutorul funcţiei ŞI logic

Aplicaţia cu care s-a făcut prelucrarea este prezentată în continuare şi a fost concepută pe baza tutorialului de prelucrări de imagini din Matlab

```
load imdemos steel;
imshow(steel)
bw_1=steel>70;
bw_2=steel>210;
figure,imshow(bw_1),figure,imshow(bw_2)
[r,c]=find(bw_1==0);%se determina pixelii ON
bw_3=bwselect(~bw_2,c,r,8);% se selecteaza regiunile comune
```

```
figure,imshow(~bw_3)
bw_skel=bwmorph(bw_3,'skel',6);% se face scheletarea
figure,imshow(~bw_skel)
bw_4=bwmorph(bw_skel,'spur',8);% se elimina pixelii ce nu apartin
frontierelor
figure,imshow(~bw_4)
graunti=~bw_4;
[labeled,N]=bwlabel(graunti,4);
gri=ind2gray(labeled+1,[0 0 0;jet(N)]);
figure,imshow(gri)
```

Cea mai extinsă clasă de operaţii asupra imaginilor binare sunt operaţiile morfologice, care se utilizează foarte mult în analiza formelor din imagine. În MATLAB, există numeroase operații morfologice implementate pe imagini binare, dar pot fi aplicate și în cazul imaginilor monocrome.

Există două operații morfologice foarte simple: translația și reflexia, care intervin în multe din operațiile morfologice.

Pentru efectuarea unei translații cu t(tx, ty), în cazul unei imagini binare, se detrmină coordonatelor pixelilor on (cu valoare 0), se definește matricea translatată, de aceleași dimensiuni cu matricea inițială și se setează on pixelii aflați în coordonatele: coordonata inițială plus translația corespunzătoare.

	x			
x	x	x		
	x			

			x	
		x	x	x
			x	

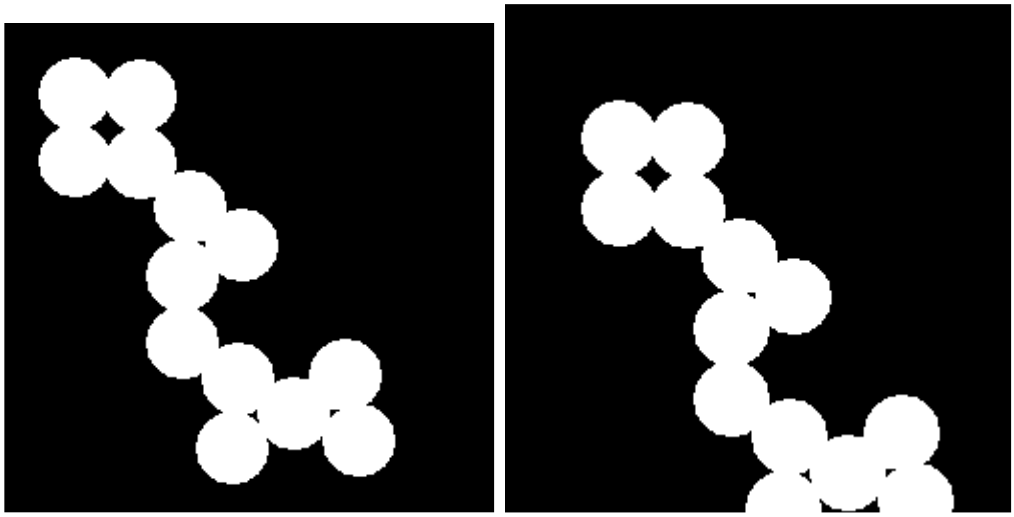
Translație cu (2,2)

Dacă A este matricea inițială și t(tx, ty) este vectorul translației, se scrie codul:

```
a=imread('circles.png');
[r,c]=find(a==1);
t=[30,20];
[n,m]=size(a);
at=a;
at=zeros(size(a));
r1=r+t(1);
c1=c+t(2);
for i=1:length(r);
    if ((r1(i)<=n)&(c1(i)<=m)&(r1(i)>0)&c1(i)>0)
    at(r1(i),c1(i))=1;
    end
```

```
end
```

Imagine translatata



Având un set de pixeli, reflexia acestora față de un punct se obține construind simetricul fiecărui pixel în raport cu punctul de reflexie, conform figurii.

		x				
	x	x				
x	x	x				
	x				x	
				x	x	x
				x	x	
				x		

Punct de reflexie

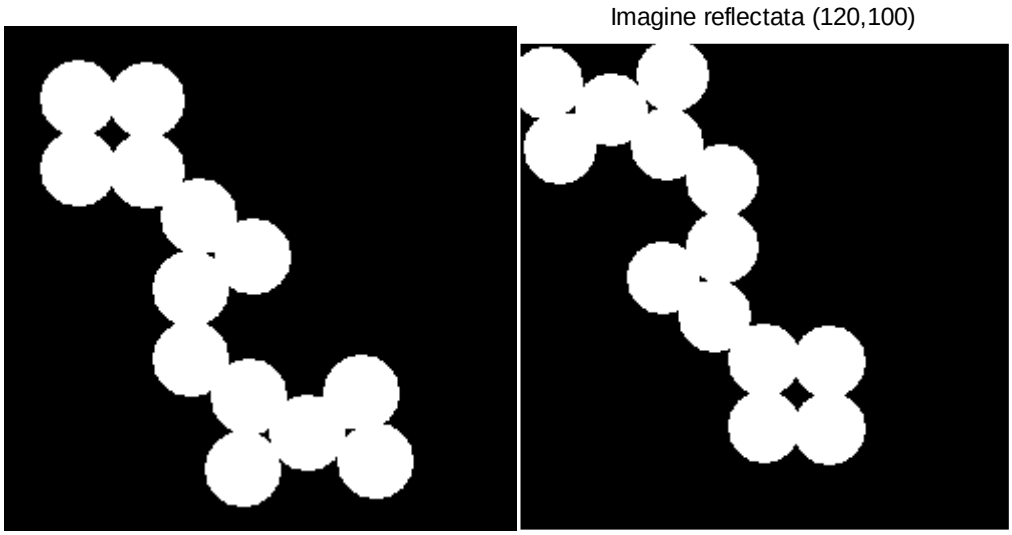
Se observă din figură că noile coordinate ale pixelilor se obțin din relația:

$(x_r, y_r) = (p_x + p_x - x, p_y + p_y - y) = (2p_x - x, 2p_y - y)$

Unde (x,y) sunt coordonatele punctului current, (x_r, y_r) sunt coordonatele punctului reflectat și (p_x, p_y) sunt coordonatele punctului de reflexie.

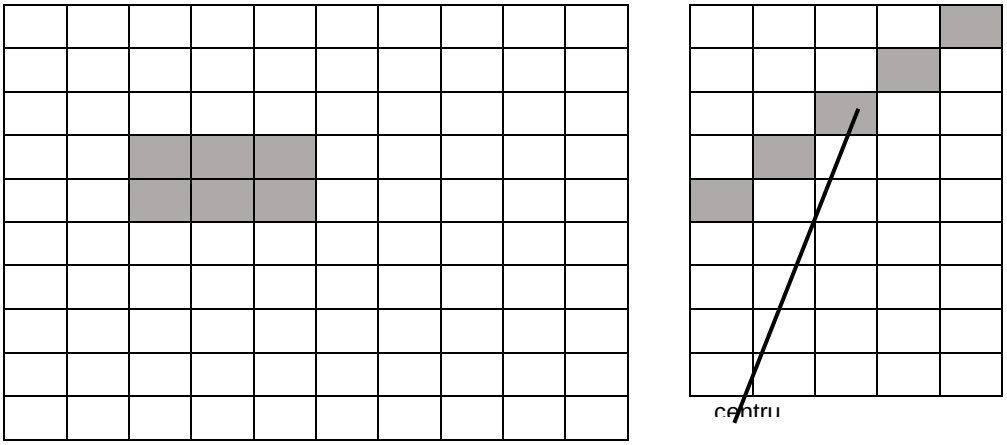
```
rx=[120,100];
ar=zeros(size(a));
rr=2*rx(1)-r;
cr=2*rx(2)-c;
for i=1:length(r);
    if (rr(i)<=n)&(cr(i)<=m)&(rr(i)>0)&(cr(i)>0)
    ar(rr(i),cr(i))=1;
    end
```

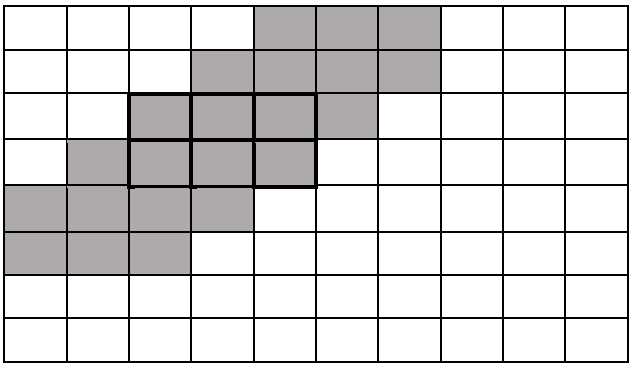
```
end
```



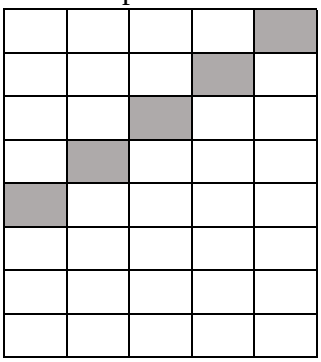
Două operații fundamentale în prelucrările morfologice sunt dilatarea și erodarea. Aceste operaţii pot fi descrise ca adăugări, respectiv eliminări de pixeli din imaginile binare conform anumitor reguli ce depind de pixelii din vecinătate.

Dilatarea este o operație de creștere sau îngroșare a unui obiect dintr-o imagine binară. Modul specific de creștere se controlează cu ajutorul unei forme sau element structural. Centrul acestui element se deplasează prin toți pixelii imaginii, verificându-se suprapunerea elementului cu pixeli ON. Dacă există cel puțin o suprapunere între elementul structural și un pixel de valoare 1 în imagine, în imaginea rezultată, pixelul corespunzător centrului elementului structural va fi 1.

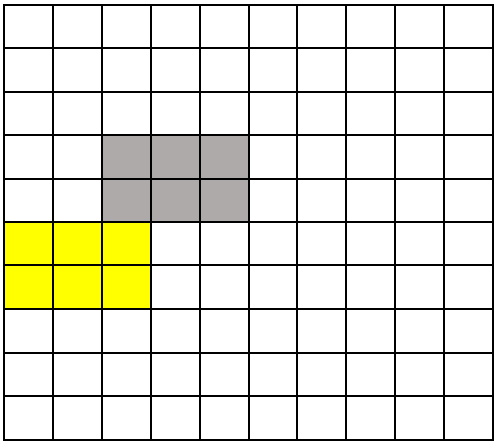




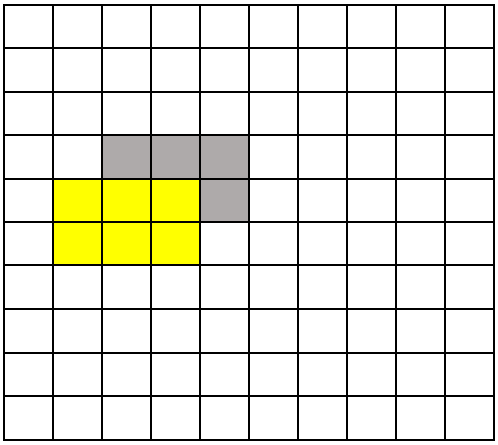
Operația de dilatare poate fi definită și prin intermediul translației. Având un element structural cu care se face dilatarea, fiecare pixel 1 din acesta are coordonatele exprimate în raport cu pixelul central. Pentru elementul structural din exemplul anterior coordonatele pixelilor 1 sunt:



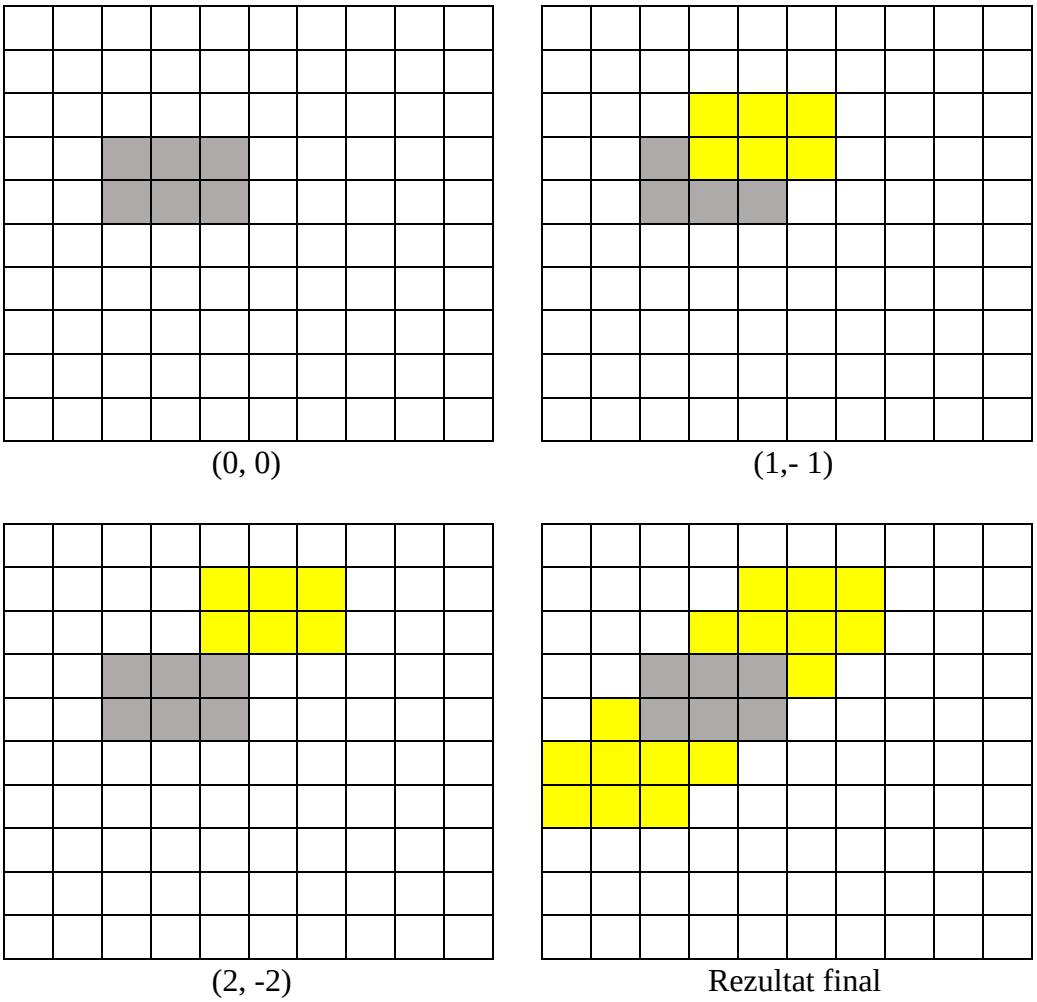
(2,-2)
(1,-1)
(0,0)
(-1.1)
(-2,2)



(-2, 2)



(-1, 1)



Rezultatul dilatării poate fi obținut prin reuniunea tuturor translațiilor cu aceste coordonate.

Funcţia imdilate realizează dilatarea unei imaginii. Sintaxa este: `I1=imdilate(I,SE)`, unde I este imaginea ce se dilată, iar SE este un element structural ce controlează operaţia şi este constituit din 1 şi 0. Poziţia elementelor egale cu 1 indică vecinătatea în care se face operaţia. SE poate fi definit de utilizator direct sau poate fi obţinut cu ajutorul funcţiei strel. Cu funcţia imdilate se poate realiza o dilatare binară, dacă imaginea I este o imagine binară sau o dilatare pe niveluri de gri, în cazul când I este o astfel de imagine.

Pentru dilatarea binară elementul structural furnizat de funcţia strel poate fi de mai multe tipuri: line, disk, square, diamond, octagon, pair sau periodicline. În continuare se prezintă sintaxa pentru fiecare din funcţiile menţionate şi un

exemplu de element structural generat cu aceasta.

`se=strel('line',lungime,grade)` – crează un element structural de un număr de pixeli egal cu lungime, iar grade specifică unghiul măsurat în sens trigonometric, de la axa orizontală. **`se=strel('line',3,90)`** generează elementul se=[1;1;1], **`se=strel('line',5,0)`** generează se=[1 1 1 1 1], iar **`se=strel('line',3,45)`** generează se=[0 0 1;0 1 0;1 0 0]. Efectul aplicării unei operaţii de dilatare cu aceste elemente structurale poate fi urmărit în fig. 6.35 b, c şi d.

`se=strel('disk',r,n)` – crează un element structural de forma unui disc cu raza r, iar n poate fi 0,4,6 sau 8. Pentru valori pozitive ale lui n, elementul se se descompune în mai multe elemente. Valoarea implicită a lui n este 4. **`se=strel('disk',3,0)`** generează elementul structural :

0 0 0 1 0 0 0
0 1 1 1 1 1 0
0 1 1 1 1 1 0
1 1 1 1 1 1 1
0 1 1 1 1 1 0
0 1 1 1 1 1 0
0 0 0 1 0 0 0

iar rezultatul operaţiei este prezentat în fig. 6.35 e.

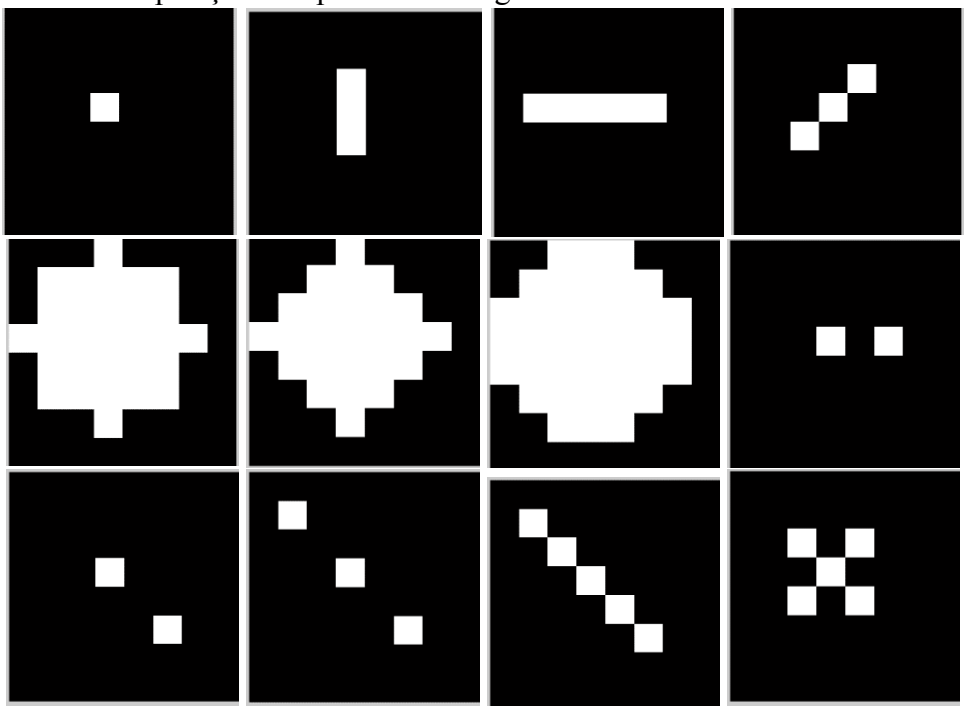


Fig. 6.35. Exemple de dilatări cu diferite elemente structurale

`se=strel('diamond',r)` – generează un element în formă de romb, având semidiagonalele egale cu r. **`se=strel('diamond',3)`** are ca efect imaginea din fig. 6.35 f.

Cu comanda se=strel('octagon',r) se obţine un element de formă octogonală, r fiind distanţa măsurată pe direcţie orizontală şi verticală din centrul octogonului la laturi. Efectul aplicării comenzii se=strel('octagon',3) se poate vedea în fig. 6.35 g.

`se=strel('pair',offset)` crează un element structural cu două elemente egale cu 1, unul este în origine şi celălalt are locaţia specificată de vectorul offset. Acesta trebuie să fie un vector cu două componente întregi, prima indică offset-ul pe linie, iar cea de-a doua pe coloane. **`se=strel('pair',[0 2])`** va genera elementul structural [0 0 1 0 1], imaginea fiind vizualizată în fig. 6.35 h, iar **`se=strel('pair',[2 2])`** are efectul din fig. 6.35 i, iar elementul este de forma:

0 0 0 0 0
0 0 0 0 0
0 0 1 0 0
0 0 0 0 0
0 0 0 0 1.

`se=strel('periodicline',P,V)` furnizează un element structural ce conţine 2P+1 elemente egale cu 1. V este un vector cu două componente şi reprezintă offset-ul pe linii şi coloane. Un element este localizat în origine, iar ceilalţi în poziţiile: 1V, -1V, 2V, -2V, … până la PV, -PV. **`se=strel('periodicline',1,[2 2])`** determină imaginea din fig. 6.35 j şi elementul structural este:

1 0 0 0 0
0 0 0 0 0
0 0 1 0 0
0 0 0 0 0
0 0 0 0 1,

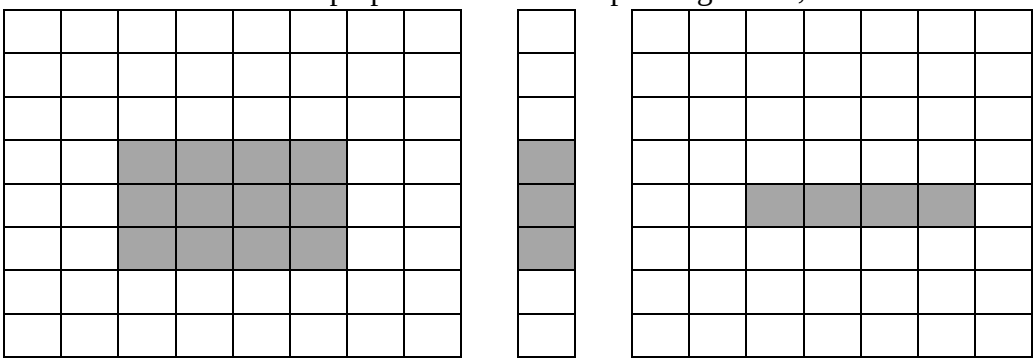
iar **`se=strel('periodicline',2,[1 2])`** are ca efect imaginea din fig. 6.35 k, elementul structural fiind:

1 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 1 0

0 0 0 0 1.

În afara apelării funcţiei strel, elementul structural poate fi definit direct de utilizator. Un element de forma: [1 0 1;0 1 0;1 0 1] determină imaginea din fig. 6.35 l.

Erodarea subțiază obiectele dintr-o imagine. Operația lucrează similar cu dilatarea, fiind controlată printr-un element structural. Dacă elementul structural se suprapune complet peste obiectul din imagine, rezultatul va fi 1, iar dacă elementul structural se suprapune cu minim un pixel egal cu 0, rezultatul va fi 0.



Erodarea elimină pixelii din imagine, respectiv îi aduce în starea OFF, dacă au fost In ON. Scopul poate fi eliminarea unor pixeli selectaţi în urma unei operaţii de segmentare cu prag, deoarece ei au valoare intensităţii corespunzătoare, dar nu aparţin regiunii de interes. Acest lucru poate fi cauzat de zgomot sau de situarea lor în apropierea unei frontiere dintre o regiune mai luminoasă şi una mai întunecată.

Cea mai simplă variantă de erodare este eliminarea oricărui pixel ce este în vecinătatea fundalului. Efectul este înlăturarea unui strat de pixeli aflaţi la periferia tuturor obiectelor din imagine, ceea ce va diminua dimensiunile, respectiv va separa părţi componente din unele obiecte.

Operaţia complementară este dilatarea, cu ajutorul căreia se adaugă pixeli. O variantă simplă este adăugarea tuturor pixelilor ce aparţin fundalului şi sunt în contact cu pixeli aparţinători unei regiuni. Efectul este adăugarea unui strat de pixeli la periferia obiectelor, o creştere a dimensiunilor şi o eventuală fuziune a unor obiecte.

În practică, dilatarea și erodarea se utilizează în diverse succesiuni.Între cele mai utilizate combinații sunt:

- deschiderea (opening),
- închiderea (closing),
- transformarea hit-or-miss.

În următorul exemplu, fig. 6.32, relativ simplu, se ilustrează eliminarea

pixelilor izolaţi. Se operează o erodare, urmată de o dilatare, pentru a elimina efectul de modificare a dimensiunilor obiectului. Prima imagine este cea iniţială, a doua este obţinută în urma unei segmentări cu prag şi în final este imaginea rezultată după erodare şi dilatare.

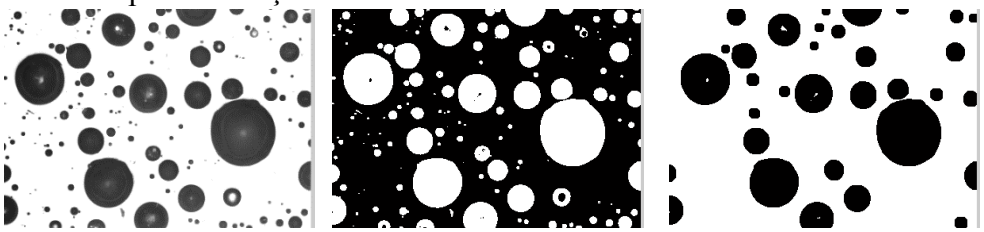
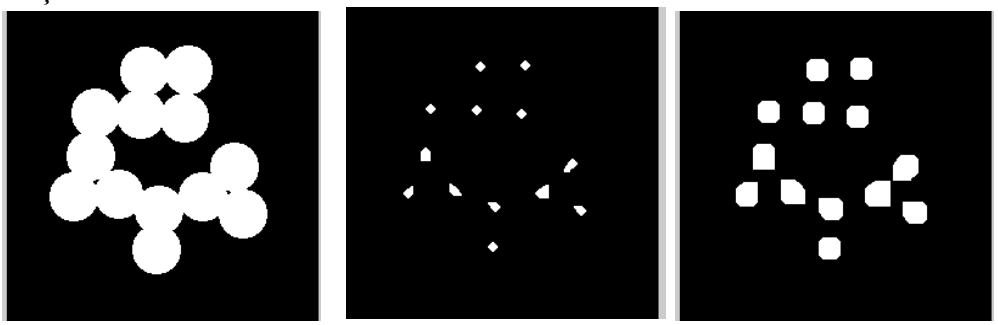


Fig. 6.32. Eliminarea pixelilor izolaţi prin erodare şi dilatare

Combinarea operaţiilor, erodare urmată de dilatare, se numeşte deschidere, datorită faptului că prin aceste operaţii se pot marca zonele de contact dintre obiecte ce se ating. Ea netezește conturul obiectelor, dar rupe conexiunile subțiri și netezește proeminențele de dimensiuni mici.Parametrii prin care se controlează erodarea şi dilatarea sunt tipul de vecinătate şi numărul de iteraţii. În cazul operaţiei de deschidere aceşti parametrii se menţin constanţi atât pentru erodare, cât şi pentru dilatare.

În continuare se prezintă un exemplu de prelucrare a imaginii pe baza acestei operaţii, fig. 6.33. Imaginea iniţială este formată din obiecte de dimensiuni apropiate, astfel încât se poate executa o erodare succesivă, fără a pierde unul din obiecte. Ulterior se aplică dilatarea. În urma aplicării succesive a dilatării, zonele de interes fuzionează. Acest lucru poate fi împiedecat prin intermediul unei operaţii logice, ce nu permite trecerea unui pixel în stare ON, dacă are în vecinătate un pixel aparţinător unui obiect diferit. Acest lucru presupune o etichetare a regiunilor şi operaţia logică trebuie executată la fiecare iteraţie a dilatării.



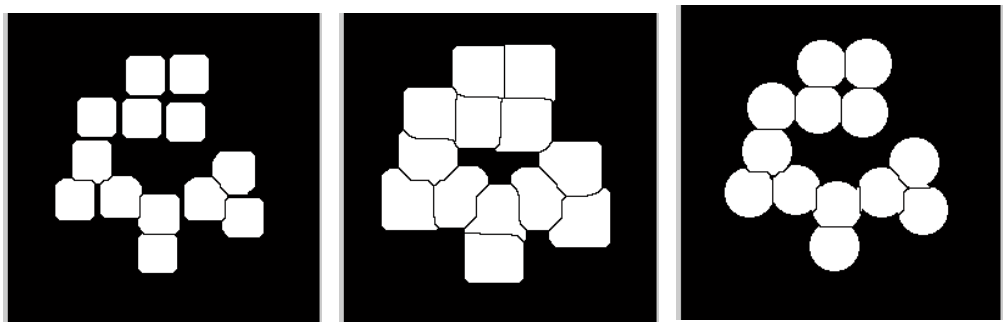


Fig. 6.33. Erodare şi dilatare controlată: imaginea iniţială, după 12 erodări,după 5 dilatări, după 12 dilatări, după 20 dilatări, imaginea finală

În cazul când ordinea de aplicare este inversă, deci dilatare, urmată de erodare, se obţine operaţia numită închidere, rezulatul fiind complet diferit. Prin închidere se poate realiza fuzionarea anumitor obiecte, care în imaginea iniţială prezintă interstiţii fine (fig.6.34). Închiderea netezşte conturul obiectelor, dar reface conexiuni cu întreruperi mici, umple găuri de dimensiuni mai mici decât elemental structural sau decupaje de dimensiuni mici.

Aceste două operaţii se pot aplica în MATLAB cu imopen(imagine, el_struct), respective imclose(imagine, el_struct). In ambele cazuri imagine este o imagine binară sau de intensitate, iar elemental structural este fie o matrice de 0 şi 1, fie este definit cu funcţia strel.

Fig. 6.34. Închiderea interstiţiilor: imagine iniţială, imaginea după dilatare şi imaginea după erodare

În Matlab există o serie de funcţii definite în vederea efectuării operaţiilor morfologice: imdilate, imerode, imclose, imopen, bwmorph etc. În continuare se vor prezenta pe scurt aceste funcţii.

Erodarea unei imagini se poate executa în Matlab cu funcţia imerode. Sintaxa este: `imerode(I,se)`, I fiind imaginea asupra căreia se aplică operaţia, iar se un element structural ce se poate genera în aceleaşi condiţii ca anterior.

Deschiderea se poate executa prin apelarea funcţiei imopen. Sintaxa este:

imopen(I,se), unde se este elementul structural. În fig. 6. 36 se prezintă un exemplu de operaţie de deschidere. Elementul structural utilizat a fost de tip ‘disk’.

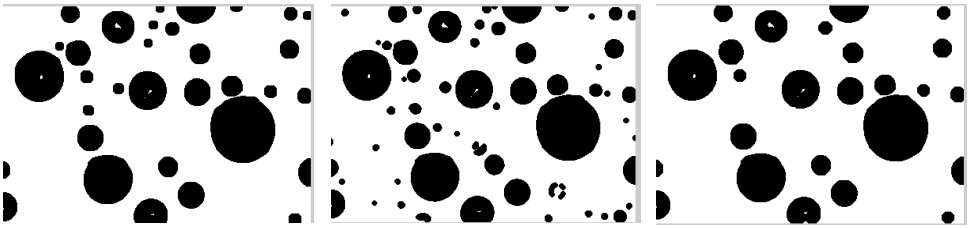
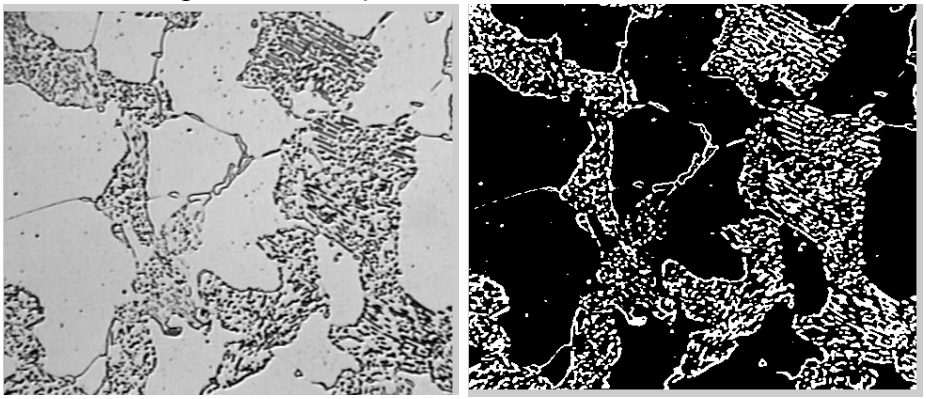


Fig. 6.36 Efectul utilizării a diferite raze la elementul structural de tip ‘disk’ în cazul operaţiei de deschidere

În fig. 6. 36 se prezintă modificarea imaginii finale pentru diferite valori ale razei elementului. Pentru prima imagine r = 6, la a doua r = 4, iar în final r = 8. Se remarcă că prin valoarea razei se poate controla mărimea componentelor ce rămân în imaginea finală.

Închiderea imaginilor binare se realizează în Matlab cu funcţia imclose, având sintaxa: `imclose(I,se)`, I şi `se` având acceaşi semnificaţie ca în cazul precedent. În exemplul următor (fig. 6. 37) se prezintă prelucrarea unei imagini, unde se doreşte unificarea unor componente în urma segmentării cu nivel de prag. Prima imagine este cea iniţială, la care se aplică o segmentare cu nivel de prag (fig. 6.37 b). Închiderea se execută pe baza unui element structural de tip ‘disk’, având raza 6 (fig. 6.37 c), iar în final se aplică o deschidere, cu acelaşi element, pentru eliminarea pixelilor izolaţi.



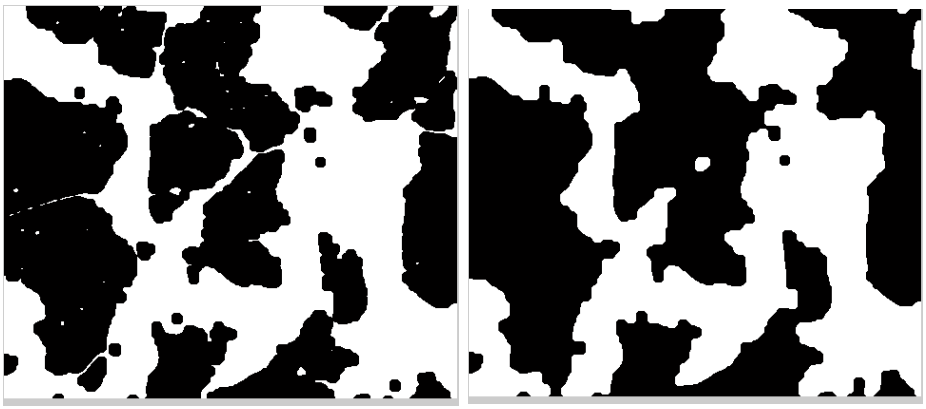


Fig. 6.37. Aplicarea operaţiei de închidere pentru fuziunea unor componente şi a celei de deschidere pentru eliminarea pixelilor izolaţi

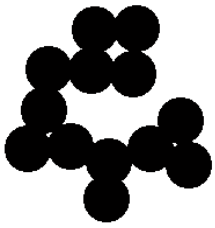
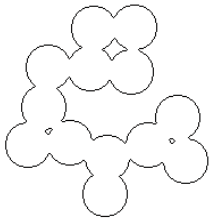
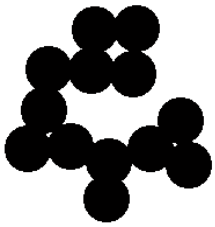
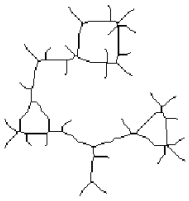
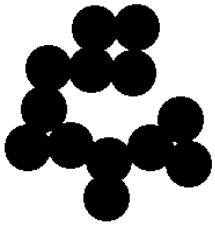
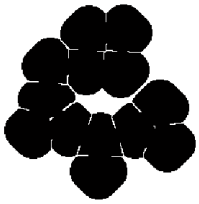
O altă comandă cu care se pot realiza operaţii morfologice este bwmorph. Sintaxa este: **`bwmorph(I,operatie)`** sau **`bwmorph(I,operatie,n)`**, I este imaginea asupra căreia se aplică operaţia specificată, iar dacă apare specificat n, operaţia se aplică de n ori. **`operatie`** este este un şir de caractere ce poate lua următoarele valori:

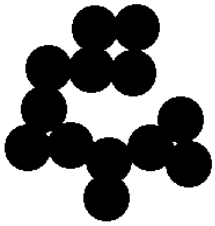
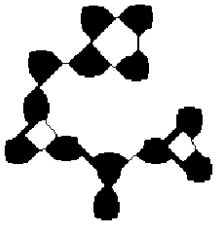
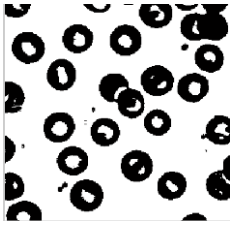
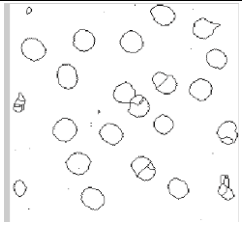
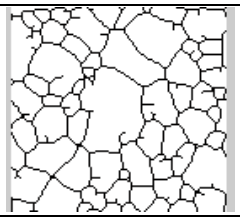
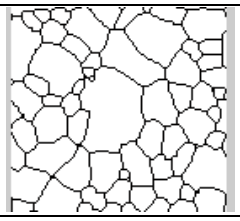
- dilate – face o dilatare cu elementul structural **`ones(3)`**;
- erode – efectuează o erodare cu elementul structural **`ones(3)`**;
- close – efectuează o operaţie de închidere binară (dilatare, urmată de erodare);
- open – efectuează o operaţie de deschidere binară (erodare, urmată de dilatare);
- clean – elimină pixelii izolaţi (un pixel 1, înconjurat de 0);
- fill – trece în stare ON pixelii din starea OFF, care sunt înconjuraţi de pixeli ON;
- majority – setează pixeli pe ON, dacă au în vecinătate (3x3) cel puţin 5 pixeli în stare ON;
- remove – trece în stare OFF pixelii ce au în vecinătatea 4x4 pixeli în stare ON, adică menţine doar pixelii aparţinători frontierelor;
- bothat – face diferenţa dintre o imagine şi imaginea rezultată prin închiderea ei;
- tophat – face diferenţa dintre o imagine şi cea rezultată prin deschiderea ei;
- bridge – conectează pixelii anterior neconectaţi;
- skel – efectuează scheletarea obiectelor, iar dacă n = Inf elimină pixelii din frontierele obiectelor, fără a permite separarea acestora;

- spur – elimină pixelii de capăt a liniilor, fără a elimina obiectele de dimensiuni mici;
- thicken – cu n = Inf, măreşte dimensiunea obiectelor prin adăugare de pixeli la exteriorul acestora, fără a permite conectarea obiectelor anterior neconectate;
- thin – cu n = Inf, îndepărtează pixeli aparţinători obiectelor, astfel încât componenetele cu găuri sunt reduse la un inel, iar cele pline se reduc la minimum fără a se separa;
- shrink – cu n = Inf, reduce componentele până la un punct, iar cele cu găuri le transformă în inele;
- diag – efectuează o umplere pe diagonală pentru a elimina conectivitatea de 8 a fundalului;
- hbreak – îndepărtează pixelii conectaţi în H.

Rezultatul este o matrice binară. În tabelul 6. 3 sunt prezentate câteve exemple de operaţii morfologice executate cu funcţia bwmorph.

Tabel 6.3 Operaţii morfologice

operaţia	Imaginea iniţială	Imaginea finală
remove		
skel		
thicken		

thin		
shrink		
spur		

Un aspect demn de semnalat la aplicarea succesivă a operaţiilor de dilatare şi erodare este modificarea formei obiectului iniţial. Pe o direcţie de 45°, fenomenul apare mai pregnant, datorită spaţierii mai mari a pixelilor. În primele două imagini din fig. 6. 38 se prezintă rezultatul a 50 de dilatări succesive şi a 50 de erodări succesive aplicate unui cerc. Zona adăugată, respectiv eliminată este reprezentată cu alb.

În varianta tradiţională, ambele operaţii, erodare şi dilatare, modifică starea unui pixel, dacă în vecinătatea specificată există un singur pixel aflat în starea opusă. Se pot concepe şi alte reguli cum ar fi: se contorizează numărul de pixeli din vecinătate care sunt în stare opusă şi se compară cu o valoare impusă, iar modificarea pixelului central are loc numai când valoarea este depăşită. Ex.: un coeficient 7 va determina înlăturarea pixelilor izolaţi, un coeficient 5 sau 6 va înlătura linii de pixeli ce apar la separarea anumitor regiuni.

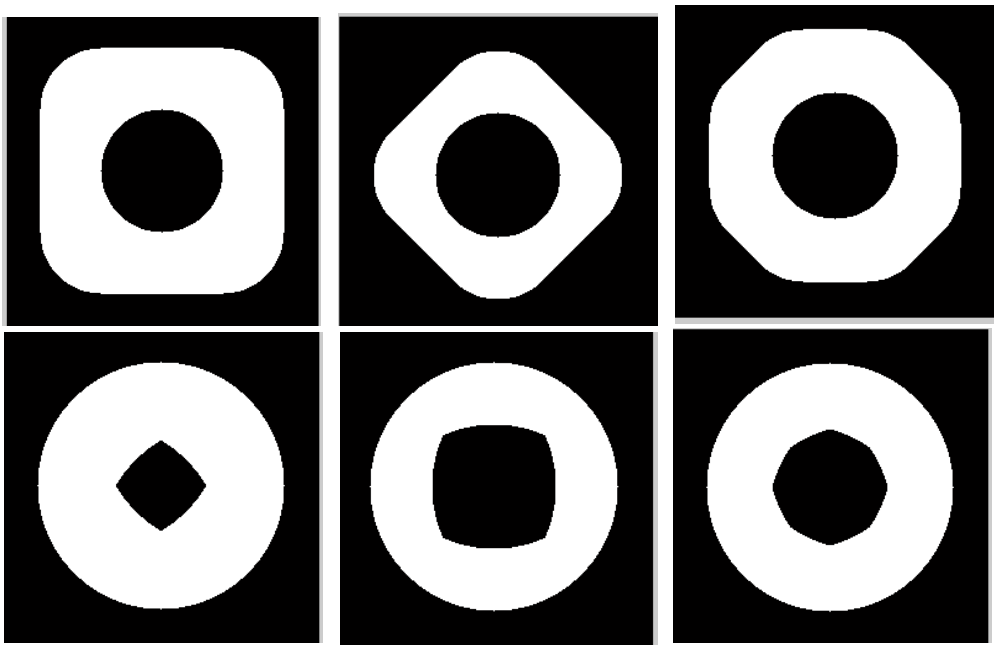


Fig. 6.38. Distorsiunea formei obiectelor la dilatare şi erodare: dilatare cu coeficient 0, cu coeficient 1, cu coeficient 0 şi 1 alternativ; erodare în aceleaşi condiţii

Utilizarea tabelelor lookup presupune calcularea valorii de ieşire pentru toate configurațiile posibile ale vecinătății și stocarea rezultatului într-un tabel. Tabelul, o dată calculat, se reține într-o variabilă declarată persistent. Acest tip de variabilă se aseamănă cu variabilele globale, deci este accesibilă pentru toate funcțiile, dar valoarea stocată nu poate fi modificată decât din funcția de unde a fost declarată.

Într-o vecinătate 3x3 binară pot exista 2^9 = 512 configurații posibile. Fiecare configurație trebuie să fie asociată univoc o anumită valoare și o variantă simplă este asocierea unui număr binar cu fiecare configurație:

2^0 2^3 2^6
2^1 2^4 2^7
2^2 2^5 2^8

În acest mod o anumită vecinătate se va asocia cu o anumită valoare. De ex.

1 1 1
1 0 1
1 0 1

Reprezintă: 1+1(2)+1(4)+1(8)+0(16)+0(32)+1(64)+1(128)+1(256) = 367

MATLAB are implementate două funcții pentru această tehnică makelut și bwlookup. Funcția makelut construiește un tabel lookupn pe baza unei funcții introduse de utilizator, iar funcția bwlookup prelucrează imaginea binară pe baza

acestui tabel. Funcșia ce trebuie furnizată de utilizator este o funcție ce are ca date de intrare o matrice bunară 3 x 3 și returnează o singură valoare, 0 sau 1. Funcția makelut va rula funcția introdusă pentru toate configurațiile posibile și stochează rezultatul într-un vector cu 512 elemente. Funcția bwlookup prelucrează imaginea, trecând prin toate vecinătățiile și înlocuiește pixelul central cu valoarea citită din tabel.

Se remarcă distorsiunea formei ce apare la aplicarea dilatării cu coeficient 0. Pentru a contracacra acest neajuns se poate aplica dilatarea, respectiv erodarea cu alternarea coeficienţilor: 0 şi 1. Modificarea coeficienţilor la aceste operaţii se poate realiza cu ajutorul unor tabele: look-up tables (lut). Prin intermediul lor se pot implementa mai uşor anumite operaţii morfologice ţinând seama de vecinătăţi. Aceste tabele definesc setarea bitului central în cazul unor vecinătăţi de 4, respectiv 8. De exemplu, în cazul când se doreşte setarea bitului central ON, în cazul când cel puţin doi pixeli din vecinătate sunt în stare ON (echivalentul coeficientului 1 la operaţia de dilatare), definirea tabelului se face: **`lut = makelut('sum(x(:)) >= 2', 3)`** pentru vecinătate de 8, respectiv **`lut = makelut('sum(x(:)) >= 2', 2)`** pentru vecinătate de 4. Operaţia se execută asupra imaginii I cu comanda: bwlookup(I,lut).

Codul ce a generat fig. 6.38 este:

```
%%dilatare_lut.m
%% generare imagine cu cerc raza 50
I=zeros(256);
I(128,128)=1;
for i=1:256
    for j=1:256
        if sqrt((i-128)^2+(j-128)^2)<=50
            I(i,j)=1;
        end
    end
end
%% dilatare I de 50x
b=bwmorph(I,'dilate',50);
%% generare cerc de raza 100
A=zeros(256);
A(128,128)=1;
for i=1:256
    for j=1:256
        if sqrt((i-128)^2+(j-128)^2)<=100
            A(i,j)=1;
        end
    end
end
%% erodare A de 50x
c=bwmorph(A,'erode',50);
```

```
%% evidentiere suprafata adaugata, respectiv scazuta din imagine
q=(~I&b)|(I&~b);
figure,imshow(q),title('50 dilatari')
t=(~A&c)|(A&~c);
figure,imshow(t),title('50 erodari')

 %% dilatare controlata de numarul de pixeli din vecinatate pe
baza de tabel lookup
lut = makelut('sum(x(:)) >= 2', 3);
test=I;
for i=1:50
    s=bwlookup(test,lut);
    im=test|s;
    test=im;
end

t=(~I&test)|(I&~test);
figure,imshow(t),title('50 dilatari cu coeficient 1')
test=I;
for i=1:25;
    BW2=bwmorph(test,'dilate',1);% dilatare coeficient 0
    s=bwlookup(BW2,lut);% dilatare coeficient 1
    im=test|s;
    test=im;
end
%figure,imshow(test)
q=(~I&test)|(I&~test);
figure,imshow(q),title('50 dilatari cu alternanta 0,1')
%% erodarea se face prin dilatarea ~imagine
A1=~A;
for i=1:50
    A2=bwlookup(A1,lut);
    aux=~A|A2;
    A1=aux;
end
A3=(A1&A)|(~A1&~A);figure,imshow(A3),title('50 erodari cu
coeficient 1')
test=A;
for i=1:25
    BW=bwmorph(test,'erode',1);
    A4=~BW;
    A5=bwlookup(A4,lut);
    A6=A4|A5;
    test=~A6;
end
aux=(~test&A)|(test&~A);
figure,imshow(aux),title('50 erodari cu coeficient 0,1')
```

Erodarea aplicată de n ori (cu un coeficient 0, 1 sau cu alternarea lor) produce îngustarea radială cu aproximativ n pixeli (într-un mod ce depinde de forma obiectului iniţial). Acest fapt determină dispariţia obiectelor ce sunt mai mici de 2n pixeli pe o anumită direcţie. Prin numărarea obiectelor ce dispar (respectiv scăderea numărului obiectelor rămase din numărul iniţial de obiecte) se poate determina numărul obiectelor mai mici decât mărimea corespunzătoare numărului de pixeli. Metoda poate fi aplicată numai la obiecte având o formă regulată. În cazul unor obiecte concave sau de formă neregulată, prin erodare, ele se pot separa în mai multe părţi componente, ceea ce conduce la rezultate irelevante. Pentru a contracara acest neauns se poate aplica o operaţie ŞI logic pe grupuri de pixeli, în sensul că se contorizează doar acele obiecte din imaginea iniţială, ce sunt atinse de de obiecte din imaginea erodată.

Pe baza dilatării succesive se poate determina distanţa dintre două obiecte.

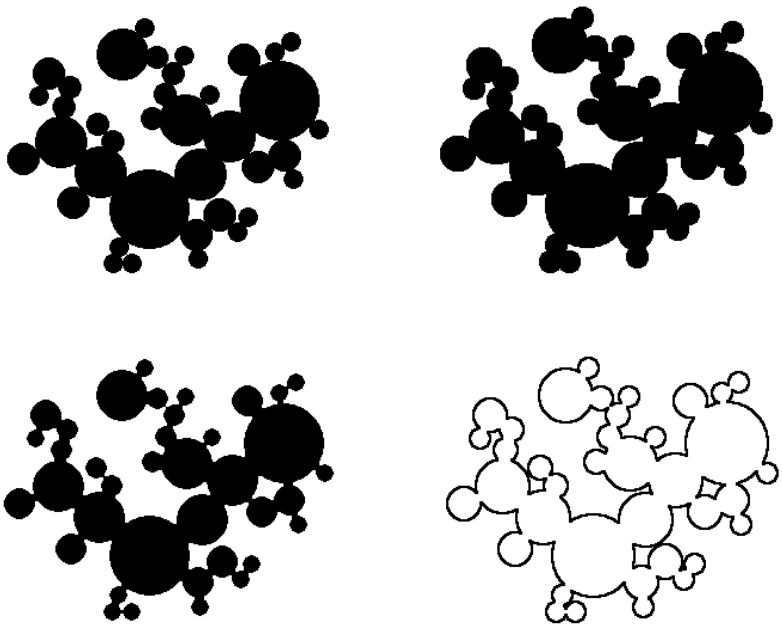


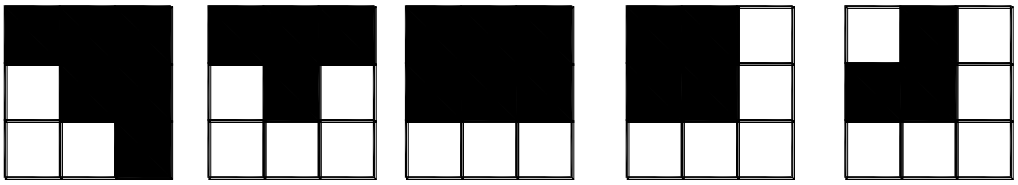
Fig. 6.39. Determinarea conturului obiectelor prin dilatare şi erodare

În paragraful anterior a fost prezentată o metodă de prelucrare prin convoluţie, la care se înlocuieşte valoarea pixelului central cu valoarea maximă sau minimă a pixelilor din vecinătate. Această operaţie se mai numeşte dilatare sau erodare pe niveluri de gri. În baza unui astfel de procedeu se poate determina numărul de obiecte de un anumit fel, ce apar într-o imagine, prin contorizarea

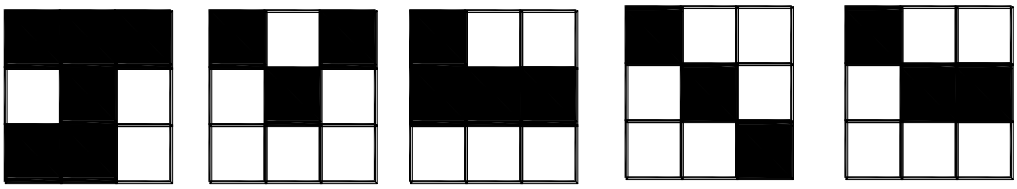
obiectelor ce dispar la fiecare iteraţie.

Erodarea şi dilatarea se aplică deseori împreună cu operaţii booleene, etichetări de obiecte şi umpleri. Prin combinarea cu SAU exclusiv a imaginilor obţinute prin erodare şi dilatare se poate obţine conturul unor obiecte. Spre deosebire de aplicarea funcţiei bwmorph, cu opţiunea remove, în acest caz se obţine un cotur format din două rânduri de pixeli, deci mai îngroşat. Această operaţie este evidenţiată în fig. 6. 39.

Erodarea poate fi realizată cu ajutorul unor reguli speciale, ce nu permit înlăturarea pixelului central în cazul când o regiune se separă în două. Câteva tipare de vecinătate, ce permit sau nu înlăturarea pixelului central sunt prezentate în fig. 6. 40.



Se permite inlaturarea pixelului central



Nu se permite inlaturarea pixelului central

Fig. 6.40 Tipare de vecinătate ce permit sau nu modificarea pixelului central în cazul erodării pentru a împiedeca separarea unui obiect

Prin intermediul unei astfel de erodări se obţine scheletarea imaginii, ce reprezintă un factor de formă puternic în recunoaşterea caracteristicilor, deoarece conţine atât informaţii topologice, cât şi metrice. Valorile topologice include: numărul de puncte de capăt, numărul de noduri (unde se întâlnesc mai multe ramuri) şi numărul de goluri (interioare). Valorile metrice sunt: lungimea ramurilor (atât a celor interioare, cât şi a celor libere).

Localizarea nodurilor şi a punctelor de capăt este o simplă problemă de numărare a vecinilor. Un punct de capăt are un singur vecin, iar un nod mai mult de doi. Lungimea segmentelor (ramuri) poate fi determinat prin numărarea perechilor de pixeli ce sunt conectaţi ortogonal sau diagonal. Principala problemă a scheletării este sensibilitatea acesteia la modificări minore în forma unui obiect. În cazul obiectelor lungi şi subţiri se obţine o bună reprezentare a formei, însă în

cazul obiectelor cu dimensiuni comparabile pot apărea probleme, lucru evidenţiat în fig. 6.41.

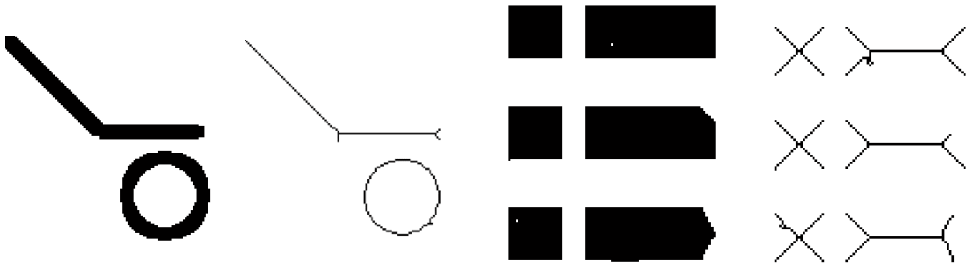


Fig. 6.41. Exemple de scheletare

O aplicaţie a scheletării este subţierea frontierelor, care au grosimi diferite. Acest fenomen este uzual la imagini microscopice, când frontierele sunt evidenţiate prin procese chimice. Pentru a putea măsura mărimea cristalelor, lungimea frontierelor etc. este recomandabil să se subţieze frontierele prin scheletare.

În numeroase situații este util să se poată identifica anumite configurații de pixeli, de ex. pixeli izolați din imagine sau pixeli ce reprezintă extremități de linii. Transformarea hit – or –miss permite acest lucru. În cazul acestei treansformări se folosesc două elemente structurale, se1 și se2. Rezultatul acestei transformări este intersecția dintre erodarea imaginii cu elementul se1 și al erodării complementului imaginii cu se2.

Denumirea transformării provine de la modul cum este afectat rezultatul de cele două erodări. Primul element structural determină toate locațiile care se potrivesc cu acest element – punct lovit, iar al doilea element va determina pozițiile unde configurația nu se potrivește cu acesta – punct ratat. Cele două elemente se aleg, de regulă, astfel încât primul este identic cu configurația căutată, iar al doilea este complementul acestuia. Dacă există pixeli în elementul structural a căror valoare este indiferentă, ei pot fi setați pe 0.

Ex. Se doreşte găsirea unei configurații de pixeli în formă de cruce

0 1 0
1 1 1
0 1 0

Elementele structurale sunt $s1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ și $s2 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}$

Sintaxa este

```
bwhitmiss(i,se1,se2)
```

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	1	1	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	0	1	1	0	0
0	0	1	0	0	0	0	0	0	0	0	0	1	1	1	0
0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0
0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

a. Imagine inițială

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b. Imagine dupa erodare s1

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	0	0	0	0	1	1	1	1	1	1
1	0	0	0	1	1	1	1	1	1	1	1	0	0	1	1
1	1	0	1	1	1	1	1	1	1	1	1	0	0	0	1
1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1
1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1
1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

c. Complementul imaginii – (Imagine dupa erodare s2, valoarea 1 este marcată prin fundal gri)

Rezultatul transformării se obține aplicând ȘI logic între b. și c. Se vede că rezultatul transformării indică pixelii din (4,3) și (7,6).

Un alt exemplu de aplicare a transformării hit – or - miss este găsirea

colțurilor din stânga sus al unor obiecte de diferite dimensiuni, de formă rectangulară. Configurația ce trebuie găsită este de forma

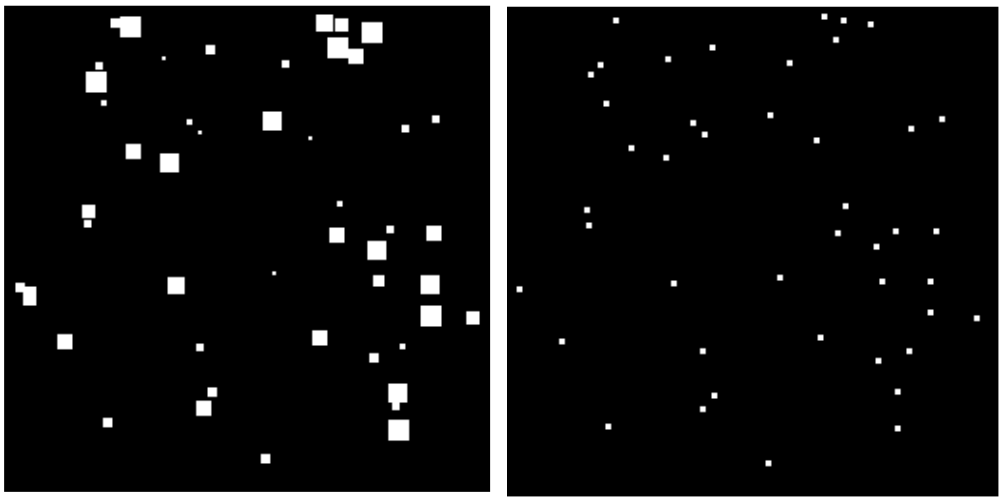
0 0 0
0 1 1
0 1 *

Asteriscul marchează un pixel a cărui valoare este indiferentă (în acest caz, în ambele elemente cu care se aplică transformarea, pixelul va avea valoarea 0). Cele două elemente structurale cu care se poate rezolva problema sunt:

B1=strel([0 0 0;0 1 1;0 1 0]);

B2=strel([1 1 1;1 0 0;1 0 0]);

Imagine initiala



```
I=zeros(256);
r=ceil(randi(255,1,50));
c=ceil(randi(255,1,50));
l=ceil(randi(10,1,50));
for k=1:50;
    linie=r(k)-l(k);
    coloana=c(k);
    lat=l(k);
    if (linie>0)&(coloana>0)
    I(linie:linie+lat,coloana:coloana+lat)=1;
    end
end
figure,imshow(I),title('Imagine initiala')
b1=strel([0 0 0;0 1 1;0 1 0]);
b2=strel([1 1 1;1 0 0;1 0 0]);
bw=bwhitmiss(I,b1,b2);
bw1=bwmorph(bw,'dilate',1);
```

```
figure,imshow(bw1),title('Transformare hit-or-miss si dilatare')
```

În cazul unor elemente structurale mai mici o metodă mai rapidă decât transformarea hit-or-miss este cea bazată pe tabele lookup (LUT).

Operaţiile şi funcţiile prezentate în cadrul acestui paragraf se aplică asupra unor imagini binare rezultând tot imagini binare sau asupra unor imagini de intensitate rezultând tot imagini de intensitate. Cu ajutorul distanţei euclidiene se pot prelucra imagini binare rezultând imagini de intensitate. Conceptul este foarte simplu: fiecărui pixel I se atribuie un nivel de gri egal cu distanţa până la cea mai apropiată frontieră. În cazul imaginilor digitale, apare o problemă datorită faptului că distanţa pe diagonală (în cazul vecinătăţii de 8) este mai mare decât distanţa pe direcţie ortogonală. Pentru a genera o astfel de imagine se percurge următorul algoritm:

- se atribuie valoarea 0 fiecărui pixel aparţinând fundalului;
- se setează o variabilă `gri:=0;`
- fiecărui pixel ce este în contact cu fundalul I se atribuie valoarea `gri:=gri+1;`
- se incrementează variabila gri şi se repetă pasul anterior până când toţi pixelii din imagine au primit o valoare.

Timpul necesar aplicării unui astfel de algoritm depinde de mărimea obiectelor ce există în imagine. O variantă mai eficientă este [Russ]:

- se atribuie valoarea 0 fiecărui pixel din fundal şi o valoare mare (mai mare decât lungimea maximă a obiectelor din imagine) pentru restul pixelilor;
- de la stânga la dreapta şi de sus în jos se atribuie fiecărui pixel aparţinător prim-planului o valoare mai mare cu 1 decât cea mai mică valoare din vecinătate;
- se repetă pasul doi, dar imaginea se parcurge de la dreapta spre stânga şi de jos în sus.

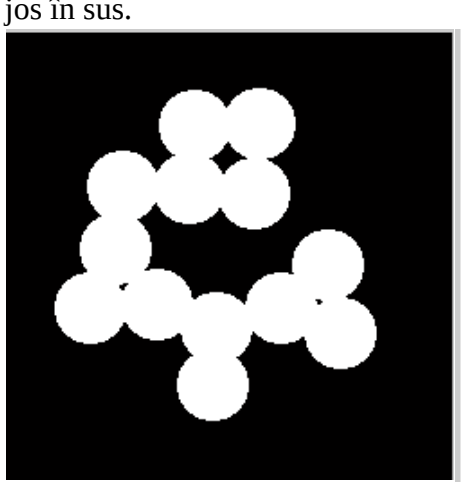


Fig. 6. 42. Reprezentarea distanţei euclidene

O dificultate ce poate apare la măsurarea imaginilor este contactul dintre diferite obiecte, ce determină probleme la identificarea lor în vederea numărării sau măsurării. O metodă de separare a obiectelor în contact este segmentarea pe baza liniei de contur (watershed). Ea se bazează pe observaţia că erodarea determină separarea obiectelor înaintea dispariţiei lor.

Metoda clasică de segmentare este una iterativă. Imaginea este erodată succesiv şi la fiecare pas, obiectele care dispar din imagine se consideră ultimele puncte erodate şi se salvează ca imagine împreună cu numărul iteraţiei. Procesul continuă până în momentul dispariţiei tuturor obiectelor din imagine. În faza următoare , plecând de la ultima imagine se aplică o dilatare, la care se impune şi o restricţie logică: nici un pixel nu se trece în stare ON, dacă determină apariţia unei conexiuni între obiecte diferite sau a fost în stare OFF în imaginea iniţială. La fiecare iteraţie, imaginea ultimelor puncte erodate, corespunzătoare iteraţiei respective, se combină cu imaginea prin SAU logic. Acest proces determină ca obiectele să revină la dimensiunile iniţiale, cu excepţia faptului că apar linii de separare între obiectele aflate în contact.

În Matlab acest proces se poate realiza cu funcţia watershed. Sintaxa este: `A=watershed(I,vecinatate)`. Vecinătatea implicită este 3x3. Matricea I poate fi binară sau numerică, iar matricea de ieşire, L, este o matrice numerică, ce conţine doar valori întregi pozitive sau egale cu 0. Pixelii ce au valoarea 0 aparţin zonei de demarcaţie dintre obiecte. Pixelii cu valoarea 1 aparţin primei regiuni din imagine, cei cu valoarea 2, altei regiuni etc. În general, funcţia se aplică matricei ce conţine distanţa euclideană a unei imagini. În fig. 6.42 se prezintă o imagine, distanţa euclideană a acesteia şi rezultatul aplicării funcţiei watershed.

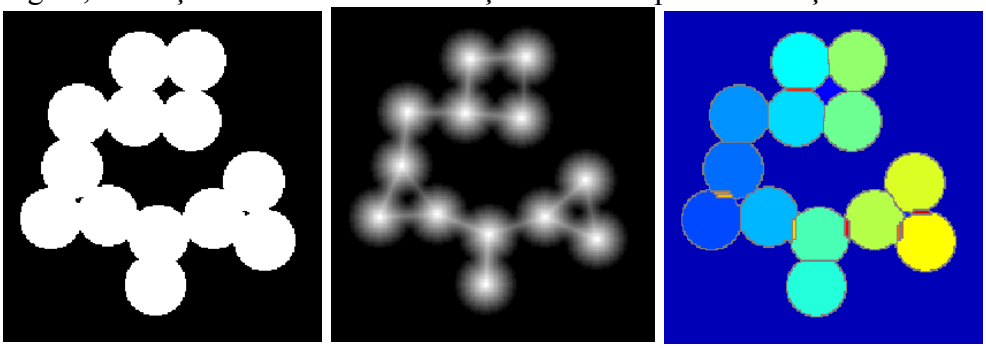


Fig. 6.42. Segmentarea pe baza liniei de contur

6.2.6. Măsurări ale imaginii

Măsurarea imaginii presupune o prelucrare anterioară a imaginii:

corectarea defectelor, îmbunătăţirea vizibilităţii unor structuri particulare, delimitarea obiectelor faţă de fundal etc., care au fost prezentate în paragrafele anterioare.

Măsurările pot fi grupate în 4 clase, în funcţie de măsurand:

- măsurări de intensitate;
- măsurări de localizare;
- măsurări de mărime;
- măsurări de formă.

În mod normal, în cazul tipurilor de imagine prezentate în acest capitol, fiecare pixel înregistrează o valoare numerică, ce reprezintă intensitatea luminoasă a unui punct din imagine sau există mai multe valori care se combină pentru a obţine o imagine color. Domeniul uzual este între 0 şi 255.

Determinarea poziţiei unui obiect din imagine poate fi făcută în mai multe moduri. O variantă este determinarea punctului central (aflat la mijlocul distanţei dintre pixelul minim şi maxim pe două direcţii). Această variantă poate fi aplicată în cazul când obiectul are formă regulată. De fapt, prin această metodă se aproximează forma obiectului cu un dreptunghi şi punctul central este cel aflat la intersecţia diagonalelor. O variantă aplicabilă în cazul oricărei forme este determinarea centrului de greutateşi care se recomandă

$$x_{cg} = \frac{\sum x_i}{arie} \quad y_{cg} = \frac{\sum y_i}{arie}, \tag{6.15}$$

unde arie este numărul total de pixeli aparţinători obiectului.

Centrul de greutate se poate calcula şi pe baza punctelor aparţinătoare frontierei obiectului, cu formula [Russ]:

$$\begin{aligned} x_{cg} &= \frac{\sum_i (x_i + x_{i+1})^2 \cdot (y_i - y_{i-1})}{a} \\ y_{cg} &= \frac{\sum_i (y_i + y_{i+1})^2 \cdot (x_i - x_{i-1})}{a} \end{aligned}, \tag{6.16}$$

unde x_0, y_0 şi x_n, y_n reprezintă acelaşi punct (conturul obiectului este închis), iar a este:

$$a = \frac{\sum (x_i + x_{i+1})(y_i - y_{i+1})}{2}, \tag{6.17}$$

dar varianta cea mai sigură este determinarea centrului de masă pe baza tuturor punctelor aparţinătoare obiectului (6.15).

Pentru determinarea poziţiei obiectului coordonatele centrului de masă nu sunt suficiente, fiind necesară şi determinarea orientării obiectului. Pentru orientarea se pot defini mai multe mărimi, printre care direcţia celui mai lung

segment aparţinător obiectului. Dar, la fel cu cazul precedent, pentru caracterizarea orientării se recomandă utilizarea axei de inerţie (axa faţă de care momentul de rotaţie este minim sau axa faţă de care suma pătratelor distanţelor de la fiecare pixel la axă este minimă). Axa de iner'ie este, de fapt, dreapta de regresie asociată obiectului. Determinarea orientării acestei axe se poate face cu următorul set de ecuaţii:

$$\begin{aligned} M_{xx} &= S_{xx} - \frac{S_x^2}{arie} \\ M_{yy} &= S_{yy} - \frac{S_y^2}{arie} \\ M_{xy} &= S_{xy} - \frac{S_x \cdot S_y}{arie} \\ \theta &= arctg\left(\frac{M_{xx} - M_{yy} + \sqrt{\left(M_{xx} - M_{yy}\right)^2 + 4M_{xy}^2}}{2M_{xy}} \right) \end{aligned} \quad , \qquad (6.18)$$

unde M sunt momentele faţă de axele de coordonate, θ este unghiul format de axă cu orizontala, iar $S_x = \Sigma x_i$, $S_y = \Sigma y_i$, $S_{xx} = \Sigma x_i^2$, $S_{yy} = \Sigma y_i^2$, $S_{xy} = \Sigma x_i y_i$.

În unele aplicaţii, localizarea anumitor obiecte este mai puţin importantă decât distanţa dintre ele. S-a demonstrat în literatură, [Russ], că histograma distribuţiei distanţelor faţă de cel mai apropiat vecin poate descrie distribuţia obiectelor. Această distanţă faţă de cel mai apropiat vecin se poate determina cu uşurinţă după ce s-a calculat centrul de masă al fiecărui obiect. Distribuţia acestei distanţe poate fi caracterizată prin medie şi deviaţie standard. Trebuie menţionat faptul că acele obiecte ce sunt situate în apropierea extremităţilor câmpului imagine (la o distanţă mai mică decât cel mai apropiat vecin), nu trebuie luate în considerare. În cazul când obiectele au o distribuţie aleatoare, distanţa medie dintre cei mai apropiaţi vecini este [Russ]:

$$\bar{d} = \frac{0.5}{\sqrt{N/arie}} , \qquad (6.19)$$

unde N este numărul de obiecte din imagine, iar arie este aria câmpului vizual. Deviaţia standard este:

$$S = \sqrt{\bar{d}} , \qquad (6.20)$$

ceea ce înseamnă că în cazul unei distribuţii aleatoare a obiectelor, prin determinarea numărului de obiecte / unitatea de suprafaţă se poate caracteriza distribuţia obiectelor.

Prin găsirea celui mai apropiat vecin se poate determina şi izotropia în distribuţia obiectelor. În acest scop trebuie determinată direcţia spre cel mai

apropiat vecin. În cazul unei distribuţii izotrope, direcţia spre cel mai apropiat vecin trebuie să fie o funcţie uniformă în raport cu unghiul.

Problema numărării obiectelor dintr-o imagine este una din cele mai uzuale proceduri din analiza imaginii. Fiecare obiect trebuie să fie definit de câte un singur punct şi aceste puncte trebuie numărate. Când câmpul vizualizat este o parte a unei structuri, rezultatele se prezintă sub forma număr/unitate de suprafaţă. Dintre obiectele ce intersectează extremităţile câmpului vizual se iau în considerare doar cele ce ating două din margini (una dintre marginea de sus sau jos, respectiv una dintre cea din stânga sau dreapta). De ex., selectarea marginii superioare şi a celei din stânga este echivalentă cu contorizarea obiectelor prin pixelul aparţinător situat în partea inferioară-dreaptă.

În cazul când se fac şi măsurări ale obiectelor, problema este puţin mai complicată. Nu se pot măsura obiectele ce intersectează una din extremităţi, deoarece nici o informaţie legată de formă, mărime sau poziţie nu poate fi corect determinată. În cazul când se măsoară doar obiectele ce nu atingnici o extremitate a câmpului imagine, nu vor fi luate în consederare o serie de dimensiuni mari, ceea ce ar denatura rezultatele. Din acest motiv se defineşte un câmp de măsurare (marcat cu linie cotinuă în fig. 6.43) şi vor fi măsurate şi contorizate, toate obiectele ce se află în întregime în interiorul zonei respective împreună cu acele obiecte ce intersectează extremitatea stângă şi superioară a zonei marcate. Aceste obiecte se află în întregime în câmpul vizual, astfel încât informaţiile obţinute sunt corecte. Evident contorizarea, respectiv măsurarea trebuie raportată la aria suprafeţei marcate.

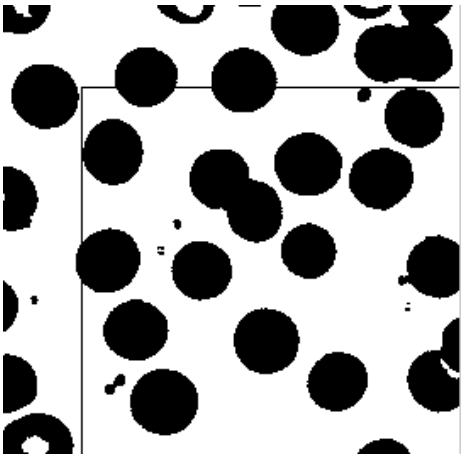


Fig. 6.43. Definirea unui câmp de măsurare

În unele situaţii, în cazul unor imagini particulare, trebuie imaginate anumite metode specifice de numărare. De ex., în cazul unei imagini de fibre ce

se intersectează unele cu altele, cum pot apare în mostre biologice, alimentare, textile etc., din punct de vedere al prelucrării imaginii, fibrele sunt identificate ca un singur obiect ce atinge toate extremităţile câmpului vizual. Într-un astfel de caz se poate face scheletarea imaginii şi se contorizează toate capetele libere, N. Numărul de fibre va fi egal cu N/2. Împărţind acest număr la aria câmpului vizual rezultă numărul de fibre/unitatea de suprafaţă, iar dacă lungimea totală a fibrelor se împarte la numărul lor, se obţine lungimea medie a fibrelor.

Pentru determinarea mărimii obiectelor dintr-o imagine, cea mai uzuală mărime este aria, care poate fi calculată prin numărul pixelilor aparţinători obiectului respectiv. Trebuie menţionat faptul că această mărime din spaţiul bidimensional trebuie corelată cu mărimea din spaţiul 3D. Această corelaţie poate fi stabilită ţinând cont de modul de obţinere a imaginii (prin proiecţie sau secţiune), iar această corelaţie se face pe baza stereologiei, ştiinţa ce se fundamentează pe calcule probabilistice.

Determinarea ariei este, de fapt, o operaţie de contorizare a pixelilor aparţinători suprafeţei, care trebuie asociaţi cu un factor de conversie dintre mărimea pixelului şi dimensiunea din lumea reală. În general acest lucru se realizează prin capturarea unei imagini de la un obiect de dimensiuni cunoscute. În unele situaţii, după determinarea ariei se calculează un diametru echivalent:

$$d_{eq} = \sqrt{\frac{4}{\pi} \cdot arie} \,. \tag{6.21}$$

Sunt destul de frecvente situaţiile când nu trebuie determinată aria unei suprafeţe aparţinătoare unui obiect individual. Dacă imaginea provine din secţionarea unui solid, una din regulile stereologiei, afirmă că fracţia de volum a unei faze (orice structură sau componentă identificată) este egală cu fracţia de arie a fazei respective determinate printr-o secţiune aleatoare.

O altă mărime ce poate caracteriza un obiect este dimensiunea de gabarit. Ea se determină pe orizontală şi verticală, prin sortarea pixelilor pentru a găsi poziţia minimului şi maximului. Ulterior se roteşte sistemul de coordonate cu un număr relativ mic de paşi pe 180° (se recomandă 16 paşi, adică 11.25°/pas, [Russ]) şi procedeul se repetă, obţinând un poligon circumscris obiectului. În acest poligon, se determină lungimea maximă şi minimă între vârfuri. În multe situaţii dimensiunea maximă a poligonului este considerată lungimea maximă, deoarece aproximează destul de bine cea mai mare distanţă ce poate fi construită între două puncte aparţinătoare obiectului.

Prin rotirea sistemului de coordonate în vederea găsirii valorilor minime şi maxime ale pixelilor, se calculează noile sisteme de coordonate:

$$x' = x\cos\alpha - y\sin\alpha$$
$$y' = y\sin\alpha - x\cos\alpha \quad . \tag{6.22}$$

Dimensiunile de gabarit caracterizează bine un obiect rigid, dar în cazul unui obiect elastic sau dacă imaginea este preluată de la un obiect în mişcare (deci este o imagine aleatoare) este mai corectă caracterizarea obiectului prin lungimea şi lăţimea fibrei. Determinarea lungimii fibrei înseamnă mai mult decât simpla numărare a pixelilor, datorită faptului că într-o reţea de pixeli, distanţa depinde de direcţia în care apare contactul. S-a demonstrat că este mai corect să se calculeze lungimea fibrei cu formula [Russ]:

$$L = 0.948 \cdot nr\_pixeli\_orto + 1.34 \cdot nr\_pixeli\_diag \,, \tag{6.23}$$

unde nr_pixeli_orto reprezintă numărul de contacte ce se stabilesc pe orizontală şi verticală, iar nr_pixeli_diag este numărul de contacte pe diagonală. Cu această formulă se obţine o eroare medie de 2.5%.

În cazul când se foloseşte distanţa euclidiană pentru determinarea axei mediane, nivelurile de gri asociate pixelilor reprezintă, de fapt, cercul înscris în obiect. Prin medierea acestor valori se poate obţine lăţimea fibrei.

Determinarea perimetrului unui obiect nu este o problemă simplă. În cazul când obiectul este reprezentat prin frontiera sa, perimetrul paote fi exprimat:

$$p = \sum_i \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \quad . \tag{6.24}$$

Forma obiectelor reale este foarte diversă şi este dificil de găsit caracteristici ce să permită clasificare şi compararea lor. Între primii descriptori de formă utilizaţi se numără raportul a două dimensiuni, deci factori adimensionali, cum ar fi raportul lungime/lăţime. Descriptorii de formă nu sunt standardizaţi. În literatură apar sub diferite variante şi denumiri. În tabelul 6. 4 sunt reprezentaţi câţiva din cei mai reprezentativi.

Tabelul 6.4. Descriptori de formă

Nr.crt.	Denumirea	Expresia	Obs.
1.	Factor de formă	$\frac{4\pi \cdot arie}{perimetru^2}$	Circularitate 1
2.	Rotunjimea	$\frac{4 \cdot arie}{D_{max}^2}$	D_{max} –diamtrul maxim
3.	Raport de formă	D_{max} / D_{min}	D_{min} – diametrul minim
4.	Alungirea	$lungime\_fibra / latime\_fibra$	
5.	Factor de buclare (curl)	$lungime / lung\_fibra$	Lungime – segmentul cel mai

			lung ∈ obiectului
6.	Convexitate	$Perim\_convex / Perim$	Perim_convex – perimetrul poligonului circumscris
7.	Soliditatea	$arie / arie\_convexa$	Arie_convexa – aria poligonului circumscris
8.	Compactitatea	$\frac{\sqrt{4arie/\pi}}{D_{\max}}$	
9.	Factor de variaţie	$D_{inscris} / D_{\max}$	$D_{inscris}$ – diamtrul cercului înscris

În Matlab, dintre mărimile de măsurare şi factorii de formă prezentaţi, se poate calcula aria unei imagini binare cu funcţia bwarea, având sintaxa **`bwarea(I)`** şi calculează numărul pixelilor în stare ON din imaginea I.

Mai exista funcţia bwperim, cu sintaxa bwperim(I,vecinatate), care returnează o imagine binară cu pixelii aparţinători frontierei obiectelor, deci cu care se poate calcula mai uşor perimetrul.

O altă funcție implementată pentru determinarea unor proprietăți ale imaginii este **`regionprops(BW,prprietati)`**, BW este o imagine binară sau funcția se poate aplica și matricei obținută în urma etichetării imaginii. Printre proprietați se menționează Area, boundingBox, centroid, EquivDiameter, Perimeter, Orientation.

Prelucrarea culorii

Pentru ființa umană, culoarea reprezintă cel mai important descriptor al lumii înconjurătoare. Vederea umană receptează preponderent muchii și culori.

Lumina vizibilă este parte a spectrului elecromagnetic, radiațiim în care energia se materializează sub formă de unde, cu diferite lungimi de undă. Acest domeniu variază de la radiația cosmică, cu lungimi de undă foarte mici, până la lungimi foarte mari, ce apar în electricitate. Domeniul vizibil este între $1 \cdot 10^{-7}$ m (albastru) și 70^{-7} m (roșu).

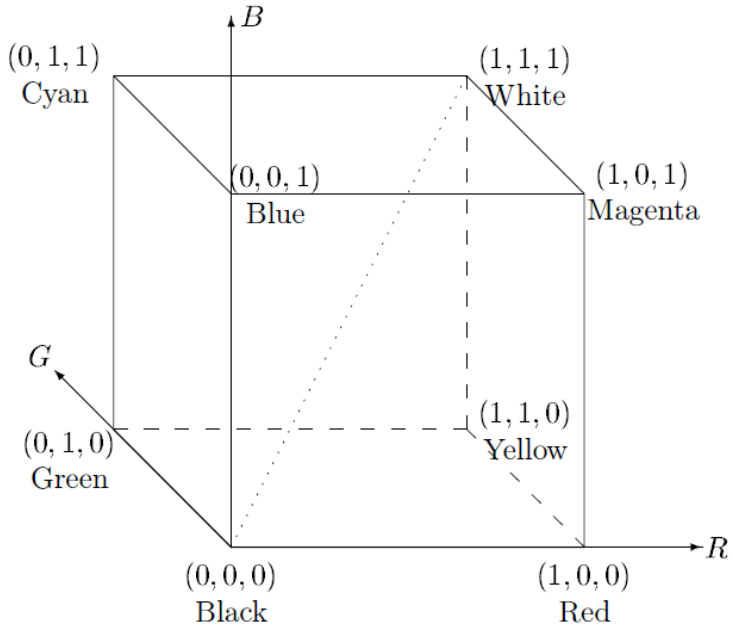
Vederea umană are tendința de a percepe culoarea pe baza combinării culorilor fundamentale, RGB, care se consideră culori primare. Prin combinarea a două culori primare se obțin culorile secundare:

Magenta = red + blue

Cyan = green + blue

Yellow= red + green

Un model de culoare este o metodă de specificare a culorii într-un mod standardizat. De regulă, constă dintr-un sistem de coordonate tridimensional și un subspațiu al acelui sistem, în care fiecare culoare este reprezentată printr-un singur punct.

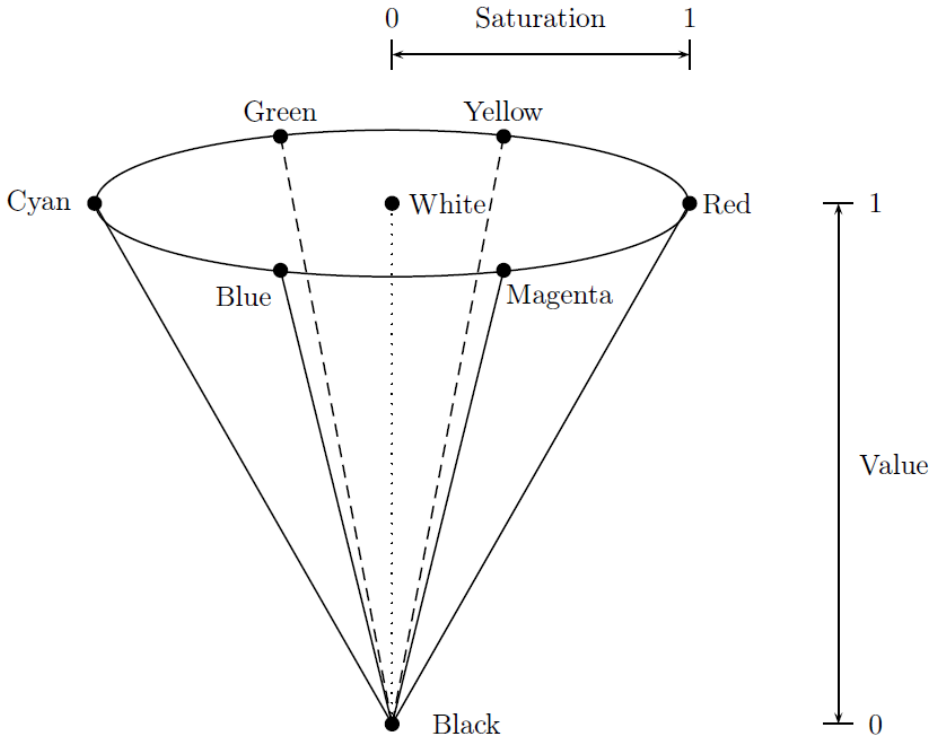


Diagonala cubului ce unește punctele (0, 0, 0) cu (1, 1, 1), unde toate punctele au aceeași cantitate de roșu, verde și albastru, reprezintă nuanțele de gri.

Un alt model de culoare este modelul INS intensitate - nuanță – saturație – valoare (HSV – Hue Saturation Value).

Nuanța reprezintă atributul de valoare. Saturația este cantitatea cu care culoarea a fost diluată cu alb. Cu cât cantitatea de alb crește, cu atât saturația scade. Deci, un roșu intens are saturație mare, iar roz are saturație mică. Valoarea este valoarea intensității luminoase. Cu cât culoarea este mai luminoasă, cu atât crește valoarea.

Acest model constituie o metodă mai intuitivă de descriere a culorii și având în vedere că intensitatea este independentă de informația de culoare este foarte utilă pentru prelucrarea imaginilor.



Modelul poate fi reprezentat sub forma unui con sau sub forma unei piramide cu baza hexagon.Înălțimea reprezintă de valoarea, iar raza bazei conului este saturația. Orice punct de pe suprafața conului este o culoare saturată, iar saturația reprezintă distanța relativă până la axa conului. Nuanța este reprezentată prin unghiul măsurat față de o axă prestabilită, de ex. roșu.

Nuanța se calculează ca raportul dintre lungimea arcului parcurs și lungimea cercului, astfel încât se pot specifica nuanțele pentru culori: roșu – 0, galben – 0.1667 (1/6), verde – 0.333, cyan – 0.5, albastru – 0.667, magenta – 0.8333.

Cunoscînd valoarea corespunzătoare RGB se poate face conversia între cele două modele. În MATLAB, conversia se face cu rgb2hsv([valR valG valB]), iar transformarea inversă se face cu hsv2rgb.

Un alt model de culoare este YIQ sau NTSC, ce se utilizează în Statele Unite la semnale video/TV. Avantajul utilizării acestui sistem este separarea informației de intensitate de cea de culoare, ceea ce permite ca acest semnal să poată fi utilizat atât pentru echipamentele xcolor, cât și cele monocrome. Y este luminanța, ce poate fi aproximată grosier cu intensitatea, iar I și Q conțin informații despre culoare. Acest model este o transformare liniară a modelului RGB, fegătura dintre ele fiind:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0.956 & 0.621 \\ 1 & -0.272 & -0.647 \\ 1 & -1.106 & 1.703 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$$

În MATLAB conversia între modelele RGB și NTSC se face cu funcțiile: rgb2ntsc() și ntsc2rgb().

Imaginile color sunt stocate în MATLAB în trei matrici separate, fie valorile RGB, fie HSV. Vizualizarea unei anumite componente se poate face solicitând o anumită dimensiune din imagine:

imshow(imagine(:,:,1)) → roșu

imshow(imagine(:,:,2)) → verde

imshow(imagine(:,:,3)) → albastru

Pseudocolorarea înseamnă atribuirea culorii unei imagini în nuanțe de gri, pentru a scoate în evidență anumite aspecte. Există mai multe metode de pseudocolorare:

a. împărțirea pe domenii – din cele 256 niveluri de gri, se atribuie culoarea pe grupuri ordonate.

Ex.

Nivel gri	0 - 63	64 - 127	128 - 191	192 - 255
Culoare	albastru	magenta	verde	roșu

b. transformarea pe bază de funcții

Se consider trei funcții $f_R(x)$, $f_G(x)$ și $f_B(x)$, ce atribuie valori fiecărei componente, iar rezultatul obținut, eventual scalat, dacă este necesar, se utilizează pentru afișare.

În MATLAB se poate colora o imagine prin introducerea unui parametru, colormap, la comanda imshow. Trebuie remarcat că o alegere nepotrivită de hartă a culorii poate strica imaginea.

În MATLAB există mai multe hărți de culoare, printre care:

- autumn – conține culori calde, de la galben până la roșu, cu modificare lentă;
- bone – nuanțe de gri cu o valoare mai ridicată pentru component albastru;
- cool – nuanțe ce se modifică lent, de la cyan la magenta;
- copper – conține nuanțe de la arămiu luminos până la negru;
- flag – conține culorile roșu, alb, albastru și negru, modificări mari la incrementare;

- gray – nuanțe de gri;
- spring – nuanțe de magenta și galben;
- summer – nuanțe de verde și galben;
- winter – nuanțe de albastru și verde.

Procesarea imaginilor color se poate clasifica în două mari categorii de operații:

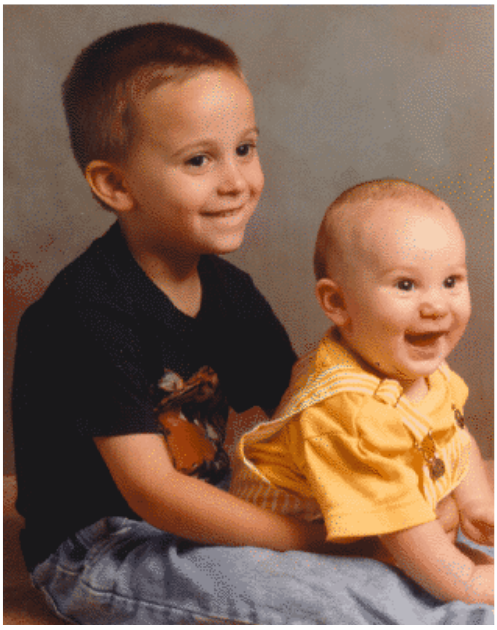
- se prelucrează fiecare matrice de culoare separate;
- se transformă spațiul culorii, separând culoarea de intensitate și se prelucrează doar intensitatea.

Îmbunătățirea contrastului se face mai bine prelucrând intensitatea. În cazul când este vorba de o imagine indexată, ea trebuie convertită în imagine RGB. Un exemplu se prezintă în continuare:

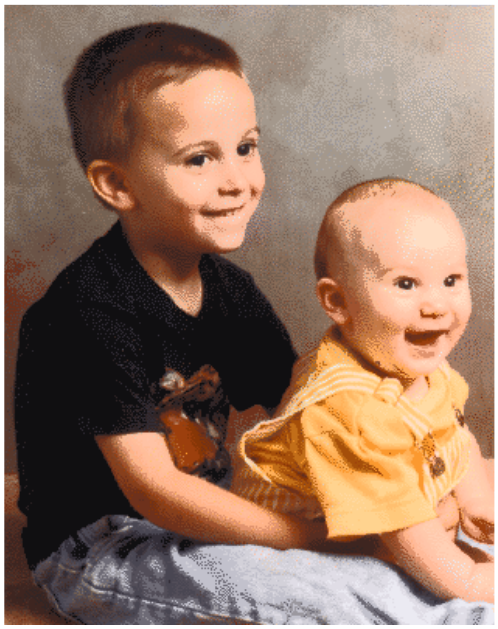
```
%% imbunatatire contrast
[x,map]=imread('kids.tif');
figure,imshow(x,map),title('Imagine initiala')
I=ind2rgb(x,map);
In=rgb2ntsc(I);% se face conversia in ntsc pt separare
intensitate
In(:,:,1)=histeq(In(:,:,1));% egalizare histograma pt
intensitate
Ip=ntsc2rgb(In);
figure,imshow(Ip),title('Egalizare histograma pentru
intensitate')
% egelizare histograma pentru fiecare culoare
Ir=histeq(I(:,:,1));
Ig=histeq(I(:,:,2));
Ib=histeq(I(:,:,3));
I3=cat(3,Ir,Ig,Ib);
figure,imshow(I3),title('Egalizare histograma pentru
fiecare culoare')
```

Pentru a egaliza histograma se face conversia în sistemul ntsc, unde prima matrice, din cele trei ale imaginii, conține informația de intensitate, separate de informația de culoare. În cazul când se aplică egalizarea histogramei pentru fiecare culoare în parte, cele trei rezultate se combină cu comanda cat, care face concatenarea matricelor după dimensiunea specificată ca prim argument.

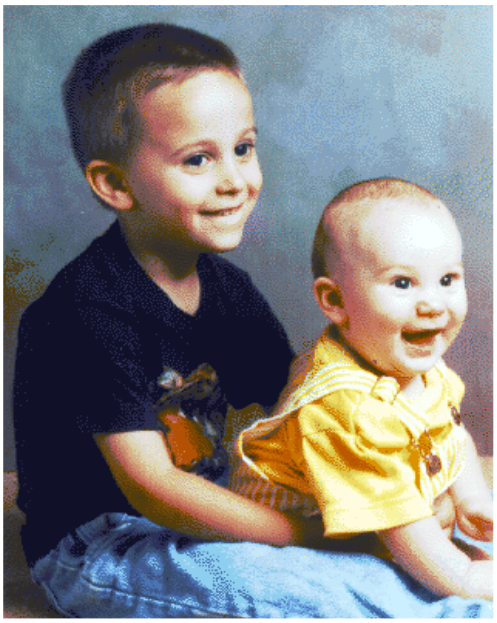
Imagine initiala



Egalizare histograma pentru intensitate



Egalizare histograma pentru fiecare culoare



Segmentarea imaginilor color este dificilă. Selectarea unui nivel de prag specificând componentele RGB nu este simplă. În primul rând intensităţile de roşu, verde şi albastru sunt determinate de modul de lucru al senzorilor de

imagine şi nu corespund felului în care sunt percepute de utilizatori. Percepţia culorilor este mai bine analizată în sistemul IHS (intensitate – nuanţă – saturaţie). Intensitatea măsoară energia radiantă totală, ca sumă a componentelor RGB; nuanţa defineşte principalul atribut al culorii şi poate fi considerată aproximativ proporţională cu lungimea de undă medie a spectrului; saturaţia defineşte proporţia de alb în componenţa culorii. Culorile saturate nu conţin lumină albă, ceea ce în reprezentarea RGB presupune absenţa a cel puţin uneia din cele trei componente.

O metodă mai bună este utilizarea unui prag bidimensional. Cel mai frecvent acest lucru se face în planul nuanţă – saturaţie. Acest plan este reprezentat în fig. 6.20 şi el se reprezintă sub forma unui cerc, unghiul măsurat în sens trigonometric fiind proporţional cu nuanţa, iar raza este proporţională cu saturaţia.

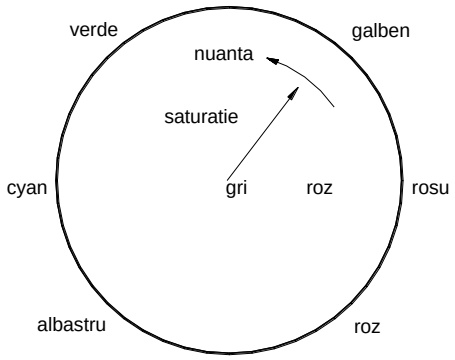


Fig. 6.20. Planul saturaţie – nuanţă [Russ]

Este posibil să se stabilească nivelul de prag folosind una din componentele IHS. Chiar în condiţiile când imaginea RGB se converteşte IHS şi se construiesc histogramele aferente, este foarte dificil să se stabilească pe baza a trei histograme nivelurile de prag corespunzătoare.

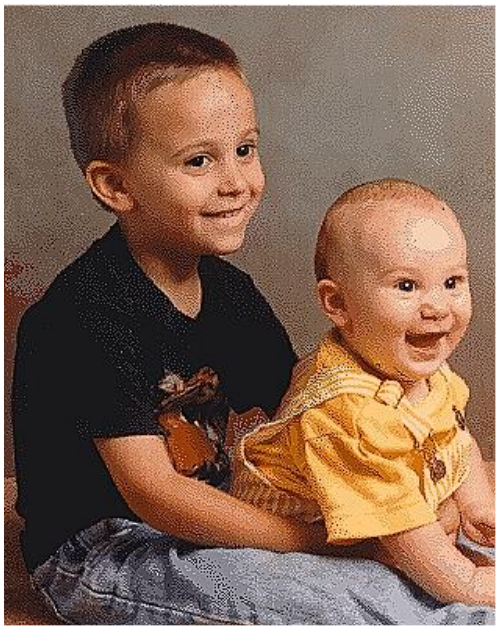
În cazul filtrării imaginilor color, metoda de abordare depinde de tipul filtrului aplicat. În cazul filtrelor trece jos se poate lucra atât pe componentele R, G, B, cât ţi pe cocmponenta de intensitate. Pentru un filtru trece sus se recomandă să se lucreze cu component de intensitate. Pentru exemplificare se aplică un filtru de estompare (trece jos) şi unul de accentuare a contururilor (trece sus).

```
f=fspecial('average');
Ir=filter2(f,I(:,:,1));
Ig=filter2(f,I(:,:,2));
Ib=filter2(f,I(:,:,3));
Iblur=cat(3,Ir,Ig,Ib);
figure,imshow(Iblur),title('Filtru estompare')
```

```
In=rgb2ntsc(I);
f1=fspecial('unsharp');%filru trece sus
In(:,:,1)=filter2(f1,In(:,:,1));
Iu=ntsc2rgb(In);
figure,imshow(Iu),title('Filtru unsharp')
```

Filtru estompare

Filtru unsharp

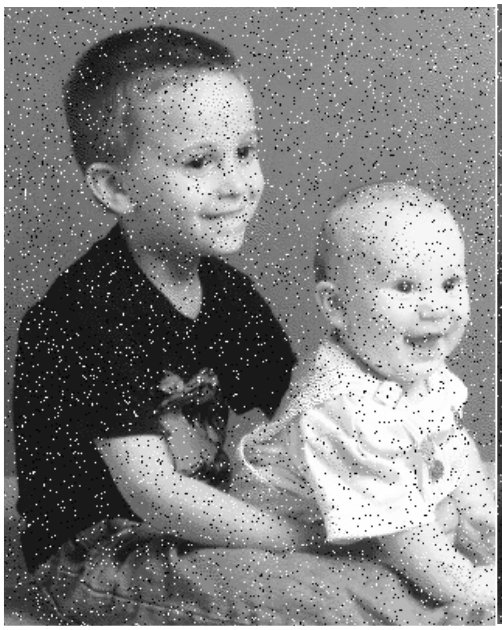


În cazul reducerii zgomotului, aplicarea filtrului median trebuie făcută pe fiecare component. Dacă se aplică doar pe matricea de intensitate, zgomotul va rămâne în celelalte două componente, care sunt și ele afectate de zgomot în urma conversiei.

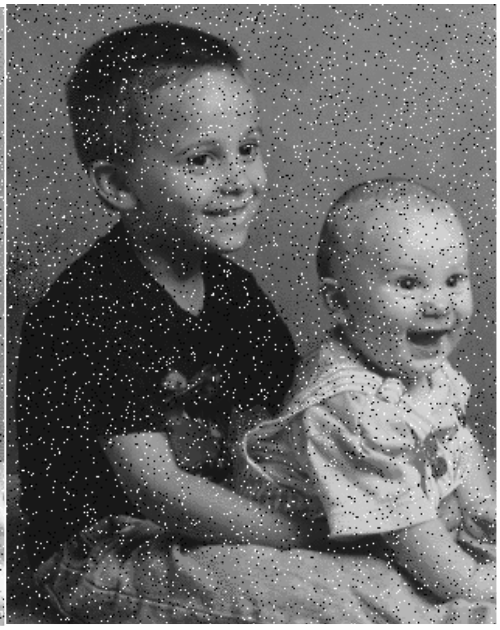
```
in=imnoise(I,'salt & pepper');
figure,imshow(in(:,:,1)),title('Componenta R')
figure,imshow(in(:,:,2)),title('Componenta G')
figure,imshow(in(:,:,3)),title('Componenta B')
irm=medfilt2(in(:,:,1));
igm=medfilt2(in(:,:,2));
ibm=medfilt2(in(:,:,3));
im=cat(3,irm,igm,ibm);
figure,imshow(im),title('Filtrul median')
% eliminare zgomot doar pe componenta de intensitate
inn=rgb2ntsc(in);inn(:,:,1)=medfilt2(inn(:,:,1));
im2=ntsc2rgb(inn);
```

```
figure,imshow(im2),title('Filtru median doar pe componenta intensitate')
```

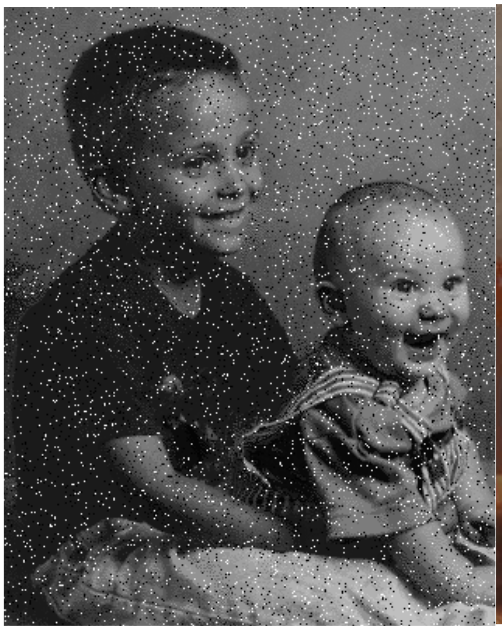
Componenta R



Componenta G



Componenta B



Filtrul median

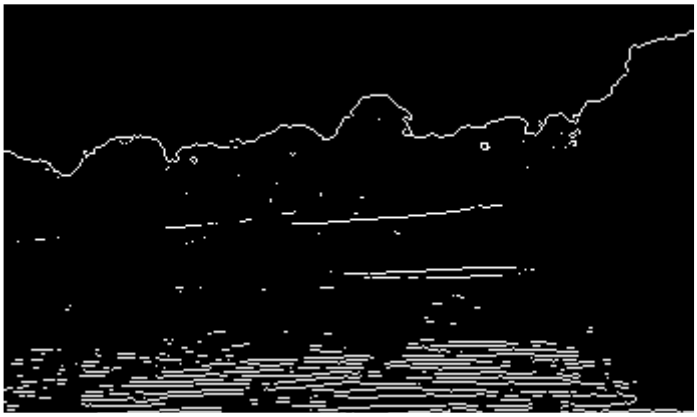
Filtru median doar pe componenta intensitate

Pentru detectarea muchiilor se poate aplica funcția edge componentei de intensitate sau se aplică funcția pe fiecare component în parte și se reunește rezultatul. Trebuie menționat, că în numeroase situații, detectarea muchiilor pe componente este mai eficientă.

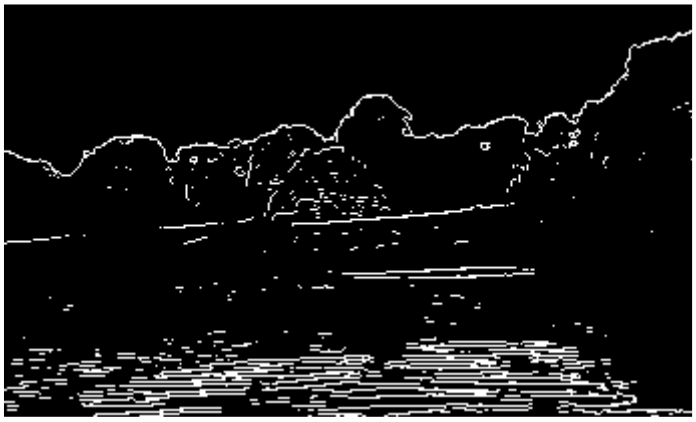
```
s=imread('autumn.tif');
figure,imshow(s),title('Imagine initiala')
sg=rgb2gray(s);
M1=edge(sg);
figure,imshow(M1),title('Muchii detectate global')
s1=edge(s(:,:,1));
s2=edge(s(:,:,2));
s3=edge(s(:,:,3));
M2=s1|s2|s3;
figure,imshow(M2),title('Muchii detectate pe componente')
```

Imagine initiala

Muchii detectate global



Muchii detectate pe componente



Bibliografie

1. Gui, V., Lacrămă, D., Pescaru, Prelucrarea imaginilor, Ed. Politehnica, 1999.
2. Hanselman, D., Littlefield, B., The Student Edition of Matlab®. User's Guide, Prentice Hall, 1995.
3. Hanselman, D., Littlefield, B., Mastering Matlab® 5. A comprehensive tutorial and reference, Prentice Hall, 1998.
4. Ingle, V., Proakis, J., Digital signal Processing Using Matlab V4®, PWS Publishing Company, 1997.
5. Neagoe, V.E., Stănăşilă, O., Teoria recunoaşterii formelor, Ad. Academiei Române, 1992.
6. Proakis, J., Manolakis, D., Digital signal Processing. Principles, Algorithms and Applications, Prentice Hall, 1996.
7. Pop, E., Naforniţă, I., Tiponuţ, V., ş.a., Metode în prelucrarea numerică a semnalelor, Ed. Facla, Timişoara, 1989.
8. Russ, J., The Image Processing Handbook, CRC Press, 1999.
9. Russ, J., Dehoff, R., Practical Stereology, Plenum Press, New York, 1999.
10. Stoenescu, R.I., Prelucrarea şi recunoaşterea imaginilor, Lito UPT, 1996.