Laurenţiu FRANGU, Laurențiu BAICU

PRELUCRAREA IMAGINILOR

Îndrumar de laborator

Introducere

Acest îndrumar este destinat studenților de la specializarea de masterat: Sisteme Electronice Avansate, ca suport pentru disciplina Prelucrarea Imaginilor.

Obiectivele activității de laborator:
- înțelegerea algoritmilor de prelucrare a imaginilor
- abilitatea de a folosi software specializat în prelucrarea imaginilor
- abilitatea de a utiliza funcțiile MATLAB, din *toolbox*-ul Image Processing, pentru manipularea și prelucrarea imaginilor.

Fiecare referat al lucrării de laborator conține un breviar teoretic și descrierea modului de lucru. La referat sînt asociate programele Matlab date ca exemplu (extensia .m) și fișierele cu imagini, care vor fi supuse prelucrării. Algoritmii de prelucrare sînt descriși în materialul de curs. Unele dintre lucrări sînt ample, încît trebuie efectuate în două ședințe de laborator.

Informații despre tooolbox se găsesc, în Matlab, astfel: în bara de meniu a programului Matlab apăsați pe simbolul "?" (link către meniul Help), apoi selectați pagina "Contents", apoi selectați Image Processing Toolbox. Dintre opțiunile care se deschid, puteți selecta o scurtă introducere ("Getting started"), un set de exemple avansate ("Examples") sau descrierile amănunțite ale seturilor de funcții de prelucrare.

Autorii urează succes tuturor celor care vor utiliza acest material și acceptă observațiile critice pe adresele: Laurentiu.Frangu@ugal.ro, Laurentiu.Baicu@ugal.ro.

Octombrie 2020 Autorii

L. Frangu, L. Baicu: Prelucrarea Imaginilor, îndrumar de laborator

Laboratorul 7: Manipularea datelor din fişiere şi afişarea imaginilor

Breviar teoretic

Imaginile sînt transmise, între calculatoare, prin fişiere.
Există mai multe formate pentru fişierele care conţin imagini.
Informaţia din imagine este codificată în două moduri:
- o listă a proprietăţilor pixelilor (luminanţă şi crominanţă), aranjată ca o hartă (*bitmap*)
- o listă a parametrilor elementelor simple care se suprapun în imagine (numai pentru imaginile sintetice, construite vectorial, cum ar fi schemele şi desenele de cablaj, create de Eagle, Proteus etc.)

În acest curs interesează numai fişierele organizate ca o hartă a pixelilor. Fişierele comprimate au la bază tot o hartă a pixelilor. În toate exemplele, reţeaua de pixeli este ortogonală.

Formatele fişierelor sînt adaptate scopului pentru care are loc transmiterea sau stocarea imaginilor. Cele mai populare formate: JPG, BMP, PNG, GIF, TIF. Programul Matlab (*toolbox* IMAGE PROCESSING) recunoaşte aceste formate, plus altele, mai puţin frecvent întîlnite.

Structura fişierului în format .bmp este:
Header, 54 octeţi (*bytes*)
Harta culorilor (sau tabelul de definiţie a culorilor), dacă aceasta există
Harta pixelilor.

Headerul conţine mai multe informaţii, dintre care cîteva sînt relevante pentru această lucrare:
octeţii 3-6 conţin lungimea fişierului
octeţii 11-14 conţin lungimea headerului plus a hărţii culorilor (dincolo de ele începe harta pixelilor)
octeţii 19-22 conţin numărul de pixeli pe linie (dimensiunea orizontală a imaginii)
octeţii 23-26 conţin numărul de pixeli pe coloană (dimensiunea verticală a imaginii)
octeţii 29-30 conţin numărul de biţi/pixel
octeţii 47-50 conţin numărul de culori în harta culorilor.
În aceste grupuri, octeţii sînt dispuşi începînd cu cel mai puţin semnificativ.

Harta culorilor conţine grupuri de cîte 4 octeţi, pentru fiecare culoare prezentă în imagine. Primii trei octeţi conţin intensităţile culorilor de bază în culoarea din hartă. Ordinea lor este: R (roşu), G (verde), B (albastru). Al patrulea octet este nefolosit. În reprezentarea cu 24 biţi/pixel, nu există harta culorilor
Pentru imaginile cu niveluri de gri, cu 8 biţi/pixel, harta culorilor conţine, cel mai adesea, 256 grupuri de cîte 4 octeţi, reprezentînd cele 256 niveluri. Grupurile sînt în ordinea crescătoare a luminanţei. În aceste fişiere, pixelul este reprezentat printr-un octet, care dă adresa de intrare în harta culorilor (implicit el dă luminanţa pixelului).

În general, în harta pixelilor, proprietăţile acestora pot fi reprezentate cu 1, 4, 8 sau 24 biţi/pixel.
1 bit/pixel: pentru imagini cu doar două culori (alb/negru sau similare)
4 biţi/pixel: pentru imagini cu maxim 16 culori (sau 16 niveluri de gri)
8 biţi/pixel: pentru imagini cu maxim 256 culori (sau 256 niveluri de gri)
În cazurile de mai sus, fiecare pixel este reprezentat în harta pixelilor prin adresa de intrare în harta culorilor.
24 biţi/pixel: nu se foloseşte hartă a culorilor, fiecare pixel este codificat prin 3 octeţi, care reprezintă intensităţile celor 3 culori.

Modul de lucru

Fişiere folosite: L7.m, lena.bmp, bliss.bmp, mammogram.jpg, lowlib.jpg.
Fişier creat: mammo.bmp.

1. Citirea imaginii din fişier
Se foloseşte funcţia imread. Pentru descrirea funcţiei, tastaţi
>>help imread
în fereastra de comandă Matlab.
În varianta cea mai simplă a sintaxei, folosiţi:
>>I=imread(filename);
unde "filename" este numele fişierului care conţine imaginea. Funcţia încearcă să deducă formatul fişierului din datele citite. Dacă este un format cunoscut, variabila I va conţine imaginea, sub forma unei matrice – pentru imagini monocrome, sau a unui set de trei matrice – pentru imagini color. Dimensiunea variabilei va fi M x N, respectiv M x N x 3, în cele două cazuri.
M este numărul de linii (= numărul de pixeli pe coloană), iar N este numărul de coloane (= numărul de pixeli pe linie).
Elementele matricei sînt numere întregi, între 0 .. 255. În Matlab, acest tip de date este numit *uint8*.
O altă variantă a sintaxei, pentru citirea din fişier, este:
>>I=imread(filename, fmt);
unde "fmt" este formatul fişierului.

2. Citirea hărţii culorilor
Harta culorilor (daca există) este citită odată cu imaginea, folosind sintaxa
>>[I,map]=imread(filename);
La afişarea hărţii culorilor, Matlab normează valoarea fiecărui octet la intervalul [0 .. 1] şi neglijează octetul nefolosit.

3. Verificarea dimensiunii hărţii culorilor şi a imaginii
Citirea unui fişier, ca şir de date binare, necesită trei paşi: deschiderea fişierului, citire, închiderea fişierului. Secvenţa de mai jos este un exemplu de citire.

```
fid=fopen(filename,'r');          % deschide fisierul cu imagine, ca sir de octeti
A=fread(fid);                     % citeste datele, ca octeti, in vectorul A
st=fclose(fid);                   % inchide fisierul
```

Din vectorul A se recompun headerul, harta culorilor şi harta pixelilor. Pentru a verifica aceste informaţii, se afişează headerul (54 octeţi).
Folosind datele din header, verificaţi că dimensiunea fişierului, dimensiunile imaginii, numărul de biţi/pixel şi numărul de culori corespund cu cele afişate.

4. Afişarea unei imagini, deja încărcate în variabila de tip imagine, se poate face cu comenzile:
>>imshow(I)
>>imview(I)
Funcţia imshow afişează imaginea la fel ca graficele. Funcţia imview afişează imaginea în cadru separat, în care se foloseşte indicatorul mouse-ului pentru a observa coordonatele pixelului şi intensitatea (adresa de intrare în harta cuorilor). Originea sistemului de coordonate este în colţul din stînga sus.

Deplasaţi cursorul spre colţul din stînga sus, pentru a observa originea.
Deplasaţi cursorul spre colţul din dreapta jos, pentru a observa dimensiunea imaginii.

Deplasați cursorul spre zonele întunecate, pentru a observa valoarea mică a luminanței (apropiată de 0).
Deplasați cursorul spre zonele luminoase, pentru a observa valoarea mare a luminanței (apropiată de 255).

Pentru a reveni la rularea programului în Matlab, selectați cu mouse-ul imaginea precedentă sau fereastra de comandă.
Pentru a citi mesajele din fereastra de comandă, poate fi necesar să deplasați fereastra în care e afișată imaginea. În acest scop, click stînga pe bara superioară a ferestrei și deplasați spre poziția dorită.

5. Încărcarea şi afişarea imaginii color, format .bmp
Pentru încărcarea imaginii se folosește tot funcția imread.
Imaginea color, în format .bmp, codifică proprietățile pixelului cu 24 biți/pixel = 3 octeți/pixel. Nu folosește harta culorilor. Se observă dimensiunea hărții culorilor = 0.

Se rezervă spațiu pentru afișarea imaginii şi a componentelor sale (R, G, B), folosind funcția subplot. Argumentele (2,2,1) semnifică: un spațiu rezervat de 2 x 2 imagini, iar poziția curentă este prima din cele rezervate.

Pentru afișarea componentelor, se folosește secvența:
```
f1=I1(:,:,1);
colormap(gray(256));
image(f1);
```

Prima linie selectează componenta roşie din imaginea I1 şi o copiază în variabila f1, care va conține doar codificarea acestei componente, fără header şi fără hartă a culorilor. Atributul R e reprezentat cu un octet/pixel şi dispunerea sa în coordonatele variabilei f1 copiază dispunerea din imaginea I1.
A doua linie crează o hartă a culorilor, compusă numai din nivelurile de gri, între 0 şi 255. A treia linie afişează variabila f1 ca o imagine, folosind harta culorilor definită mai sus.
Secvența de mai sus se repetă, pentru atributele G şi B, modificînd valoarea indicelui al treilea: f1=I1(:,:,2), respectiv f1=I1(:,:,3);

Poziționarea celor trei imagini, cu componentele R, G, B, folosește tot funcția subplot(2,2,i), unde i este poziția în care e plasată imaginea, din cele 4 poziții disponibile.

Cîteva observații simple, legate de conținutul de crominanță:
Norii sînt albi, deci conțin, în egală măsură, componentele R, G, B. În cele trei imagini parțiale, apar cu intensitate mare.
Cerul este predominant albastru, deci el apare cu intensitate mică în componentele R şi G, dar cu intensitate mare în componenta albastră.
Cîmpia este verde-gălbui, de aceea apare cu intensitate mare în componenta G, puțin mai mică în componenta R și foarte puțin în componenta B.

6. Încărcarea şi afişarea imaginii, formatul fişierului este .jpg
Matlab recunoaște acest format, se folosesc aceleași funcții ca mai sus.
Imaginea este – la origine – o imagine cu niveluri de gri, fiind obținută dintr-un aparat medical (mamograf), care detectează numai intensitatea radiației. Totuși, dispozitivul care recepționează imaginea codifică cu 24 biți/pixel. Din acest motiv, fişierul nu foloseşte o hartă a culorilor.
Fişierul .jpg este comprimat, ceea ce face ca dimensiunea sa să fie mai mică decît cea a unui fişier .bmp.

Verificați acest lucru din afișarea dimensiunii imaginii şi a dimensiunii fişierului.

Dacă Matlab constată că dimensiunea imaginii de reprezentat este mai mare decît numărul de pixeli disponibili pe ecran (numărul depinde de rezoluția aleasă), va rescala imaginea (în cazul de față, la 75%) – vezi mesajul din fereastra de comandă.

7. Scrierea unui fișier care conține o imagine
Pentru scriere, se folosește funcția imwrite, cu sintaxa:
>>imwrite(I,filename, fmt);
unde I este imaginea de memorat, filename este numele pe care vrem să-l dăm fişierului, iar fmt este un şir cu numele formatului dorit. Formatul trebuie să fie dintre cele acceptate de Matlab.
În programul folosit ca exemplu, imaginea I1 provine de la pasul precedent (încărcarea unei imagini .jpg). Are dimensiunea 570 x 482, iar elementele sale sînt octeți (codificare cu 8 biți/pixel). Acest lucru se întîmplă pentru că, la încărcarea imaginii .jpg, Matlab a sesizat că toți pixelii au conținul nul de crominanță, deci pot fi reprezentați numai prin informația de luminanță.
La scrierea fişierului mammo.bmp, Matlab adaugă headerul, harta culorilor şi harta pixelilor, aşa cum apare în imaginea I1.
Verificați că, în directorul de lucru, a apărut fişierul "mammo.bmp" şi că acesta are dimensiunea mult mai mare decît fişierul comprimat, "mammogram.jpg".
Atenție: dacă fişierul cu numele filename există deja, Matlab îl va rescrie, iar vechiul conținut se pierde!!

8. Conversia din color în niveluri de gri
Matlab dispune de funcția rgb2gray, prin care convertește o imagine color într-o imagine cu niveluri de gri. Aceasta înseamnă că se renunță la informația de crominanță şi se păstrează doar informația de luminanță. Imaginea rezultată, prin conversia imaginii I, este I1.
Conversia se realizează prin suma ponderată a informațiilor de crominanță R, G, B, cu ponderile cunoscute de la semnalul TV. Acest lucru este verificat prin instrucțiunea în care se realizează efectiv o sumă ponderată. Rezultatul este imaginea I2.
Se observă că dimensiunea imaginii inițiale este 384 x 560 x 3 (color), în timp ce imaginile în care este reprezentată numai luminanța au dimensiunea 384 x 560.

9. Transformarea pseudocolor
În această transformare se pleacă de la o imagine cu niveluri de gri. Se atribuie culori arbitrare fiecărei luminanțe (sau interval de luminanțe). În acest scop, se crează o hartă a culorilor, care va fi folosită la afişarea cu funcția imshow. Crearea hărții se face cu aceeaşi funcție colormap(), care a fost folosită la punctul 5 al lucrării. Ea influențează numai afişarea, nu schimbă conținutul imaginii inițiale, I.
Argumentul funcției colormap este o matrice, de aceeaşi dimensiune cu harta inițială a culorilor. Pe liniile acestei matrice, cele trei componente sînt intensitățile componentelor R, G, B, normate la intervalul [0 .. 1]. Prin alegerea convenabilă a valorilor de pe liniile matricei, rezultă culorile dorite.

L. Frangu, L. Baicu: Prelucrarea Imaginilor, îndrumar de laborator

Laboratorul 8: Filtrarea liniară şi neliniară a imaginilor

Breviar teoretic

Scopurile uzuale pentru filtrarea liniară a imginilor sînt reducerea zgomotului şi accentuarea contururilor. Exemplul cel mai frecvent din primul caz este filtrarea "trece-jos". La această operație sînt atenuate trecerile bruște între detalii luminoase şi detalii întunecate. Aceste tranziții bruște corespund cu zona frecvențelor înalte din transformata Fourier bidimensională a imginii. De aceea, filtrarea care păstrează doar componentele de frecvență joasă se numește "trece-jos". Efectul nedorit este că, odată cu filtrarea zgomotului, sînt atenuate şi detaliile de frecvență mare ale imaginii originale.

O altă operație de filtrare liniară, cu scop de reducere a zgomotului, este filtrarea "oprește-bandă", adecvată imaginilor afectate de zgomot de bandă îngustă, cum ar fi zgomotul periodic, provenit de la alimentare.

Pentru accentuarea contururilor, se urmărește punerea în evidență a tranzițiilor bruște, între detalii luminoase şi detalii întunecate. Aceasta înseamnă păstrarea componentelor de frecvență înaltă şi neglijarea celor de frecvență joasă, adică filtrare "trece-sus". Efectul nedorit al filtrării "trece-sus" este accentuarea zgomotului, care se află tot în zona frecvențelor înalte.

La prelucrarea sunetului sau altor semnale similare, filtrările sînt cauzale. Eşantionul curent al rezultatului se obține numai prin prelucrarea eşantioanelor anterioare ale semnalului de intrare, întrucît eşantioanele din viitor nu s-au produs încă. Prin opoziție, imaginile nu sînt semnale definite pe un interval de timp, ci pe un domeniu din spațiu, deci poate fi folosită orice operație de prelucrare necauzală. Eşantionul curent al rezultatului, din poziția curentă, este obținut inclusiv pe baza eşantioanelor vecine, din orice direcție, ale imaginii originale.

Modelul filtrării liniare, în variabila spațiu, este tot o ecuație în diferențe, ca şi în cazul filtrării în timp. Pentru simplitate, considerăm cazul imaginilor monocrome (niveluri de gri), în care fiecare pixel este reprezentat prin intensitatea luminoasă. Imaginea este o funcție scalară de două argumente spațiale, I(x,y). Operatorul de filtrare este reprezentat de matricea M(x,y), a ponderilor cu care contribuie eşantionul curent şi vecinii săi la eşantionul curent al rezultatului.

Pentru un caz simplu şi des întîlnit, cu matrice de dimensiune 3 x 3, ecuația în diferențe a eşantionului curent al rezultatului are aspectul:

$$\begin{aligned} F(x,y) = {} & I(x-1,y-1)\cdot M(1,1) + I(x-1,y)\cdot M(1,2) + I(x-1,y+1)\cdot M(1,3) + \\ & + I(x,y-1)\cdot M(2,1) + I(x,y)\cdot M(2,2) + I(x,y+1)\cdot M(2,3) + \\ & + I(x+1,y-1)\cdot M(3,1) + I(x+1,y)\cdot M(3,2) + I(x+1,y+1)\cdot M(3,3). \end{aligned}$$

Se observă cum pixelul curent al rezultatului *F* este obținut prin suma ponderată a pixelului curent şi vecinilor săi, din imaginea originală, *I*.

Evident, relația nu este valabilă pentru prima şi ultima coloană, nici pentru prima şi ultima linie (de obicei, pixelii de la extreme se copiază nemodificați). Această ecuație în diferențe poate fi exprimată prin operatorul de convoluție bidimensională:

$F = I \otimes M$.

Valorile ponderilor din matricea M determină tipul filtrului.

Exemple de măşti de filtrare liniară:

$M = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$, pentru filtrare "trece-jos". În mod intuitiv, pixelul de pe poziția centrală este înlocuit cu media pixelilor vecini, ceea ce reduce variațiile bruște de luminanță. Se observă că suma ponderilor a fost aleasă 1, astfel încît media luminanței din imaginea rezultată să fie aceeaşi cu media luminanței din imaginea originală. Din cauza medierii, nici o valoare din imaginea rezultată nu poate depăşi intervalul valorilor luminanței din imaginea originală.

$M = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$, tot pentru filtrare “trece-jos”. În mod intuitiv, filtrul dat de această mască “netezește” imaginea mai puțin decît filtrul precedent, întrucît ponderea pixelului curent este mai mare.

$M = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ sau $M = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}$. Măști care lasă imaginea neschimbată (înlocuiesc pixelul curent cu el însuși).

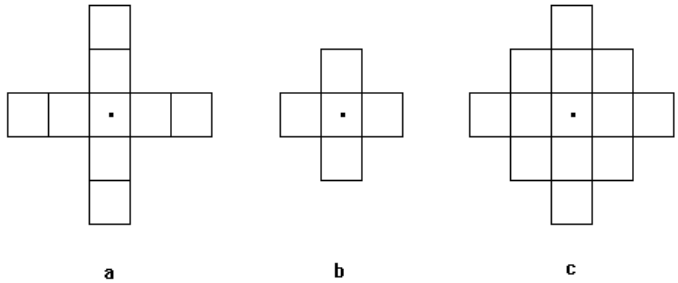
$M = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} + \begin{bmatrix} -1 & 2 & -1 \end{bmatrix} = \begin{bmatrix} -1 & 3 & -1 \end{bmatrix}$. Mască pentru filtrare “trece-sus”, accentuarea muchiilor verticale. Se observă cum intensitatea pixelului curent al rezultatului este cu atît mai mare, cu cît este mai mare diferența între pixelul curent original și vecinii săi. Masca prezentată are dimensiune 1 x 3, deci ține cont numai de diferențele între pixelii de pe aceeași linie. În acest fel, este accentuată muchia pe verticală.

$M = \begin{bmatrix} -1 \\ 3 \\ -1 \end{bmatrix}$, pentru filtrare “trece-sus”, accentuarea muchiilor orizontale.

$M = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} -1 & -2 & -1 \\ -2 & 12 & -2 \\ -1 & -2 & -1 \end{bmatrix} = \begin{bmatrix} -1 & -2 & -1 \\ -2 & 13 & -2 \\ -1 & -2 & -1 \end{bmatrix}$, pentru pentru filtrare “trece-sus”, accentuarea muchiilor, indiferent de orientare (operatorul se numește “invariant la rotație”).

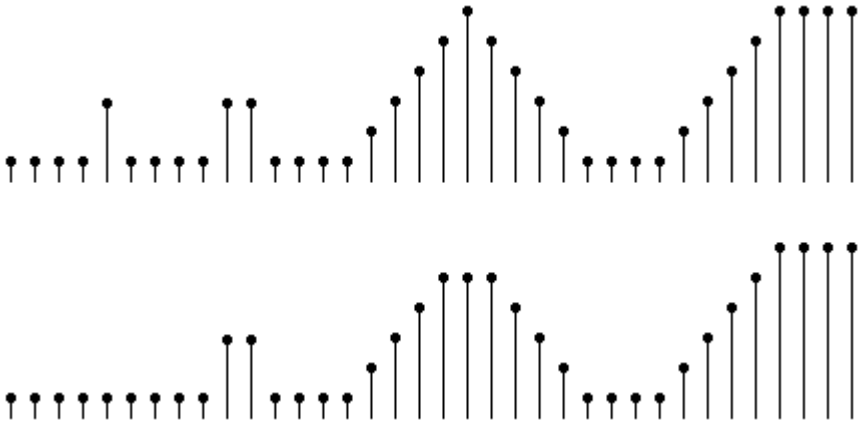
Există şi alte operații de filtrare liniară, mai sofisticate, în care este mai intuitivă utilizarea transformatelor integrale, decît convoluția cu o mască de filtrare. Un exemplu este cazul filtrului “oprește-bandă”. Acestea vor fi prezentate la lucrarea destinată transformărilor.

Filtrarea neliniară. Imaginile care au suferit perturbații, îndeosebi perturbațiile de impulsuri, produc atît efecte vizibile neplăcute, cît şi erori ale algoritmilor de prelucrare sau ale celor de recunoaştere a obiectelor. (Atunci cînd ni se cere să introducem coduri vizuale de pe o pagină web, pentru ca serverul să ştie că nu este inspectat de un robot, codurile sînt formulate astfel încît să deruteze algoritmii de prelucrare.) Exemple de perturbații de impulsuri sînt zgomotul cu puncte albe şi negre (“sare şi piper”) şi liniile parazite. Impulsurile mari nu pot fi filtrate liniar, întrucît filtrarea liniară redistribuie amplitudinea impulsurilor de la pixelul afectat către pixelii vecini. De aceea, trebuie introdusă o operație diferită, care încearcă să “reteze” impulsul parazit. Modelul acestei operații depinde de modelul perturbației. Ca exemplu tipic este prezentată filtrarea mediană, care folosește tot o mască de filtrare, dar nu într-o operație de convoluție (reamintire: convoluția este un operator liniar). Exemple de măşti:



a b c

Masca se deplasează pe suprafața imaginii, la fel ca în cazul filtrării liniare. Pixelul curent nu mai este înlocuit cu o medie ponderată a pixelilor vecini, ci este înlocuit cu pixelul cu luminanță mediană, adică acea valoare care este mai mică decît jumătate din luminanțele cuprinse în mască și mai mare decît cealaltă jumătate. În acest fel, este păstrată luminanța cea mai plauzibilă să fi existat în poziția curentă, înainte de apariția perturbației. Filtrarea este evident neliniară și nu garantează recuperarea imaginii originale, ci a unei versiuni plauzibile a acesteia. Cum impulsurile parazite sînt, adesea, de durată mică, ele apar pe imagini în puncte izolate, nu apar în "pete". Ca urmare, filtrarea mediană se aplică, cel mai adesea, pe o singura direcție. În exemplul de mai jos, apare transformarea suferită de o linie a imaginii (sus – imaginea perturbată, jos – imaginea filtrată), după filtrare mediană, cu mască 1 x 3. Se observă că un impuls izolat, de lungime 1, a fost eliminat, în timp ce impulsurile de lungime mai mare rămîn nemodificate. În general, un impuls de lățime "n", va fi eliminat de o mască cu dimensiunea 2n+1. Pentru perturbații de întindere mai mare sînt necesare alte operații, întrucît filtrarea mediană va deteriora semnificativ conținutul.



Modul de lucru

Fişiere folosite: L8.m, lena.bmp, build.tif.

1. Filtrarea liniară, "trece-jos", prin utilizarea convoluției în două dimensiuni.
Citirea imaginii din fişier se realizează cu funcția imread (vezi lucrarea de laborator precedentă).
Pentru păstrarea imaginii originale, în matricea F1, se foloseşte masca de filtrare M1, care conține doar ponderea 1, pentru pixelul curent.
Pentru filtrare "trece-jos", se foloseşte masca de filtrare M2:

$$M_2 = \frac{1}{8}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

Suma ponderilor este 1, astfel încît să nu se modifice luminanța medie a imaginii rezultate.
Imaginea filtrată se obține într-o matrice a luminanțelor, folosind funcția conv2, astfel:
F2=conv2(double(I),M2,'same');
Această funcție realizează convoluția dintre imaginea originală, I, și masca de filtrare, M2.
Argumentul I este transformat în dublă precizie, folosind funcția double(I), atunci cînd este necesar calcul foarte precis. În situația de față, dubla precizie nu este esențială.
Argumentul 'same' asigură trunchierea imaginii rezultate la aceeaşi dimensiune cu imaginea originală.
Funcția figure deschide o nouă fereastră de afişare. Pentru a vedea ambele imagini de interes (cea originală și cea transformată), se vor folosi două cîmpuri în imagine, cu funcțiile subplot. În

comanda subplot(1,2,1), semnificațiile argumentelor sînt, în ordine: cîmpuri plasate pe 1 linie, pe 2 coloane, cîmpul pregătit pentru afișare este nr. 1.
Pentru ca imaginile să fie afișate la dimensiunea originală, trebuie redimensionată fereastra de afișare, cu funcția:
set(gcf,'Position',[181 280 1056 420]);
Comanda de mai sus stabilește atributul poziție, pentru figura curentă. Primele două argumente numerice stabilesc coordonatele colțului din stînga jos al figurii, iar următoarele două argumente numerice stabilesc dimensiunile figurii (orizontală, verticală). (Dacă dorim să observăm toate atributele figurii, folosim comanda: get(gcf).)

Înainte de afișarea imaginilor, în cîmpurile potrivite, se folosește comanda
colormap(gray(256));
prin care Matlab alege harta culorilor cu care vor fi asociate intensitățile pixelilor (în acest caz, harta cu 256 niveluri de gri).
Pentru verificare, cîmpurile din figură capătă cîte un titlu sugestiv, iar în fereastra de comandă afișăm dimensiunile imaginii rezultate, folosind comenzile title, respectiv display.

În momentul în care execuția se oprește, datorită comenzii pause, matlab afișează în colțul din stînga jos mesajul “paused”. Cele două cîmpuri conțin imaginea originală și cea filtrată “trece-jos”. Se poate observa efectul filtrării privind detaliile fine din imagine, cum ar fi: genele subiectului, marginile penelor de la pălărie, textura stofei de pe calota pălăriei. Toate arată că detaliile fine au fost uniformizate, sînt încețoșate, față de original.
Pentru a trece la următoarea filtrare se apasă orice tastă. Comanda de filtrare, cu aceeași mască, se aplică încă de 4 ori, succesiv, imaginii filtrate în etapa anterioară. La final, imaginea filtrată de 5 ori diferă de original, în privința detaliilor.

2. Realizarea convoluției, prin două bucle FOR
Convoluția realizează suma ponderată, pentru fiecare pixel al imaginii, conform cu relația scrisă în breviarul teoretic. Ca urmare, ea poate fi implementată prin două bucle FOR, ca în secțiunea inactivă a programului (toate comenzile sînt precedate de semnul de comentariu, %). Această variantă funcționează corect, dar are nevoie de mai multe linii de comandă și este foarte lentă, în comparație cu comanda conv2.

3. Filtrarea liniară, “trece-sus”
Pentru accentuarea muchiilor, indiferent de orientarea lor, a fost aleasă masca:

$$M_2 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$, care păstrează media luminanței.

Imaginea originală este aceeași de la filtrarea precedentă. Efectele filtrării “trece-sus” sînt observabile, în două direcții diferite. Pe de o parte, pe textura calotei pălăriei, pe firele de păr sau pe pene se văd foarte clar muchiile. Pe de altă parte, pe suprafețele care erau netede, în imaginea originală, acum au apărut alternanțe de puncte albe și negre. Aceast efect se datorează accentuării zgomotului de frecvență înaltă (diferențele întîmplătoare între pixelii vecini, care păreau că formează o suprafață uniformă).
Se repetă filtrarea “trece-sus” și se observă că zgomotul este amplificat și mai mult, cu riscul de a face imaginea neinteligibilă.

4. Filtrarea selectivă, a muchiilor orizontale sau verticale
Pentru accentuarea doar a muchiilor cu anumită orientare, se folosesc măștile de filtrare prezentate în secțiunea breviarului teoretic. Vizualizarea efectelor se poate face și pe imaginea anterioară (lena.bmp), dar aici am ales o imagine nouă: build.tif, pentru că ea conține multe muchii orizontale

şi verticale evidente. Totuşi, apar cîteva elemente noi, ceea ce face ca evidenţierea muchiilor prin filtrare să necesite cîteva acţiuni de pregătire.

a. Tipul fişierului este .tif, în care pixelii sînt codificaţi pe 3 culori, chiar dacă conţinutul imaginii este monocrom. Ca urmare, variabila I1, care conţine imaginea citită, este organizată în trei planuri succesive, cu luminanţele corespunzătoare celor trei culori. Întrucît toţi pixelii conţin niveluri de gri, cele trei planuri sînt identice, deci putem reduce cu uşurinţă imaginea la un singur plan. Acest lucru se realizează cu comanda

```
I2(:,:)=I1(:,:,1);
```

prin care imaginea I2 reţine doar primul plan din imaginea I1.

b. Apoi, dimensiunea imaginii este prea mare, pentru rezoluţia uzuală a ecranului. De aceea, am omis primele 150 linii (cele de sus), atunci cînd am copiat imaginea I2 în I3, folosind comanda:

```
I3=I2(151:m,:);
```

c. În fine, pentru afişare, am ales să evidenţiem numai muchiile, fără a aduna conţinutul imaginii originale, ceea ce modifică masca de filtrare: $M = \begin{bmatrix} -1 & 2 & -1 \end{bmatrix}$.

Repoziţionarea figurii şi afişarea se fac la fel ca în secţiunea precedentă.

5. Filtrarea mediană

Imaginea originală este citită, ca în exemplele precedente, apoi este copiată în noua imagine, F1. În aceasta este introdusă o linie aproape în întregime perturbată: pe linia 200 au fost introduşi 470 pixeli cu luminanţă 0. Diferenţa între imaginea originală şi cea perturbată se observă uşor, dacă respectăm rezoluţia minimă de afişare (operaţia de afişare devine nesigură, dacă reducem rezoluţia).

Pentru a afişa decalat cele trei imagini de interes, folosim comanda prin care reaşezăm fereastra:

```
set(gcf,'Position',[x1 y1 xx yy]);
```

Parametrii x1, y1, xx, yy reprezintă, respectiv: abscisa şi ordonata colţului din stînga jos, apoi lungimea şi înălţimea ferestrei, exprimate în pixeli. Evident, limitele depind de rezoluţia curentă a ecranului.

Imaginea perturbată este copiată în F2, care suferă operaţia de filtrare mediană. Întrucît ştiam că perturbaţia s-a manifestat pe linie şi că înălţimea ei nu depăşeşte un pixel, folosim o mască coloană, 3 x 1. (Pentru perturbaţii verticale, trebuia folosită mască linie.)

Cele două comenzi, repetate în buclă:

```
    x=[F1(i-1,j) F1(i,j) F1(i+1,j)];
    F2(i,j)=median(x);
```

În prima linie, masca 3 x 1 selectează pixelul curent, pe cel de deasupra şi pe cel de dedesubt. În a doua linie, pixelul curent din imaginea filtrată este ales cel cu luminanţa mediană (nu medie!), dintre cei trei selectaţi. Întrucît operaţia este repetată în buclă, de 512 x 512 ori, durata este mare. Pe durata operaţiei, Matlab afişează mesajul “Busy”, în colţul din stînga jos.

După afişarea celei de a treia imagini, cea filtrată, este dificil de observat vreo diferenţă, faţă de original (deşi mici diferenţe există, în mod inerent).

Laboratorul 9: Filtrare cu funcții dedicate din toolbox Image Processing

Breviar teoretic

O mare parte dintre oprațiile de prelucrare a imaginilor beneficiază de comenzi dedicate, din toolbox Image Processing. Exemple:
funcțiile *imfilter, filter2, conv2, fspecial*, pentru filtrare liniară a imaginilor, în domeniul spațial
funcția *ftrans2*, pentru filrare liniară a imaginilor, în domeniul frecvență
funcțiile *imhist, imadjust*, pentru calculul histogramei și corectarea contrastului.

Un prim avantaj al utilizării acestor comenzi dedicate este acela că simplifică scrierea programului, ca număr de linii de cod. Totuși, un avantaj încă mai important, care se manifestă atunci cînd programul conține multe execuții în buclă, este că se micșorează timpul de execuție. La execuția în buclă, programele interpretate (este cazul Matlab) fac interpretarea fiecărei instrucțiuni, de atîtea ori de cîte ori se repetă bucla. Prin opoziție, programele compilate (este cazul celor scrise în C, Pascal etc.) execută în buclă doar operațiile la nivel de cod-mașină, întrucît compilarea a avut loc anterior. Pentru ilustrare, în cazul Matlab, se poate compara operația de atribuite a valorilor unui vector, în buclă *for*, cu operația de atribuire a acelorași valori, realizată dintr-o singură comandă.

Pentru filtrarea imaginii, se poate folosi funcția *imfilter*, cu sintaxa:
I1=Imfilter(I,h),
unde I este imaginea originală, h este masca de filtrare, iar I1 este imaginea rezultată prin filtrare. Dintre opțiunile de format, 'same' impune ca dimensiunile imaginii filtrate să fie aceleași cu ale imginii originale iar 'conv' impune ca filtrarea să se realizeze prin convoluție.
Masca de filtrare poate fi generată element cu element, ca în lucrarea anterioară (măștile M1, M2 la filtrarea trece-jos, trece-sus), sau poate fi generată cu funcția *fspecial*, care a memorat cîteva tipuri de măști folosite frecvent.

Modul de lucru

Fișiere folosite: L9.m, lena.bmp.

1. Comparația între duratele de execuție, pentru buclă *for* programată explicit și buclă inclusă în comanda Matlab.
În program, sînt creați vectorii X, Y, de lungime 50.000 elemente, apoi se calculează suma lor, Z. Prima versiune se execută în buclă, în mod explicit. A doua versiune se execută prin trei instrucțiuni, tipice pentru lucrul cu vectori, în care bucla este conținută în mod implicit. Ambele versiuni execută același număr de operații aritmetice.
După lansarea programului, în fereastra de comandă se afișează mesajul 'apasati o tasta'. Din momentul apasarii, programul calculeaza vectorii X, Y, Z, în prima versiune, iar în stînga jos apare mesajul 'Busy'. Cronometrați durata calculelor, pînă cînd se termină execuția, semnalizată în fereastra de comandă prin 'terminat lucrul in bucla'. O nouă apăsare de tastă pornește a doua versiune de calcul. Ar trebui cronometrată durata calculelor, pentru comparație, dar veți constata că e de prisos!

2. Filtrare cu funcția *imfilter*
Se încarcă imaginea monocromă din fișierul lena.bmp. Pentru filtrare, se creează masca M2, folosind funcția *fspecial*('laplacian'). Dacă afișăm matricea M2, se constată că este un operator de derivare, invariant la rotație, cu suma coeficienților egală cu 0:

$$M_2 = \frac{1}{6}\begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix}.$$

Funcția *imfilter* realizează accentuarea muchiilor, folosind acest operator.
Pentru a observa muchiile aceentuate, suprapuse peste imaginea originală, se adună termenul 1 la coeficientul central al lui M2 și se repetă filtrarea.
Cele trei imagini afișate arată că s-a obținut același rezultat ca în lucrarea precedentă.

Laboratorul 10: Modificarea scării de gri

Breviar teoretic

Uneori, operațiile de filtrare pe imagine au efect neglijabil, deoarece contrastul imaginii este scăzut. Este necesară îmbunătățirea contrastului, atît pentru uzul operatorului uman, cît și pentru a obține efecte semnificative ale filtrării, cum ar fi în cazul accentuării conturului. Îmbunătățirea contrastului este frecvent folosită la imaginile monocrome, provenind din observațiile meteo sau din investigațiile radiologice medicale. Este esențial ca, în timpul acestei operații, să nu se schimbe informația conținută de imagine.
Pentru simplitate, considerăm imagine monocromă (toate investigațiile radiologice medicale intră în această categorie). Calitatea constrastului se poate observa în histograma luminanței, adică un grafic al frecvenței de apariție a luminanței, ca funcție de valoarea luminanței. Dacă luminanța este distribuită într-un interval îngust, ca în figura 1a, contrastul este scăzut. Pentru corectarea contrastului, valorile luminanței trebuie modificate ca în figura 1b, astfel încît distribuția ei să acopere tot intervalul posibil. În cazul imaginilor monocrome, reprezentate cu 8 biți/pixel, acest interval este de 256 niveluri de gri. În figura 1, cele 256 niveluri au fost reprezentate în grupuri de 4, astfel încît au rezultat numai 64 intervale. Reprezentarea pe un număr redus de intervale este utilă, atunci cînd varianța curbei este prea mare și introduce amănunte nesemnificative. Calculul histogramei se realizează numărînd cîți pixeli au luminanța cuprinsă în fiecare din intervalele reprezentate pe abscisă. Matlab are o funcție dedicată, *imhist*, iar funcția *imtool* permite, printre alte operații, reprezentarea grafică a histogramei. Funcția *imtool* este disponibilă numai pentru variantele recente de Matlab (după 2012).

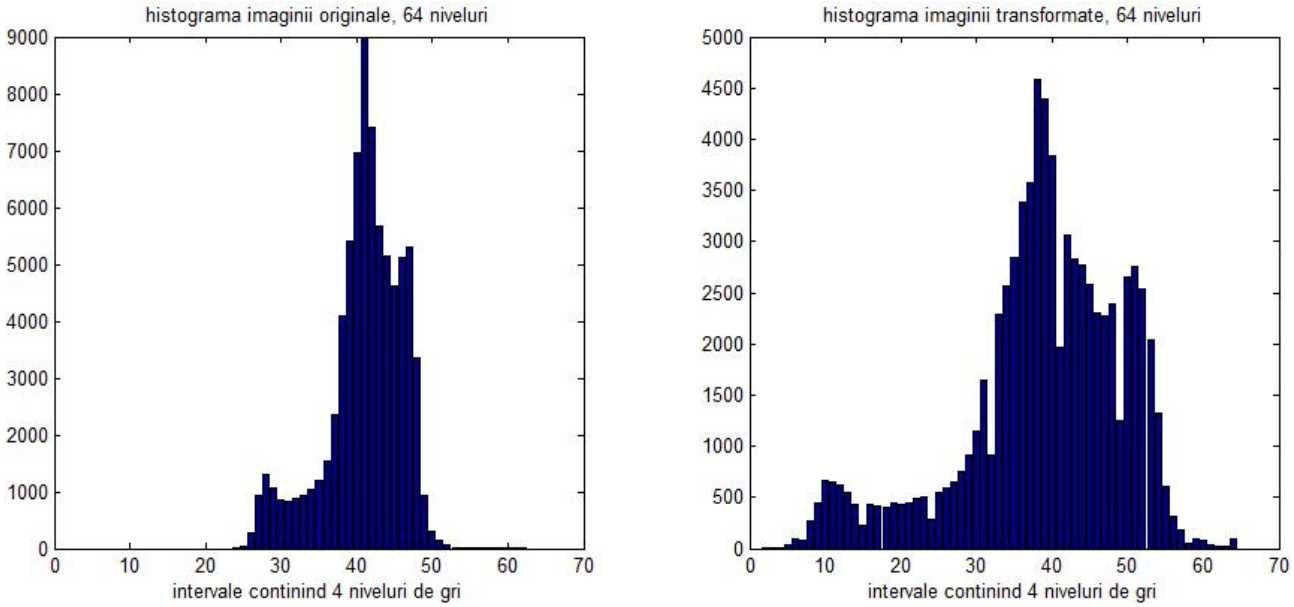


Figura 1: Histograma luminanței pentru o imagine cu contrast scăzut (a) și pentru o imagine cu contrast normal (b)

Funcția de transformare a luminanței, în scopul îmbunătățirii contrastului, poate urma orice curbă monotonă, ca în figura 2 (preluată din [1]). Pe abscisă este reprezentată luminanța pixelilor din imaginea originală, iar pe ordonată apare luminanța pixelilor din imaginea transformată. Curba indexată cu valoarea 1 nu modifică imaginea. Curbele din stînga măresc contrastul pentru zonele întunecate din imagine, iar curbele din dreapta măresc contrastul pentru zonele luminoase. În

situația în care zonele întunecate sau/și cele luminoase sînt foarte slab reprezentate, funcția de transformare poate urma curba din figura 3.

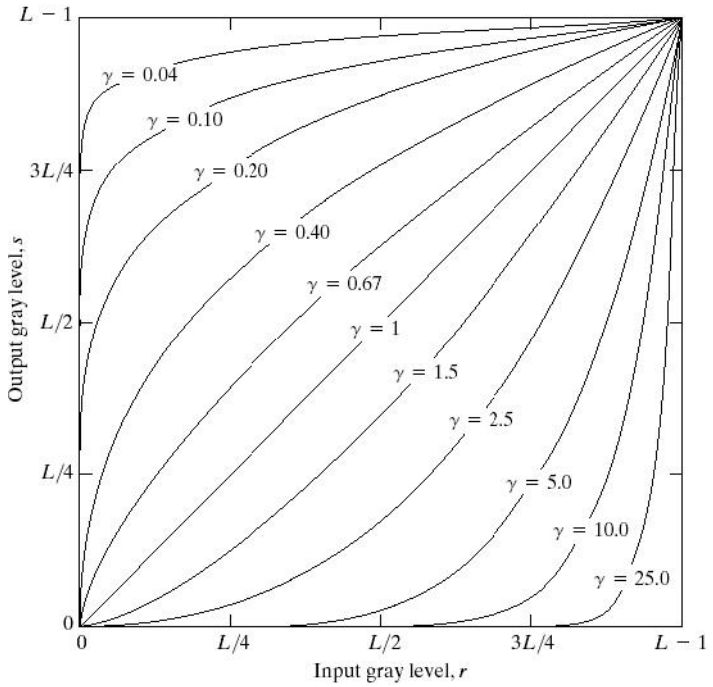


FIGURE 3.6 Plots of the equation $s = cr^{\gamma}$ for various values of γ ($c = 1$ in all cases).

Figura 2: Funcții uzuale de transformare a luminanței (preluate din [1])

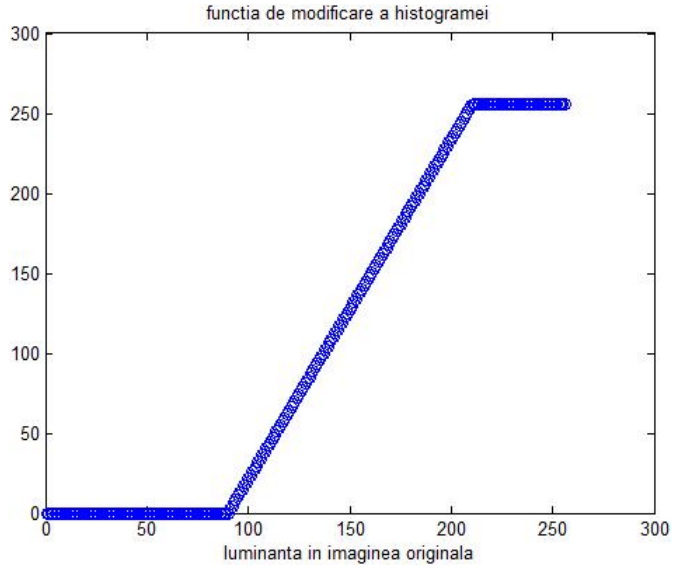


Figura 3: Funcție liniară de transformare a luminanței

Modul de lucru

Fișiere folosite: L10.m, constrast.jpg

1. Încărcarea și obținerea imaginii monocrome
A fost aleasă o imagine cu contrast slab, pentru ilustrarea transformării. Întrucît imaginea este reprezentată cu 24 biți/pixel (formatul .jpg), nu are hartă a culorilor. Pentru conversia în imagine monocromă, cu 8 biți/pixel, se poate folosi funcția *rgb2gray* (va fi folosită în lucrare viitoare). Aici am profitat de faptul că imaginea inițială nu conține informație de crominanță, așa că am definit o imgine nouă (I1), cu 8 biți/pixel, în care am păstrat un singur plan al imaginii inițiale.

```
I1=I(:,:,1);                % copiez numai primul plan in imaginea I1
```

2. Calculul histogramei

Pentru calculul histogramei (expresia corectă: calculul frecvenței de apariție a luminanței), folosim două bucle *for*, care parcurg toată imginea și în care vectorul hist numără pixelii cu fiecare valoare a luminanței. Ulterior, se calculează o nouă histogramă (vectorul hist1), în care se numără pixelii a căror luminanță este cuprinsă în intervale de 4 niveluri de gri consecutive (ca cea din figura 1).

La afișare, apar: histograma pe 256 intervale, cea pe 64 intervale, și varianta obținută cu funcția *bar*. Trecerea la următoarea figură se face cu apăsarea unei taste.

Pentru comparație, se mai folosește funcția *imhist* pentru calculul și afișarea histogramei, într-o figură nouă, cu comanda:

```
imhist(I1)
```

Se observă că afișarea trunchiază o parte din forma histogramei. Totuși, valorile calculate sînt corecte, așa cum se observă din figura următoare, în care funcția *stem* a folosit valorile calculate de funcția *imhist*:

```
[counts,X]=imhist(I1);
stem(X,counts)
```

Inspectînd histograma, se observă intervalele de luminanță neocupate de pixelii imaginii originale: între 0 și 90, apoi între 210 și 255.

3. Transformare pentru îmbunătățirea contrastului

Se folosește o transformare ca cea din figura 3, folosind limitele de luminanță observate anterior. Prima secvență de comenzi și figura aferentă au doar rolul de a afișa forma funcției de transformare. Funcția se aplică fiecărui pixel, deci este nevoie, din nou, de două bucle *for*, care parcurg toată imaginea I1. Înainte de memorarea în imaginea I2, rezultatul este rotunjit, cu funcția *round*, pentru că luminanța este reprezentată, în imaginea monocromă, prin numere întregi, între 0 și 255. Calculul histogramei *hist2* și reducerea rezoluției la 64 intervale, în *hist3*, se realizează ca la punctul precedent. În următoarele două figuri sînt prezentate, comparativ, imaginea transformată, față de cea originală, și histograma imaginii transformate, față de cea inițială (pe 64 intervale). Se observă cum, în imaginea transformată, luminanța ocupă tot intervalul disponibil. Efectul este vizibil în constrastul crescut al imaginii I2.

Laboratorul 11: Segmentarea imaginii

Breviar teoretic

Unul dintre obiectivele prelucrării imaginilor, anume recunoaşterea obiectelor şi măsurarea poziţiilor lor, necesită detectarea de obiecte distincte în imagine, ai căror pixeli au o proprietate comună. Operaţia prin care pixelii sînt grupaţi în submulţimi de pixeli, cu aceeaşi proprietate, se numeşte segmentare. Cea mai intuitivă segmentare este segmentarea după luminanţă, în care proprietatea comună este aceea că luminanţa pixelilor se află într-un interval definit. Spre exemplu, într-o imagine monocromă, pixelii pot fi declaraţi ca aparţinînd obiectelor, dacă luminanţa lor depăşeşte un anumit prag. Pixelii cu luminanţa sub acel prag sînt declaraţi ca aparţinînd fondului. Această variantă simplă de segmentare se numeşte, în limba engleză, *thresholding*. Intervalul luminanţelor pixelilor obiect poate fi definit şi în alte moduri. Segmentarea mai poate lua în considerare informaţia de crominanţă, textura sau semnificaţia conţinutului.

Pentru varianta simplă de segmentare după luminanţă, este utilă inspectarea histogramei. Din ea rezultă intervalul ocupat de obiecte şi pragul sau pragurile optime pentru segmentare. Spre exemplu, din histograma prezentată în figura 1, rezultă că pixelii obiect şi pixelii fondului au – în general – luminanţe distincte. Presupunem că pixelii luminoşi aparţin fondului. Alegerea unui prag de segmentare prea înalt va plasa mai mulţi pixeli în categoria obiectelor, în dauna fondului, ceea ce va uni obiecte şi va creşte ariile lor. Alegerea unui prag prea scăzut va avea efect contrar: mai puţine obiecte, de arii mai mici. De aceea, alegerea pragului este rezultatul unui compromis, ajustat după experienţa proiectantului.
Pentru segmentarea după culoare, criteriile sînt mai complicate. Un exemplu este gruparea pixelilor care au culori apropiate, care se realizează prin încadrarea ponderilor culorilor de bază în intervale predefinite.
Operaţia de segmentare după luminanţă este relativ simplă, în timp ce alegerea criteriului necesită cunoştinţe anterioare despre statistica imaginilor studiate. De aceea, nu sînt utile funcţii specializate pentru segmentare.

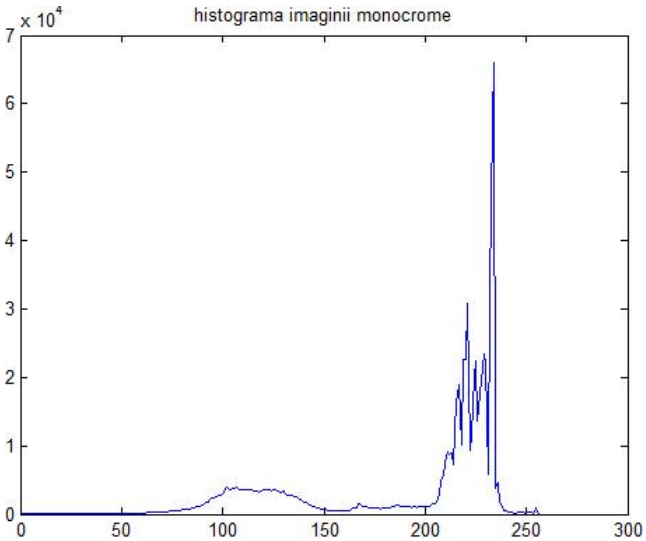


Figura 1: Histograma luminanţei pentru o imagine cu obiecte întunecate, pe fond luminos

Modul de lucru

Fişiere folosite: L11.m, obiecte.jpg, bliss.bmp

1. Încărcarea și obținerea imaginii monocrome
A fost aleasă o imagine cu obiecte care constrastează cu fondul, pentru simplitatea segmentării. Imaginea este color, dar conținutul de crominanță este minor, segmentarea poate fi realizată numai pe baza informației de luminanță. Imaginea este reprezentată cu 24 biți/pixel (formatul .jpg), poate fi convertită în imagine monocromă, folosind funcția *rgb2gray*. Imaginea rezultată, I1, este reprezentată cu 8 biți/pixel. După două apăsări de taste, se afișează imaginea originală și cea monocromă.

2. Calculul histogramei
Se folosește același procedeu ca în lucrarea precedentă, fără a folosi o funcție din toolbox. După o apăsare de tastă, se afișează histograma. Pe grafic se observă două zone distincte, cu frecvență de apariție mai mare, care corespund obiectelor și fondului. Ele par să se întindă pînă la nivelul de gri 150, respectiv de la nivelul 200 în sus. Întrucît în imagine apar mai multe umbre, cu luminanță intermediară, în histogramă apar valori nenule ale frecvenței de apariție, între nivelurile 150 și 200.

3. Alegerea pragului optim de segmentare.
Pentru segmentare este nevoie, din nou, de două bucle *for*, care parcurg toată imaginea. La fiecare pixel examinat, se ia decizia că aparține fondului, dacă luminanța este mai mare decît pragul, sau aparține unui obiect, în caz contrar.
Secvența de comenzi se repetă, pentru două valori diferite ale pragului, generînd două imagini segmentate: I2 pentru pragul 210 și I3 pentru pragul 154. Afișarea celor două imagini rezultate pune în evidență efectele alegerii pragului. Pentru valori mari ale pragului, mulți pixeli vor fi alocați la zonele întunecate. În cazul de față, asta înseamnă creșterea ariilor obiectelor, urmată de creșterea numărului de obiecte, sau de unirea obiectelor sau de amîndouă. Pentru valori mici ale pragului, mulți pixeli vor fi alocați la zonele luminoase, cu efecte simetrice față de cele amintite mai sus.

4. Segmentarea după informația de crominanță
La această operație, aplicarea criteriului este mai complicată, întrucît lucrează cu mai multe variabile. În cazul de față, ne propunem să păstrăm în imagine numai pixelii de culoare galbenă, și să-i suprimăm pe ceilalți. Criteriul după care stabilim culoarea galbenă a fost formulat astfel:
- dacă culoarea roșie are ponderea, în luminanța pixelului, între pragurile pragr1 și pragr2
ȘI
- dacă culoarea verde are ponderea, în luminanța pixelului, între pragurile pragg1 și pragg2.
În prima figură se afișează imaginea RGB inițială și trei imagini monocrome, reprezentînd intensitățile celor trei culori componente.
Se realizează o copie color a imaginii originale, I1, care va fi supusă segmentării. Se mai formează o matrice de dimensiunea imaginii, f1, care va afișa monocrom rezultatul segmentării. Ca și la operația precedentă, este nevoie de două bucle *for*, în care se parcurge imaginea și se calculează criteriul de segmentare. Pixelii care nu corespund criteriului sînt suprimați din copia color, iar cei care corespund sînt marcați cu 0, în copia monocromă.
La final, imaginea originală, imaginea segmentată color și cea segmentată monocromă sînt afișate în trei zone ale figurii. Cele două variante reflectă aceeași informație, numai că una conține pixelii galbeni, așa cum erau în versiunea inițială, iar cealaltă conține pixeli negri, în locul celor galbeni.

Laboratorul 12: Transformări morfologice, etichetarea

Breviar teoretic

Deseori, imaginea este afectată de perturbații, care provin din iluminare neuniformă, umbre, reflexii, pete etc. Efectul lor se simte în imaginea segmentată, în sensul că obiectele pot fi fragmentate sau pot fi unite accidental, pot apărea obiecte sau găuri false. Înainte de extragerea trăsăturilor, este util să îndepărtăm cît mai multe dintre efectele perturbațiilor. O clasă de transformări pe care le putem aplica, în acest scop, constă în operațiile de dilatare, erodare și combinațiile lor. Sînt numite transformări morfologice, întrucît ele pot conduce la modificarea morfologiei imaginii: ariile obiectelor, numărul de obiecte și de găuri etc. Modificarea poate fi în sensul restaurării morfologiei reale, existente în absența perturbațiilor, sau poate fi în sensul alterării acelei morfologii. Operația are cu atît mai mult succes, cu cît avem mai multe informații despre natura perturbației. În cele ce urmează, considerăm doar acțiunea asupra unui obiect.
Una dintre transformări este erodarea obiectelor, prin care o parte dintre pixelii obiect, aflați pe contur, sînt transformați în pixeli fond. Algoritmul de erodare, ca și celelalte care vor fi descrise aici, folosește o mască, cea mai des întîlnită fiind B3. Cea mai frecvent utilizată regulă de erodare, folosind masca B3, este următoarea:
- pixelii obiect care au cel puțin un vecin tetraconetat, aparținînd fondului, devin pixeli fond;
- pixelii fondului rămîn neschimbați.

Efectele erodării sînt: dispar obiectele formate din pixeli izolați, se reduc terminațiile subțiri de pe conturul obiectului, se măresc găurile, scade aria obiectelor, se separă în bucăți obiectele care sînt unite prin linii subțiri de 1-2 pixeli.
O altă transformare, cu efect cumva opus, este dilatarea, prin care se adaugă pixeli pe conturul obiectelor existente. Cea mai frecvent utilizată regulă de dilatare este următoarea:
- pixelii tetraconectați cu un pixel obiect devin și ei pixeli obiect;
- pixelii obiectului rămîn neschimbați.

Efectele dilatării sînt: scade aria găurilor, dispar găurile de dimensiuni mici, se umplu fisurile înguste ale obiectelor, crește aria obiectelor, se unesc obiectele care sînt despărțite prin intervale ale fondului de 1-2 pixeli.
O operație de erodare, urmată de dilatare, se numește deschiderea imaginii. O operație de dilatare, urmată de erodare, se numește închiderea imaginii. Ele au avantajul de a modifica mai puțin ariile obiectelor. Folosim deschiderea sau închiderea, după cum perturbațiile au produs pixeli izolați și obiecte unite accidental, sau găuri mici și obiecte divizate accidental.
Pentru operațiile de erodare și dilatare, toolbox-ul Image Processing conține funcțiile *imerode* și *imdilate*, în sintaxa cărora trebuie precizată masca folosită (*structuring element*).

Etichetarea este o operație în care sînt detectate grupurile conectate de pixeli din imaginea segmentată și în care este atribuită cîte o etichetă fiecărui grup. Ea precede extragerea trăsăturilor și măsurarea poziției. La sfîrșitul operației, fiecare pixel este însoțit de această etichetă, care arată grupul cu care este conectat acest pixel. Se realizează prin parcurgerea imaginii, pixel cu pixel, și examinarea conectivității cu pixelii deja etichetați. În acest scop, toolbox-ul Image Processing conține funcția *bwlabel*, care etichetează imagini 2D binare. Convenția respectată de această funcție este că pixelii fondului au valoarea 0, iar pixelii obiect au valoare diferită de 0. Se presupune implicit că pixelii obiect sînt octoconectați, dar există posibilitatea de a preciza, prin sintaxa funcției, convenția pixelilor tetraconectați. Rezultatul întors de funcție este o matrice, similară cu imaginea, care conține etichetele, în locul valorilor pixelilor. Toți pixelii de fond primesc eticheta 0.

Dezavantajul acestei funcții este că nu atașează etichete găurilor din obiecte, nici nu memorează informații structurale, cum ar fi incluziunea găurilor în obiecte sau invers. Pentru a extrage trăsăturile, informații structurale și pe cele de poziție, este necesară parcurgerea, încă o dată, a

matricei rezultate. Cu această ocazie, se poate completa partea de etichetare pe care funcția nu a rezolvat-o. În această lucrare, sînt prezentate determinarea ariei, care este suma pixelilor unui obiect, şi a centrului de greutate, care este compus din media absciselor şi media ordonatelor pixelilor obiectului.

Modul de lucru

Fişiere folosite: L12.m, obiecte.jpg

1. Încărcarea, obținerea imaginii monocrome şi segmentarea se realizează ca în lucrările precedente, pornind de la imginea din fişierul obiecte.jpg. Variabila dim este dimensiunea imaginii, care va fi valabilă pentru toate imaginile prelucrate în această lucrare. Se foloseşte convenția ca pixelii fondului să aibă valoarea 0, ca în funcțiile de transformare morfologică și în funcția *bwlabel*. După prima apăsare de tastă, apare imaginea segmentată, lîngă cea originală, color.

2. Pentru comparație, se realizează erodarea, cu o funcție proprie și cu funcția *imerode*. După trei apăsări de taste, se afişează, în fereastră nouă, imaginea segmentată, imaginea erodată cu algoritm propriu și imaginea erodată cu funcția *imerode*. În corpul programului, se observă cele două bucle *for*, din algoritmul propriu, definirea măștii B3 și sintaxa funcției.
Privind imaginile, se constată că nu există diferență între cele două variante de erodare, cel puțin pînă la limita rezoluției permise de ecran. Se mai constată că, în imaginea erodată sînt mai puține obiecte mici, cu aria de cîțiva pixeli, dar găurile au arii mai mari.
După încă o apăsare de tastă, se afişează imaginea dilatată. Se observă că s-au redus substanțial găurile, dar obiectele de dimensiuni mici persistă, ba chiar cu arii mai mari. În corpul programului, se observă utilizarea funcției imdilate.

3. Deschiderea şi închiderea imaginii
Programul continuă cu operațiile de deschidere și închidere. În corpul programului, se observă aplicarea dilatării pe imaginea erodată, apoi aplicarea erodării pe imaginea dilatată. După două apăsări de taste, apar cele două imagini prelucrate, lîngă imaginea segmentată inițială. Observați efectul operațiilor asupra găurilor și obiectelor mici.
În fine, încă o apăsare de tastă duce la afişarea imaginii supuse la închidere, apoi la deschidere.

4. Etichetarea
Pentru comparație, etichetarea a fost efectuată pe imaginea segmentată inițială și pe imaginea prelucrată prin închidere şi deschidere, I5. În program se observă două operații, legate de etichetare și extragerea trăsăturilor. Prima se referă la utilizarea funcției *bwlabel*, care furnizează matricea L, a etichetelor, şi numărul de obiecte detectate. Observați sintaxa funcției. A doua se referă la trăsăturile obiectelor și măsurările de poziție: ariile obiectelor, respectiv centrele lor de greutate. Măsurarea pe imagine se efectuează prin două bucle *for*, în care se examinează matricea L a etichetelor. Determinarea centrelor de greutate a fost introdusă numai pentru a doua versiune a imaginii, I5. Aria este numărul de pixeli din obiect, iar coordonatele centrului de greutate sînt sumele coordonatelor, împărțite la numărul de pixeli.
După prima apăsare de tastă, citiți în fereastra de comandă numărul de obiecte detectate şi ariile primelor 16 obiecte. Se observă numărul mult mai mare de obiecte, existente în imaginea inițială. Se mai observă că primele 4 arii mari sînt asemănătoare, în cele două liste, dar primele 3 obiecte mici, din prima listă, nu mai apar în cea de a doua.
După încă o apăsare de tastă se afişează coordonatele centrelor de greutate, din imaginea I5. Cu ajutorul lor, se pot identifica obiectele prezente în imagine. Ultima apăsare încheie programul.

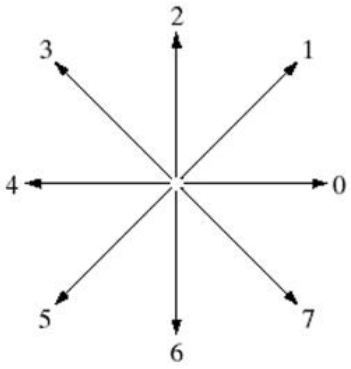
Laboratorul 13: Extragerea trăsăturilor, în scopul recunoaşterii

Breviar teoretic

Simultan cu etichetarea obiectelor, sau imediat după aceasta, este posibilă extragerea unor informații asociate cu etichetele. În această lucrare avem în vedere aria, limitele poziției în imagine, codul lanț al conturului și perimetrul, dar pot fi extrase și alte informații, cum ar fi numărul de găuri dintr-un obiect, proprietățile găurilor, factori de formă ai perimetrului etc.

În mare măsură, algoritmii sînt similare cu ceea ce a fost descris în lucrarea anterioară. Aria obiectului este numărul de pixeli avînd eticheta acelui obiect. Aria se determină în cursul etichetării, prin simpla numărare a pixelilor care au primit o anumită etichetă. Pozițiile extreme ale unui obiect, pe axele *x* și *y*, se determină tot în cursul etichetării, sau imediat după aceasta, prin compararea coordonatelor pixelului curent cu extremele memorate pînă la pixelul anterior. De remarcat că funcția din Matlab, *bwlabel*, realizează doar asocierea etichetelor. Celelalte informații trebuie extrase printr-o funcție de etichetare proprie, sau după terminarea funcției *bwlabel*, prin parcurgerea imaginii cu etichete, în două bucle *for*. Abscisa centrului de greutate al unui obiect se calculează prin împărțirea sumei absciselor pixelilor din acel obiect la arie. Ordonata urmează un algoritm similar. Conturul este format din pixelii obiectului, care au cel puțin un vecin tetraconetat, aparținînd fondului. De observat că, atunci cînd ne interesează prezența găurilor, se face distincția între conturul exterior și cel interior (conturul găurii).

Conturul este codificat prin codul lanț, conform cu codurile din figura de mai jos. În acest caz, a fost aleasă convenția cea mai frecventă, prin care pixelii obiectului sînt considerați octoconectați, iar pixelii fondului sînt tetraconectați. Din codul lanț se poate calcula perimetrul, însumînd numărul de cifre pare ale lanțului, cu numărul de cifre impare, înmulțit cu $\sqrt{2}$. Ca și în cazul ariei, unitatea de măsură a perimetrului este distanța dintre doi pixeli vecini, fără legătură cu lungimeaa din lumea fizică. Alți factori de formă a obiectului, momente etc., se pot calcula tot prin parcurgerea conturului. Trăsătura numită *compactness* este cel mai simplu de calculat, pentru că este raportul dintre pătratul perimetrului și arie. Din considerente geometrice, valoarea minimă a acestui raport este 4*π și corespunde cu obiectul în formă de cerc, indiferent de mărimea lui. Pentru informații topologice, legate de numărul de găuri, incluziune etc., este necesar un algoritm propriu de etichetare, întrucît funcția *bwlabel* asociază aceeași etichetă tuturor pixelilor de culoarea fondului (valoarea 0), indiferent de poziția lor, exterioară sau interioară obiectelor.



Modul de lucru

Fişiere folosite: L13.m, obiecte.jpg

1. Încărcarea, obținerea imaginii monocrome, segmentarea, filtrarea prin închidere și deschidere se realizează ca în lucrările precedente, pornind de la imginea din fișierul obiecte.jpg. Variabila dim este dimensiunea imaginii. Imediat după pornire, apare imaginea originală. După prima apăsare de

tastă, apare imaginea segmentată şi filtrată, lîngă cea originală. Pentru aceasta, în continuare, se foloseşte convenţia ca pixelii fondului să aibă valoarea 0, ca în funcţiile de transformare morfologică şi în funcţia *bwlabel*.

2. După o apăsare de tastă, are loc etichetarea, cu funcţia *bwlabel*. Variabila *L* este o matrice a etichetelor, cu aceeaşi dimensiune ca imaginea, iar variabila *num* este numărul de obiecte etichetate (exceptînd fondul, care primeşte eticheta 0). Se iniţializează variabilele care vor memora sumele coordonatelor, ariile, poziţiile extreme, pentru fiecare obiect. În continuare, imaginea este parcursă în două bucle for, în cuprinsul cărora sînt calculate variabilele de mai sus. Numărul de obiecte şi ariile lor sînt afişate în fereastra de comandă.

3. O nouă apăsare de tastă determină calculul şi afişarea coordonatelor centrelor de greutate (abscisa x, apoi ordonata y), pentru obiectele detectate. Pentru afişare, valorile sînt rotunjite, ca să corespundă cu poziţia unui pixel. Centrul de greutate al fondului nu are relevanţă.

4. Următoarele apăsări de taste păstrează imaginea segmentată în partea stîngă, iar în dreapta afişează, pe rînd, peste imaginea segmentată, cîte un cadru al unui obiect (*bounding box*). Afişarea se face în ordinea etichetelor, iar obiectele mici, cu arie sub 100 pixeli, sînt neglijate.

5. După afişarea cadrelor de mai sus, în partea dreaptă este afişată imaginea contururilor obiectelor. În funcţie de rezoluţia ecranului, este posibil ca aceste contururi să fie observate corect sau să lipsească fragmente mari din contur. Obţineţi cea mai bună detaliere dacă maximizaţi imaginea. În scopul afişării mai clare, am renunţat la o parte din imagine, am păstrat doar obiectele cu etichetele 3 şi 4. Contururile lor se afişează în figură nouă, după apăsarea de tastă.

6. Următoarea apăsare de tastă determină etichetarea noii imagini (imaginea restrînsă), care conţine doar contururi. Etichetele nu mai coincid cu cele precedente, întrucît contururile găurilor devin obiecte de sine stătătoare. În fereastra de comandă se afişează numărul de contururi găsite. Cu fiecare apăsare de tastă, se afişează unul dintre contururi. Tasta apăsată după afişarea ultimului contur trimite spre două bucle *for*, în care se determină numărul de pixeli ai fiecărui contur (variabila *arie1*) şi coordonatele primului pixel găsit în respectivul contur. Valorile din această variabilă se afişează în fereastra de comandă. Fără pauză, programul determină codul lanţ al unui singur contur: a fost ales conturul interior (eticheta 3), dar poate fi schimbată valoarea prin care este ales acesta. Pentru codificare, este necesară parcurgerea, în buclă *for*, a tuturor pixelilor conturului. La fiecare pas, este căutată direcţia următorului pixel din lanţ, mereu în acelaşi sens de rotaţie. Codurile sînt cele din figura inclusă mai sus, în breviarul teoretic. Codul lanţ este memorat în variabila *cod*, iar variabila numerică, prin care se afişează codul, este *codalfa*. Această secvenţă de program se încheie cu afişarea, în fereastra de comandă, a coordonatelor primului punct găsit, în acest contur, şi a codului lanţ asociat. Numărul de caractere al codului este mare, aşa încît afişarea lui depăşeşte lăţimea normală a ferestrei de comandă.

7. Ultima apăsare de tastă trece la o nouă buclă *for*, în care este examinat codul lanţ. Sînt numărate cifrele pare şi cele imapare, din care se calculează perimetrul. Rezultatele sînt afişate în fereastra de comandă.