7. Operaţii morfologice pe imagini binare

7.1. Introducere

Operaţiile morfologice pe imagini afectează forma sau structura unui obiect. Ele se aplică doar pe imagini binare (imagini cu doar două culori, alb şi negru). Operaţiile morfologice se folosesc de obicei ca etape de pre- sau post-procesare a imaginilor (filtrare, subţiere sau eliminare a protuberanţelor) sau pentru obţinerea unei reprezentări sau descrieri a formei obiectelor sau regiunilor (contururi, schelete, înfăşurători convexe).

7.2. Consideraţii teoretice

Operaţiile morfologice principale sunt *dilatarea* şi *eroziunea* [1]. Dilatarea măreşte obiectele, permiţând umplerea unor mici goluri şi conectarea obiectelor disjuncte. Eroziunea micşorează obiectele prin erodarea marginilor obiectelor. Aceste operaţii pot fi adaptate diverselor aplicaţii prin selectarea elementului structural folosit, care determină modul în care vor fi dilatate sau erodate obiectele.

Notaţii:

pixeli obiect: mulţimea pixelilor de interes (asupra cărora se aplica operaţiile morfologice)

pixeli fundal: complementul mulţimii pixelilor de obiect

7.2.1. Dilatarea

Dilatarea se face prin suprapunerea unui element structural, **B**, peste imaginea **A**, şi mutarea lui peste imagine, într-o manieră similară convoluţiei (care va fi prezentată în laboratorul următor). Diferenţa dintre convoluţia şi dilatarea folosind element structural este felul în care se aplică operaţia, convoluţia fiind definită ca un operator liniar iar dilatarea fiind definită cel mai bine prin următoarea secvenţă de paşi:

1. Dacă originea elementului structural se suprapune cu un pixel „fundal” din imagine, nu se efectuează nicio modificare şi se trece mai departe la următorul pixel.
2. Dacă originea elementului structural coincide cu un pixel „obiect” din imagine, atunci toţi pixelii acoperiţi de elementul structural devin pixeli „obiect”.

Notaţie:

$A \oplus B$

Elementul structural poate avea orice formă. Câteva forme des întâlnite sunt:

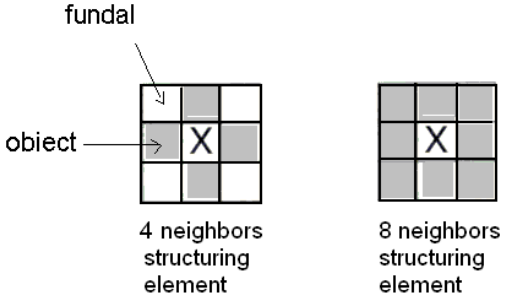


Fig. 7.1 Forme tipice pentru elementele structurale (**B**)

În Fig. 7.2 este prezentat un exemplu de dilatare. A se observa că în cazul operaţiei de dilatare toţi pixelii „obiect” vor fi păstraţi, marginile obiectelor vor fi extinse iar micile goluri vor fi umplute.

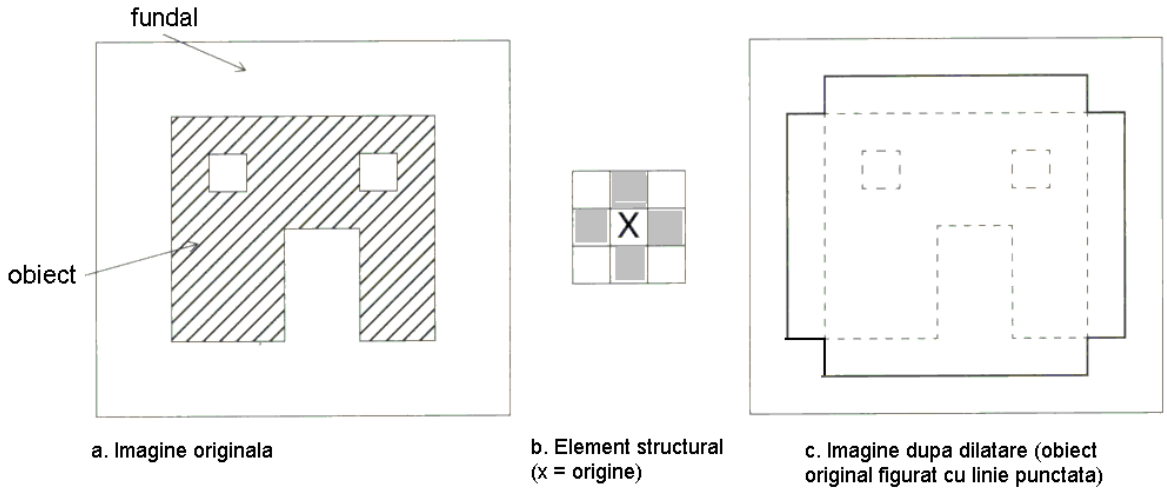


Fig. 7.2 Ilustrarea procesului de dilatare

a. b.

Fig. 7.3 Exemplu de dilatare (obiect = negru / fundal = alb): a. Imaginea originală **A**;
b. Imaginea rezultată în urma operaţiei: $\mathbf{A} \oplus \mathbf{B}$

7.2.2. Eroziunea

Operaţia de eroziune este similară cu cea de dilatare, dar anumiţi pixeli „obiect” vor fi transformaţi în pixeli „fundal”, invers decât la dilatare. Ca mai sus, elementul structural este deplasat peste imagine, dar se va folosi următoarea secvenţă de paşi:

1. Dacă originea elementului structural se suprapune peste un pixel „fundal” din imagine atunci nu se efectuează nicio modificare şi se trece la următorul pixel.

2. Dacă originea elementului structural se suprapune peste un pixel „obiect” din imagine şi există cel puţin un pixel „obiect” al elementului structural care se suprapune peste un pixel „fundal” din imagine, atunci pixelul curent din imagine va fi transformat în „fundal”.

Notaţie:

$A \ominus B$

În Fig. 7.4 au rămas doar pixelii „obiect” care coincid cu originea elementului structural dacă acesta s-a suprapus în întregime peste pixelii „obiect” ai unui obiect existent. Pentru că

elementul structural are 3 pixeli lăţime, partea din dreapta a obiectului a fost complet erodată, dar partea din stânga şi-a menţinut câţiva dintre pixeli, pentru că iniţial avea 3 pixeli lăţime.

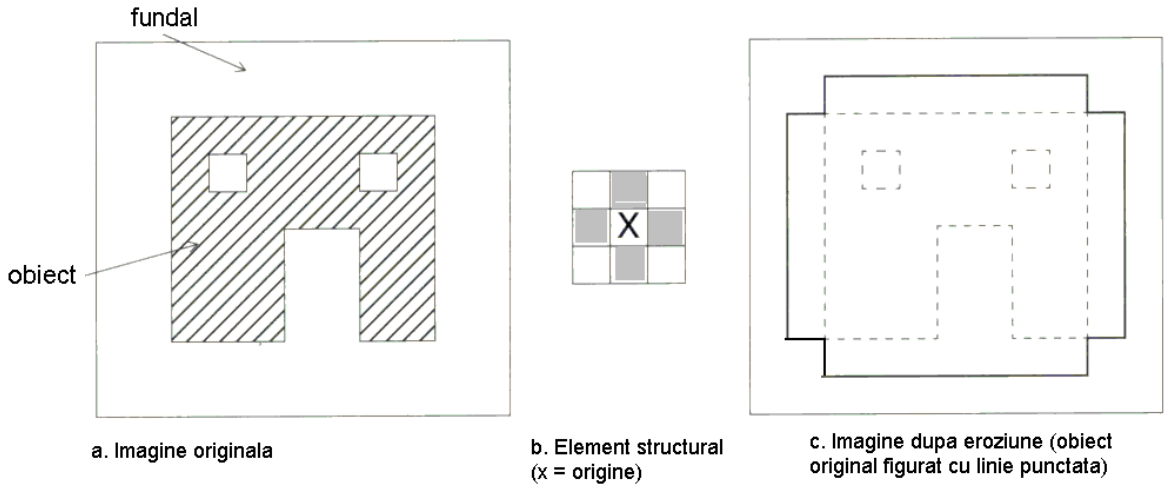
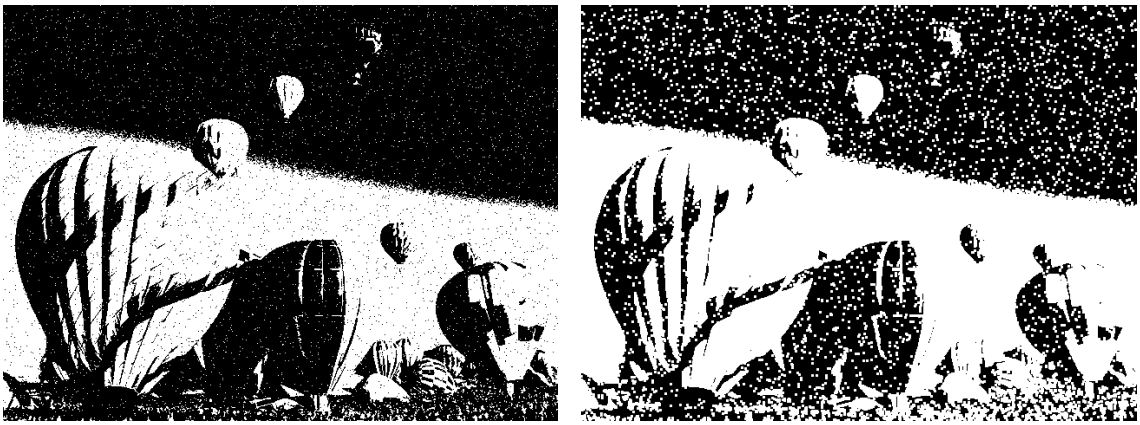


Fig. 7.4 Ilustrarea procesului de eroziune



a. b.

Fig. 7.5 Exemplu de eroziune (obiect = negru / fundal = alb): a. Imaginea originală **A**; b. Imaginea rezultat: **A** Θ **B**

7.2.3. Deschidere şi închidere

Cele două operaţii de bază, dilatarea şi eroziunea pot fi combinate în secvenţe de operaţii complexe. Cele mai utile operaţii de filtrare morfologică sunt deschiderea şi închiderea [1]. *Deschiderea* constă într-o eroziune urmată de o dilatare, şi poate fi folosită pentru eliminarea pixelilor din regiunile care sunt prea mici pentru a conţine elementul structural. În acest caz, elementul structural este adesea numit sondă, pentru că acesta caută obiectele prea mici şi le filtrează. Vezi Fig. 7.6 ca exemplu pentru deschidere.

Notaţie:

$$A \circ B = (A \Theta B) \oplus B$$

Închiderea constă într-o dilatare urmată de o eroziune, şi poate fi folosită pentru umplerea de goluri şi mici discontinuităţi. În Fig. 7.7 se poate vedea că operaţia de închidere are ca efect umplerea golurilor şi a discontinuităţilor. Comparând imaginile din partea stângă şi partea dreaptă a Fig. 7.8, putem observa că ordinea operaţiilor de dilatare şi eroziune este

importantă. Închiderea şi deschiderea au rezultate diferite, chiar dacă ambele constau dintr-o operaţie de eroziune şi una de dilatare.

Notaţie:

$$A \bullet B = (A \oplus B) \Theta B$$

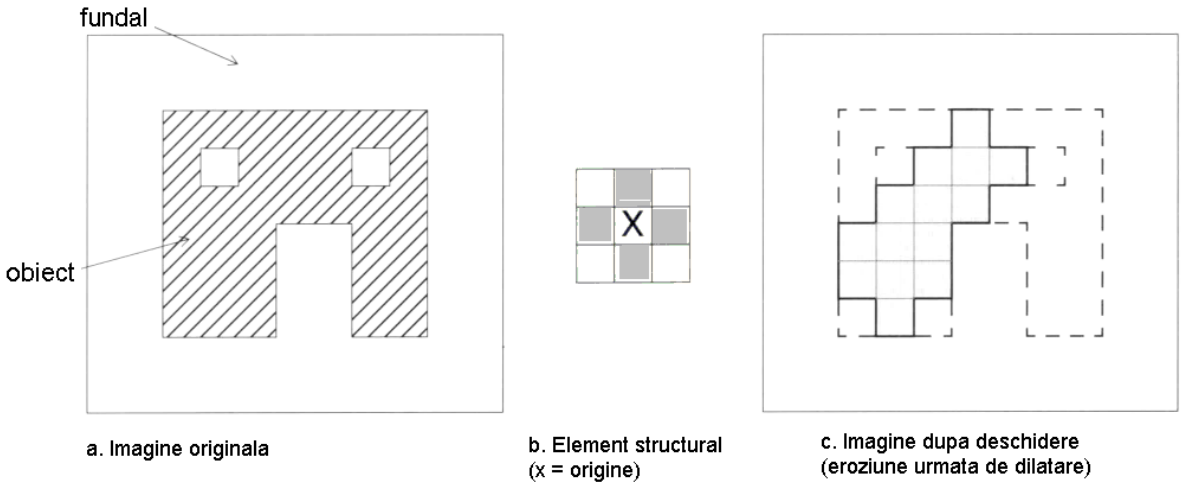


Fig. 7.6 Ilustrarea operaţiei de deschidere.

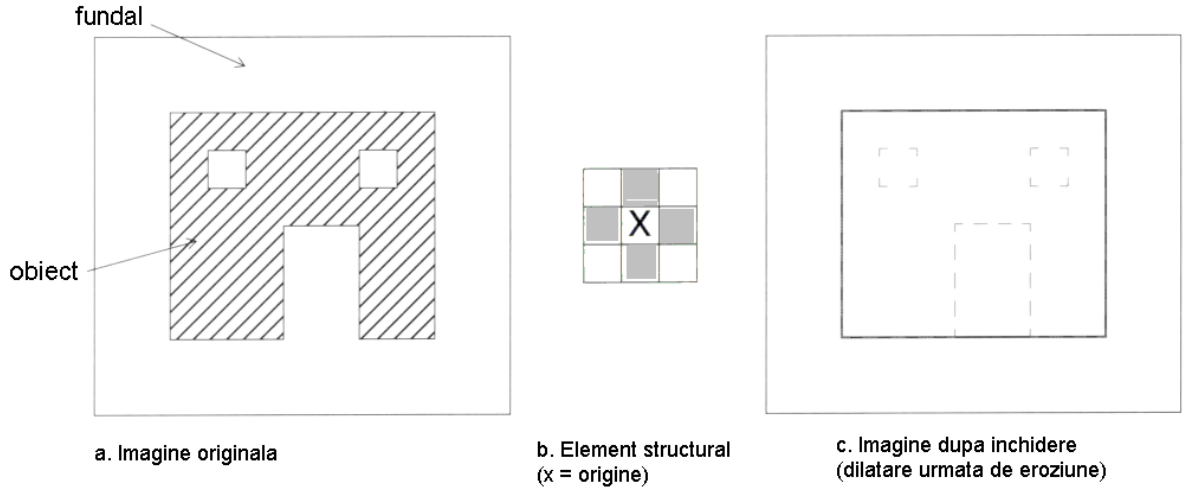


Fig. 7.7 Ilustrarea operaţiei de închidere (obiect = negru / fundal = alb):

a. b.

Fig. 7.8 Rezultatul deschiderii (a) şi al închiderii (b) aplicată pe imaginea originală, din Fig. 7.5a (obiect = negru / fundal = alb).

7.2.4. Prezentarea unor algoritmi morfologici de bază [2]

7.2.4.1. Extragerea conturului

Conturul unei mulţimi *A*, notat prin *β(A)*, poate fi obţinut prin erodarea mulţimii *A* cu elementul structural *B* urmată de efectuarea diferenţei dintre *A* şi rezultatul eroziunii ei, mai precis:

$$\beta(A) = A - (A \Theta B)$$

unde

B este un element structural.

'–' reprezintă operaţia de diferenţă a două mulţimi (ilustrată în Fig. 7.10)

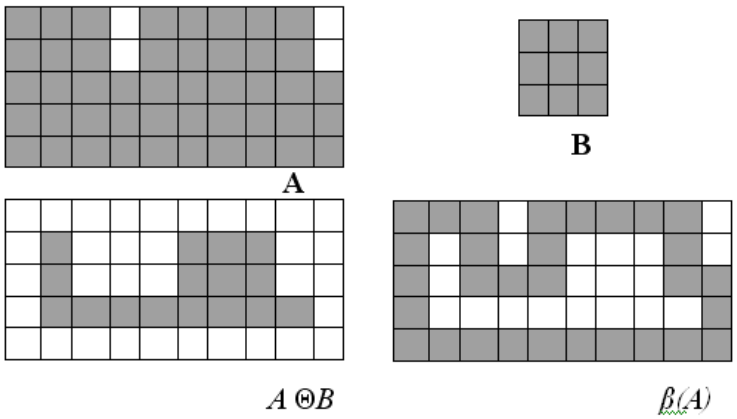


Fig. 7.9 Ilustrarea algoritmului de extragere a conturului

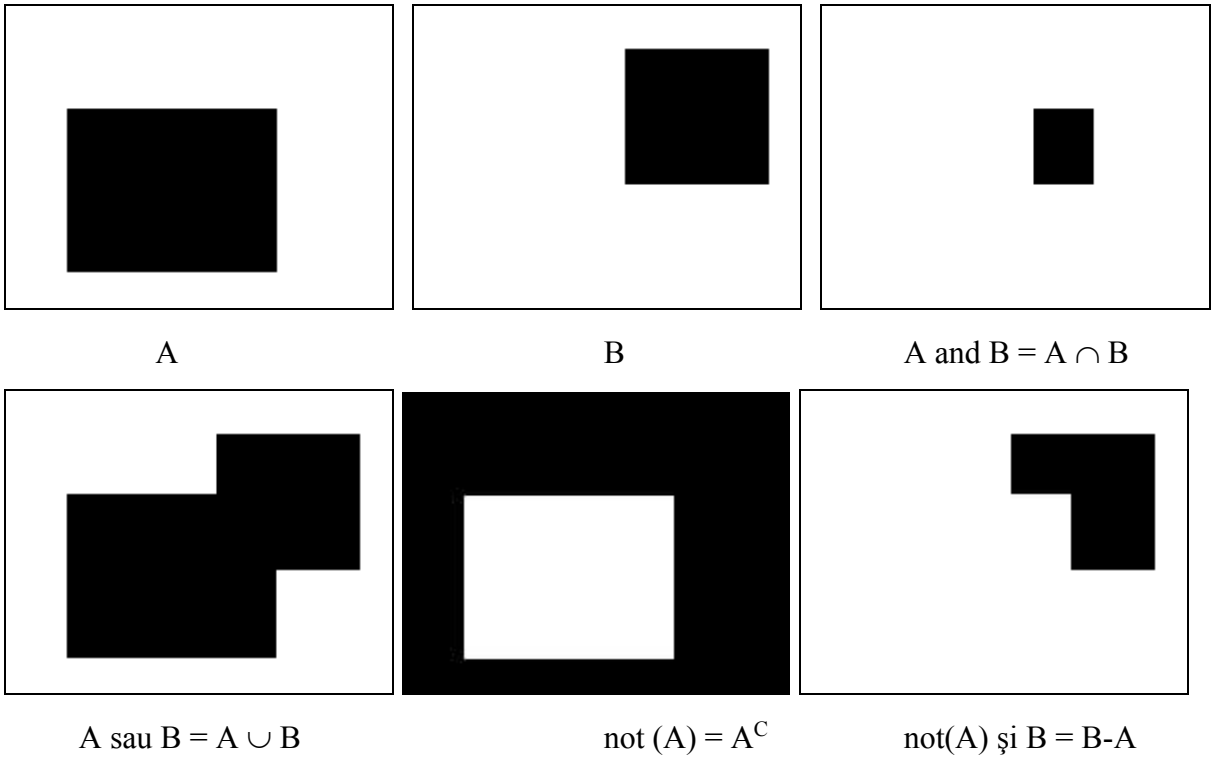


Fig. 7.10 Ilustrarea operaţii pe mulţimi (prin diagrame Venn-Euler)

7.2.4.2. Umplerea regiunilor

În continuare vom prezenta un algoritm simplu de umplere a regiunilor, bazat pe dilatări, complement şi intersecţii.

Pornind de la un punct *p* aflat în interiorul unei regiuni, obiectivul algoritmului este umplerea întregii regiuni cu pixeli „obiect". Dacă adoptăm convenţia că toate punctele care nu aparţin conturului regiunii sunt de „fundal", atunci vom atribui valoarea „obiect" punctului *p* la începutul algoritmului. Prin folosirea următorului algoritm vom umple întreaga regiune cu pixeli „obiect":

$$X_k = (X_{k-1} \oplus B) \cap A^C \qquad k=1,2,3,\ldots$$

unde

$X_0=p$,

B – reprezintă elementul structural

$\cap$ – reprezintă operatorul de intersecţie (vezi Fig. 7.10)

A^C – reprezintă complementul mulţimii A (vezi Fig. 7.10)

Algoritmul se termină atunci când la iteraţia k are loc egalitatea $X_k=X_{k-1}$. Reuniunea mulţimilor X_k şi A conţine atât interiorul mulţimii A cât şi conturul acesteia, ambele conţinând doar puncte „obiect".

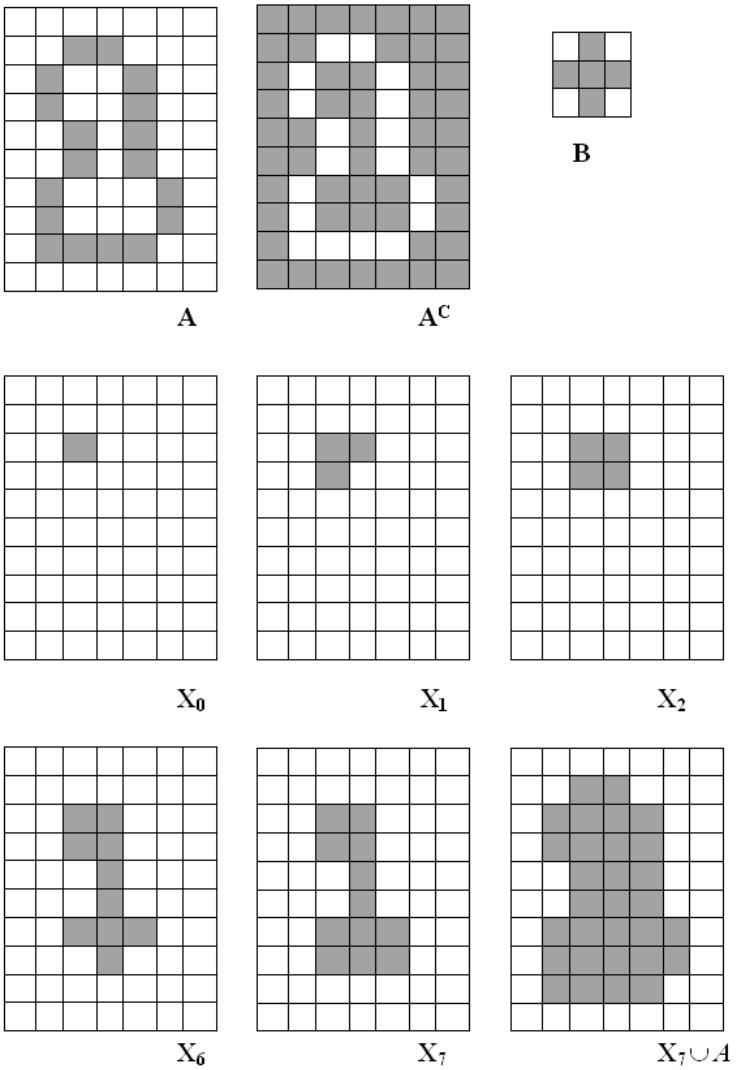


Fig. 7.11 Ilustrarea algoritmului de umplere a regiunilor

7.3. Detalii de implementare

7.3.1. Folosirea unui buffer suplimentar pentru procesări înlănţuite

Aplicarea operaţiilor morfologice (dilatare şi eroziune) trebuie făcută în felul următor:

Imagine destinaţie = *Imagine sursă* (operator) *Element structural*

Imaginea sursă nu trebuie să fie afectată în niciun fel!

Pentru implementarea operaţiilor morfologice complexe (deschidere şi închidere) sau a operaţiilor repetate (de exemplu: *n* eroziuni consecutive) într-o singură funcţie de procesare, este necesară crearea unui buffer de imagine suplimentar.

7.4. Activităţi practice

1. Adăugaţi la framework-ul OpenCV Application funcţii de procesare care să implementeze operaţiile morfologice de bază (dilatare, eroziune, deschidere, inchidere).
2. Adăugaţi facilităţi care să permită aplicarea în mod repetat (de *n* ori) a operaţiilor morfologice. Introduce numărul de repetiţii de la linia de comanda. Remarcati proprietatea de idempotenţa a deschiderii si inchiderii (vezi curs).
3. Implementaţi algoritmul de extragere a conturului.
4. Implementaţi algoritmul de umplere a regiunilor.
5. **Salvaţi-vă ceea ce aţi lucrat. Utilizaţi aceeaşi aplicaţie în laboratoarele viitoare. La sfârşitul laboratorului de procesare a imaginilor va trebui să prezentaţi propria aplicaţie cu algoritmii implementaţi!!!**

Referinţe

[1]. Umbaugh Scot E, *Computer Vision and Image Processing*, Prentice Hall, NJ, 1998, ISBN 0-13-264599-8

[2] R.C.Gonzales, R.E.Woods, *Digital Image Processing. 2-nd Edition*, Prentice Hall, 2002.