

7. Operații morfologice pe imagini binare

7.1. Introducere

Operațiile morfologice pe imagini afectează forma sau structura unui obiect. Ele se aplică doar pe imagini binare (imagini cu doar două culori, alb și negru). Operațiile morfologice se folosesc de obicei ca etape de pre- sau post-procesare a imaginilor (filtrare, subțiere sau eliminare a protuberanțelor) sau pentru obținerea unei reprezentări sau descrieri a formei obiectelor sau regiunilor (contururi, schelete, înfășurători convexe).

7.2. Considerații teoretice

Operațiile morfologice principale sunt *dilatarea* și *eroziunea* [1]. Dilatarea mărește obiectele, permițând umplerea unor mici goluri și conectarea obiectelor disjuncte. Eroziunea micșorează obiectele prin erodarea marginilor obiectelor. Aceste operații pot fi adaptate diverselor aplicații prin selectarea elementului structural folosit, care determină modul în care vor fi dilatate sau erodate obiectele.

Notății:

pixeli obiect: mulțimea pixelilor de interes (asupra cărora se aplica operațiile morfologice)

pixeli fundal: complementul mulțimii pixelilor de obiect

7.2.1. Dilatarea

Dilatarea se face prin suprapunerea unui element structural, **B**, peste imaginea **A**, și mutarea lui peste imagine, într-o manieră similară convoluției (care va fi prezentată în laboratorul următor). Diferența dintre convoluția și dilatarea folosind element structural este felul în care se aplică operația, convoluția fiind definită ca un operator liniar iar dilatarea fiind definită cel mai bine prin următoarea secvență de pași:

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

Notăție:

$$A \oplus B$$

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

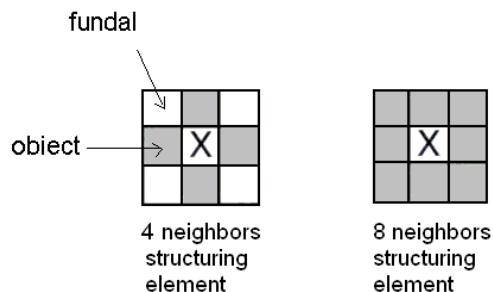


Fig. 7.1 Forme tipice pentru elementele structurale (**B**)

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

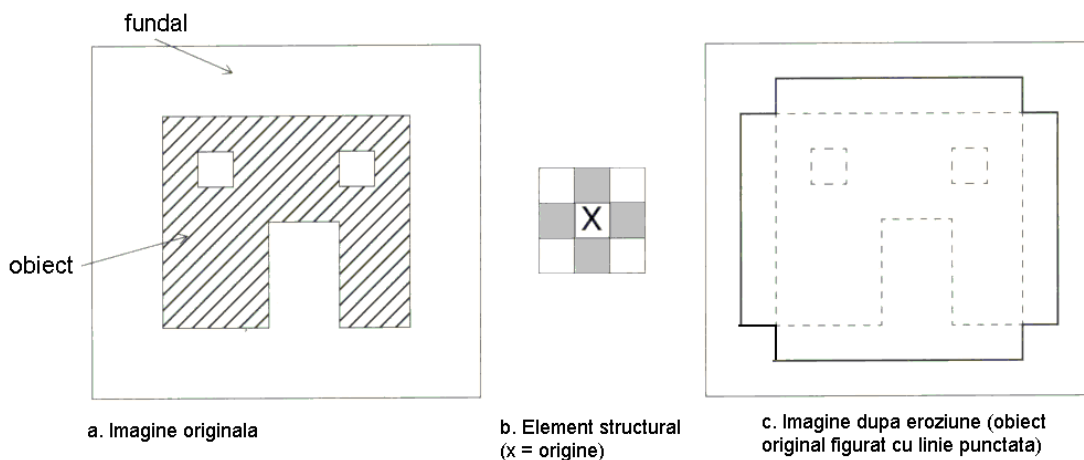


Fig. 7.2 Ilustrarea procesului de dilatare



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

7.2.2. Eroziunea

Operația de eroziune este similară cu cea de dilatare, dar anumiți pixeli „obiect” vor fi transformați în pixeli „fundal”, invers decât la dilatare. Ca mai sus, elementul structural este deplasat peste imagine, dar se va folosi următoarea secvență de pași:

1. Dacă originea elementului structural se suprapune peste un pixel „fundal” din imagine atunci nu se efectuează nicio modificare și se trece la următorul pixel.
2. Dacă originea elementului structural se suprapune peste un pixel „obiect” din imagine și există cel puțin 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”.

Notăție:

$$A \ominus B$$

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 lățime, partea din dreapta a obiectului a fost complet erodată, dar partea din stânga și-a menținut câțiva dintre pixeli, pentru că inițial avea 3 pixeli lățime.

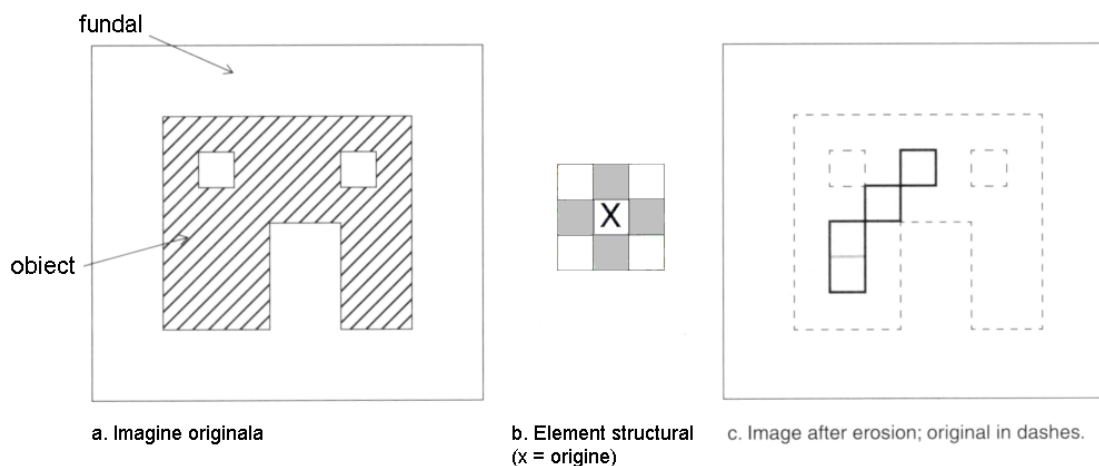


Fig. 7.4 Ilustrarea procesului de eroziune



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

7.2.3. Deschidere și închidere

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

Notăție:

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

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

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

Notăție:

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

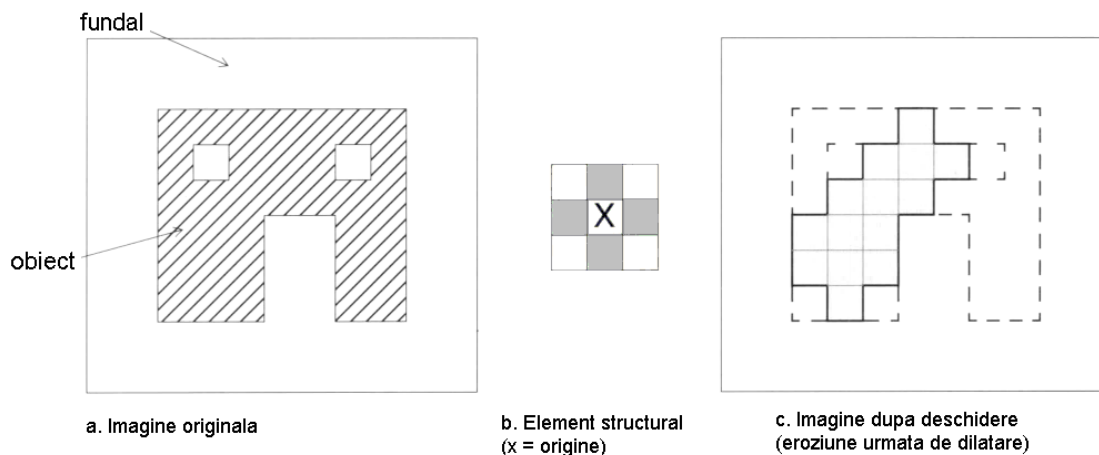


Fig. 7.6 Ilustrarea operației de deschidere.

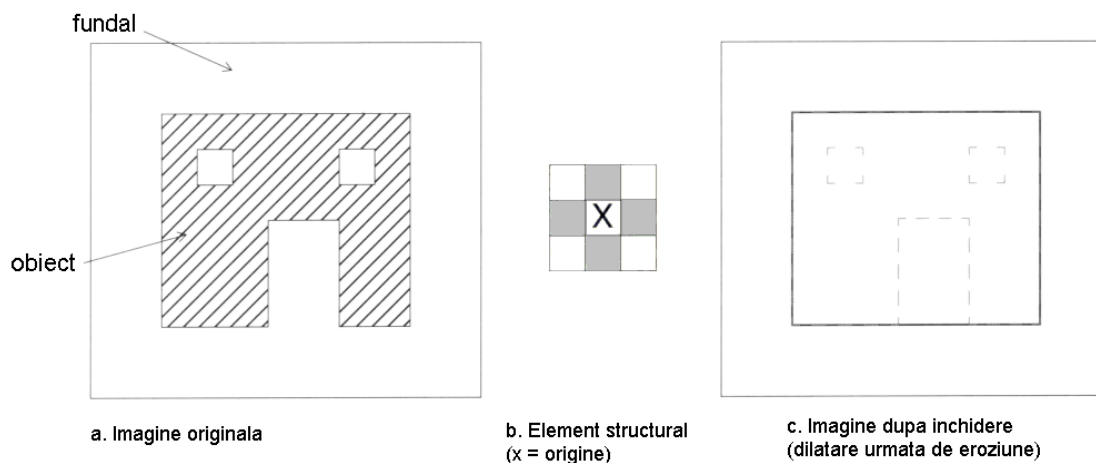


Fig. 7.7 Ilustrarea operației de închidere (obiect = negru / fundal = alb):



Fig. 7.8 Rezultatul deschiderii (a) și 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 mulțimi A , notat prin $\beta(A)$, poate fi obținut prin erodarea mulțimii A cu elementul structural B urmată de efectuarea diferenței dintre A și rezultatul eroziunii ei, mai precis:

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

unde

B este un element structural.

'-' reprezintă operația de diferență a două mulțimi (ilustrată în Fig. 7.10)

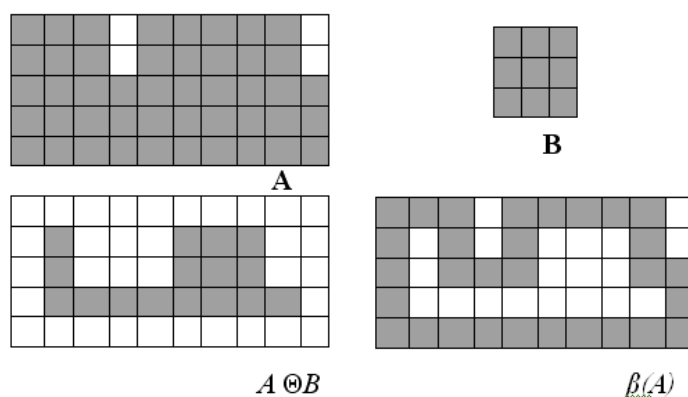


Fig. 7.9 Ilustrarea algoritmului de extragere a conturului

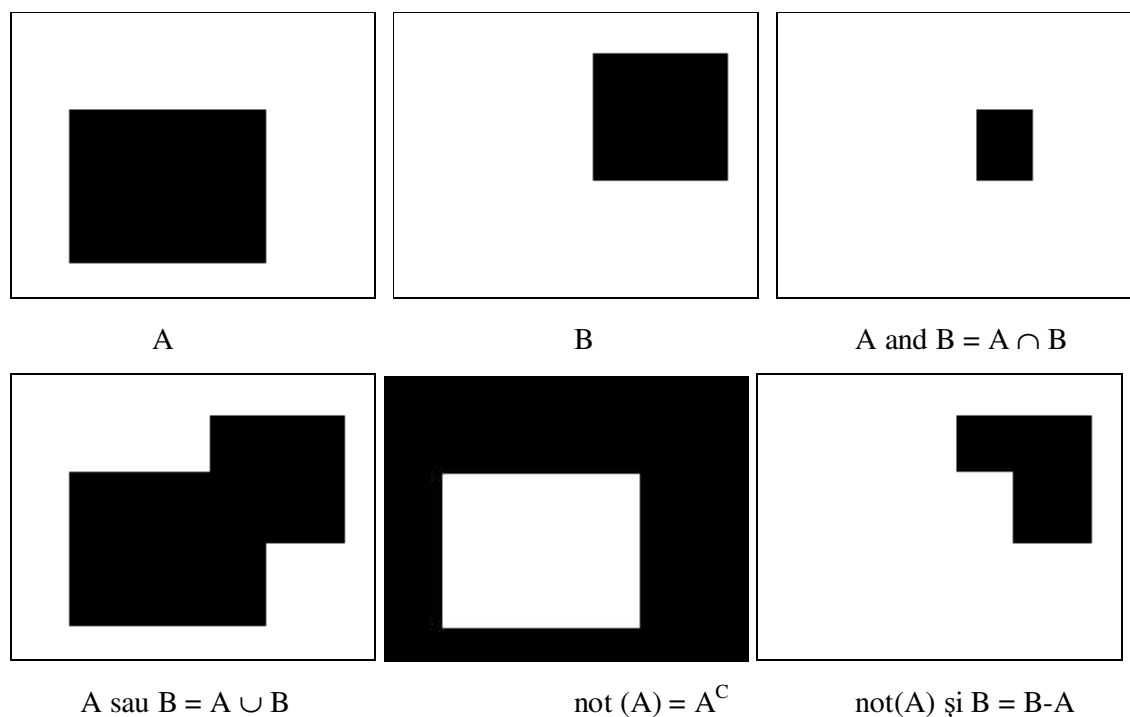


Fig. 7.10 Ilustrarea operațiilor pe mulțimi (prin diagrame Venn-Euler)

7.2.4.2. Umplerea regiunilor

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

Pornind de la un punct p aflat în interiorul unei regiuni, obiectivul algoritmului este umplerea întregii regiuni cu pixeli „obiect”. Dacă adoptăm convenția că toate punctele care nu aparțin 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”:

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

unde

$X_0 = p$,

B – reprezintă elementul structural

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

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

Algoritmul se termină atunci când la iterația k are loc egalitatea $X_k = X_{k-1}$. Reuniunea mulțimilor X_k și A conține atât interiorul mulțimii A cât și conturul acesteia, ambele conținând doar puncte „obiect”.

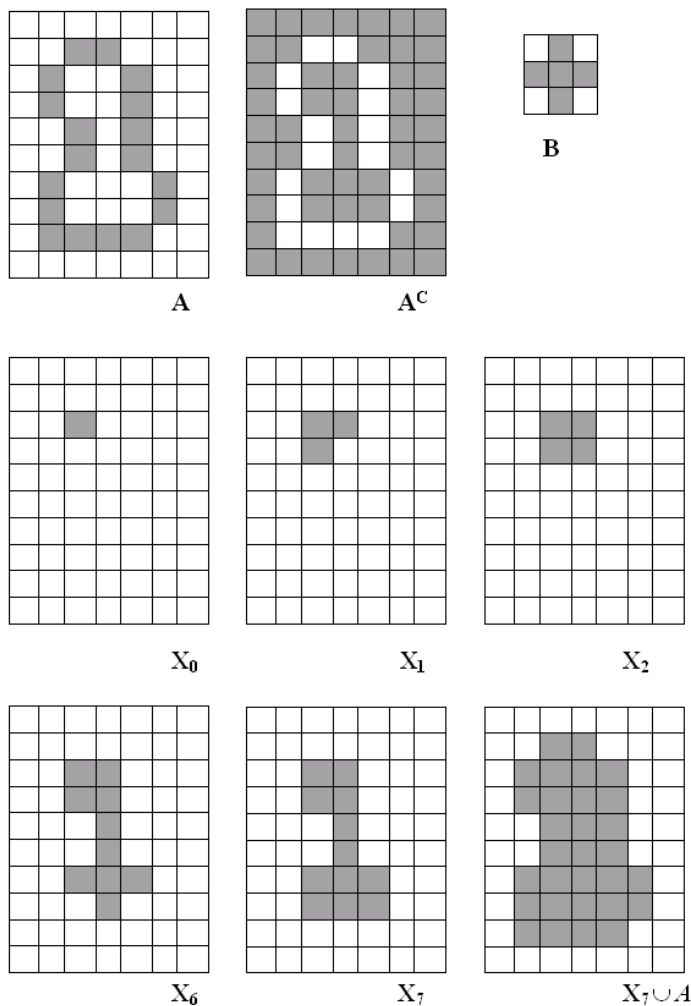


Fig. 7.11 Ilustrarea algoritmului de umplere a regiunilor

7.3. Detalii de implementare

7.3.1. Folosirea unui buffer suplimentar pentru procesări înlănțuite

Aplicarea operațiilor morfologice (dilatare și eroziune) trebuie făcută în felul următor:

$$\text{Imagine destinație} = \text{Imagine sursă} (\text{operator}) \text{Element structural}$$

Imaginea sursă nu trebuie să fie afectată în niciun fel!

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

7.4. Activități practice

1. Adăugați la framework-ul OpenCV Application funcții de procesare care să implementeze operațiile morfologice de bază (dilatare, eroziune, deschidere, închidere).
2. Adăugați facilități care să permită aplicarea în mod repetat (de n ori) a operațiilor morfologice. Introduce numărul de repetiții de la linia de comanda. Remarcati proprietatea de idempotență a deschiderii și închiderii (vezi curs).
3. Implementați algoritmul de extragere a conturului.
4. Implementați algoritmul de umplere a regiunilor.
5. **Salvați-vă ceea ce ați lucrat. Utilizați aceeași aplicație în laboratoarele viitoare. La sfârșitul laboratorului de procesare a imaginilor va trebui să prezentați propria aplicație cu algoritmi implementați!!!**

Referințe

- [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.