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 şi negru). Operațiile morfologice se folosesc de obicei la 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, înfăşurători convexe).

7.2. Considerații teoretice

Operațiile morfologice principale sunt *dilatarea* şi *eroziunea* [1]. Dilatarea măreşte obiectele, permițând umplerea unor mici goluri şi conectarea obiectelor disjuncte. Eroziunea micşorează 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.

Notații:

pixel negru: în cazul imaginilor grayscale cu paletă, având 8 biți/pixel, este un pixel având indexul (valoare) 0

pixel alb: în cazul imaginilor grayscale cu paletă, având 8 biți/pixel, este un pixel având indexul (valoare) 255

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 convoluției (care va fi prezentată în laboratorul următor). Diferența dintre convoluția şi 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 paşi:

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

Notație:

$A \oplus B$

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

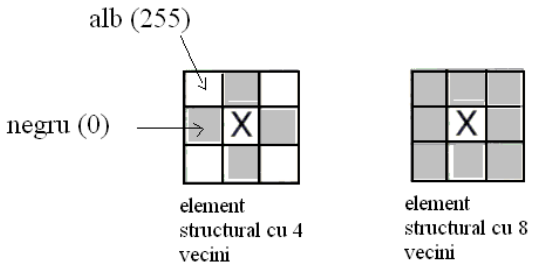


Fig. 7.1 Forme tipice pentru elementele structurale (**B**)

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

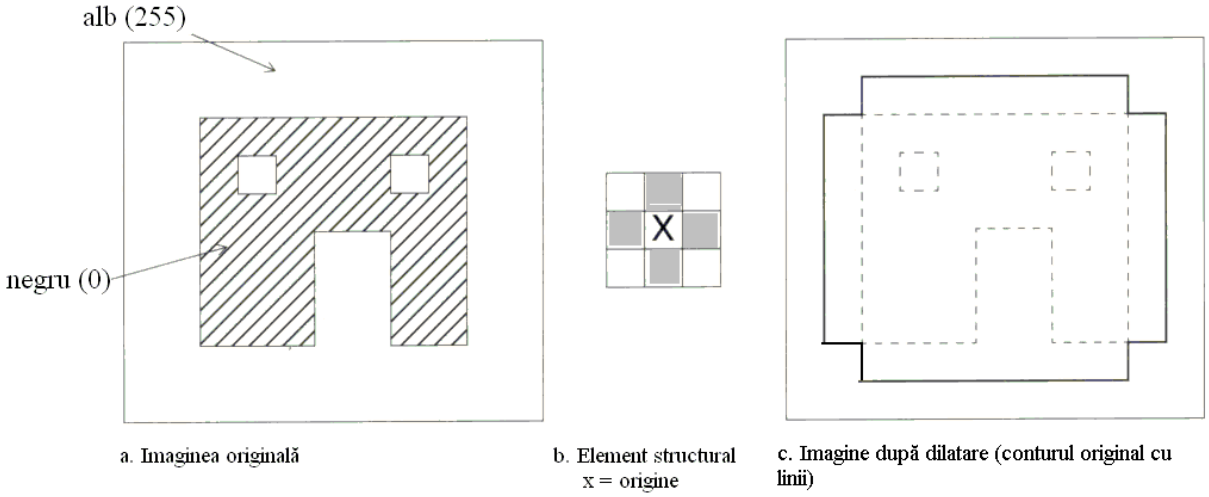


Fig. 7.2 Ilustrarea procesului de dilatare

a. b.

Fig. 7.3 Exemplu de dilatare: a. Imaginea originală **A**; b. Imaginea rezultată în urma operației: $\mathbf{A} \oplus \mathbf{B}$

7.2.2. Eroziunea

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

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

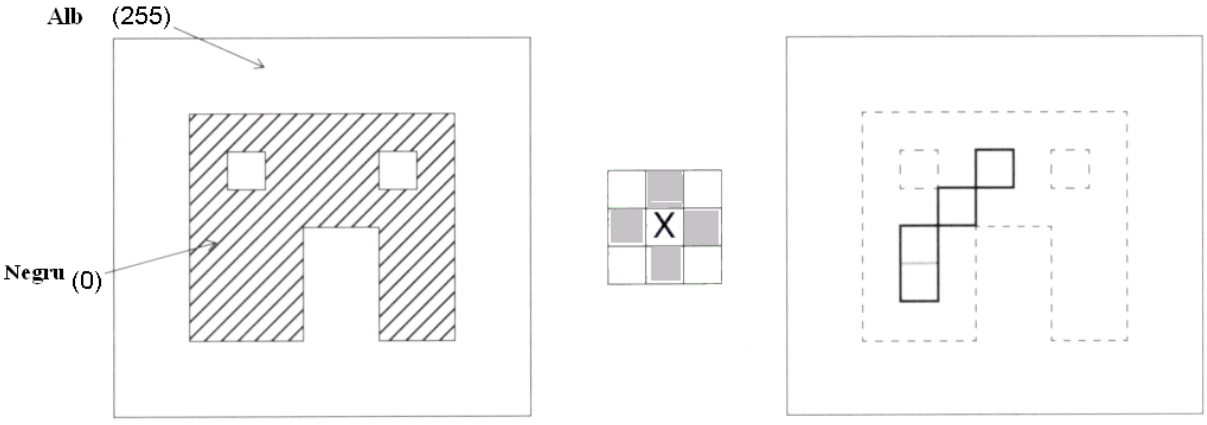
2. Dacă originea elementului structural se suprapune peste un pixel „negru” din imagine şi există cel puțin un pixel „negru” al elementului structural care se suprapune peste un pixel „alb” din imagine, atunci pixelul curent din imagine va fi transformat într-unul „alb”.

Notație:

$A \ominus B$

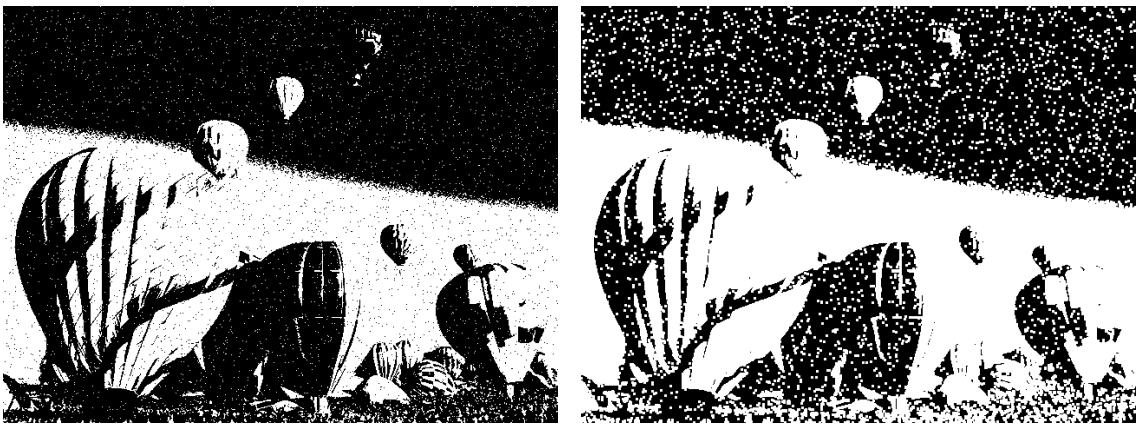
În Fig. 7.4 au rămas doar pixelii negri care coincid cu originea elementului structural dacă acesta s-a suprapus în întregime peste pixelii negri ai unui obiect existent. Pentru că elementul structural are 3 pixeli lățime, partea din dreapta a obiectului a fost complet erodată,

dar partea din stânga şi-a menținut câțiva dintre pixeli, pentru că inițial avea 3 pixeli lățime.



a. Imaginea originală b. Element structural x = origine c. Imagine după eroziune(conturul original cu linii)

Fig. 7.4 Ilustrarea procesului de eroziune



a. b.

Fig. 7.5 Exemplu de eroziune: a. Imaginea originală **A**; b. Imaginea rezultat: $\mathbf{A} \ominus \mathbf{B}$

7.2.3. Deschidere şi închidere

Cele două operații de bază, dilatarea şi eroziunea pot fi combinate în secvențe de operații complexe. Cele mai utile operații 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 conține 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.

Notație:

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

Închiderea constă într-o dilatare urmată de o eroziune, şi poate fi folosită pentru umplerea de goluri şi mici discontinuități. În Fig. 7.7 se poate vedea că operația de închidere are ca efect umplerea golurilor şi a discontinuităților. Comparând imaginile din partea stângă şi partea dreaptă a Fig. 7.8, putem observa că ordinea operațiilor de dilatare şi eroziune este importantă. Închiderea şi deschiderea au rezultate diferite, chiar dacă ambele constau dintr-o operație de eroziune şi una de dilatare.

Notație:

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

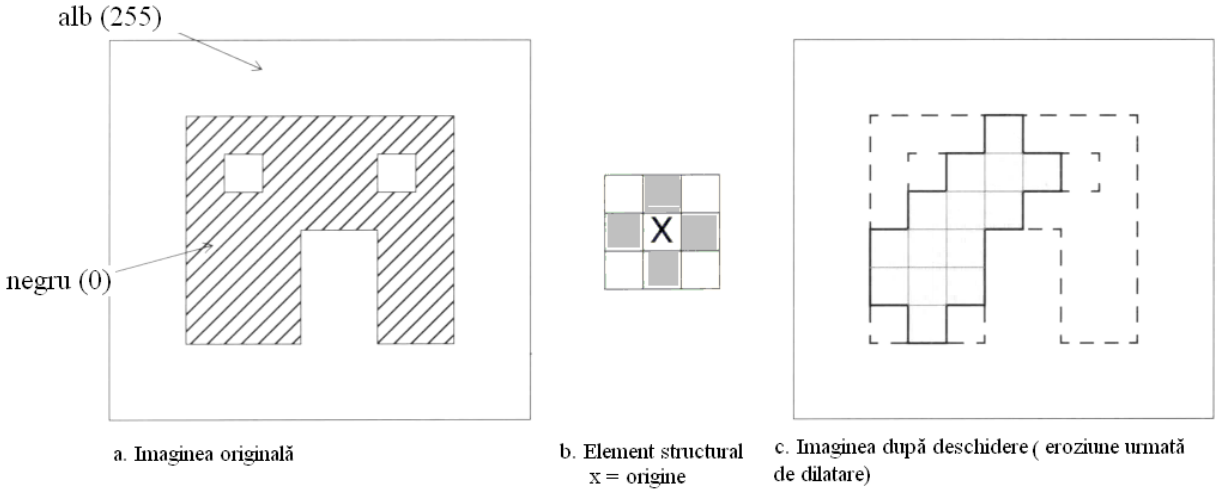


Fig. 7.6 Ilustrarea operației de deschidere

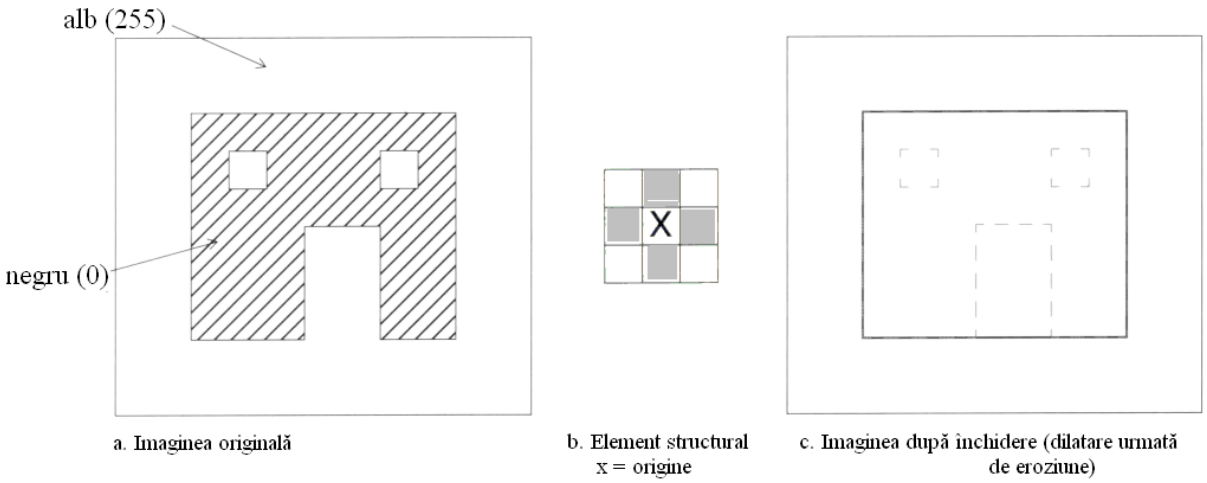


Fig. 7.7 Ilustrarea operației de închidere

a. b.

Fig. 7.8 Rezultatul deschiderii (a) şi al închiderii (b) aplicată pe imaginea originală, din Fig. 7.5a

7.2.4. Prezentarea unor algoritmi morfologici de bază [2]

7.2.4.1. Extragerea conturului

Conturul unei mulțimi *A*, notat prin *β(A)*, poate fi obținut prin erodarea mulțimii *A* cu elementul structural *B* urmată de efectuarea diferenței dintre *A* şi rezultatul eroziunii ei, mai precis:

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

unde

B este un element structural.

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

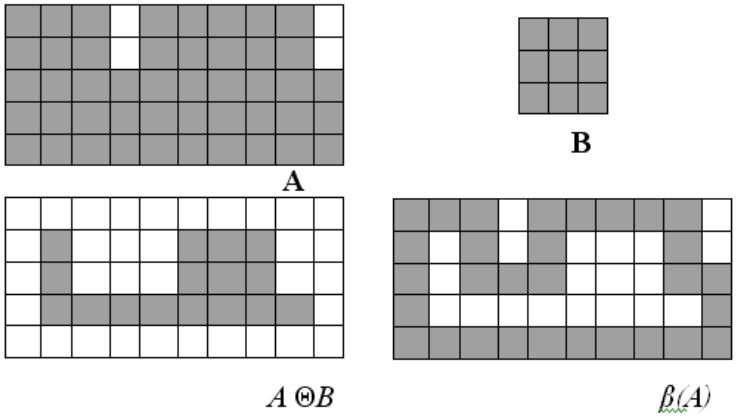


Fig. 7.9 Ilustrarea algoritmului de extragere a conturului

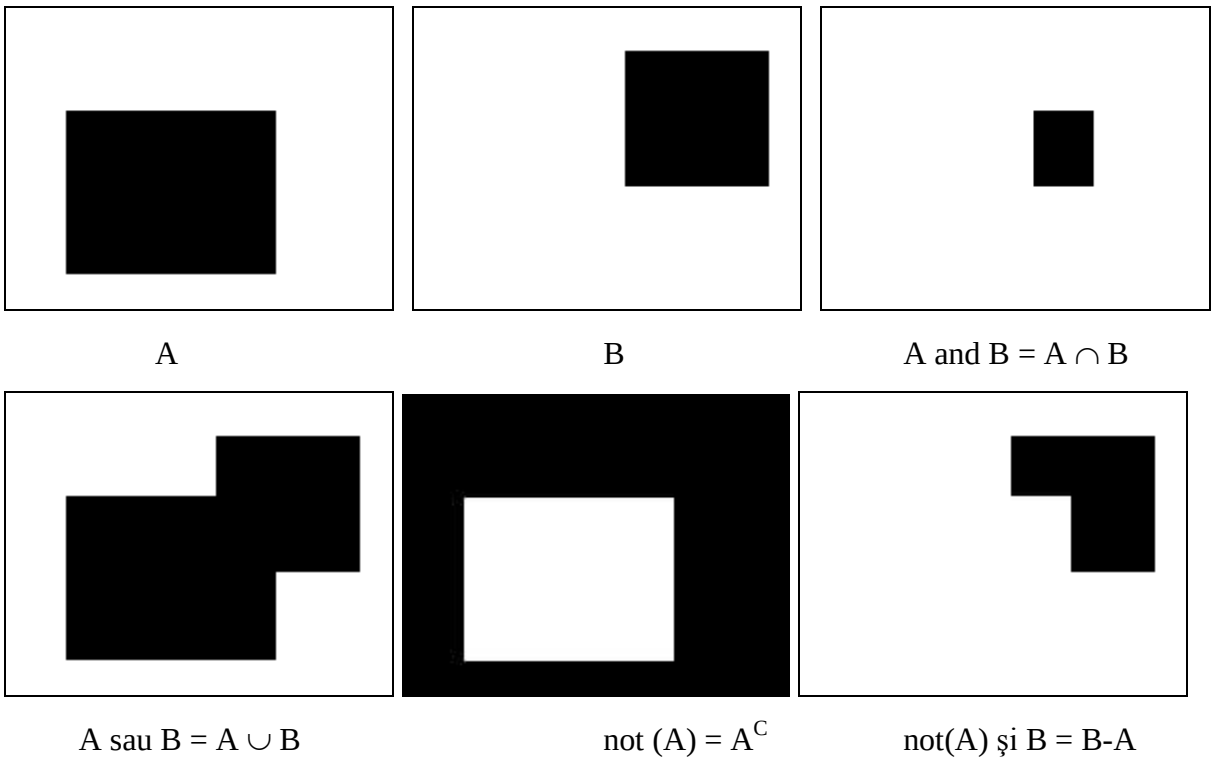


Fig. 7.10 Ilustrarea operații pe mulțimi (prin diagrame Venn-Euler)

7.2.4.2. Umplerea regiunilor

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

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

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

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 „negre”.

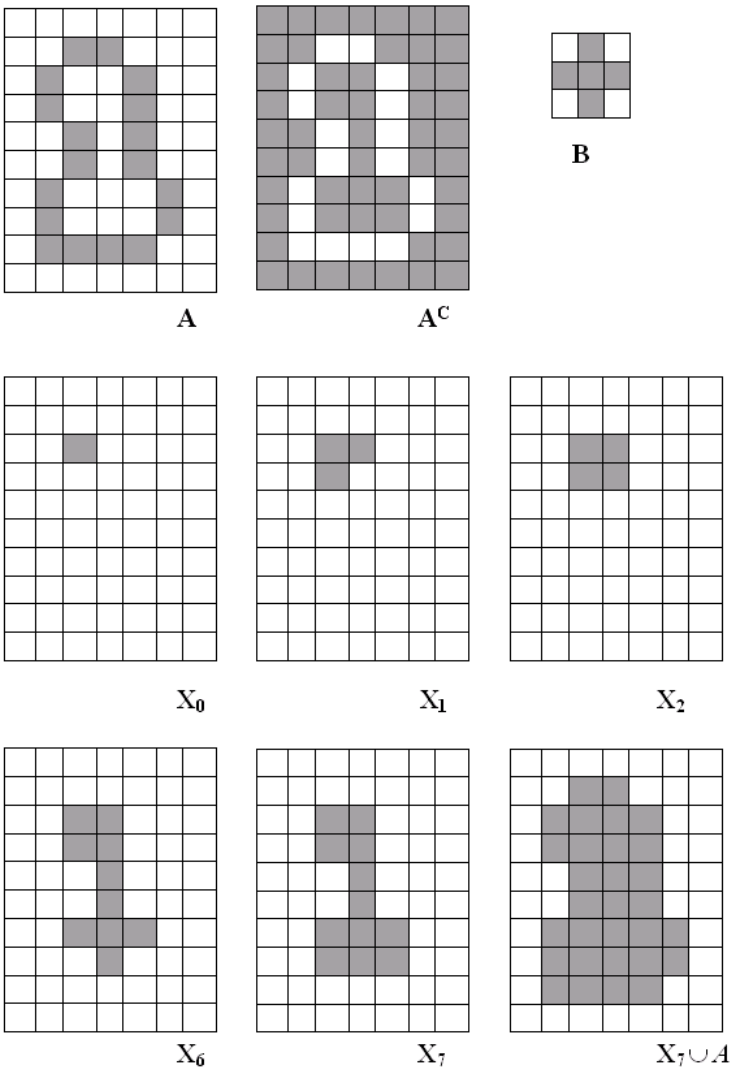


Fig. 7.11 Ilustrarea algoritmului de umplere a regiunilor

7.3. Detalii de implementare

7.3.1. Folosirea unui buffer suplimentar pentru procesări înlănțuite

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

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

Imaginea sursă nu trebuie să fie afectată în niciun fel!

Pentru implementarea operațiilor morfologice complexe (deschidere şi închidere) sau a operațiilor repetate (de exemplu: *n* eroziuni consecutive) într-o singură funcție de procesare, este necesară crearea unui buffer de imagine suplimentar. Aceasta se poate face în modul următor (codul ilustrează implementarea a două dilatări consecutive folosind un buffer auxiliar):

```
//alocare buffer temporar
//dimensiunile sale trebuie să fie w şi dwHeight (la fel
//ca şi cele ale matricii de pixeli din imaginea sursă sau destinație
unsigned char *lpTemp = new unsigned char [w*dwHeight];
//aplică prima dilatare şi scrie rezultatul
//în buffer-ul temporar.
//aplică a doua dilatare, şi scrie rezultatul
//în imaginea destinație
//eliberează buffer-ul! C++ nu are mecanism de garbage collection
//foloseşte operatorul delete[] nu delete!!
delete[] lpTemp;
```

7.3.2. Detalii de implementare suplimentare pentru proiectarea ferestrelor de dialog

Selectarea operației morfologice dorite şi a numărului de repetiții al ei se poate face printr-o fereastră de dialog. Crearea unei ferestre de dialog si adăugarea de controale de tip edit box a fost prezentată în laboratorul 2.

Următorul exemplu ilustrează implementarea unei operații de selecție a unor opțiuni multiple (de exemplu tipul de operație morfologică, dilatare, eroziune, deschidere, închidere, extragerea conturului, umplerea de regiuni) folosind butoane „radio”. Un buton radio este similar unui check box, diferența fiind dată de faptul că doar un singur buton radio din cadrul unui grup poate fi selectat la un moment dat. Pentru a crea un grup de butoane radio, permiterea interacțiunii utilizatorului cu el şi obținerea butonului selectat se vor parcurge următorii paşi (vezi Fig. 7.12):

1. Primul pas constă în crearea unui dialog şi adăugarea clasei corespunzătoare acestuia *CDlgSelectMorphologicalOperation* (vezi laboratorul 2!).
2. Pentru a grupa vizual butoanele radio, se va adăuga un control de tip Group Box la dialog. Dați un nume sugestiv group box-ului, cum ar fi “Operation Type”.
3. Adăugați primul buton radio la group box. Dați-i acestuia un ID sugestiv, cum ar fi IDC_OPERATION_TYPE. Selectați opțiunea Group!! Totodată, pentru acest buton şi următoarele, aveți grijă să selectați opțiunea „Auto” din secțiunea „Appearance”!!
4. Adăugați celelalte butoane radio. Nu este necesar să le modificați ID-ul. Aveți grijă ca opțiunea „Group” să NU fie selectată pentru acestea!!
5. Asociați o variabilă membru de tip întreg primului radio button (cel pentru care ați selectat opțiunea group).
6. Variabila membru asociată va conține index-ul radio-button-ului selectat. Indexul porneşte de la 0, astfel încât dacă primul buton este selectat indexul va fi 0, dacă este selectat al doilea buton atunci indexul selectat va fi 1 şi aşa mai departe.

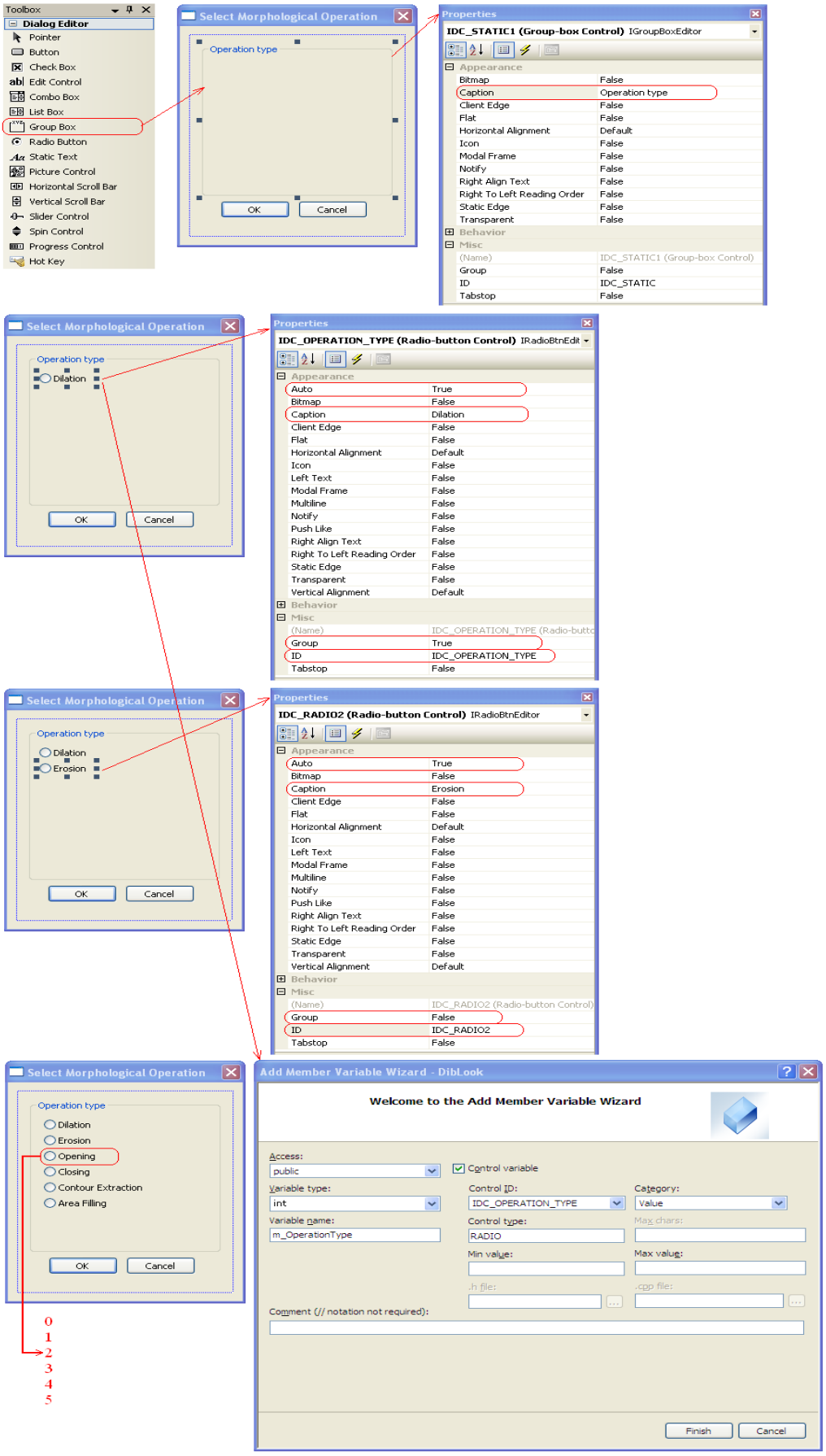


Fig. 7.12 Adăugarea unui group box, a butoanelor radio şi asocierea primului buton radio cu o variabilă membru index, de tip întreg

Un mod elegant de a obține opțiunea selectată dintr-un grup de butoane radio este următorul:

1. Definiți o enumerare în cadrul clasei dialog (*CDlgSelectMorphologicalOperation* în exemplul curent) care să conțină opțiunile dorite. Această enumerare trebuie adăugată în fișierul header al clasei (*DlgSelectMorphologicalOperation.h* în acest exemplu), în secțiunea de membri publici ai acesteia:

```
class CDlgSelectMorphologicalOperation : public CDialog {
public:
   //enumerare pentru reprezentarea operației dorite
   //(enumerările încep implicit de la valoarea 0)
   enum EOperationType {
         Dilation,
         Erosion,
         Opening,
         Closing,
         ContourExtraction,
         AreaFilling,
  };
    ……
    };
```

2. Adăugați cod în metoda de procesare (în fișierul *dibview.cpp*) pentru instanțierea dialogului și obținerea opțiunii selectate:

```
//instanțiați clasa dialog
CDlgSelectMorphologicalOperation dlgSelect;
//setați opțiunea implicită ca fiind prima
dlgSelect.m_OperationType = 0;
//afișați modal dialogul
dlgSelect.DoModal();
//obțineți opțiunea selectată
switch(dlgSelect.m_OperationType) {
case CDlgSelectMorphologicalOperation::Dilation:
      …… //cod pentru dilatare
      break; //nu uitați să puneți câte un break după fiecare caz
case CDlgSelectMorphologicalOperation::Erosion:
      …… //cod pentru eroziune
      break;
case CDlgSelectMorphologicalOperation::Opening:
      …… //cod pentru deschidere
      break;
case CDlgSelectMorphologicalOperation::Closing:
      …… //cod pentru închidere
      break;
case CDlgSelectMorphologicalOperation::ContourExtraction:
      …… //cod pentru extragerea conturului
      break;
case CDlgSelectMorphologicalOperation::AreaFilling:
      …… //cod pentru umplerea de regiuni
      break;
}
```

7.4. Activități practice

1. Adăugați la framework-ul DIBLook funcții de procesare care să implementeze operațiile morfologice de bază.
2. Adăugați facilități care să permită aplicarea în mod repetat (de *n* ori) a operațiilor morfologice. Folosiți o fereastră de dialog pentru a introduce numărul de repetiții (printr-un control de tip edit box) și pentru a selecta tipul operației dorite (folosind butoane „radio”).
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 algoritmii 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.