|  |  |
| --- | --- |
| **Morfologie matematică**  **1.Notiunea de obiect și de fundal**  **2.Elementul structurant**  **3.Dilatarea**  **4.Erodarea**  **5.Operațiile de deschidere și închidere**  **6.Gradientul morfologic, conturul interior și exterior** | **7. Transformări morfologice**  **7.1 Transformarea Black Hat**  **7.2 Transformarea Top-Hat**  **7. Transformarea Hit-or-Miss**  **8. Scheletonizarea**  **9. Procesarea imaginilor cu Skimage**  **9. Operații morfologice pentru niveluri de gri**  **Aplicații**  **Webografie** |

**Cursul 3**

**Morfologie matematică**

Operațiile morfologice se aplică numai pe **imagini binare[[1]](#footnote-1)** și au ca efect modificarea structurii obiectelor coțiunte de imagine. Operațiile morfologice sunt etape intermediare de procesare a imaginii, în etapele de pre- sau post-procesare acestea se folosesc pentru obținerea unei reprezentări sau descrieri a formei obiectelor sau regiunilor (contururi, schelete, înfăşurători convexe).

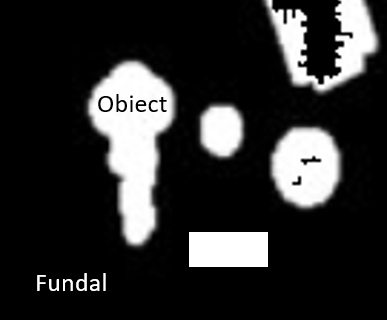
Principalele operații morfologice sunt dilatarea şi eroziunea[[2]](#footnote-2). Efectul acestora este de mărire / micșorare a obiectelor din imagine.

Transformările morfologice au nevoie de două argumente, primul este imaginea originală iar a doi-lea se numește element structurant sau nucleu care acționează în funcție de natura operației morfologice și de forma acestuia[[3]](#footnote-3).

**1.Notiunea de obiect și de fundal**

O imagine digitală este formată dintr-un fundal și din obiecte[[4]](#footnote-4). Prin operația de biarizare fiecare pixel este restrâns la o valoare de **0 sau 1**[[5]](#footnote-5), astfel obiectele devin mai ușor de identificat și analizat, identificarea obiectelor dintr-o imagine color sau cu niveluri de gri poate fi o sarcină foarte dificilă.

Metoda de analiză a obiectelor în acest capitol este una morfologică (din cuvântul grecesc *morphē*, adică înfățișare sau formă).



Modalitatile de procesare ale imaginilor cu operații morfologice in Python se poate realiza cu proceduri din biblioteca OpenCv sau biblioteca Skimage, pentru a o folosi pe cea din urmă instalați scikit-image. Skimage este o colecție de algoritmi implementați pentru prelucrarea imaginilor in Python, usor de acesat.

Toate procedurile vor fi importate în antet.

Exemplu:

**2.from** skimage.morphology **import** square, dilation,erosion, opening, closing,white\_tophat,black\_tophat, skeletonize,convex\_hull\_image  
**3. from** skimage.morphology **import** disk

Importarea imaginilor se poate face de pe un disk al PC (se adugă calea), din folder-ul curent (nu se mai adugă calea) sau imagini din biblioteca skimage

**from** skimage **import** data

horse = data.horse()

Mai multe imagini care pot acesate cu procedura data le gasiți accesând link-ul:

<https://scikit-image.org/docs/stable/api/skimage.data.html>

**2.Elementul structurant**

Transformările morfologice modifică structura obiectelor ca urmare a interacțiunii dintre elementul structurant și imaginea originală binarizată.

Elementul structurant este o multime geometrică, arbitrară, impusă și cunoscută. Forma elementului structurant determină proprietatile obiectelor din imagine[[6]](#footnote-6).

Forme ale elementuli structurant (din biblioteca skimage)

* Pătratic square

from skimage.morphology import (, rectangle, diamond, disk)

kernel = square(10)

array([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]], dtype=uint8)

Sau

from skimage.morphology import (square, rectangle, diamond, disk)

kernel = rectangle(3, 5)

array([[1, 1, 1, 1, 1],

[1, 1, 1, 1, 1],

[1, 1, 1, 1, 1]], dtype=uint8)

* Diamant (cu raza de 7)
* from skimage.morphology import (square, rectangle, diamond, disk)

kernel = diamond(7)

array([[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0],

[0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0],

[0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0],

[0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0],

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],

[0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0],

[0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0],

[0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0],

[0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]], dtype=uint8)

* Disc (cu raza de 5)
* from skimage.morphology import (square, rectangle, diamond, disk, octagon)

kernel = disk(5)

array([[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],

[0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0],

[0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0],

[0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0],

[0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0],

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],

[0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0],

[0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0],

[0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0],

[0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0],

[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]], dtype=uint8)

* Octogon (cu latimea de 9 unități și înălțimea de 5)
* from skimage.morphology import (square, rectangle, diamond, disk, octagon)

kernel = octagon(9, 5)

array([[0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0],

[0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0],

[0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0],

[0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0],

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],

[0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0],

[0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0],

[0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0],

[0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0]],

dtype=uint8)

* Cruce (cv.MORPH\_CROSS) (din libraria OpenCv)
* Elipsă (cv.MORPH\_ELLIPSE) (din libraria OpenCv)
* Dreptunghiulară ( cv.MORPH\_RECT) (din libraria OpenCv)
* Sau oricare altă formă definită de utilizator

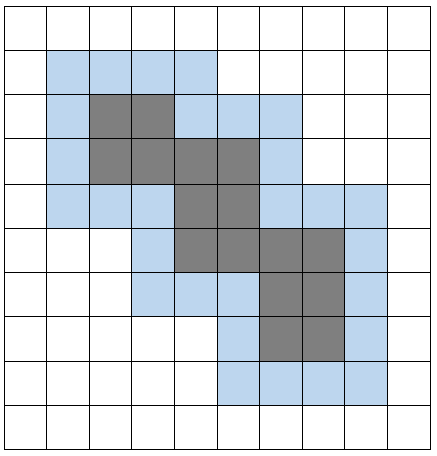
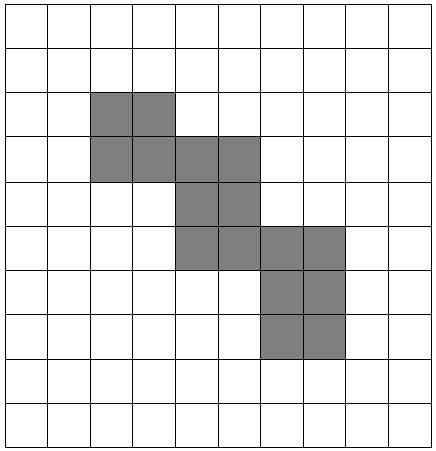
**3. Dilatarea**

Dacă A este o imagine binară și p un pixel, o translație a lui A cu p este o nouă imagine dată de

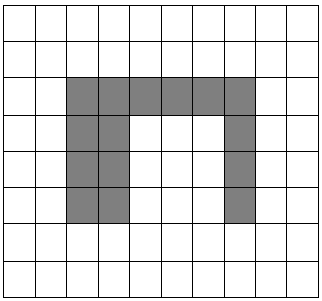
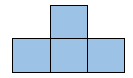
 (1)

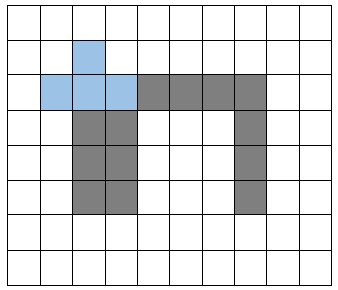
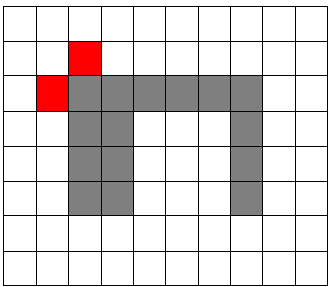
Dacă reprezintă mulțimea tuturor translațiilor dintr-o imagine binară, iar **mulțimea pixelilor translatați, atunci reuniunea tuturor translațiilor din A și B** **se numește dilatarea a lui A prin elementul structurant B[[7]](#footnote-7)**, aceasta este dată de:

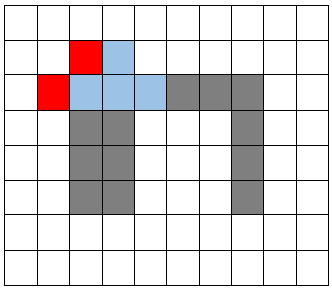
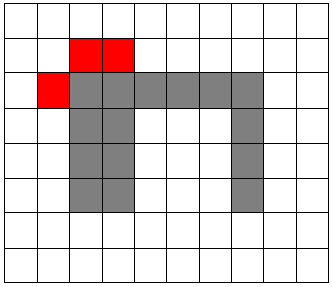


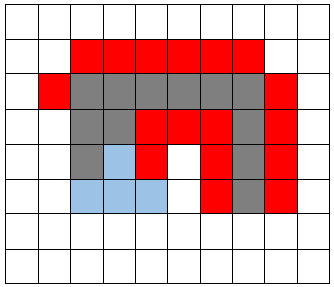
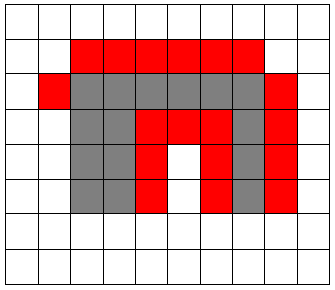


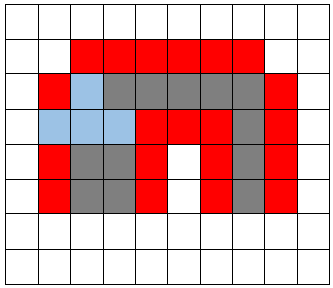
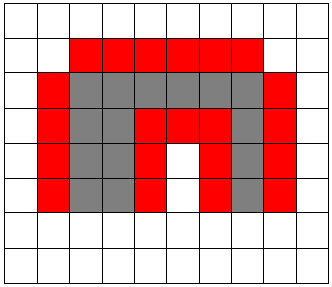
Dilatarea are proprietățile de asociativitate și comutativitate[[8]](#footnote-8). Astfel dacă avem mai multe operații succesive de dilatare nu este importantă ordinea în care se aplelează operațiile. Astfel formele complexe pot fi descompuse în forme mult mai simple care pot fi recombinate ca o secvență de dilatații.

**4. Erodarea (coroziunea)**

Erodarea este operația opusă dilatării. Eroziunea unei imagini binare A cu un element structurant B induce o imagine rezultantă formată din acei pixeli care după operația de translatare sunt eliminați respectandu-se regula:

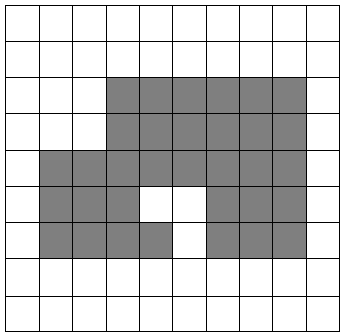
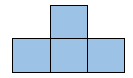


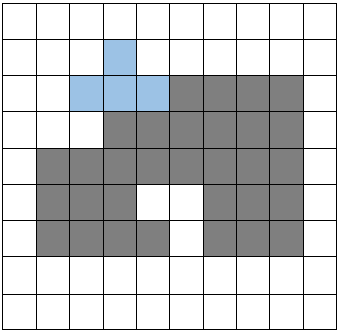
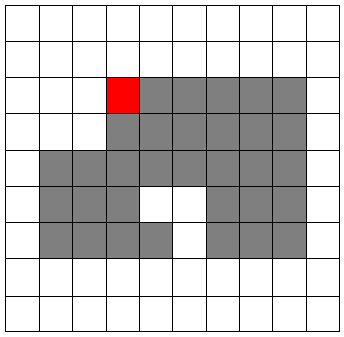
Adesea elementul structurant B are o formă regulată, acesta modifică obiectele din imaginea A în funcție de forma și dimensiunea sa.

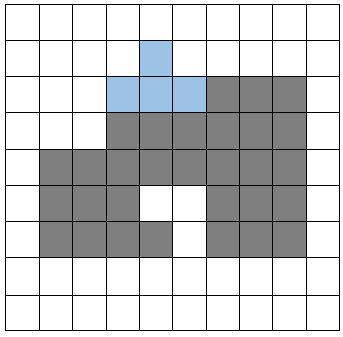
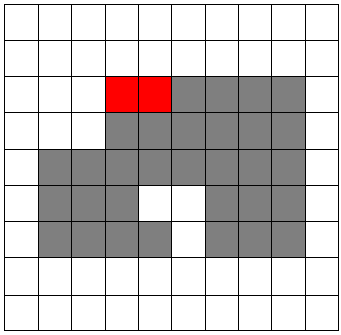
Eroziunea unei imagini prin elementul structurant joacă un rol foarte important în multe aplicații, unul fiind acela de **determinare a acelor zone din imagine unde elementul structurant este conținut de imagine**.[[9]](#footnote-9)

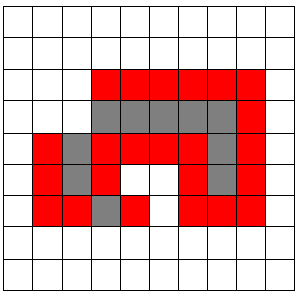
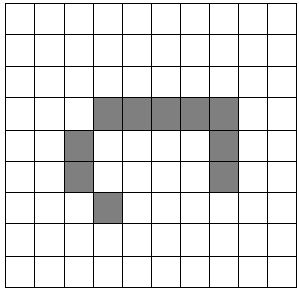
Figurile următoare ilustrează operația de eroziune cu un element structurant B, acesta din urmă având forma simbolului perpendicular. În figurile următoare sunt afișate inclusiv translațiile elementului structurant cu un pixel.

**Algoritm de eroziune a unui obiect**

Figurile ilustrează translații ale elementului structurant B cu un pixel. Pe parcursul operației de dilatare fiecare pixel din elementul structurant va fi prezent în imaginea finală, pe parcursul eroziunii, pixelul origine a elementului structurant va fi șters.

Pe parcursul operației de eroziune pixelul origine a elementului structurant va fi eliminat deoarce elementul structurant nu se află în întregine în interiorul obiectului.

Dilatarea și erodarea prezintă o natură duală, mai degrabă de natura geometrică decât logică, implicând complementul geometric la fel ca complementul logic. Complementul geometric pentru o imagine binară este numit reflecție. Refecția unei imagini binare B este o imagine binară B’ care este simetrică față de origine.



Dualitatea geometrică a dilatării și erodării este exprimată de următoarele relații:



și



Dualitatea geometrică versus dualitatea logică



și

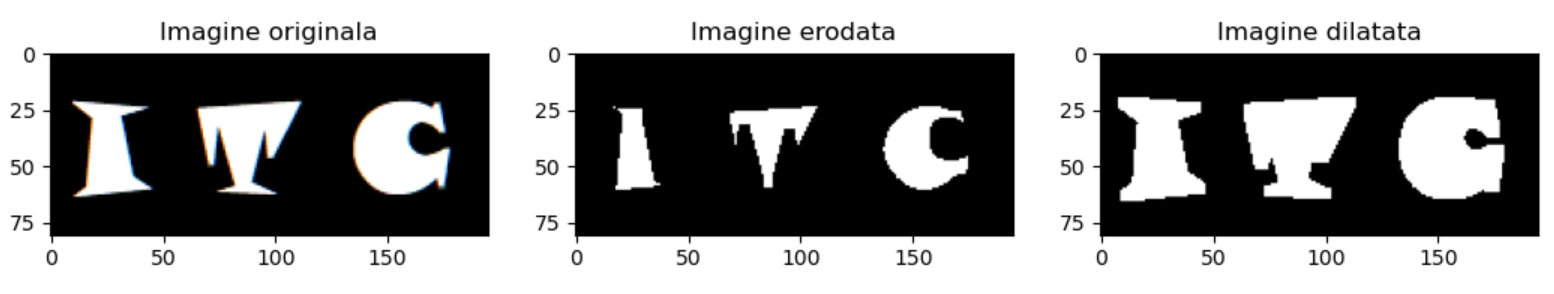


Demunite și legile De Morgan.

In OpenCv funcțiile folosite la erodarea/dilatarea unei imagini sunt cv2.erode() respectiv cv2.dilate().

**import** cv2  
**import** numpy **as** np  
**from** matplotlib **import** pyplot **as** plt  
path =**r'C:\Users\elev\Desktop\nou\_pr\efel.jpg'**img = cv2.imread(path)  
gri = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
(thresh1, I1) = cv2.threshold(gri,127,255,cv2.THRESH\_BINARY)  
kernel = np.ones((5,5),np.uint8)  
erosion = cv2.erode(I1,kernel,iterations = 1)  
dilation = cv2.dilate(I1,kernel,iterations = 1)  
plt.subplot(131), plt.imshow(img, **'gray'**),plt.title(**'Imagine originala'**)  
plt.subplot(132), plt.imshow(erosion,**'gray'**),plt.title(**'Imagine erodata'**)  
plt.subplot(133), plt.imshow(dilation,**'gray'**),plt.title(**'Imagine dilatata'**)  
plt.show()  
cv2.waitKey(1000)  
cv2.destroyAllWindows()

Unde kernel este o matrice de forma 



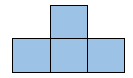
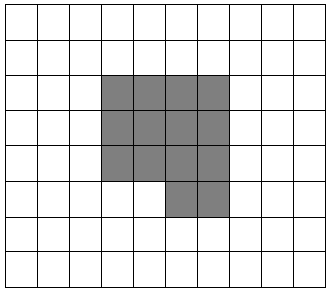
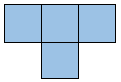
Sau

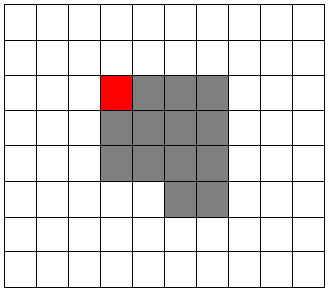
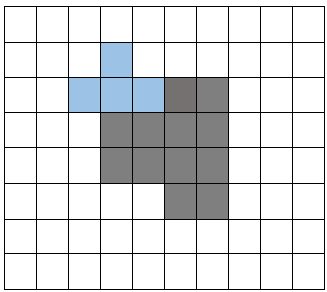
**import** cv2  
**import** numpy **as** np  
**from** matplotlib **import** pyplot **as** plt  
**from** skimage.morphology **import** (square, rectangle, diamond, disk)  
path =**r'C:\Users\elev\Desktop\nou\_pr\efel.jpg'**img = cv2.imread(path)  
gri = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
(thresh1, I1) = cv2.threshold(gri,127,255,cv2.THRESH\_BINARY)  
*#kernel = np.ones((5,5),np.uint8)*kernel = square(10)  
erosion = cv2.erode(I1,kernel,iterations = 1)  
dilation = cv2.dilate(I1,kernel,iterations = 1)  
plt.subplot(131), plt.imshow(img, **'gray'**),plt.title(**'Imagine originala'**)  
plt.subplot(132), plt.imshow(erosion,**'gray'**),plt.title(**'Imagine erodata'**)  
plt.subplot(133), plt.imshow(dilation,**'gray'**),plt.title(**'Imagine dilatata'**)  
plt.show()  
cv2.waitKey(1000)  
cv2.destroyAllWindows()

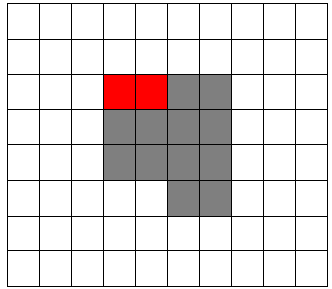
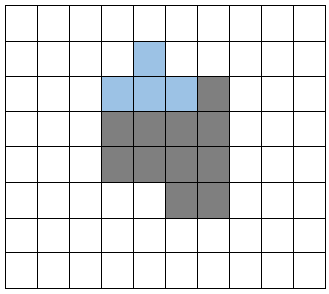
**5. Operațiile de deschidere și închidere**

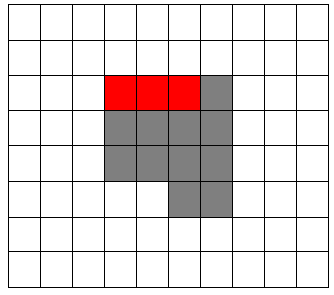
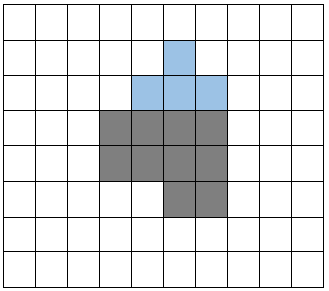
Operația de deschidere este formată dintr-o combinație dintre operație de erodare cu un element structurant B urmată de o operație de dilatare cu simetricul elementului structurant.[[10]](#footnote-10)

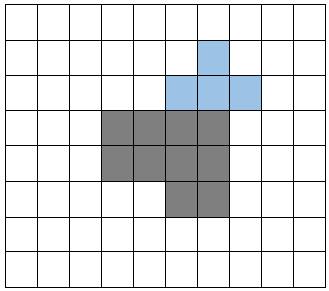
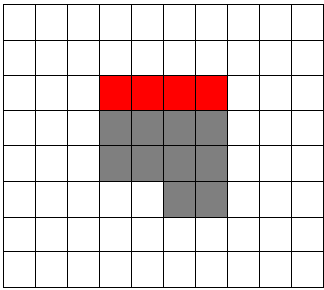


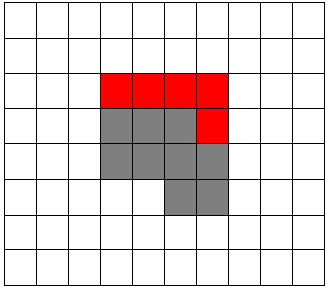
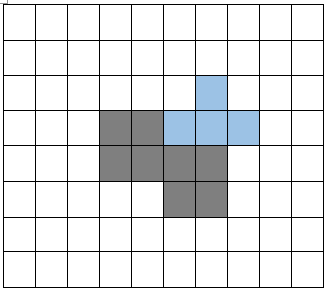
 

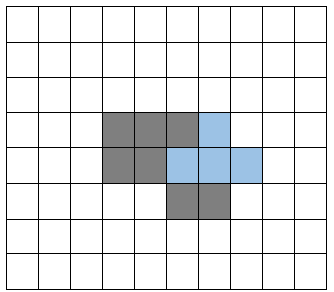
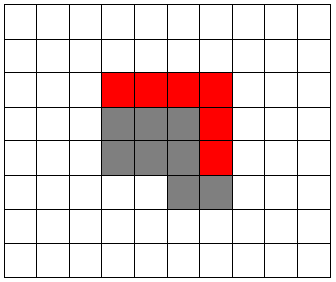


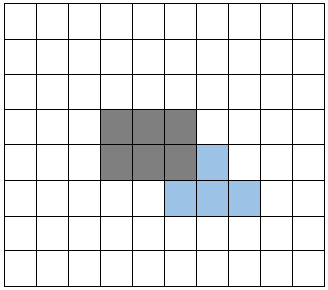
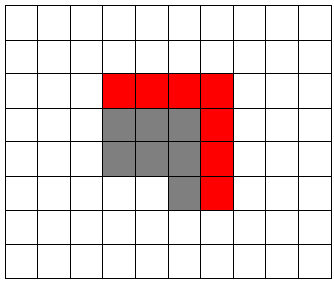


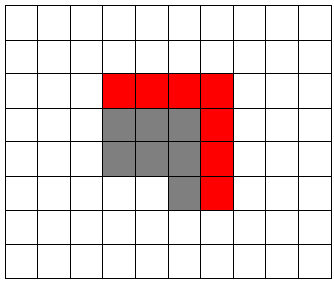
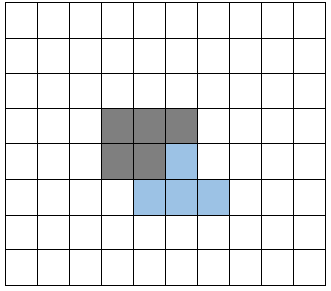


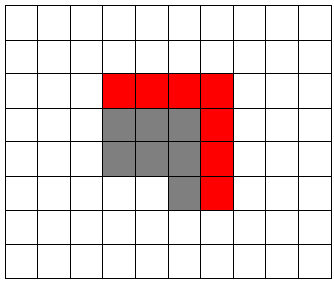
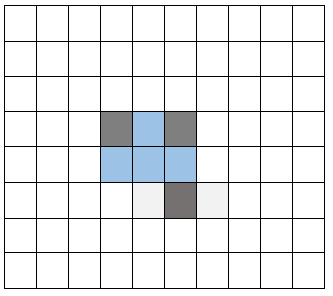
 

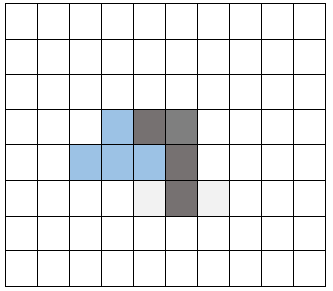
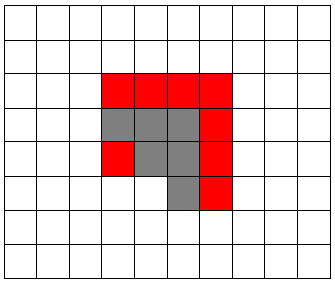


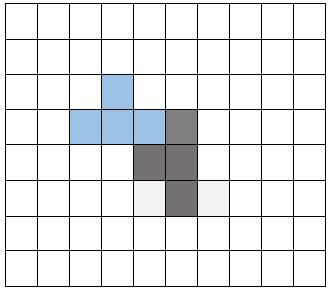
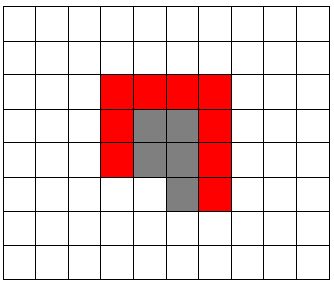
 

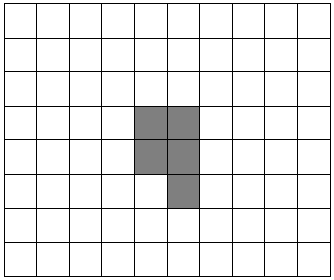
 

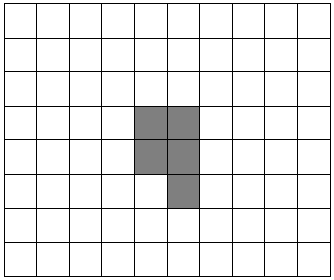
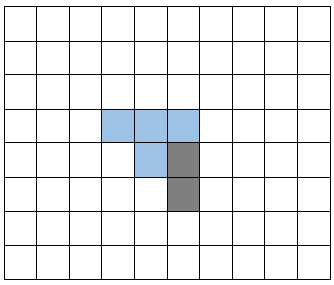


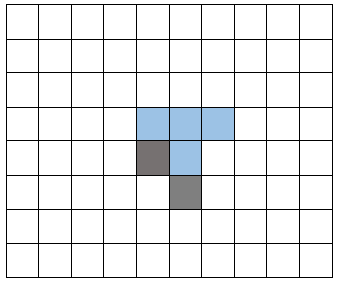
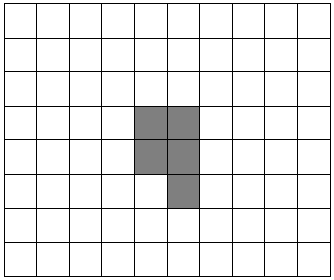


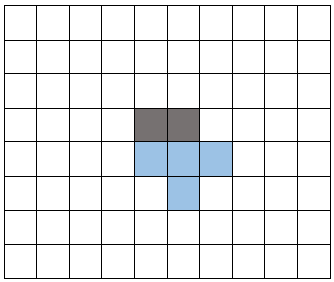
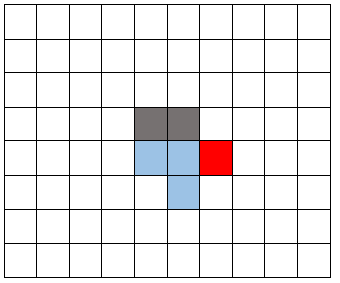
 

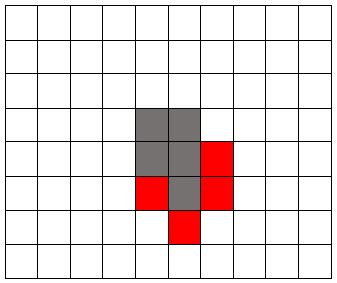
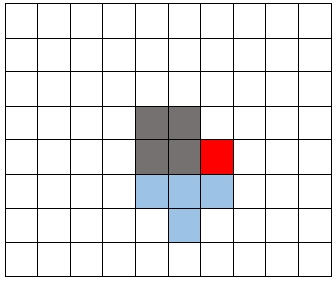
 

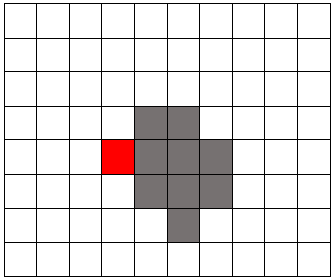
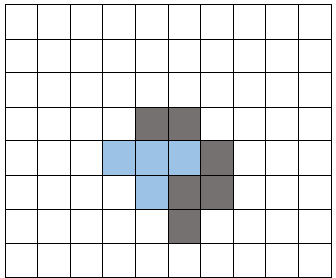


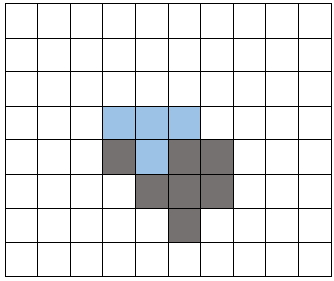


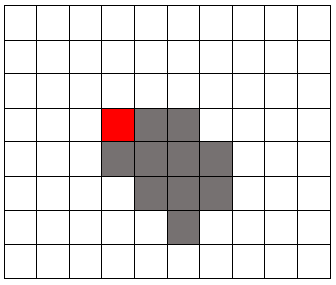
 

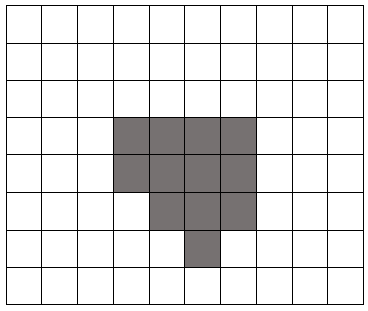
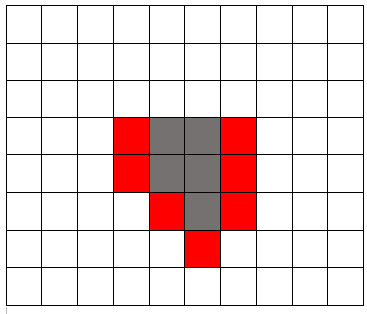
 











**Inchiderea morfolgică** este o operație care combină o dilatare combinată cu o erodare cu un element structurant simetric celui folosit în operația de dilatare[[11]](#footnote-11).

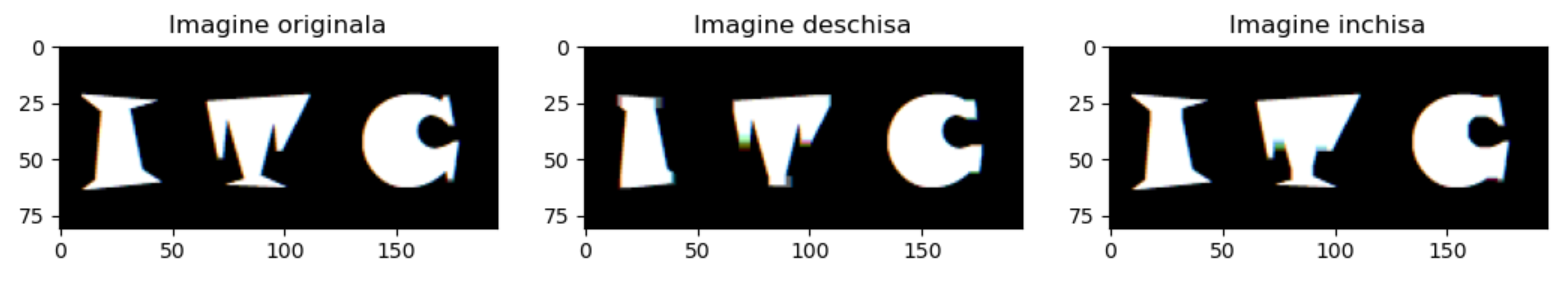


Pentru realizarea operațiilor de închidere/deschidere pe o imagine binară se apelează argumentele cv2.MORPH\_CLOSE și cv2.MORPH\_OPEN în funcția morphologyEx(). Elementul structurant este dat de kernel si are forma sus amintită. Inchiderea și deschiderea se realizează prin apelul complet cv2.morphologyEx(img,cv2.MORPH\_OPEN, kernel) și  
cv2.morphologyEx(img, cv2.MORPH\_CLOSE, kernel)

**import** cv2  
**import** numpy **as** np  
**from** matplotlib **import** pyplot **as** plt  
path =**r'C:\Users\elev\Desktop\nou\_pr\efel.jpg'**img = cv2.imread(path)  
gri = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
(thresh1, I1) = cv2.threshold(gri,127,255,cv2.THRESH\_BINARY)  
kernel = np.ones((5,5),np.uint8)  
opening = cv2.morphologyEx(I1, cv2.MORPH\_OPEN, kernel)  
closing = cv2.morphologyEx(I1, cv2.MORPH\_CLOSE, kernel)  
dilation = cv2.dilate(I1,kernel,iterations = 1)  
plt.subplot(131), plt.imshow(img, **'gray'**),plt.title(**'Imagine originala'**)  
plt.subplot(132), plt.imshow(opening,**'gray'**),plt.title(**'Imagine deschisa'**)  
plt.subplot(133), plt.imshow(closing,**'gray'**),plt.title(**'Imagine inchisa'**)  
plt.show()  
cv2.waitKey(1000)  
cv2.destroyAllWindows()

sau

**import** cv2  
**import** numpy **as** np  
**from** matplotlib **import** pyplot **as** plt  
**from** skimage.morphology **import** (square, rectangle, diamond, disk)  
path =**r'C:\Users\elev\Desktop\nou\_pr\monede.png'**img = cv2.imread(path)  
gri = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
(thresh1, I1) = cv2.threshold(gri,127,255,cv2.THRESH\_BINARY)  
kernel = square(12)  
opening = cv2.morphologyEx(I1, cv2.MORPH\_OPEN, kernel)  
closing = cv2.morphologyEx(I1, cv2.MORPH\_CLOSE, kernel)  
dilation = cv2.dilate(I1,kernel,iterations = 1)  
plt.subplot(131), plt.imshow(img, **'gray'**),plt.title(**'Imagine originala'**)  
plt.subplot(132), plt.imshow(opening,**'gray'**),plt.title(**'Imagine deschisa'**)  
plt.subplot(133), plt.imshow(closing,**'gray'**),plt.title(**'Imagine inchisa'**)  
plt.show()  
cv2.waitKey(1000)  
cv2.destroyAllWindows()



**6.Gradientul morfologic, conturul interior și exterior**

Gradientul morfologic este diferența dintre o dilatare și o erodare cu același element structurant.[[12]](#footnote-12)



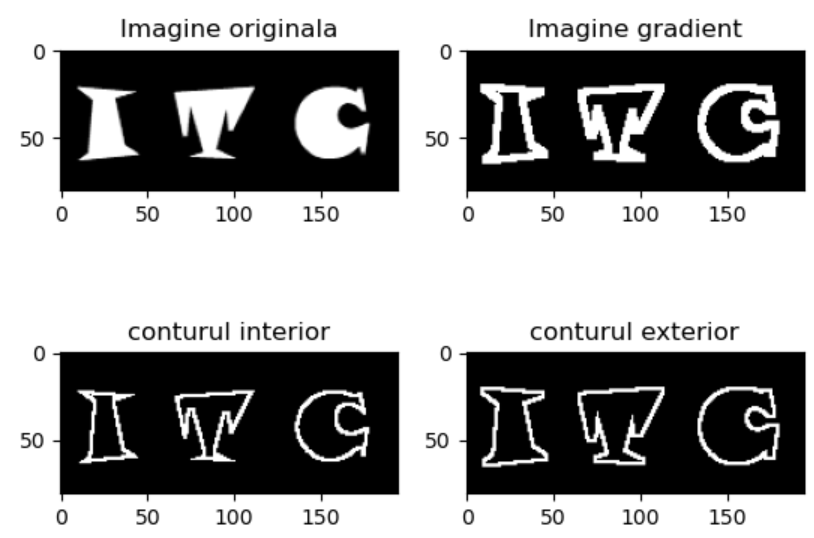
Conturul interior este diferența dintre imagine și erodarea imaginii[[13]](#footnote-13)



Conturul exterior este diferența dintre o dilatare a imaginii și imaginea[[14]](#footnote-14)



**import** cv2  
**import** numpy **as** np  
**from** matplotlib **import** pyplot **as** plt  
path =**r'C:\Users\elev\Desktop\nou\_pr\efel.jpg'**img = cv2.imread(path)  
gri = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
(thresh1, I1) = cv2.threshold(gri,127,255,cv2.THRESH\_BINARY)  
kernel = np.ones((5,5),np.uint8)  
erosion = cv2.erode(I1,kernel,iterations = 1)  
dilation = cv2.dilate(I1,kernel,iterations = 1)  
gradient=dilation-erosion  
*#sau  
#gradient = cv.morphologyEx(img, cv.MORPH\_GRADIENT, kernel)*contur\_int=I1-erosion  
contru\_ext=dilation-I1  
plt.subplot(221), plt.imshow(gri, **'gray'**),plt.title(**'Imagine originala'**)  
plt.subplot(222), plt.imshow(gradient,**'gray'**),plt.title(**'Imagine gradient'**)  
plt.subplot(223), plt.imshow(contur\_int,**'gray'**),plt.title(**'conturul interior'**)  
plt.subplot(224), plt.imshow(contru\_ext,**'gray'**),plt.title(**'conturul exterior'**)  
plt.show()  
cv2.waitKey(1000)  
cv2.destroyAllWindows()



### 7. Transformări morfologice

### 7.1 Transformarea  Black Hat

Este diferența dintre închiderea unei imagini și imaginea binară.[[15]](#footnote-15)



### 7.2 Transformarea   Top-Hat

Este diferența dintre o imagine binara și deschidere ei.[[16]](#footnote-16)



### 7.3 Transformarea   Hit-or-Miss

Transformata Hit-or-Miss este utilă atunci când se dorește a se a găsi modele în imagini binare[[17]](#footnote-17). În particular, ea găsește acei pixeli al căror vecinătate se potrivește cu forma unui prim element structurant B1, fără a se potrivi în același timp cu forma celui de-al doilea element structurant B2. Matematic, operația aplicată unei imagini A poate fi exprimată după cum urmează:



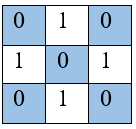
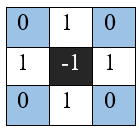
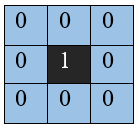
Structurat transformata Hit-or-Miss acționează astfel asupra unei imagini.

I. Erodează imaginea a cu elementul structurant B1

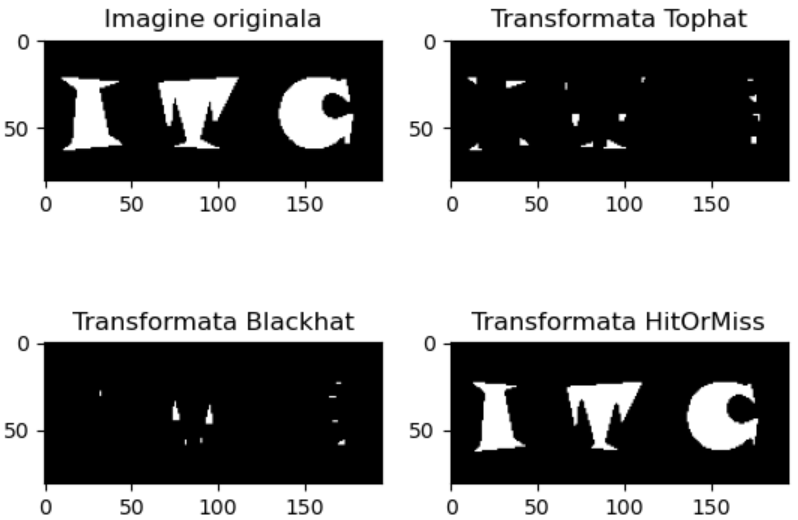
II. Erodează complementul imaginii A cu elementul structurant B2

III. Aplică operatorul AND între imaginile output de la pasul I și II.

Elementele structurante B1 și B2 poate avea următoarele forme iar combinate sa dea elementul B

**import** cv2  
**import** numpy **as** np  
**from** matplotlib **import** pyplot **as** plt  
path =**r'C:\Users\elev\Desktop\nou\_pr\efel.jpg'**img = cv2.imread(path)  
gri = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
(thresh1, I1) = cv2.threshold(gri,127,255,cv2.THRESH\_BINARY)  
kernel = np.ones((5,5),np.uint8)  
kernel1=np.array(([[0,1,0],  
 [1,-1,1],  
 [0,1,0]]), dtype=**"uint8"**)  
tophat = cv2.morphologyEx(I1, cv2.MORPH\_TOPHAT, kernel)  
blackhat = cv2.morphologyEx(I1, cv2.MORPH\_BLACKHAT, kernel)  
hitmiss = cv2.morphologyEx(I1, cv2.MORPH\_HITMISS, kernel1)  
plt.subplot(221), plt.imshow(I1, **'gray'**),plt.title(**'Imagine originala'**)  
plt.subplot(222), plt.imshow(tophat,**'gray'**),plt.title(**'Transformata Tophat'**)  
plt.subplot(223), plt.imshow(blackhat,**'gray'**),plt.title(**'Transformata Blackhat'**)  
plt.subplot(224), plt.imshow(hitmiss,**'gray'**),plt.title(**'Transformata HitOrMiss'**)  
plt.show()  
cv2.waitKey(1000)  
cv2.destroyAllWindows()



**8. Scheletonizarea**

Scheletonizarea este un proces de reducere a regiunilor din prim-planul unei imagi binare la un schelet care păstrează în mare măsură extinderea și conectivitatea regiunilor originale, renunțând în același timp la majoritatea pixelilor originali din prim-planul imaginii.[[18]](#footnote-18)

Scheletonizarea este utilizată pe scară largă în reprezentarea, găsirea, manipularea, potrivirea, înregistrarea, urmărirea, recunoașterea și compresia obiectelor[[19]](#footnote-19), etc.

1. Se crează un schelet gol.

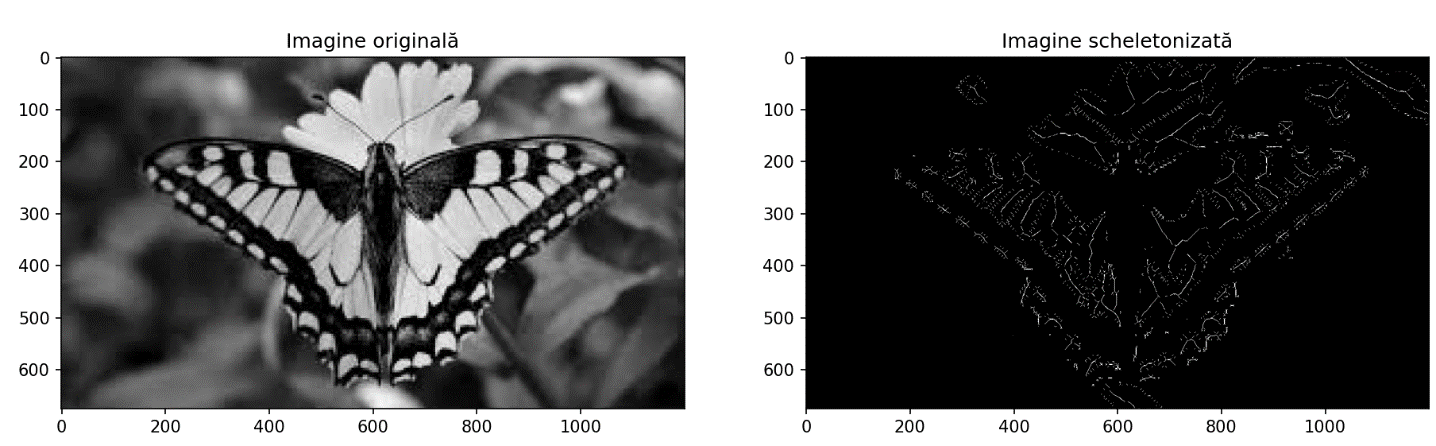
2. Se aplică imaginii originale operația morfologică de deschidere.

3. Se scade imaginea originală din deschidere obținându-se imaginea *temp;*

4. Se erodează imaginea originală și se rafinează scheletul prin operații de reuniune dintre scheletul curent si imaginea *temp*.

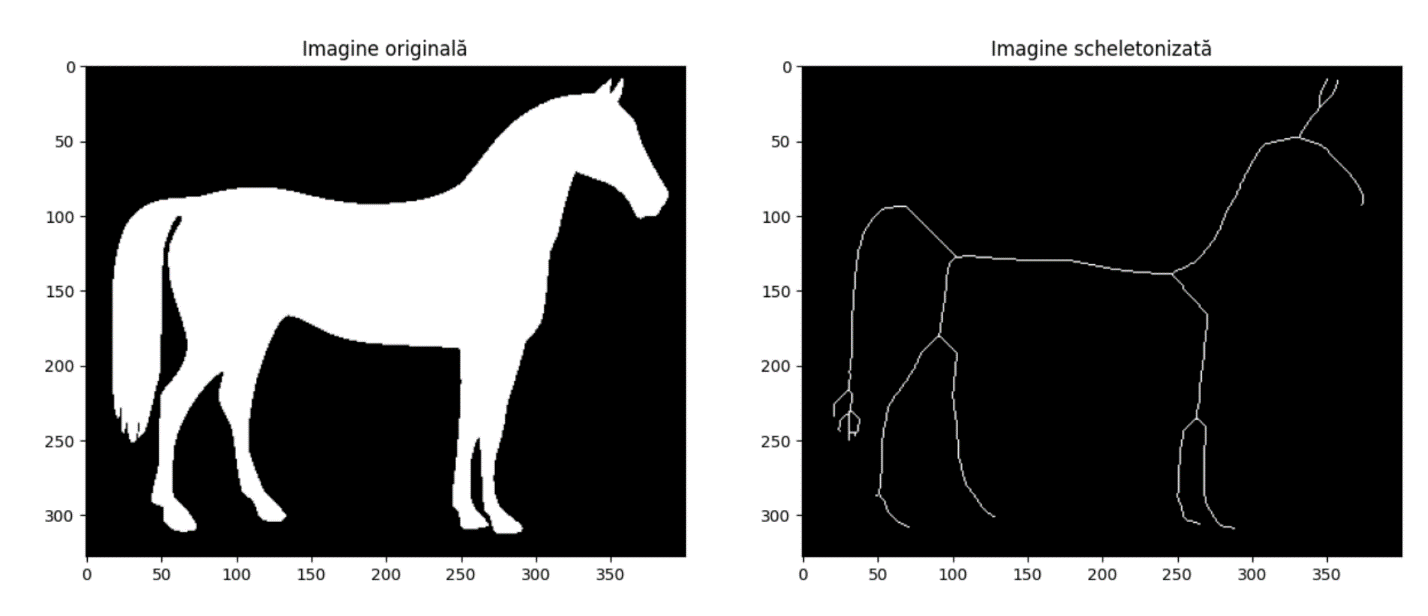
5. Se repetă pașii 2-4 până când imaginea originală este complet erodată.

**import** cv2  
**import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
  
*# citim si transformam o imagine în niveluri de gri*imagine1 = cv2.imread(**'efel.jpg'**, 0)  
  
*# binarizam imaginea*ret,imagine = cv2.threshold(imagine1, 127, 255, 0)  
  
*# pas 1: cream un schelet nul pe care se va construi noul schelet al imaginii*size = np.size(imagine)  
skel = np.zeros(imagine.shape, np.uint8)  
  
*# Alegem o froma a unui element structural*element = cv2.getStructuringElement(cv2.MORPH\_CROSS, (3,3))  
  
*# Repetam pasii 2,3,4***while True**:  
 *#Step 2: Deschidem imaginea* open = cv2.morphologyEx(imagine, cv2.MORPH\_OPEN, element)  
 *#Step 3: Realizam diferența dintre imaginea originală  
 # si imaginea suspua operației morfologice de deschidere* temp = cv2.subtract(imagine, open)  
 *#Step 4: Erodam imaginea cu un element structural* eroded = cv2.erode(imagine, element)  
 skel = cv2.bitwise\_or(skel,temp)  
 imagine = eroded.copy()  
 *# Step 5: If there are no white pixels left ie.. the image has been completely eroded, quit the loop* **if** cv2.countNonZero(imagine)==0:  
 **break***# Afișam imaginea scheletizată*plt.subplot(121), plt.imshow(imagine1,**'gray'**),plt.title(**'Imagine originală'**)  
plt.subplot(122), plt.imshow(skel,**'gray'**),plt.title(**'Imagine scheletonizată'**)  
cv2.waitKey(0)  
plt.tight\_layout()  
plt.show()



**Sau**

**import** matplotlib.pyplot **as** plt  
**from** skimage **import** data  
**from** skimage.morphology **import** skeletonize  
**from** skimage.util **import** invert  
**import** cv2  
*# Invert the horse image*image = invert(data.horse())  
*# perform skeletonization*skeleton = skeletonize(image)  
*# display results*plt.subplot(121), plt.imshow(image,**'gray'**),plt.title(**'Imagine originală'**)  
plt.subplot(122), plt.imshow(skeleton,**'gray'**),plt.title(**'Imagine scheletonizată'**)  
cv2.waitKey(0)  
plt.tight\_layout()  
plt.show()

****

**9. Procesarea imaginilor cu Skimage**

Skimage este o colecție de algoritmi implementați pentru prelucrarea imaginilor in Python. Pentru a folosi biblioteca Skimage instalați scikit-image.

Operațiile morfologice sunt același ca in OpenCv se modifică doar aplearea lor.

**1. Instalati scikit-image**

**import** numpy **as** np  
**from** matplotlib **import** pyplot **as** plt  
**import** cv2  
**from** skimage **import** data  
**from** skimage.morphology **import** square, dilation,erosion, opening, closing,white\_tophat,black\_tophat, skeletonize,convex\_hull\_image  
**from** skimage.morphology **import** disk  
img = cv2.imread(**'monede.png'**)  
gri = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
(thresh1, an) = cv2.threshold(gri,127,255,cv2.THRESH\_BINARY)  
dilate=dilation(an, square(3))  
es2 = disk(6)  
eroded = erosion(an, es2)  
open=opening(an, es2)  
close=closing(an,es2)  
w\_tophat = white\_tophat(an, es2)  
b\_tophat = black\_tophat(an, es2)  
*#citim imaginea din data*horse = data.horse()  
sk = skeletonize(horse == 0)  
*#invelitoarea convexa*hull1 = convex\_hull\_image(horse == 0)  
plt.subplot(431), plt.imshow(img,**'gray'**),plt.title(**'imagine originala'**)  
plt.subplot(432), plt.imshow(gri,**'gray'**),plt.title(**'imagine cu niveluri de gri'**)  
plt.subplot(433), plt.imshow(dilate,**'gray'**),plt.title(**'imagine eu obiecte dilatate'**)  
plt.subplot(434), plt.imshow(eroded,**'gray'**),plt.title(**'imagine eu obiecte erodate'**)  
plt.subplot(435), plt.imshow(open,**'gray'**),plt.title(**'imagine eu obiecte deschise'**)  
plt.subplot(436), plt.imshow(close,**'gray'**),plt.title(**'imagine eu obiecte inschise'**)  
plt.subplot(437), plt.imshow(w\_tophat,**'gray'**),plt.title(**'palarie alba'**)  
plt.subplot(438), plt.imshow(b\_tophat,**'gray'**),plt.title(**'palarie neagra'**)  
plt.subplot(439), plt.imshow(sk,**'gray'**),plt.title(**'imagine scheletonizata'**)  
plt.subplot(4,3,10), plt.imshow(horse,**'gray'**),plt.title(**'imagine'**)  
plt.subplot(4,3,11), plt.imshow(hull1,**'gray'**),plt.title(**'invelitoare convexa'**)  
plt.show()  
cv2.waitKey()  
cv2.destroyAllWindows()

**10. Operații morfologice pentru niveluri de gri**

Operatiile morfologice pot fi extinse şi pentru cazul imaginilor cu niveluri de gri, totuşi această extensie nu este “evidentă/naturală”:

Ipoteze: - imaginea contine obiecte cu niveluri de gri distincte față de fundalul imaginii, de asemenea reprezentat cu niveluri de gri, obiectele şi fundalul sunt considerate a fi relativ uniforme spațial,

Soluție: imaginea cu niveluri de gri este binarizată şi apoi sunt aplicate metodele de morfologie binară, de ce să nu folosim această abordare ?

Soluția matematică: fie A(x,y) imaginea cu niveluri de gri inițială definită pe domeniul DA şi B(x,y) elementul structurant definit pe domeniul DB

Dilatarea pe niveluri de gri

cu alte cuvinte, dilatarea imaginii A cu elementul B în punctul curent (s,t) este dată de valoarea maximă a sumei dintre valorile pixelilor imaginii şi valorile corespunzătoare din elementul structurant.

dilatare niveluri de gri ~ valoare maximă din vecinătatea elementului structurant considerat, = dacă B=0:

- dacă valorile elementului structurant sunt pozitive, atunci imaginea devine mai luminoasă,

- se dilată obiectele, unde un obiect este o zonă din imagine mărginită de valori mai întunecate (ex. de fundal),

Erodarea pe niveluri de gri

Cu alte cuvinte, erodarea imaginii A cu elementul B în punctul curent (s,t) este dată de valoarea minimă a diferențelor dintre valorile pixelilor imaginii şi valorile corespunzătoare din elementul structurant.

erodare niveluri de gri ~ valoare minimă din vecinătatea elementului structurant considerat, = dacă B=0:

Deschiderea pe niveluri de gri

Efect general: sunt eliminate obiectele luminoase mai mici decât elementul structurant.

Evident acceasi definitie ca in cazul binar

Caz linita , B=0

Închiderea pe niveluri de gri



-caz limita, B=0: 

**Efect general**: sunt eliminate obiectele întunecate mai mici decât elementul structurant.

**Netezirea morfologică**

**Efect general**: sunt înlăturate/atenuate atât obiectele închise cât şi obiectele deschise mai mici decât elementul structurant (ex. zgomot).

- deschidere + închidere: (A°B) • B

- închidere + deschidere: (A• B)°B

Gradient morfologic

-din imaginea dilatata se scade erodata

-caz limita (B=0)

Transformarea Top-Hat

White Top-Hat

Din imaginea inițială A se extrage imaginea deschisă cu elementul structurant B

Black Top-Hat : (A• B)-A

Din imaginea închisă cu elementrul structurant B se extrage imaginea inițială

Efect global: eliminarea tranzițiilor lente şi astfel creşterea contrastului anumitor regiunii ale imaginii.

**Exemplu: eroziune, dilatare, deschidere, închidere, scheletonizare, transformări Top Hat**

**import** numpy **as** np  
**from** matplotlib **import** pyplot **as** plt  
**import** cv2  
**from** skimage **import** data  
**from** skimage.morphology **import** square, dilation,erosion, opening, closing,white\_tophat,black\_tophat, skeletonize,convex\_hull\_image  
**from** skimage.morphology **import** disk  
img = cv2.imread(**'monede.png'**)  
an = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
*#(thresh1, an) = cv2.threshold(gri,127,255,cv2.THRESH\_BINARY)*dilate=dilation(an, square(3))  
es2 = disk(6)  
eroded = erosion(an, es2)  
open=opening(an, es2)  
close=closing(an,es2)  
w\_tophat = white\_tophat(an, es2)  
b\_tophat = black\_tophat(an, es2)  
*#citim imaginea din data*horse = data.horse()  
sk = skeletonize(horse == 0)  
plt.subplot(431), plt.imshow(img,**'gray'**),plt.title(**'imagine originala'**)  
*#plt.subplot(432), plt.imshow(gri,'gray'),plt.title('imagine cu niveluri de gri')*plt.subplot(432), plt.imshow(dilate,**'gray'**),plt.title(**'imagine eu obiecte dilatate'**)  
plt.subplot(433), plt.imshow(eroded,**'gray'**),plt.title(**'imagine eu obiecte erodate'**)  
plt.subplot(434), plt.imshow(open,**'gray'**),plt.title(**'imagine eu obiecte deschise'**)  
plt.subplot(435), plt.imshow(close,**'gray'**),plt.title(**'imagine eu obiecte inschise'**)  
plt.subplot(437), plt.imshow(w\_tophat,**'gray'**),plt.title(**'palarie alba'**)  
plt.subplot(438), plt.imshow(b\_tophat,**'gray'**),plt.title(**'palarie neagra'**)  
plt.subplot(439), plt.imshow(sk,**'gray'**),plt.title(**'imagine scheletonizata'**)  
plt.subplot(4,3,10), plt.imshow(horse,**'gray'**),plt.title(**'imagine'**)  
*#plt.subplot(4,3,11), plt.imshow(hull1,'gray'),plt.title('invelitoare convexa')*plt.show()  
cv2.waitKey()  
cv2.destroyAllWindows()

**Exemplu: Netezire, gradient**

**from** matplotlib **import** pyplot **as** plt  
**import** cv2  
**from** skimage **import** data  
**from** skimage.morphology **import** square, dilation,erosion, opening, closing,white\_tophat,black\_tophat, skeletonize,convex\_hull\_image  
**from** skimage.morphology **import** disk  
img = cv2.imread(**'monede.png'**)  
an = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
*#(thresh1, an) = cv2.threshold(gri,127,255,cv2.THRESH\_BINARY)*dilate=dilation(an, square(3))  
es2 = disk(6)  
eroded = erosion(an, es2)  
open=opening(an, es2)  
close=closing(an,es2)  
gradient=open-close  
netezire1=open+close  
netezire2=close+open  
*#plt.subplot(432), plt.imshow(gri,'gray'),plt.title('imagine cu niveluri de gri')*plt.subplot(221), plt.imshow(an,**'gray'**),plt.title(**'imagine originala'**)  
plt.subplot(222), plt.imshow(gradient,**'gray'**),plt.title(**'gradient imagini gri'**)  
plt.subplot(223), plt.imshow(netezire1,**'gray'**),plt.title(**'netezire'**)  
plt.subplot(224), plt.imshow(netezire2,**'gray'**),plt.title(**'netezire'**)  
*#plt.subplot(4,3,11), plt.imshow(hull1,'gray'),plt.title('invelitoare convexa')*plt.show()  
cv2.waitKey()  
cv2.destroyAllWindows()

**Webografie**

1. <https://medium.com/analytics-vidhya/skeletonization-in-python-using-opencv-b7fa16867331>
2. <https://scikit-image.org/docs/dev/auto_examples/edges/plot_skeleton.html>
3. <https://www.cse.usf.edu/~r1k/MachineVisionBook/MachineVision.files/MachineVision_Chapter2.pdf>
4. <http://users.utcluj.ro/~igiosan/Resources/PI/L7/PI-L7r.pdf>
5. <https://mctr.mec.upt.ro/wp-content/uploads/2018/12/PrIII_imagini.pdf>
6. <https://www.cse.usf.edu/~r1k/MachineVisionBook/MachineVision.files/MachineVision_Chapter3.pdf>
7. <https://www.cse.usf.edu/~r1k/MachineVisionBook/MachineVision.files/MachineVision_Chapter4.pdf>
8. <https://www.cse.usf.edu/~r1k/MachineVisionBook/MachineVision.files/MachineVision_Chapter5.pdf>
9. <https://www.cse.usf.edu/~r1k/MachineVisionBook/MachineVision.files/MachineVision_Chapter6.pdf>
10. <https://www.cse.usf.edu/~r1k/MachineVisionBook/MachineVision.files/MachineVision_Chapter7.pdf>
11. <https://medium.com/analytics-vidhya/skeletonization-in-python-using-opencv-b7fa16867331>
12. <http://users.utcluj.ro/~igiosan/teaching_pi.html>
13. <https://pythonmana.com/2022/140/202205202055225092.html>
14. <https://code.adonline.id.au/structuring-elements-in-python/#rectangle>
15. [TAPAI\_BIonescu\_M7\_morfologie.pdf](file:///D:/2020/~cursuri/~curs_PI_Python/B_ionescu/TAPAI_BIonescu_M7_morfologie.pdf)

1. Pe ce tipuri de imagini se aplica metodele morfologiei matematice? [↑](#footnote-ref-1)
2. Ca sunt principalele operații morfologice? [↑](#footnote-ref-2)
3. Care sunt argumentele unei operatii morfologice? [↑](#footnote-ref-3)
4. Din ce tipuri de elemente este formată o imagine digitală? [↑](#footnote-ref-4)
5. Ce valori au pixelii în urma operației de binarizare? [↑](#footnote-ref-5)
6. Definiți elementul structural. [↑](#footnote-ref-6)
7. Definiți operația de dilatare. [↑](#footnote-ref-7)
8. Enumerați proprietățile operației de dilatare. [↑](#footnote-ref-8)
9. Rolul operației de eroziune [↑](#footnote-ref-9)
10. Din ce operații este formată operația de deschidere? [↑](#footnote-ref-10)
11. Din ce operații este formată operația de închidere? [↑](#footnote-ref-11)
12. Definiți gradientul morfologic. [↑](#footnote-ref-12)
13. Definiți conturul interior [↑](#footnote-ref-13)
14. Definiți conturul Exterior [↑](#footnote-ref-14)
15. Definiți transformarea Black Hat [↑](#footnote-ref-15)
16. Definiți transformarea Top- Hat [↑](#footnote-ref-16)
17. Care este scopl folosirii Definiți transformării Hit-or-Miss [↑](#footnote-ref-17)
18. Definiți scheletonizarea [↑](#footnote-ref-18)
19. Operațiile de urmărire, recunoaștere pot fi realizate prin procesul de scheletonizare? [↑](#footnote-ref-19)