Operatii morfologice in Python

November 16, 2020

Catalin Stoean

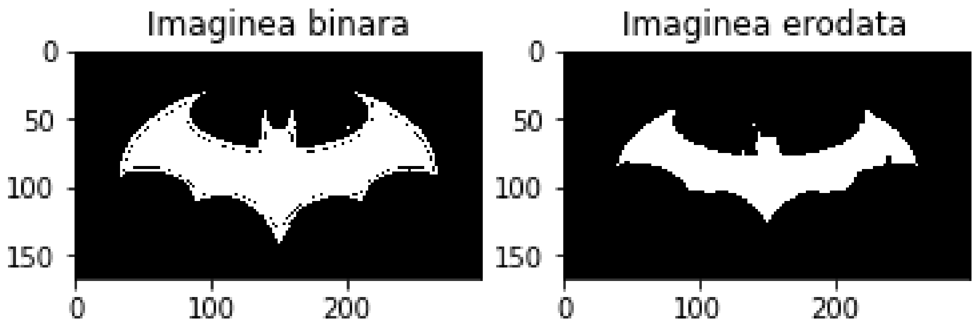
catalin.stoean@inf.ucv.ro

http://inf.ucv.ro/~cstoean

1 Erodarea

[146]:
```python
import cv2
import numpy as np
from matplotlib import pyplot as plt

elemSize = 5
img = cv2.imread('D:/batman.jpg',0)
ret,imgThresh = cv2.threshold(img, 20, 255, cv2.THRESH_BINARY)
kernel = np.ones((elemSize, elemSize), np.uint8)
erosion = cv2.erode(imgThresh, kernel, iterations = 1)

fig = plt.figure()
ax1 = fig.add_subplot(121)
ax1.set_title('Imaginea binara')
ax1.imshow(imgThresh, cmap='gray')
ax2 = fig.add_subplot(122)
ax2.set_title('Imaginea erodata')
ax2.imshow(erosion, cmap='gray')
```

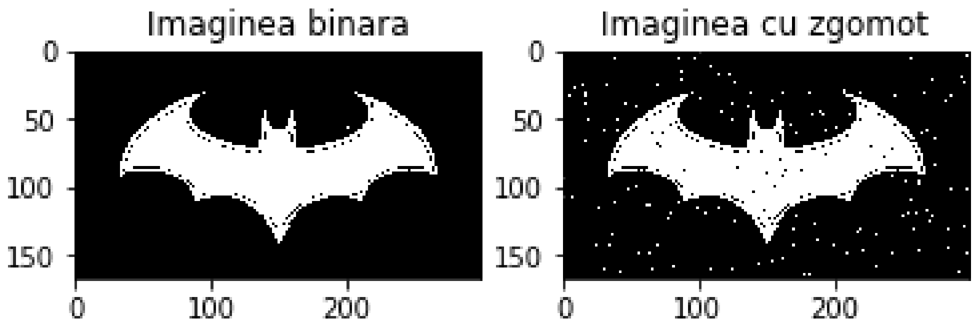
[146]:
```
<matplotlib.image.AxesImage at 0x1b98660e948>
```



2 Adaugam pixeli albi si negri la imaginea binara

[147]:
```python
import random

def zgomot(im, n):
    h, w = im.shape
    for k in range(n):
        i = random.randint(0, h - 1)
        j = random.randint(0, w - 1)
        if len(im.shape) == 2:
            im[i, j] = 255
    for k in range(n):
        i = random.randint(0, h - 1)
        j = random.randint(0, w - 1)
        if len(im.shape) == 2:
            im[i, j] = 0
    return
imZgomot = imgThresh.copy()
zgomot(imZgomot, 500)

fig = plt.figure()
ax1 = fig.add_subplot(121)
ax1.set_title('Imaginea binara')
ax1.imshow(imgThresh, cmap='gray')
ax2 = fig.add_subplot(122)
ax2.set_title('Imaginea cu zgomot')
ax2.imshow(imZgomot, cmap='gray')
```

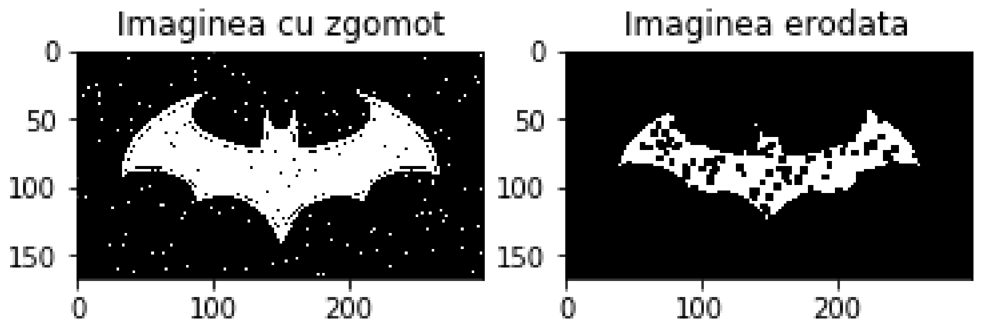
[147]:
```
<matplotlib.image.AxesImage at 0x1b9866cd588>
```



3 Erodarea pe imaginea cu zgomot

[148]:
```
erosion = cv2.erode(imZgomot, kernel, iterations = 1)

fig = plt.figure()
ax1 = fig.add_subplot(121)
ax1.set_title('Imaginea cu zgomot')
ax1.imshow(imZgomot, cmap='gray')
ax2 = fig.add_subplot(122)
ax2.set_title('Imaginea erodata')
ax2.imshow(erosion, cmap='gray')
```

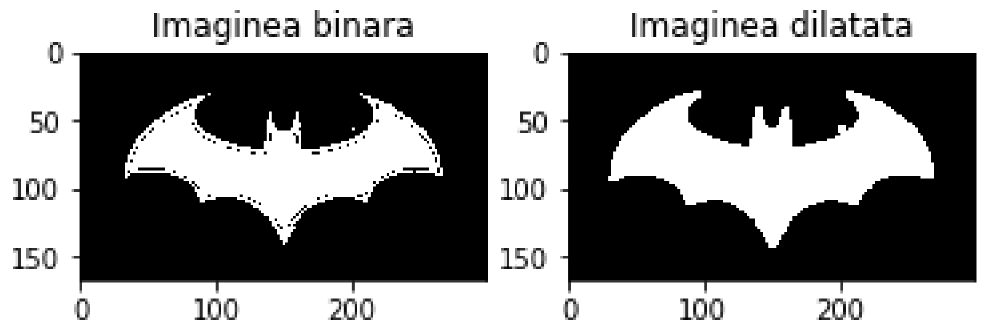
[148]:
```
<matplotlib.image.AxesImage at 0x1b98677fc88>
```



4 Dilatarea

[152]:
```
dilation = cv2.dilate(imgThresh, kernel, iterations = 1)

fig = plt.figure()
ax1 = fig.add_subplot(121)
ax1.set_title('Imaginea binara')
ax1.imshow(imgThresh, cmap='gray')
ax2 = fig.add_subplot(122)
ax2.set_title('Imaginea dilatata')
ax2.imshow(dilation, cmap='gray')
```

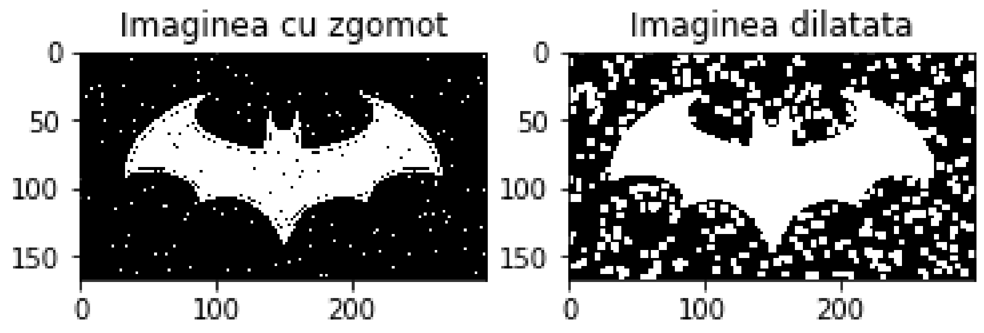
[152]:
```
<matplotlib.image.AxesImage at 0x1b986d77088>
```



5 Dilatarea pe imaginea cu zgomot

[153]:
```python
dilation = cv2.dilate(imZgomot, kernel, iterations = 1)

fig = plt.figure()
ax1 = fig.add_subplot(121)
ax1.set_title('Imaginea cu zgomot')
ax1.imshow(imZgomot, cmap='gray')
ax2 = fig.add_subplot(122)
ax2.set_title('Imaginea dilatata')
ax2.imshow(dilation, cmap='gray')
```

[153]:
```
<matplotlib.image.AxesImage at 0x1b986e48fc8>
```

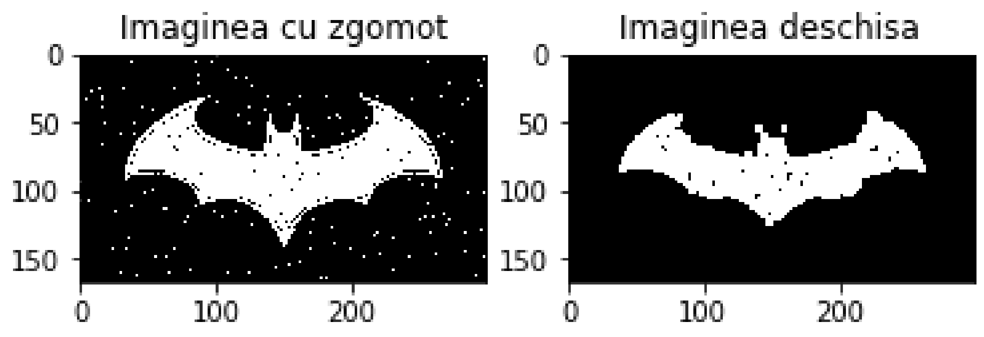


6 Deschiderea pe imaginea cu zgomot

[154]:
```python
opening = cv2.morphologyEx(imZgomot, cv2.MORPH_OPEN, kernel)

fig = plt.figure()
```

```
ax1 = fig.add_subplot(121)
ax1.set_title('Imaginea cu zgomot')
ax1.imshow(imZgomot, cmap='gray')
ax2 = fig.add_subplot(122)
ax2.set_title('Imaginea deschisa')
ax2.imshow(opening, cmap='gray')
```

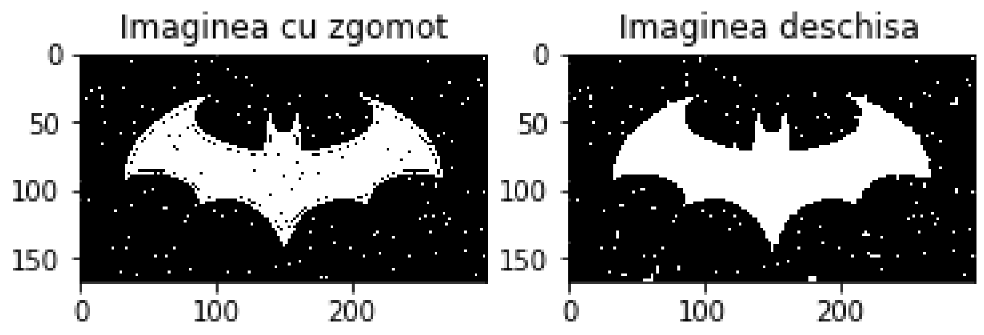
```
[154]: <matplotlib.image.AxesImage at 0x1b9875d1788>
```



7 Inchiderea pe imaginea cu zgomot

```
[155]: closing = cv2.morphologyEx(imZgomot, cv2.MORPH_CLOSE, kernel)

fig = plt.figure()
ax1 = fig.add_subplot(121)
ax1.set_title('Imaginea cu zgomot')
ax1.imshow(imZgomot, cmap='gray')
ax2 = fig.add_subplot(122)
ax2.set_title('Imaginea deschisa')
ax2.imshow(closing, cmap='gray')
```

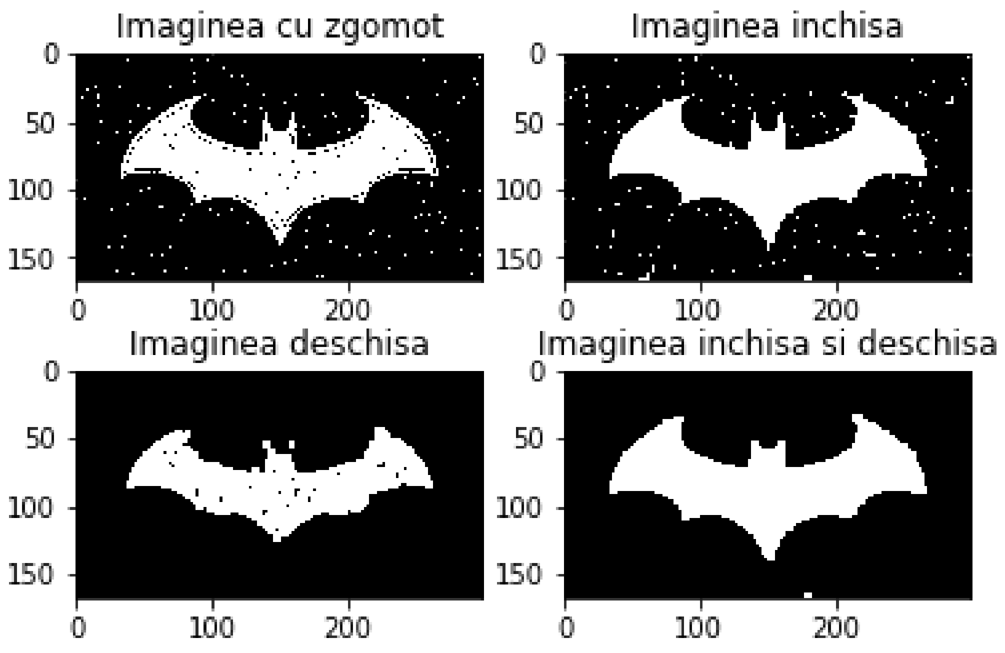
```
[155]: <matplotlib.image.AxesImage at 0x1b98791d048>
```



8 Inchidere si apoi deschidere pe imaginea cu zgomot

[156]:
```
closeOpen = cv2.morphologyEx(closing, cv2.MORPH_OPEN, kernel)

fig = plt.figure()
ax1 = fig.add_subplot(221)
ax1.set_title('Imaginea cu zgomot')
ax1.imshow(imZgomot, cmap='gray')
ax2 = fig.add_subplot(222)
ax2.set_title('Imaginea inchisa')
ax2.imshow(closing, cmap='gray')
ax3 = fig.add_subplot(223)
ax3.set_title('Imaginea deschisa')
ax3.imshow(opening, cmap='gray')
ax4 = fig.add_subplot(224)
ax4.set_title('Imaginea inchisa si deschisa')
ax4.imshow(closeOpen, cmap='gray')
```

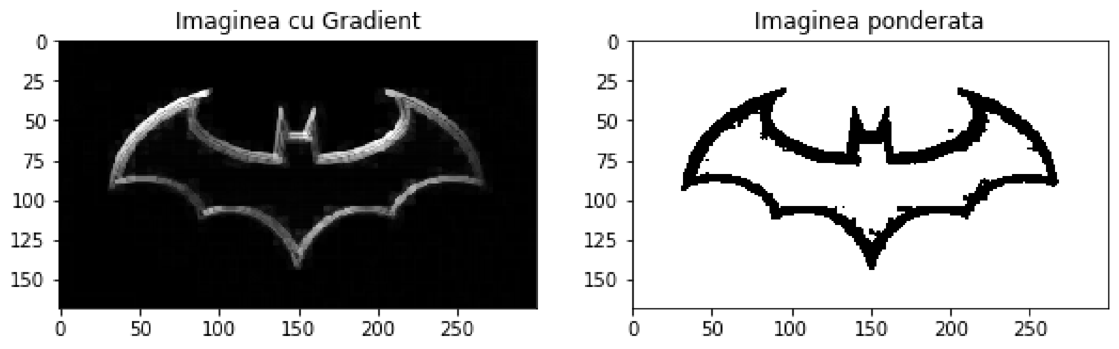
[156]:
```
<matplotlib.image.AxesImage at 0x1b987aef508>
```



[159]:
```
elemSize = 3
kernel = np.ones((elemSize, elemSize), np.uint8)
```

```
edges = cv2.morphologyEx(img, cv2.MORPH_GRADIENT, kernel)

ret, edgesThresh = cv2.threshold(edges, 20, 255, cv2.THRESH_BINARY_INV)

fig = plt.figure(figsize = (10, 4))
ax1 = fig.add_subplot(121)
ax1.set_title('Imaginea cu Gradient')
ax1.imshow(edges, cmap='gray')
ax2 = fig.add_subplot(122)
ax2.set_title('Imaginea ponderata')
ax2.imshow(edgesThresh, cmap='gray')
```

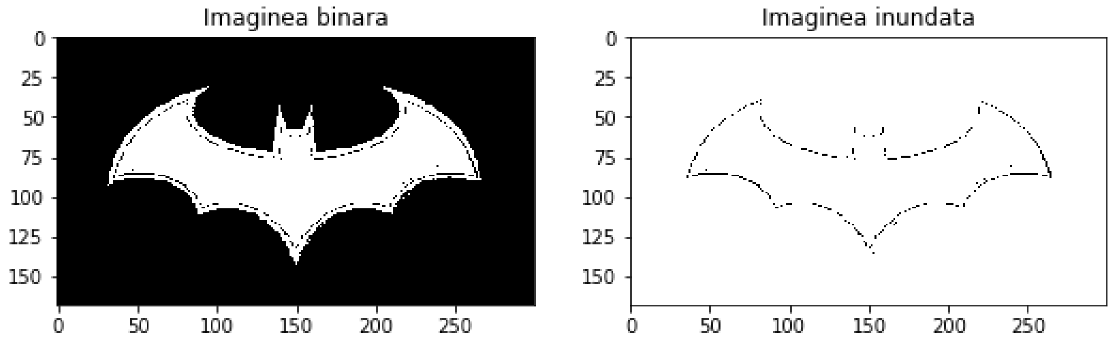
[159]:
```
<matplotlib.image.AxesImage at 0x1b987d5bf08>
```



9 Inundarea unei imagini

[164]:
```
h, w = imgThresh.shape[:2]

masca = np.zeros((h+2, w+2), np.uint8)

edgesFlood = imgThresh.copy()
# Inundam pornind de la punctul (0, 0)
#pixelii ce nu sunt afectati sunt cei din interiorul regiunii delimitate
cv2.floodFill(edgesFlood, masca, (0,0), 255)

fig = plt.figure(figsize = (10, 4))
ax1 = fig.add_subplot(121)
ax1.set_title('Imaginea binara')
ax1.imshow(imgThresh, cmap='gray')
ax2 = fig.add_subplot(122)
ax2.set_title('Imaginea inundata')
ax2.imshow(edgesFlood, cmap='gray')
```

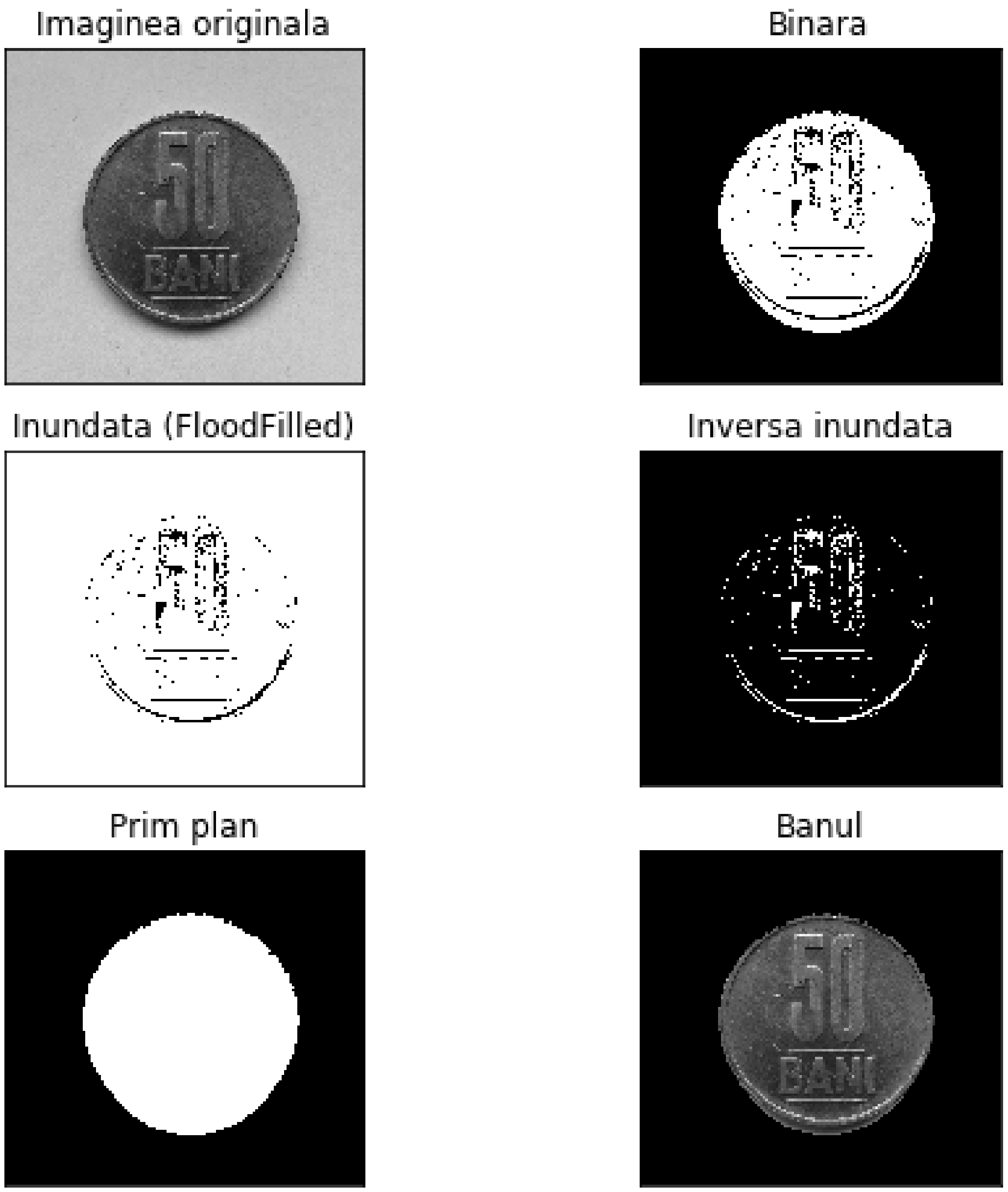
```
[164]: <matplotlib.image.AxesImage at 0x1b98adbc948>
```



10 Detectarea obiectului principal dintr-o imagine

```
[116]: import cv2
import numpy as np
import matplotlib.pyplot as plt

im = cv2.imread("D:/ban.jpg", 0)

# Aplicam thresholding binar astfel incat fundalul sa fie negru
# Valoarea pragului de mai jos este foarte importanta pentru un rez bun
_, imBinara = cv2.threshold(im, 100, 255, cv2.THRESH_BINARY_INV)

# Facem o copie a imaginii binare.
imFloodFill = imBinara.copy()

# Facem masca pentru flood filling
# Dimensiunea trebuie sa fie cu 2 pixeli mai mare decat imaginea.
#[:2] face sa ia doar primele 2 argumente (fara numarul de canale)
h, w = imBinara.shape[:2]
masca = np.zeros((h+2, w+2), np.uint8)

# Inundam pornind de la punctul (0, 0)
#pixelii ce nu sunt afectati sunt cei din interiorul regiunii delimitate
cv2.floodFill(imFloodFill, masca, (0,0), 255)

# In unele cazuri imFloodFill poate deja contine obiectul cautat

# Inversam imaginea inundata
imFloodFillInv = cv2.bitwise_not(imFloodFill)

'''
```

```
Combinam cele doua imagini imBinara si imFloodFillInv
ca sa obtinem prim planul, iar fundalul sa fie alb.
Se poate scrie si ca mai jos in loc de bitwise_or
imPrim = imBinara | imFloodFillInv
'''
imPrim = cv2.bitwise_or(imBinara, imFloodFillInv)

imBan = im & imPrim #sau cv2.bitwise_and(im, imPrim)

titles = ['Imaginea originala','Binara','Inundata (FloodFilled)',
          'Inversa inundata','Prim plan', 'Banul']
images = [im, imBinara, imFloodFill, imFloodFillInv, imPrim, imBan]
fig = plt.figure(figsize=(8, 8))
for i in range(6):
    plt.subplot(3,2,i+1)
    plt.imshow(images[i],'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])
plt.show()
```

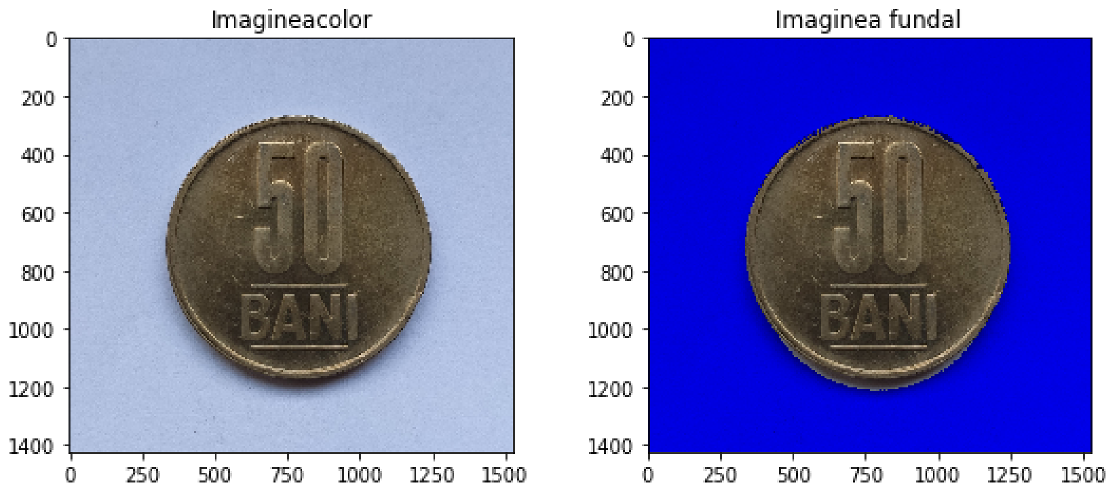


11 Putem sa schimbam culoarea de fundal

[166]:
```
#gasim toti pixelii negri din imaginea cu prim planul
indices = np.where(imPrim==0)

#transformam prim planul in format RGB
backtorgb = cv2.cvtColor(imPrim,cv2.COLOR_GRAY2RGB)

#schimbam pixelii negri cu unii albastri
```

```
backtorgb[indices[0], indices[1], :] = [0, 0, 255]

#recitim poza, de aceasta data color
imColor = cv2.imread("D:/ban.jpg")
imColor = cv2.cvtColor(imColor,cv2.COLOR_BGR2RGB)

#Combinam imaginea color cu cea cu fundalul
imBanBlue = imColor & backtorgb

fig = plt.figure(figsize = (10, 4))
ax1 = fig.add_subplot(121)
ax1.set_title('Imagineacolor')
ax1.imshow(imColor, cmap='gray')
ax2 = fig.add_subplot(122)
ax2.set_title('Imaginea fundal')
ax2.imshow(imBanBlue)
```

[166]:
```
<matplotlib.image.AxesImage at 0x1b98b5b5e88>
```



[ ]: