Sveučilište u Splitu
Fakultet elektrotehnike, strojarstva i brodogradnje

Laboratorijske vježbe iz
digitalne obrada i analiza slike

Vježba 5:

Morfološke operacije na digitalnoj slici

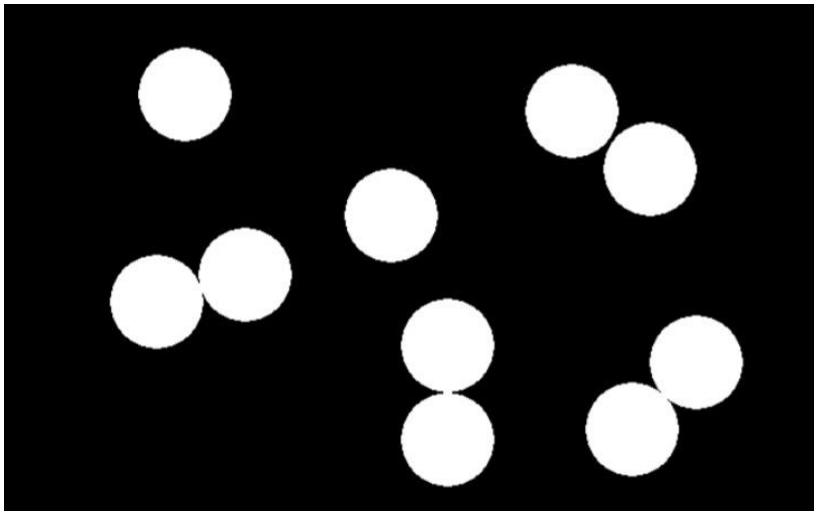
Split, 2020.

Zadatak 1.

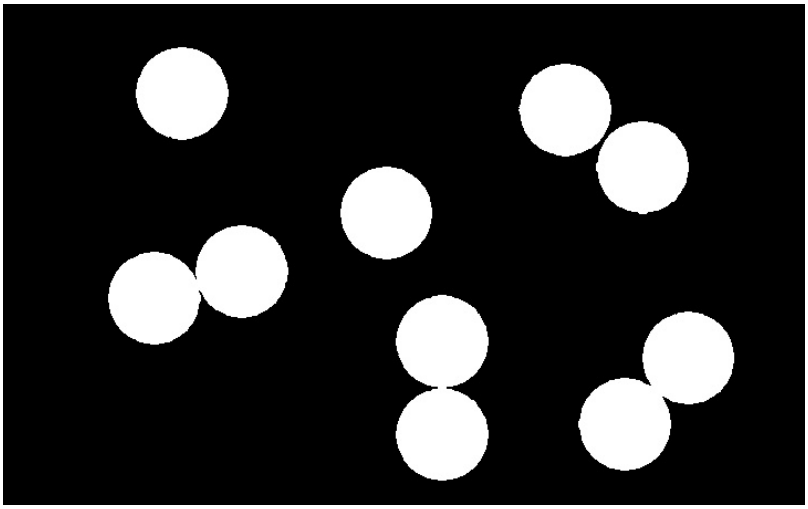
U Pythonu učitajte sliku prikazanu na slici 5.5.

- Učitanu sliku konvertirajte u binarnu sliku.
- Pomoću funkcije cv2.SimpleBlobDetector() odredite koliko se pojedinačnih regija nalazi na učitanoj slici.
- Na učitanu sliku primijenite operaciju erozije i odredite koliko se regija nalazi na novoj slici. Pokušajte koristiti različite veličine filtera za operaciju erozije da vidite koja će vam najviše odgovarati.

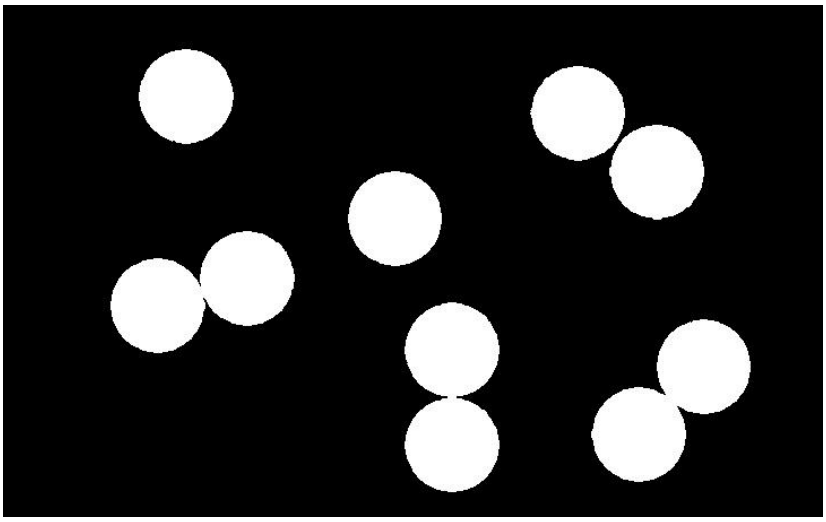
Originalna slika:



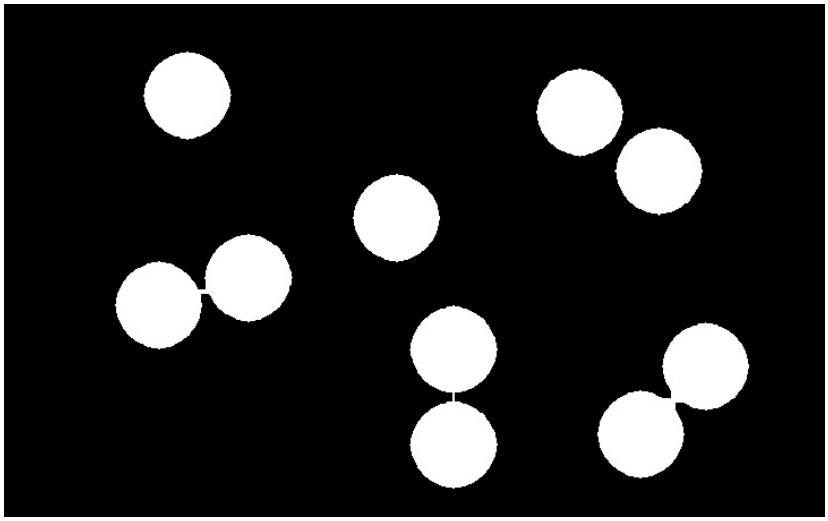
Binarna slika:



Nakon cv2.SimpleBlobDetector():



Nakon mog filtera:



Kod:

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

img = cv2.imread('../img/img_5_1.jpg', cv2.IMREAD_GRAYSCALE)

(prag, img_binary) = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)

params = cv2.SimpleBlobDetector_Params()

# Change thresholds
params.minThreshold = 10;
params.maxThreshold = 200;
```

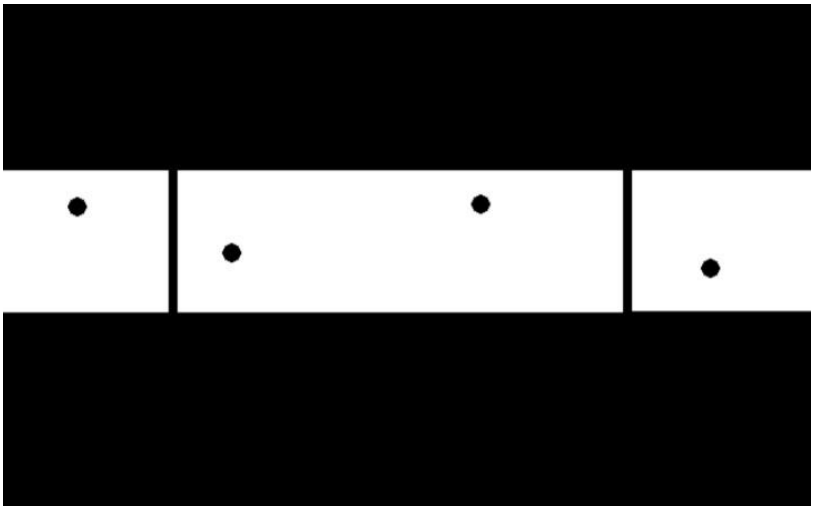
```
# Filter by Area.
params.filterByArea = True
params.minArea = 1500

# Filter by Circularity
params.filterByCircularity = True
params.minCircularity = 0.1

# Filter by Convexity
params.filterByConvexity = True
params.minConvexity = 0.87

# Filter by Inertia
params.filterByInertia = True
params.minInertiaRatio = 0.01

# Create a detector with the parameters
ver = (cv2.__version__).split('.')
if int(ver[0]) < 3 :
    detector = cv2.SimpleBlobDetector(params)
else :
    detector = cv2.SimpleBlobDetector_create(params)

keypoints = detector.detect(img_binary)
with_keypoints = cv2.drawKeypoints(img_binary, keypoints, np.array([]), (0,0,2
55), cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
cv2.imwrite("../img/z1_keypoints.jpg", with_keypoints)

my_filter = np.ones((7,7), np.uint8)
img_erozija = cv2.erode(img_binary, my_filter, iterations = 1)

cv2.imwrite('../img/img_binary.jpg', img_binary)
cv2.imwrite('../img/img_erozija.jpg', img_erozija)
```

Zadatak 2.

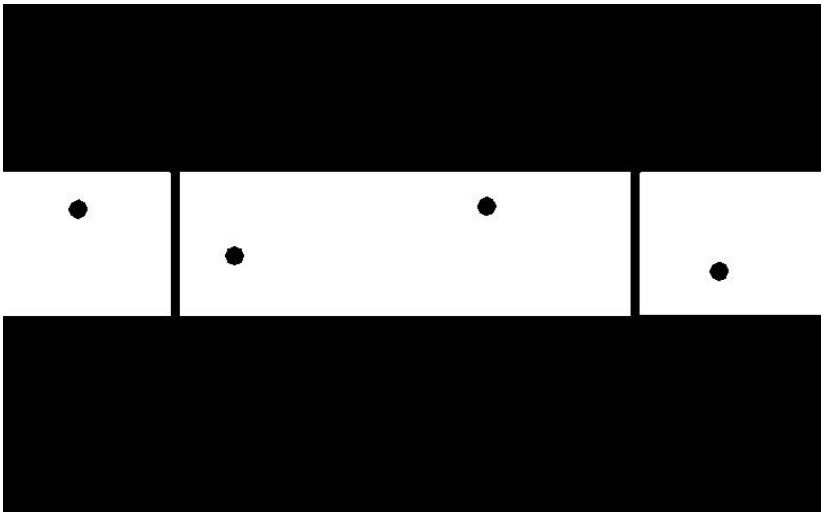
U Pythonu učitajte sliku prikazanu na slici 5.6.

- Učitanu sliku konvertirajte u binarnu sliku.
- Pomoću funkcije cv2.SimpleBlobDetector() odredite koliko se pojedinačnih regija nalazi na učitanoj slici.
- Na učitanu sliku primijenite operaciju dilatacije i odredite koliko se regija nalazi na novoj slici. Pokušajte koristiti različite veličine filtera za operaciju dilatacije da vidite koja će vam najviše odgovarati.

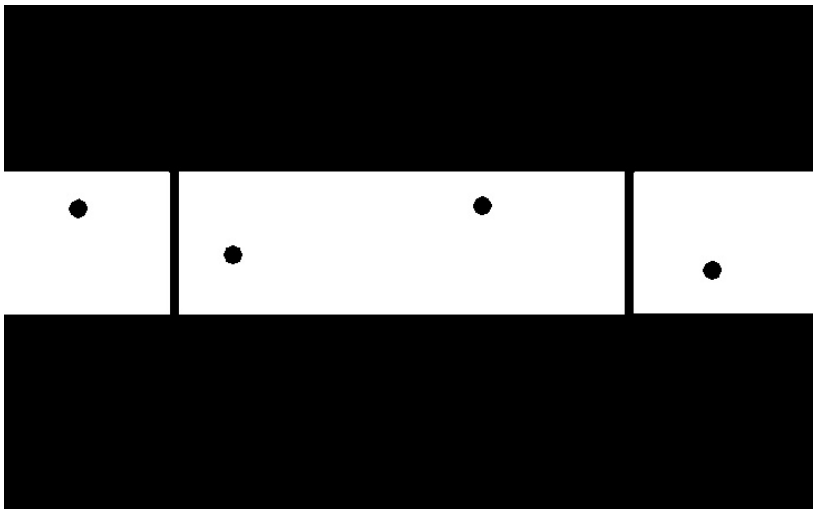
Originalna slika:



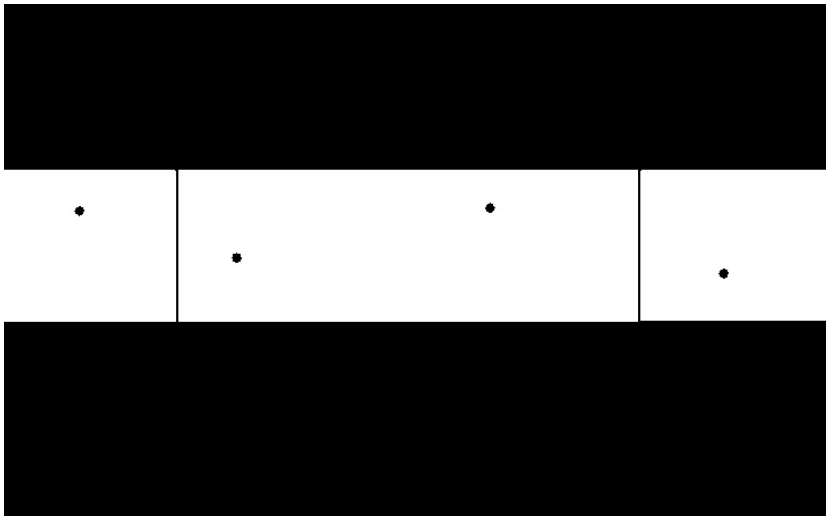
Binarna slika:



Nakon cv2.SimpleBlobDetector():



Nakon mog filtera:



Kod:

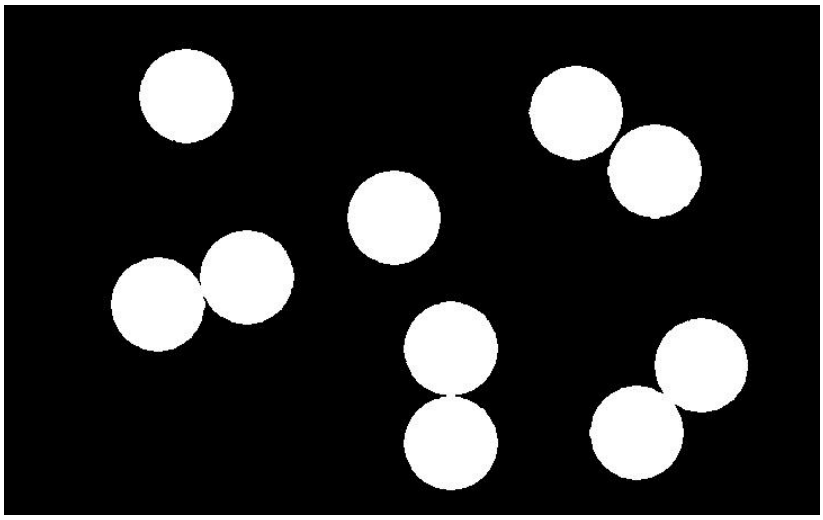
```
import numpy as np
import cv2
from matplotlib import pyplot as plt

img = cv2.imread('../img/img_5_2.jpg', cv2.IMREAD_GRAYSCALE)

(prag, img_binary) = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)

params = cv2.SimpleBlobDetector_Params()

# Change thresholds
params.minThreshold = 10;
params.maxThreshold = 200;
```

```
# Filter by Area.
params.filterByArea = True
params.minArea = 1500

# Filter by Circularity
params.filterByCircularity = True
params.minCircularity = 0.1

# Filter by Convexity
params.filterByConvexity = True
params.minConvexity = 0.87

# Filter by Inertia
params.filterByInertia = True
params.minInertiaRatio = 0.01

# Create a detector with the parameters
ver = (cv2.__version__).split('.')
if int(ver[0]) < 3 :
    detector = cv2.SimpleBlobDetector(params)
else :
    detector = cv2.SimpleBlobDetector_create(params)

keypoints = detector.detect(img_binary)
with_keypoints = cv2.drawKeypoints(img_binary, keypoints, np.array([]), (0,0,2
55), cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
cv2.imwrite("../img/z2_keypoints.jpg", with_keypoints)

my_filter = np.ones((7,7), np.uint8)
img_dilatacija = cv2.dilate(img_binary, my_filter, iterations = 1)

cv2.imwrite('../img/img_z2_binary.jpg', img_binary)
cv2.imwrite('../img/img_z2_dilatacija.jpg', img_dilatacija)
```

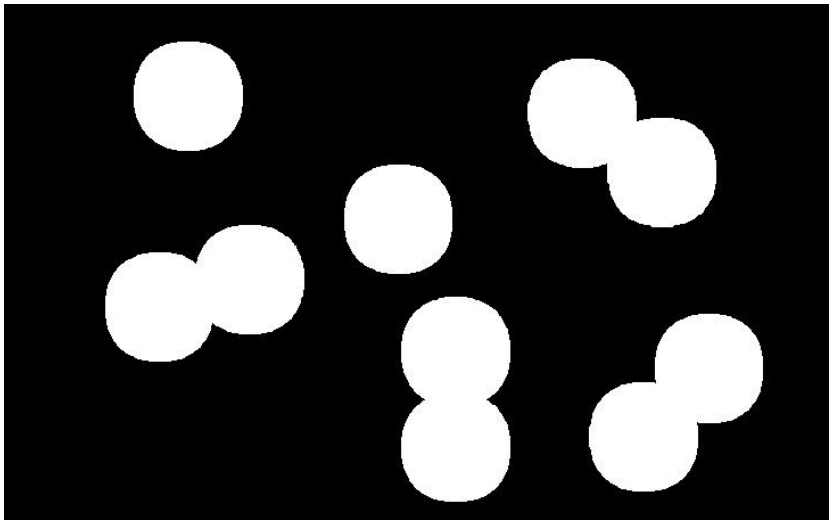
Zadatak 3.

U Pythonu učitajte sliku prikazanu na slici 5.5 te na nju primijenite morfološku operaciju otvaranja. Po čemu se dobiveni rezultati razlikuju od onih dobivenih u 1. zadatku?

Binarna slika:



Filtrirana slika:



Kod:

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

img = cv2.imread('../img/img_5_1.jpg', cv2.IMREAD_GRAYSCALE)

(prag, img_binary) = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)

params = cv2.SimpleBlobDetector_Params()
```

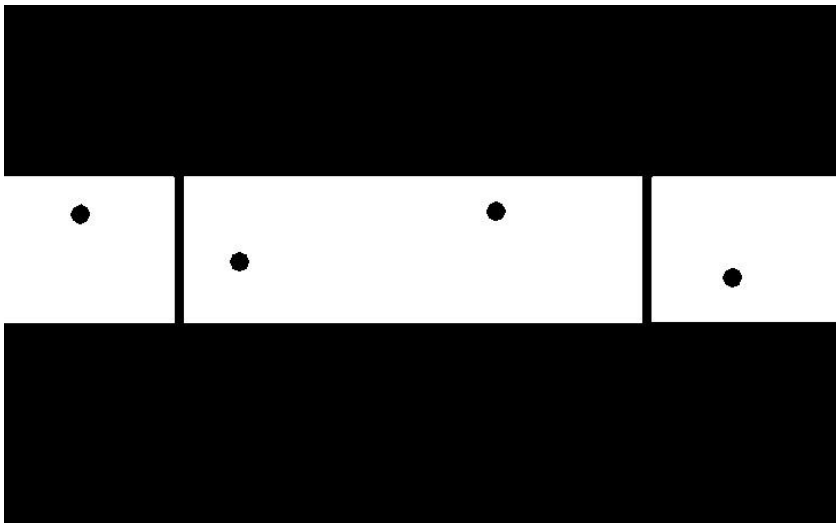
```
# Change thresholds
params.minThreshold = 10;
params.maxThreshold = 200;

# Filter by Area.
params.filterByArea = True
params.minArea = 1500

# Filter by Circularity
params.filterByCircularity = True
params.minCircularity = 0.1

# Filter by Convexity
params.filterByConvexity = True
params.minConvexity = 0.87

# Filter by Inertia
params.filterByInertia = True
params.minInertiaRatio = 0.01

# Create a detector with the parameters
ver = (cv2.__version__).split('.')
if int(ver[0]) < 3 :
    detector = cv2.SimpleBlobDetector(params)
else :
    detector = cv2.SimpleBlobDetector_create(params)

keypoints = detector.detect(img_binary)
with_keypoints = cv2.drawKeypoints(img_binary, keypoints, np.array([]), (0,0,2
55), cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
cv2.imwrite("../img/z3_keypoints.jpg", with_keypoints)

my_filter = np.ones((7,7), np.uint8)
img_erozija = cv2.dilate(img_binary, my_filter, iterations = 1)
img_dilatacija = cv2.dilate(img_erozija, my_filter, iterations = 1)

cv2.imwrite('../img/img_z3_otvaranje.jpg', img_dilatacija)
```

U prvom smo zadatku rupe (cjeline) dodatno razdvojili, dok smo morfološkim otvaranjem bliske rupe tj. cjeline pospajali.

Zadatak 4.

U Pythonu učitajte sliku prikazanu na slici 5.6 te na nju primijenite morfološku operaciju zatvaranja. Po čemu se dobiveni rezultati razlikuju od onih dobivenih u 2. zadatku?

Binarna slika:



Filtrirana slika:

Kod:

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

img = cv2.imread('../img/img_5_2.jpg', cv2.IMREAD_GRAYSCALE)

(prag, img_binary) = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)
```

```
params = cv2.SimpleBlobDetector_Params()

# Change thresholds
params.minThreshold = 10;
params.maxThreshold = 200;

# Filter by Area.
params.filterByArea = True
params.minArea = 1500

# Filter by Circularity
params.filterByCircularity = True
params.minCircularity = 0.1

# Filter by Convexity
params.filterByConvexity = True
params.minConvexity = 0.87

# Filter by Inertia
params.filterByInertia = True
params.minInertiaRatio = 0.01

# Create a detector with the parameters
ver = (cv2.__version__).split('.')
if int(ver[0]) < 3 :
    detector = cv2.SimpleBlobDetector(params)
else :
    detector = cv2.SimpleBlobDetector_create(params)

keypoints = detector.detect(img_binary)
with_keypoints = cv2.drawKeypoints(img_binary, keypoints, np.array([]), (0,0,2
55), cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
cv2.imwrite("../img/z4_keypoints.jpg", with_keypoints)

my_filter = np.ones((7,7), np.uint8)
img_dilatacija = cv2.dilate(img_binary, my_filter, iterations = 1)
img_erozija = cv2.dilate(img_dilatacija, my_filter, iterations = 1)

cv2.imwrite('../img/img_z4_zatvaranje.jpg', img_erozija)
```

Za razliku od 2 zadatka gdje smo praznine, tj. cjeline samo smanjili ovdje smo ih „zatvorili“ tj. popunili/uklonili.