Laboratorijske vježbe iz

digitalne obrada i analiza slike

**Vježba 5:**

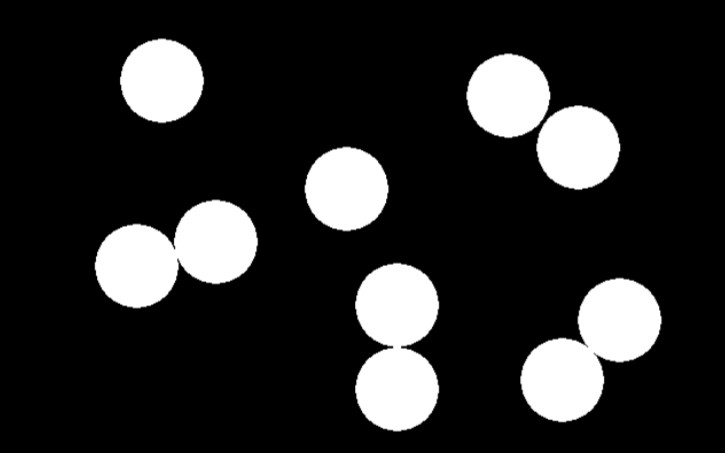
**Morfološke operacije na digitalnoj slici**

**Zadatak 1.**

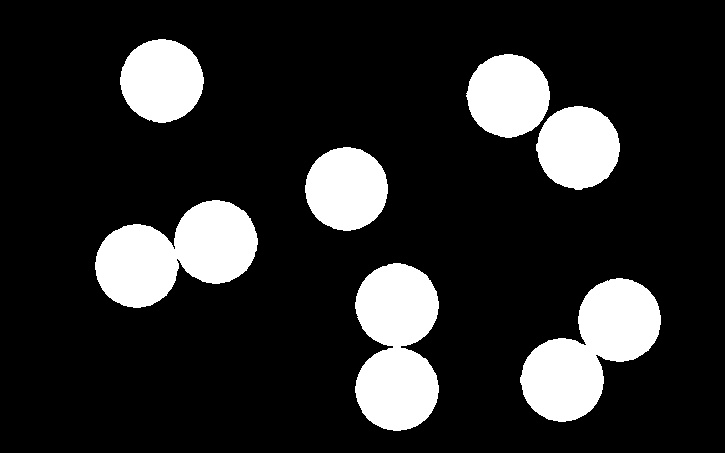
U Pythonu učitajte sliku prikazanu na slici 5.5.

* 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.

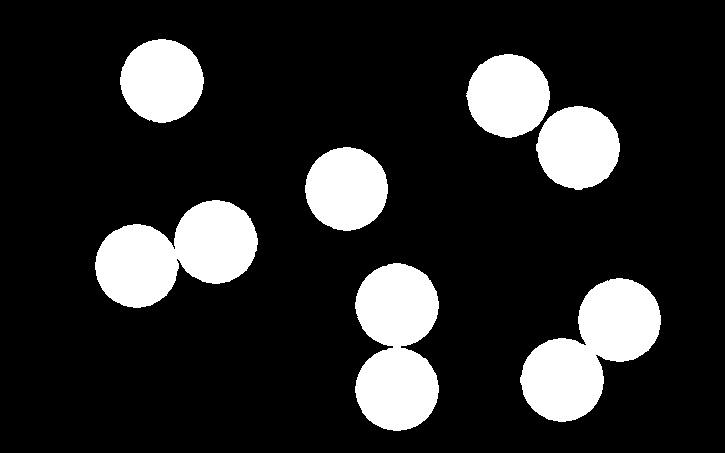
**Originalna slika:**

****

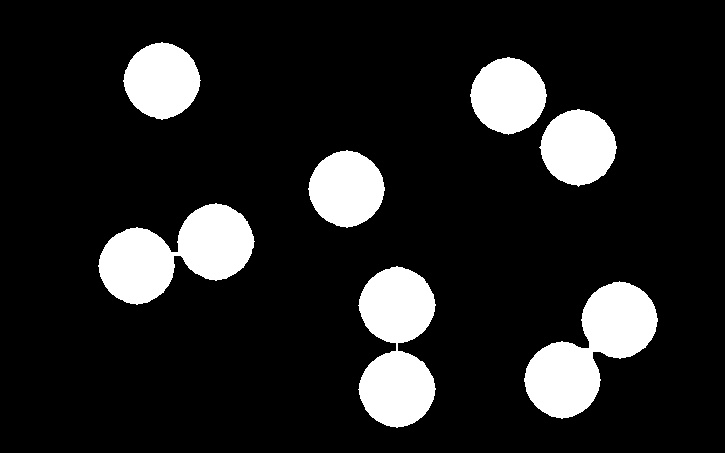
**Binarna slika:**

****

**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([]), (0,0,255), 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)

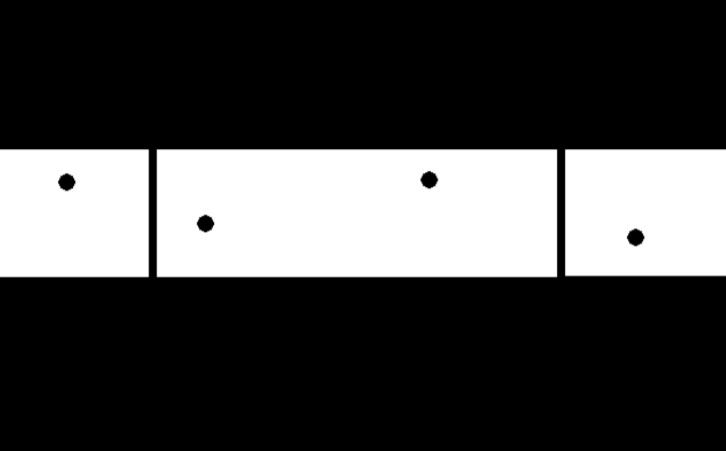
cv2.imwrite('../img/img\_erozija.jpg', img\_erozija)

**Zadatak 2.**

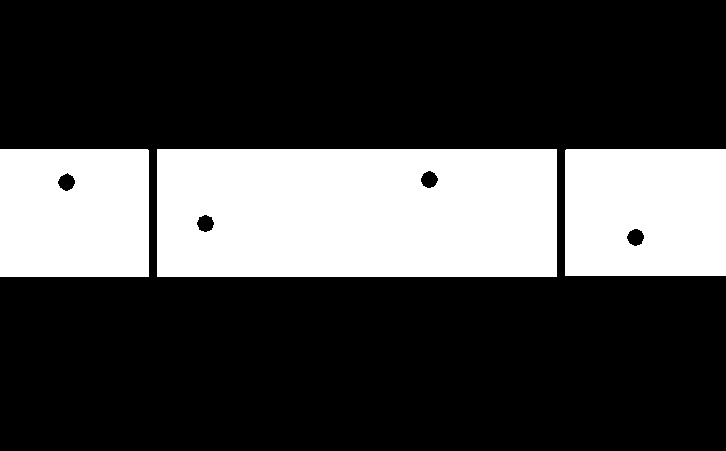
U Pythonu učitajte sliku prikazanu na slici 5.6.

* 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.

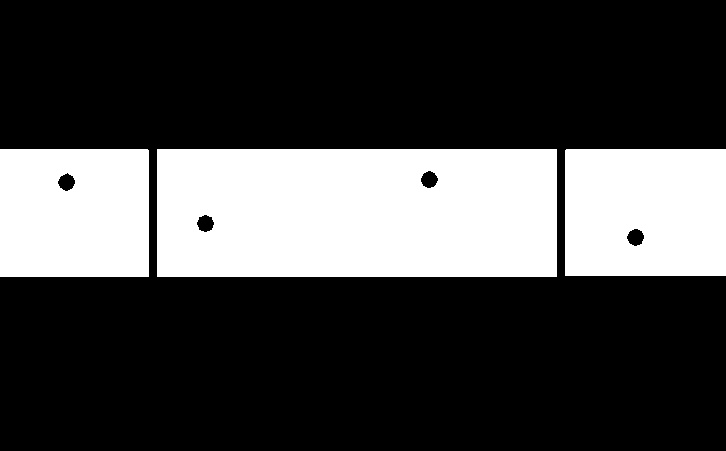
**Originalna slika:**



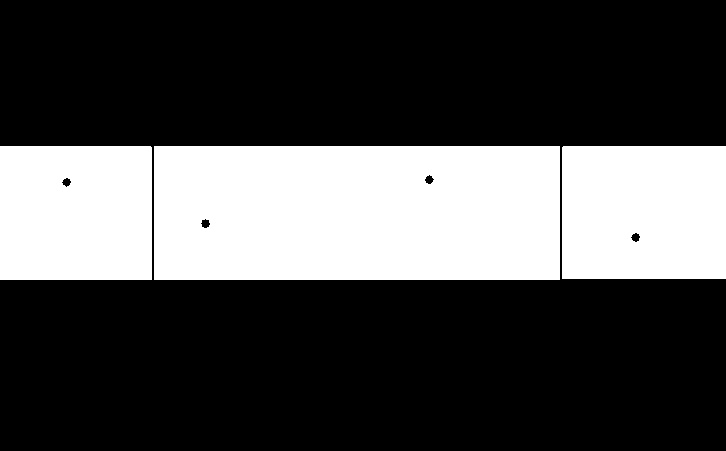
**Binarna slika:**



**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([]), (0,0,255), 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)

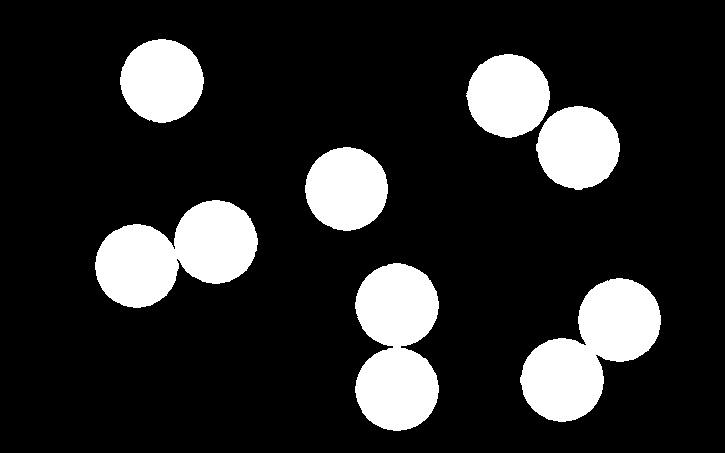
cv2.imwrite('../img/img\_z2\_binary.jpg', img\_binary)

cv2.imwrite('../img/img\_z2\_dilatacija.jpg', img\_dilatacija)

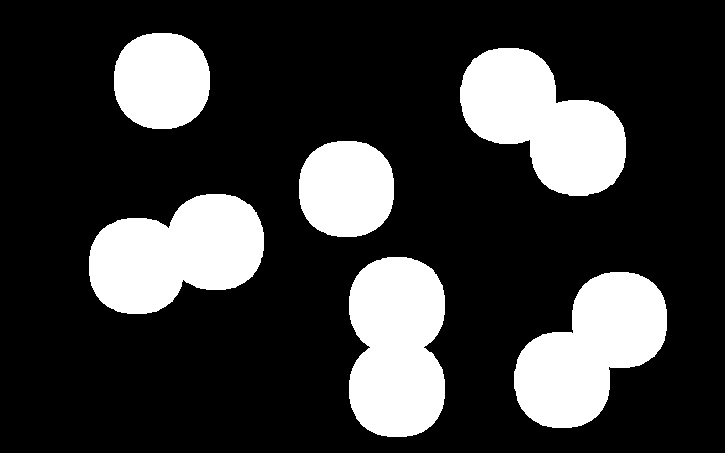
**Zadatak 3.**

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([]), (0,0,255), 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)

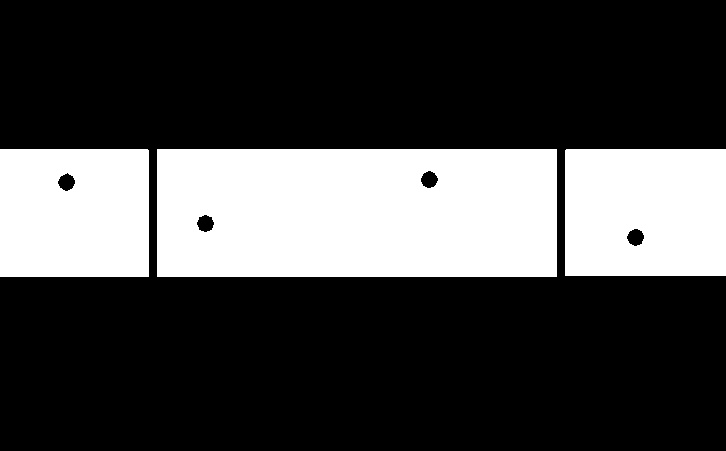
cv2.imwrite('../img/img\_z3\_otvaranje.jpg', img\_dilatacija)

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([]), (0,0,255), 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.