**ЛАБОРАТОРНА РОБОТА № 8**

**ДОСЛІДЖЕННЯ МЕТОДІВ КОМП’ЮТЕРНОГО ЗОРУ**

**Мета**: використовуючи спеціалізовані бібліотеки та мову програмування Python навчитися обробляти зображення за допомогою бібліотеки OpenCV

**Хід роботи:**

Завдання №1: Завантаження зображень та відео в OpenCV.

Лістинг програми:

import cv2  
# LOAD AN IMAGE USING 'IMREAD'  
img = cv2.imread("Voitko.jpg")  
# DISPLAY  
cv2.imshow("Voitko", img)  
cv2.waitKey(0)

Зображення, що містить небо, вода, надворі, земля

Автоматично згенерований опис

Рис. 1. Результат виконання програми

Завдання №2: Дослідження перетворень зображення.

Лістинг програми:

import cv2  
import numpy as np  
img = cv2.imread("Voitko.jpg")  
kernel = np.ones((5, 5), np.uint8)  
imgGray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
imgBlur = cv2.GaussianBlur(imgGray, (7, 7), 0)  
imgCanny = cv2.Canny(img, 150, 200)  
imgDialation = cv2.dilate(imgCanny, kernel, iterations=1)  
imgEroded = cv2.erode(imgDialation, kernel, iterations=1)  
cv2.imshow("Gray Image", imgGray)  
cv2.imshow("Blur Image", imgBlur)  
cv2.imshow("Canny Image", imgCanny)  
cv2.imshow("Dialation Image", imgDialation)  
cv2.imshow("Eroded Image", imgEroded)  
cv2.waitKey(0)

Зображення, що містить вода, надворі, особа

Автоматично згенерований опис

Рис. 2. Результат виконання програми

Зображення, що містить надворі, особа

Автоматично згенерований опис

Рис. 3. Результат виконання програми

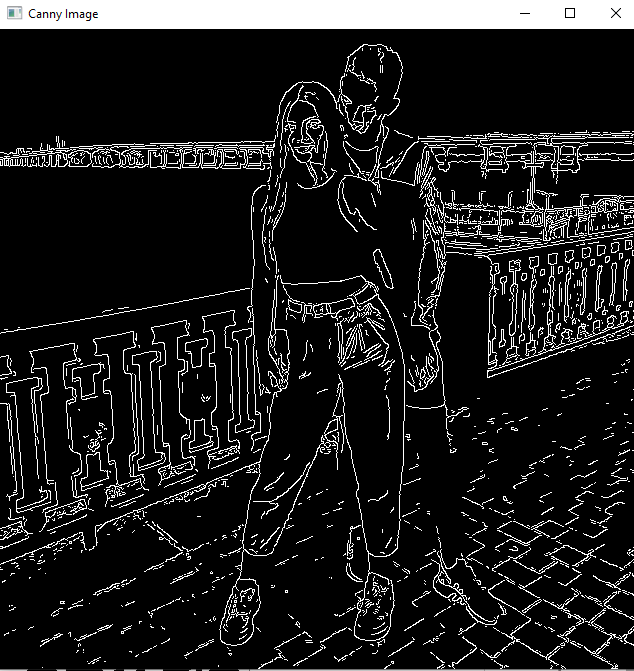


Рис. 4. Результат виконання програми

Зображення, що містить текст

Автоматично згенерований опис

Рис. 5. Результат виконання програми

Зображення, що містить текст

Автоматично згенерований опис

Рис. 6. Результат виконання програми

Метод **cvtColor** використовується для перетворення зображення з одного колірного простору в інший, в результаті його використання було отримано зображення у градації сірого кольору.

Метод **GaussianBlur** використовується для застосування Гаусового згладжування до зображення, в результаті його використання було отримано замилене зображення.

Метод **Canny** використовується для виявлення країв зображення, в результаті його використання було отримано зображення з контурами обличчя.

Метод **dilate** використовується для зменшення особливостей зображення, в результаті його використання було отримано зображення з контурами обличчя.

Метод **erode** використовується для підкреслення рис, в результаті його використання було отримано зображення з розмитим контуром обличч

Завдання №3: Вирізання частини зображення.

Лістинг програми:

import cv2  
import numpy as np  
img = cv2.imread("Voitko.jpg")  
print(img.shape)  
imgResize = cv2.resize(img, (1000, 500))  
print(imgResize.shape)  
imgCropped = img[75:400, 30:350]  
cv2.imshow("Image", img)  
# cv2.imshow("Image Resize",imgResize)  
cv2.imshow("Image Cropped", imgCropped)  
cv2.waitKey(0)

Зображення, що містить текст, вода, особа, жінка

Автоматично згенерований опис

Рис. 7. Результат виконання програми

Завдання №4: Розпізнавання обличчя на зображенні.

Лістинг програми:

import cv2  
faceCascade = cv2.CascadeClassifier("haarcascade\_frontalface\_default.xml")  
img = cv2.imread('Voitko.jpg')  
imgGray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
faces = faceCascade.detectMultiScale(imgGray, 1.1, 4)  
for (x, y, w, h) in faces:  
 cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)  
cv2.imshow("Result", img)  
cv2.waitKey(0)

Зображення, що містить небо, вода, надворі, особа

Автоматично згенерований опис

Рис. 8. Результат виконання програми

Було отримано поганий результат з розпізнавання обличчя на прикладі фото. Тому що нас 2, а обличчя показало тільки 1

Звдання №5: Розпізнавання об’єктів на зображенні за допомогою методів зіставлення шаблонів (Template Matching).

import cv2 as cv  
import numpy as np  
from matplotlib import pyplot as plt  
  
img = cv.imread('VOITKO.jpg', 0)  
img2 = img.copy()  
template = cv.imread('myface\_face.jpg', 0)  
w, h = template.shape[::-1]  
# All the 6 methods for comparison in a list  
methods = ['cv.TM\_CCOEFF', 'cv.TM\_CCOEFF\_NORMED', 'cv.TM\_CCORR',  
 'cv.TM\_CCORR\_NORMED', 'cv.TM\_SQDIFF', 'cv.TM\_SQDIFF\_NORMED']  
for meth in methods:  
 img = img2.copy()  
 method = eval(meth)  
 # Apply template Matching  
 res = cv.matchTemplate(img, template, method)  
 min\_val, max\_val, min\_loc, max\_loc = cv.minMaxLoc(res)  
 # If the method is TM\_SQDIFF or TM\_SQDIFF\_NORMED, take minimum  
 if method in [cv.TM\_SQDIFF, cv.TM\_SQDIFF\_NORMED]:  
 top\_left = min\_loc  
 else:  
 top\_left = max\_loc  
 bottom\_right = (top\_left[0] + w, top\_left[1] + h)  
 cv.rectangle(img, top\_left, bottom\_right, 255, 2)  
 plt.subplot(121), plt.imshow(res, cmap='gray')  
 plt.title('Matching Result'), plt.xticks([]), plt.yticks([])  
 plt.subplot(122), plt.imshow(img, cmap='gray')  
 plt.title('Detected Point'), plt.xticks([]), plt.yticks([])  
 plt.suptitle(meth)  
 plt.show()

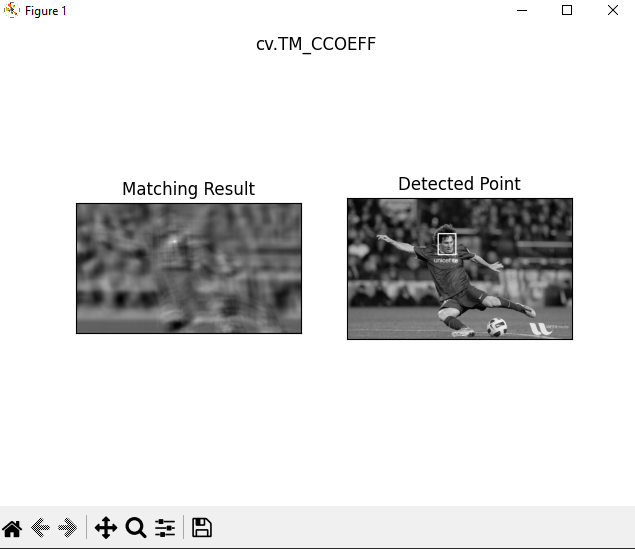


Рис. 8. Результат виконання програми

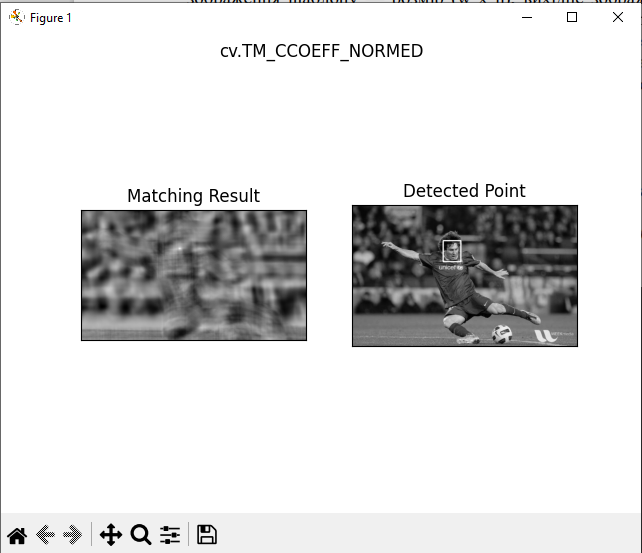


Рис. 9. Результат виконання програми

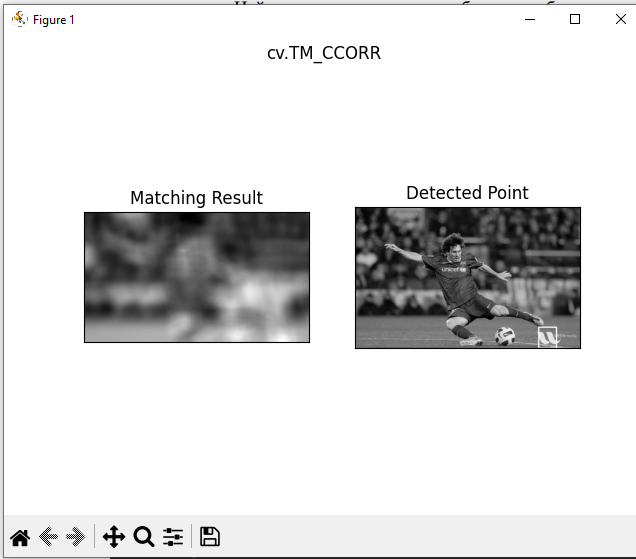


Рис. 10. Результат виконання програми

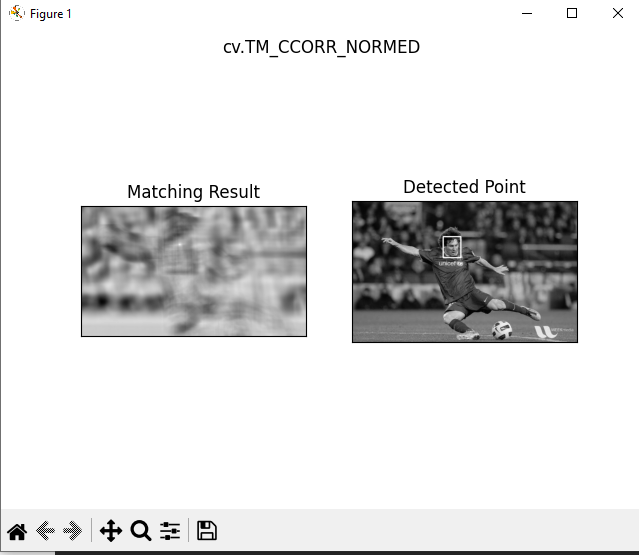


Рис. 11. Результат виконання програми

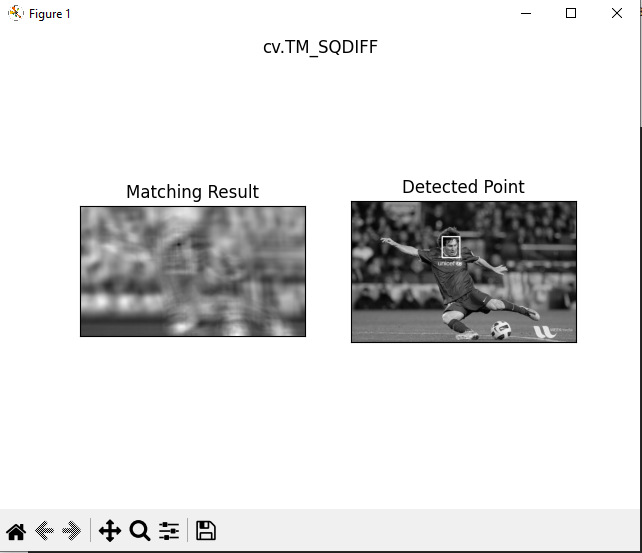


Рис. 12. Результат виконання програми



Рис. 13. Результат виконання програми

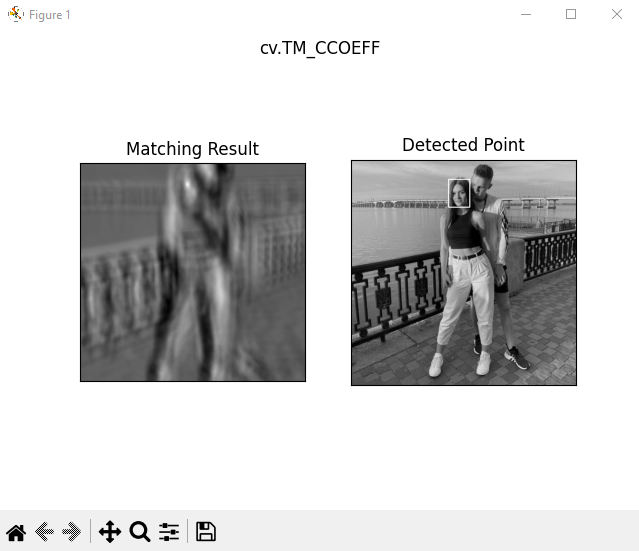


Рис. 14. Результат виконання програми

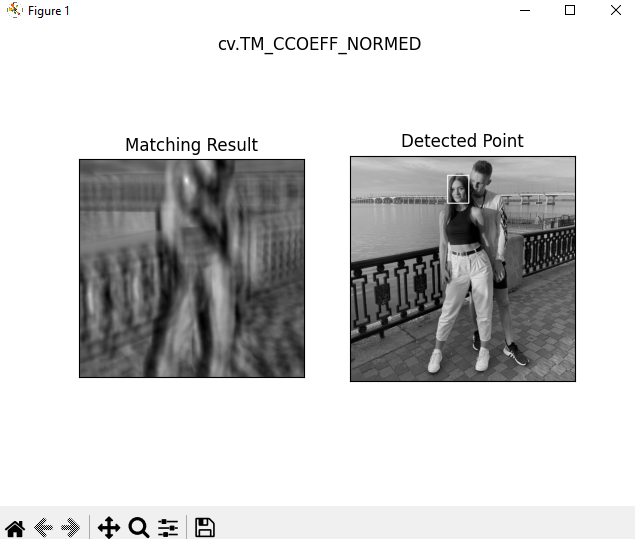


Рис. 15. Результат виконання програми

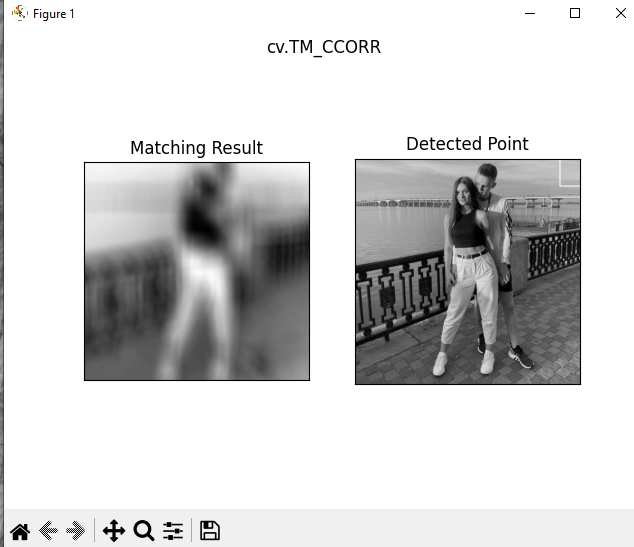


Рис. 16. Результат виконання програми

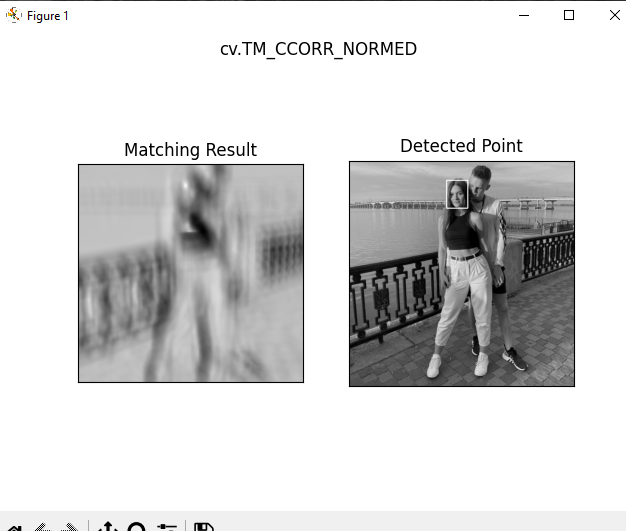


Рис. 17. Результат виконання програми

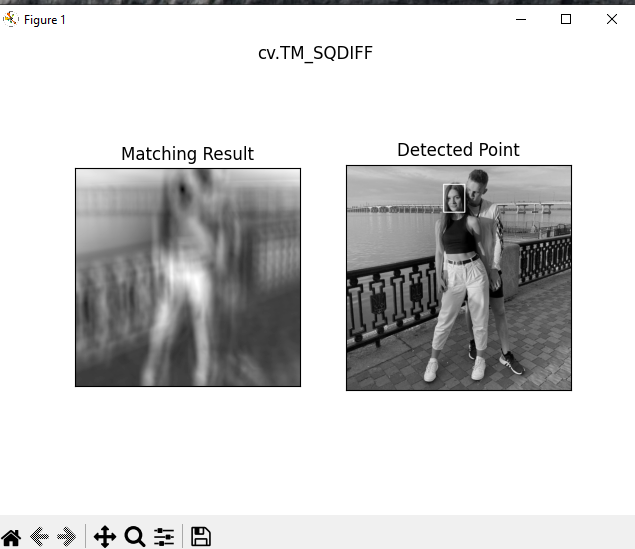


Рис. 18. Результат виконання програми

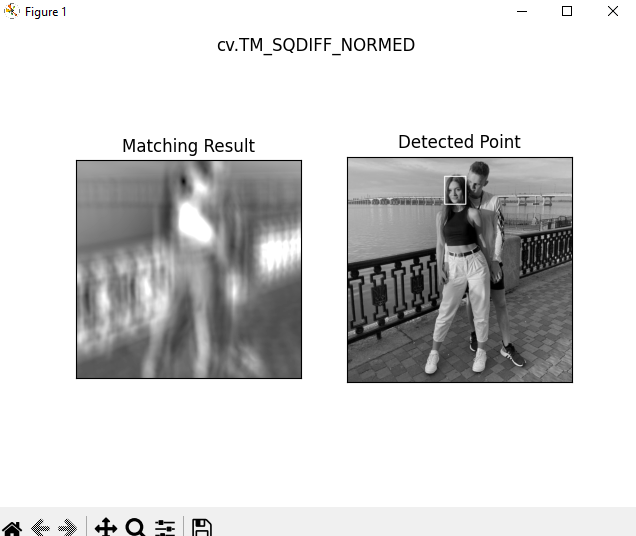


Рис. 19. Результат виконання програми

Зображення, що містить текст

Автоматично згенерований опис

Зображення, що містить текст

Автоматично згенерований опис

На мою думку, cv2.TM\_SQDIFF – найкращий метод для поставленої задачі, бо мінімальне значення дає найкращий збіг.

Завдання №6: Сегментація зображення алгоритмом водорозподілу. Лістинг програми:

import numpy as np  
import cv2  
from matplotlib import pyplot as plt  
  
img = cv2.imread('coins.jpg')  
cv2.imshow("coins", img)  
cv2.waitKey(0)  
  
gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
ret, thresh = cv2.threshold(gray, 0, 255, cv2.THRESH\_BINARY\_INV + cv2.THRESH\_OTSU)  
cv2.imshow("coins bin ", thresh)  
cv2.waitKey(0)  
  
# видалення шуму  
kernel = np.ones((3, 3), np.uint8)  
opening = cv2.morphologyEx(thresh, cv2.MORPH\_OPEN, kernel, iterations=2)  
# певна фонова область  
sure\_bg = cv2.dilate(opening, kernel, iterations=3)  
# Пошук впевненої області переднього плану  
dist\_transform = cv2.distanceTransform(opening, cv2.DIST\_L2, 5)  
ret, sure\_fg = cv2.threshold(dist\_transform, 0.7 \* dist\_transform.max(), 255, 0)  
# Пошук невідомого регіону  
sure\_fg = np.uint8(sure\_fg)  
unknown = cv2.subtract(sure\_bg, sure\_fg)  
cv2.imshow("coins ", opening)  
cv2.waitKey(0)  
  
# Маркування міток  
ret, markers = cv2.connectedComponents(sure\_fg)  
# Додайте один до всіх міток, щоб впевнений фон був не 0, а 1  
markers = markers + 1  
# Тепер позначте область невідомого нулем  
markers[unknown == 255] = 0  
  
markers = cv2.watershed(img, markers)  
img[markers == -1] = [255, 0, 0]  
cv2.imshow("coins\_markers", img)  
cv2.waitKey(0)

Зображення, що містить у приміщенні, кілька

Автоматично згенерований опис

Рис. 20. Результат виконання програми

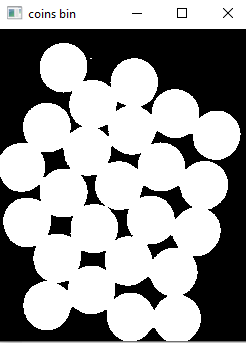


Рис. 21. Результат виконання програми

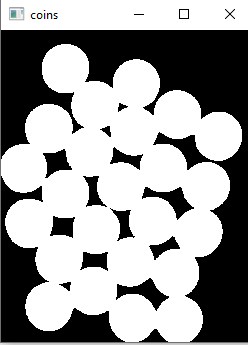


Рис. 22. Результат виконання програми

Зображення, що містить текст

Автоматично згенерований опис

Рис. 23. Результат виконання програми

Після виконання програми для більшості монет було правильно відсегментовано області, але проблеми виникли для областей, де монети торкаються одне одного. Через що певні області були не зовсім валідно визначені.

Завдання №7: Сегментація зображення.

Лістинг програми:

import cv2  
import numpy as np  
from scipy import ndimage as ndi  
from skimage.feature import peak\_local\_max  
from skimage.segmentation import watershed  
import matplotlib.pyplot as plt  
  
img = cv2.imread('coins\_2.JPG')  
gray = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)  
filtro = cv2.pyrMeanShiftFiltering(img, 20, 40)  
gray = cv2.cvtColor(filtro, cv2.COLOR\_BGR2GRAY)  
\_, thresh = cv2.threshold(gray, 0, 255, cv2.THRESH\_BINARY\_INV | cv2.THRESH\_OTSU)  
  
contornos, \_ = cv2.findContours(thresh, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)  
buracos = []  
for con in contornos:  
 area = cv2.contourArea(con)  
 if area < 1000:  
 buracos.append(con)  
cv2.drawContours(thresh, buracos, -1, 255, -1)  
  
dist = ndi.distance\_transform\_edt(thresh)  
dist\_visual = dist.copy()  
  
local\_max = peak\_local\_max(dist, indices=False, min\_distance=20, labels=thresh)  
  
markers = ndi.label(local\_max, structure=np.ones((3, 3)))[0]  
  
labels = watershed(-dist, markers, mask=thresh)  
titulos = ['Original image', 'Binary Image', 'Distance Transform', 'Watershed']  
imagens = [img, thresh, dist\_visual, labels]  
fig = plt.gcf()  
fig.set\_size\_inches(16, 12)  
for i in range(4):  
 plt.subplot(2, 2, i + 1)  
 if i == 3:  
 cmap = "jet"  
 else:  
 cmap = "orange"  
 plt.imshow(imagens[i], cmap)  
 plt.title(titulos[i])  
 plt.xticks([]), plt.yticks([])  
plt.show()

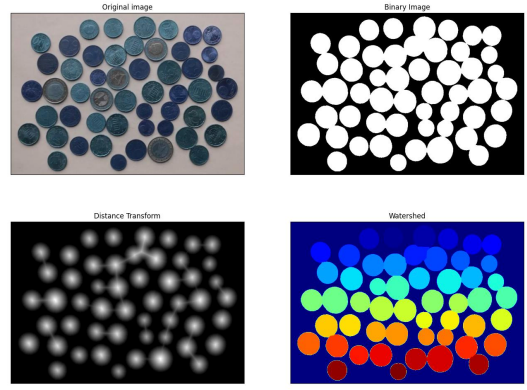


Рис. 24. Результат виконання програми

Програма показала валідний результат.

Посилання на GitHub: https://github.com/pvoitko/II

**Висновки:** Після виконання лаби я дослідив оброблення зображення за допомогою бібліотеки OpenCV, використовуючи спеціалізовані бібліотеки та мову програмування Python. Було розглянуто сегментацію зображення алгоритмом водорозподілу. Було проаналізовано розпізнавання об’єктів на зображенні за допомогою методів зіставлення шаблонів