**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 2.1.** Завантаження зображень та відео в OpenCV

Лістинг програми:

import cv2  
  
frameWidth = 640  
frameHeight = 480  
cap = cv2.VideoCapture(0)  
cap.set(3, frameWidth)  
cap.set(4, frameHeight)  
cap.set(10, 150)  
while True:  
 success, img = cap.read()  
 cv2.imshow("Result", img)  
 if cv2.waitKey(1) & 0xFF == ord('q'):  
 break  
  
img = cv2.imread("dubinchenko.jpg")  
cv2.imshow("dubinchenko", img)  
cv2.waitKey(0)

Результат виконання програми:



Рис. 7.1. Результат виконання програми

**Завдання 2.2.** Дослідження перетворень зображення

Лістинг програми:

import cv2  
import numpy as np  
  
img = cv2.imread("dubinchenko.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)

Результат виконання програми:

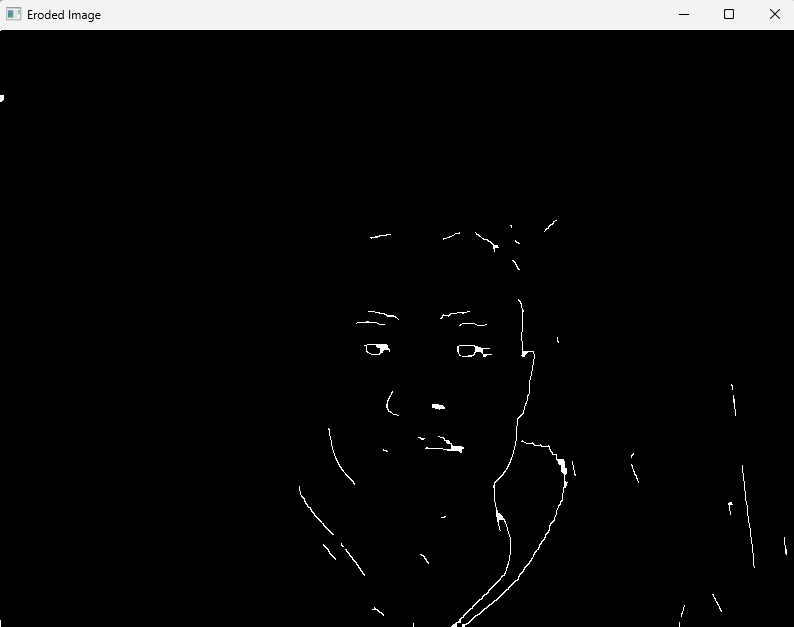


Рис. 7.2. Результат виконання програми

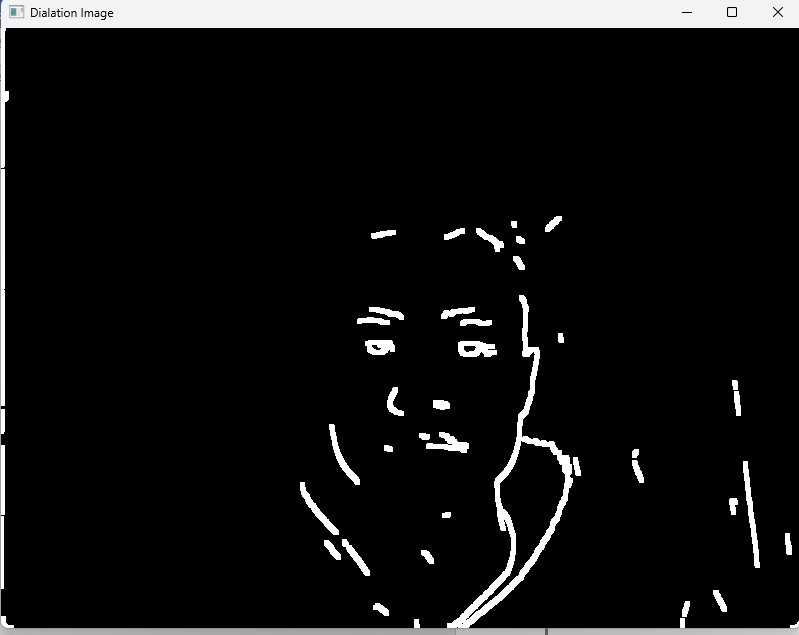


Рис. 7.3. Результат виконання програми

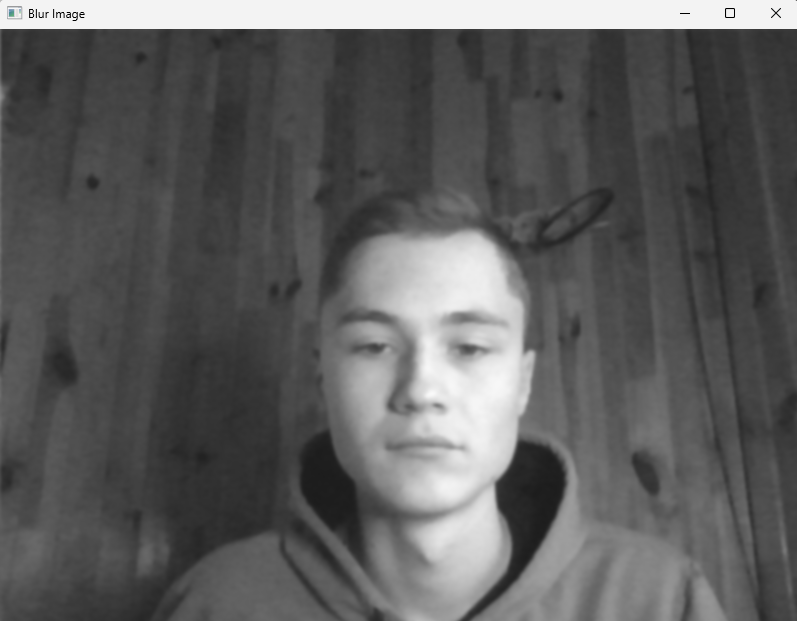


Рис. 7.4. Результат виконання програми



Рис. 7.5. Результат виконання програми

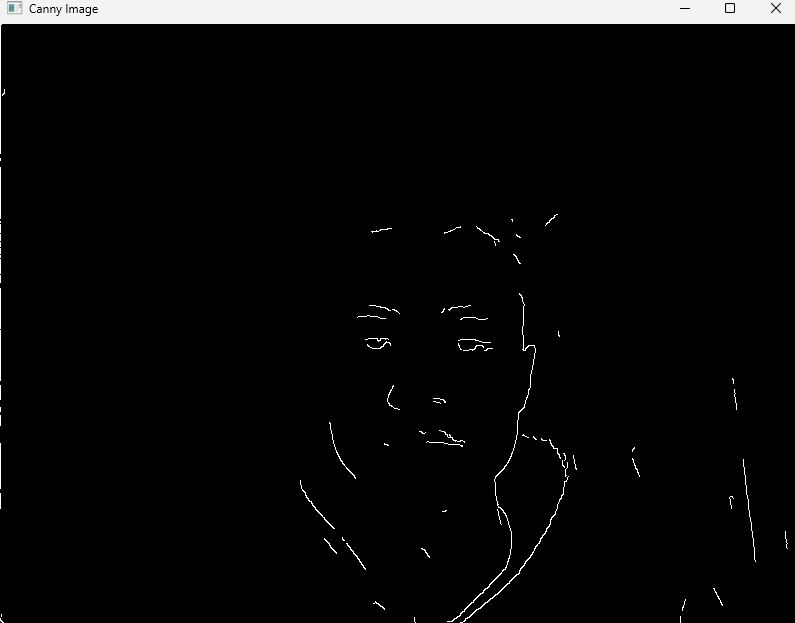


Рис. 7.6. Результат виконання програми

cvtColor - використовується для перетворення зображення з одного колірного простору на інший.

dilate приймає два вхідні параметри, один із яких є нашим вхідним зображенням; другий називається структуруючим елементом чи ядром, яке визначає характер операції. Розширення зображення Збільшує об'єкт.

erode - метод використовується для виконання розмивання зображення.

GaussianBlur - розмите зображення.

Canny - використовується для визначення країв зображення.

**Завдання 2.3.** Вирізання частини зображення

Лістинг програми:

import cv2  
import numpy as np  
  
img = cv2.imread("shapes.png")  
print(img.shape)  
  
imgResize = cv2.resize(img, (1000, 500))  
print(imgResize.shape)  
  
imgCropped = img[46:119, 352:495]  
  
cv2.imshow("Image", img)  
cv2.imshow("Image Resize", imgResize)  
cv2.imshow("Image Cropped", imgCropped)  
cv2.waitKey(0)

Результат виконання програми:

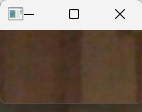


Рис. 7.7. Результат виконання програми

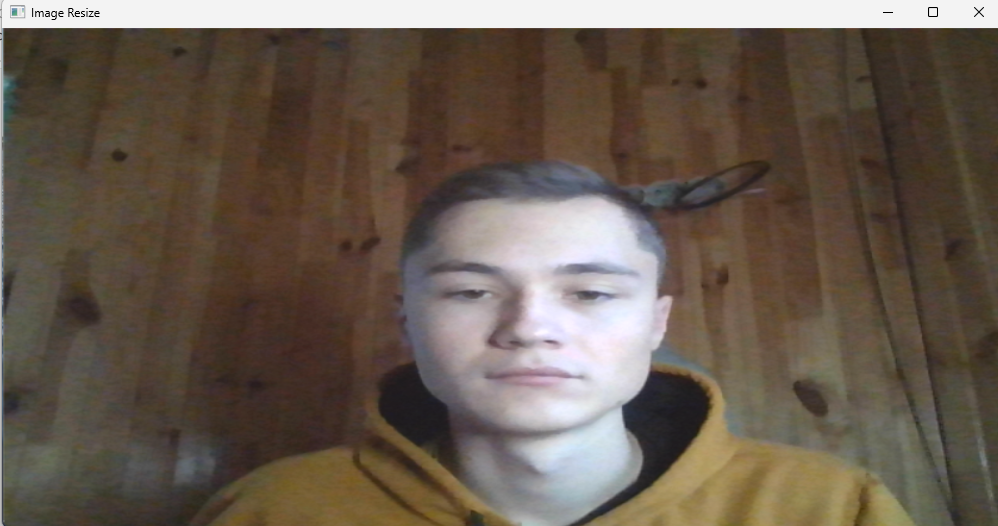


Рис. 7.8. Результат виконання програми

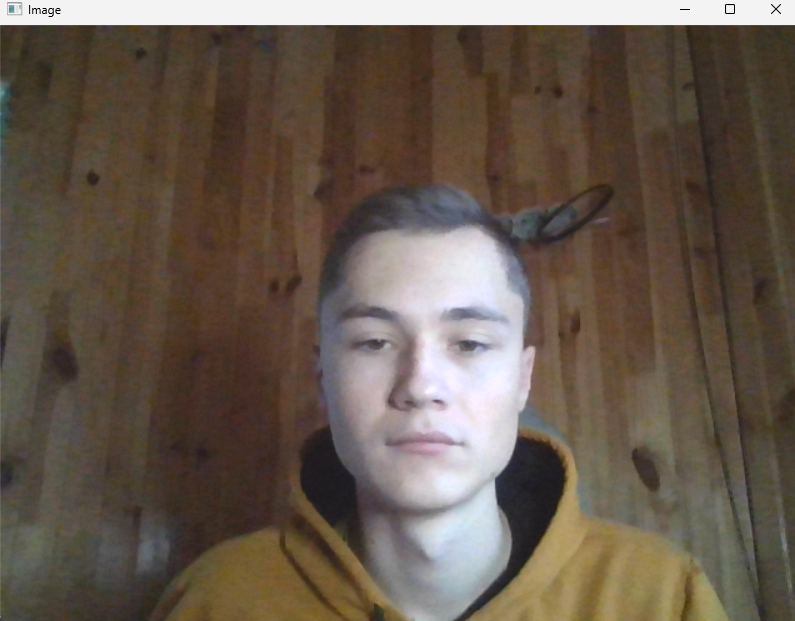


Рис. 7.9. Результат виконання програми

**Завдання 2.4.** Розпізнавання обличчя на зображенні

Лістинг програми:

import cv2  
  
faceCascade = cv2.CascadeClassifier("haarcascade\_frontalface\_default.xml")  
img = cv2.imread('dubinchenko.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)

Результат виконання програми:

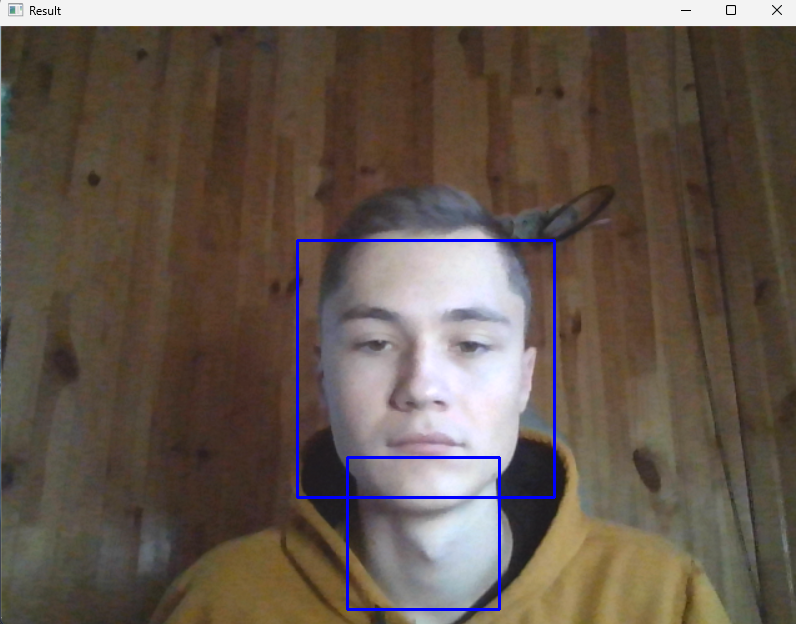


Рис. 7.10. Результат виконання програми

**Завдання 2.5.** Розпізнавання об’єктів на зображенні за допомогою методів зіставлення шаблонів (Template Matching)

Лістинг програми:

import cv2 as cv  
import numpy as np  
from matplotlib import pyplot as plt  
  
img = cv.imread('messi\_full.JPG', 0)  
img2 = img.copy()  
template = cv.imread('messi\_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 method in [cv.TM\_CCOEFF, cv.TM\_CCOEFF\_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()

Результат виконання програми:

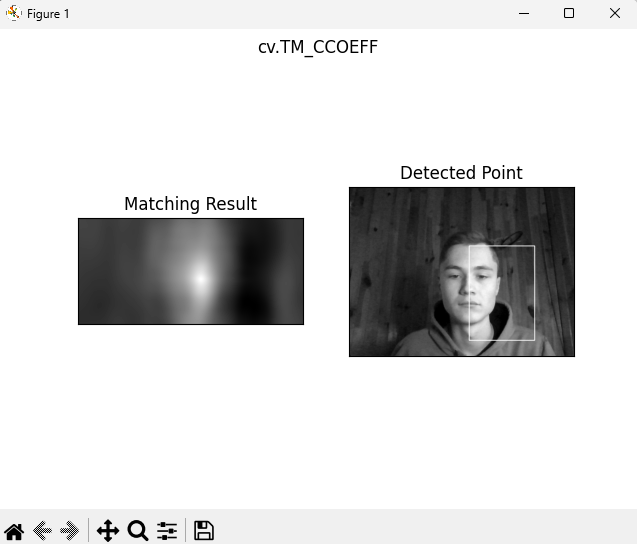


Рис. 7.11. Результат виконання програми

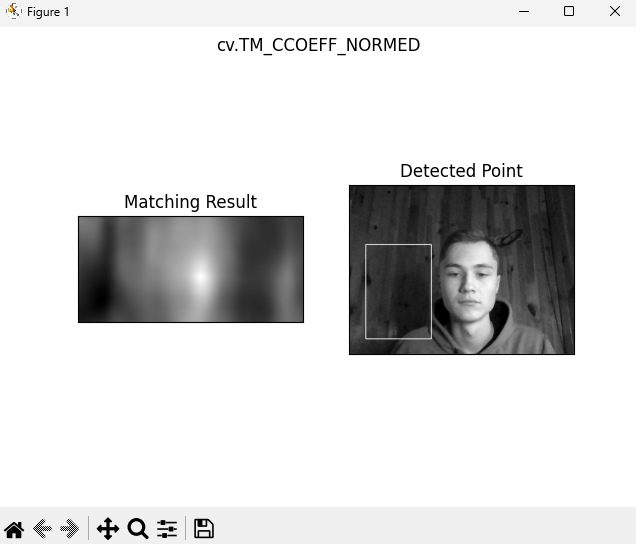


Рис. 7.12. Результат виконання програми

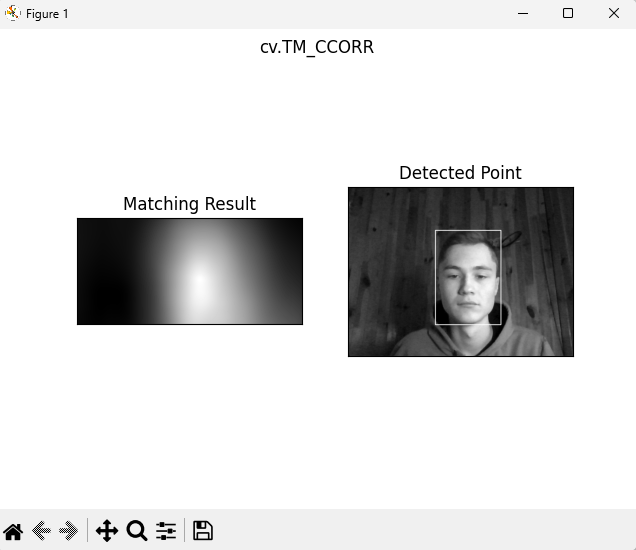


Рис. 7.13. Результат виконання програми

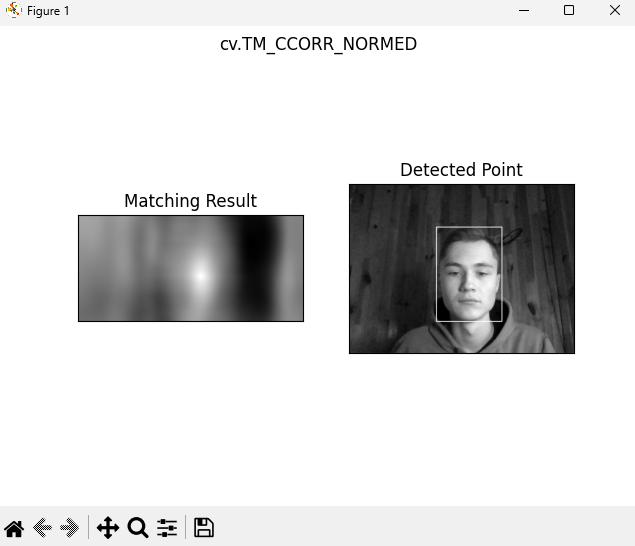


Рис. 7.14. Результат виконання програми

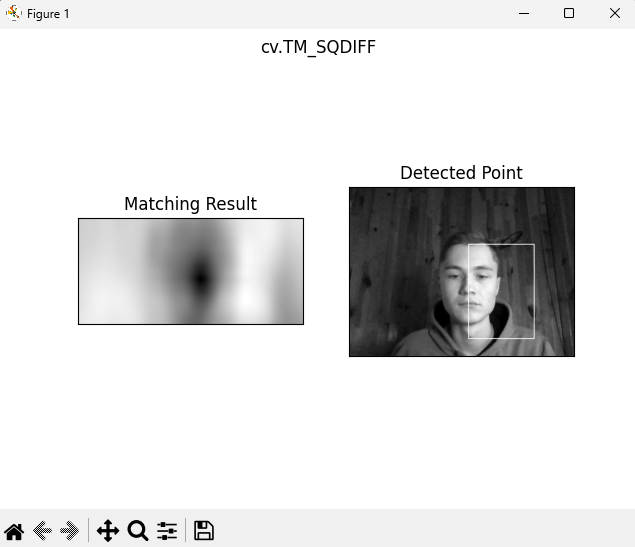


Рис. 7.15. Результат виконання програми

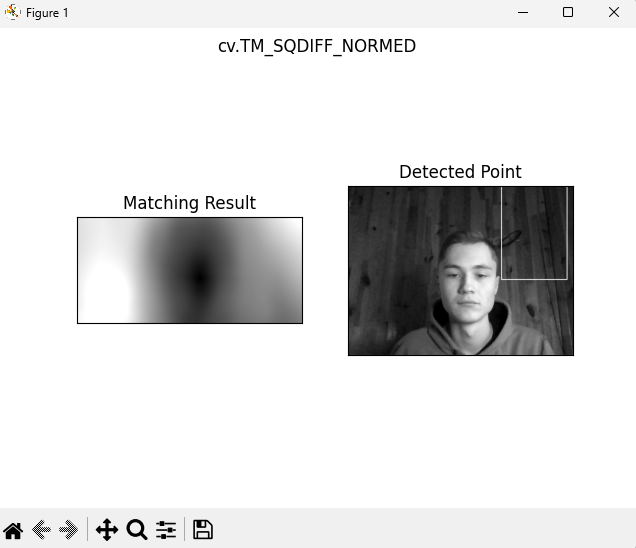


Рис. 7.16. Результат виконання програми

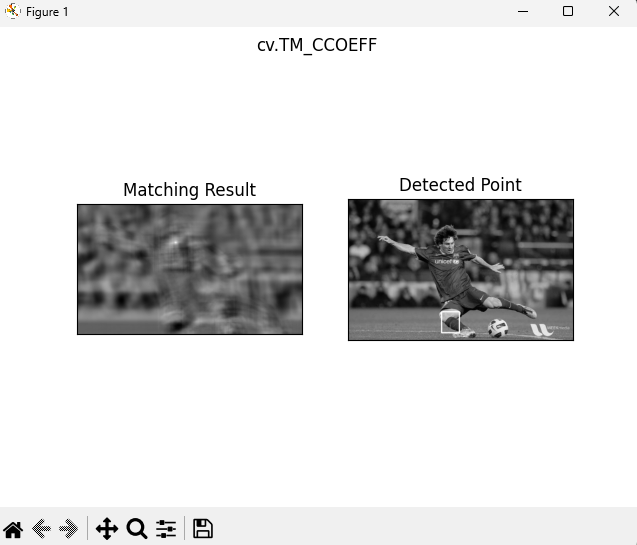


Рис. 7.17. Результат виконання програми

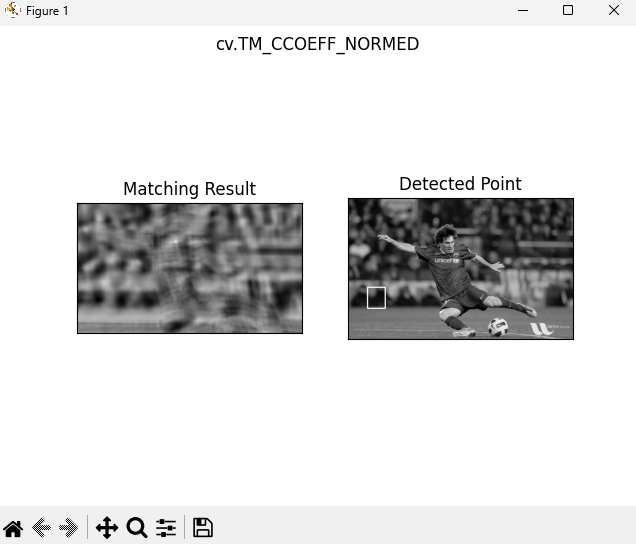


Рис. 7.18. Результат виконання програми



Рис. 7.19. Результат виконання програми

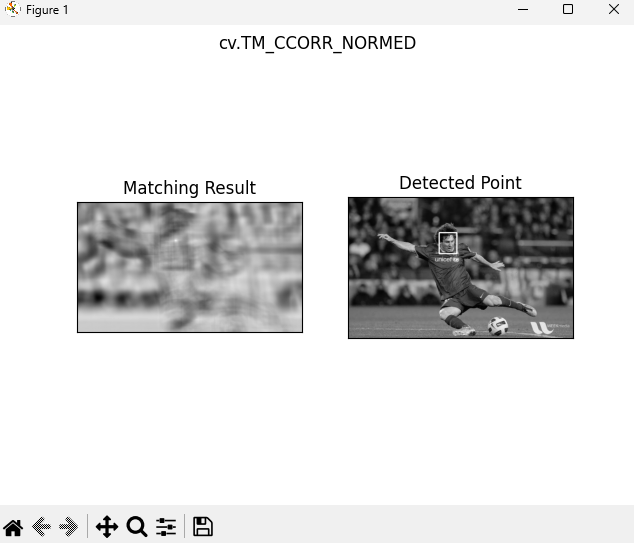


Рис. 7.20. Результат виконання програми

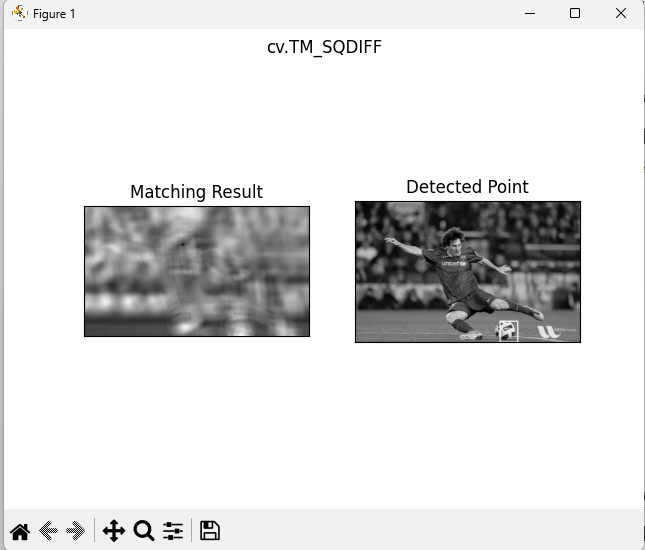


Рис. 7.21. Результат виконання програми

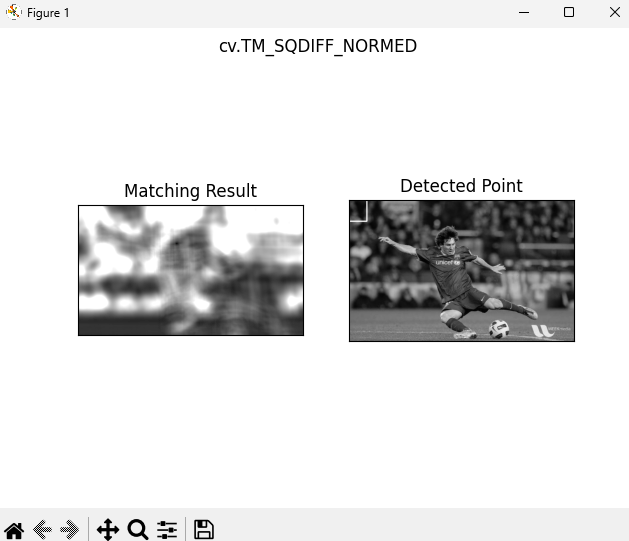


Рис. 7.22. Результат виконання програми

**Завдання 2.6.** 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)

Результат виконання завдання:



Рис. 7.23. Результат виконання програми

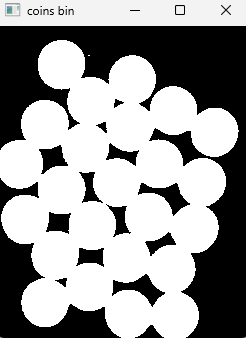


Рис. 7.24. Результат виконання програми

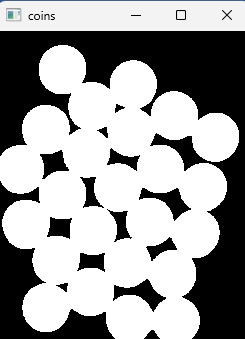


Рис. 7.25. Результат виконання програми



Рис. 7.26. Результат виконання програми

**2.7.** Сегментація зображення

Лістинг програми:

import numpy as np  
import cv2  
from matplotlib import pyplot as plt  
  
img = cv2.imread('coins\_2.JPG')  
cv2.imshow("coins", img)  
cv2.waitKey(0)  
  
# Конвертуємо зображення в відтінки сірого  
gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
  
# Порогова обробка  
ret, thresh = cv2.threshold(gray, 0, 255, cv2.THRESH\_BINARY\_INV + cv2.THRESH\_OTSU)  
  
# Видаляємо шум  
kernel = np.ones((3, 3), np.uint8)  
opening = cv2.morphologyEx(thresh, cv2.MORPH\_OPEN, kernel, iterations=2)  
  
# Визначаємо фонову область  
sure\_bg = cv2.dilate(opening, kernel, iterations=3)  
  
# Відстанційне перетворення  
dist\_transform = cv2.distanceTransform(opening, cv2.DIST\_L2, 5)  
ret, sure\_fg = cv2.threshold(dist\_transform, 0.7 \* dist\_transform.max(), 255, 0)  
  
# Визначаємо невідому область  
sure\_fg = np.uint8(sure\_fg)  
unknown = cv2.subtract(sure\_bg, sure\_fg)  
  
# Маркування міток  
ret, markers = cv2.connectedComponents(sure\_fg)  
  
# Додаємо кожній маркованій області свій унікальний колір  
colors = []  
for i in range(1, ret + 1):  
 colors.append(np.random.randint(0, 255, 3))  
  
# Розфарбовуємо зображення відповідно до маркування  
colored\_markers = np.zeros\_like(img)  
for label in range(1, ret + 1):  
 colored\_markers[markers == label] = colors[label - 1]  
  
# Додаємо невідому область червоним кольором  
colored\_markers[unknown == 255] = [0, 0, 255]  
  
# Відображення результату  
cv2.imshow("Coins Segmentation", colored\_markers)  
cv2.waitKey(0)  
cv2.destroyAllWindows()  
  
# 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)

Результат виконання програми:



Рис. 7.27. Результат виконання програми

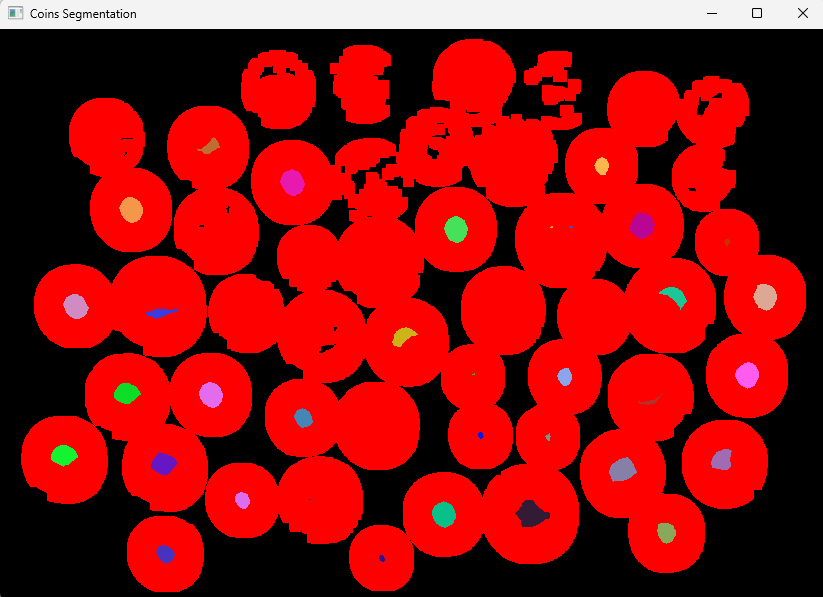


Рис. 7.28. Результат виконання програми



Рис. 7.29. Результат виконання програми

Посилання на GitHub: <https://github.com/BogdanStelmah/Basics-of-AI_labs>

**Висновок:** На даній лабораторній роботі мивикористовуючи спеціалізовані бібліотеки та мову програмування Python навчилися обробляти зображення за допомогою бібліотеки OpenCV.