**ЛАБОРАТОРНА РОБОТА №8**

**ДОСЛІДЖЕННЯ МЕТОДІВ КОМП’ЮТЕРНОГО ЗОРУ**

**Мета роботи:**

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 коду:

import cv2

# LOAD AN IMAGE USING 'IMREAD'

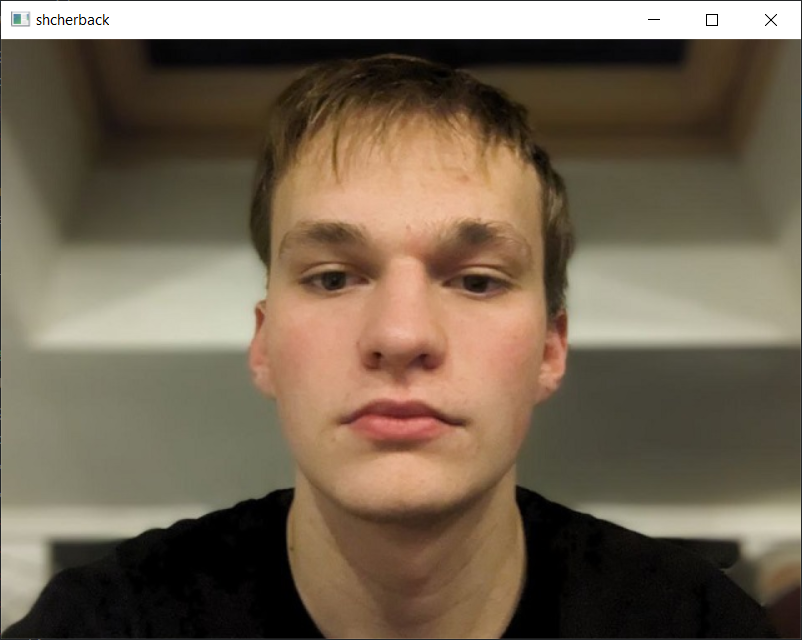
img = cv2.imread("shcherback.jpg")

# DISPLAY

cv2.imshow("shcherback",img)

cv2.waitKey(0)

Результат виконання програми:



Висновок: За допомогою бібліотеки cv2 ми відобразили зображення.

**Завдання 2.2. Дослідження перетворень зображення**

Лістинг коду:

import cv2

import numpy as np

img = cv2.imread("shcherback.jpg")

kernel = np.ones((5, 5), np.uint8)

# Перетворення зображення у відтінки сірого

imgGray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

# Застосування гауссівського розмиття

imgBlur = cv2.GaussianBlur(imgGray, (7, 7), 0)

# Застосування алгоритму Canny для виявлення границь

imgCanny = cv2.Canny(img, 150, 200)

# Збільшення областей границь на зображенні за допомогою операції діляції

imgDialation = cv2.dilate(imgCanny, kernel, iterations=1)

# Зменшення областей границь за допомогою операції ерозії

imgEroded = cv2.erode(imgDialation, kernel, iterations=1)

cv2.imshow("Gray Image", imgGray)

cv2.imshow("Blur Image", imgBlur)

cv2.imshow("Canny Image", imgCanny)

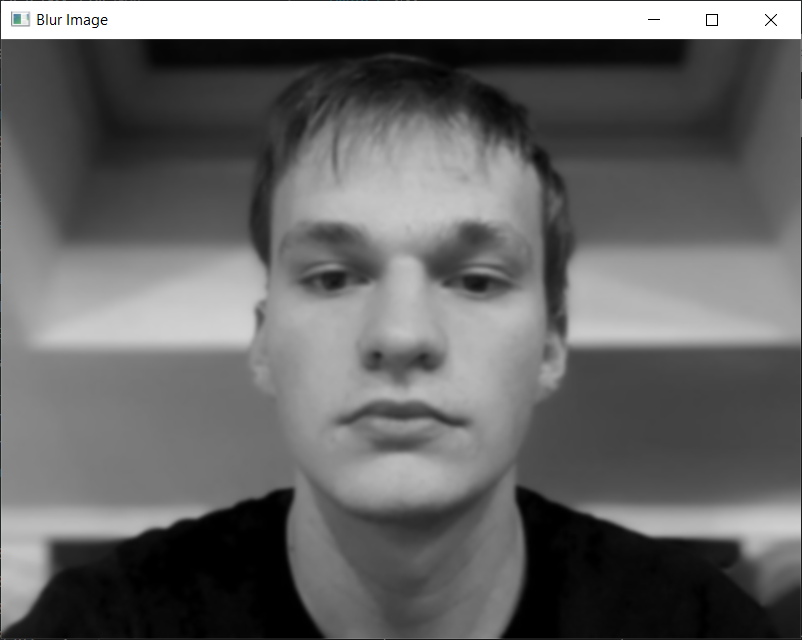
cv2.imshow("Dialation Image", imgDialation)

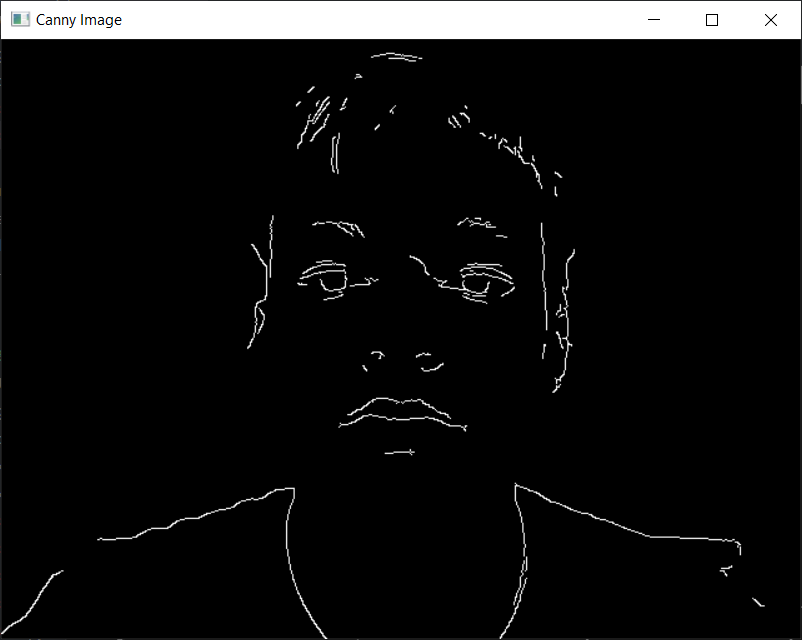
cv2.imshow("Eroded Image", imgEroded)

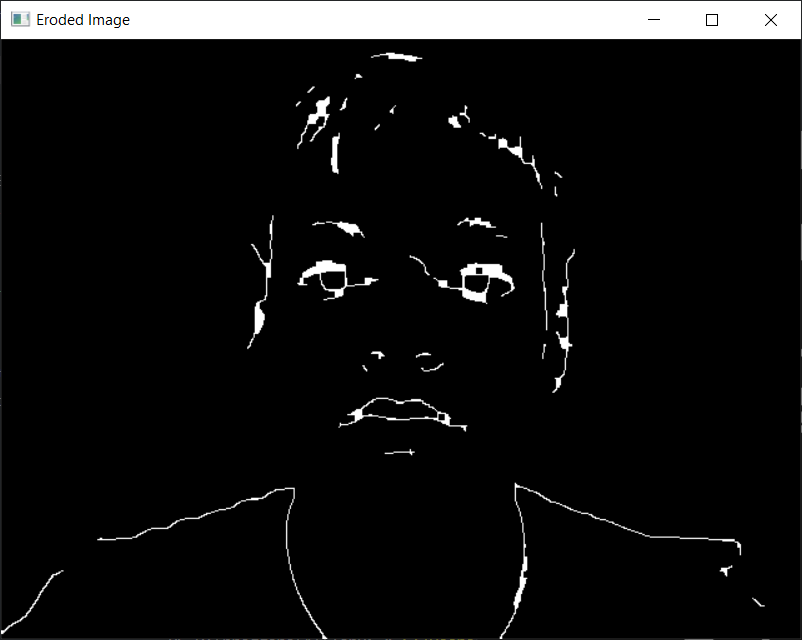
cv2.waitKey(0)

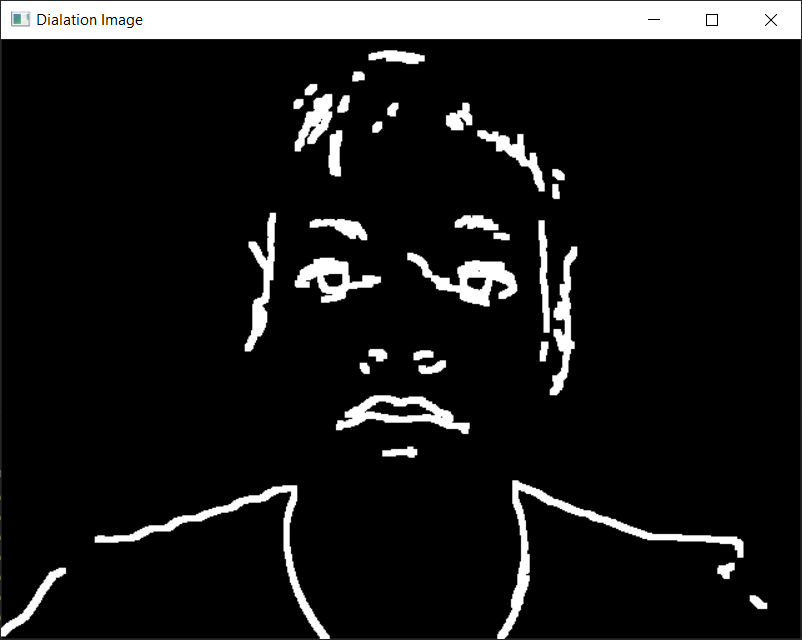
Результат виконання програми:











**Завдання 2.3. Вирізання частини зображення**

Лістинг коду:

import cv2

img = cv2.imread("shcherback.jpg")

print(img.shape)

imgResize = cv2.resize(img, (1000, 500))

print(imgResize.shape)

imgCropped = img[46:119, 352:495]

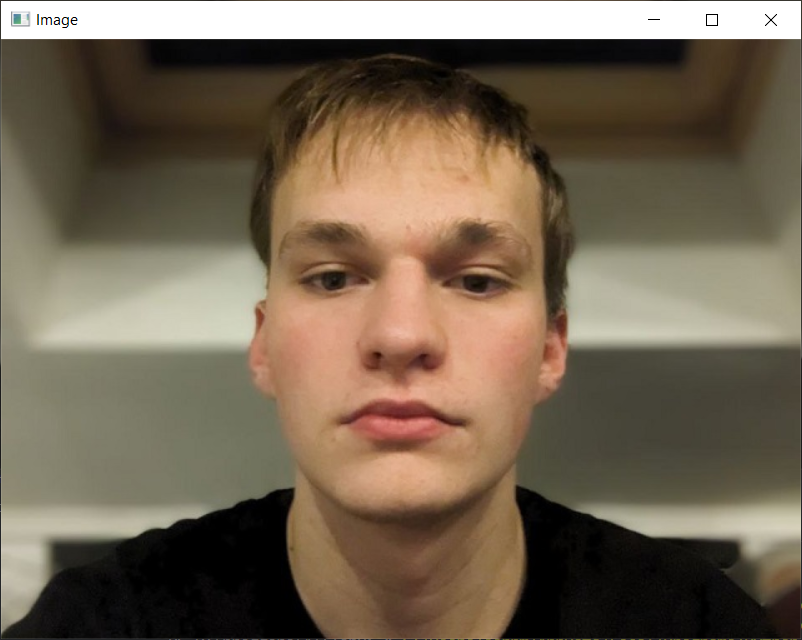
cv2.imshow("Image", img)

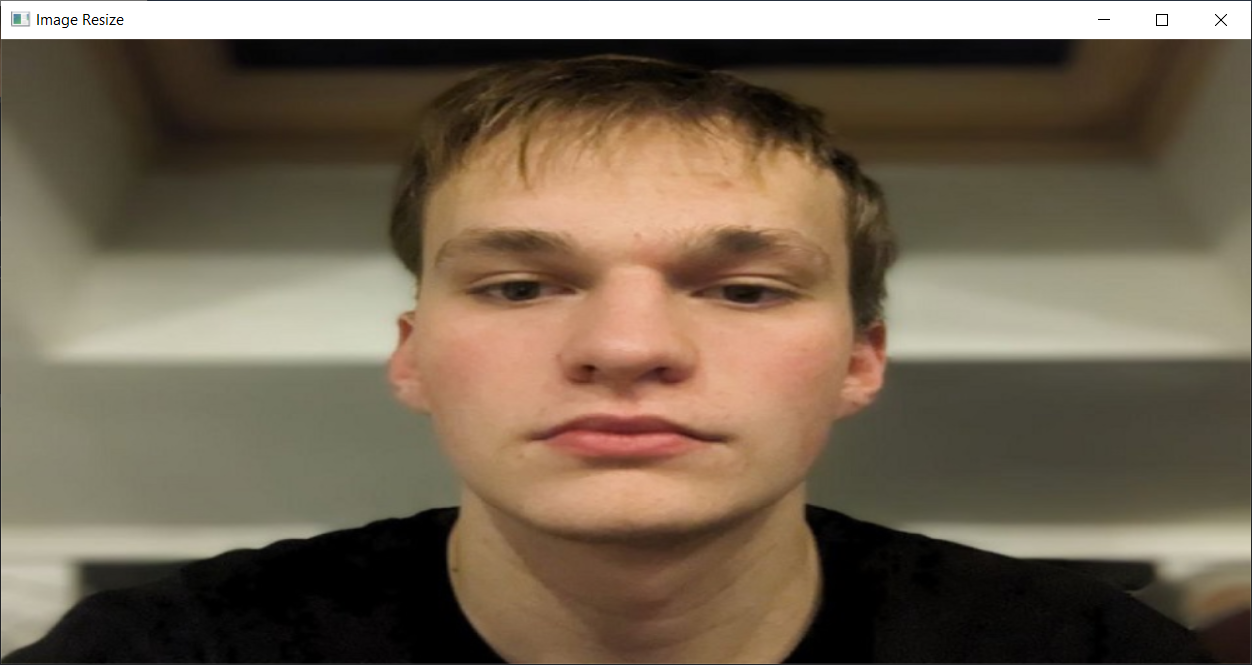
cv2.imshow("Image Resize", imgResize)

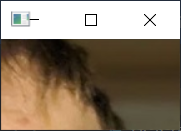
cv2.imshow("Image Cropped", imgCropped)

cv2.waitKey(0)

Результат виконання програми:







**Завдання 2.4. Розпізнавання обличчя на зображенні**

Лістинг коду:

import cv2

faceCascade = cv2.CascadeClassifier("haarcascade\_frontalface\_default.xml")

img = cv2.imread('shcherback.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:



Висновок:

Спочатку відбувається перетворення зображення в відтінки сірого, а потім використовується класифікатор каскаду Haar для визначення координат обличчя на зображенні. Знайдені обличчя помічається синім прямокутником.

**Завдання 2.5. Розпізнавання об’єктів на зображенні за допомогою**

**методів зіставлення шаблонів (Template Matching)**

Лістинг коду:

import cv2 as cv

from matplotlib import pyplot as plt

img = cv.imread('shcherback.jpg', 0)

import cv2 as cv

from matplotlib import pyplot as plt

img = cv.imread('shcherback.jpg', 0)

img2 = img.copy()

template = cv.imread('shcherback-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([])

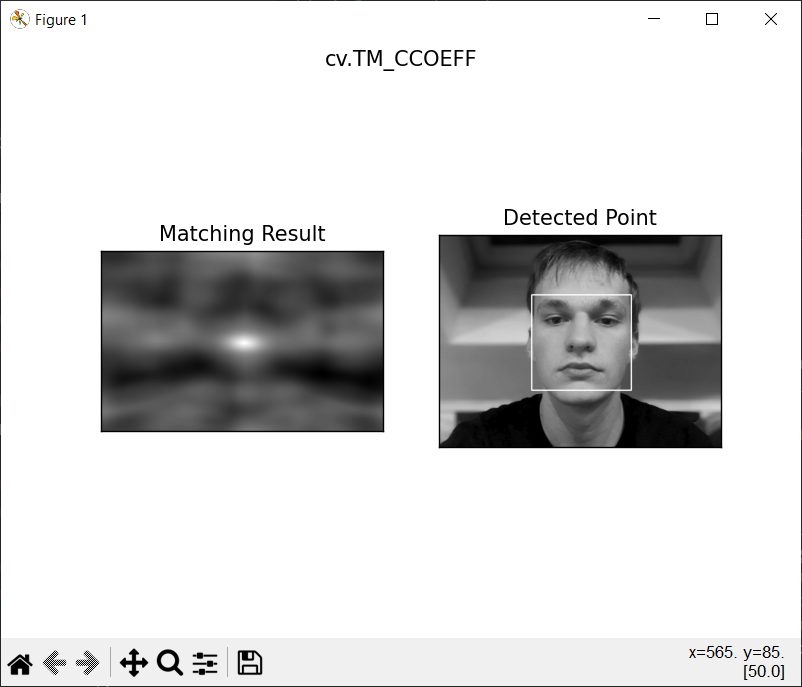
    plt.subplot(122), plt.imshow(img, cmap='gray')

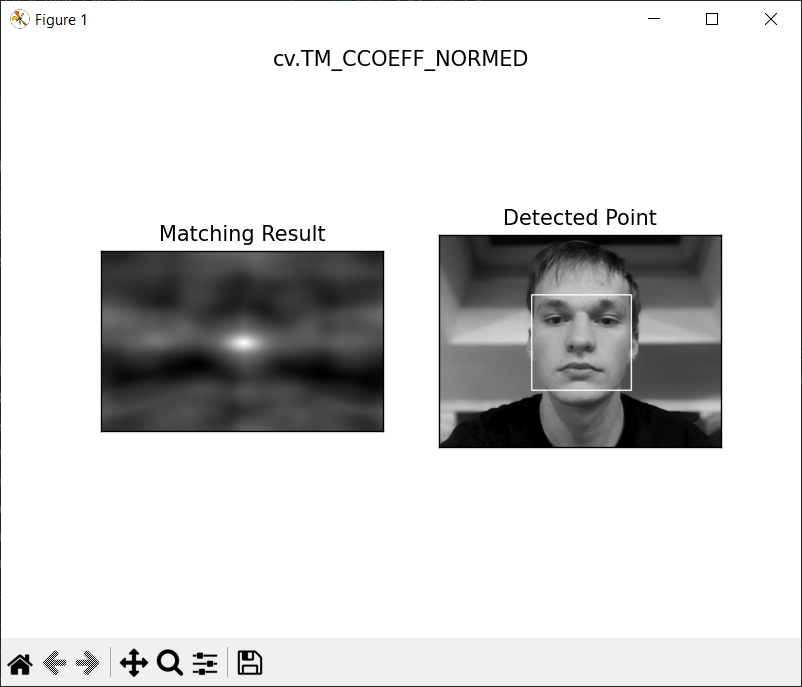
    plt.title('Detected Point'), plt.xticks([]), plt.yticks([])

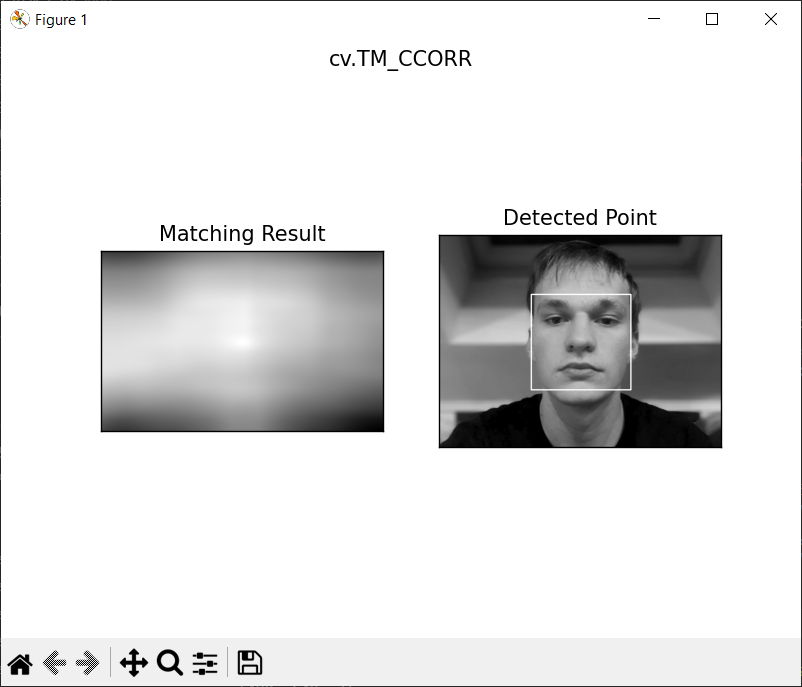
    plt.suptitle(meth)

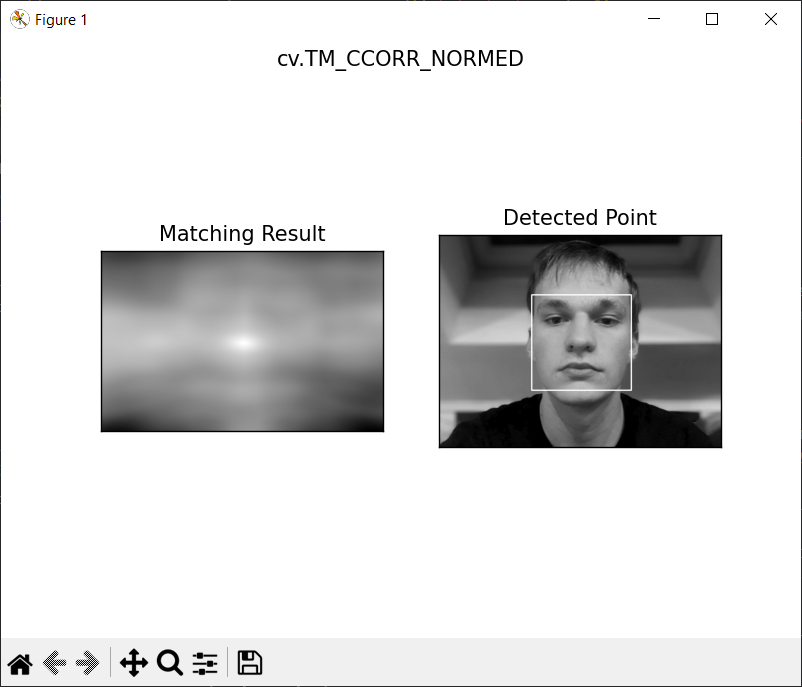
    plt.show()

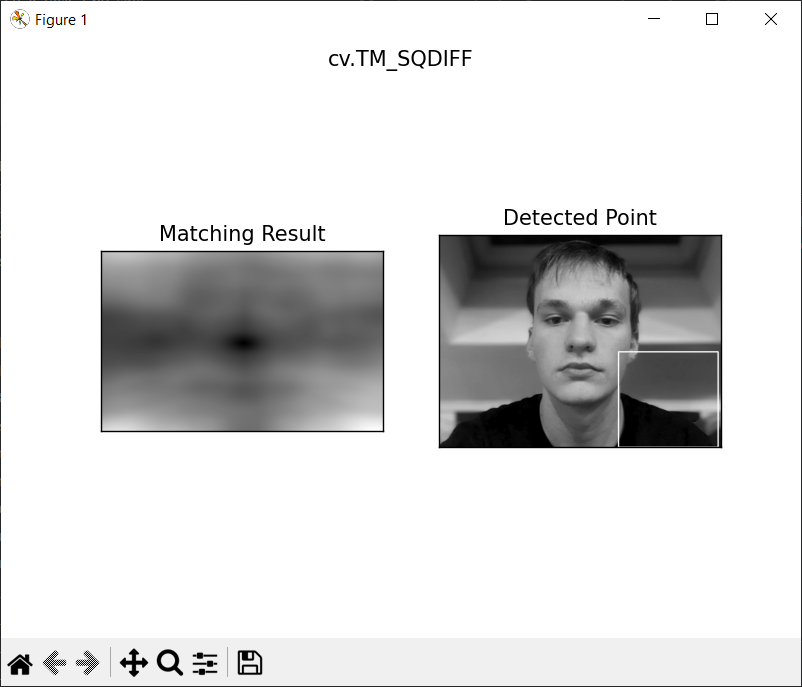
Результат виконання програми:

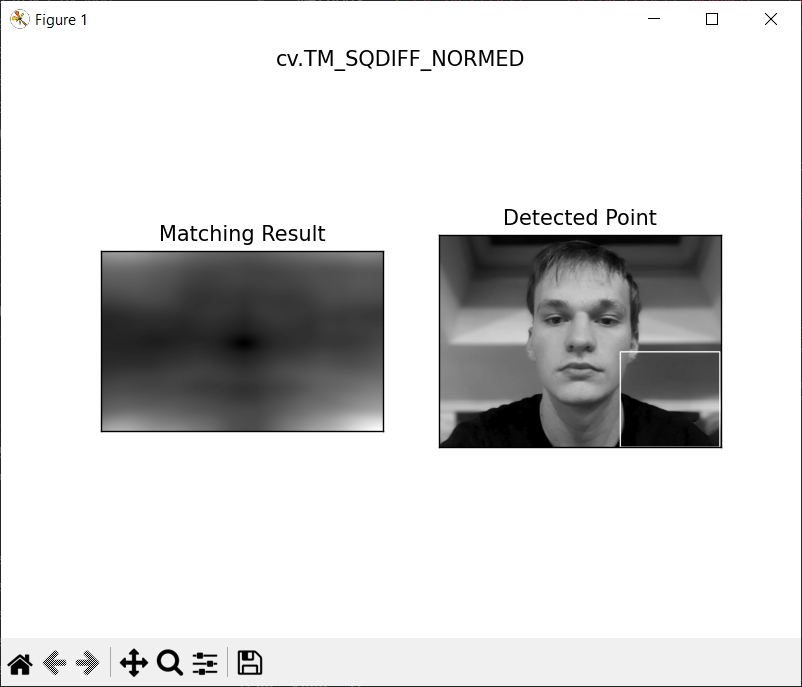












Висновок:

cv.TM\_CCOEFF: Коефіцієнт кореляції. Більше значення вказує на більш точне співпадіння.

cv.TM\_CCOEFF\_NORMED: Нормалізований коефіцієнт кореляції. Значення нормалізовані в межах від -1 до 1, де 1 - ідеальне співпадіння.

cv.TM\_CCORR: Коефіцієнт кореляції. Більше значення вказує на більш точне співпадіння, але цей метод використовується для яскравих областей.

cv.TM\_CCORR\_NORMED: Нормалізований коефіцієнт кореляції. Значення нормалізовані в межах від 0 до 1, де 1 - ідеальне співпадіння.

cv.TM\_SQDIFF: Квадрат різниці. Менше значення вказує на більш точне співпадіння.

cv.TM\_SQDIFF\_NORMED: Нормалізований квадрат різниці. Значення нормалізовані в межах від 0 до 1, де 0 - ідеальне співпадіння.

На мою думку обидва TM\_CCORR та cv.TM\_CCOEFF добре впорадися з задачею.

**Завдання 2.6. Сегментація зображення алгоритмом водорозподілу**

Лістинг коду:

import numpy as np

import cv2

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]

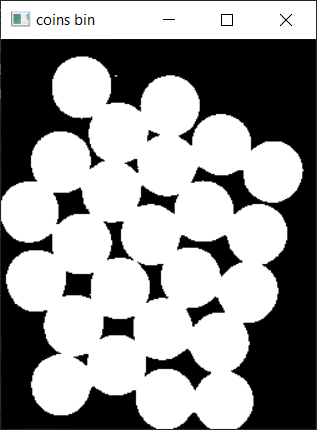
cv2.imshow("coins\_markers", img)

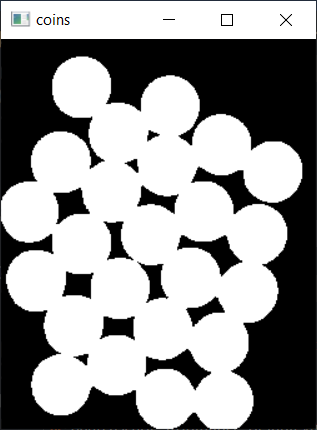
cv2.waitKey(0)

Результат виконання програми:



Coins.jpg







Висновок:

Програма досить добре впоралася із сегментацією монет на зображенні, але області монет, де вони торкаються мають дефекти.

**Додаткове завдання 2.7. Сегментація зображення** Лістинг коду:

import numpy as np

import cv2

img = cv2.imread('coins\_2.JPG')

cv2.imshow('coins', img)

shifted = cv2.pyrMeanShiftFiltering(img, 20, 50)

gray = cv2.cvtColor(shifted, cv2.COLOR\_BGR2GRAY)

ret, thresh = cv2.threshold(gray, 0, 255, cv2.THRESH\_BINARY\_INV + cv2.THRESH\_OTSU)

# Видалення шуму

kernel = np.ones((3, 3), np.uint8)

opening = cv2.morphologyEx(thresh, cv2.MORPH\_OPEN, kernel, iterations=3)

# Певна фонова область

sure\_bg = cv2.dilate(opening, kernel, iterations=3)

# Пошук певної області переднього плану

dist\_transform = cv2.distanceTransform(sure\_bg, cv2.DIST\_L2, 5)

# Область переднього плану

ret, sure\_fg = cv2.threshold(dist\_transform, 0.5 \* dist\_transform.max(), 255, 0)

# Пошук невідомої області

sure\_fg = np.uint8(sure\_fg)

unknown = cv2.subtract(sure\_bg, sure\_fg)

# Визначення маркерів та застосування методу watershed

ret, markers = cv2.connectedComponents(sure\_fg)

markers += 1

markers[unknown == 255] = 0

markers = cv2.watershed(img, markers)

# Розмічення областей монет різними кольорами

for label in range(2, ret + 1):

    img[markers == label] = np.random.randint(0, 255, 3)

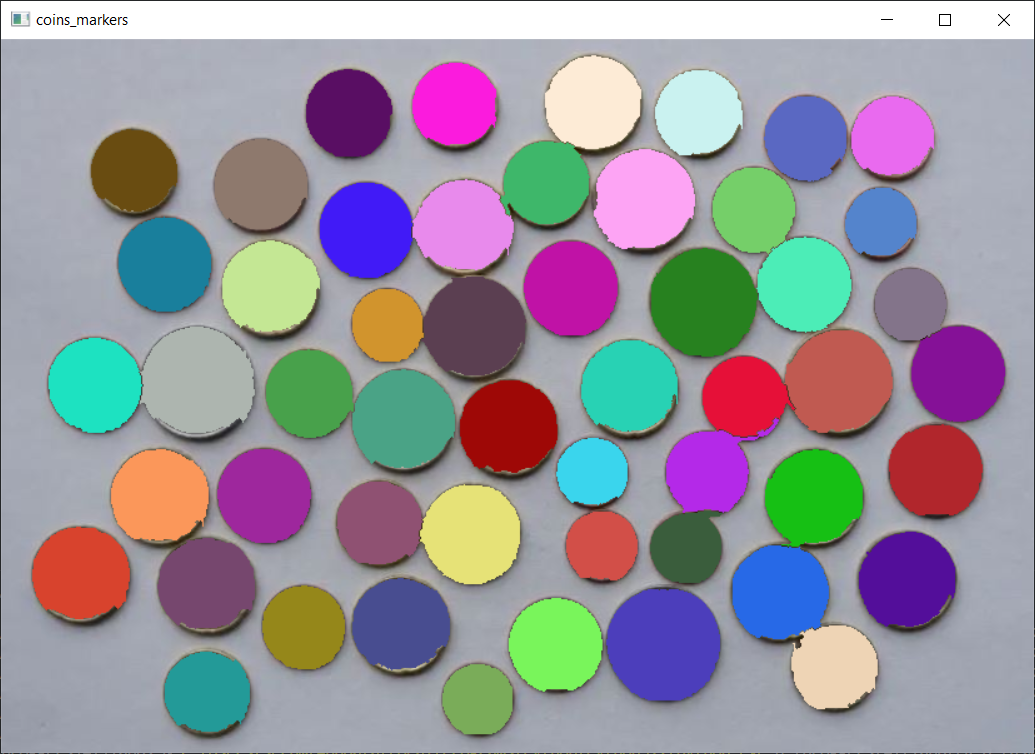
cv2.imshow("coins\_markers", img)

cv2.waitKey(0)

cv2.destroyAllWindows()

Результат виконання програми:





***Github:*** [***https://github.com/mtvi/ipz202\_Shcherback\_lab8***](https://github.com/mtvi/ipz202_Shcherback_lab8)

***Висновки:*** використовуючи спеціалізовані бібліотеки та мову програмування Python дослідили деякі типи нейронних мереж.