**ЛАБОРАТОРНА РОБОТА № 8**

**ДОСЛІДЖЕННЯ МУРАШИНИХ АЛГОРИТМІВ**

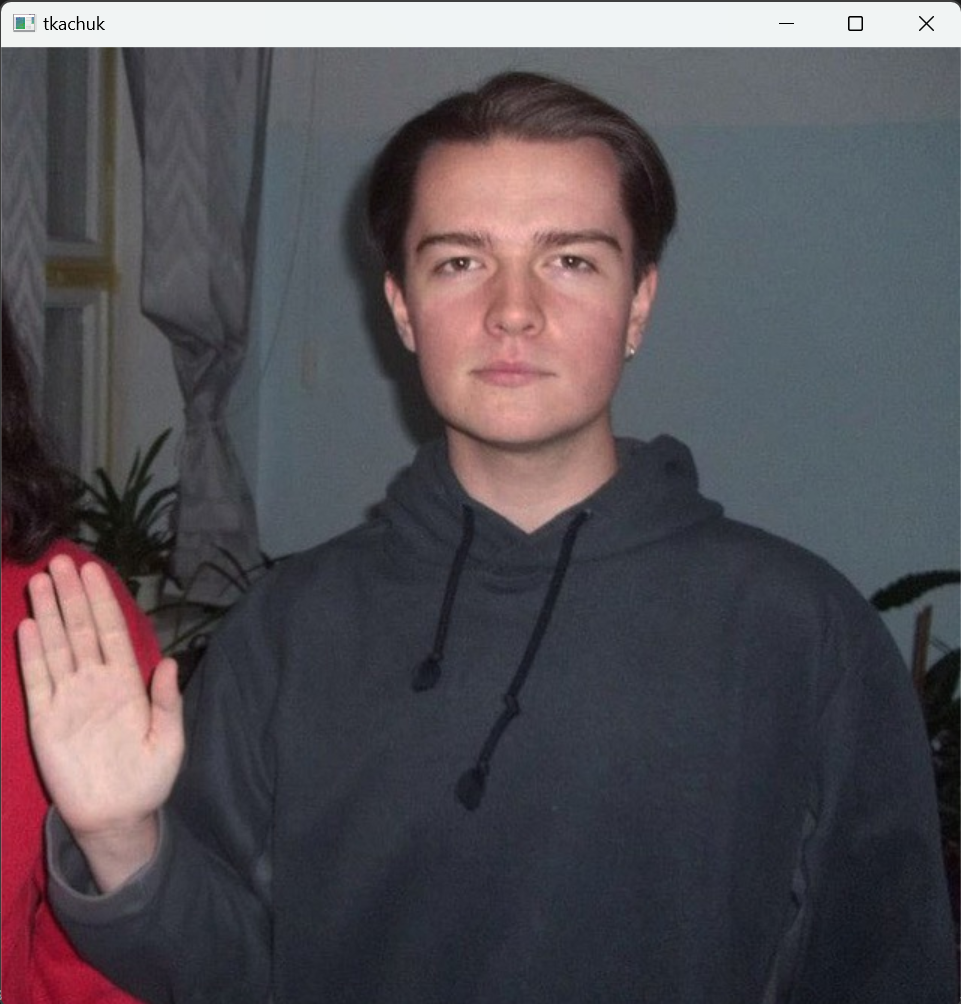
***Мета***: *використовуючи спеціалізовані бібліотеки та мову програмування Python навчитися обробляти зображення за допомогою бібліотеки OpenCV*.***.***

**GitHub:** [**https://github.com/ingaliptn/AI**](https://github.com/ingaliptn/AI)

**Хід роботи:**

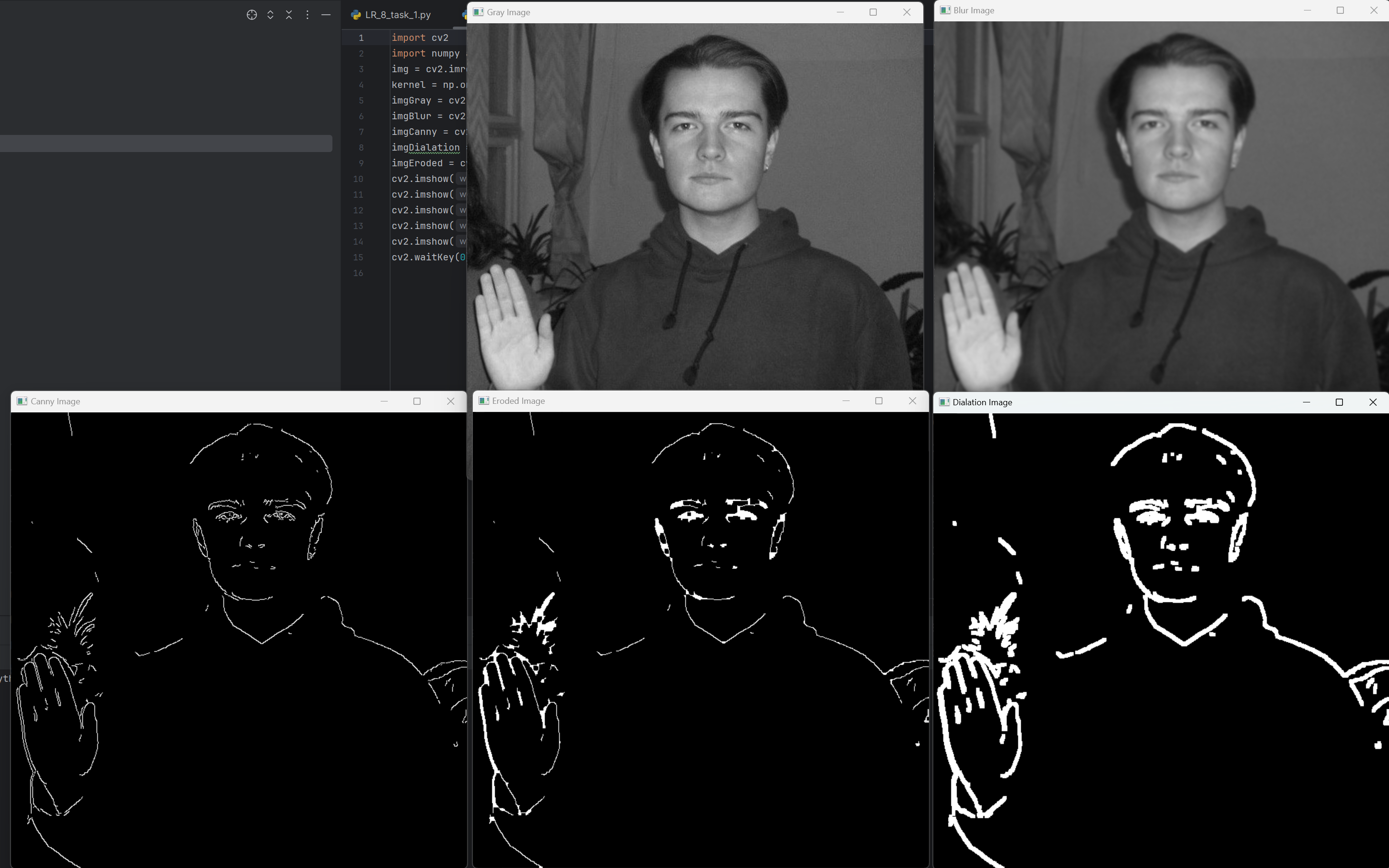
**Завдання 2.1. Завантаження зображень та відео в OpenCV**

|  |
| --- |
| import cv2 # LOAD AN IMAGE USING 'IMREAD' img = cv2.imread("tkachuk\_full.jpg") # DISPLAY cv2.imshow("tkachuk", img) cv2.waitKey(0) |

****

**Завдання 2.2. Дослідження перетворень зображення**

|  |
| --- |
| import cv2 import numpy as np img = cv2.imread("tkachuk\_full.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) |

****

**cvtColor:** Цей метод призначений для конвертації кольорових просторів у зображеннях. Наприклад, він може перетворити зображення з одного кольорового простору в інший: з RGB у відтінки сірого, з RGB у HSV і т.д. Результатом його застосування може бути змінене зображення у новому кольоровому просторі.

**GaussianBlur:** Цей метод використовується для розмиття зображення з допомогою гаусового фільтру. Він зменшує шум та деталі на зображенні, згладжує його. Результатом застосування цього методу є зображення з меншою різницею між сусідніми пікселями, що може бути корисним для подальшого аналізу зображень, наприклад, виявлення контурів або об'єктів.

**Canny:** Метод Canny використовується для виявлення границь на зображенні. Він використовує алгоритм знаходження границь, який включає кілька етапів: згладжування гаусовим фільтром, визначення градієнтів інтенсивності пікселів, підсилення та пригнічення границь, виявлення потенційних границь та їхнє припорядкування. Результатом є зображення, на якому виділені границі об'єктів або контурів.

**dilate:** Метод dilate використовується для розширення областей об'єктів на зображенні. Він робить об'єкти на зображенні більшими, збільшуючи їхню площу шляхом додавання пікселів до меж об'єктів. Результатом є зображення з розширеними областями об'єктів.

**erode:** Метод erode використовується для зменшення областей об'єктів на зображенні. Він зменшує розмір об'єктів шляхом видалення пікселів з їхніх меж. Результатом є зображення з зменшеними областями об'єктів.

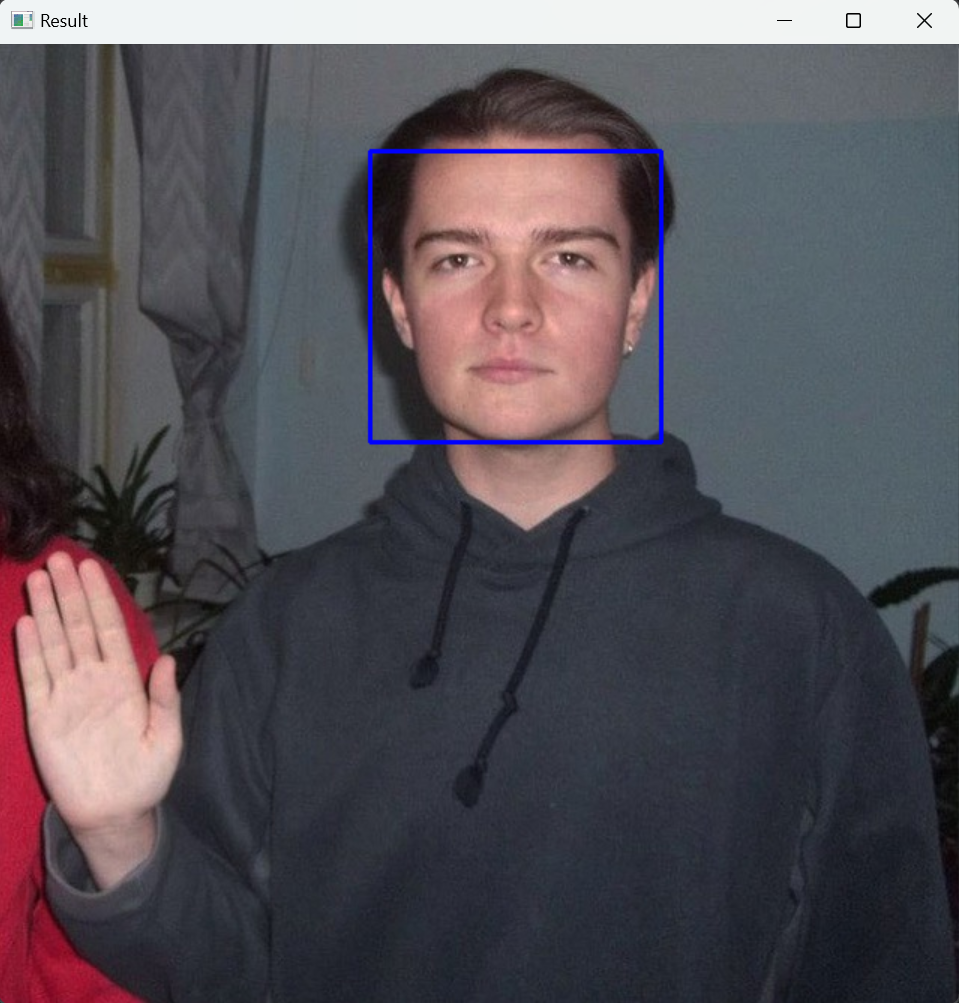
**Завдання 2.3. Вирізання частини зображення**

|  |
| --- |
| import cv2 img = cv2.imread("tkachuk\_full.jpg") print(img.shape) imgResize = cv2.resize(img, (1000, 450)) print(imgResize.shape) imgCropped = img[20:350, 250:460] cv2.imshow("Image", img) cv2.imshow("Image Cropped", imgCropped) cv2.waitKey(0) |

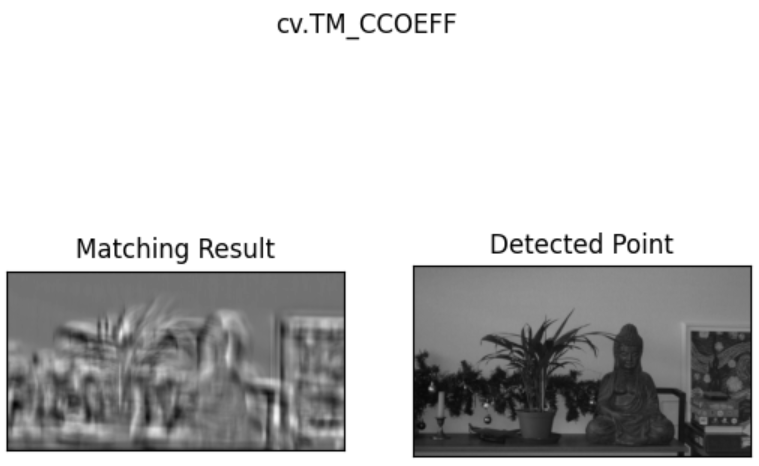
****

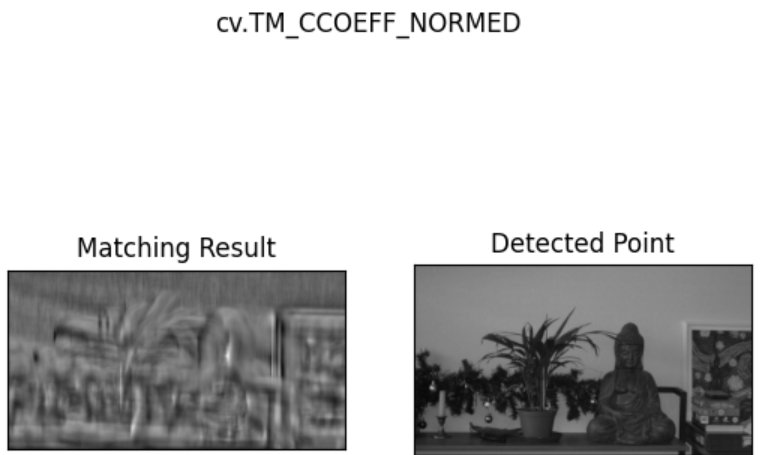
**Завдання 2.4. Розпізнавання обличчя на зображенні**

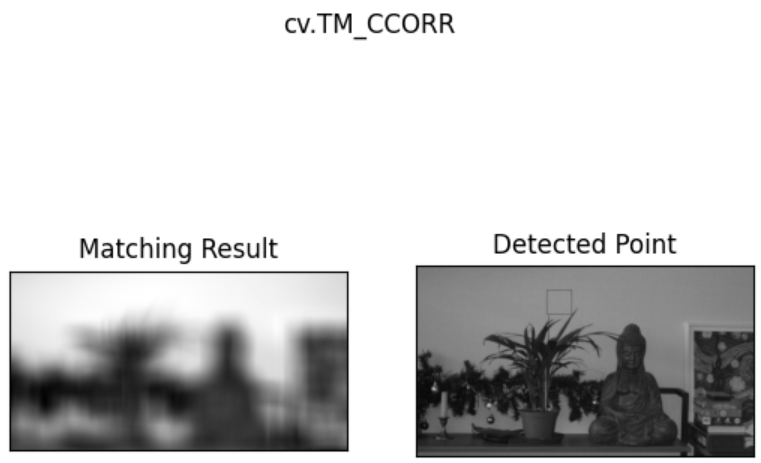
|  |
| --- |
| import cv2 faceCascade = cv2.CascadeClassifier("haarcascade\_frontalface\_default.xml") img = cv2.imread('tkachuk\_full.jpg') imgGray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY) faces = faceCascade.detectMultiScale(imgGray, 1.1, 4) for (x, y, w, h) in faces:  cv2.rectangle(img, (x, y), (x+w,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

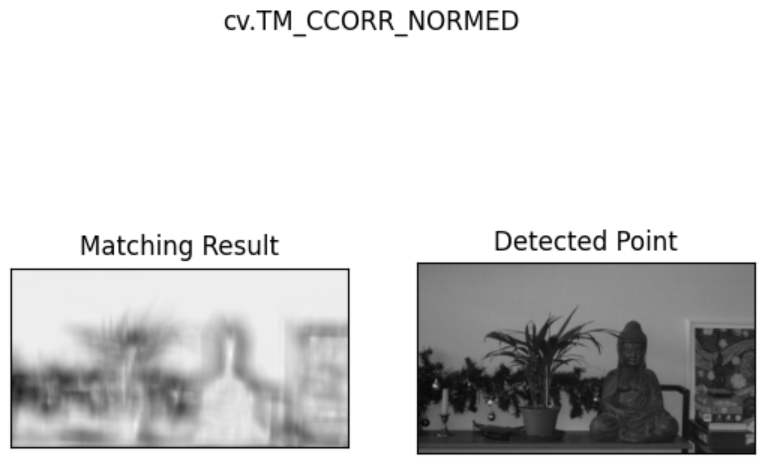
****

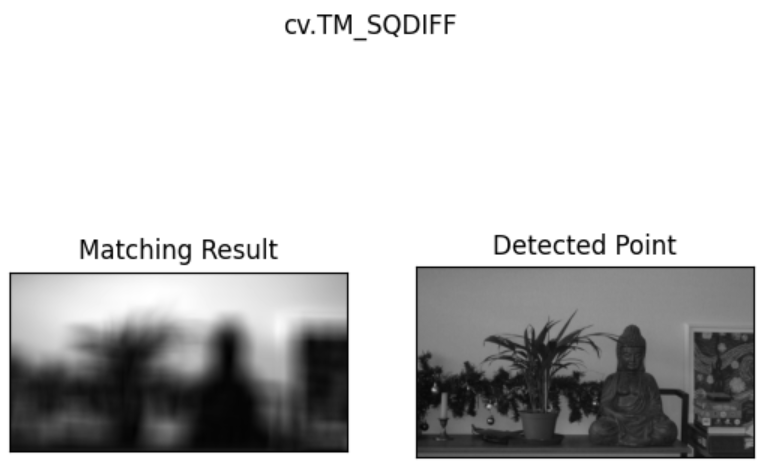
**Завдання 2.5. Розпізнавання об’єктів на зображенні за допомогою методів зіставлення шаблонів (Template Matching)**

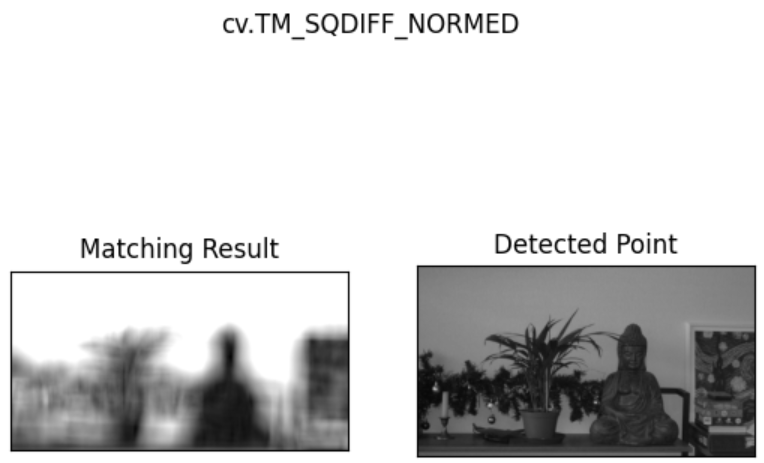
****

****

****

****

****

****

**cv.TM\_CCOEFF** - Коефіцієнт кореляції Пірсона. Вираховує кореляцію між шаблоном та регіонами зображення.

**cv.TM\_CCOEFF\_NORMED** - Нормалізований коефіцієнт кореляції Пірсона. Тут значення нормалізовані до діапазону від -1 до 1.

**cv.TM\_CCORR** - Коефіцієнт кореляції. Порівнюєс шаблон і частини зображення, шукаючи максимальну кореляцію.

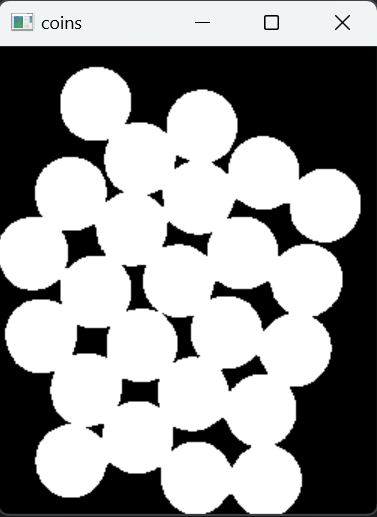
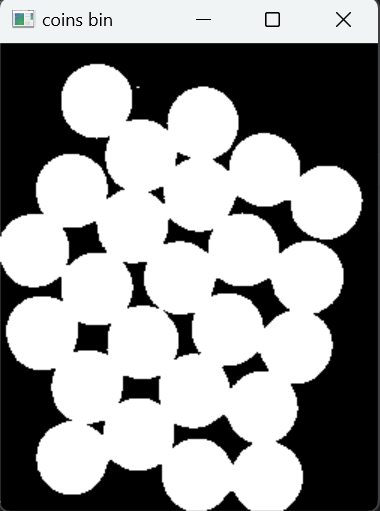
**cv.TM\_CCORR\_NORMED** - Нормалізований коефіцієнт кореляції. Значення також нормалізовані до діапазону від 0 до 1.

**cv.TM\_SQDIFF** - Квадрат різниці. Він шукає мінімальне значення суми квадратів різниць між шаблоном і регіонами зображення.

**cv.TM\_SQDIFF\_NORMED** - Нормалізований квадрат різниці. Тут значення нормалізовані до діапазону від 0 до 1, де найменше значення вказує на найкраще зібрання.

|  |
| --- |
| import cv2 as cv from matplotlib import pyplot as plt  img = cv.imread('budda\_full.JPG', 0) img2 = img.copy() template = cv.imread('budda\_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, 15, 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() |

**Завдання 2.6. Сегментація зображення алгоритмом водорозподілу**

** **

|  |
| --- |
| 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)  kernel = np.ones((3, 3), np.uint8) opening = cv2.morphologyEx(thresh, cv2.MORPH\_OPEN, kernel, iterations=2)  sure\_bg = cv2.dilate(opening, kernel, iterations=3)  dist\_transform = cv2.distanceTransform(opening, cv2.DIST\_L2, 5) ret, sure\_fg = cv2.threshold(dist\_transform, 0.7 \* dist\_transform.max(), 255, 0)  sure\_fg = np.uint8(sure\_fg) unknown = cv2.subtract(sure\_bg, sure\_fg) cv2.imshow("coins ", opening) cv2.waitKey(0)  ret, markers = cv2.connectedComponents(sure\_fg)  markers = markers + 1  markers[unknown == 255] = 0 markers = cv2.watershed(img, markers) img[markers == -1] = [255, 0, 0] cv2.imshow("coins\_markers", img) cv2.waitKey(0) |

***Висновки***: в ході виконання лабораторної, я, використовуючиспеціалізовані бібліотеки та мову програмування Python навчився обробляти зображення за допомогою бібліотеки OpenCV.