**ЗВІТ**

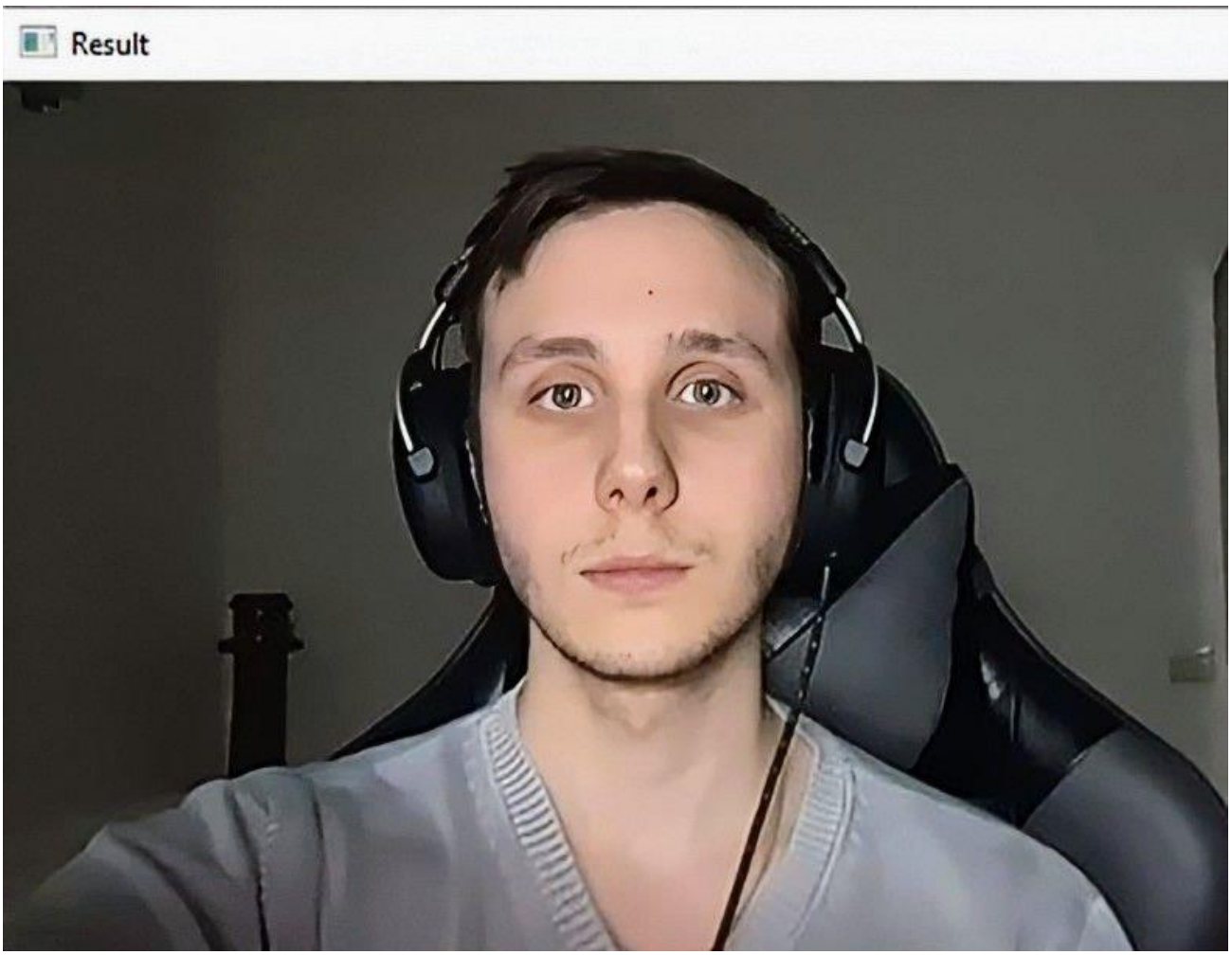
**З лабораторної роботи №8**

**ДОСЛІДЖЕННЯ МЕТОДІВ КОМП'ЮТЕРНОГО ЗОРУ**

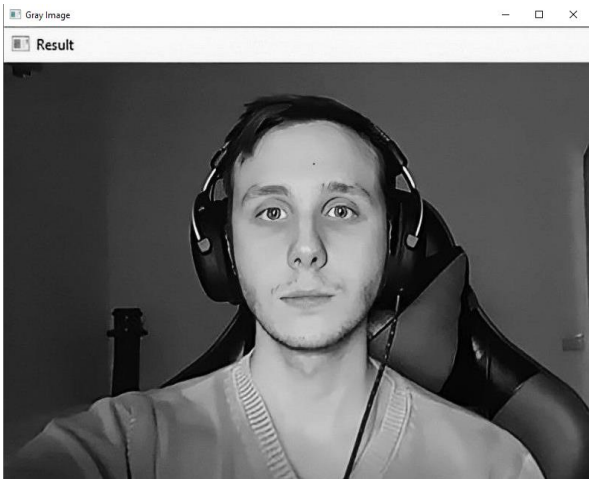
КН-20-1 навчальної групи

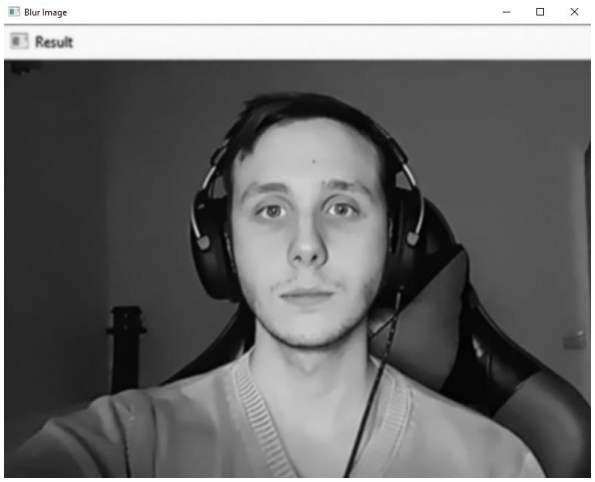
Кірія Даніли Олеговича варіант №6

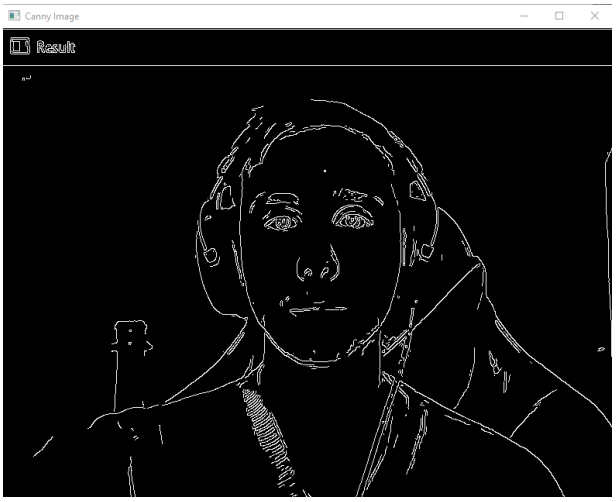
**Завдання 2.1.** Завантаження зображень та відео в OpenCV

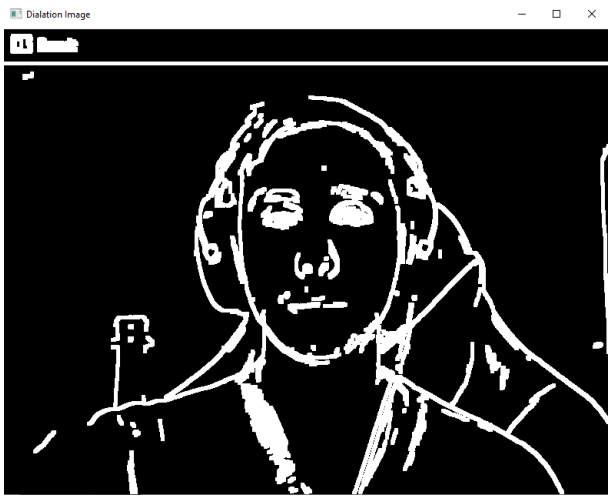


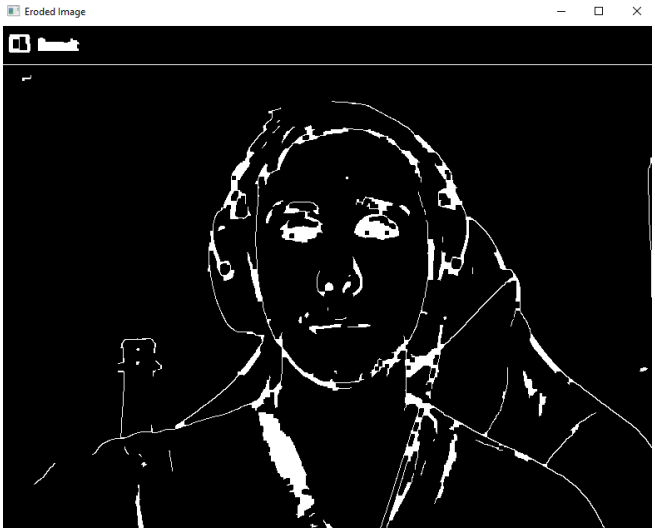

**Завдання 2.2.** Дослідження перетворень зображення




```python
import cv2
import numpy as np
img = cv2.imread("kirii.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)
```

1. Метод cvtColor:

- Застосування: Цей метод використовується для конвертації кольорового простору зображення. Наприклад, якщо вихідне зображення має кольоровий простір BGR (Blue, Green, Red), то cvtColor дозволяє перетворити його у інший колірний простір, такий як Grayscale або HSV (Hue, Saturation, Value).
- Результат: Застосування цього методу призводить до отримання нового зображення зі зміненим колірним простором. Наприклад, якщо вхідне зображення було кольоровим, вихідне зображення може бути чорно-білим (Grayscale) або в іншому кольоровому просторі.

2. Метод GaussianBlur:

- Застосування: Цей метод використовується для зменшення шуму в зображенні шляхом застосування фільтра Гаусса. Він розгладжує зображення, зменшуючи вплив високочастотних компонентів.
- Результат: Застосування GaussianBlur призводить до отримання зображення, у якому менше шуму, і більш плавні переходи між пікселями. Це може бути корисно перед подальшою обробкою, такою як виявлення країв.

3. Метод Canny:

- Застосування: Метод Canny використовується для виявлення країв на зображенні. Він використовує градієнти яскравості для визначення місць зміни інтенсивності.
- Результат: Застосування методу Canny призводить до отримання чорно-білого зображення, де білі пікселі представляють визначені краї, а чорні пікселі відповідають фону. Це дозволяє виділити структурні елементи та контури на зображенні.

4. Метод dilate:

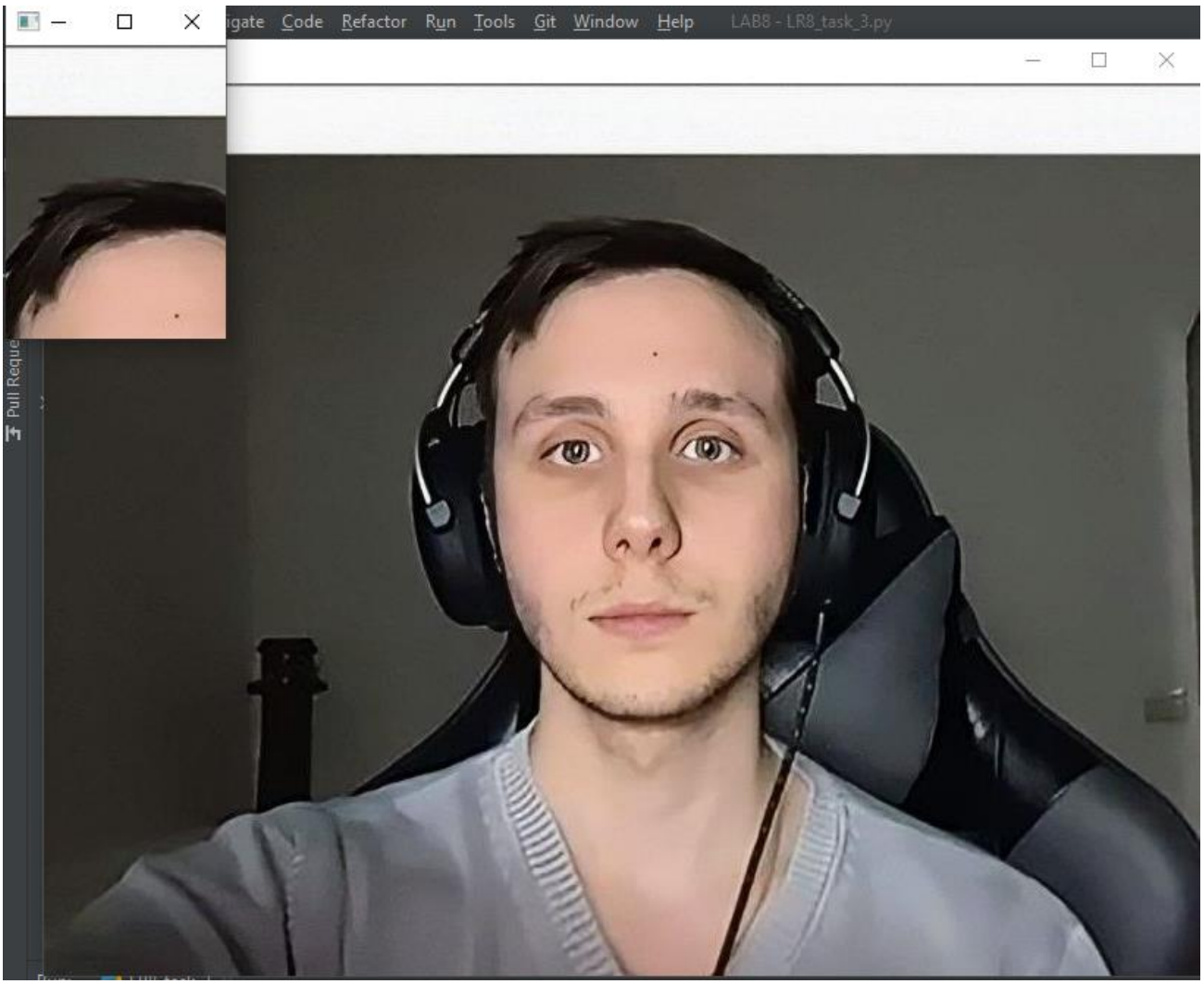
- Застосування: dilate використовується для збільшення розмірів об'єктів на зображенні. Він розширює білі області (об'єкти) на зображенні.

- Результат: Застосування dilate призводить до збільшення розміру об'єктів на зображенні, що може бути корисним у випадках, коли потрібно об'єднати близькі області або відновити деякі деталі.

5. Метод erode:

- Застосування: erode використовується для зменшення розмірів об'єктів на зображенні. Він зменшує білі області (об'єкти), зменшуючи їхні розміри.
- Результат: Застосування erode призводить до зменшення розміру об'єктів на зображенні. Це може бути корисно для відокремлення близько розташованих об'єктів або видалення деяких дрібних деталей.

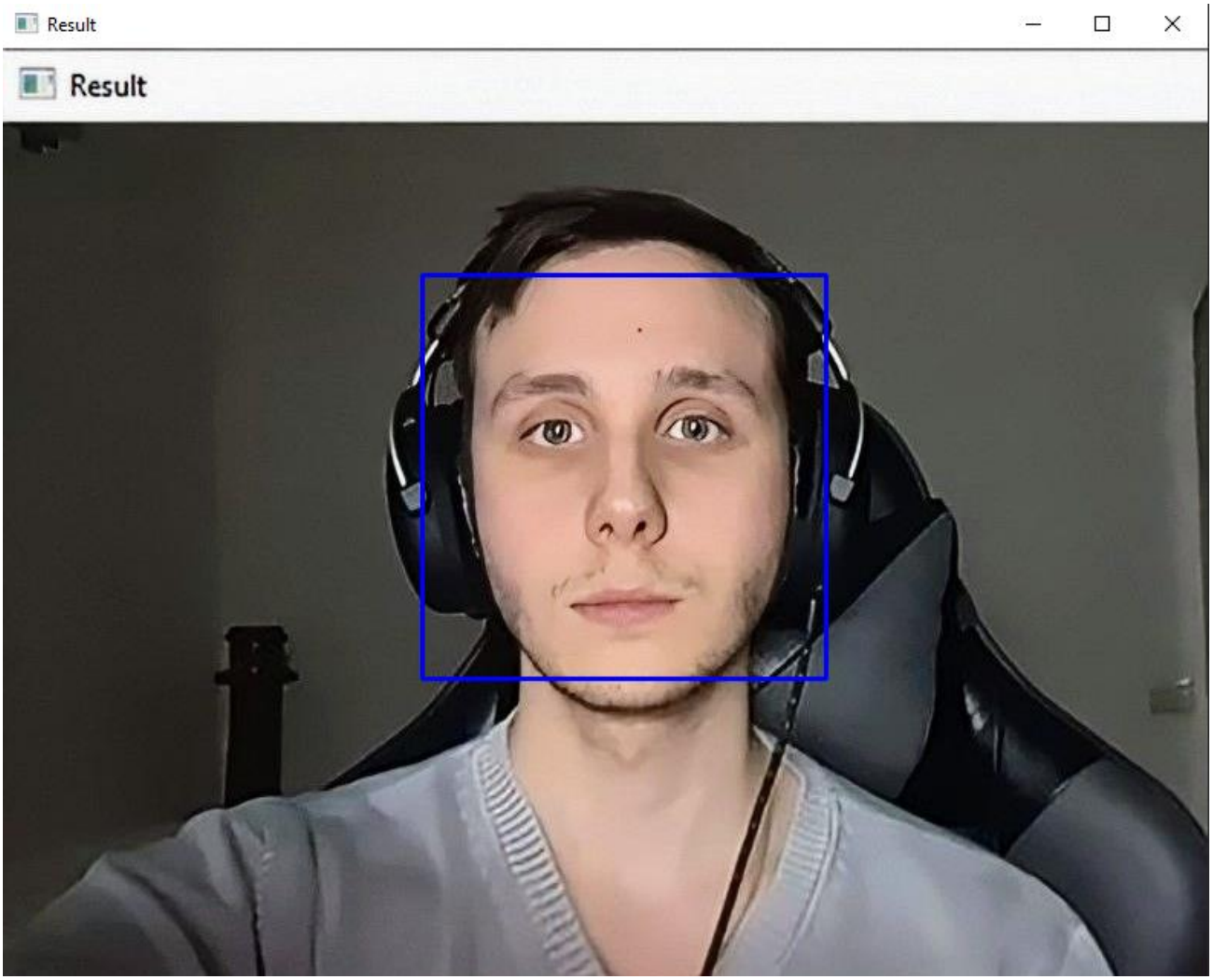
**Завдання 2.3.** Вирізання частини зображення




```
import cv2
import numpy as np
img = cv2.imread("kirii.jpg")
print(img.shape)
imgResize = cv2.resize(img,(1000,500))
```

```
print(imgResize.shape)
imgCropped = img[0:200,300:450]
cv2.imshow("Image",img)
#cv2.imshow("Image Resize",imgResize)
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('kirii.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)
```

Висновок:

Метод виявлення об'єктів за допомогою каскадних класифікаторів на основі функцій Хаара, як запропоновано Віолою та Джонсом, є ефективним підходом до розпізнавання обличчя. Процес включає наступні кроки:

1. **Навчання класифікатора:**
    - Збір великої кількості позитивних (зображення облич) і негативних (зображення без облич) зразків для тренування класифікатора.
    - Використання функцій Хаара для витягування ознак з облич та навчання класифікатора на цих ознаках.
2. **Використання каскаду класифікаторів:**
    - Групування функцій Хаара в етапи класифікаторів, де кожен етап має свій піднабір функцій.
    - Застосування класифікатора в каскаді, де кожен етап розпізнає об'єкт на зображенні. Якщо вікно не проходить перший етап, воно відкидається, інакше продовжується до наступних етапів.
3. **Використання Adaboost для вибору функцій:**
    - Застосування Adaboost для визначення оптимальних порігів для класифікації облич та ваг функцій.
    - Вибір функцій, які мають мінімальну частоту помилок, що дозволяє відібрати найточніші функції для класифікації.
4. **Каскадний підхід для оптимізації часу обчислень:**
    - Відкидання вікон, які вже на початку не є потенційними обличчями, що дозволяє скоротити кількість функцій для обчислення.
    - Зосередження на областях, де може бути обличчя, для оптимізації часу обчислень.

Загалом, цей метод дозволяє швидко та ефективно виявляти обличчя на зображеннях, використовуючи машинне навчання та каскадний підхід для оптимізації обчислень. Автори стверджують, що із всією кількістю функцій, які можуть бути використані, лише невелика їх частина дійсно несе значущу інформацію для розпізнавання обличчя, і вони ефективно відібрані за допомогою методу Adaboost.

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

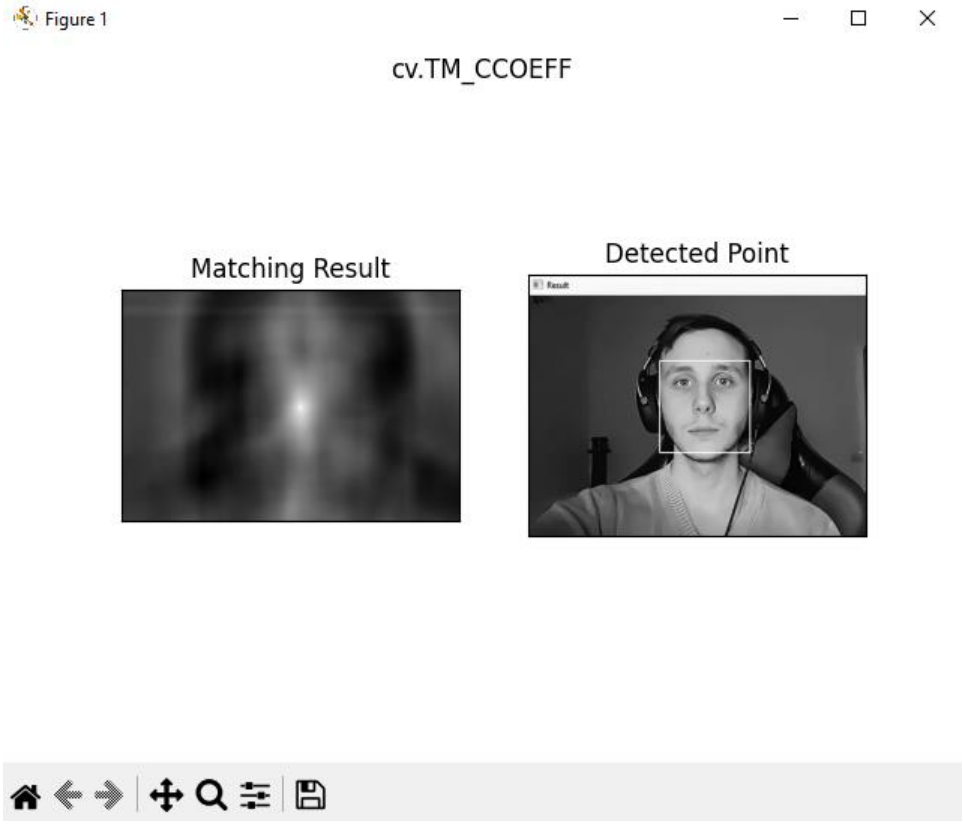


Figure 1
cv.TM_CCOEFF
Matching Result
Detected Point
Result

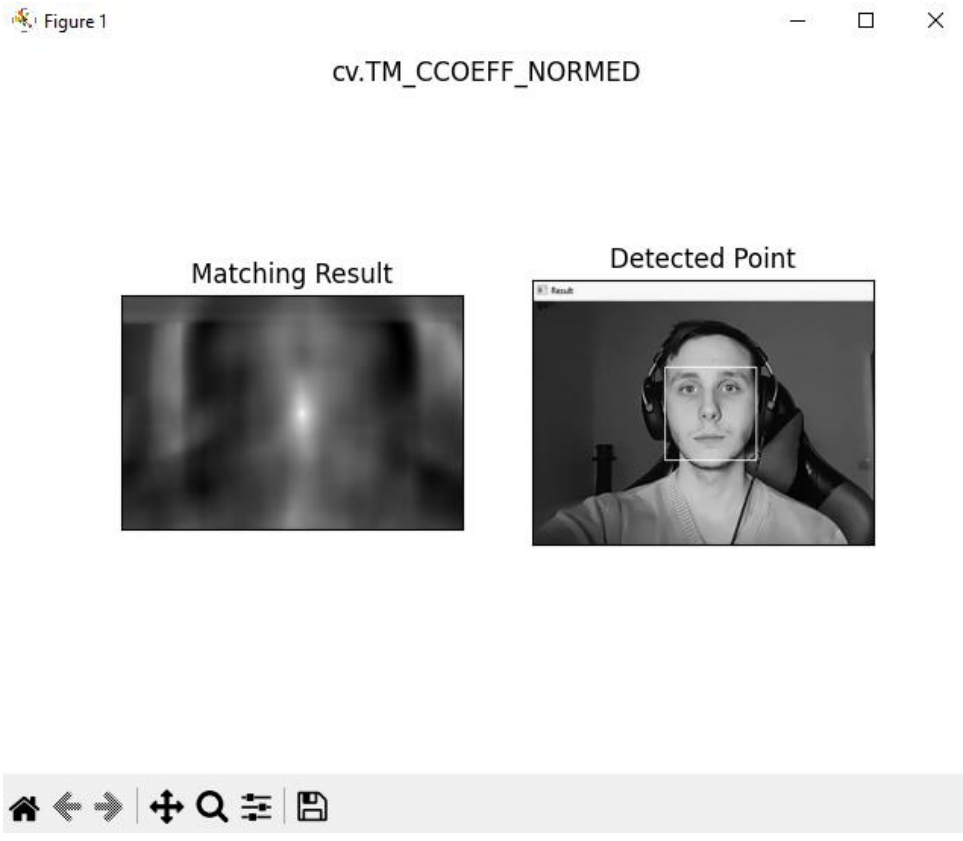


Figure 1
cv.TM_CCOEFF_NORMED
Matching Result
Detected Point
Result

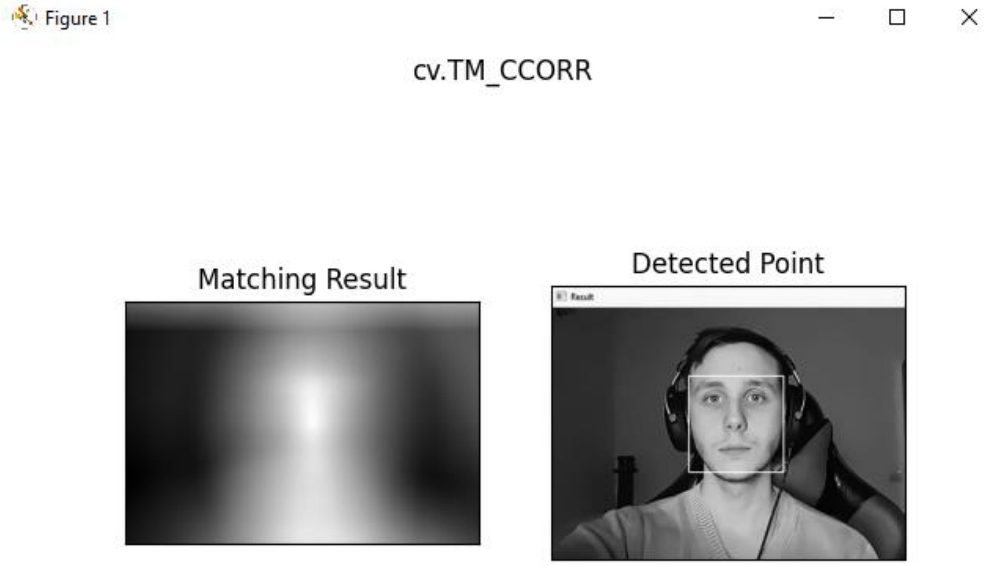


Figure 1
cv.TM_CCORR
Matching Result
Detected Point

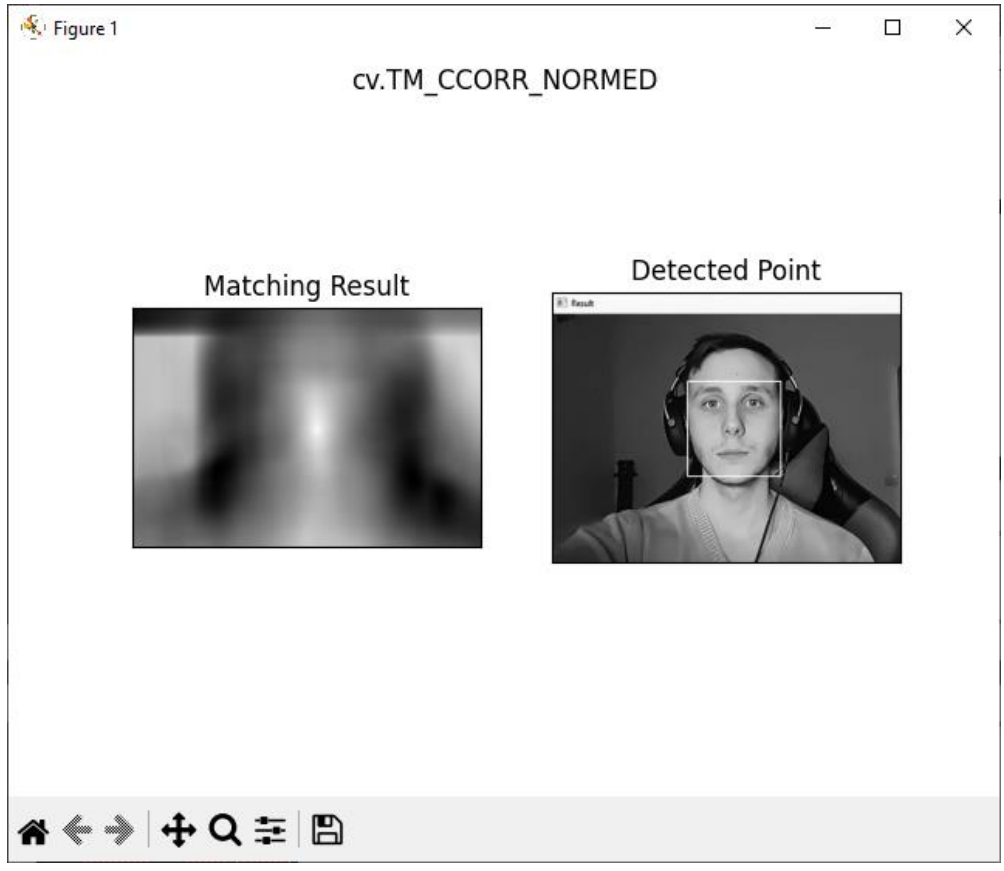


Figure 1
cv.TM_CCORR_NORMED
Matching Result
Detected Point

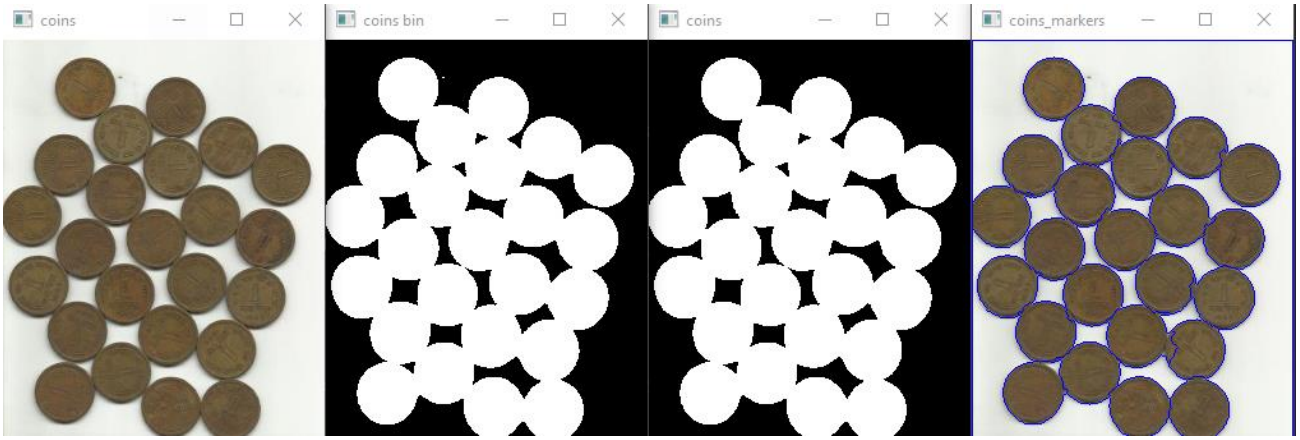
1. **cv.TM_CCOEFF (Cross-Correlation):**
    - Велике значення вказує на високу подібність між шаблоном і зображенням.
    - Використовується, коли важливо виявити будь-які входження шаблону, не обов'язково точну копію.
2. **cv.TM_CCOEFF_NORMED (Normalized Cross-Correlation):**
    - Повертає нормалізовані значення, що лежать в діапазоні від -1 до 1.
    - Використовується, коли важливо отримати найбільш точне зіставлення незалежно від абсолютних значень пікселів.
3. **cv.TM_CCORR (Cross-Correlation):**
    - Якщо вихідне зображення містить шаблон, результат буде максимальним.
    - Використовується для пошуку точних входжень шаблону.
4. **cv.TM_CCORR_NORMED (Normalized Cross-Correlation):**
    - Повертає нормалізовані значення в діапазоні від 0 до 1.
    - Використовується для отримання найбільш точних зіставлень незалежно від абсолютних значень пікселів.
5. **cv.TM_SQDIFF (Sum of Squared Differences):**
    - Мінімальне значення вказує на найкраще зіставлення.
    - Використовується, коли шаблон може бути меншим за область на зображенні.
6. **cv.TM_SQDIFF_NORMED (Normalized Sum of Squared Differences):**
    - Повертає нормалізовані значення в діапазоні від 0 до 1.
    - Використовується для отримання найкращих зіставлень, де менше значення вказує на краще відповідність.

**Вибір методу для задачі:**

- Який метод краще використовувати залежить від конкретної задачі та властивостей даних.

- Для точного визначення відповідності можна використовувати cv.TM_CCORR або cv.TM_CCORR_NORMED.
- Для виявлення областей або об'єктів, які можуть бути відрізняються за розміром або освітленням, можна спробувати cv.TM_CCOEFF_NORMED або cv.TM_SQDIFF_NORMED.

**Завдання 2.6.** Сегментація зображення алгоритмом водорозподілу




```
import numpy as np
import cv2
from matplotlib import pyplot as plt
img = cv2.imread('coins.jpg')
cv2.imshow("coins",img)

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)

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)

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)
```

Висновок:

Алгоритм вододілу є потужним інструментом для сегментації зображень, де зображення розглядається як топографічна поверхня, а процес розподілу нагадує підняття рівнів води в долинах та формування бар'єрів у високих місцях.

Процес вододілу включає кілька етапів:

1. **Створення бінарного зображення:** Зображення розділяється на об'єкт та фон.
2. **Обчислення відстані перетворення:** Визначає відстань від кожного пікселя до найближчого пікселя об'єкту.
3. **Знаходження точок локальних максимумів:** Визначає місця, де вода починає заповнювати долини.
4. **Маркування позначок:** Маркери вказують, які області потрібно об'єднати чи відокремити.

Алгоритм вододілу на основі маркерів в OpenCV дозволяє користувачу інтерактивно визначати області переднього плану, фону та невизначеності, що призводить до точної сегментації зображень.

**Висновки:** в ході виконання лабораторної роботи, використовуючи спеціалізовані бібліотеки та мову програмування Python було оброблено зображення за допомогою бібліотеки OpenCV

GitHub: https://github.com/invicibleee/Artificial-intelligence.git