**ЛАБОРАТОРНА РОБОТА №8**

**ПОЛОНЕВИЧ Д.В.**

**ДОСЛІДЖЕННЯ МЕТОДІВ КОМП’ЮТЕРНОГО ЗОРУ**

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

**ХІД РОБОТИ**

Завдання №1. Завантаження зображень та відео в OpenCV.

LR\_8\_task\_1.py

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

Результат виконання:

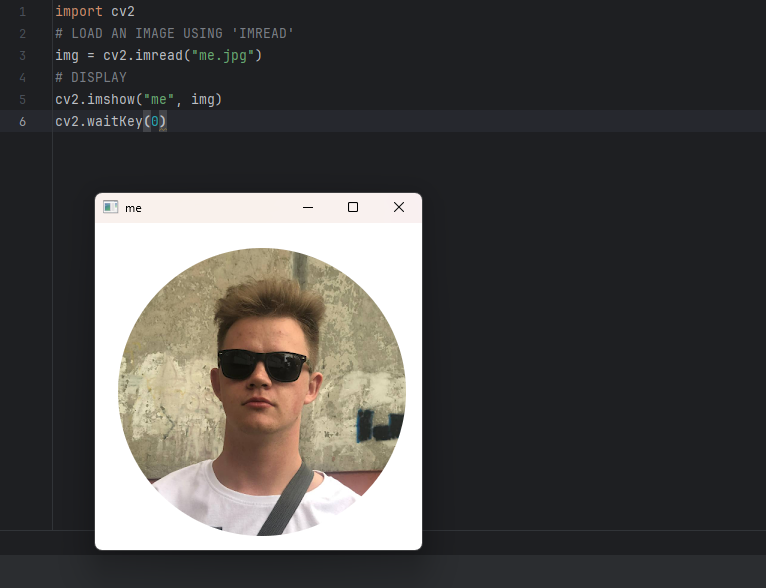
****

Рис. 1. Завдання №1.

Завдання №2. Дослідження перетворень зображення.

LR\_8\_task\_2.py

import cv2  
import numpy as np  
img = cv2.imread("me.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:

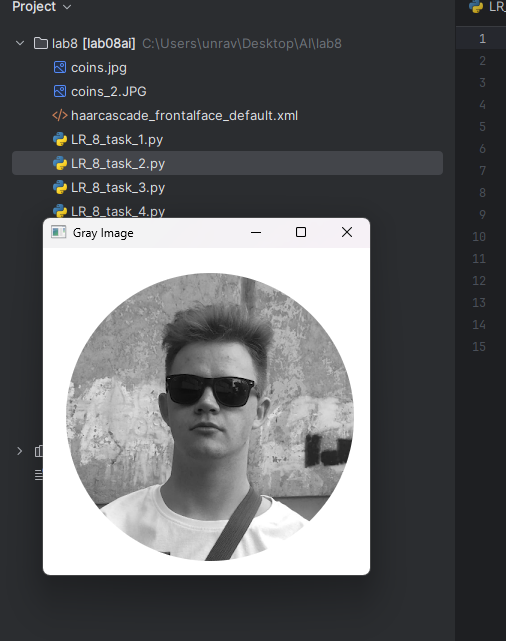


Рис. 2. Завдання №2: Gray Image.

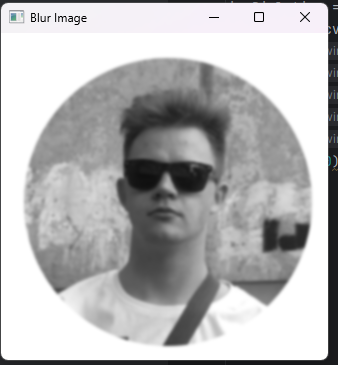


Рис. 3. Завдання №2: Blur Image.



Рис. 4. Завдання №2: Canny Image.

****

Рис. 5. Завдання №2: Dialation Image.

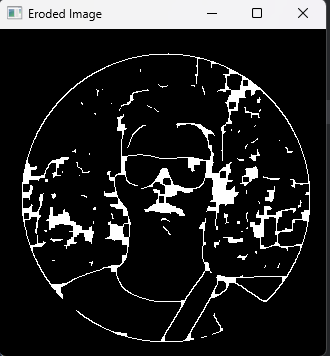


Рис. 6. Завдання №2: Eroded Image.

Метод **cvtColor** застосовується для зміни колірного простору зображення. В результаті його використання отримано зображення у градації сірого кольору.

Метод **GaussianBlur** використовується для застосування Гаусового згладжування до зображення, призводячи до отримання замиленого зображення.

Метод **Canny** застосовується для виявлення країв зображення, що призводить до отримання зображення з контурами обличчя.

Метод **dilate** використовується для збільшення особливостей зображення, що призводить до отримання зображення з контурами обличчя.

Метод **erode** використовується для зменшення рис зображення, результатом є зображення з розмитим контуром обличчя.Завдання №3. Вирізання частини зображення.

LR\_8\_task\_3.py

import cv2  
import numpy as np  
img = cv2.imread("me.jpg")  
print(img.shape)  
imgResize = cv2.resize(img, (1000, 500))  
print(imgResize.shape)  
imgCropped = img[150:800, 60:700]  
cv2.imshow("Image", img)  
# cv2.imshow("Image Resize",imgResize)  
cv2.imshow("Image Cropped", imgCropped)  
cv2.waitKey(0)

Результат виконання:

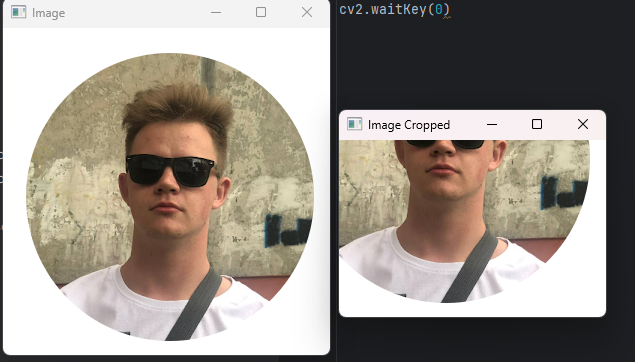
****

Рис. 7. Завдання №3.

Завдання №4. Розпізнавання обличчя на зображенні.

LR\_8\_task\_4.py

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

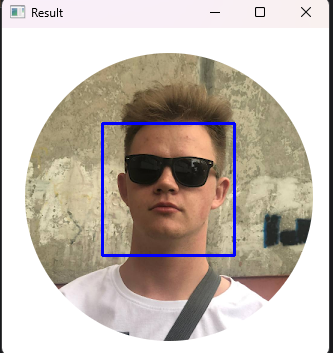


Рис. 8. Завдання №4.

Були отримані непогані результати з розпізнавання обличчя на прикладі мого зображення.

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

LR\_8\_task\_5.py

import cv2 as cv  
import numpy as np  
from matplotlib import pyplot as plt  
img = cv.imread('me.jpg', 0)  
img2 = img.copy()  
template = cv.imread('my\_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:

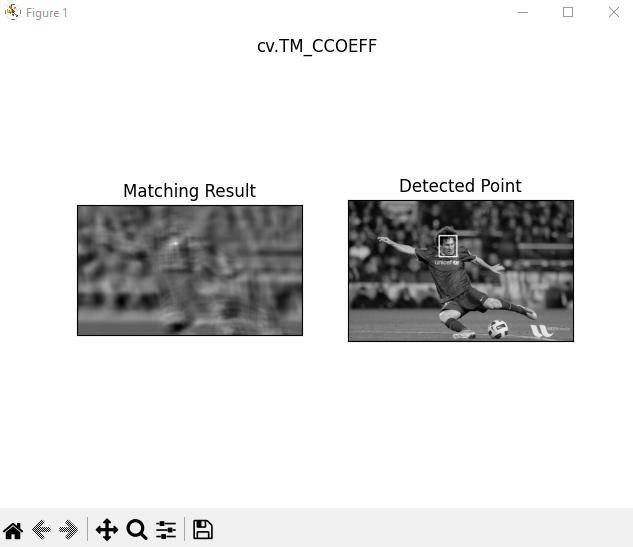


Рис. 9. Завдання №5.

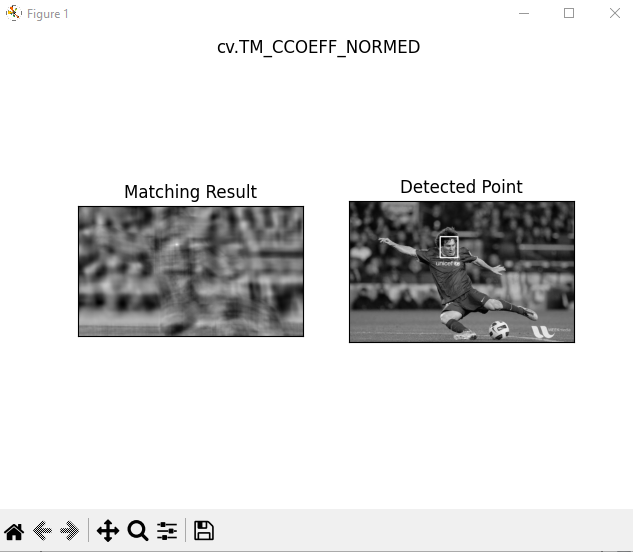


Рис. 10. Завдання №5.

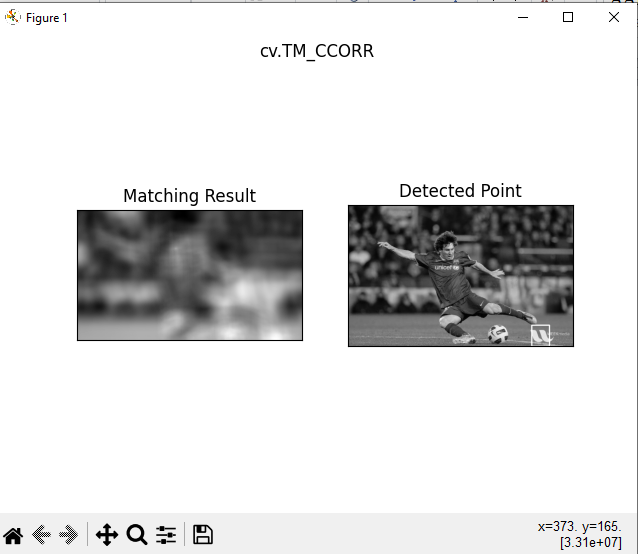


Рис. 11. Завдання №5.

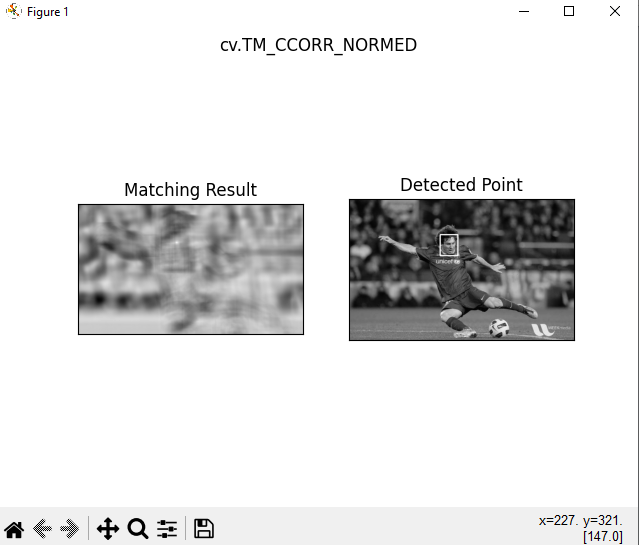


Рис. 12. Завдання №5.

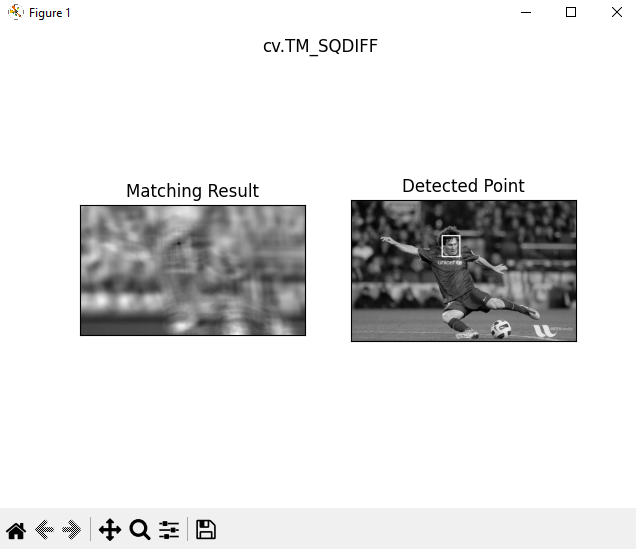


Рис. 13. Завдання №5.

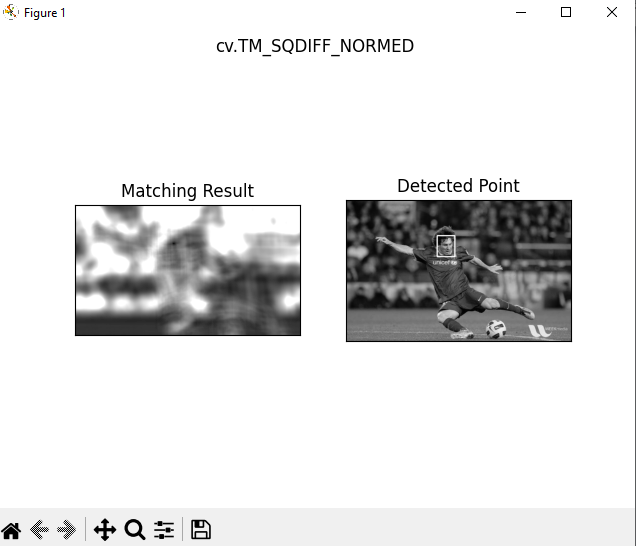


Рис. 14. Завдання №5.

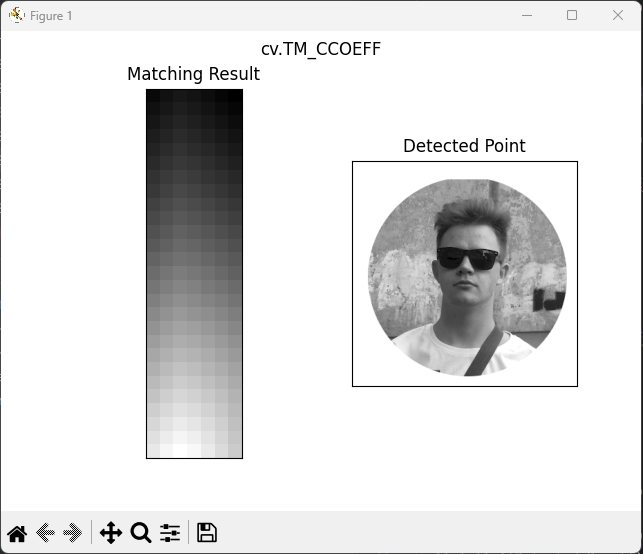


Рис. 15. Завдання №5.

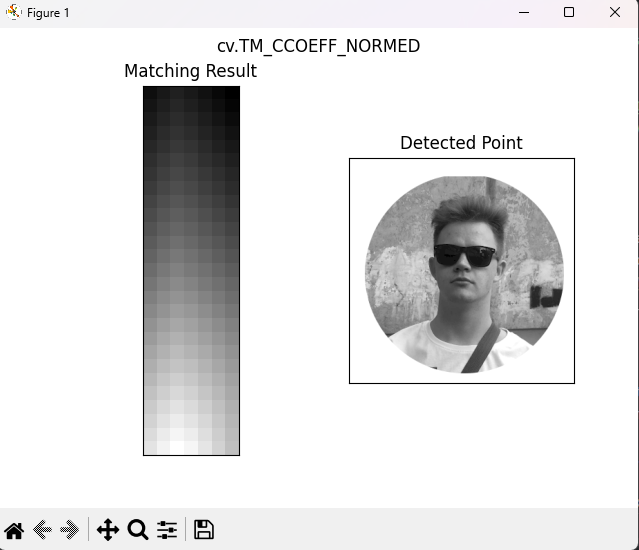


Рис. 16. Завдання №5.

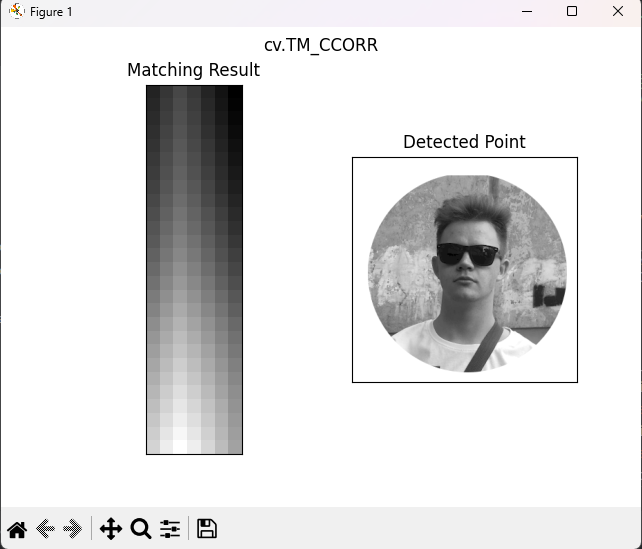


Рис. 17. Завдання №5.

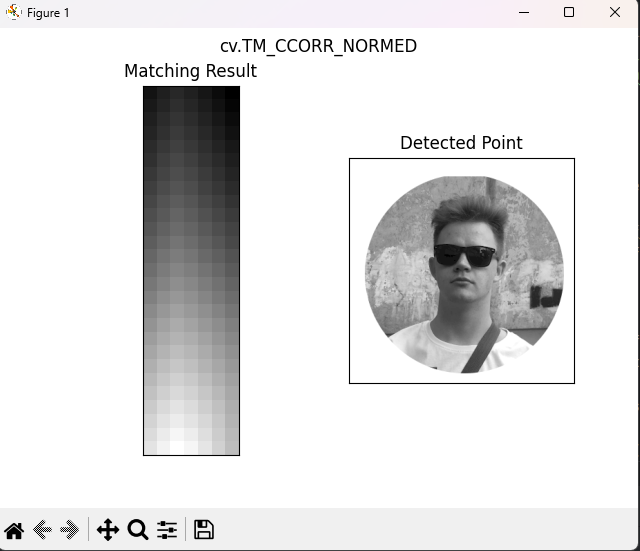
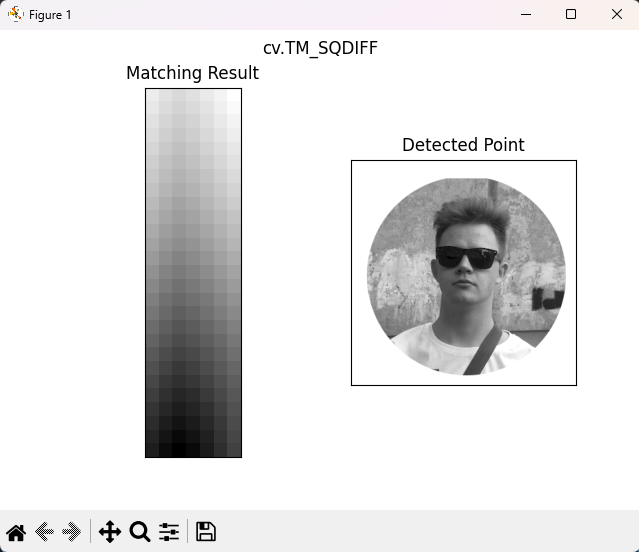


Рис. 18. Завдання №5.



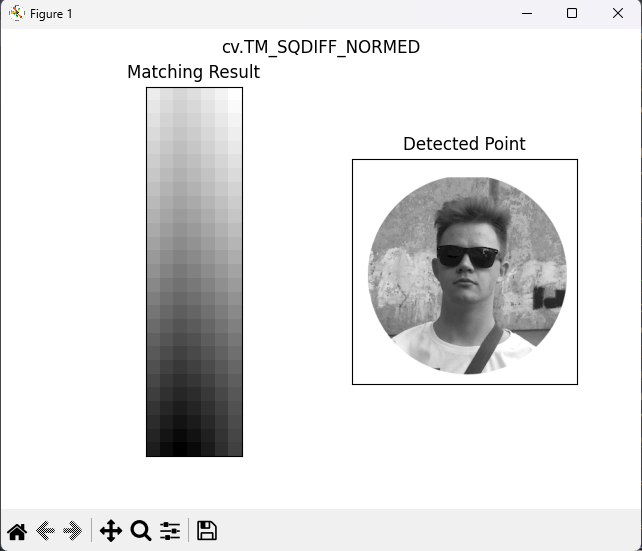
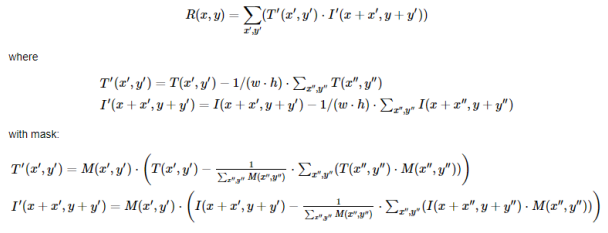


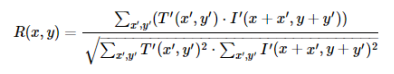
Рис. 19. Завдання №5.

Насправді, обрана мною фотографія не найкраще сюди підходить, але на прикладі фотографії з Мессі можна сказати наступне:

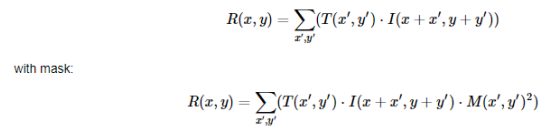
cv.TM\_CCOEFF:

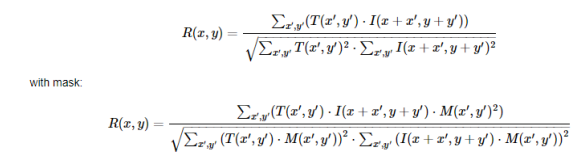


cv.TM\_CCOEFF\_NORMED:

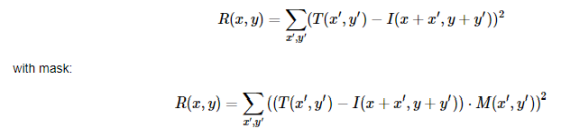


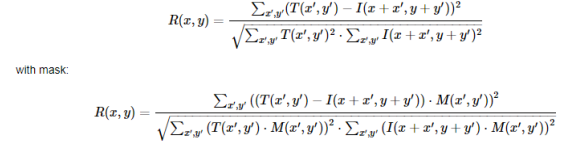
cv.TM\_CCORR:

 cv.TM\_CCORR\_NORMED:



cv.TM\_SQDIFF:

 cv.TM\_SQDIFF\_NORMED:



**Я вважаю, що cv2.TM\_SQDIFF – найкращий метод для поставленої задачі, бо мінімальне значення дає найкращий збіг.**

Завдання №6. Сегментація зображення алгоритмом водорозподілу.

LR\_8\_task\_6.py

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:



Рис. 20. Завдання №6.

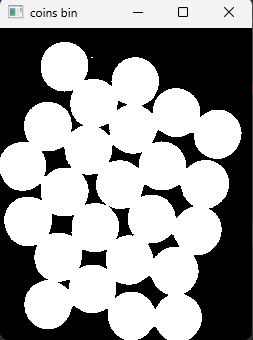


Рис. 21. Завдання №6.

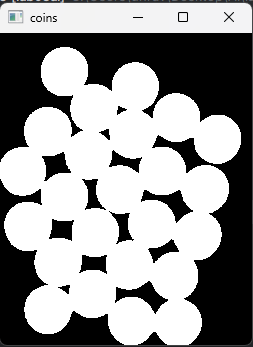


Рис. 22. Завдання №6.



Рис. 23. Завдання №6.

В результаті 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 коректно визначені.

Завдання №7. Сегментація зображення.

LR\_8\_task\_7.py

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 = "gray"  
 plt.imshow(imagens[i], cmap)  
 plt.title(titulos[i])  
 plt.xticks([]), plt.yticks([])  
plt.show()

Результат виконання завдання:

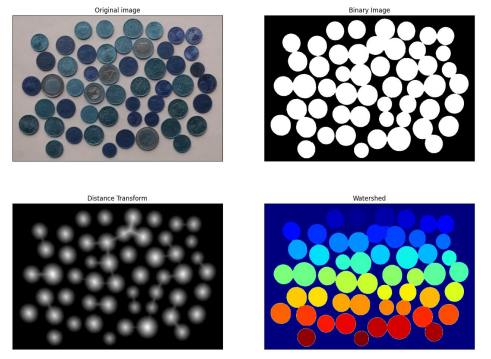


Рис. 24. Завдання №7.

Було отримано зображення з виявленими сегментами. Результат є досить непоганим для такої якості вхідного зображення.

**Висновок**: В лабораторній роботі було проведено оброблення зображення за допомогою бібліотеки OpenCV, використовуючи спеціалізовані бібліотеки та мову програмування Python. Основний акцент був зроблений на сегментації зображення за допомогою алгоритму водорозподілу. Також було проведено аналіз розпізнавання об'єктів на зображенні за допомогою методів зіставлення шаблонів.

GitHub: https://github.com/unravee1/AI\_labs