**ЛАБОРАТОРНА РОБОТА № 8**

**ДОСЛІДЖЕННЯ МЕТОДІВ КОМП’ЮТЕРНОГО ЗОРУ**

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

**Хід роботи:**

Завдання 1. Завантаження зображень та відео в OpenCV

Лістинг коду файлу Task\_1.py:

import cv2  
  
  
def get\_image():  
 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  
  
  
def load\_image():  
 img = cv2.imread("Chyzhmotria.png")  
 cv2.imshow("Output", img)  
 cv2.waitKey(0)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 # get\_image()  
 load\_image()



Рис.8.1 – Збережене та завантажене зображення з веб-камери

Завдання 2. Дослідження перетворень зображення

Лістинг коду файлу Task\_2.py:

import cv2  
import numpy as np  
  
img = cv2.imread("Chyzhmotria.png")  
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)  
imgDilation = cv2.dilate(imgCanny, kernel, iterations=1)  
imgEroded = cv2.erode(imgDilation, kernel, iterations=1)  
  
cv2.imshow("Gray Image", imgGray)  
cv2.imshow("Blur Image", imgBlur)  
cv2.imshow("Canny Image", imgCanny)  
cv2.imshow("Dilation Image", imgDilation)  
cv2.imshow("Eroded Image", imgEroded)  
cv2.waitKey(0)

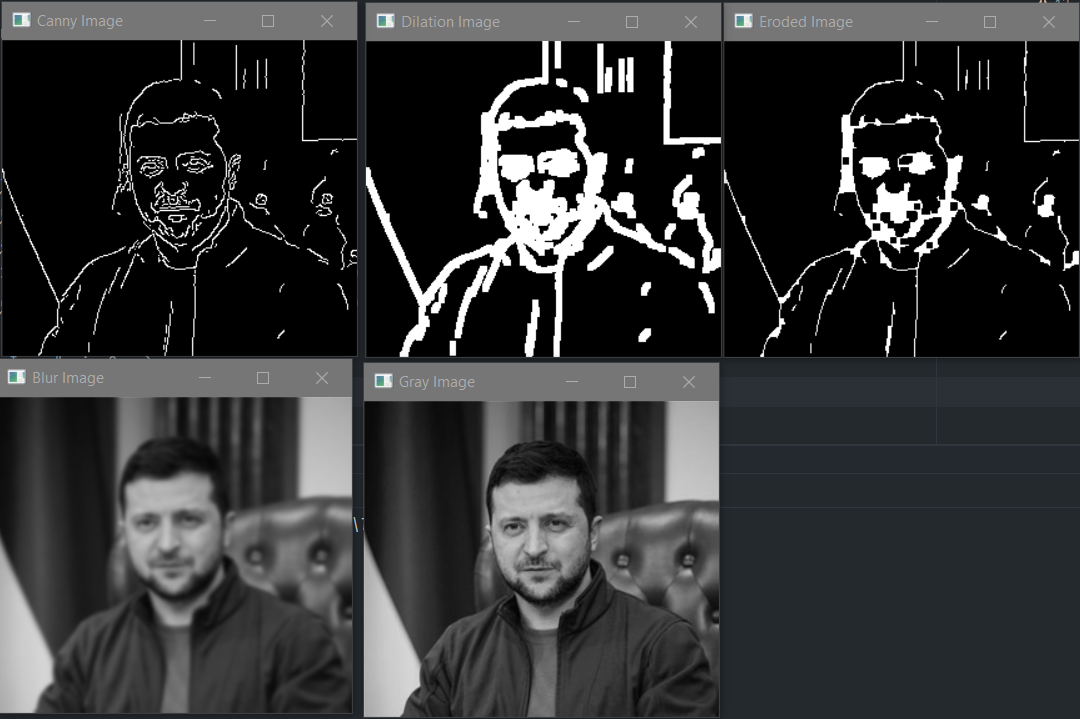


Рис.8.2 – Результати накладання фільтрів

Різні фільтри, надані OpenCV, надають можливість зміни кольорової гами зображення (зробити сірим), розмити, виявити кути, зробити більш різким тощо.

Завдання 3. Вирізання частини зображення

Лістинг коду файлу Task\_3.py:

import cv2  
import numpy as np  
  
img = cv2.imread("Chyzhmotria.png")  
print(img.shape)  
  
imgResize = cv2.resize(img, (300, 200))  
print(imgResize.shape)  
  
imgCropped = img[0:300, 100:500]  
  
cv2.imshow("Image", img)  
cv2.imshow("Image Resize", imgResize)  
cv2.imshow("Image Cropped", imgCropped)  
cv2.waitKey(0)

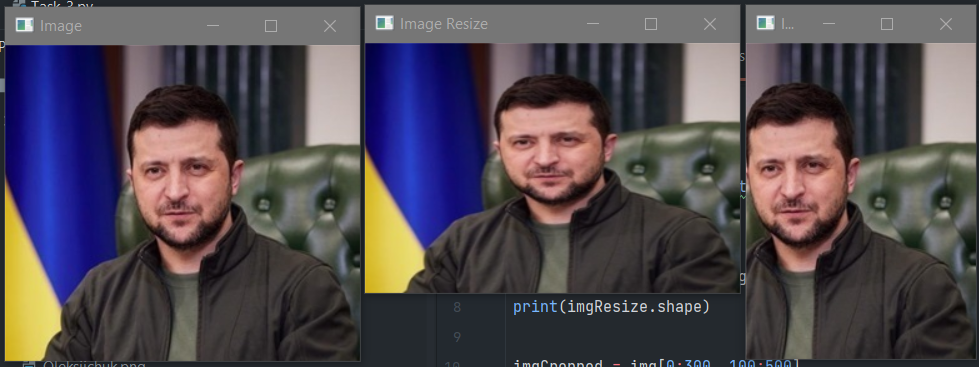


Рис.8.3 – Результат деформації зображень

Завдання 4. Розпізнавання обличчя на зображенні

Лістинг коду файлу Task\_4.py:

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

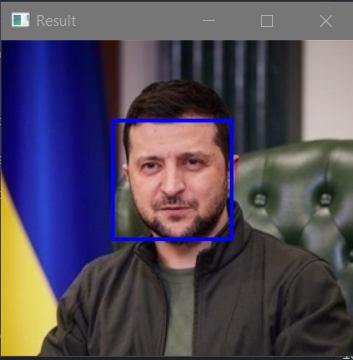


Рис.8.4 – Знайдене обличчя на зображенні

Використання каскадного класифікатора із наданою бібліотекою OpenCV надає правильний результат зі знаходження обличчя. Область обличчя виділена синім прямокутником.

Завдання 5. Розпізнавання об’єктів на зображенні за допомогою методів зіставлення шаблонів

Лістинг коду файлу Task\_5.py:

import cv2  
import numpy as np  
import matplotlib.pyplot as plt  
  
  
def main(full\_img="messi\_full.jpg", face\_img="messi\_face.jpg"):  
 img = cv2.imread(full\_img, 0)  
 img2 = img.copy()  
 template = cv2.imread(face\_img, 0)  
  
 w, h = template.shape[::-1]  
 methods = ["cv2.TM\_CCOEFF", "cv2.TM\_CCOEFF\_NORMED", "cv2.TM\_CCORR",  
 "cv2.TM\_CCORR\_NORMED", "cv2.TM\_SQDIFF", "cv2.TM\_SQDIFF\_NORMED"]  
  
 for meth in methods:  
 img = img2.copy()  
 method = eval(meth)  
  
 res = cv2.matchTemplate(img, template, method)  
 min\_val, max\_val, min\_loc, max\_loc = cv2.minMaxLoc(res)  
  
 if method in [cv2.TM\_SQDIFF, cv2.TM\_SQDIFF\_NORMED]:  
 top\_left = min\_loc  
 else:  
 top\_left = max\_loc  
  
 bottom\_right = (top\_left[0] + w, top\_left[1] + h)  
  
 cv2.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()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main('Chyzhmotria.png', 'Chyzhmotria\_face.png')

plt.show()

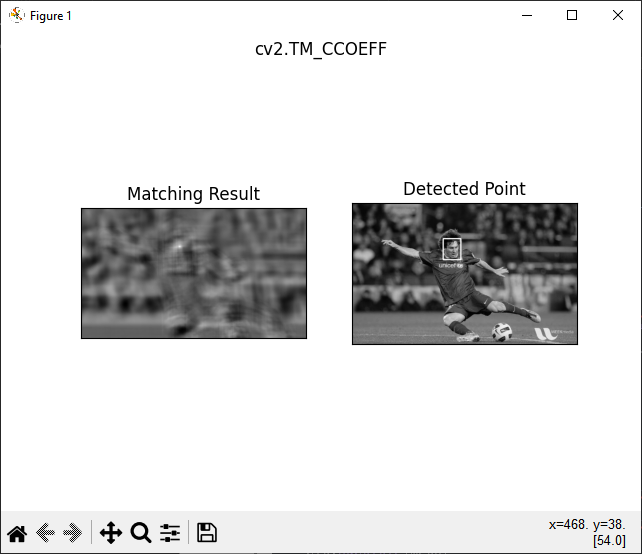


Рис.8.5 – Робота методу TM\_CCOEFF

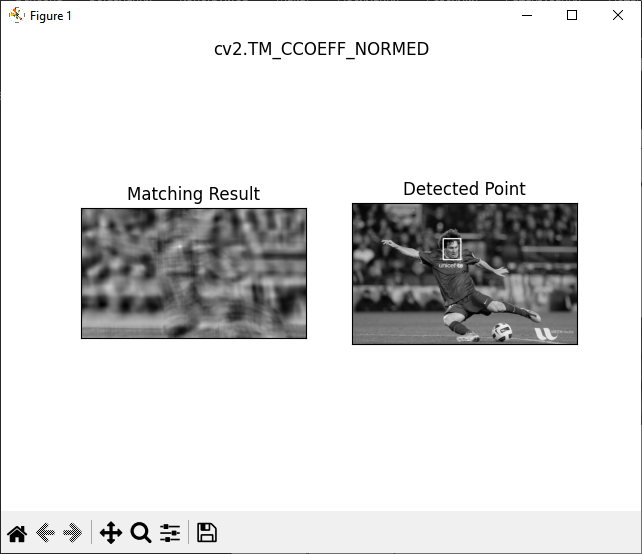


Рис.8.6 – Робота методу TM\_CCOEFF\_NORMED

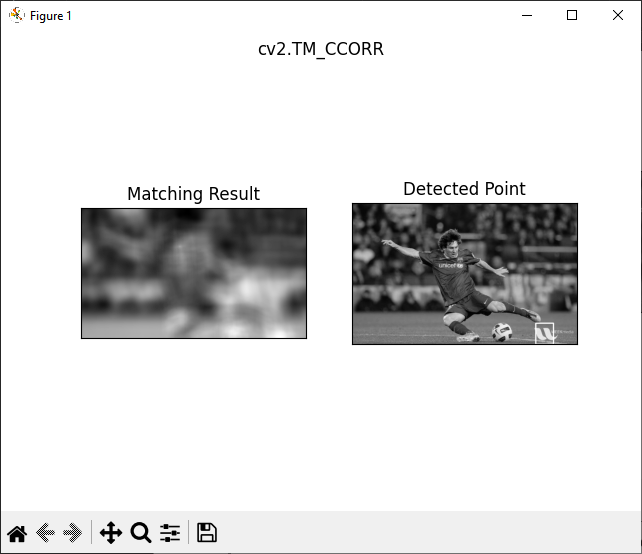


Рис.8.7 – Робота методу TM\_CCORR

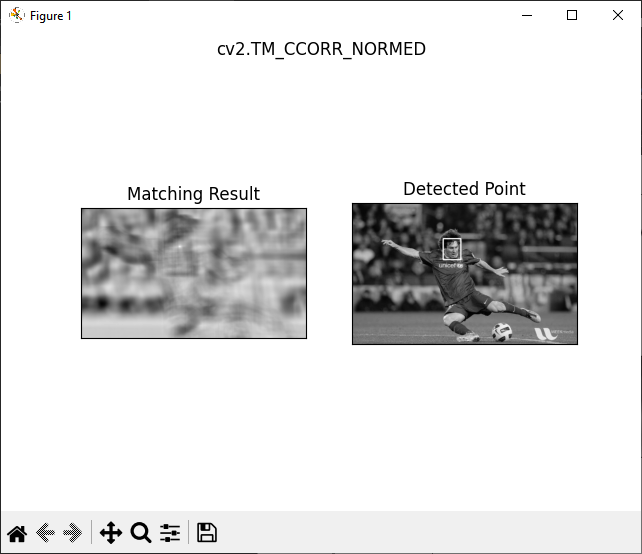


Рис.8.8 – Робота методу TM\_CCORR\_NORMED

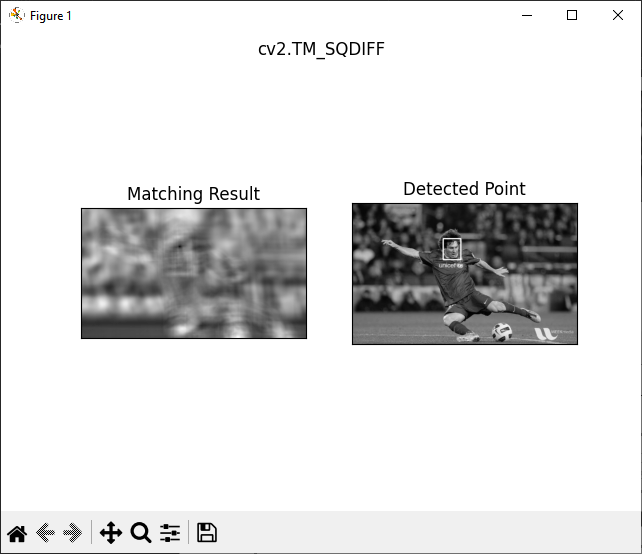


Рис.8.9 – Робота методу TM\_SQDIFF

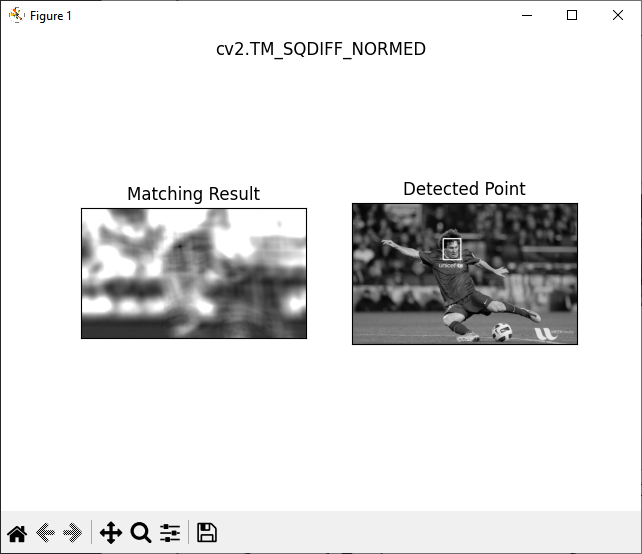
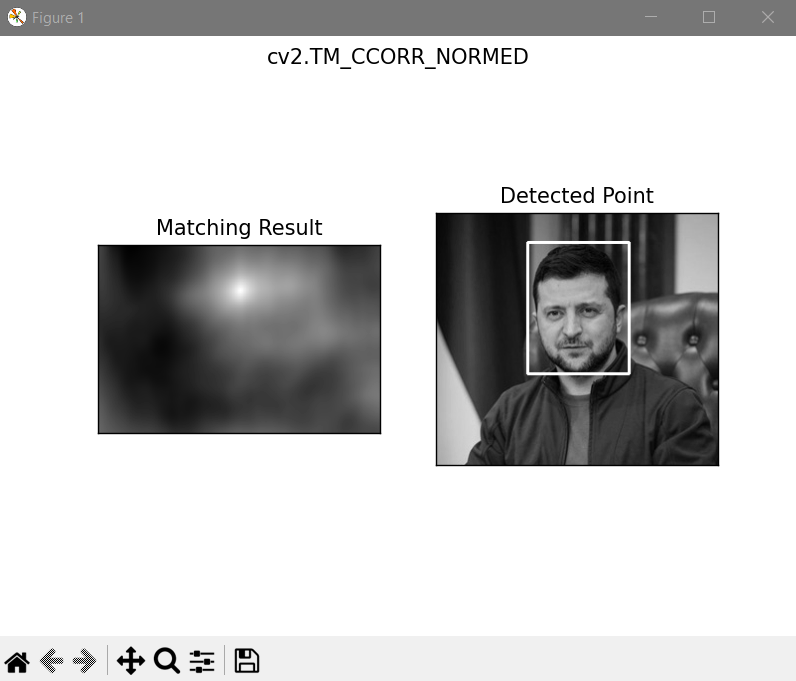


Рис.8.10 – Робота методу TM\_SQDIFF\_NORMED





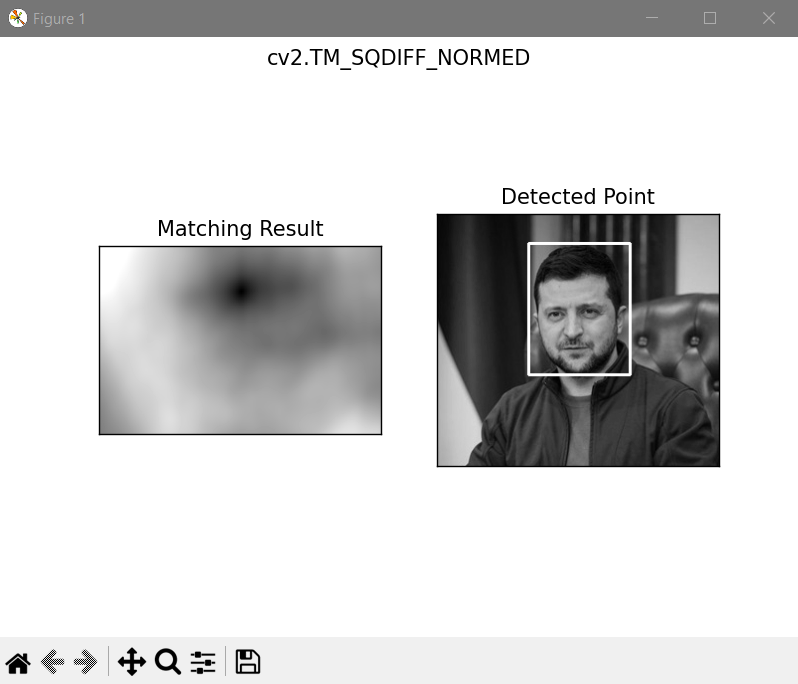


Рис.8.11 – Робота методів над власним зображенням

Робота більшості методів пошуку обличчя за шаблоном є ефективною, окрім методу TM\_CCORR. Роботу решти методів, які не вказано, на власному зображенні, мають ідентичний результат зі вказаними.

Завдання 6. Сегментація зображення алгоритмом водорозподілу

Лістинг коду файлу Task\_6.py:

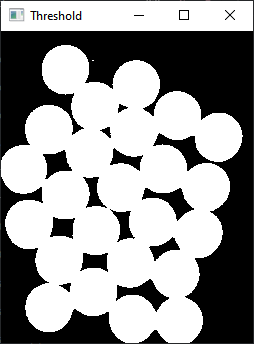
 

Рис.8.12 – Оригінальне та чорно-біле зображення монет

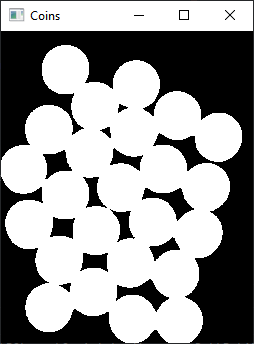


Рис.8.13 – Знешумлене зображення монет



Рис.8.14 – Помічені монети

Завдання 7. Дослідження перетворень зображення

Лістинг коду файлу 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')  
filtered = cv2.pyrMeanShiftFiltering(img, 20, 40)  
  
gray = cv2.cvtColor(filtered, cv2.COLOR\_BGR2GRAY)  
\_, thresh\_img = cv2.threshold(gray, 0, 255, cv2.THRESH\_BINARY\_INV | cv2.THRESH\_OTSU)  
  
contours, \_ = cv2.findContours(thresh\_img, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)  
contours\_filtered = []  
for con in contours:  
 area = cv2.contourArea(con)  
 if area < 1000:  
 contours\_filtered.append(con)  
  
cv2.drawContours(thresh\_img, contours\_filtered, -1, 255, -1)  
dist = ndi.distance\_transform\_edt(thresh\_img)  
dist\_copy = dist.copy()  
  
local\_max = peak\_local\_max(dist, indices=False, min\_distance=20, labels=thresh\_img)  
markers = ndi.label(local\_max, structure=np.ones((3, 3)))[0]  
  
watershed\_img = watershed(-dist, markers, mask=thresh\_img)  
titles = ['Original image', 'Binary Image', 'Watershed']  
images = [img, thresh\_img, watershed\_img]  
  
fig = plt.gcf()  
fig.set\_size\_inches(8, 6)  
  
for i, img in enumerate(images):  
 plt.subplot(3, 1, i + 1)  
 plt.imshow(img, "jet" if i == 2 else "gray")  
 plt.title(titles[i])  
 plt.xticks([]), plt.yticks([])  
  
plt.show()

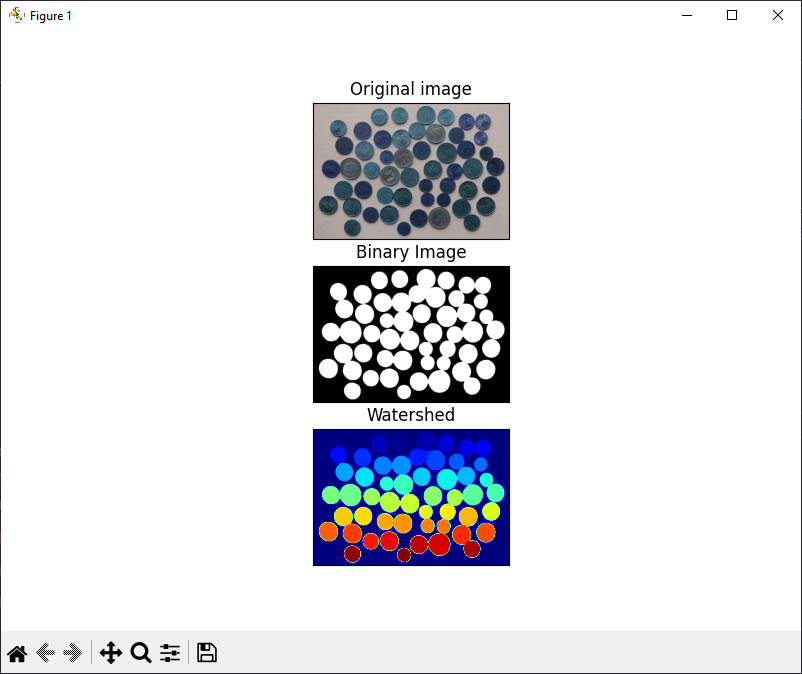


Рис.8.15 – Результат пошуку монет на іншій фотографії

Розрізнення монет по їх ціні виявилось на даний момент неможливим з урахуванням часу, що є в наявності для виконання завдання, та зображення кутів, яке не дає можливості знайти точні співпадіння для класифікації монет.

***Висновок:*** протягом виконання завдань лабораторної роботи було отримано навички з обробки зображення використовуючи бібліотеку OpenCV та мову програмування Python.

Github: https://github.com/mikrorobot/Python\_AI