**Department of Computing**

**EE433: Digital Image Processing**

**Class: BSCS-11C**

**Assignment 1**

**Deadline: 27-09-2023**

**Instructor: Asad Khan**

Group Members

|  |  |
| --- | --- |
| Name | CMS |
| Hamaz Hamza | 365670 |
| Omer Farooq | 377187 |
| Syed Hamza Ali Shah | 396358 |

**Objectives:**

* Learning and practically using OpenCV using python
* Using built-in OpenCV functions to manipulate Images
* Learning and implementing object segmentation by image processing

**Repository Link:** [DIP Assignment 1](https://github.com/Hamaz-Hamza/Digital-Image-Processing-Assignment1)

Image Chosen:



Task 1: resizing image to 256x256:

Code:

import cv2 as opencv

imageLocation = "image.bmp"

# read image from file

image = opencv.imread(imageLocation)

# resize the image

image = opencv.resize(image, (256, 256))

# display the image

opencv.imshow("image",image)

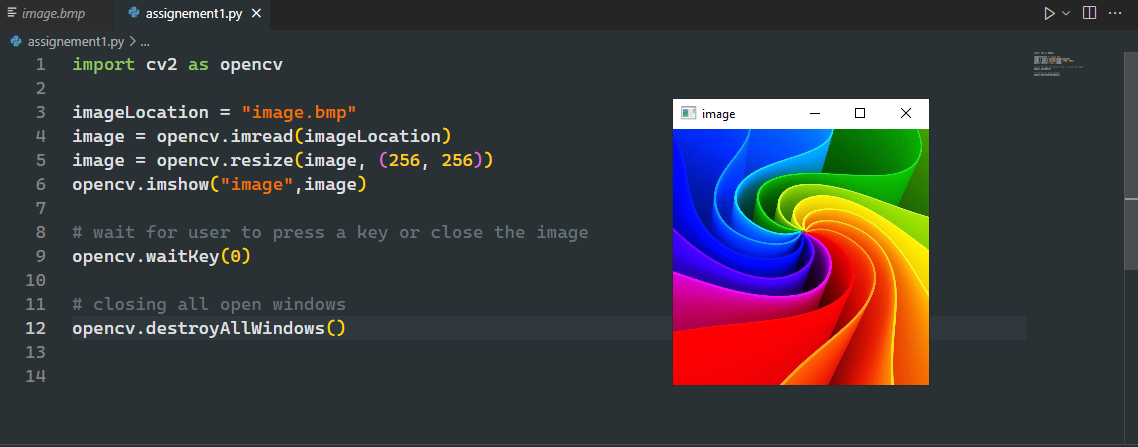
# wait for user to press a key or close the image

opencv.waitKey(0)

# closing all open windows

opencv.destroyAllWindows()

Result:



Task 2: Convert the RGB image to grayscale

Code:

import cv2 as opencv

imageLocation = "image.bmp"

# read image from file

image = opencv.imread(imageLocation)

# resize the image

image = opencv.resize(image, (256, 256))

# rgb to grayscale conversion

image = opencv.cvtColor(image, opencv.COLOR\_BGR2GRAY)

# display the image

opencv.imshow("image",image)

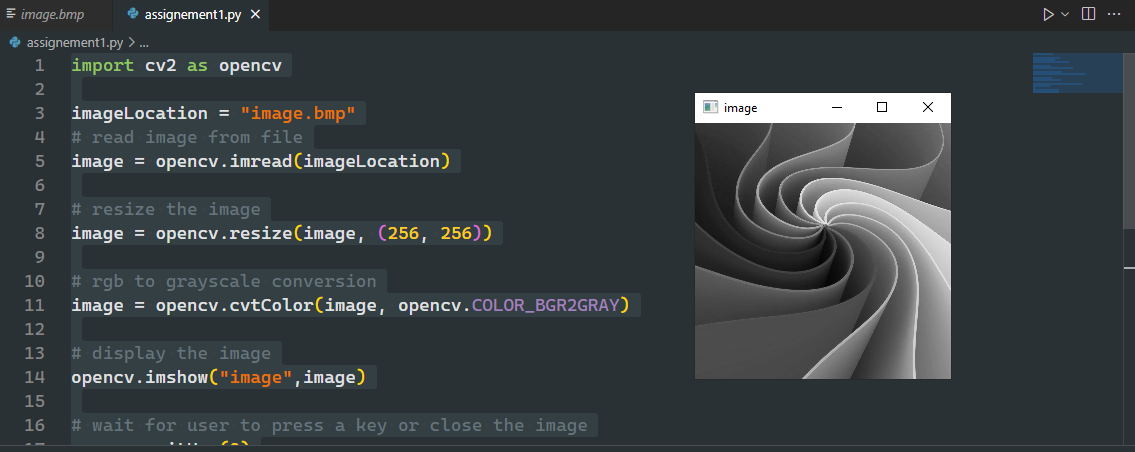
# wait for user to press a key or close the image

opencv.waitKey(0)

# closing all open windows

opencv.destroyAllWindows()

Result:



Task 3: Convert the RGB image to binary

Code:

import cv2 as opencv

imageLocation = "image.bmp"

# read image from file

image = opencv.imread(imageLocation)

# resize the image

image = opencv.resize(image, (256, 256))

# convert to grayscale

image = opencv.cvtColor(image, opencv.COLOR\_BGR2GRAY)

# convert to binary

ret, bw\_img = opencv.threshold(image, 127, 255, opencv.THRESH\_BINARY)

# display the image

opencv.imshow("image",bw\_img)

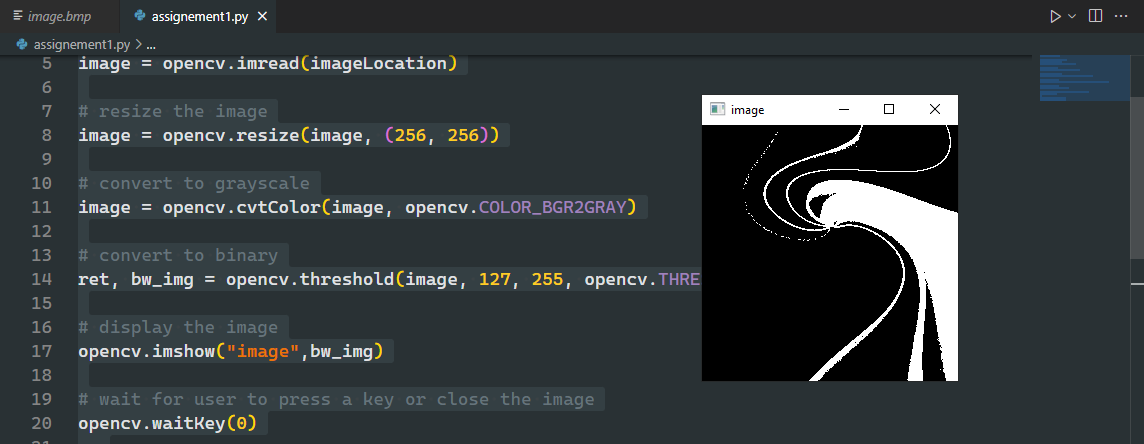
# wait for user to press a key or close the image

opencv.waitKey(0)

# closing all open windows

opencv.destroyAllWindows()

Result:



Task 4: Image Segmentation

Part 1: First we will convert the image to grayscale and then blur the image. This will help to separate the objects. Otherwise regions inside the coins will also become segmented.

Code:

import cv2

def ShowImage(image):

    cv2.imshow("result", image)

    cv2.waitKey(0)

    cv2.destroyAllWindows()

# load image

img = cv2.imread("task4.png")

# convert to grayscale

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

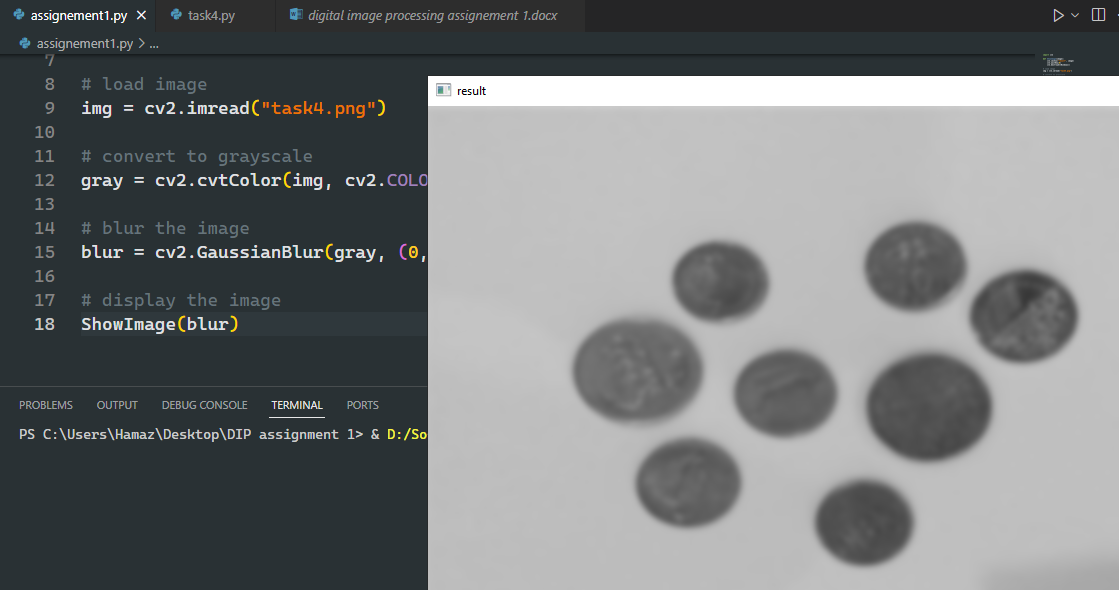
# blur the image

blur = cv2.GaussianBlur(gray, (0,0), sigmaX=3, sigmaY=3)

# display the image

ShowImage(blur)

Result:



Part 2: Secondly we will apply ***Otsu’s binarization process*** to simplify the image.

***Otsu’s binarization process:*** Otsu’s binarization is a technique used in image processing to separate the foreground and background of an image into two distinct classes. This is done by finding the optimal threshold value that maximizes the variance between the two classes.

Code:

import cv2

def ShowImage(image):

    cv2.imshow("result", image)

    cv2.waitKey(0)

    cv2.destroyAllWindows()

# load image

img = cv2.imread("task4.png")

# convert to grayscale

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

# blur the image

blur = cv2.GaussianBlur(gray, (0,0), sigmaX=3, sigmaY=3)

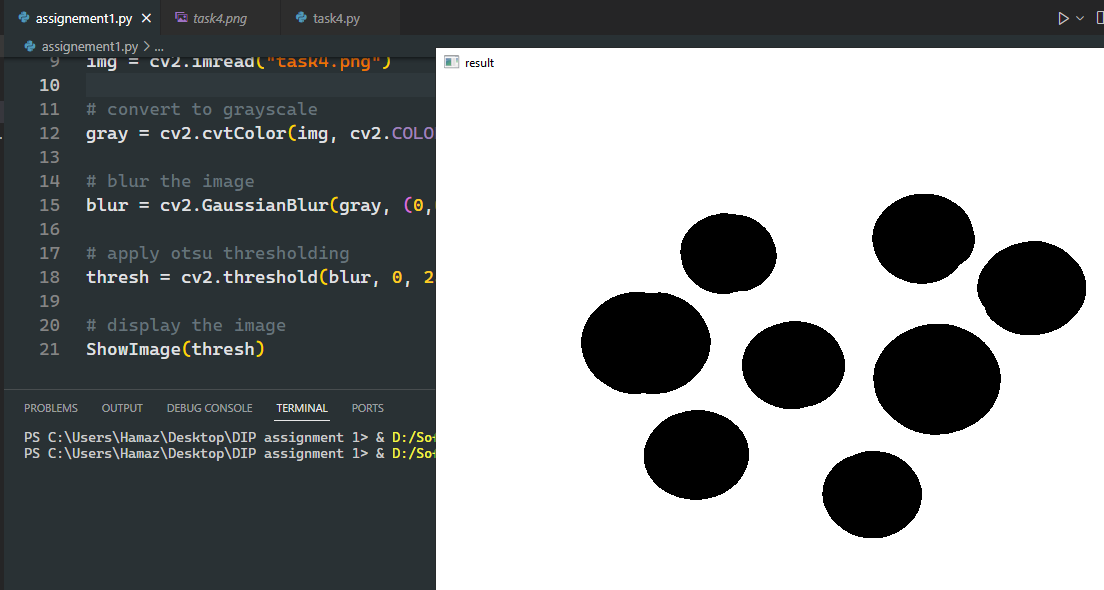
# apply otsu thresholding

thresh = cv2.threshold(blur, 0, 255, cv2.THRESH\_BINARY+cv2.THRESH\_OTSU)[1]

# display the image

ShowImage(thresh)

Result:



Part 3: Find the contours (sharp boundaries) in the image.

Code:

import cv2

def ShowImage(image):

    cv2.imshow("result", image)

    cv2.waitKey(0)

    cv2.destroyAllWindows()

# load image

img = cv2.imread("task4.png")

# convert to grayscale

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

# blur the image

blur = cv2.GaussianBlur(gray, (0,0), sigmaX=3, sigmaY=3)

# apply otsu thresholding

thresh = cv2.threshold(blur, 0, 255, cv2.THRESH\_BINARY+cv2.THRESH\_OTSU)[1]

# find contours(boundaries between background and foreground)

contours,h = cv2.findContours(thresh,1,2)

# show contours

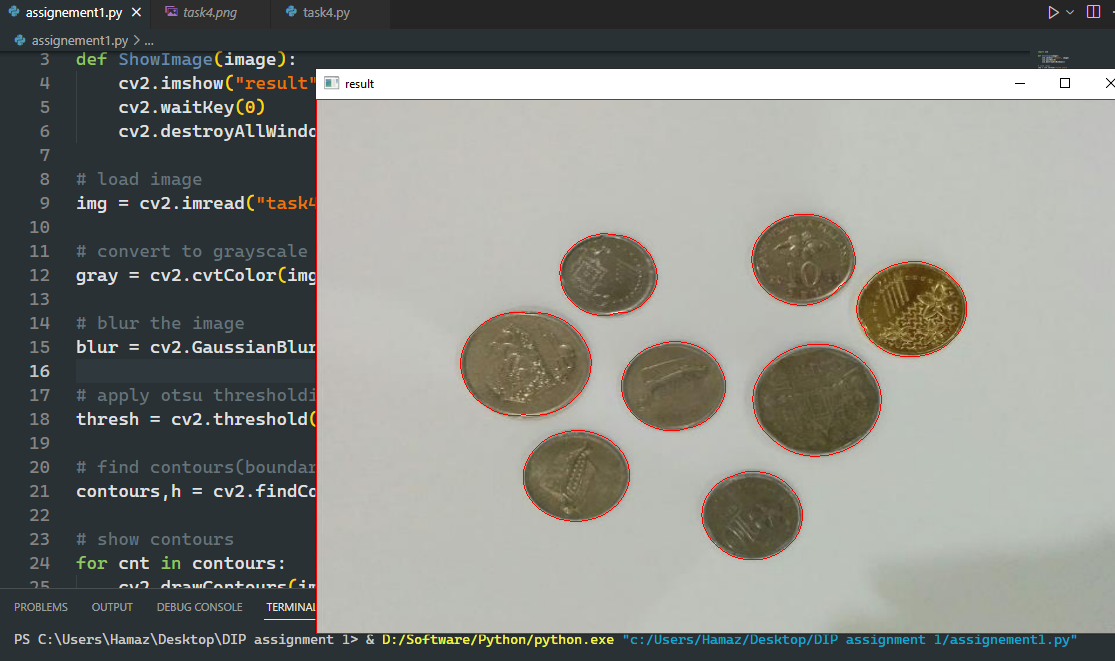
for cnt in contours:

    cv2.drawContours(img,[cnt],0,(0,0,255),1)

# display image

ShowImage(img)

Result:



Part 4: Count the number of contours and subtract one (because we don’t want to count the background) to get the total number of foreground objects

Code:

import cv2

def ShowImage(image):

    cv2.imshow("result", image)

    cv2.waitKey(0)

    cv2.destroyAllWindows()

# load image

img = cv2.imread("task4.png")

# convert to grayscale

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

# blur the image

blur = cv2.GaussianBlur(gray, (0,0), sigmaX=3, sigmaY=3)

# apply otsu thresholding

thresh = cv2.threshold(blur, 0, 255, cv2.THRESH\_BINARY+cv2.THRESH\_OTSU)[1]

# find contours(boundaries between background and foreground)

contours,h = cv2.findContours(thresh,1,2)

print("Number of objects in image = " + str(len(contours)-1))

Result:

