

**School of Electrical Engineering and Computer Sciences (SEECS)**

**Name**: Mohammad Ahmad Safvi

**CMS** **ID**: 384629

**Class**: BSCS 11-B

**Submitted** **to**: Asad Khan

**Date**: 24/09/2023

Ahmed Safvi | [Course Title] | [Date]

Digital Image Processing Assignment Report

pROCESSING AN IMAGE AND COUNTING OBJECTS

# 

# Original Image:

# Task 1: Image Resizing

### Resize the input image to a fixed resolution of 256x256 pixels.

Steps:

1. Import cv2
2. Load the image into a file (and optionally printing its details)
3. Resizing it using the cv2.resize() function
4. Printing details of the new image

Code:

*#loading image*

img = cv2.imread("curfur.jpg")

cv2.imshow('image', img)

print(f"\nimg dimensions = {img.shape} \nimg height = {img.shape[0]} \nimg width = {img.shape[1]} \nimg channels = {img.shape[2]}")

cv2.waitKey(0)

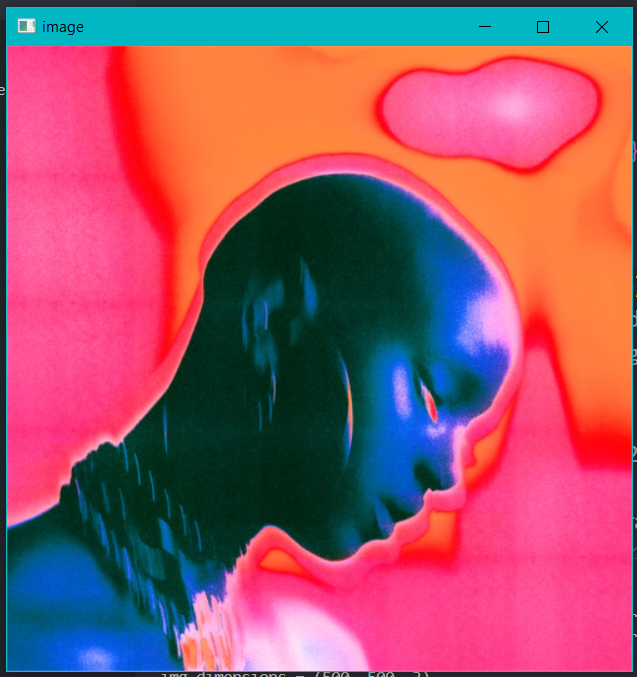
*#resizing image*

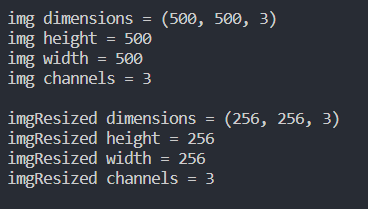
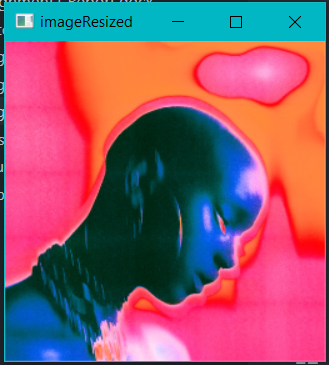
imgResized = cv2.resize(img, (256, 256))

cv2.imshow('imageResized', imgResized)

print(f"\nimgResized dimensions = {imgResized.shape} \nimgResized height = {imgResized.shape[0]} \nimgResized width = {imgResized.shape[1]} \nimgResized channels = {imgResized.shape[2]}")

cv2.waitKey(0)



Output:

# Task 2: RGB to Grayscale Conversion

### Convert the RGB image to grayscale

Steps:

Apply the cv2.cvColor() function with the arguments being the original file and cv2.COLOR\_BGR2GRAY

Code:

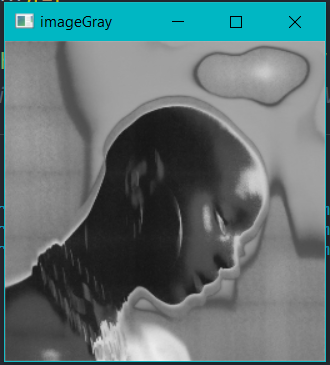
imgGray = cv2.cvtColor(imgResized, cv2.COLOR\_BGR2GRAY)

cv2.imshow("imageGray", imgGray)

print(f"\nimgGray dimensions = {imgGray.shape} \nimgGray height = {imgGray.shape[0]} \nimgGray width = {imgGray.shape[1]}")

*# print(f"\nimgGray dimensions = {imgGray.shape} \nimgGray height = {imgGray.shape[0]} \nimgGray width = {imgGray.shape[1]} \nimgGray channels = {imgGray.shape[2]}")*

cv2.waitKey(0)

Output:

# Task 3: RGB to Binary Conversion

### •Convert the RGB image to a binary image

Steps:

1. First convert the image to grayscale using the aforementioned steps
2. Apply a threshold at which the intensity is either maxed or minned from the greyscale image using the cv2.threshold() function

Code:

imgGray = cv2.cvtColor(imgResized, cv2.COLOR\_BGR2GRAY)

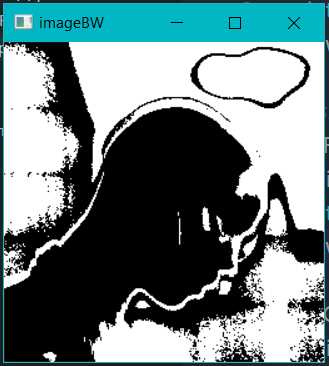
cv2.imshow("imageGray", imgGray)

cv2.waitKey(0)

imgBW = cv2.threshold(imgGray, 128, 255, cv2.THRESH\_BINARY)[1]

cv2.imshow("imageBW", imgBW)

cv2.waitKey(0)



Output:

# Task 4: Coin Counting and Segmentation

### •Segment the foreground objects (coins) from the RGB Image

Steps:

1. First we convert the image to Binary using the aforementioned steps, successfully segmenting the image
2. Since the cv2.findContours() function finds white on top of a black background, we invert the binary image using cv2.bitwise\_not()
3. And then apply the cv2.findContours() function
4. We then initialize a count variable, and for each contour found that has a greater area than 100, we increment the count variable
5. And also draw a rectangle around the found contour
6. Hence, we have a count of the number of coins and bounding boxes around them

Code:

*################# Coin Counting ################*

*#loading  img*

coins = cv2.imread("coins.png")

cv2.imshow("coins", coins)

cv2.waitKey(0)

*#graying*

coinsGray = cv2.cvtColor(coins, cv2.COLOR\_BGR2GRAY)

*#threshing*

coinsBW = cv2.threshold(coinsGray, 128, 255, cv2.THRESH\_BINARY)[1]

*#inverting*

coinsInv = cv2.bitwise\_not(coinsBW)

*#countouring*

contours, hierarchy = cv2.findContours(coinsInv, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

count = 0

for contour in contours:

        if cv2.contourArea(contour) > 100:

                count+=1

                x, y, w, h = cv2.boundingRect(contour)

                cv2.rectangle(coins, (x, y), (x + w, y + h), (0, 255, 0), 3)

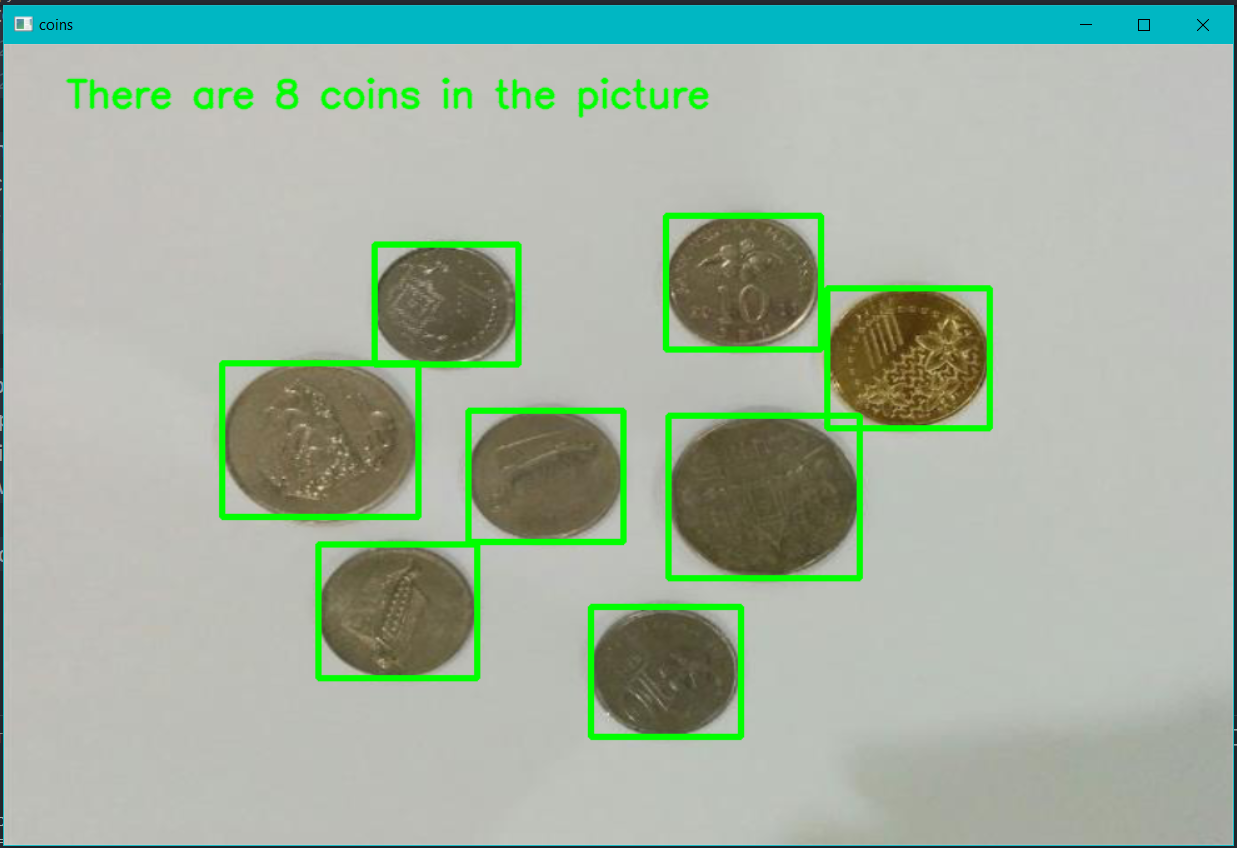
output = f"There are {count} coins in the picture"

cv2.putText(coins, output, (50, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 255, 0), 2, cv2.LINE\_AA)

cv2.imshow("coins", coins)

cv2.waitKey(0)

cv2.destroyAllWindows()



GitHub Link: https://github.com/asafvi/DIP-Assignment-1