Task

On

OpenCV

**Course**: Artificial Intelligence

(Machine Learning & Deep Learning)

Name: Abbas Shafi

Father Name: Dr Muhammad Shafi

Week: 20

Submitted to: Sir Syed Nazir Afridi



National Vocational & Technical Training Commission

National Center For Big Data & Cloud Computing

University of Engineering & Technology

Peshawar

[2]:

**import cv2**

**import numpy as np**

image = cv2.imread('E:\Speech to Text\Hasan-Ali.jpg')

# Color Spaces

## Let’s look at the individual color levels for the first pixel (0,0)

[4]:

[8]:

12 15 29

(234, 351, 3)

Let’s see what happens when we convert it to grayscale

[9]:

(234, 351)

19

It’s now only 2 dimensions. Each pixel coordinate has only one value (previously 3) with a range of 0 to 255

[9]: 19

[3]:

## Another useful color space is HSV

Infact HSV is very useful in color filtering.

hsv\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2HSV) cv2.imshow('Orignl', image)

cv2.imshow('HSV image', hsv\_image) cv2.imshow('Hue channel', hsv\_image[:, :, 0])

cv2.imshow('Saturation channel', hsv\_image[:, :, 1]) cv2.imshow('Value channel', hsv\_image[:, :, 2])

cv2.waitKey() cv2.destroyAllWindows()

## Let’s now explore lookng at individual channels in an RGB image

[5]:

image = cv2.imread('E:\Speech to Text\Hasan-Ali.jpg')

*# OpenCV's 'split' function splites the image into each color index*

B, G, R = cv2.split(image)

print(B.shape) cv2.imshow("Red", R) cv2.imshow("Green", G) cv2.imshow("Blue", B) cv2.waitKey(0) cv2.destroyAllWindows()

*# Let's re-make the original image,* merged = cv2.merge([B, G, R]) cv2.imshow("Merged", merged) cv2.waitKey(0) cv2.destroyAllWindows()

(800, 1200)

[7]:

[6]:

zeros = np.zeros(image.shape[:2], dtype = "uint8")

cv2.imshow("Red", cv2.merge([zeros, zeros, R])) cv2.imshow("Green", cv2.merge([zeros, G, zeros])) cv2.imshow("Blue", cv2.merge([B, zeros, zeros]))

cv2.waitKey(0) cv2.destroyAllWindows()

[ ]:

# Contours

[ ]:

*# Let's load a simple image with 3 black squares*

image = cv2.imread('E:\Speech to Text\Blac.jpg') *#shapes\_donut*

cv2.imshow('Input Image', image) cv2.waitKey(0)

*# Grayscale*

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

*# Find Canny edges*

edged = cv2.Canny(gray, 30, 200) cv2.imshow('Canny Edges', edged) cv2.waitKey(0)

*# Finding Contours*

*# Use a copy of your image e.g. edged.copy(), since findContours alters the*␣

*‹→image*

contours, hierarchy = cv2.findContours(edged, cv2.RETR\_EXTERNAL, cv2.

*‹→*CHAIN\_APPROX\_SIMPLE) *#retr\_list* cv2.imshow('Canny Edges After Contouring', edged) cv2.waitKey(0)

print("Number of Contours found = " + str(len(contours)))

*# Draw all contours*

*# Use '-1' as the 3rd parameter to draw all*

cv2.drawContours(image, contours,-1, (0,0,100), 1) *# second BGR and third for*␣

*‹→thickness*

cv2.imshow('Contours', image) cv2.waitKey(0) cv2.destroyAllWindows()

[ ]:

*# Load our image*

image = cv2.imread('E:\Speech to Text\Blac.jpg') cv2.imshow('Original image' , image) cv2.waitKey(0)

*# Create a black image with same dimensions as our loaded image* blank\_image = np.zeros((image.shape[0], image.shape[1], 3)) cv2.imshow('blank\_imagee', blank\_image)

cv2.waitKey(0)

*# Create a copy of our original image*

orginal\_image = image

*# Grayscale our image*

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

*# Find Canny edges*

edged = cv2.Canny(gray, 50, 200) cv2.imshow('1 - Canny Edges', edged) cv2.waitKey(0)

*# Find contours and print how many were found*

contours, hierarchy = cv2.findContours(edged.copy(), cv2.RETR\_EXTERNAL, cv2.

*‹→*CHAIN\_APPROX\_NONE)

print ("Number of contours found = ", len(contours))

*#Draw all contours*

cv2.drawContours(blank\_image, contours, -1, (0,255,0), 3) cv2.imshow('2 - All Contours over blank image', blank\_image) cv2.waitKey(0)q

*# Draw all contours over blank image* cv2.drawContours(image, contours, -1, (0,255,0), 3) cv2.imshow('3 - All Contours', image) cv2.waitKey(0)

cv2.destroyAllWindows()