TASK

QUESTION NO: 3

PROBLEM STATEMENT:

Understand the given video clip and Determine the Pixel Variations across the Video using Machine Learning Algorithms in C++ from scratch. Represent the Variations in Graph with RGB values and interpret your results.

SOURCE CODE:

import pandas as pd

import numpy as np

import numpy as np

import cv2

cap = cv2.VideoCapture("C:/Users/Krishnadev/Downloads/Question 3/DS-IQ-003-PixelVariation-Video")

while(cap.isOpened()):

# Capture frame-by-frame

ret, frame = cap.read() # ret = 1 if the video is captured; frame is the image

# Our operations on the frame come here

img = cv2.flip(frame,1) # flip left-right

img = cv2.flip(img,0) # flip up-down

# Display the resulting image

if cv2.waitKey(1) & 0xFF == ord('q'): # press q to quit

break

# When everything done, release the capture

cap.release()

cv2.destroyAllWindows()

#SAVING THE VIDEO

# create writer object

fileName='output.avi' # change the file name if needed

imgSize=(640,480)

frame\_per\_second=30.0

writer = cv2.VideoWriter(fileName, cv2.VideoWriter\_fourcc(\*"MJPG"), frame\_per\_second,imgSize)

cap = cv2.VideoCapture(0)

while(cap.isOpened()):

ret, frame = cap.read()

if ret==True:

writer.write(frame) # save the frame into video file

if cv2.waitKey(1) & 0xFF == ord('q'): # press q to quit

break

else:

break

# Release everything if job is finished

cap.release()

writer.release()

cv2.destroyAllWindows()

#LOADING AND PLAYING THE VIDEO

fileName='output.avi' # change the file name if needed

cap = cv2.VideoCapture(fileName) # load the video

while(cap.isOpened()): # play the video by reading frame by frame

ret, frame = cap.read()

if ret==True:

# optional: do some image processing here

cv2.imshow('frame',frame) # show the video

if cv2.waitKey(1) & 0xFF == ord('q'):

break

else:

break

cap.release()

cv2.destroyAllWindows()

# COLOR TRANSFORMATION:

cap = cv2.VideoCapture(0)

while(True):

# Capture frame-by-frame

ret, frame = cap.read()

# Our operations on the frame come here

#img = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

img = cv2.cvtColor(frame,cv2.COLOR\_BGR2GRAY) # BGR color to gray level

# Display the resulting image

if cv2.waitKey(1) & 0xFF == ord('q'): # press q to quit

break

# When everything done, release the capture

cap.release()

cv2.destroyAllWindows()

# EDGE DETECTION AND SMOOTHING:

kernelSize=21 # Kernel Bluring size

# Edge Detection Parameter

parameter1=20

parameter2=60

intApertureSize=1

cap = cv2.VideoCapture(0)

while(True):

# Capture frame-by-frame

ret, frame = cap.read()

# Our operations on the frame come here

frame = cv2.GaussianBlur(frame, (kernelSize,kernelSize), 0, 0)

frame = cv2.Canny(frame,parameter1,parameter2,intApertureSize) # Canny edge detection

#frame = cv2.Laplacian(frame,cv2.CV\_64F) # Laplacian edge detection

#frame = cv2.Sobel(frame,cv2.CV\_64F,1,0,ksize=kernelSize) # X-direction Sobel edge detection

#frame = cv2.Sobel(frame,cv2.CV\_64F,0,1,ksize=kernelSize) # Y-direction Sobel edge detection

# Display the resulting frame

cv2.imshow('Canny',frame)

if cv2.waitKey(1) & 0xFF == ord('q'): # press q to quit

break

# When everything done, release the capture

cap.release()

cv2.destroyAllWindows()

# OPTICAL FLOW:

cap = cv2.VideoCapture(0)

ret, frame1 = cap.read()

prvs = cv2.cvtColor(frame1,cv2.COLOR\_BGR2GRAY)

hsv = np.zeros\_like(frame1)

hsv[...,1] = 255

while(1):

ret, frame2 = cap.read()

# Our operations on the frame come here

next = cv2.cvtColor(frame2,cv2.COLOR\_BGR2GRAY)

flow = cv2.calcOpticalFlowFarneback(prvs,next, None, 0.5, 3, 15, 3, 5, 1.2, 0)

mag, ang = cv2.cartToPolar(flow[...,0], flow[...,1])

hsv[...,0] = ang\*180/np.pi/2

hsv[...,2] = cv2.normalize(mag,None,0,255,cv2.NORM\_MINMAX)

bgr = cv2.cvtColor(hsv,cv2.COLOR\_HSV2BGR)

prvs = next

# Display the resulting frame

cv2.imshow('Optical Flow Aura',bgr)

if cv2.waitKey(2) & 0xFF == ord('q'): # press q to quit

break

# When everything done, release the capture

cap.release()

cv2.destroyAllWindows()

# IMAGE DIFFERENCE : MOTION DETECTION

color=(255,0,0)

thickness=2

cap = cv2.VideoCapture(0)

while(True):

# Capture two frames

ret, frame1 = cap.read() # first image

time.sleep(1/25) # slight delay

ret, frame2 = cap.read() # second image

Img1 = cv2.absdiff(frame1,frame2) # image difference

# get theshold image

gray = cv2.cvtColor(img1,cv2.COLOR\_BGR2GRAY)

blur = cv2.GaussianBlur(gray,(21,21),0)

ret,thresh = cv2.threshold(blur,200,255,cv2.THRESH\_OTSU)

# combine frame and the image difference

img2 = cv2.addWeighted(frame1,0.9,img1,0.1,0)

# get contours and set bounding box from contours

img3, contours, hierarchy = cv2.findContours(thresh,cv2.RETR\_TREE,cv2.CHAIN\_APPROX\_NONE)

if len(contours) != 0:

for c in contours:

rect = cv2.boundingRect(c)

height, width = img3.shape[:2]

if rect[2] > 0.2\*height and rect[2] < 0.7\*height and rect[3] > 0.2\*width and rect[3] < 0.7\*width:

x,y,w,h = cv2.boundingRect(c) # get bounding box of largest contour

img4=cv2.drawContours(img2, c, -1, color, thickness)

img5 = cv2.rectangle(img2,(x,y),(x+w,y+h),(0,0,255),2) # draw red bounding box in img

else:

img5=img2

else:

img5=img2

# Display the resulting image

cv2.imshow('Motion Detection by Image Difference',img2)

if cv2.waitKey(1) & 0xFF == ord('q'): # press q to quit

break

# When everything done, release the capture

cap.release()

cv2.destroyAllWindows()

# BACKGROUND SUBTRACTION

alpha=0.999

isFirstTime=True

cap = cv2.VideoCapture(0)

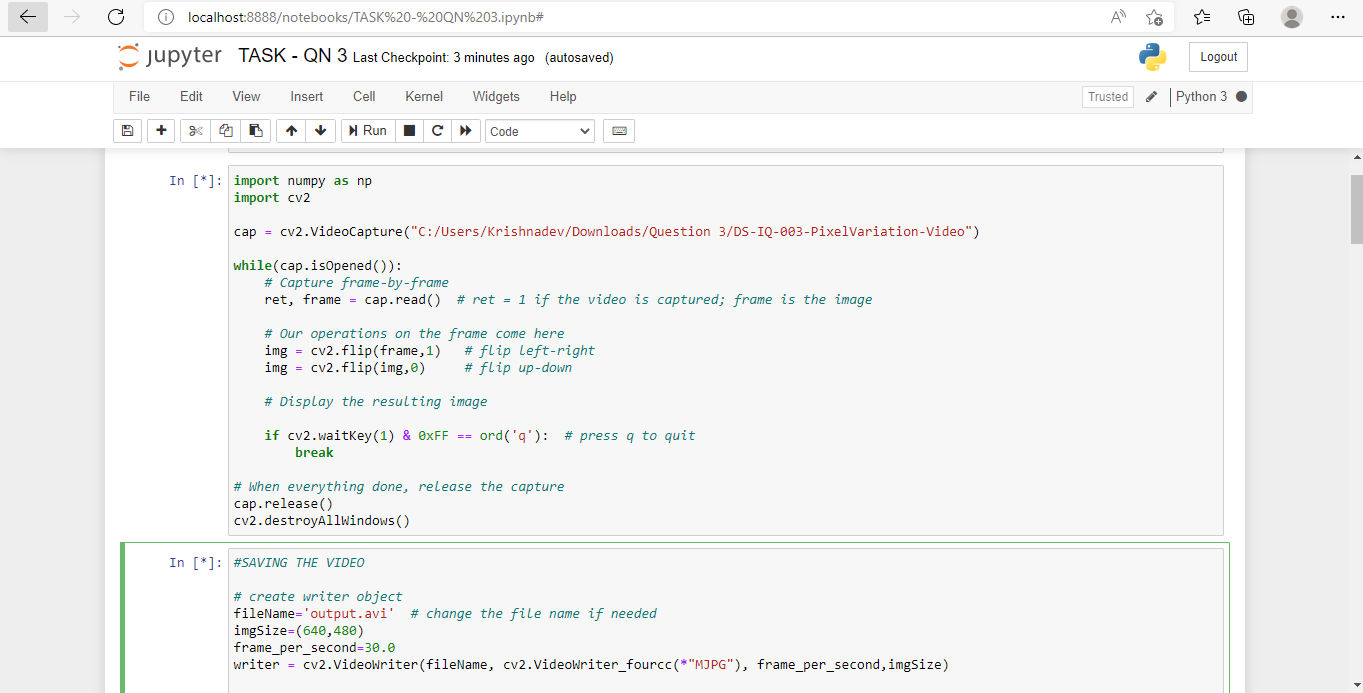
break

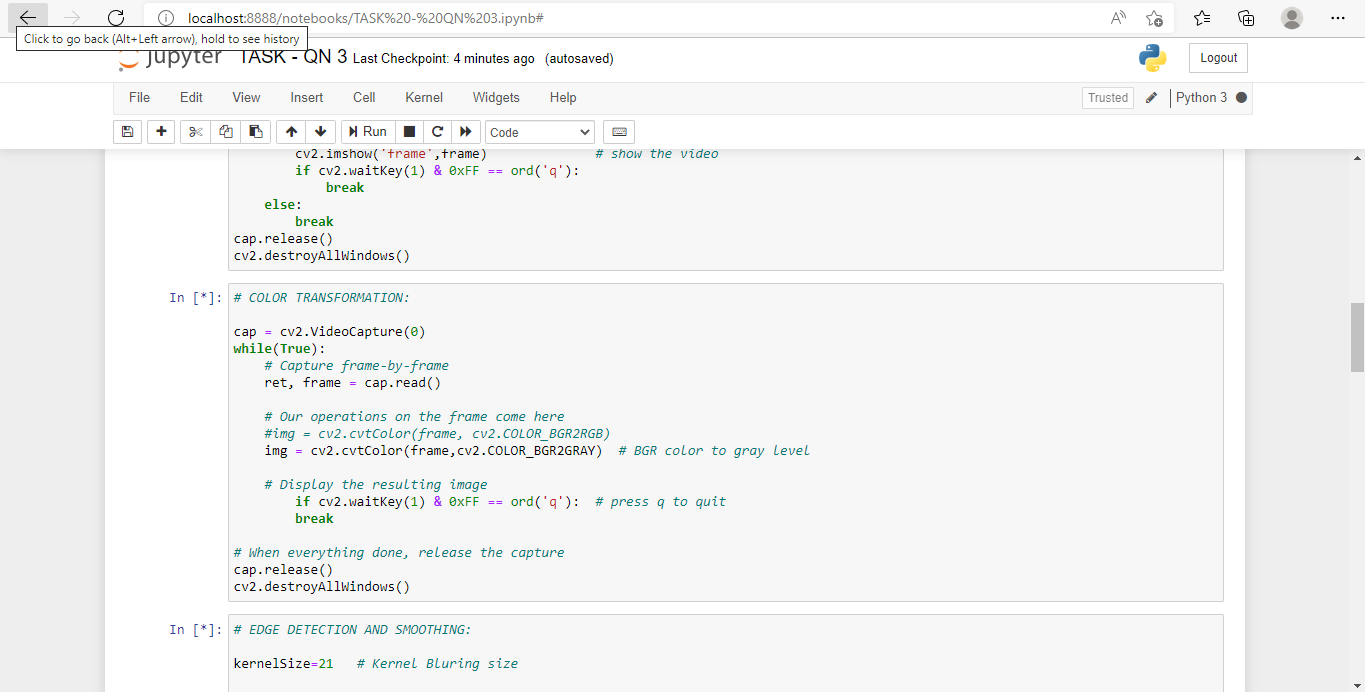
# When everything done, release the capture

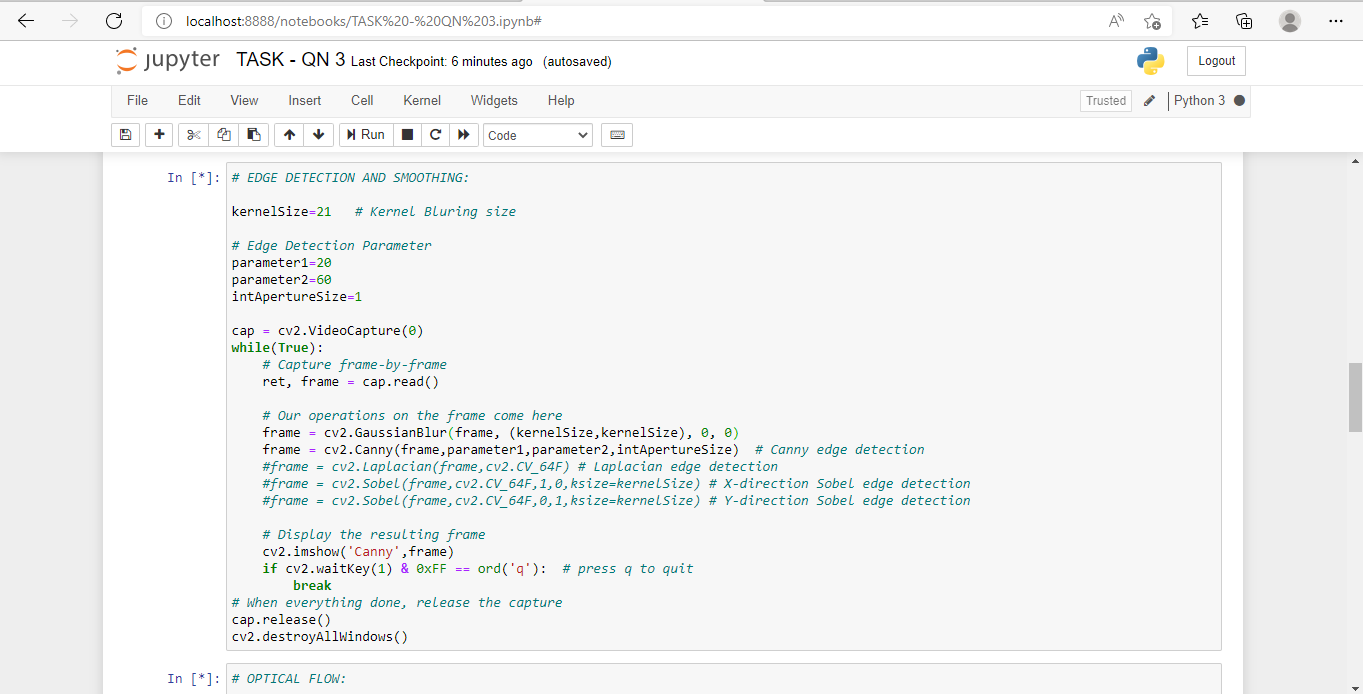
cap.release()

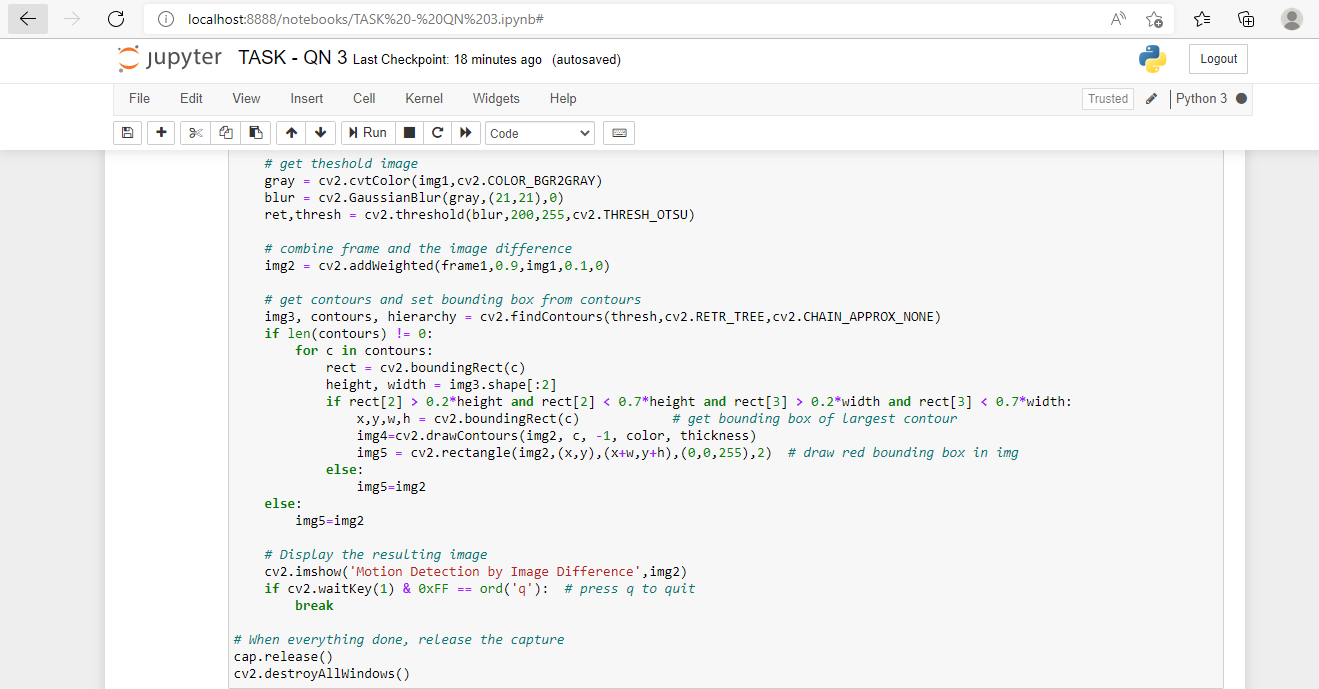
cv2.destroyAllWindows()

OUTPUT:









EXPLANATION:

First analyse the given video clip. With the relevant code do the pixel variation completely for the given video. Then do the relevant changes in the video like color transformation etc. After doing pixel variation, use the relevant machine learning algorithm to form the pixel variation. Then visualise the given data with the help of plots and graphs. Visualise the plot with RGB values and interpret the results. In the graphs, the variations must also be mentioned with respect to RGB values.