import cv2

import numpy as np

def empty(a):

    pass

def stackImages(scale,imgArray):

    rows = len(imgArray)

    cols = len(imgArray[0])

    rowsAvailable = isinstance(imgArray[0], list)

    width = imgArray[0][0].shape[1]

    height = imgArray[0][0].shape[0]

    if rowsAvailable:

        for x in range ( 0, rows):

            for y in range(0, cols):

                if imgArray[x][y].shape[:2] == imgArray[0][0].shape [:2]:

                    imgArray[x][y] = cv2.resize(imgArray[x][y], (0, 0), None, scale, scale)

                else:

                    imgArray[x][y] = cv2.resize(imgArray[x][y], (imgArray[0][0].shape[1], imgArray[0][0].shape[0]), None, scale, scale)

                if len(imgArray[x][y].shape) == 2: imgArray[x][y]= cv2.cvtColor( imgArray[x][y], cv2.COLOR\_GRAY2BGR)

        imageBlank = np.zeros((height, width, 3), np.uint8)

        hor = [imageBlank]\*rows

        hor\_con = [imageBlank]\*rows

        for x in range(0, rows):

            hor[x] = np.hstack(imgArray[x])

        ver = np.vstack(hor)

    else:

        for x in range(0, rows):

            if imgArray[x].shape[:2] == imgArray[0].shape[:2]:

                imgArray[x] = cv2.resize(imgArray[x], (0, 0), None, scale, scale)

            else:

                imgArray[x] = cv2.resize(imgArray[x], (imgArray[0].shape[1], imgArray[0].shape[0]), None,scale, scale)

            if len(imgArray[x].shape) == 2: imgArray[x] = cv2.cvtColor(imgArray[x], cv2.COLOR\_GRAY2BGR)

        hor= np.hstack(imgArray)

        ver = hor

    return ver

path = 'Resources/lambo.png'

cv2.namedWindow("TrackBars")

cv2.resizeWindow("TrackBars",640,240)

cv2.createTrackbar("Hue Min","TrackBars",0,179,empty)

cv2.createTrackbar("Hue Max","TrackBars",19,179,empty)

cv2.createTrackbar("Sat Min","TrackBars",110,255,empty)

cv2.createTrackbar("Sat Max","TrackBars",240,255,empty)

cv2.createTrackbar("Val Min","TrackBars",153,255,empty)

cv2.createTrackbar("Val Max","TrackBars",255,255,empty)

while True:

    img = cv2.imread(path)

    imgHSV = cv2.cvtColor(img,cv2.COLOR\_BGR2HSV)

    h\_min = cv2.getTrackbarPos("Hue Min","TrackBars")

    h\_max = cv2.getTrackbarPos("Hue Max", "TrackBars")

    s\_min = cv2.getTrackbarPos("Sat Min", "TrackBars")

    s\_max = cv2.getTrackbarPos("Sat Max", "TrackBars")

    v\_min = cv2.getTrackbarPos("Val Min", "TrackBars")

    v\_max = cv2.getTrackbarPos("Val Max", "TrackBars")

    print(h\_min,h\_max,s\_min,s\_max,v\_min,v\_max)

    lower = np.array([h\_min,s\_min,v\_min])

    upper = np.array([h\_max,s\_max,v\_max])

    mask = cv2.inRange(imgHSV,lower,upper)

    imgResult = cv2.bitwise\_and(img,img,mask=mask)

    # cv2.imshow("Original",img)

    # cv2.imshow("HSV",imgHSV)

    # cv2.imshow("Mask", mask)

    # cv2.imshow("Result", imgResult)

    imgStack = stackImages(0.6,([img,imgHSV],[mask,imgResult]))

    cv2.imshow("Stacked Images", imgStack)

    cv2.waitKey(1)

# get image

import cv2

print("Package Imported")

# Read Images

img = cv2.imread("images/nithi.jpg")

cv2.imshow("Output",img)

cv2.waitKey(0)

#get video.

import cv2

cap = cv2.VideoCapture("images/hasle.mp4")

while True:

    success, img = cap.read()

    cv2.imshow("Video", img)

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

        break

# get webcam

import cv2

cap = cv2.VideoCapture(0)

cap.set(3,640)

cap.set(4,480)

cap.set(10,100)

while True:

    success, img = cap.read()

    cv2.imshow("Video", img)

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

        break

#img processing

import cv2

import numpy as np

img = cv2.imread("images/nithi.jpg")

kernal = 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,kernal,iterations=1)

imgEroted = cv2.erode(imgDilation,kernal,iterations=1)

cv2.imshow("Gray Image",imgGray)

cv2.imshow("Blue Image",imgBlur)

cv2.imshow("Canny Image",imgCanny)

cv2.imshow("Dialation Image",imgDilation)

cv2.imshow("ErotedImage",imgEroted)

cv2.waitKey(0)

# resize Image

import cv2

import numpy as np

img = cv2.imread("images/urmi.jpg")

print(img.shape)

imgResize = cv2.resize(img,(300,200))

print(img.shape)

cv2.imshow("Image",img)

cv2.imshow("Image resize",imgResize)

cv2.waitKey(0)

# Crop  Image

import cv2

import numpy as np

img = cv2.imread("images/urmi.jpg")

print(img.shape)

#image Crop Just use matrix funtionality

imgCropped = img[00:600,600:1200]

imgResize = cv2.resize(img,(1000,500))

print(imgResize.shape)

cv2.imshow("Image",img)

# cv2.imshow("Image resize",imgResize)

cv2.imshow("Image croped",imgCropped)

cv2.waitKey(0)

# Crop  Image

import cv2

import numpy as np

img = np.zeros((512,512,3))

# print(img)

# img[:]=255,255,0

# cv2.line(img,(0,0),(300,300),(0,255,170),10)

cv2.line(img,(0,0),(img.shape[1],img.shape[0]),(0,255,170),10)

cv2.rectangle(img,(0,0),(250,350),(0,255,180),cv2.FILLED)

cv2.circle(img,(400,50),30,(255,255,0),5)

cv2.putText(img," OPENCV",(300,100),cv2.FONT\_HERSHEY\_COMPLEX,1,(0,150,0),5)

cv2.imshow("image",img)

cv2.waitKey(0)

# Crop  Image

import cv2

import numpy as np

img = np.zeros((512,512,3))

# print(img)

# img[:]=255,255,0

# cv2.line(img,(0,0),(300,300),(0,255,170),10)

cv2.line(img,(0,0),(img.shape[1],img.shape[0]),(0,255,170),10)

cv2.rectangle(img,(0,0),(250,350),(0,255,180),cv2.FILLED)

cv2.circle(img,(400,50),30,(255,255,0),cv2.FILLED)

cv2.putText(img," OPENCV",(300,100),cv2.FONT\_HERSHEY\_TRIPLEX,-1,(0,150,0),1)

cv2.imshow("image",img)

cv2.waitKey(0)

# wrap respictive

import cv2

import numpy as np

img = cv2.imread("images/urmi.jpg")

width,height = 250,350

pst1 = np.float32([[111,219],[287,188],[154,482],[352,440]])

pst2 = np.float32([[0,0],[width,0],[0,height],[width,height]])

matrix = cv2.getPerspectiveTransform(pst1 ,pst2)

imgOutput = cv2.warpPerspective(img,matrix,(width,height))

cv2.imshow("Image",img)

cv2.imshow("Output",imgOutput)

cv2.waitKey(0)

# Wromg phone camera

import cv2, time

face\_cascade = cv2.CascadeClassifier("haarcascade-frontalface\_default.xml")

video = cv2.VideoCapture(0)

address = "http//:192.168.0.101:8080/video"

video.open(address)

while True:

    check, frame = video.read()

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

    face = face\_cascade.detectMultiScale(gray,scaleFactor = 1.1,minNeighbors=5)

    for x,y,w,h in face:

        img = cv2.rectangle(frame,(x,y),(x+y,y+h),(0,255,0),3)

    cv2.imshow("gottcha",frame)

    key = cv2.waitKey(1)

    if key ==ord('q'):

        break

    video.release()

    cv2.destroyAllWindows

#  Security camera

import cv2

import winsound

cam = cv2.VideoCapture(0)

while cam.isOpened():

    ret, frame1 = cam.read()

    ret, frame2 = cam.read()

    diff = cv2.absdiff(frame1,frame2)

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

    blur = cv2.GaussianBlur(gray,(5,5),0)

    # Theresold

    \_\_, thresh= cv2.threshold(blur,20,255,cv2.THRESH\_BINARY)

    # dialation

    dialated = cv2.dilate(thresh,None,iterations=3)

    contours, \_\_ = cv2.findContours(dialated,cv2.RETR\_TREE,cv2.CHAIN\_APPROX\_SIMPLE)

    # cv2.drawContours(frame1,contours,-1,(0,255,0),2)

    for  c in contours:

        if cv2.contourArea(c) < 5000:

            continue

        x,y,w,h = cv2.boundingRect(c)

        cv2.rectangle(frame1,(x,y),(x+w,y+h),(0,255,0),2)

        winsound.Beep(500,200)

    if cv2.waitKey(10) == ord('q'):

        break

    cv2.imshow('Granny Cam', frame1)

import glob

import imagehash

from PIL import Image

my\_img\_url = './boys/rayhan1.jpg'

my\_hash = imagehash.average\_hash(Image.open(my\_img\_url))

girls = glob.glob('./girls/\*.jpg')

selected = girls[0]

accepted\_diff = 1000

for girl in girls:

    girl\_hash = imagehash.average\_hash(Image.open(girl))

    diff = girl\_hash - my\_hash

    if diff < accepted\_diff:

        selected = girl

        accepted\_diff = diff

bf\_img = Image.open(my\_img\_url)

gf\_img = Image.open(selected)

couple\_img = Image.new('RGB', (bf\_img.width + gf\_img.width, bf\_img.height))

couple\_img.paste(bf\_img, (0, 0))

couple\_img.paste(gf\_img, (bf\_img.width, 0))

couple\_img.save('my\_valentine\_day\_date.jpg')

couple\_img.show()

import cv2

import numpy as np

img = cv2.imread("images/card.png")

width,height = 350,350

pts1 = np.float32([[520,40],[605,212],[250,150],[340,330]])

pts2 = np.float32([[0,0],[width,0],[0,height],[width,height]])

matrix = cv2.getPerspectiveTransform(pts1,pts2)

imgOutput = cv2.warpPerspective(img,matrix,(width,height))

cv2.imshow("Image",img)

cv2.imshow("Output",imgOutput)

cv2.waitKey(0)

# image joining

import cv2

import numpy as np

def stackImages(scale,imgArray):

    rows = len(imgArray)

    cols = len(imgArray[0])

    rowsAvailable = isinstance(imgArray[0], list)

    width = imgArray[0][0].shape[1]

    height = imgArray[0][0].shape[0]

    if rowsAvailable:

        for x in range ( 0, rows):

            for y in range(0, cols):

                if imgArray[x][y].shape[:2] == imgArray[0][0].shape [:2]:

                    imgArray[x][y] = cv2.resize(imgArray[x][y], (0, 0), None, scale, scale)

                else:

                    imgArray[x][y] = cv2.resize(imgArray[x][y], (imgArray[0][0].shape[1], imgArray[0][0].shape[0]), None, scale, scale)

                if len(imgArray[x][y].shape) == 2: imgArray[x][y]= cv2.cvtColor( imgArray[x][y], cv2.COLOR\_GRAY2BGR)

        imageBlank = np.zeros((height, width, 3), np.uint8)

        hor = [imageBlank]\*rows

        hor\_con = [imageBlank]\*rows

        for x in range(0, rows):

            hor[x] = np.hstack(imgArray[x])

        ver = np.vstack(hor)

    else:

        for x in range(0, rows):

            if imgArray[x].shape[:2] == imgArray[0].shape[:2]:

                imgArray[x] = cv2.resize(imgArray[x], (0, 0), None, scale, scale)

            else:

                imgArray[x] = cv2.resize(imgArray[x], (imgArray[0].shape[1], imgArray[0].shape[0]), None,scale, scale)

            if len(imgArray[x].shape) == 2: imgArray[x] = cv2.cvtColor(imgArray[x], cv2.COLOR\_GRAY2BGR)

        hor= np.hstack(imgArray)

        ver = hor

    return ver

img = cv2.imread("images/nithi.jpg")

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

imgStack = stackImages(0.5,([img,imgGray,img],[img,img,img],[img,img,img]))

# imgHor = np.hstack((img,img))

# imgVer = np.vstack((img,img))

# cv2.imshow("Horizontal",imgHor)

# cv2.imshow("Vertical",imgVer)

# # cv2.imshow("Output",img)

cv2.imshow("Output",imgStack)

cv2.waitKey(0)

# tracbars

import cv2

import numpy as np

def empty(a):

    pass

def stackImages(scale,imgArray):

    rows = len(imgArray)

    cols = len(imgArray[0])

    rowsAvailable = isinstance(imgArray[0], list)

    width = imgArray[0][0].shape[1]

    height = imgArray[0][0].shape[0]

    if rowsAvailable:

        for x in range ( 0, rows):

            for y in range(0, cols):

                if imgArray[x][y].shape[:2] == imgArray[0][0].shape [:2]:

                    imgArray[x][y] = cv2.resize(imgArray[x][y], (0, 0), None, scale, scale)

                else:

                    imgArray[x][y] = cv2.resize(imgArray[x][y], (imgArray[0][0].shape[1], imgArray[0][0].shape[0]), None, scale, scale)

                if len(imgArray[x][y].shape) == 2: imgArray[x][y]= cv2.cvtColor( imgArray[x][y], cv2.COLOR\_GRAY2BGR)

        imageBlank = np.zeros((height, width, 3), np.uint8)

        hor = [imageBlank]\*rows

        hor\_con = [imageBlank]\*rows

        for x in range(0, rows):

            hor[x] = np.hstack(imgArray[x])

        ver = np.vstack(hor)

    else:

        for x in range(0, rows):

            if imgArray[x].shape[:2] == imgArray[0].shape[:2]:

                imgArray[x] = cv2.resize(imgArray[x], (0, 0), None, scale, scale)

            else:

                imgArray[x] = cv2.resize(imgArray[x], (imgArray[0].shape[1], imgArray[0].shape[0]), None,scale, scale)

            if len(imgArray[x].shape) == 2: imgArray[x] = cv2.cvtColor(imgArray[x], cv2.COLOR\_GRAY2BGR)

        hor= np.hstack(imgArray)

        ver = hor

    return ver

path = "images/car.jpg"

cv2.namedWindow("TracBars")

cv2.resizeWindow("TracBars",640,240)

cv2.createTrackbar("Hue Min","TracBars",0,179,empty)

cv2.createTrackbar("Hue Max","TracBars",49,179,empty)

cv2.createTrackbar("Sat Min","TracBars",49,255,empty)

cv2.createTrackbar("Sat max","TracBars",2255,255,empty)

cv2.createTrackbar("Val min","TracBars",38,255,empty)

cv2.createTrackbar("val max","TracBars",255,255,empty)

while True:

    img = cv2.imread(path)

    imgHsv = cv2.cvtColor(img,cv2.COLOR\_BGR2HSV)

    h\_min = cv2.getTrackbarPos("Hue Min","TracBars")

    h\_max = cv2.getTrackbarPos("Hue Max","TracBars")

    s\_min = cv2.getTrackbarPos("Sat Min","TracBars")

    s\_max = cv2.getTrackbarPos("Sat max","TracBars")

    v\_min = cv2.getTrackbarPos("Val min","TracBars")

    v\_max = cv2.getTrackbarPos("val max","TracBars")

    print(h\_min,h\_max,s\_min,s\_max,v\_min,v\_max)

    lower = np.array([h\_min,s\_min,v\_min])

    upper = np.array([h\_max,s\_max,v\_max])

    mask = cv2.inRange(imgHsv,lower,upper)

    imgResult = cv2.bitwise\_and(img,img,mask=mask)

    imgStack = stackImages(0.6,([img,imgHsv],[mask,imgResult]))

    # cv2.imshow("Original",img)

    # cv2.imshow("HSV",imgHsv)

    # cv2.imshow("Mask",mask)

    # cv2.imshow("Imgresult",imgResult)

    cv2.imshow("Stack image",imgStack)

    cv2.waitKey(1)

import cv2

import numpy as np

def stackImages(scale,imgArray):

    rows = len(imgArray)

    cols = len(imgArray[0])

    rowsAvailable = isinstance(imgArray[0], list)

    width = imgArray[0][0].shape[1]

    height = imgArray[0][0].shape[0]

    if rowsAvailable:

        for x in range ( 0, rows):

            for y in range(0, cols):

                if imgArray[x][y].shape[:2] == imgArray[0][0].shape [:2]:

                    imgArray[x][y] = cv2.resize(imgArray[x][y], (0, 0), None, scale, scale)

                else:

                    imgArray[x][y] = cv2.resize(imgArray[x][y], (imgArray[0][0].shape[1], imgArray[0][0].shape[0]), None, scale, scale)

                if len(imgArray[x][y].shape) == 2: imgArray[x][y]= cv2.cvtColor( imgArray[x][y], cv2.COLOR\_GRAY2BGR)

        imageBlank = np.zeros((height, width, 3), np.uint8)

        hor = [imageBlank]\*rows

        hor\_con = [imageBlank]\*rows

        for x in range(0, rows):

            hor[x] = np.hstack(imgArray[x])

        ver = np.vstack(hor)

    else:

        for x in range(0, rows):

            if imgArray[x].shape[:2] == imgArray[0].shape[:2]:

                imgArray[x] = cv2.resize(imgArray[x], (0, 0), None, scale, scale)

            else:

                imgArray[x] = cv2.resize(imgArray[x], (imgArray[0].shape[1], imgArray[0].shape[0]), None,scale, scale)

            if len(imgArray[x].shape) == 2: imgArray[x] = cv2.cvtColor(imgArray[x], cv2.COLOR\_GRAY2BGR)

        hor= np.hstack(imgArray)

        ver = hor

    return ver

# Countours Function

def getContours(img):

    contours,hierarchy = cv2.findContours(img,cv2.RETR\_EXTERNAL,cv2.CHAIN\_APPROX\_NONE)

    for cnt in contours:

        area = cv2.contourArea(cnt)

        print(area)

        if area > 500:

            peri = cv2.arcLength(cnt,True)

            # print(peri)

            approx = cv2.approxPolyDP(cnt,0.02\*peri,True)

            print(len(approx))

            objCor = len(approx)

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

            if objCor ==3: ObjectType = "tri"

            elif objCor == 4:

                aspRatio = w/float(h)

                if aspRatio > 0.95 and aspRatio <1.05: ObjectType = "Square"

                else:ObjectType = "Rectangle"

            elif objCor > 4 : ObjectType = "Circles"

            else: ObjectType ="None"

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

            cv2.putText(imgContour,ObjectType,(x+(w//2)-10,y+(h//2)-10),cv2.FONT\_HERSHEY\_COMPLEX,0.7,(0,0,0),2)

path = 'images/shapes.png'

img = cv2.imread(path)

imgContour = img.copy()

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

imgBlur = cv2.GaussianBlur(imgGray,(7,7),1)

imgCanny = cv2.Canny(imgBlur,50,50,)

getContours(imgCanny)

imgBlack = np.zeros\_like(img)

imgStack = stackImages(0.8,([img,imgGray,imgBlur],[imgCanny,imgContour,imgBlack]))

# cv2.imshow("Original Image",img)

# cv2.imshow("Gray",imgGray)

# cv2.imshow("Blur",imgBlur)

cv2.imshow("Image stack",imgStack)

cv2.waitKey(0)

# face detection

import cv2

faceCascade = cv2.CascadeClassifier("haarcascades/haarcascade\_frontalface\_default.xml")

img = cv2.imread("images/urmi.jpg")

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),3)

    # cv2.circle(img,(x+w)//2,(y+h)//2,15,(155,160,125),5)

cv2.imshow("Result",img)

cv2.waitKey(0)

<http://192.168.0.101:8080/video>

# Mobile Connection

import cv2

import numpy as np

cap = cv2.VideoCapture("http://192.168.0.101:8080/video")

while True:

    ret , frame = cap.read()

    cv2.imshow("Frame", frame)

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

        break

cap.release()

cv2.destroyAllWindows()