import cv2

import numpy as np

frameWidth = 640

frameHeight = 480

cap = cv2.VideoCapture(0)

cap.set(3, frameWidth)

cap.set(4, frameHeight)

def empty(a):

    pass

cv2.namedWindow("Parameters")

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

cv2.createTrackbar("Threshold1","Parameters",23,255,empty)

cv2.createTrackbar("Threshold2","Parameters",20,255,empty)

cv2.createTrackbar("Area","Parameters",5000,30000,empty)

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

def getContours(img,imgContour):

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

    for cnt in contours:

        area = cv2.contourArea(cnt)

        areaMin = cv2.getTrackbarPos("Area", "Parameters")

        if area > areaMin:

            cv2.drawContours(imgContour, cnt, -1, (255, 0, 255), 7)

            peri = cv2.arcLength(cnt, True)

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

            print(len(approx))

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

            cv2.rectangle(imgContour, (x , y ), (x + w , y + h ), (0, 255, 0), 5)

            if len(approx) == 4:

                cv2.putText(imgContour, "Rectangle" , (x + w + 20, y + 20), cv2.FONT\_HERSHEY\_COMPLEX, .7,

                        (0, 255, 0), 2)

            elif len(approx) >=8 :

                cv2.putText(imgContour, "Circle" , (x + w + 20, y + 20), cv2.FONT\_HERSHEY\_COMPLEX, .7,

                        (0, 255, 0), 2)

            cv2.putText(imgContour, "Area: " + str(int(area)), (x + w + 20, y + 45), cv2.FONT\_HERSHEY\_COMPLEX, 0.7,

                        (0, 255, 0), 2)

while True:

    success, img = cap.read()

        # converting frame(img == BGR) to HSV(hue-saturation-value)

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

    # red color

    red\_lower = np.array([136,87,111],np.uint8)

    red\_upper = np.array([180,255,255],np.uint8)

    # blue color

    blue\_lower = np.array([99,115,150],np.uint8)

    blue\_upper = np.array([110,255,255],np.uint8)

    # yellow color

    yellow\_lower = np.array([22,60,200],np.uint8)

    yellow\_upper = np.array([60,255,255],np.uint8)

    # all color together

    red = cv2.inRange(hsv, red\_lower, red\_upper)

    blue = cv2.inRange(hsv, blue\_lower, blue\_upper)

    yellow = cv2.inRange(hsv, yellow\_lower, yellow\_upper)

    # Morphological Transform, Dilation

    kernal = np.ones((5, 5), "uint8")

    red = cv2.dilate(red, kernal)

    res\_red = cv2.bitwise\_and(img, img, mask = red)

    blue = cv2.dilate(blue, kernal)

    res\_blue = cv2.bitwise\_and(img, img, mask = blue)

    yellow = cv2.dilate(yellow, kernal)

    res\_yellow = cv2.bitwise\_and(img, img, mask = yellow)

  # Tracking red

    ( contours, hierarchy)=cv2.findContours(red, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

    for pic, contour in enumerate(contours):

        area = cv2.contourArea(contour)

        if(area > 300):

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

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

            cv2.putText(img, "Red Colour", (x, y), cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 255))

  # Tracking blue

    ( contours, hierarchy)=cv2.findContours(blue, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

    for pic, contour in enumerate(contours):

        area = cv2.contourArea(contour)

        if(area > 300):

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

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

            cv2.putText(img, "Blue Colour", (x, y), cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (255, 0, 0))

    # Tracking yellow

    ( contours, hierarchy)=cv2.findContours(yellow, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

    for pic, contour in enumerate(contours):

        area = cv2.contourArea(contour)

        if(area > 300):

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

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

            cv2.putText(img, "Yellow Colour", (x, y), cv2.FONT\_HERSHEY\_SIMPLEX, 1.0, (0, 255, 0))

    imgContour = img.copy()

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

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

    threshold1 = cv2.getTrackbarPos("Threshold1", "Parameters")

    threshold2 = cv2.getTrackbarPos("Threshold2", "Parameters")

    imgCanny = cv2.Canny(imgGray,threshold1,threshold2)

    kernel = np.ones((5, 5))

    imgDil = cv2.dilate(imgCanny, kernel, iterations=1)

    getContours(imgDil,imgContour)

    imgStack = stackImages(0.8,([img,imgCanny],

                                [imgDil,imgContour]))

    cv2.imshow("Result", imgStack)

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

        break