```python
#!/usr/bin/env python
# coding:utf-8
import cv2
import numpy as np
import rospy

# set color HSV
HSV = {
"yellow": [np.array([20, 43, 46]), np.array([26, 255, 255])],
"red": [np.array([0, 43, 46]), np.array([10, 255, 255])],
"green": [np.array([50, 43, 46]), np.array([65, 255, 255])],
"blue": [np.array([100, 43, 46]), np.array([124, 255, 255])],
"purple": [np.array([125, 43, 46]), np.array([155, 255, 255])],}
def color_trace(img):
    for mycolor, item in HSV.items():
            redLower = np.array(item[0])
            redUpper = np.array(item[1])
            # transfrom the img to model of gray
            hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV) #转换至 HSV 空间
            # wipe off all color expect color in range
            mask = cv2.inRange(hsv, item[0], item[1])   #过滤，hsv 颜色空间的上下限
            # a etching operation on a picture to remove edge roughness
            erosion = cv2.erode(mask, np.ones((1, 1), np.uint8), iterations=2) #腐蚀处理
            # the image for expansion operation, its role is to deepen the color depth in
the picture
            dilation = cv2.dilate(erosion, np.ones((1, 1), np.uint8), iterations=2) #膨胀处
理
            # adds pixels to the image
            #target = cv2.bitwise_and(img, img, mask=dilation)
            # the filtered image is transformed into a binary image and placed in binary
            #ret, binary = cv2.threshold(dilation, 127, 255, cv2.THRESH_BINARY) #二值化
            # get the contour coordinates of the image, where contours is the coordinate
value, here only the contour is detected
            contours, hierarchy = cv2.findContours(dilation, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE) #轮廓检测
            print("contours:")
            print(contours)
            if len(contours) > 0:
                # do something about misidentification
                boxes = [
                    box
                    for box in [cv2.boundingRect(c) for c in contours]
                    if 110 < min(box[2], box[3]) and  max(box[2], box[3]) < 170 #过滤检测框
大小
                    # if min(img.shape[0], img.shape[1]) / 10
                    # < min(box[2], box[3])
                    # < min(img.shape[0], img.shape[1]) / 1
                ]
                #print(boxes)
```

```
                if boxes:
                    for box in boxes:
                        # print(box)
                        x, y, w, h = box
                        # if abs(w-h)>15:
                        #     return None
                    # find the largest object that fits the requirements
                    c = max(contours, key=cv2.contourArea)
                    # cv2.contourArea(cnt， True)   计算轮廓的面积
                    # get the lower left and upper right points of the positioning object
                    x, y, w, h = cv2.boundingRect(c) #获得外接矩形
                    #     x，y，w，h 分别表示外接矩形的 x 轴和 y 轴的坐标，以及矩形的宽和高，
                    #     cnt 表示输入的轮廓值
                    #print(x, y, w, h)
                    # locate the target by drawing rectangle
                    cv2.rectangle(img, (x, y), (x+w, y+h), (153, 153, 0), 2) #根据坐标在图像上画出矩形
                    #     img 表示传入的图片，
                    #     (x, y)表示左上角的位置，
                    #     (x+w， y+h）表示加上右下角的位置，
                    #     (0, 255, 0)表示颜色，
                    #     2 表示线条的粗细
    return img

if __name__ == "__main__":
    cap = cv2.VideoCapture(3)  # #创建一个 VideoCapture 对象，笔记本摄像头设为 0
    while True:
       # 逐帧捕获
        #第一个参数返回一个布尔值（True / False），代表有没有读取到图片；第二个参数表示截取到一帧的图片
        ret, frame = cap.read()
        img_result = color_trace(frame)
        cv2.imshow("img_result", img_result)
        if cv2.waitKey(50) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
```