链滴

进行虚拟拖放

作者：xuexiaolei1997

原文链接：https://ld246.com/article/1631023083349

来源网站：链滴

效果如下：

df5a8c1c3f994fadca71f25b5a41b711.mp4

bug提示：

1、此代码需在linux环境运行；

2、无法拖出边界，否则直接退出（这里可以通过修改边界条件来控制，本文没做）

3、图片可以自行添加，但不可超出预先指定的大小，否则报错。

```
import cv2 as cv
import numpy as np
from cvzone.HandTrackingModule import HandDetector  # 手识别
import cvzone

cap = cv.VideoCapture(0)  # 获取本机摄像头
# 设置宽高
cap.set(3, 1280)
cap.set(4, 720)
detector = HandDetector(detectionCon=0.8)  # 需要精准检测，这里类似于IoU

img_filenames = ['1.png', '2.png', '3.png']  # 可以追加图片，建议追加小图
img_list = []
for img_filename in img_filenames:
    t_img = cv.imread(img_filename)
    t_img = cv.flip(t_img, 1)  # 图像翻转，没有必要

    # 这里是为了解决宽高为奇数时，下方宽高需要除以2的问题存在的问题。
    t_img = cv.resize(t_img, (t_img.shape[1]//2*2, t_img.shape[0]//2*2))
    img_list.append(t_img)

class DragRect(object):
```

```
    # 这个类两组参数，第一组表示中心位置，第二组表示宽高
    def __init__(self, posCenter, size=[200, 200]):
        self.posCenter = posCenter
        self.size = size

    def update(self, cursor):
        cx, cy = self.posCenter
        w, h = self.size

        # if the index figure tip is in the rectangle region
        if cx - w // 2 < cursor[0] < cx + w // 2 and cy - h // 2 < cursor[1] < cy + h // 2:
            self.posCenter = cursor

while True:
    success, img = cap.read()
    # 这里需要翻转，解决镜像翻转问题
    img = cv.flip(img, 1)

    # 找到手，并绘制20个点，对应于人手的20个关节，从大拇指（0）开始，直到小拇指指尖结束
    img = detector.findHands(img)
    # 找出手位置，给出20个点
    lmList, _ = detector.findPosition(img)

    if lmList:
        # 8代表食指指尖，12代表中指指尖，此处是为了找到食指与中指指尖的距离
        # 因为单个指头操作，很有可能会出现误触，因此，相当于通过手势来执行操作
        l, _, _ = detector.findDistance(8, 12, img)
        if l < 80:  # 这个距离可以子当以修改
            cursor = lmList[8]  # index figure tip landmark
            # call update here
            for rect in rect_list:
                rect.update(cursor)  # 更新中心点位置

    for rect, sub_img in zip(rect_list, img_list):
        # imgNew = np.zeros_like(img, np.uint8)
        cx, cy = rect.posCenter
        w, h = rect.size
        # 将对应位置替换为子图片
        img[cy - w // 2: cy + w // 2, cx - h // 2: cx + h // 2, :] = sub_img
        # 绘制四个绿色小角
        cvzone.cornerRect(img, (cx - h // 2, cy - w // 2, h, w), 20, rt=0)

    cv.imshow("Image", img)
    cv.waitKey(1)
```