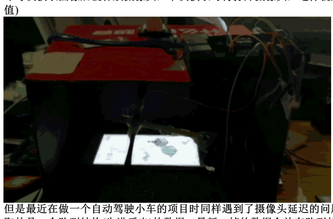
## 树莓派摄像头慢？使用OpenCV做图像处理摄像头图像延迟问题解决方法

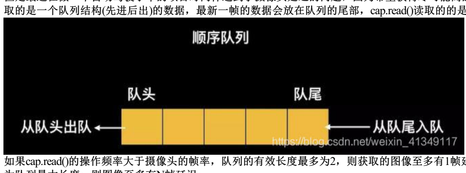
很早之前(2018年左右)在做一个当时风靡一时"跳一跳"小游戏的自动运行小工具时（树莓派通过摄像头获取手机屏幕图像，再驱动舵机云台去点击手机屏幕），就有发现树莓派在做图像处理时，OpenCV连续从摄像头获取的图像和实际的画面可能存在比较大的延迟(秒级别以上)，但是当时因为对图像帧数基本没有要求，所以使用了一个投机取巧的方法来解决。

```
cap = cv2.VideoCapture(0)
ret, frame = cap.read()
cap.release()
```

即每次获取图像后就释放摄像头，下次获取时再打开摄像头，这样就规避啊了连续获取图像的需求，下面是最终运行的效果(项目地址:WeChat-JumpGame-Cheat 代码写的很烂，没啥参考价值)



但是最近在做一个自动驾驶小车的项目时同样遇到了摄像头延迟的问题，因为希望获得尽可能高的图像帧率，所以就没有用之前的奇技淫巧来逃避，大致猜想一下，cap.read()操作实际读取的是一个队列结构(先进后出)的数据，最新一帧的数据会放在队列的尾部，cap.read()读取的的是队列的头部；



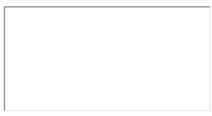

如果cap.read()的操作频率大于摄像头的帧率，队列的有效长度最多为2，则获取的图像至多有1帧延迟，如果操作频率小于摄像头帧率，队列的有效长度为N，并且随着时间推移，N最终会为队列最大长度；则图像至多有N帧延迟；
有了以上猜想，解决思路有两个，
**第一，降低摄像头帧率以满足图像处理的周期**

```
cap.set(cv2.CAP_PROP_FPS, 10)
```

但是这个操作在我的设备上是无效的，摄像头的实际帧率依然为默认的30FPS，可能是摄像头硬件不支持设置帧率，换第二种方法
**第二，保证cap.read()操作频率大于等于摄像头帧率**
在程序中新开辟一个进程，以不低于30hz的频率循环去执行cap.read()操作，这种方法我认为是比较好的，可以适应图像处理部分动态的执行周期和调整，并且可以充分的利用摄像头的帧率

```
class Camera:

    def __init__(self, camera):
        self.frame = []
        self.ret = False
        self.cap = object
        self.camera = camera
        self.openflag = False

    def open(self):
        # if self.cap == object:
        self.cap = cv2.VideoCapture(self.camera)
        self.ret = self.cap.set(3, 320)
        self.ret = self.cap.set(4, 240)
        self.ret = False
        self.openflag = True
        threading.Thread(target=self.queryframe, args=()).start()

    def queryframe(self):
        # self.openflag = True
        # while True:
        while self.openflag:
            self.ret, self.frame = self.cap.read()
            # pass

    def getframe(self):
        return self.ret, self.frame

    def close(self):
        self.openflag = False
        self.cap.release()


def cameratest():
    camera = Camera(0)
    camera.open()
    while True:
        ret,image = camera.getframe()
        if ret:
            print(type(image))
            cv2.imshow('img',image)
            cv2.waitKey(10)
        else:
            print("read faild")

    camera.close()
    cv2.destroyAllWindows()
```

最终运行效果



以上问题原因为个人猜想，如有错误欢迎留言指正，如果你有更好的方法也欢迎分享出来。