自动驾驶项目第三次报告——第十九组

孙朗宸 2018202159
董佳铭 2018202172

一、项目目标

实现 Tello 无人机通过电脑或自身的摄像头功能，检测视野内人的举止，对不同的举止做出反馈。

二、实现思路

首先通过电脑与 Tello 连接进行视频和信号的交互。Tello 视频流传回电脑后解析，目前的想法是每隔一段时间进行一次快照，通过快照总结手势，和上一各有效手势对比，分析操控者动作的变化，然后根据变化由电脑端程序发出指令，Tello 做出反馈。程序由 python 语言写成。

三、最终结果

通过对视频流的处理，可以对手势进行识别。目前录入手势 0~9 共 200 张图片，由于训练集不足，手势识别率不稳定，在电脑端几乎可以做到稳定的 100%识别，但在实际应用中误差较大。

目前实现的功能有：

通过手势进行预定义动作展示，和通过手势进行自定义动作。软件运行后需电脑端和手势配合控制。

首先通过电脑端展示的视频调节图像窗口和人的位置，待识别区域稳定后摁 s 键开始手势识别。程序会先识别手势决定进入自定义模式还是预定义模式。识别到手势 5 会进入预定义模式，9 会进入自定义模式，选择这两个手势是因为在背景不稳定的情况下，这两个手势识别度最高。

在预定义模式下，手势 1-6 对应不同预定义动作，详细描述可见程序备注。在自定义模式下，手势 1-6 对应不同动作组件，需注意自定义动作对于相同动作要求不可连续出现。

四、具体实现

1. 与 tello 进行视频流连接并操控 tello

   使用了 easytello 库对 tello 进行操控。

   与 tello 进行视频流连接也基于这个库，首先向 tello 发布命令打开视频流，然后再通过主程序的 opencv 库打开对应的视频流。在与 tello 进行视频流连接和控制时遇到了一个问题，就是 tello 打开视频和执行命令的顺序是一定的，即先执行命令后打开视频总是失败，只能先打开视频，再执行命令。同时在执行控制命令时视频流会严重卡顿，猜测之前的失败也是因为 tello 未处于一个稳定的状态。

```
mytello.streamon()
#mytello.land()
#mytello.takeoff()
orders = []          #记录指令（含历史）
captures = []        #记录间隔内投票选手
second_count = 0     #记录响应间隔
power_up = 0
power_down = 2
ans_flag = 0     #记录是否应该响应
ans_pos = 0      #记录该响应哪个指令      #暂时不需要
frame = cv.VideoCapture('udp://'+'192.168.10.1'+':11111')
```

2. 手势训练集的获取与训练

手势训练集来自于电脑前置摄像头的快照。对手势训练集的处理，采用将视野中的边界取出来，然后仅保留包围面积最大的边界的方法，这样可以尽量避免背景对手势边界的干扰；对该边界进行快速傅里叶变换以及标准化处理，得到一个 31 维向量。用该向量进行 svm 模型的训练，训练采用五折交叉检验；需要设置最大迭代次数才可停止，这间接说明了数据差异较小。最终生成一个模型并保存起来。值得注意的是，最终版较上一次未对数据进行扩展，即未通过旋转数据来增加样本，在实际中（背景干净的情况下）正确率大幅提升，接近 100%。

3. 两种模式的实现

首先需要实现进入模式。为了解决这个问题，程序会在电脑展示摄像头识别区域的内容，当使用者调整完成以后，按 s 键后开始进行识别，此时识别到手势 5 则进入预定义模式，识别到手势 9 进入自定义模式，选取这两个手势是因为在室外环境的实验中，这两个手势的识别度最高。同时使用者可以通过按键 jkil 来对识别区域进行调节。

```
key = cv.waitKey(5) & 0xFF
if key == ord('i'):
    y0 += 5
elif key == ord('k'):
    y0 -= 5
elif key == ord('l'):
    x0 += 5
elif key == ord('j'):
    x0 -= 5
elif key == ord('q'):
    break
elif key == ord('s'):
    start_flag = 1
elif key == ord('r'):
    start_flag = 0
elif key == ord('f'):
    mytello.takeoff()
    mytello.up(70)
elif key == ord('w'):
    print(mytello.get_battery())
```

在进入任何模式后，为了避免识别到不需要的手势，程序在预定义模式默认第一个

手势不会是 5，在自定义模式要求起始和终止手势为 1。由于预定义模式一次性识别一个手势，而自定义模式需维护一个手势队列，故自定义模式还默认连续两个命令不会是相同的手势。

```
#如果没有定义mode但已经启动程序，就先判断mode，手势5表示预定义，9表示自定义
if ans_flag == 1 and mode == 0 and start_flag == 1:
    if orders[-1] == 4:
        print('进入预定义模式')
        mode = 1
    elif orders[-1] == 8:
        print('进入自定义模式')
        mode = 2
elif ans_flag == 1 and mode == 1 and start_flag == 1:
    print('预定义判断')
    if orders[-1] != orders[-2]:
        start_time = time.time()
        run_flag = 1
        solve_order(orders[-1])
```

```
#自定义要求以1开始，以1结束
elif ans_flag == 1 and mode == 2 and start_flag == 1:
    print('自定义指令判断')
    if len(design) == 0 and orders[-1] == 0:
        print('自定义指令开始')
        design.append(orders[-1])
    else:
        if len(design) != 0 and orders[-1] != design[-1]:
            print('自定义指令增加')
            design.append(orders[-1])
if mode == 2 and len(design) != 0 and len(design) != 1 and design[-1] == 0:
    print('执行自定义指令')
    start_time = time.time()
    run_flag = 1
    solve_design(design)
    design.clear()
```

在任何模式完成手势识别后，进入到各自的解析函数向 tello 发送命令。在自定义部分额外设计了自动执行逆指令，为了使 tello 最终会飞回原地。

```
#重调任务队列，使回到原点
#玩家二次开发时请谨记修改此循环和下面的循环
for i in range(len(design)):
    if design[i] == 1:
        design[i] = 2
    elif design[i] == 2:
        design[i] = 1
    elif design[i] == 3:
        design[i] = 4
    elif design[i] == 4:
        design[i] = 3
    elif design[i] == 5:
        design[i] = 6
    elif design[i] == 6:
        design[i] = 5
```

4. 具体手势的判断

由于手势的判断是对单张图片的，可能与使用者的实际期望不符，并且为了尽量解决户外环境对手势识别的干扰，采用了投票的办法。即维护一个手势的集合，每 5 毫秒识别一次，每秒清空一次，清空时根据集合内的手势，取最多的为这一秒内的实际手势。

```
#这个函数用于得到数组中的众数
def find_max(list):
    count = [0,0,0,0,0,0,0,0,0,0]
    for i in range(200):
        count[int(list[i])]+=1
    j = count[0]
    ret = 0
    for i in range(10):
        if count[i] > j:
            j = count[i]
            ret = i
    return ret
```

五、改进空间

1. 手势的训练集较小，但同时还包含了同一个手势的正反图片的处理信息，故虽然在电脑前置摄像头和干净背景的条件下识别正确率极高，在户外飞行时识别正确率仍偏低。可以添加更多的训练数据，并增加带一定噪声的户外数据进行训练。

2. 未避免 tello 在飞行过程中通过摄像头识别到别的手势干扰正常程序进行，在编写程序时设定了一个最短执行间隔，但实际操作时发现，tello 内部处理指令应该是一个队列，每完成一条指令，才开始下一条，期间视频甚至会中断，最短执行间隔没有存在的意义，可以删除。

六、文件说明

SVM.py：对得到的图片数据进行 SVM 训练并保存模型的文件。

Tello_final.py：最终的控制 tello 进行手势识别的文件。