无人机人脸跟踪期末实验报告

环境学院 金泊翰 2018200684

一、 项目简介

本项目的核心任务是实现无人机跟随人脸的移动。我们通过无人机与电脑间的通信传输视频和命令、通过多线程协同进行实时控制、通过 XGBoost 和 MTCNN 实现计算机视觉，最终实现了上述目标。虽然由于与硬件交互效率方面的限制，整体性能还有待提升，相比于另一组还有些差距，但是对于我一个非信息专业的同学来说，能够设计并编写这样一个工程框架、学习多个重要的 Python 库，这本身已经让我感到很有收获。

二、 我的分工

本项目中，除了人脸位置检测模型之外的部分均由我设计和编写完成。同时，我也负责整体的测试和调参工作，并尝试了一个 CNN 网络的搭建。

三、 项目设计

1、 软硬件情况

本项目的无人机使用大疆-睿炽科技的 Tello 编程无人机，电脑端使用 Windows10 系统，主要通过 Python 进行信息交互、图像识别、指令生成等工作。由于官方提供的视频解码器只能部署在 Python2 上，而 MTCNN 模型需要部署在 Python3 的 pytorch 上，所以我们的程序需要同时利用 PyCharm 和 Jupiter 两个软件来运行。

2、 总体框架

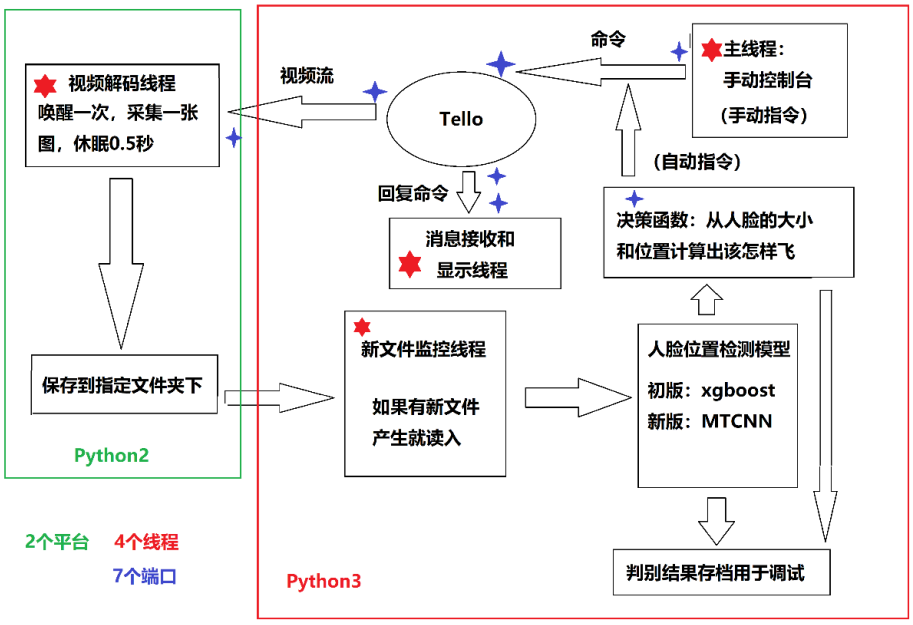

图 1 项目程序框架图

本项目所涉及的主体很多、对实时控制的需求很强，因此需要使用多线程的方式来运行。总共涉及 2 个平台、4 个线程、7 个端口之间的相互配合，整体的复杂度较高。

在 Python3 上，我编写了一个可以通过键盘向无人机发出指令的主线程，它控制着无人机的起飞、降落和自动跟踪的开启和关闭。

在 Python2 上，我编写了一个实时接收 Tello 发送的视频流的线程，为了照顾 Python3 这边的运算速度，视频解码器每 0.5 秒工作一次，提取一张图像并将其保存在一个固定的文件夹下。

Python3 上的新文件监控线程会不断地检测文件夹下是否产生了新文件，每次有新文件产生后，它就会将该文件读入人脸检测模型中。

人脸检测模型会返回人脸的大小和在图像中的坐标，决策函数会根据这个些参数计算出人脸与无人机视野中心点的偏差，进而返回一个“go x y z”的命令，使得无人机的位置向人脸方向移动，并最终收敛到人脸正前方（不过由于无人机镜头略向下倾斜，所以实际上是悬停在人脸斜上方）。

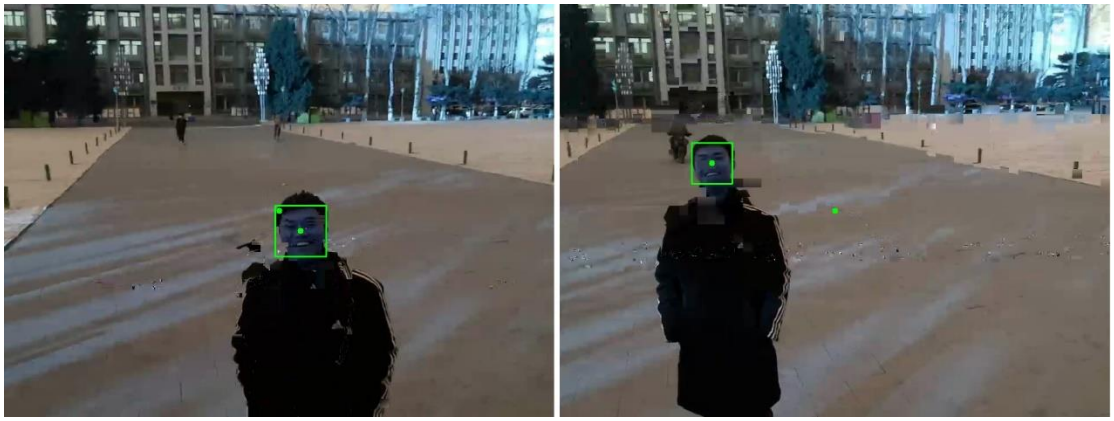

图 2 XGBoost 模型室外实验（教二草坪）

除此之外，我还设置了消息接受和显示线程、图像和决策保存等功能，以便对项目进行实时的监督和后续调试。

四、 实现过程

1、 信息交互的实现

Tello 与电脑之间的信息交互采用套接字格式。数据以字符串形式编写后，被转化为二进制，通过 UDP 协议发送到制定的 IP 和端口。这一过程在 Python 上可以借助 socket 库实现。通过 socket.sendto 函数，可以将字符串发送到制定的端口。通过 socket.recv_from 函数可以接受套接字，并获取发出该信息的服务器，这样就可以实现对于来自指定端口的信息进行接收。

2、 视频流的解码

Tello 所拍摄的每张照片是 720×960×3 的大小，信息量较大，因此在传输的过程中需要分为小包，每次发送 1460 个字节，但最后一个包不到 1460 字节。每次解码器发现接收到的信息不到 1460 字节时，就说明之前的信息拼接起来正好是一张图片，就可以对之前积累的包进行 H264 解码，得到一张图片。H264 视频的解码通过睿炽科技的技术人员提供的 Python2-C^{++}代码实现。

即便如此，上述方法解码得到的视频流仍然存在不稳定的现象，当无人机或无人机视野中的对象出现移动的时候就容易产生颜色混乱、马赛克的现象。这对于我们后续的人脸检测工作造成了不小的挑战。

3、 人脸检测模型

这部分我们尝试使用过两个模型：

我们初期使用的是 XGBoost 算法训练出来的级联分类器。它使用一些卷积核来提取图像的特征，之后根据这些特征的值来以类似决策树的形式对目标是否属于人脸进行判断，通过每一级判断的区域会被认定为人脸。但是我们后来发现，这个算法经常会将一些明显不存在人脸的图片（往往是马赛克的图片，如右图所示）上硬找出一个人脸来，因此我们探索了一些改进的方式。

图 3 被错误识别的马赛克图片

首先我通过查阅资料了解到拉普拉斯变换可以反映图片的清晰度。然而尝试之后发现，拉普拉斯变换可以衡量一张图片上边缘的清晰程度，但是充满马赛克的图片也有很多边缘“清晰”的方格子，所以会被误判为清晰的图片。

其次，我又尝试使用 pytorch 搭建了一个 CNN 图像分类器接在级联分类器后面，试图通过在自己拍摄的数据集上进行的训练，来将级联分类器的输出结果分为“right”和“wrong”两类。“right”表示该图片的判别结果正确，可以用于决策，“wrong”表示不能用于决策。

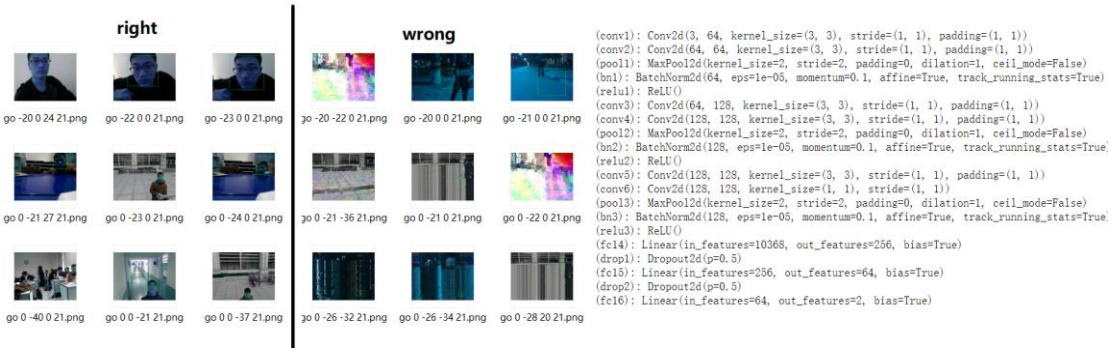

图 4 CNN 使用的训练集和网络结构

但是经过多次训练，该模型始终将所有的图片判别为“wrong”。猜想可能是因为不论图像是否清晰、图中是否有人脸，都有一定量的判别错误存在。所以模型很难以“right”或“wrong”作为标准进行分类。后来由于王玺同学提供的 MTCNN（多任务卷积神经网络模型）基本解决了上述问题，于是我这部分就没有搭载在最终的项目上。

4、 决策模块的设计

在实现了人脸检测之后，我们需要根据模型检测到的人脸位置确定飞行方向和所需移动的距离。我们首先确定了如下的公式：

$$\triangle forward \propto \log\left(\frac{Ideal\_Area}{face\_Area}\right)$$

$$\triangle left \propto x_{face} - x_{center}$$

$$\triangle up \propto y_{face} - y_{center}$$

其中，x_{face}和y_{face}为人脸所在的横纵坐标值，x_{center}和y_{center}为照片中心点和坐标值，Ideal_Area 为理想中的人脸大小，face_Area 为实际人脸在照片中的大小。

之后我们先任意给三个式子赋予比例系数，之后通过批量拍摄以决策结果为名称的照片来查看这个比例系数是否合理。如此反复调试，得到最终的比例系数。

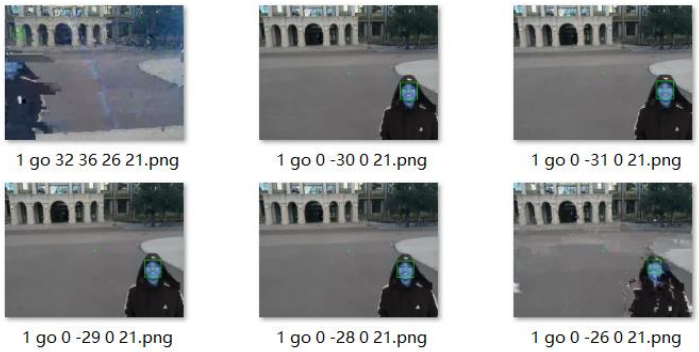

图 5 调参时所使用的一些样本（教二草坪）

五、 项目成果

我们分别在明德国际楼、教二草坪进行了人脸跟踪测试，并在课堂上进行了现场演示，测试效果随着整体代码的不断调试而改善。

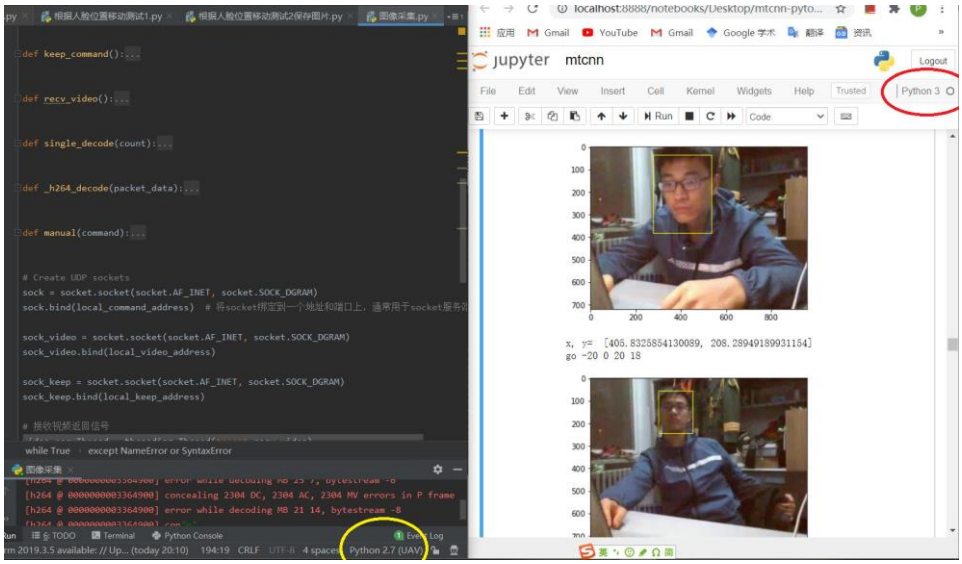

图 6 Python2 和 Python3 共同运行代码在宿舍进行静态调试的效果

最终，我们实现了多线程的流畅协调，时间延迟缩减到 2 秒左右。最终基本上不会出现对于人脸的任何误判，飞行的稳定性也大大提升。只是如果画面中的人移动，飞行器会出现短暂的马赛克和色彩混乱，需要 5~10 秒中来适应才能重新根据人的新位置找寻目标。

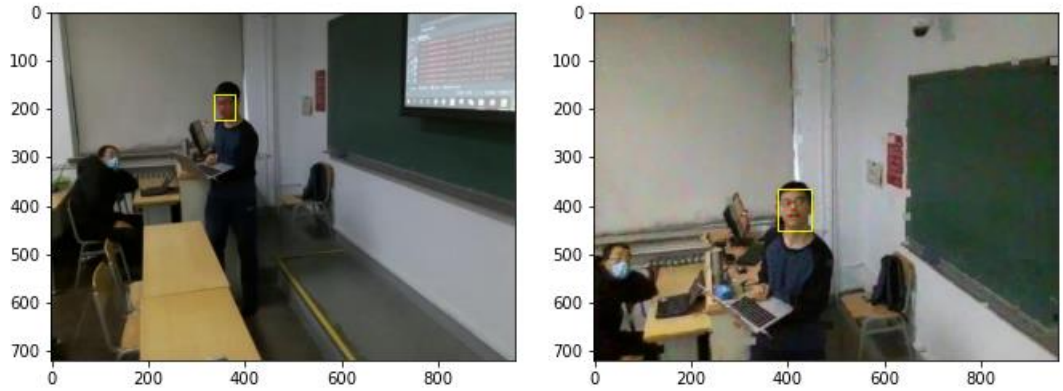

图 7 最终在教室进行演示时从起飞（左）到基本对正（右）

六、 反思与收获

1、项目存在的不足

本次实验中我最开始花了较大的精力在视频流的解码上，而最后还是依靠官方的技术支持解决。另一组似乎在这个问题上找到了现有的资源，不仅视频解码的质量比我们高，还留出了更多的时间用于算法的优化和拓展上。因此我也感受到从网络上寻找优质的 IT 资源并在稍微修改后灵活地使用是十分重要的能力。未来我也要学习一下 GitHub 等技术社区的使用方法，以便更快速地调用这些资源。

2、收获 1：体验了项目开发

本次实验中，我编写了很多函数以至于发现把所有代码放在一个文件中十分难以阅读。后来请教了一下信院的同学才发现原来看似神秘的 Python 库只是编写好的脚本而已，我也可以把自己编写的函数和类放在与主程序相同的文件夹下，用 import 的方式来调用，这样就大大增加了代码的可读性。这似乎也是大型编程项目开发的必备技能。

除此之外，本实验也是我目前编写过的结构最复杂的项目，其框架的设计基本上都是我的原创，对于自己的编程能力提升很明显。

3、收获 2：学会了很多库

本次实验中，我使用到了 threading 多线程库、socket 套接字库、OpenCV 计算机视觉库、PyTorch 神经网络库等重要的 Python 包。在调用他们的过程中，我也学习到了一些和计算机网络、计算机系统有关的知识，这都是我之前仅仅用 Python 做数据分析所没有接触到的。最后晒一下笔记，未来还有继续多多实践、努力学习！