工程实践与科技创新III-F

结题报告

项 目 名 称 ： 循线倒车树莓派智能车

姓 名 ： 徐康彦

合作人姓名 ： 白益、孙凯文

2018 年 6 月

（一）研究内容综述

本项目主要利用运行在树莓派上的 Python 代码与 OpenCV 库来实现小车巡线、识别停车标志、倒车入库这三个功能。由于小车的输入仅仅是车前后的摄像头接收的图像数据，所以项目的主要难点在于如何对图像进行处理，去除无关信息（如反光、非相关颜色等）的干扰，只对有效信息（如巡线轨迹、特定标志等）做出正确反应。本项目很契合“自动化”这一主题。

（二）研究方案

在本项目中，代码编写与基本思路的构想由我负责，故以下将详细介绍各步骤实现的方式。

PART 1. 巡线

首先是对摄像头及周围的物理改造，固定住摄像头，将镜头前的线路全部压低以免干扰到图像。

代码方面，第一步是对图像进行二值化。二值化的阈值设定受环境光线干扰（反光等）极为显著，属于**经常需要调整的参数**。

二值化后，将图像转为巡线为黑，背景为白的图像。考虑到交叉线、大块黑色区域等的干扰，决定采用留下图像垂直边缘的 **mask** 对图像进行卷积，卷积后仅保留偏垂直方向的边缘线条。

mask 为：$\begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix}$

卷积后用 5*5 方阵对图像进行膨胀，以使巡线轨迹更为明显。

图像原点在左上方，向右有 640 列，向下有 480 行。选取靠近小车部分的巡线轨迹而丢弃远处的，同时将图像左右也丢弃一小部分，故所选区域为 440 行至 480 行，30 列至 610 列。

对于 440～480 这每一行，从左至右记录黑色点的坐标，例如 307、331 等，并同时记录此行的黑点个数 i，每行进行完此操作后取所有黑点坐标的平均，并将其与 320 相减，记为 err。可以的到 41 个 err 值，再将其取平均，记为 error。此 error 做为巡线的**核心参数**。

在巡线时，考虑到测试场地的转弯半径很小，难以使用一固定的舵机转角成功进行转弯。如果设定固定值为 1 或-1，会出现偏离巡线等问题。经交流后，引入了类似 P 控制的方法，将舵机转角设定为 servo = -st，其中 st = error * 0.005。此 0.005 即位 Kp 值，也是一个需要调整优化的**参数**。注意到，当巡线轨迹为直线段时，error 一般为 10 以内的值，则 st 通常远小于 0.1，可以认为直线行进。而当 error 达到 200+时，则说明处在较陡的弯上，此时舵机转角要打满为 1 或-1 才可成功巡线转弯。

最后，还增加了一个补救措施以免小车失控。当小车脱离巡线时，理论上所得图像应为一片白色，此时单行黑点个数 i 将会少于某设定值（例如 5），对小车进行定距离倒车操作以回到巡线。但在实际测试中此补救措施效果不可靠，常发

生倒车过头等错误，故在代码中被注释掉了。

以上除了最后的补救措施，都采用了 speed 模式。

【流程示例】

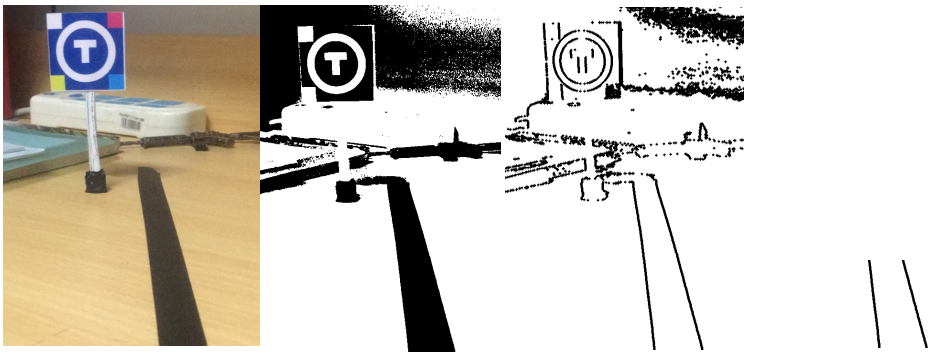

图 1 巡线图像处理示意

从左至右分别为原图像，二值化后图像，卷积膨胀后图像以及求取 error 时所用的图像。

PART2. 停车标志识别

停车标志如上所示，由于标志中含有圆，故可以采用 OpenCV 中的霍夫圆检测，只要设定所能检测出的圆的**半径范围**，就能完成识别。

使用现成的 circles = cv2.HoughCircles()函数，其中设定两**半径参数** minRadius 为 30, maxRadius 为 40，只要圆的个数为 1，则认为识别到停车标志。

为了避免检测不到圆而报错，使用如下代码：

if circles is not None: num = len(circles)

从而当 num = 1 时就认为完成识别。

【流程示例】

图 2 停车标志的检测

PART3.倒车入库

同样，第一步是物理方式，对于后置摄形头虽不采用物理固定，但每次调整到与竖直方向大约 70 度左右，从而能进行倒车操作。

如何将车库与周围的环境区别开来是一大难点，这里有两种方式，一种是对 RGB 图像操作，一种是对 HSV 图像操作，使用 cv2.inRange()函数进行图像的分割。需要注意的是，阈值的写入是倒序的，比如 RGB 图的阈值设定需按照 BGR 的顺序。

无论是 RGB 图像还是 HSV 图像，实测下来阈值收到光线的影响都很大。

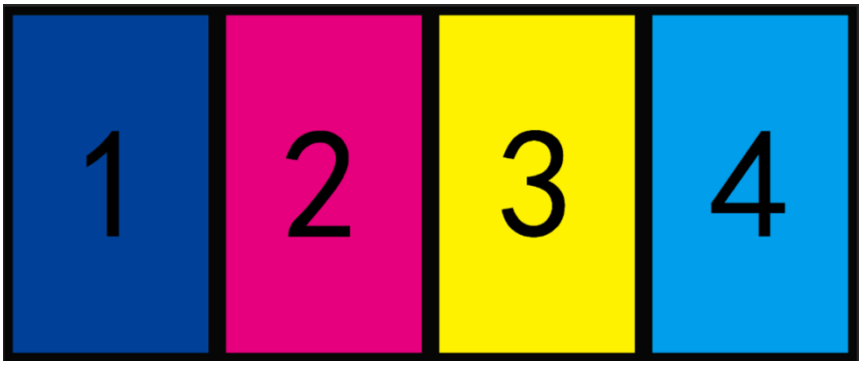

图 3 彩色车库

【流程示例】

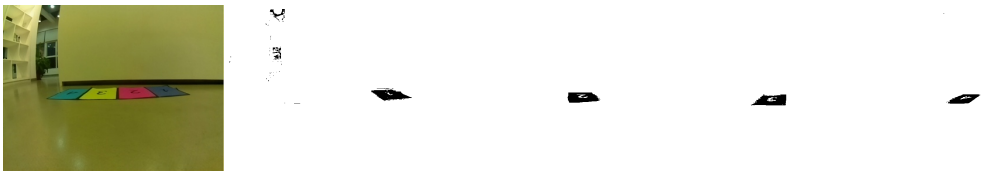

图 4 采用 RGB 进行的分割(环境:光彪楼二楼夜晚)

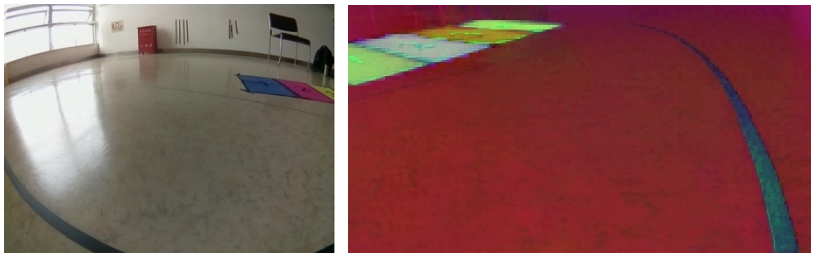

图 5 镜头中的车库及车库的 HSV 图像

测得可以用来区分的 HSV 阈值：

```
low_1 = np.array([75, 90, 60])
high_1 = np.array([130, 180, 130])
low_2 = np.array([145, 90, 99])
high_2 = np.array([200, 185, 155])
low_3 = np.array([25, 145, 105])
high_3 = np.array([40, 260, 185])
low_4 = np.array([55, 65, 25])
high_4 = np.array([95, 255, 135])
```

倒车入库的实现方式如下：

在接受所需要停入的车库序号后，选用此序号所对应的阈值参数，对摄像头得到的图像循环进行 mask = cv2.inRange((hsv, low_ , high_))操作，得到 mask 图像。

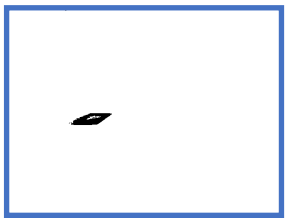

图 6 示例 mask 图像(蓝色边框为区分白底所加)

注意到，虽然图像中的车位发生变形，但其中心横坐标大致可通过将图中所有黑点横坐标求和后取平均得到，坐标设为 x，黑点总数设为 m，同时另设 dis = 320 - x。本项目倒车采用类似 bang-bang-control 的方法，当 dis 处在(-50,50)这一区间内，让小车采用 distance 模式直行 dist = 0xA0；当 dis 处于(-∞,-50]范围内，向左后方倒车 dist = 0xA0，motor = -0.6, servo = 1；当 dis 处于[50,+∞)范围内，向右后方倒车 dist = 0xA0，motor = -0.6, servo = -1。

当检测到 m > 6000，即黑点个数足够多，认为小车已经十分接近车库，此时直接以 motor = -0.2, dist = 0x4F0 直行一段进入车库后，切换为 stop 模式结束。

PART4.附加

由于一开始对车库位置的理解不同，以为车库位于巡线轨迹一侧，所以加入了一段抵达倒车预备位置的功能。

【流程示例】

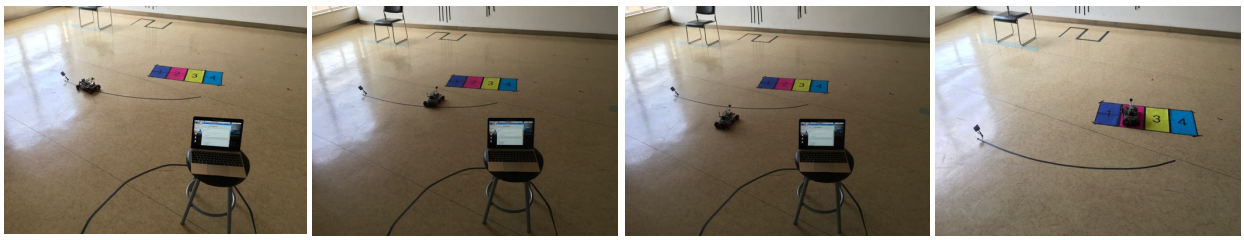

图 7 后退-大转-前行-倒车

这里首先如之前所述方法得到一号车位的横坐标，当其不小于某特定值时，持续倒车，直至其小于某值，此时认为已接近车库，进行定距离转弯到车尾接近正对车库的状态，再前行一段固定距离，达到倒车的预备位置，实施倒车。

（三）研究成果

本项目经过大量的测试，基本达到预期目标，但是少部分参数仍需依据外界环境进行调整。

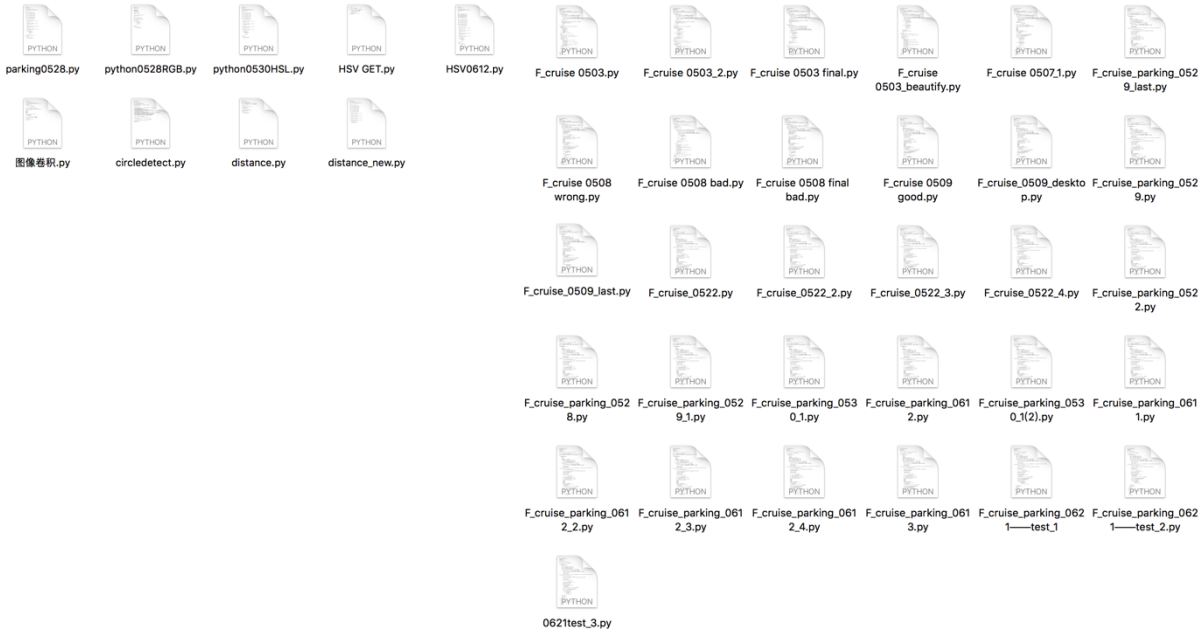

图 8 图像处理代码与小车代码存档

（四）研究总结

1. **研究中存在问题、建议及需要说明的问题？**

本项目主要问题还在于会受到外界光线、物体等干扰，比如红色灭火箱与红色车库就易于混淆。此外，电池耗电速度较快也是困扰研究长时间进行的一大问题。

2. **在本次课程实践中的感想与体会？**

本此课程实践提供了熟悉 Python 和 OpenCV 的机会，为以后工作研究打下了基础，尤其是对 OpenCV 的学习能解决许多有趣的问题。此外，由于研究内容非常开放，可以锻炼解决问题、项目交流等方面的能力。总体来说，从这次实践中能学到很多内容。非常感谢老师和助教设计制作了小车，以及在项目中给予的指导帮助。

3. **对本次课程实践的意见与建议？**

希望测试场地能提前预留较长时间以供调试，同时希望场地避免反光问题。

2018 年 6 月

上海交通大学