# **华中科技大学**

人工智能与自动化学院

**智能车课程设计报告**

班级：（给孩子手动打个码）

姓名1：（给孩子手动打个码）

学号1：（给孩子手动打个码）

姓名2：（给孩子手动打个码）

学号2：（给孩子手动打个码）

姓名3：（给孩子手动打个码）

学号3：（给孩子手动打个码）

报告提交时间:2023年11月30日

1. 课程目标
2. 学会使用虚拟机。
3. 熟悉在Linux环境上开发和运行代码的方式。
4. 熟悉智能车控制系统。
5. 设计所用设备

OriginBot导航版

|  |  |
| --- | --- |
| 效果图 | originbot_nav |
| 应用处理器 | **RDK X3** CPU：四核ARM Cortex-A53@1.5GHz BPU：双核@1GHz，等效算力5TOPS 内存：4GB LPDDR4 RAM 存储：32GB（TF卡） |
| 运动控制器 | MCU：STM32F103 Flash容量：64K RAM容量：20KB |
| 差速底盘 | 带编码器TT电机 \* 2 摄像头支架 万向轮及支架 |
| 摄像头 | 400W像素MIPI相机 |
| 激光雷达 | 5~10Hz扫描频率， 360°扫描角度 2cm绝对误差 |
| 姿态传感器 | 加速度、陀螺仪、角度输出 自带卡尔曼滤波 |
| 电池续航 | 2~4小时 2000 mAh（典型值） 12.6 V 800mA充电器 |
| 尺寸 | 长：180 mm 宽：132 mm 高：167 mm（±2） |
| 重量 | 0.78 kg（±0.05） |
| 最大速度 | 0.8 m/s（±0.05） |

1. 课程任务
2. 利用OpenCV的视觉巡线技术，使小车能够跟随地面上的黑线，并使用SLAM技术创建地图。
3. 在前述地图的基础上，实现目标导航功能，使小车能够自动驶向预先选择的目的地。
4. 已实现的功能
5. 实现了OpenCV的视觉巡线功能。

通过调整代码参数和添加功能，使小车能够应对实验场地上的特殊情况。详细操作将在设计思路中详细说明。

1. 实现了SLAM地图构建功能。

在巡线过程中利用雷达构建高质量的图像，并将其保存在小车内部相应文件夹中，以便后续的目标导航功能使用。

1. 实现了目标导航功能。

调整了部分小车参数，使得在目标导航过程中小车能以相对稳定的速度前进。这使得小车所探测到的图像与预期的图像更加匹配，从而更快地达到设定的目的地。

1. 设计思路及流程图

**OpenCV视觉巡线功能**

下面以小车自带的巡线代码follower.py为baseline，介绍一下我们的设计思路。

1、baseline

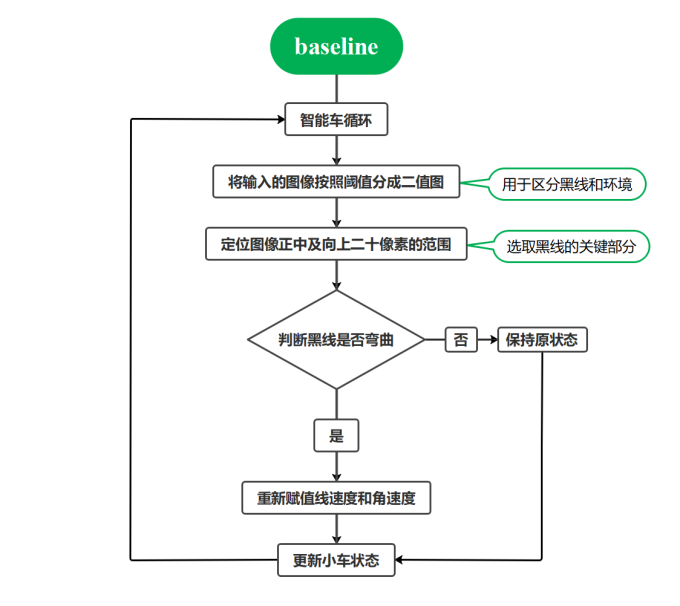


图5.1.baseline流程图

图5.1，baseline算法采用了三个模块来控制小车。第一模块是阈值分割，它对输入图像的RGB三通道设定阈值限制，将黑线以外的背景设置为同一颜色，以此提取黑色信息。第二模块是范围选择，由于只有处于合适距离的黑线才会引导小车，因此需要选取距离小车合适的一部分。该模块设定阈值[h/2,h/2+20]，以选取图像正中及向上20像素的范围。第三模块是决策模块，通过处理后的图像确定黑线中心点的位置，进而修改小车的速度。

在实际应用中，我们发现baseline算法存在以下三个问题：

（1）调整阈值分割模块参数后，小车仍不能准确识别黑线与地砖缝隙。这是因为小车在移动中，光强度不断变化，无法仅依靠简单的阈值分割来区分目标。

（2）小车的计算速度受限，如果小车在转弯处速度过快，可能导致车身越过黑线，从而丢失黑线视野，进而无法控制小车运动，产生误差，最终使得巡线失败。

（3）小车在直线行驶时速度过慢，会浪费总时长。

2、SmartFollower

针对baseline中的问题，我们对算法进行了改进，提出了SmartFollower。

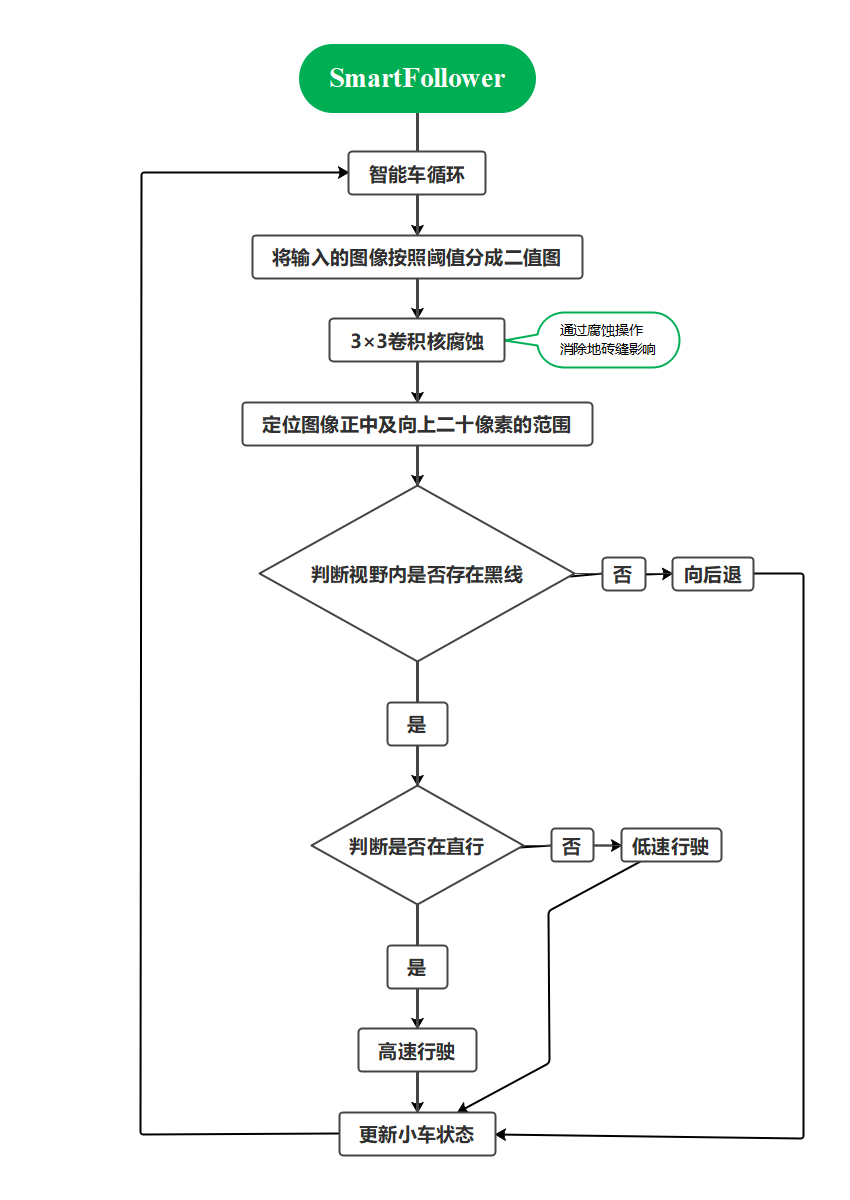


图5.2.SmartFollower流程图

图5.2，首先，增加了腐蚀模块，利用形态学腐蚀处理对图像进行预处理，消除地砖缝的痕迹，从而弥补了阈值分割的不足。其次，重新设计了决策模块，当小车视野中没有检测到黑线时，会退后直到重新获得黑线视野。同时，在识别到黑线后，如果黑线中心在视野中心附近，则判断小车正在直线行驶，可以适当提高移动速度。

详细材料见附录。

在测试环境下，这些改进措施使得我们的算法取得了更好的结果，小车能够更快、更稳定、更准确地行驶。

**SLAM地图构建功能**

由于小车本身的SLAM地图构建功能就很完善，所以我们并没有做出什么修改，在此不多赘述。

**目标导航功能**

我们在使用小车原有的目标导航功能时，遇到了一个问题：小车移动速度过快，导致实际位置与预计位置有一定偏差。虽然目标导航的矫正功能可以在一定程度上缓解这一问题，但也导致小车需要花费大量时间进行矫正。这个问题主要受限于小车性能和目标导航功能的代码，超出了我们的能力范围。为了尽可能减少位置偏差，我们进行了机器人里程计校准，并下调了小车的移动速度，希望能够减小矫正难度。虽然小车目标导航仍需要花费较长时间进行矫正，但相比最初已经有了一定的提升。

1. 课程设计总结及心得体会

1、项目成果总结

总的来说，我们取得了以下三个主要成果：

·视觉巡线功能改进：通过调整参数和增加模块，我们成功改进了小车的视觉巡线功能，使其能够在指定环境下实现更好的巡线效果。

·高质量地图构建：我们利用雷达技术，成功构建了高质量的地图，为后续的目标导航功能提供了可靠的基础。

·目标导航功能实现：基于我们所建立的地图，我们成功实现了目标导航功能，在小车巡线的同时能够准确导航到指定目标。

值得注意的是，若没有我们新提出的SmartFollower算法，小车巡线时间可能会更长，甚至可能无法完成一整圈的巡线，从而导致后续的建图和导航功能无法正常进行。通过引入SmartFollower算法，我们有效地解决了这一问题，提升了巡线的效率和准确性，为整个项目的成功实施打下了坚实的基础。

2、遇到的问题和解决方案

在课程设计的过程中，我们遇到了多个问题，涉及到小车硬件组装、环境配置和算法优化等方面。针对这些问题，我们采取了以下解决方案：

小车硬件组装：在小车的硬件组装过程中，我们遇到了一些安装和连接问题。为了解决这些问题，我们仔细阅读了小车组装指南，并观看了相关的教学视频，以确保每个零部件正确安装和连接。

环境配置：我们面临了环境配置的挑战，包括安装和配置软件、驱动和工具等。为了克服这些问题，我们在CSDN、知乎等网站查阅资料，积极寻找解决方案，并按照指导进行环境配置。

算法优化：在实现巡线、地图构建和目标导航功能的过程中，我们遇到了一些算法上的挑战，如巡线的效率和准确性、地图构建的精确度等。为解决这些问题，我们灵活运用课本中的知识，参考了相关领域的研究成果，并尝试了多个改进的算法方案，最终成功优化了算法，确保了功能的有效实现。

3、学习收获

在这次课程设计中，我们获得了以下宝贵的学习收获：

（1）虚拟机的使用方式：通过这次项目，我们学会了如何搭建和配置虚拟机环境，以便进行ROS开发和仿真实验。我们了解了虚拟机的基本原理和使用方法，提高了我们在虚拟环境下进行开发和实验的能力。

（2）ROS的使用：我们通过实际的项目实践，深入学习了ROS的使用。我们熟悉了ROS的基本概念、工具和功能包，掌握了ROS的常用命令和操作方式。这为我们日后从事机器人开发或相关领域的研究奠定了坚实的基础。

（3）理论与实践的结合：通过将课本知识与实际运用结合起来，我们深刻体会到了理论与实践的紧密联系。我们不仅了解了相关理论背景和算法原理，还通过实际项目的实施，真正将这些知识应用到实际情境中。这样的实践经验使我们的学习更加有深度和广度。

总的来说，这次课程设计为我们提供了一个全面的学习平台，让我们学到了虚拟机的使用方式、熟悉了ROS的使用，并且将课本知识与实际应用有效结合。这些收获不仅扩展了我们的技术知识和技能，同时也培养了我们的实际动手能力和解决问题的能力，为我们未来的学习和职业发展奠定了坚实的基础。

4、未来展望

虽然这一课程设计已经结束，但是我们认为该项目仍存在许多可改进的地方。

比如说，我们意识到小车的实战效果很大程度上依赖于运算速度。当前的控制策略是基于指定线速度和角速度进行控制，而不是指定一段距离。这导致小车在不同的运算速度下对指令执行效果存在变化。为解决这个问题，需要从小车的底层逻辑出发，考虑对控制策略进行改进和优化，以提升小车的运算速度和稳定性。

而且，目标导航功能的效果并不理想，我们认为这在很大程度上是由于算法本身存在的不足。然而，深入优化算法需要更多的时间和资源，在课程设计的时间限制内我们是无法完全解决这个问题的。

如果未来从事机器人控制相关的学习或工作，我们会在这基础上更进一步。

1. 附录

补充材料整理如下：

·第一部分为SmartFollower中的改动，包括阈值分割模块、腐蚀模块和决策模块的设计思路和参数代码。

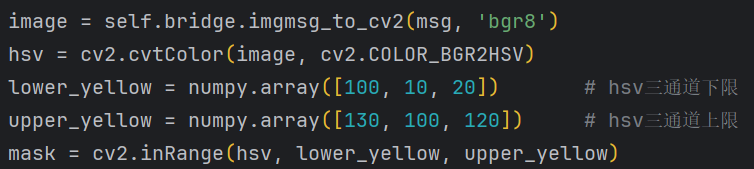
·第二部分为阈值分割模块和腐蚀模块的可视化图片结果。

1. 改动部分

**阈值分割模块**

我们对小车运行过程中的图像进行截取，并将其转换为HSV颜色空间。然后，我们使用合适的阈值（参考代码7.1.1）来排除大部分环境干扰。然而，由于小车的移动会导致光强度的不断变化，有时无法处理地砖缝、环境围墙缝、地砖花纹和黑线上的反光问题。

需要注意的是，地砖花纹与黑线之间有一定的距离，而黑线上的反光几乎不会影响黑线的定位。因此，我们仅需要处理地砖缝和环境围墙缝这两种情况。

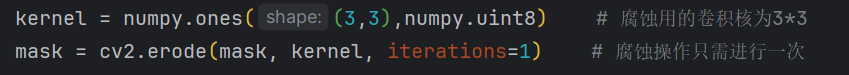


代码7.1.1.阈值分割模块上下限参数

**腐蚀模块**

阈值分割模块已经对图像进行了基本处理，但噪声如地砖缝和环境围墙缝等仍会影响结果。这些噪声的特征是线条很细，与黑线较粗有着明显区别。我们通过选用腐蚀操作来对这些噪声进行处理。（参考代码 7.1.2）

在腐蚀操作后，一般需要进行膨胀操作以避免丢失原始图像中的有效信息。然而，在实际操作中发现，即使丢失这些信息，对黑线的定位几乎没有任何影响。因此，为了提高运算效率，我们决定不加入膨胀操作。



代码7.1.2.腐蚀模块操作参数

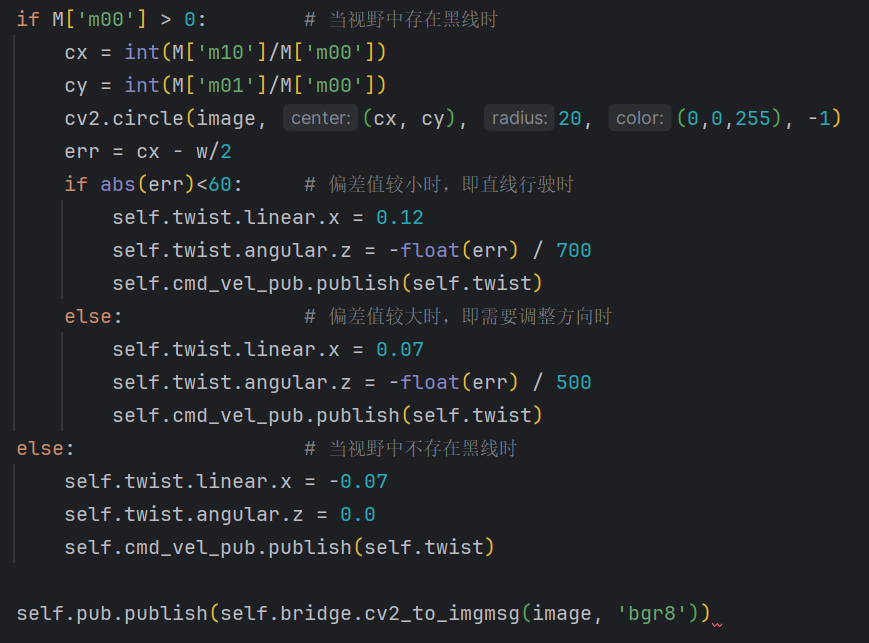
**决策模块**

根据调试经验，我们发现在线速度为0.05，角速度为-float(err) / 700时，小车可以稳定通过直角弯。其中，err表示黑线中心相对于图像中心的偏差值。

但是，这个速度不能满足我们对小车行驶时间的要求。因此，我们对决策模块进行了修改。（参考代码7.1.3）首先，我们使用条件语句 if M[‘m00’] > 0：来判断视野中是否存在黑线。如果不存在黑线，说明小车已经超过目标点，需要后退。所以，我们将线速度设为-0.07，角速度设为0。

当视野中存在黑线时，我们将偏差值的绝对值与60进行比较。如果偏差值的绝对值较小，说明黑线中心接近视野中心，可以判断为直线行驶，这时我们可以提高移动速度。因此，我们将线速度提高至0.12；而偏差值的绝对值较大时，说明需要调整行驶方向，因此我们将角速度提高至-float(err) / 500。

在测试环境下，经过这个改动的决策模块可以在保证完成巡线的基础上尽可能提高行驶速度，实现了我们的目标。



代码7.1.3.决策模块代码

1. 可视化图片

在图像处理后，本部分呈现了不同模块生成的可视化结果。

图7.2.1展示了小车得到的原始图像和阈值分割模块处理后的图像。该图像已处理了大部分的环境影响，但仍受到环境围墙缝、地砖花纹和黑线上反光的影响。在图像上方左侧的白色细线是错分环境围墙缝的结果，而其右侧的白色三角形是错分地砖花纹的结果。另外，黑线中央和起点处右侧未能成功识别的部分是受反光影响的结果。

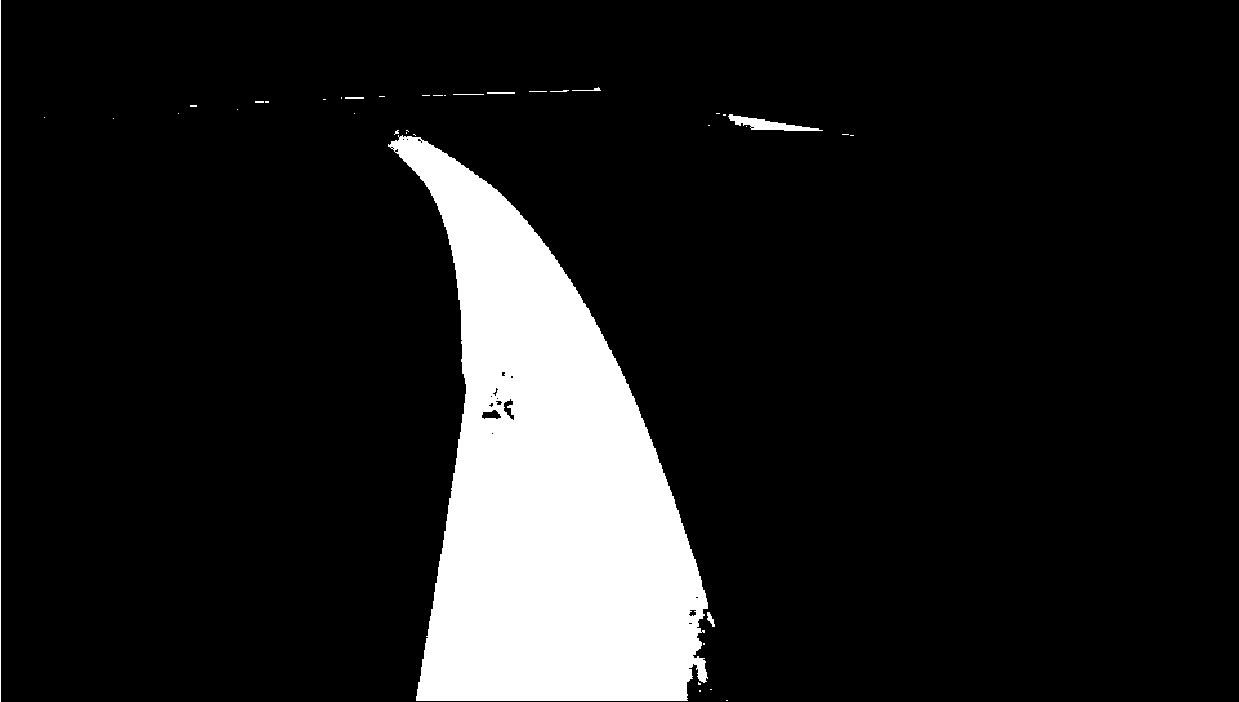


图7.2.1.小车得到的原始图像（左）和阈值分割模块处理后图像（右）

图7.2.2展示了腐蚀操作和腐蚀后的膨胀操作所得到的结果图像。可以观察到，腐蚀操作消除了环境围墙缝的干扰，并减小了地砖花纹的影响。尽管腐蚀操作也增加了反光干扰，但进行一次膨胀操作可以解决这个问题。（甚至可以在此基础上继续膨胀和腐蚀操作，以消除更多反光干扰）然而，由于反光对结果的影响并不太大，在考虑到运算速度的情况下，我们仅执行一次腐蚀操作。

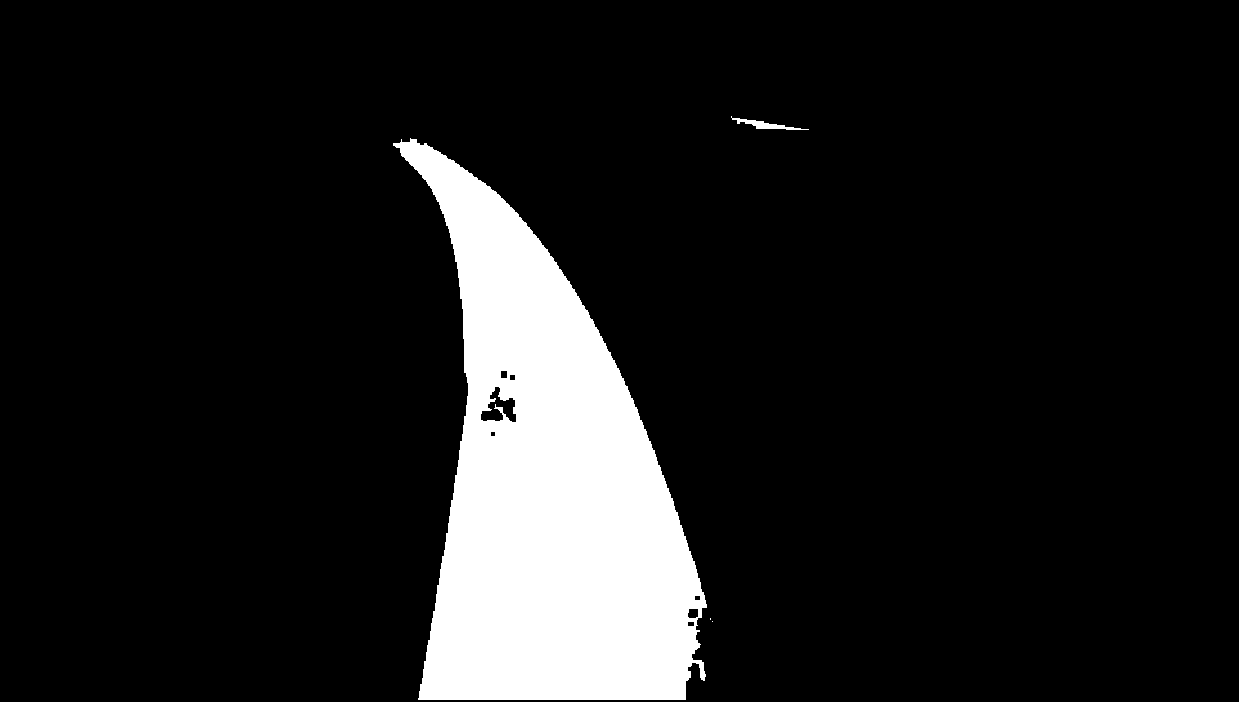


图7.2.2.腐蚀模块处理后的图像（左）和如果加上膨胀模块后的结果（右）

图7.2.3展示了巡线过程中小部分阈值分割和腐蚀模块处理效果差的图像。可以看出阈值分割后由于光线问题，黑线上有坑坑洼洼的点，在腐蚀操作后更加明显。尽管如此，我们仍然可以通过黑线中心的位置判断黑线的方向趋势并作出响应。即使在面临旋转角速度不足的情况下，后续的决策模块中的后退机制也能够有效地解决这个问题。

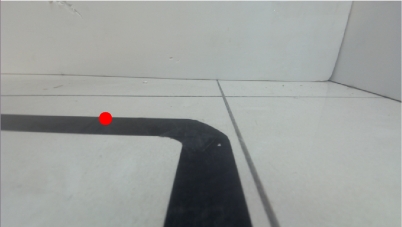
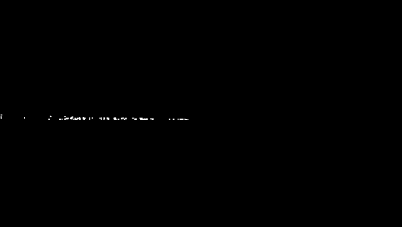


图7.2.3.效果较差的原始图像（左上）阈值分割后图像（右上）腐蚀操作后图像（左中）如果加入膨胀操作后图像（右中）范围选择后图像（左下）由此定位的黑线中心（右下）