東南大學

实习报告

项目名称：全国大学生智能汽车竞赛人工智能创意赛

项目负责人：毕志海　　学号：（08117106　　）

项目参加者：欧亚明　　（学号：08117105　　）

谢家鹏　　（学号：　08117103　　）

　　（学号：　　）

项目指导教师：谈瑛姿、孙琳

项目验收时间 _____

目　录

摘要

本文详细介绍了东南大学大学“机器不学习”在第十五届全国大学生智能汽车竞赛人工智能创意赛中制作的具有智能交通和如影随形功能的智能小车。小车的智能交通部分分为赛道检测和目标检测，对于赛道检测，我们使用手柄或 CV 找中线控制小车移动采集数据集（包括打角和对应图片），经过一定的预处理后，在优化过的 CNN 模型中进行训练获得最终模型，之后在小车部署模型实时获取打角；对于目标检测，我们将赛道检测数据集中的图片标注后通过 yolov3-tiny 模型训练并在车上部署，小车就可以实现目标检测，最终两个模型串行运行并在一定控制策略作用下，小车可实现智能交通。智能交通的避障任务我们采取 CV 提取连通域的方法实现，最简单地，小车只走灰色较宽的一边，在一些细节上优化后，小车可实现效果较好的避障。如影随形部分，我们使用图像处理时间较短的 CV，通过 CV 识别特殊标志物来追踪，实际中小车能全速精准跟随目标，追踪效果很好。综合来说，本方案实现了智能车的智能交通和目标追踪功能，具有一定的应用前景。

关键词：智能交通；赛道检测；目标检测；单目摄像头避障；目标追踪

一、绪论

1.1 课题背景

近年来随着科学技术的不断进步，人工智能技术也得到了极大的发展。自从人工智能技术诞生之后，在很多领域中都得到了广泛应用，其中在汽车制造行业，人工智能技术的应用逐渐使得自动驾驶智能汽车研究得到了更大的发展。[1]

智能车是以汽车电子为背景，涵盖控制、模式识别、传感技术、电子、电气、计算机、机械、自动化等多科学的科技创意性设计，一般主要由路径识别、速度采集、角度控制及车速控制等模块组成。其设计与开发涉及控制、模式识别、传感技术、汽车电子、电气、计算机、机械等多个学科。总体可以分为三大部分：传感器检测部分，执行部分和 CPU。可实现自动驾驶、自动变速及自动识别道路等功能，是现代电子产业发展中一项重要的组成部分。本次创意赛总体为自动驾驶的缩小模型。

1.2(相关技术)研究现状

1.2.1 AI 在汽车领域的应用

通用汽车：通用汽车在设计阶段引入人工智能技术，利用 Fusion 360 学习原始的汽车零件设计图，之后人工智能算法会根据现有的条件、重量、需要、材料等提供上百种不同设计，供设计师选择。该步骤可以减轻零件重量，优化安装流程，增加零件耐用性。

宝马汽车：宝马汽车利用人工智能优化汽车冲压过程。宝马推出的先进的测量和分析系统，可以针对每一部分材料分别收集数据，智能化调节冲压模式，让整个流程可以更精确和有针对性，产出的产品也更加稳定。

起亚汽车：起亚汽车人工智能销售聊天系统采用了 CarLabs.ai 的人工智能销售系统，让用户可以直接与机器人对话并解决销售中的问题。该虚拟助手利用了自然语言处理技术，

分析用户的问题与答复，充分理解后提供特定的答复。该聊天机器人目前主要存在于Facebook的平台内，可以为用户提供偏好匹配，车型比较，车辆细节信息，线下商店信息，试驾预约，财务计算等功能。

吉利汽车：吉利汽车借助人工智能技术将语言识别率提高到98%以上。该语音助手可以把大量需要触控的操作转换成语音操作，让驾驶员在驾驶时候更加安全，不需要转移视线即可完成众多命令。同时，由于语音助手的灵活性，车主也可以更方便的接入其他汽车服务，打通车内空间与车外世界。

1.2.2 目标检测

目标检测是计算机视觉领域和机器学习领域的研究热点,至今已有接近20年的研究历史,从过去的Viola-Jones Detector,DPM,到近6年来出现的R-CNN,OverFeat,以及后面的Fast R-CNN,Faster R-CNN,SSD,YOLO系列,再到2018年出现的Pelee.目前,基于深度学习的目标检测技术主要分为两类:two stage的目标检测算法和one stage的目标检测算法.two stage的目标检测算法是先由算法生成一系列作为样本的候选框,再通过卷积神经网络进行样本分类;one stage的目标检测算法不用产生候选框,直接将目标边框定位的问题转化为回归问题处理.正是由于两种方法的差异,其性能也有所不同,前者在检测准确率和定位精度上占优,后者在算法速度上占优.

1.3 本项目研究内容

本届人工智能创意赛全国总决赛由两个比赛科目构成：智能交通和如影随形，两科目的成绩之和，作为本组别的总成绩。

1.3.1 智能交通部分规则：

比赛开始时，无人车从起点线出发，沿着车道线行驶、运行途中需要识别人行道，经过上坡和下坡，经过路障区，需要识别并避开障碍区，继续行驶后，来到限速路段，需要识别限速标志和取消限速标志，然后继续行驶，来到了变道超车区域，需要变道并线行驶，并绕过前车，进行往前行驶，进入环岛区域，绕环岛一圈后，识别左转标识，左转驶向终点线，过了终点线后，算完成了行驶任务。

运行过程中对于赛道元素应有以下处理:

1、无人车在人行横道前需停车1S；

2、无人车在坡道红线处，需要停车1S，然后再继续前行；

3、在路障区时，无人车不能触碰障碍物；

4、在限速标志路段，行驶的时间不能少于 8 秒；

5、在变道超车区域，须进行变道行驶；

赛道元素未处理有如下判罚：

1、人行道未停止或停止时间少于 1 秒，则加罚 5 秒；

2、坡起红线前未停止或停止时间少于 1 秒，则加罚 5 秒；

3、在限速路段，用时少于 8 秒，则加罚 8 秒；

4、在路障区域，每触碰一次锥桶，加罚 5 秒；

5、在变道超车区域，不进行变道行驶（碰到小红车），计比赛失败；

智能交通部分设有附加题倒车入库，从车头在红线处开始倒车并进行计时，必须用倒车的方式倒入车库，用时需在 10S 内完成。倒车入库成功（四个轮子均在车库内，且不压黄线），奖励减免时间 8S。如果倒车入库后，无人车在车库内，车轮未出车库，只压到边缘黄线，则奖励的减免时间为 5S。车轮越过黄线驶出车库或倒车时间超过 10S 计倒车入库失败。

运行过程中，小车每压一次线罚 2S，持续压线 5S 计行驶失败；车模越过边线冲出赛道（四个车轮都在赛道外）或者中途运行停止，或运行方向错误，计行驶失败。

无人车从起点发车至越过终点线所用的时间计为有效跑完全程时间。智能交通最终的时间为有效跑完全程的时间+违规加罚时间-附加题奖励时间。最终的成绩时间越短，成绩越优。

1.3.2 如影随形部分规则：

在指定大小的场地上，人要根据标志物指定的路径，引导小车沿指定路径绕行一周，小车运行过程中不能碰到场地中摆放的锥桶，从小车冲出标记线开始计时，以小车冲过标记线为计时停止标志。

如影随形部分判罚为：小车或人碰一次锥桶罚时 5S。

如影随形可使用手持标志物或身体部位引导小车，使用标志物时标志物不能贴地，且要离地 5-10 厘米。小车运行过程中必须是跟随人的轨迹行驶，否则计比赛失败。

小车运行的整圈时间作为如影随形部分的基础时间。如影随形最终的时间为有效跑完全程的时间+碰撞锥桶加罚时间。最终的成绩时间越短，成绩越优。

1.3.3 最终排名依据：

人工智能创意赛全国总决赛的最终排名按照智能交通和如影随形两个部分的时间之和逆向排序。两个部分时间之和越短，排名越高。

二、系统总体设计方案

2.1 （设计原则与理论基础）

本次任务主要有车道线识别与标志物识别以及如影随行，其中车道线识别方面包括方向与速度的预测，依靠识别车道的弯曲程度，匹配不同的速度和方向转角，使行驶更加高效。在标志物方面对比了基于 VGG-16 的 SSD 模型和 YOLOv3 模型，最终采用的使对小物体检测更加友好的 YOLOv3 模型，识别效果达到预期，鲁棒性强。如影随行目标检测方面也采用了 YOLOv3 模型，识别效果同样出色。

本次的创新点主要有以下几点，第一：车道线识别中加入了方向和速度的识别，采取在直道高速，弯道低速的方法，使得车辆更加灵活；第二：除了基础任务中要求的标志物识别外，额外增加了障碍物识别。第三：在如影随形控制策略中加入死区控制，大于一定角度后小车原地旋转，保证小车能通过狭窄区域。

2.2 （系统总体框图）

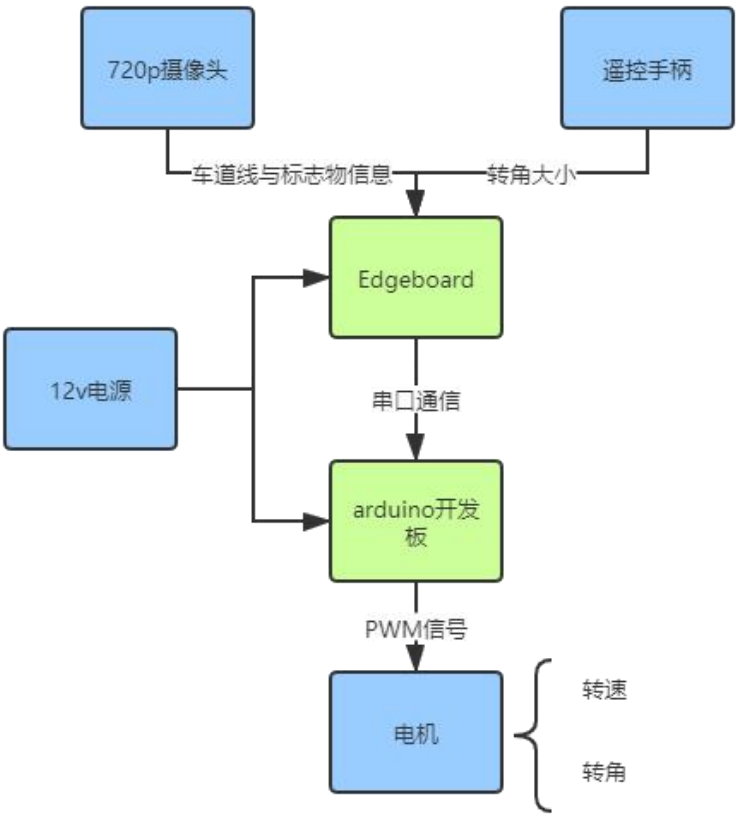

图 1 系统总体框图

2.3 （具体方案阐述）

2.3.1 车道线与标志物采集代码的实现

车道线模型的训练需要摄像头图像数据，以及同时刻小车方向转角的输出，或同时刻小车速度输出数据。

创建 3 个进程：小车控制进程 control_car_process，图像保存进程 save_image_process，转角数据保存进程 save_data_process，同步运行。

control_car_process：固定小车的速度输出值，与无线手柄建立连接，读取手柄摇杆角度，计算得到相应的小车转角输出值。（如果是采集速度训练数据，则读取手柄按键状态，得到相应的小车速度输出值）。将速度与转角值通过串口发送给下位机。

save_image_process：保存摄像头拍摄的图片，分辨率为 320*240，保存格式为“序号.jpg”，序号从 0 开始递增。

save_data_process：保存转角输出数据，写入到文本文件中，再转换为 npy 文件方便处理。数据采集的关键是摄像头图片与转角数据的一一对应，依靠互斥锁来实现。

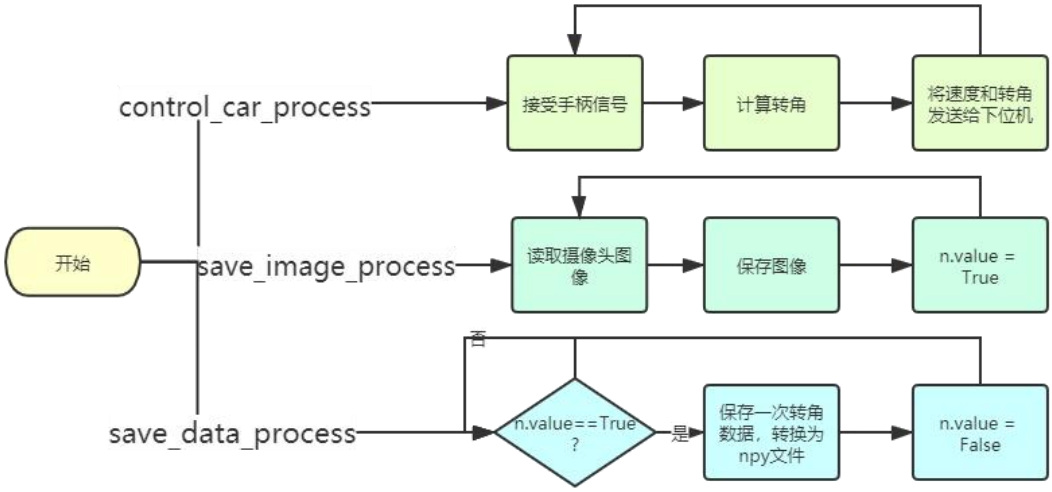

图 2 采集程序流程

方向训练数据的采集与速度训练数据采集分开进行，两者之间的切换只需要更改 save_data_process 中保存的数据。

标志物数据的采集比车道线更为简单，只需要单纯的摄像头采集到的真实道路图片即可。这里采用和车道线识别相同的程序，后期再手动挑选出含有标志物的图片进行进一步处理。

2.3.2 车道线数据处理及模型训练的实现以及优化方案

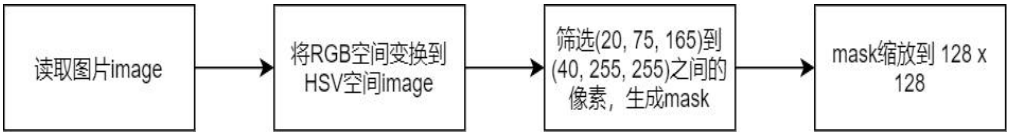

图 3 处理程序流程

车道线的分割和提取采用传统的图像处理算法，在 HSV 空间下黄色的车道线会呈现出一个在一个特定空间内分布的趋势，即在(20, 75, 164)到(40, 255, 255)内为黄色车道。车道线提取的效果如下:

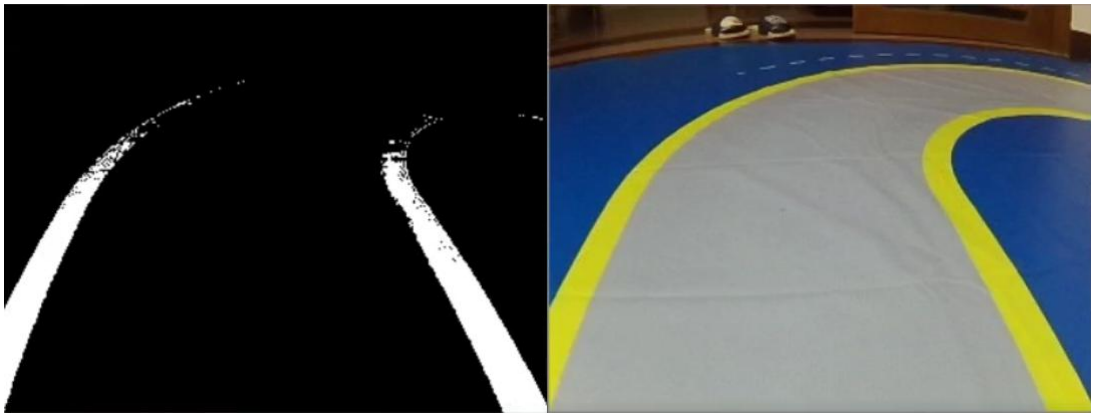

图 4 提取效果

实际输入到网络中的输入图片实际上是被认为是车道线的 mask，这显著减少了网络需要处理的数据量，加快算法的收敛速度。上面的逻辑图展示了作为神经网络输入前所需要执行的预处理步骤。

车道线处理的另一种方式是使用语义分割，可以部署轻量化的 Mask-RCNN。但其处理的速度并不能满足要求，因为车道线的信息将会直接用于决策当前的运行速度和车辆转角。这一任务对实时性要求较高，而使用语义分割方法的速度达不到实时性要求。

我们需要从车道线中提取出车辆的转角这类信息。即输入一个二值化的 mask 图片，输出一个 scalar 角度。这一过程被建模为回归过程，使用的网络如下：

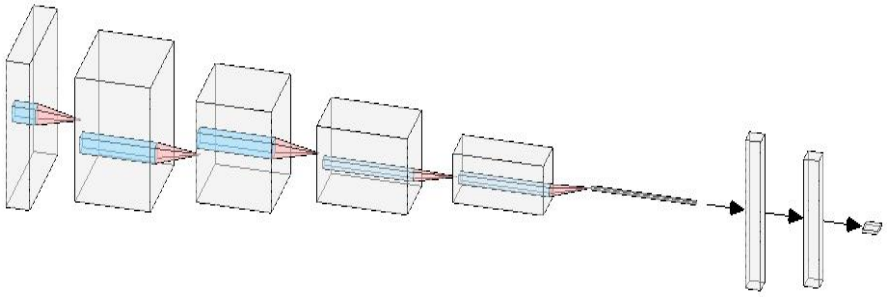

图 5 模型框图

具体参数为:

层数	类型	filter	size	stride	padding
1	Conv2D+ReLU	32	5	2	0
2	Conv2D+BN	32	5	2	0
3	Conv2D+ReLU	64	5	2	0
4	Conv2D+BN	64	3	2	0
5	Conv2D+BN	128	3	1	0
6	FC+Dropout		128		
7	FC+Dropout		64		
8	FC		1		

网络估计归一化的角度，计算公式如下:

$$\widehat{\rho} = \frac{\rho - 500}{2000}$$

对速度的估计同样采用了相同的神经网络模型，对估计的速度同样采用归一化处理。

$$\hat{v} = \frac{v - 1500}{100}$$

车道线信息分析逻辑图如下:

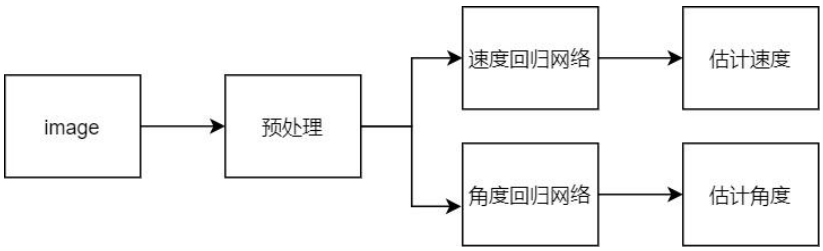

图 6 速度与角度获取

速度和角度回归模型的训练数据获取主要采用两种方式。角度回归模型采用手工标注的方式，将单张图片与在该情况下应该使用的转角联系起来。而速度回归模型的训练数据为人工操控小车在赛道上运行的时的记录数据，这使得速度回归网络可以复现人工车手在赛道上运行的速度。

2.3.3 标志物数据处理及模型训练的实现以及优化方案

2.3.3.1 数据标注方案

数据标注的准确性对识别也具有一定影响，本次检测的目标由于有圆形、矩形，考虑了两种标注方案：

2.3.3.2 多边形标注

多边形标注的优点是更更好的应对本次识别的物体三种不同的形状，能尽可能减少其他背景带来的影响，但是它最麻烦的地方就在于返回的坐标格式不一致，其更像是图像、语义分割，代码较为复杂，当然准确率可能较高。

2.3.3.3 统一矩形标注

利用传统的矩形标注，最大的优点就是方便快捷，对于任何目标，都标注最大外接矩形，并且代码方面返回的坐标形式一致（xmin,xmax,ymin,ymax），相对来说更加简单，但可能受到目标外的其他背景影响。

2.3.3.4 数据处理方案

数据处理方面，采取的方案主要是对采集的数据进行数据增强，并且要考虑好应该如何增强，初步方案是针对不同的干扰，进行相应的增强。关乎到图片的明亮程度等光照方面的影响、由于小车速度等造成图片模糊程度的影响、还有标志物识别角度不同的影响等。

2.3.3.5 模型选择方案

近年来，目标检测的算法不断涌现，如下图所示，深度学习方面的目标检测算法大致可

以分为两类，一类是 one stage 算法，另一类是 two stage 算法。区别就是 one stage 算法直接回归物体的类别概率和位置坐标值，只有一个 loss 函数，其中包括类别损失和位置损失；而 two stage 算法是通过算法生成一系列候选框，然后再将这些候选框输入至全连接网络进行分类[2]，也就是要分两步进行。两者最大的区别就是，one stage 算法速度方面优于 two stage，而 two stage 算法在较为复杂的检测方面，精度优于 one stage。

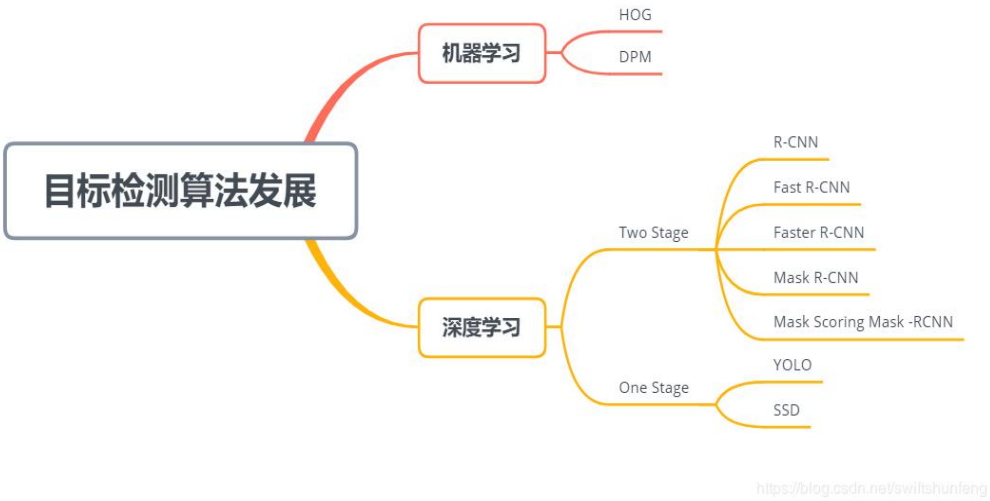

图 7 目标检测模型总览

考虑到本次任务中，考虑到此次识别任务其实背景都是赛道，识别的难度其实不大，再结合训练时间的考量，决定采用 one stage 算法的模型 SSD 和 YOLO。

2.3.4 优化方案

2.3.4.1 负样本难例挖掘

利用难例挖掘的方法排除一部分样本，思想就是将所有负样本按照正向传播计算出来的 loss 排序，选择 loss 最高的前 n 个预测框与候选负样本集匹配，匹配成功则作为最终的负样本，参与到反向传播中去，并可以适当控制最终的正负比例，一般为 1：3 左右。

2.3.4.2 非极大值抑制

对于输出的结果有重复框的情况，可以按照置信度将重叠度高的框筛选掉。具体步骤为：

- 根据框的置信度得分进行排序；
- 选择得分最高的边界框添加到输出列表；
- 计算所有边界框的面积；

- 计算置信度最高的框和其他候选框的 IOU；
- 删除 IOU 大于一定阈值的候选框。

2.3.4.3 学习率调整策略

训练过程如出现 loss 为 nan 或者 inf 的情况，或者 loss 维持在一个较高的值不往下降，则可以采取以下优化方法：

- 相应调小学习率；
- 增加更多的数据集；
- 尝试更多学习率调整策略，比如说 StepLR、ExponentialLR 等；
- 对于模型的初始化，可尝试做权值迁移，用已经训练好的相似模型的参数，作为当前模型的初始化参数。

2.3.4.4 小车自主运行代码的实现或优化方案

根据摄像头图像，提取出车道线，再分别载入模型进行预测，得到速度和转角的输出值。程序流程如下：

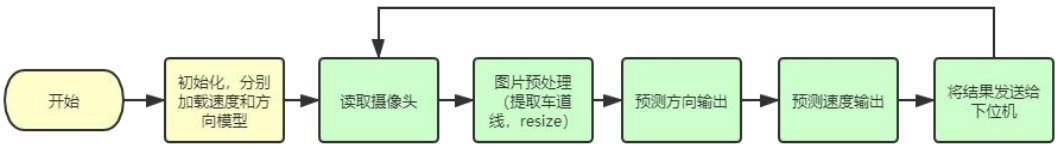

图 8 自主运行流程

初始示例代码中只有方向转角预测，而速度是恒定输出的，这导致了速度过快的时候容易出现拐弯不及的情况，加入速度控制，通过预训练的模型预测速度输出值，直道高速，弯道低速，一定程度上可以解决这个问题。

2.3.4.5 目标检测后，控制小车移动代码的实现或优化方案

在识别车道线，根据车道线行驶的基础上，加入目标检测，根据目标检测的结果控制小车完成指定动作。流程如下：

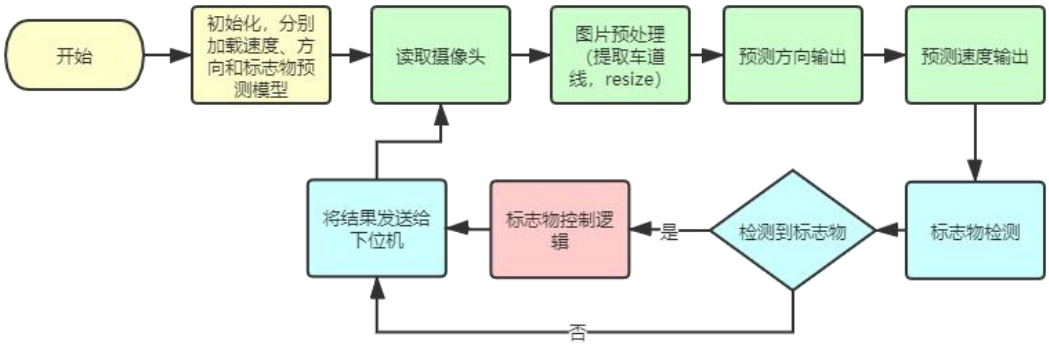

图 9 标志物检测自主运行流程

如果检测到标志物（置信度高于阈值，且标志物在图片中的大小和位置满足要求，太小或者位置不正确的标志物会被忽略），则进入标志物控制逻辑。

总共设置了 9 种标志物：

1. Start 起点
2. zebra 斑马线
3. limit_10 限速 10
4. limit_20 解除限速
5. obstacle 障碍物
6. park 停车
7. redline 红线
8. straight 直行
9. turn_left 左拐

在基础任务要求上加入了障碍物标志和红线。

标志物控制逻辑如下：

1. start 起点
   将 stopFlag 置 0，允许通行。
2. zebra 斑马线
   将 stopFlag 置 1，速度置 0，停下小车。
3. limit_10 限速 10
   将 limitFlag 置 1，降低小车速度输出，同时也要降低方向输出。
4. limit_20 解除限速
   将 limitFlag 置 0，速度限制解除。
5. obstacle 障碍物
   执行避障操作，策略是靠左行驶，绕过障碍物之后向右回到原车道。
6. park 停车
   将 parkFlag 置 1，小车速度置 0，并结束程序。
7. redline 红线
   如果画面检测到红线，停车 1s。
8. straight 直行
   保持直行。
9. turn_left 左拐
   启动定时器，计时 0.4s 后执行强制左转操作。

2.3.5 小车底层 arduino 烧录程序的实现

底层 arduino 通过串口获取上位机发来的数据帧，通过解析数据帧来调整 PWM 输出，从而实现控制小车速度和方向的目的。

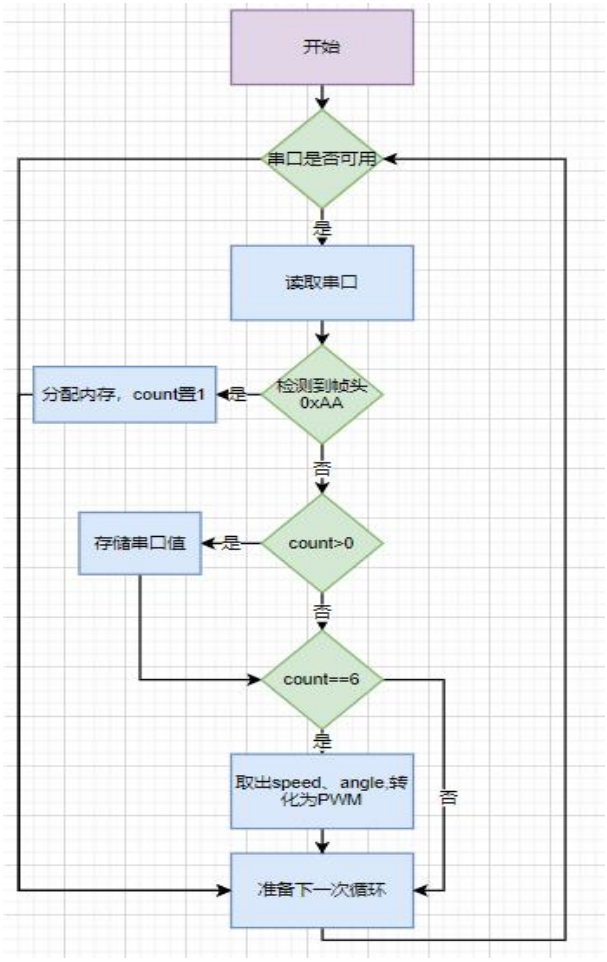

图 10 arduino 烧录程序

2.3.6 单目摄像头避障的实现

单目避障使用使用目标检测实现对障碍物检测，记录图片中障碍物的位置。根据小车模型的大小对目标障碍物的大小进行适当的膨胀。然后在图像空间内将障碍物和道路栅格化，并使用 A*算法规划避障路径。

本实现中的路径规划目标为较前方的中间点，如果该点不存在，则逐步将中间点的位置下移，直到存在路径点，并且存在一条可行路径为止。

更加合理的方式是使用双目视觉进行密集点视觉 SLAM 建图，并设计探索算法得到在陌生环境下的全局路径规划和局部避障功能。但在本设计中，鉴于车载计算设备算力和传感器的限制，不能采用该方法。

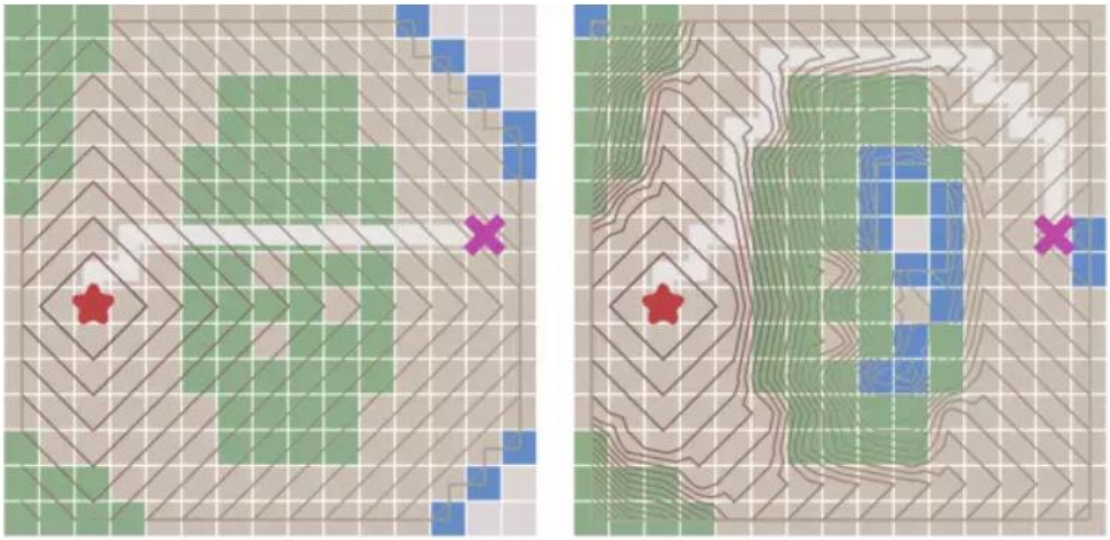

图 11 避障

2.3.7 如影随形的实现

如影随形主要包括两部分，一部分是目标检测，另一部分是位置跟随。

2.3.7.1 目标检测

为了实现更好的目标检测效果，我们选用了特征非常明显的足球作为检测对象，如下图所示：

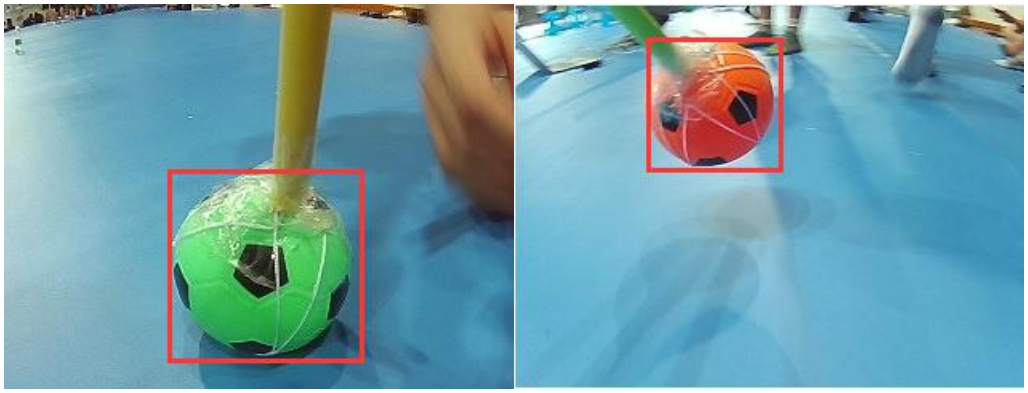

图 12 目标检测

2.3.7.2 位置跟随

当我们通过目标检测算法获取到图像中的目标位置信息后，就可以通过反馈算法来控制小车的转向和前进或后退，状态机如下：

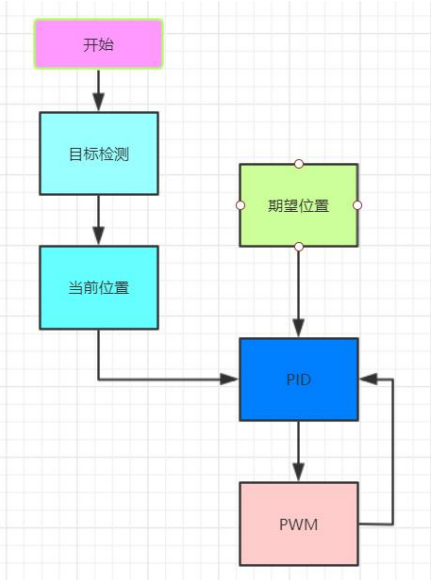

图 13 arduino 烧录程序

三、系统（软硬件）具体实现

3.1 训练数据的采集

使用手柄遥控小车完成训练数据采集。左摇杆控制小车方向，按下 Y 键小车速度切换为高速，按下 X 键速度切换为低速。

图 13 遥控手柄

为了尽可能保证训练数据质量，速度训练数据与方向训练数据需要分开采集，采集方向训练数据时，速度设置为恒定值；采集速度训练数据时，不记录方向数据。

训练方法是：方向模型数据训练时，直道尽量保持直行，偏离方向要及时修正，弯道要平稳过弯，调整方向输出大小直到在弯道能保持推动摇杆，最好不要采用连续轻推摇杆的方式过弯；速度模型数据训练时，进入直道按 Y 切换高速模式，即将进入弯道按 X 切换低速模式，在路线交错的地带一律使用低速模式。

在具有直行、左转标志的地方，小车完全可以用标志物识别的逻辑来控制，但是在有直行标志的地方仍然可以通过车道线识别来控制小车正常通过，在左转标志处通过常规车道线识别模型训练无法使小车学会拐弯，并且在左转处手动操作小车容易出现出界等情况引入污染数据，所以每次训练从起点开始，按正常路线行驶到标记位置处结束，保证数据质量。后期多组数据拼接成数据集进行训练。

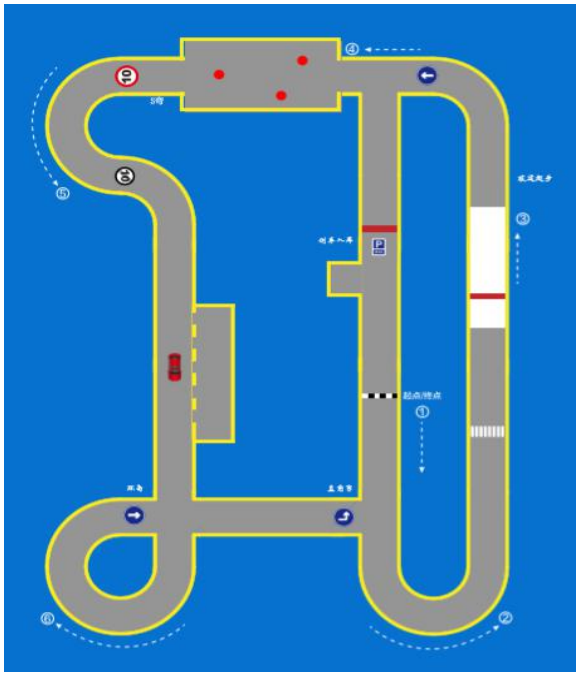

图 14 训练赛道

最终方向模型和速度模型各采集 10 万左右数据量进行训练。

3.2 车道线数据处理及模型训练

角度和速度回归模型所采用的网络参数在前面的章节有详细的叙述，本节将记录网络的数据集情况。用于车道角度回归的数据集总共包括了 8473 张图片，均为在实际的赛道上采集。每张图片对应了一个可以在赛道上运行的转角。例如下图的标签为 2500。

图 15 弯道采集

用于训练速度回归的模型需要输出在当前情况下可行的最快速度，这一速度我们没有办法直接估计，而是创造性地使用人工操作的方法，用手柄尝试在当前情况的最佳速度，并且将当前图片和当前手柄的指令记录下来。总体而言的思路是希望在直道时小车尽可能加速，并且根据弯道与车辆的距离来估计减速。通过这一创造性的方式我们获取了一个拥有 10638 张带标注图片的数据集。每一张图片都附上了一个期望速度。例如:

图 16 直道采集

正处于大直道上，与之相对应的速度为 1600。

3.3 标志物数据处理及模型训练实现

3.3.1 数据标注实现

经过数据采集后，首先将属于同一标志物的可用图片进行初步筛选归类，然后进行数据标注。尝试采用了两种标注方式，分别为多边形标注与矩形标注结合，或者是全为矩形标注。由于后期考虑到程序的可重用性，最终选择了更为简单高效的矩形标注，数据标注软件为 labelimg。

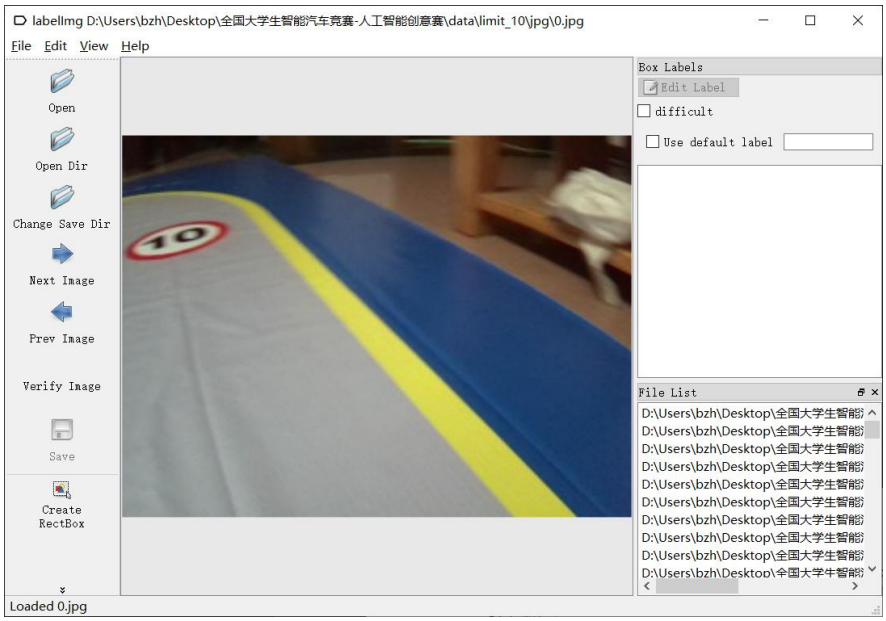

图 17 数据标注

3.3.2 数据处理实现

数据处理是为了更高效的应对不同环境下识别标志物的鲁棒性问题，数据预处理对减弱干扰的影响有很大作用。在此任务中，我们总结了以下几个主要的干扰：

3.3.3 光照情况

由于场地并不是全封闭，这种情况下，白天和晚上或者阴天和晴天等光照条件不一，并且赛道本身的材质也会造成一定的反光。因此如何解决光照对标志物的识别造成的影响很关键。Zuyao 提出了两种思路，第一是通过场地解决问题，尽量将场地的外光源遮挡，使用室内光源，但是此方法治标不治本。第二种思路是通过数据增强解决，针对光照问题，我们通过改变图片的对比度、饱和度、锐度、明亮度、高光等来模拟不同的光照情况。

图 18 亮度处理

3.3.4 标志物角度

在小车行驶过程中，不能保证每次到同一个位置，车身相对于标志物的的位置相同，也就是说不能保证摄像机拍摄的标志物处于同一个角度，并且小车与标志物的距离不同，相对拍摄的角度也不同，因此在训练前，我们需要对采集的标志物图片进行一定的旋转增强，可大大提升识别的鲁棒性。

图 19 角度处理

3.3.5 速度与抖动

小车的速度和摄像头在行驶过程中的轻微抖动，都会影响到拍摄图片的质量，根据经验，此情况下拍摄的图片具有残影等，并且速度不一致，模糊程度也不一致，于是可通过改变图片的模糊程度、锐化程度模拟不同速度下采集的数据进行训练，保证了日后速度提升的情况下，仍然能稳定地识别出标志物。

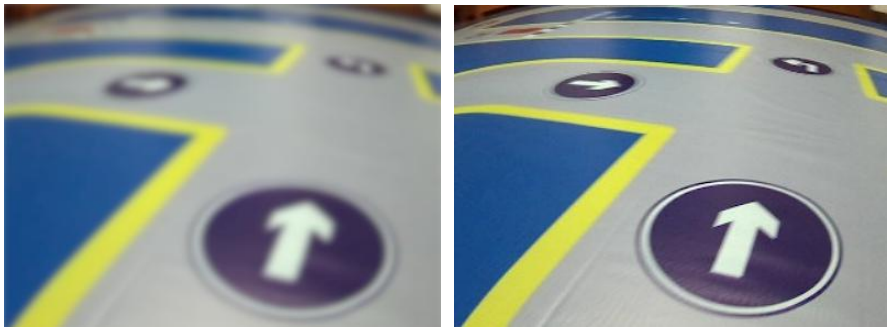

图 20 模糊处理

3.4 模型实现

虽然学习手册中给出了 AI Studio 上一个基于 YOLOv3 的模型，但是为了尽可能提高识别的准确性，我们尝试利用 AI Studio 平台实现基于 VGG16 的 SSD 模型和 YOLOv3 模型[3]做一定对比，选出综合变现好的一方。在 AI Studio 中，由于模型都封装得很好，如果仅仅是使用这一层面来说，只需要短短几行代码即可，但是为了对模型有更深入得理解，得出更好的识别效果，对模型的整体思想和网络结构总结如下：

3.4.1 基于 VGG-16 的 SSD 模型

SSD[4]的整体思想是在不同的卷积层的特征映射图上，以每一个像素点为中心，按照一定的规则画不同比例的矩形框，并将这些矩形框映射到原图中去，与给定的 label 进行比较，计算出框取的位置偏差和分类的置信偏差，根据偏差更新模型参数，不断迭代，最后希望达到预选的框和给定的目标框几乎重合。对于整个模型框架的理解，绘制出以下模型结构图：

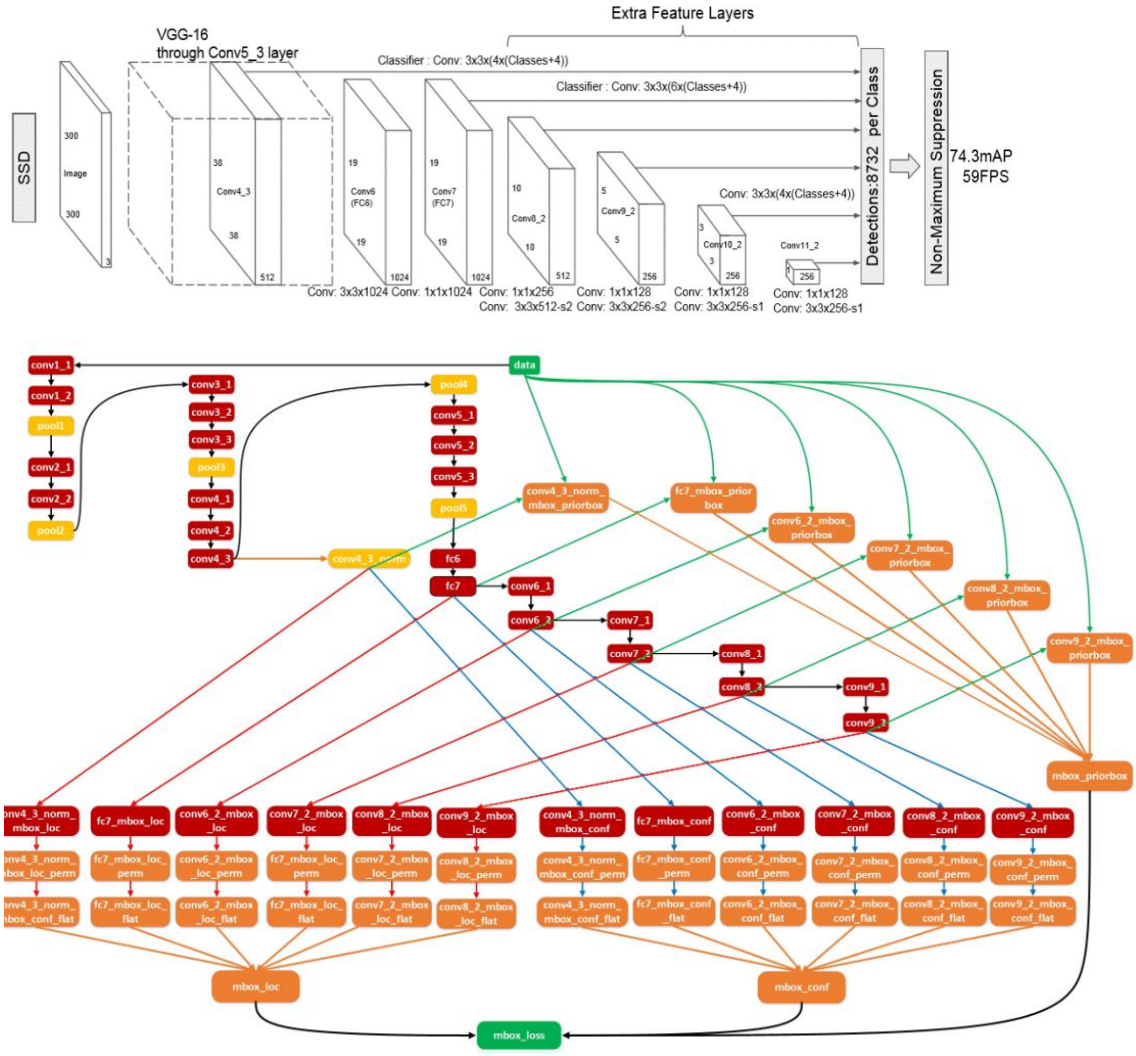

图 21 SSD 模型结构

网络总体来说一共有三个部分

基础网络：提取低尺度的特征映射图

辅组网络：提取高尺度的特征映射图

预测网络：输出特征映射图的位置信息和分类信息

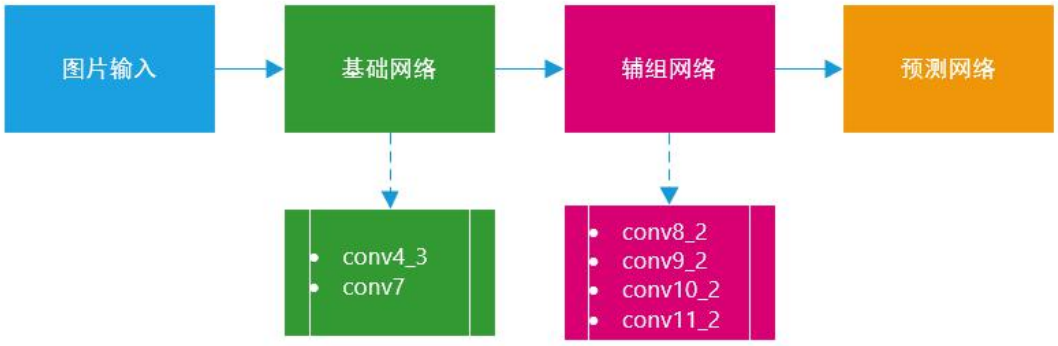

图 22 网络结构

即图片批量输入后，依次经过基础卷积层和辅组卷积层，在基础卷积层中提取出两层低尺度的特征图，在辅组卷积层中选取其中四层高尺度特征图，共得到六层特征图，并对这六层特征图进行预测。

3.4.2 YOLOv3 模型

YOLO 算法的基本思想是通过特征提取网络对输入特征进行提取，得到特定大小的特征图输出。输入图像分为 13*13 的 gird cell，如果真实框中的某个 object 的中心坐标落在某个 grid cell 中，那么就由该 grid cell 来预测该 object。每个 object 有固定数量的 bounding box，YOLOv3 中有三个 bounding box，使用逻辑回归确定用来预测的回归框。以下为 YOLO 的模型结构图：

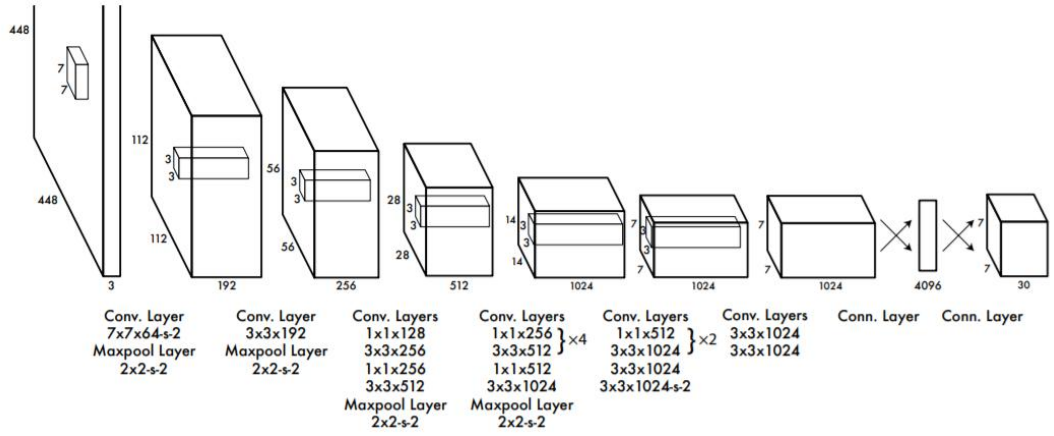

图 23 YOLO 模型结构

在整个 YOLOv3 的结构里，是没有池化层和全连接层的，向前传播过程中，张量的尺寸变换是通过改变卷积核的步长来实现的，比如 stride=（2，2），就相当于将图像边长缩小了一半。其中 YOLOv3 采用的 Backbone 为 DarkNet-53：

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

图 24 DarkNet-53

对于 YOLOv3 的输出特点：
九个 anchor 会被三个输出张量平分，根据大中小三种 size 各自取自己的 anchor；
每个输出 y 在每个自己的网格都会输出三个预测框；
使用了 logistic 回归来对每个 anchor 包围的内容进行了一个目标性评分，根据目标性评分来选择 anchor prior 进行预测，而不是所有 anchor prior 都有输出

3.4.3 YOLOv3 与 SSD 模型对比
本此项目检测中，目标物体中等偏小，而 YOLOv3 在小物体检测方面优于 SSD 模型。
由于 YOLOv3 引入 FPN 结构，同时它的检测层由三级 feature layers 融合，而 SSD 的六个特征金字塔层全部来自于 FCN 的最后一层，其实也就是一级特征再做细化，明显一级特征映射图的特征容量弱于三级，尤其是浅层包含的大量小物体特征。并且 SSD 的主网络在训练的时候小物体容易被 VGG-16 自动筛掉，而 YOLOv3 则直接把浅层网络拉出来与小物体融合，这样的结构解决了 SSD 检测小物体能力不佳的问题
因此，经过详细考虑，本次任务决定采用 YOLOv3 模型进行目标检测。
3.5 小车自主运行

方向模型与速度模型分开调试，首先调试方向模型。

训练模型时对方向输出的归一化处理是：

$$\mathrm{score} = \mathrm{float}((\mathrm{angle} - 500)/2000)$$

所以在载入模型时只需反解出 angle:

$$\mathrm{angle} = \mathrm{int}(\mathrm{score} * 2000 + 500)$$

实际运行过程中可能会发现方向输出过大或过小，重新微调预测模型输出 score 所乘和加的系数即可，例如方向输出过小导致过弯压外线时可将上式改为：

$$angle = \mathrm{int}(score * 2100 + 450)$$

小车经过不同方向弯道时发现：左转和右转转向幅度不一样。这是下位机速度控制没有闭环造成的。解决方法有两个：1、重写下位机程序，读取编码器，完成速度闭环控制；2、左右转向幅度的偏差是固定的，可以为过小的一边加上一个偏置。

方向模型调试完毕后，调试速度模型。由于训练网络模型是一样的，调试方法也相近。小车比较笨重，如果速度没有闭环的话，1600 到 1560 之间可能看不出明显的速度变化，但是通过观察程序输出日志可以验证速度控制是否正常。

3.6 目标检测后，控制小车移动

目标检测模型训练完毕后，可使用矩形框框出目标保存图片进行调试。通过矩形框大小和置信度等估算控制代码中所需的参数。

图 25 目标检测调试效果

在使用例程代码时，发现图片由蓝色和黄色倒置的情况。原因是训练模型使用的是 RGB 格式，摄像头采集的图片使用 opencv 转换成了 BGR 格式。

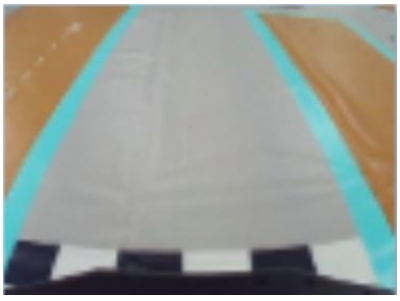

图 26 蓝色和黄色倒置的照片

只需将 BGR 转换为 RGB 即可：

$$image = Image.fromarray(cv2.cvtColor(image, cv2.COLOR\_BGR2RGB))$$

由于在有直行和左转标志处小车的控制由标志物信息决定，在识别到直行或左转标志后，启动定时器，结束计时视为进入直行或左转路段，需要屏蔽车道线识别模型的输出，强制执行直行或左转操作。执行强制操作的时间同样由新的定时器控制。识别到障碍物的操作同样由定时器+固定输出控制。

在改变速度控制模型后，强制操作的控制时间也要相应变化。

四、实现（实验）结果分析

4.1 车道线网络模型测试

方向模型训练数据量 10703 张图片，速度模型训练数据量 10638 张图片。

车道线提取实测效果可见附件视频（车道线提取实测效果）。

下面针对车道线角度回归模型和速度回归模型进行测试:

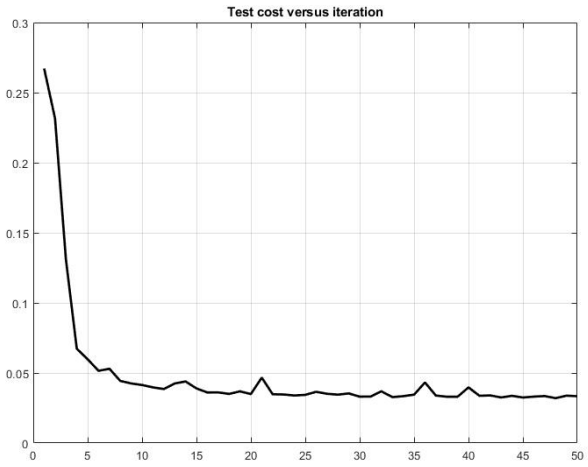

图 27 速度模型的测试集表现

上面的曲线为线速度回归模型在训练过程中在测试集上评估的表现。最终模型收敛时能够得到的角度预测模型平均误差仅有 60，能够很好的驱动小车运动。

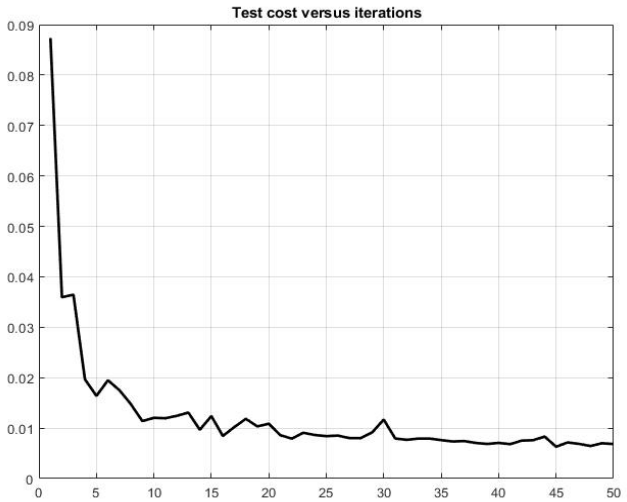

图 28 速度模型的测试结果

上图为速度回归模型在训练过程中的测试结果，在最终模型收敛时平均误差降低到 0.01 以下，这对应的速度误差是 1。模型已经能够将速度估计任务做到相当精确的等级，完全可以胜任控制小车速度的工作。

4.2 目标检测网络模型测试效果

训练中，一共有九个标志物，分别为：
Start 起点
Zebra 斑马线
limit_10 限速 10
limit_20 解除限速
obstacle 障碍物
park 停车
redline 红线
straight 直行
turn_left 左拐
其中每个标志物的训练原始数据集为 110 张，经过亮度、旋转和模糊等数据增强后，每个标志物训练数据集为 440 张，即一共 3960 张训练图片。
对于 YOLOv3 模型的测试效果图如下：

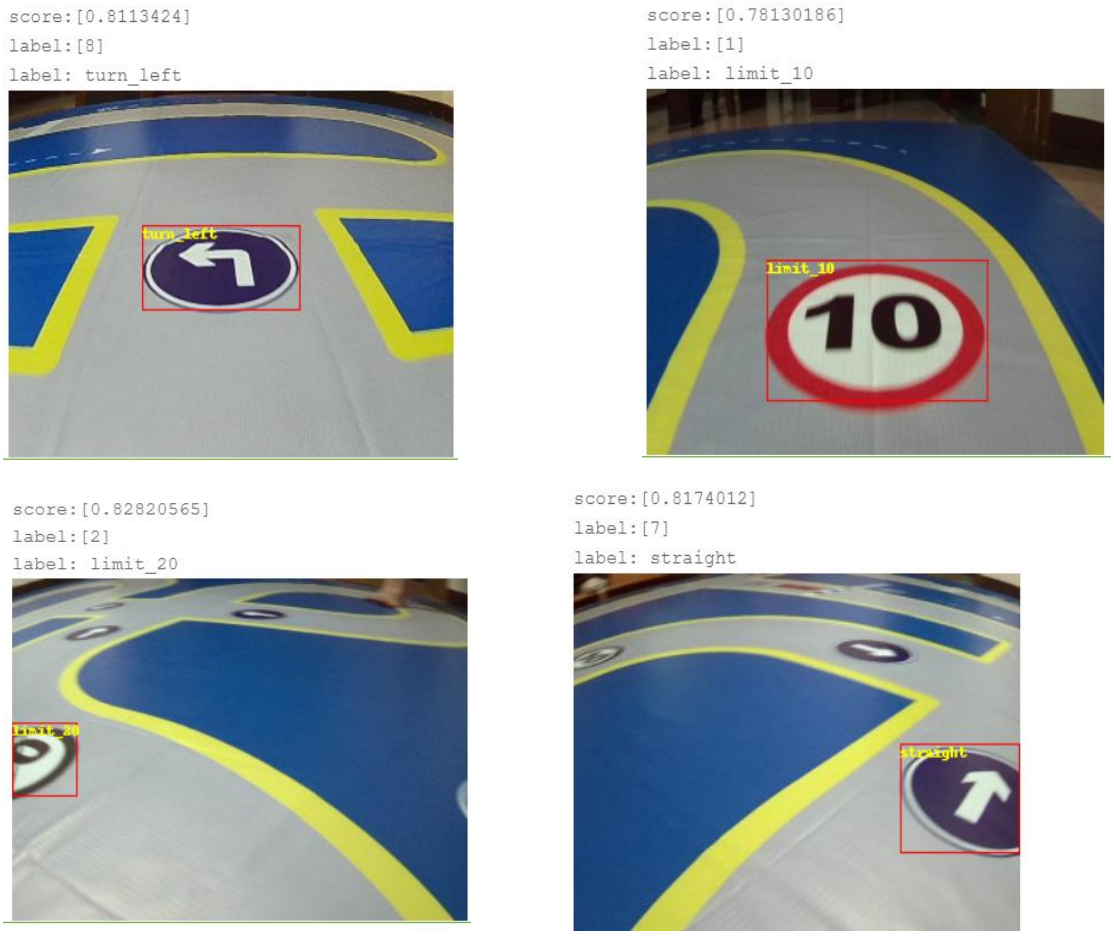

图 29 测试结果图

可见，识别的置信度均达到 0.75 以上，效果相当不错。利用 tensorboard 中可视化 loss 变化情况和目标的 PR 曲线：

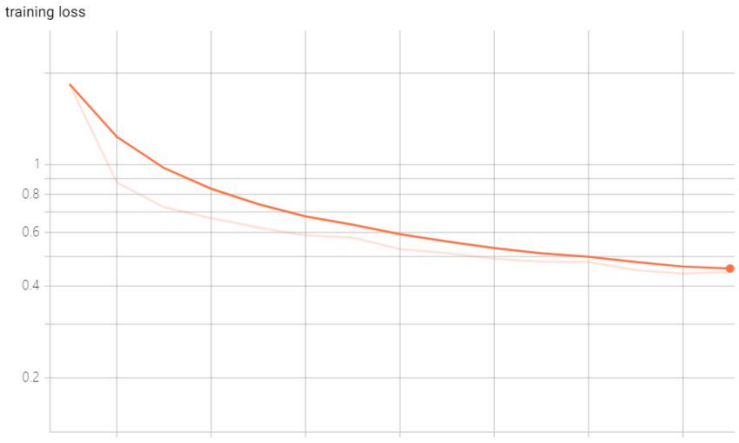

图 30 loss 收敛

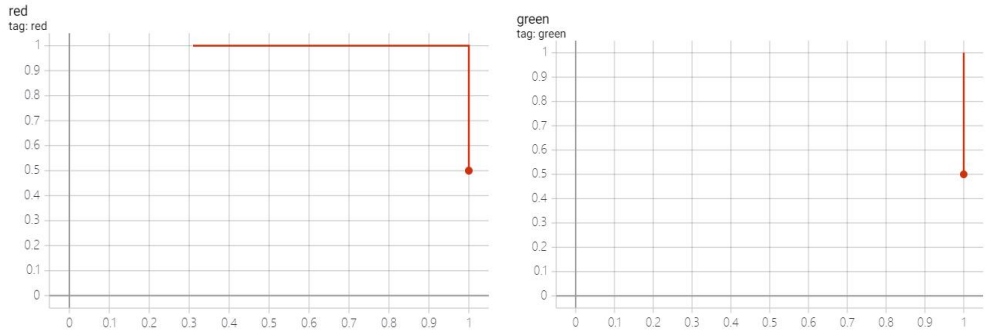

图 31　PR 曲线

由上图可见，loss 在迭代过程中趋向于收敛，并且对于标志物的 recall 和 precision 几乎达到 1。在小车上运行时，上述九个标志均能稳定识别，YOLOv3 模型的效果达到预期。

4.3 方向模型实测效果

左转和右转弯道都可以顺利通过，在直行车道可以保持在道路中心线行驶，但是在速度超过 1560 时过弯会出现不平滑的现象。见附件视频(速度和方向模型实测效果)。

4.4 速度模型实测效果

速度模型在真实赛道上的测试效果，图为小车摄像头实拍结果，左上角为速度模型实时输出值：

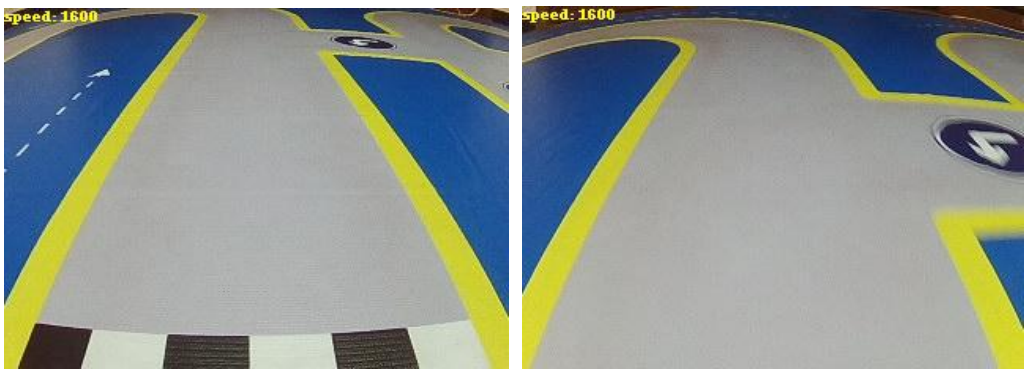

图 32 在直道速度较高

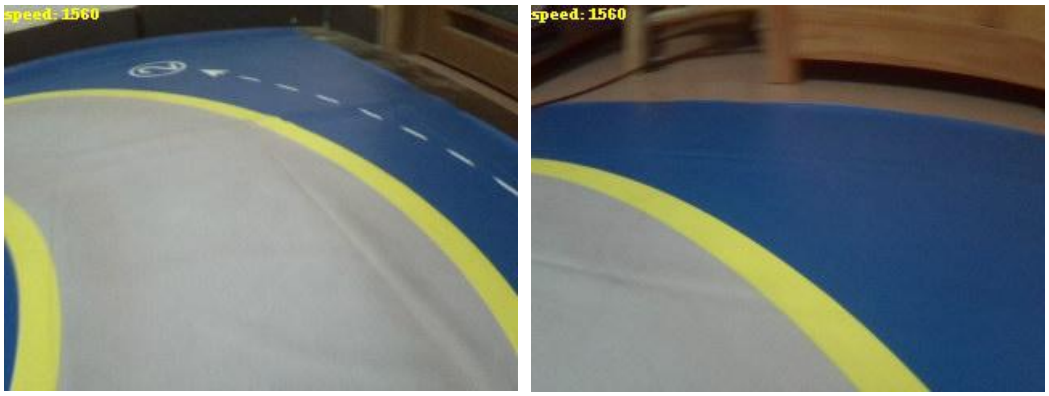

图 33 在弯道速度较低

经过测试，基本可以达到直道高速，弯道和复杂路段低速的效果，可以参考附件中的速度模型效果演示视频(速度和方向模型实测效果)。

然而就小车真实速度而言，速度的改变并不明显，最大的问题是底层没有实现速度闭环，速度提高和降低有延迟，控制不够有效。

然而使用速度模型还是有一定效果，使用后小车可以在直道以最大速度 1600 前进。如果不使用速度模型，经过测试，小车速度超过 1580 就很难通过弯道。

4.5 标志物检测模型实测效果

速度过快时远处标志可能会出现误检测的情况，但是近处标志物都能正常识别检测，且置信度大部分都高于 0.65。见附件视频（标志物检测实测效果）。

4.6 避障实测效果

在实验室环境下情况基本较好，但因为目标检测对蓝色锥桶的置信度低，在遇到蓝色锥桶时往往会导致避障失败。避障的效果和裕度可以通过调节障碍物的膨胀系数来得到。与此同时，我们加入了距离对膨胀系数的线性修正，图片中越远的障碍物膨胀系数越小，取得了更好的避障效果。

4.7 如影随形实测效果

相对于车道线行走，如影随形要简单的多，因为受环境影响的干扰比较小。我们在选择了特征非常明显的足球后，首先目标检测效果非常好，准确度可以达到 90%以上，并且每个框 score 都非常高，再通过一个 pid 来校正位姿，跟随效果非常好。

4.8 整车实测效果

以低于 1560 的恒定速度行驶时，小车可以不碰边界，沿车道线行驶，在斑马线前和上坡道的红线前停下 1s 后继续行驶，在限速路段减速行驶，通过时间 9s，顺利根据标志通过十字路口，经过大弯道，识别左转标志完成左转，最后在停车位停车。完成所有任务，顺利行驶全程。见附件视频（基本任务_不压线）。

速度高于 1560（载入速度预测模型），小车在经过第二个弯道时会有一个车轮碰到黄线，行驶完全程用时 44s。见附件视频（载入速度模型_提升速度）。

如影随行中，距离过近小车会停止，角度过大会远地旋转，其他情况控制均很平滑。

五、结论

小车在最后的测试中，速度能达到接近小车本身极限的 90%以上，在行驶过程中没有压线情况，并且对于基础部分的标志物识别，鲁棒性很强，没有出现漏识别或者误识别的情况，整体效果达到预期。

整个项目的工作主要围绕两大部分进行，分别是车道线识别和标志物识别。车道线识别方面，创新地采用了方向和速度的识别，即通过识别弯道或直道，控制小车以不同的速度进行行驶，在直道全力加速，在弯道则放慢速度，效果明显优于统一速度行驶。标志物识别方面，通过对比基于 VGG-16 的 SSD 目标检测模型和 YOLOv3 模型，并根据本次任务中识别的标志物大小为中等偏小，因此采用了对小物体检测更为友好的 YOLOv3 模型，实际测试结果中，recall 和 precision 几乎都到达 1.0，效果达到预期。

本次任务工作量主要来源于训练使用的数据集采集与处理，一共采集并使用了为 2 万余张图片，包括车道线和标志物的识别。其中，过程中耗时最长的是数据采集和数据标注。数据标注图片数量较大，耗费的精力也相对较多，数据采集方面，为了保证每次采集的数据与实际行驶路线完全一致，我们采用了单圈，多次采集的方式。

创新点有以下几点：

第一：相比于统一速度跑完全程，在识别车道线的同时，加入方向与速度识别，根据赛道的曲率区别，分别赋予小车不同的速度，此放法明显快于统一速度的跑法。

第二：加入了避障功能，当小车遇到前方有障碍物时，主动绕过障碍物，并重新回到赛道。

第三：增加停车识别功能，在斑马线处和上坡道红线处，小车将停止 1s，贴合实际情况。

本作品在识别方面效果不错，鲁棒性强。但在过弯道时，仍有一定的提升空间，目前过弯道时流畅度不够，且速度较慢，原因是速度控制不够迅速，导致弯道速度过快控制效果不佳。下一步需要为下位机编写速度闭环控制程序，提高速度控制性能。

在实际应用过程中，此作品与当前自动驾驶轿车较为相似，唯一不足的是目前小车上的设备仅有摄像头，在实际的自动驾驶任务中，仅仅依靠摄像头是远远不够的，还需要激光雷达等设备。并且当前小车应对是相对静态的情况，实际过程中，车况复杂，如果小车能很好地实现应对动态的一些特殊情况，相信能在自动驾驶领域有更高的应用性。

参考文献

[1] 戴震军.人工智能技术应用于自动驾驶汽车面临的挑战及发展趋势分析[J].无线互联科技,2020,17(06):162-163.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. Commun. ACM, 60(6):84–90, 2017.

[3] Redmon, Joseph and Farhadi, All, YOLOv3: An Incremental Improvement. Arxiv 2018.

[4] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, ChengYang Fu, and Alexander C. Berg. SSD: single shot multibox detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part I, volume 9905 of Lecture Notes in Computer Science, pages21–37. Springer, 2016.

致　谢

三周的备赛生活很快就过去，非常感谢在这过程老师、学院、学校提供的帮助，让我们得以有个非常舒适的备赛环境。当然，也很感谢主办方、南京信息工程大学的大力支持，让我们在参赛过程中没有后顾之忧。