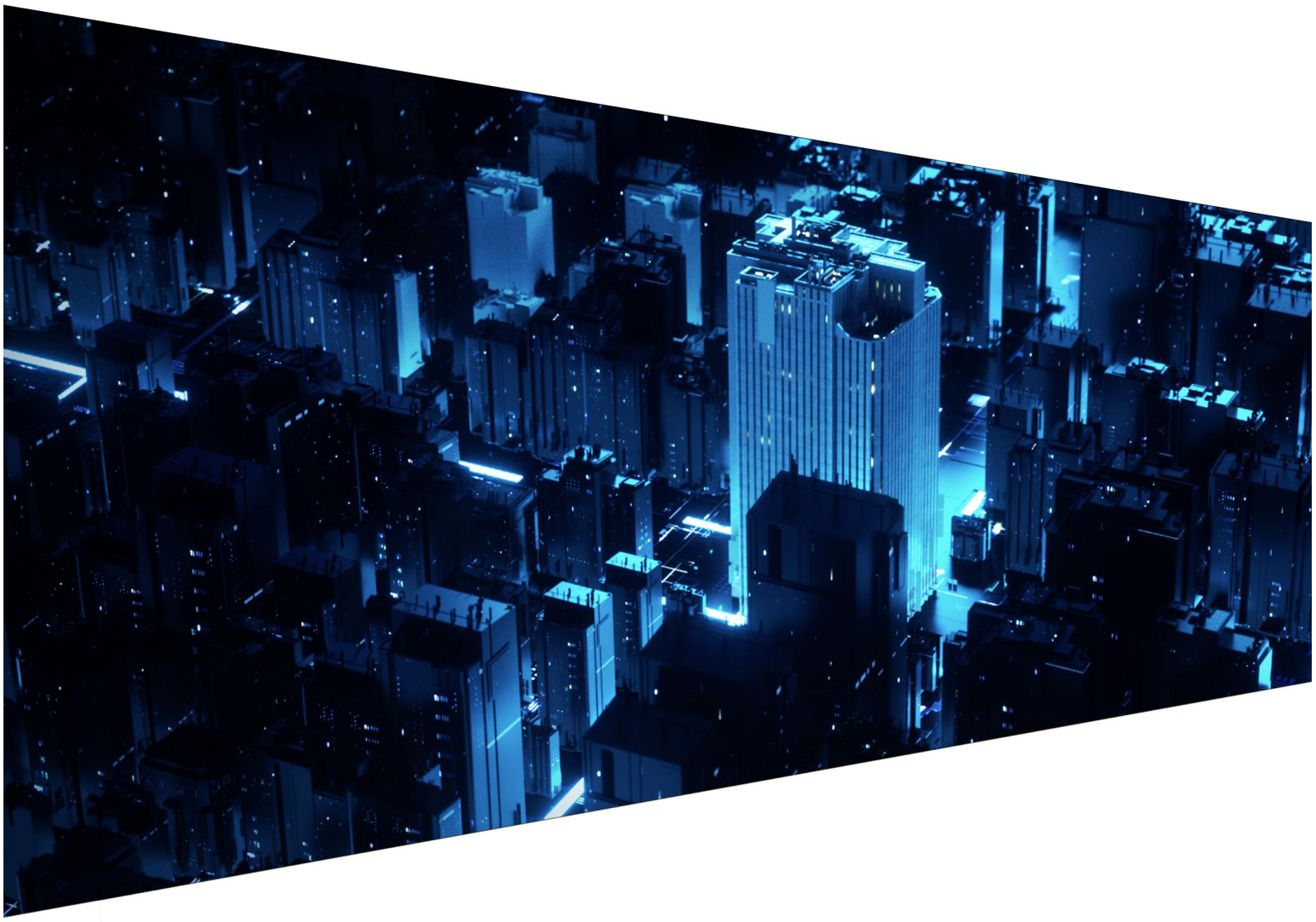


基于人工智能技术的交通灯智能控制系统设计

Design of traffic light intelligent control system based on artificial intelligence technology

目 录

第 1 章 作品概述

1.1 产生背景

交通是城市经济活动的命脉，其对于城市经济发展和人民生活水平的提高具有重要作用。然而，随着我国城市化的加快，居民收入水平不断提升，城市中的机动车保有量飞速增长，在北京、上海和深圳等地均出现了较为严重的交通问题，这些问题一定程度上困扰了城市发展、制约了城市经济建设。

而交通灯的出现在一定程度上解决了城市道路增长的有限与车辆增加的无限这一对矛盾，对城市交通规划起到了积极作用。但是随着交通工具数量的剧增，传统交通灯固定的红绿灯时长已经无法满足如今交通的需求，一旦安装，交通灯的红灯、绿灯及黄灯变换时长固定不变（如 15 秒、30 秒等），无法根据实时交通状况（车流量）进行变换时长调整，造成不必要的交通灯时间浪费，在时间调度方面灵活性差，使交通在道路拥堵时难以恢复畅通，严重影响人们的时间安排和降低行车效率。

1.2 主要功能与特色

针对传统交通灯普遍存在的时间调度不灵活问题，本项目基于人工智能技术对交通状况进行有效监控，提取出有用的数据信息，并根据监控结果对交通灯允许通行和禁止通行时间进行合理调整，有效缓解交通拥堵情况，提高人们的出行效率以及减少因车流量过大而发生的连环撞车等事故。

该项目的作品为基于人工智能的交通信号灯控制系统。系统的主要功能分为两大项，第一项是分车道车流量统计，第二项是结合车流量的交通灯控制算法。

其中，分车道的车流量统计采用自制的俯瞰视角的车道线数据集训练 LaneNet 神经网络进行车道线的检测与提取，在检测出的车道线的基础上，使用 YOLOv5 目标检测模型与 MobileNetV2 结合的 YOLOv5s-MobileNetV2 目标检测模型对车辆进行检测识别，结合 Sort 目标跟踪算法进行车流量统计，在降低系统的计算量便于部署的同时保持了合理的准确率；交通灯控制算法采用自主研究的智能交通灯算法，相较传统算法，显著提高了缓解道路拥堵的效率。

1.3 应用价值及推广前景

(1) 聚焦国内城市

城市是一个国家人民的主要聚集地。但目前，我国多地仍旧频繁出现部分时

间段市民交通出行困难的现象。以 2019 年疫情前的重庆市为例，根据百度地图提供的《2019 年度中国城市交通报告》[1]显示，重庆城市区域间拥堵不平衡系数相较于 2018 年度有所增大。这一方面说明了峰拥堵指数高、高峰车速波动系数大以及常发性严重拥堵路段里程占比高等交通形势依旧严峻、人们依旧处于出行交通拥堵的困境。另一方面，这也在一定程度上反映了交通问题对于市民生活和政府交通行政管理方面造成了诸多不便，对城市经济增长产生了相当大的阻碍。

利用新型技术提出新的信号灯配时优化方案，可以解决城市交通拥堵问题，提高城市的运作效率，方便人们的生活。

(2) 关注乡村发展

伴随着乡村振兴的大力开展，国民经济的不断提高，国家的城市化水平会越来越高。部分农村已经开始大规模的改建，农村乡镇也开始修建省道国道等核心交通道路，由此交通问题会变得越来越严重。

规划农村交通建设，规避当前城市发展中的许多问题，助力未来的城市化建设进程。

(3) 着眼未来智能城市与交通

未来的智能城市，将充分利用物联网、云计算、人工智能等先进信息技术手段，全面感知、分析、整合、共享城市运行中的各项关键信息，使得城市中各个功能彼此协调运作，实现资源优化配置、城市高效管理以及生活品质提高。

智慧交通作为未来智能城市的重要部分之一将发挥巨大的作用。本项目着眼于利用人工智能技术优化交通灯配时，为未来智能城市的建设提供方案。

第 2 章 问题分析

2.1 问题来源

传统交通灯通常具有时间调度不灵活的缺陷。在交通工具数量趋增的当下，其无法根据车流量灵活控制交通灯变换时长，导致交通在道路拥堵时难以快速恢复畅通，增大了城市交通压力。时长规划不合理的传统交通灯模式不仅严重影响到人们的日常出行，还会一系列的生态问题，包括温室效应、大气污染和交通噪声等。

国家层面对此十分重视，并大力提倡加强和改进道路交通管理工作以切实解决道路交通拥堵问题。而基于人工智能技术的交通灯智能控制系统相比于传统交通灯，能够更加合理地进行交通灯时长控制，以帮助道路交通管理工作。

2.2 现有解决方案

2.2.1 基于 PLC 的智能交通系统设计[2]

孙娟、陈韶钰和张洪森曾提出通过基于 PLC 的智能交通系统的设计，实现根据交通情况对交通信号灯的按需设计，克服以往人工指挥交通费时费力的困难,提高可靠性和效率。通过顺序功能图设计的交通信号灯的方法，结合人流和车流的情况调节交通信号灯的亮灭时间，从而使交通信号控制系统得到了优化，可以有效缓解交通拥堵,减少人流和车流的滞留, 优化交通利用率。

但是，设计中使用的 PLC 控制器存在体系结构封闭，各 PLC 厂家的硬件体系互不兼容，编程语言及指令系统也各异的问题。当用户选择了一种 PLC 产品后，必须选择与其相应的控制流程，并且学习特定的编程语言，灵活性上较差。

2.2.2 基于深度强化学习的交通信号自适应控制研究[3]

为了缓解交通拥堵,学者丁文杰对交通信号控制问题进行了优化,并采用深度强化学习算法对路口进行控制。通过 SUMO 仿真软件与 Python 语言对接的接口,并且通过这个接口调用各个功能模块，采用 DDPG 算法来控制交通路口。针对 DDPG 算法训练效率低,收敛速度慢,对奖励函数进行改进,得到改进的 DDPG 算法,在仿真软件上仿真并进行实验，具有一定效果。

然而，DDPG 不适用于随机环境的场景，难以应对剧烈变化的且具有不定性的交通大环境，由此研发的交通信号控制系统具有一定局限性。同时，DDPG 需要训练两个即策略网络和值网络，工程量较大。

2.2.3 基于 YOLO 算法的智能交通灯控制系统模型[4]

付振华、纪祥、赵坤旭等人设计了一种基于 YOLO 算法实现嵌入式系统控制交通信号灯及路灯的新型装置。该设备使用摄像头捕获视频,通过 OpenCV 对视频进行预处理,YOLOv3 算法作为识别算法和定位算法检测并判定图像中车辆的相对位置及数量,利用 STM32 单片机改变左转和直行的通行时间,从而实现智能控制。经实验仿真验证,该设备可提高当前道路的通行能力,缓解交通压力。

但其使用的 YOLOv3 网络的 anchor 机制中，需要设定的超参：尺度(scale)和长宽比(aspect ratio)是比较难设计的，需要较强的先验知识。且冗余框非常之多：一张图像内的目标毕竟是有限的，基于每个 anchor 设定大量 anchor box 会产生大量的 easy-sample，即完全不包含目标的背景框。这会造成正负样本严重不平衡问题。

2.3 本作品要解决的痛点问题

针对强化学习控制交通灯的方式无法适用于随机环境的场景，难以应对剧烈

变化的且具有不定性的交通大环境的局限性，本作品的交通灯控制设计思路是基于不同车道的车流量统计结果这一简单而又实用的方法。

本作品使用的车辆检测模型是基于当前最先进的目标检测模型之一YOLOv5，该模型相对其他目标检测模型有较大的性能提升。

为了适用于交通场景下的车辆检测，本作品训练和使用的模型满足实时性要求，更利于实际使用。

2.4 解决问题的思路

2.4.1 功能需求

(1) 对输入系统的图片进行车道检测

(2) 实时呈现各车道车流量信息

(3) 根据车流量进行交通灯时间灵活分配

2.4.2 性能需求

(1) 车道线检测模型预测正确的像素点具有较高的准确率

(2) 目标检测模型满足实时性

(3) 最大限度压缩目标检测模型模型大小

(4) 最大限度减小目标检测模型计算量

(5) 智能算法的动态绿灯时间小于传统算法的累计停车时间

2.4.3 数据集

2.4.3.1 车道线检测模型部分数据集

(1) Tusimple 数据集

LaneNet 网络的预训练模型采用的数据集是 Tusimple 这类车载端角度拍摄的数据集，如图 1 所示

图 1 Tusimple 数据集

(2) 自制数据集

本项目对自行采集的车道上方视角的数据进行数据标注，使用自制数据集对预训练模型进行微调，自制部分数据集展示如图 2 所示。

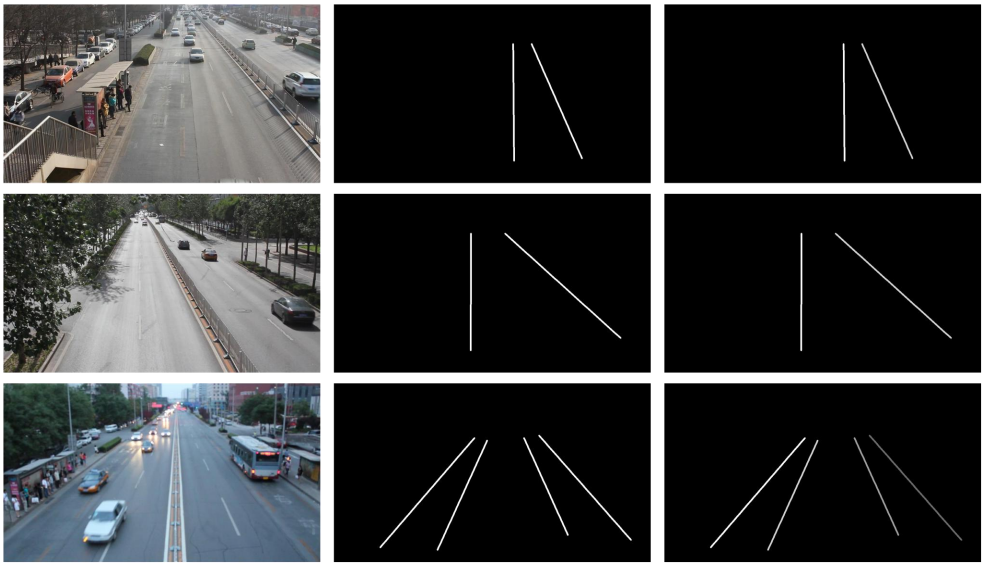
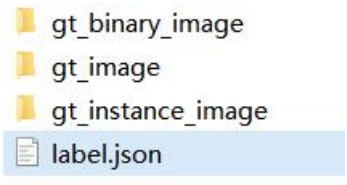


图 2 自制数据集

其中，左边图像是原图，中间图像是语义分割图像，右边图像是实例分割图像。

① 数据格式

Tusimple 数据集和自制数据集：



其中 gt_binary_image 是语义分割图像，gt_image 是原图，gt_instance_image 是实例分割图像，gt_binary_image 和 gt_instance_image 是 png 格式，大小 512×256，gt_image 是 jpg 格式，大小 960×540。label.json 中是每张图片每条车道线的像素点坐标集合。

② 数据来源

Tusimple 数据集：https://github.com/TuSimple/tusimple-benchmark/issues/3

自制数据集：网上各大视频平台的相关交通视频

③ 数据获取方式

Tusimple 数据集:https://github.com/TuSimple/tusimple-benchmark/issues/3

自制数据集：附素材源码

④ 数据特点

Tusimple 数据集:车载端角度拍摄，适用于自动驾驶领域

自制数据集：车道上方的俯瞰视角

⑤ 数据规模

Tusimple 数据集:

- ==训练集==
- 3626 个视频剪辑，每个视频剪辑为 1s 拍摄的图片，共计 20 张
- 对于每个剪辑：3626 个注释帧
- ==测试集==
- 2944 个视频剪辑
- ==每张图片==：1280 x 720
- 总计：3626 张训练图像和 2782 张测试图像

自制数据集：原图、语义分割图像、实例分割图每类图像共 1159 张。

2.4.3.2 车辆目标检测模型部分数据集

(1) UA-DETRAC 数据集

车辆检测和跟踪的大规模数据集 UA-DETRAC 如图 3 所示。

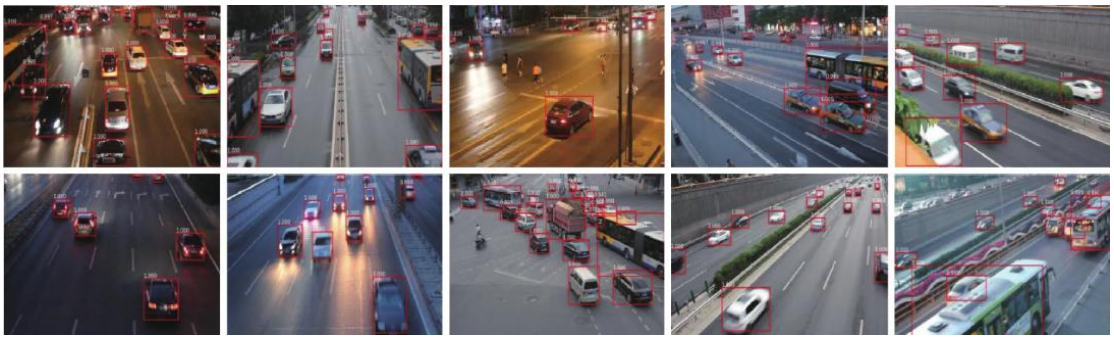


图 3 UA-DETRAC 数据集

① 数据格式

```
UA-DETRAC
├── Annotations
│   ├── DETRAC-Test-Annotations-XML.zip
│   └── DETRAC-Train-Annotations-XML.zip
```

```
└── Images
    ├── DETRAC-test-data.zip(3.94GB, 40 sequences)
    └── DETRAC-train-data.zip(5.22GB, 60 sequences)
```

② 数据来源

UA-DETRAC 数据集：https://detrac-db.rit.albany.edu/

③ 数据获取方式

UA-DETRAC 数据集：https://detrac-db.rit.albany.edu/

④ 数据特点

车辆分为四类，即轿车、公共汽车、厢式货车和其他车辆。天气情况分为四类，即多云、夜间、晴天和雨天。

⑤ 数据规模

8250 个车辆和 121 万目标对象外框。

第 3 章 技术方案

3.1 依赖技术

3.1.1 YOLOv5 目标检测模型

YOLO（you only look once）系列是一种基于深度神经网络的对象识别和定位算法，其最大的特点是运行速度很快，是当前计算机视觉目标检测领域的最受欢迎的目标检测算法之一。YOLOv5 是 YOLO 系列的第五个版本。

在该项目中，使用了修改后的 YOLOv5 进行对车辆的目标检测。在保留 YOLOv5 神经网络主要框架的基础上，将其主干网络更改为 MobileNetV2 进行特征提取，构建 YOLOv5s-MobileNetV2 网络，进一步降低网络复杂度，提高模型的推理速度，以达到实时检测的效果。

3.1.2 LaneNet 检车道线

LaneNet 车道线检测模型是一种端到端的车道线检测算法，其将实例分割任务拆解成语义分割和聚类两部分。LaneNet 有两个分支任务，其中一个负责对输入图像进行语义分割，即对像素进行二分类，判断像素属于车道线还是背景，另一个分支对像素进行嵌入式表示，训练得到的嵌入向量用于聚类。最后将两个分支的结果进行结合利用 Mean-Shift 算法进行聚类，得到实例分割的结果。

本项目利用自制的俯瞰车道线数据集训练LaneNet网络进行车道线的检测与提取。

3.1.3 SORT目标跟踪算法

SORT算法是一套简单的目标跟踪算法，其核心是卡尔曼滤波算法和匈牙利算法。其中卡尔曼滤波算法的作用是当前的一系列运动变量去预测下一时刻的运动变量。匈牙利算法的作用是把一群检测框和卡尔曼预测的框做分配，让卡尔曼预测的框找到和自己最匹配的检测框，达到追踪的效果。

本项目对车辆的跟踪采用的SORT算法，相比其他目标跟踪算法，能够降低跟踪模型的计算量，减少CPU占用率，节省计算资源，提高跟踪的实时性。

3.1.4 PyQt GUI应用程序创建工具包

PyQt是Python编程语言和Qt库的成功融合，而Qt库是最强大的库之一，由Phil Thompson 开发。其可用的类有很多，主要分成几个模块。以及用于创造经典桌面风格的用户界面的“QtWidgets”等。

本项目主要使用了QtGui来实现类窗口系统集成、事件处理、二维图形、基本成像、字体和文本的设置，同时还使用QtWidgets模块，创建桌面风格的用户界面。通过传入简单的待检测的交通道路图片与视频，利用内部函数调用启动模型，最后将处理后的图片、视频呈现在用户界面。该可视化界面对于用户友好，使其可以迅速掌握并使用该系统。

3.1.5 SUMO

SUMO（Simulation of Urban Mobility）是免费、开源的交通系统仿真软件，可以实现交通流的微观控制，即具体到道路上每一辆车的运行路线都可以单独规划。 SUMO最早发布于 2001年，主要由 German Aerospace Center 下属的Institute of Transportation Systems 的研究人员开发。SUMO允许对包括道路车辆、公共交通和行人在内的多式联运交通系统进行建模。SUMO包含了大量的支持工具，这些工具可以自动完成交通模拟的创建、执行和评估等核心任务，比如网络导入、路由计算、可视化和排放计算。可以使用自定义模型增强SUMO，并提供各种api来远程控制模拟。

3.2 技术路线

项目组针对交通道路拥堵以及一系列衍生的现实问题，设计出了一套解决该

类问题的流程和方法。其中，项目组主要针对实现交通灯的实时智能调控功能展开研究。项目组首先对交通灯算法展开了设计，以“最大化节省等待时间，最小化拥堵时间”为宗旨，自主设计了智能交通灯算法。智能交通灯算法所需的输入数据为各向车道车流量，为此，项目组以“自动化”、“实时性”为目标，设计了能够智能检测各车道车流量的系统，从而获得所需的数据信息。

3.3 总体框架

基于人工智能技术的智能交通灯控制系统包括车流量的检测，交通灯时间计算两个模块，两个模块的实现如图 4 和图 5 所示。

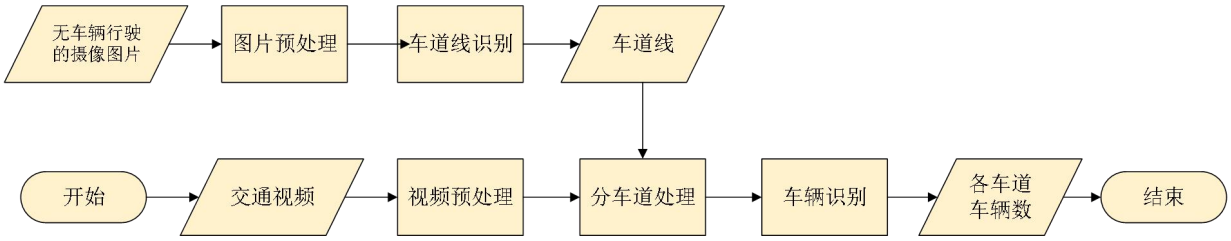


图 4 车流量检测

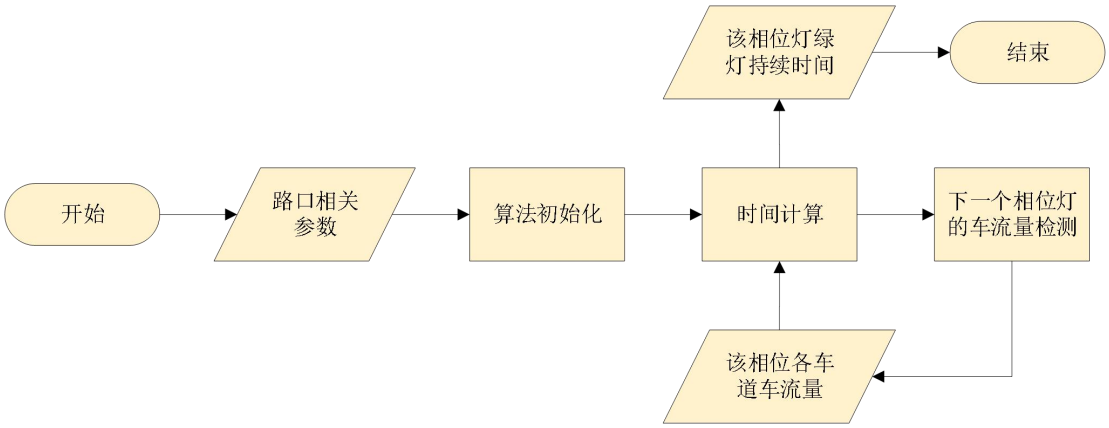


图 5 交通灯时间计算

3.4 各模块实现

3.4.1 车流量识别功能模块

3.4.1.1 车道线检测功能

（1）输入形式的选择

图 6 输入图像

完成对输入视频和图片两种情况的测试后，最终确定采用检测图片的方式。这是由于俯瞰摄像头所拍摄的视频具有不稳定的特性，复杂的交通环境和路况导致检测准确率较低。来往的车辆在镜头前经过时，系统检测出不需要的车辆边缘线条，甚至会出现车辆遮挡车道线的情况，导致误测的可能性增加。又鉴于俯瞰摄像头本身固定位置的特点，针对每个俯瞰摄像头选取视频中的一张在白天时间段的车流量相对较少的图像进行车道线检测，提高车道线检测准确率，检测后的车道线数据对于该摄像头在一般情况下可永久使用。

（2）车道线检测模型

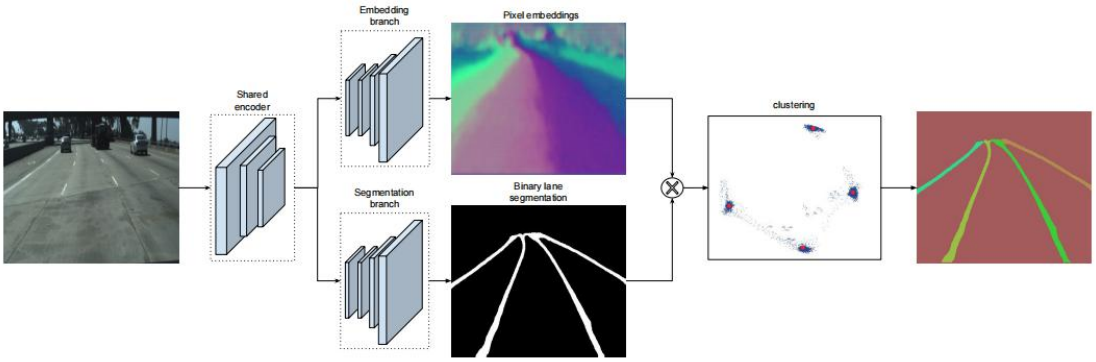


图 7 车道线检测模型

车道线检测模型采用 LaneNet 网络[5]实现。LaneNet 将车道线的实例分割拆解为语义分割和聚类两部分。如图 7 所示，LaneNet 中解码器分为两个分支，Segmentation branch 负责对输入图像进行语义分割（对像素进行二分类，判断像素属于车道线还是背景），Embedding branch 对像素进行嵌入式表示，训练得到的 embedding 向量用于聚类。最后将两个分支的结果进行结合利用 Mean-Shift 算法进行聚类，得到实例分割的结果。

（3）数据集设计

LaneNet 网络的预训练模型采用的数据集包括 Tusimple 等车载端角度拍摄的数据集，但由于 Tusimple 数据集是在车载端拍摄的数据集，检测的车道线适用于自动驾驶领域，而本项目拍摄车道的角度是在车道上方的俯瞰视角，使用原模型检测车道线不具有良好的泛化能力。因此，本项目使用自行采集的车道上方视角的进行数据标注后的数据集（自制数据集如上文图 2）。

使用此自制数据集在预训练模型的基础上进行训练，对模型参数进行微调，从而实现车道线检测。

（4）绘制检测出的车道线

将预测出的车道线所属像素点绘制出来，输出图像如图 8。

图 8 最终呈现效果

3.4.1.2 车辆识别功能

（1）车辆数据集的选用以及处理

车辆目标检测模型的训练采用 UA-DETRAC 数据集，其主要拍摄于北京和天津的道路过街天桥（京津冀场景），并手动标注了 8250 个车辆 和 121 万目标对象外框。

UA-DETRAC 的车辆是俯瞰视角下的车辆，符合本项目对车辆角度的需求。其天气情况分为四类，即多云、夜间、晴天和雨天，可加强模型在复杂环境下的检查效果。

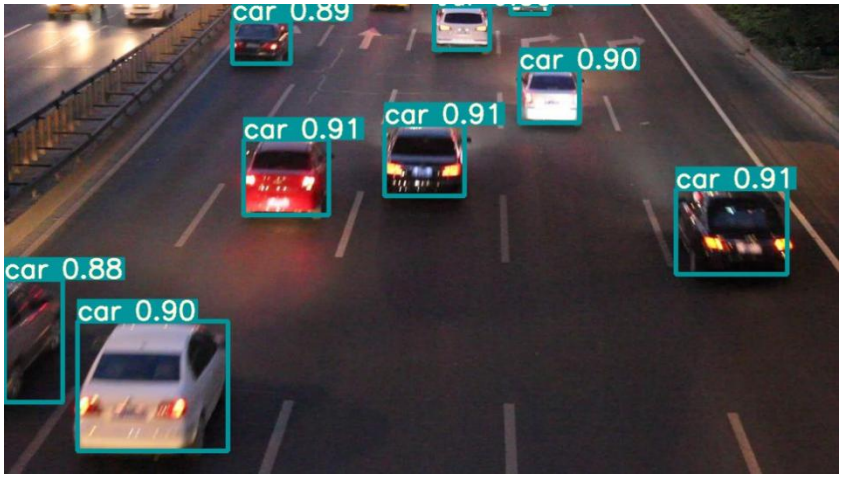


图 9 夜间检测效果

对数据集进行数据清洗，剔除目标对象框标注出现偏移的数据。采取跳帧提取图像的方法防止过拟合，将车辆标签全部改为 car，防止样本标签不均衡。并对训练和测试数据集进行 8：2 的划分，最终的训练和测试图片比例为 55257：15322。

（2）车辆目标检测模型

针对现有的目标检测模型参数多，网络复杂大，计算量大、不利于在嵌入式等设备进行部署，无法满足实时性等问题，因此本项目改进 YOLOv5 系列的 YOLOv5s，将原始 YOLOv5s 主干网络更改为 MobileNetV2[6]进行特征提取，构建 YOLOv5s-MobileNetV2 网络。MobileNet 系列网络是轻量级网络的代表，被广泛应用于移动终端中，其特点是引入了深度可分离卷积，当卷积核大小为 3 × 3 时，随着卷积核个数不断增加，深度可分离卷积计算量最多可缩小为标准卷积的 1/9，同时也能保持较好的检测精度。

MobileNetV2 在采用深度可分离卷积基础上，使用线性瓶颈的反向残差模块（Inverted Residual）提高特征提取能力。图 10 所示，卷积分为步长 1、2 两种模式，其中，Conv 1 × 1 表示使用 1 × 1 的卷积核进行卷积，Dwise 3 × 3 表示使用 3 × 3 的卷积核进行深度可分离卷积，Linear 表示线性激活，Stride = 2 表示卷积步长为 2。当卷积步长为 1 时，输入经过跳跃相加至模块输出，卷积步长为 2 时模块输入不经过相加而输出。反向残差模块与残差结构不同的是：首先经过 1 × 1 卷积操作，对通道升维，缓解 Relu6 激活函数导致的信息丢失，再由 Relu6 激活函数稀疏模型，最后经过 1 × 1 卷积以及线性激活函数降低为输入通道数。

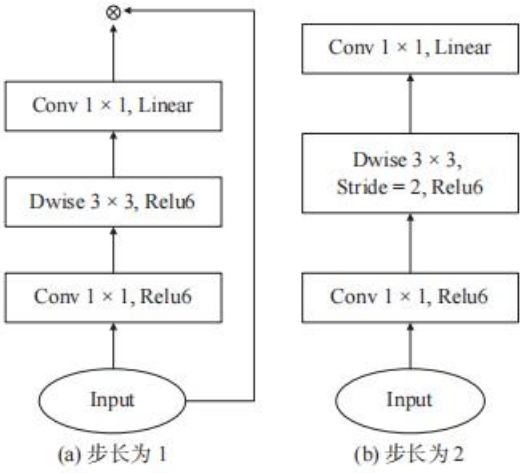


图 10 反向残差模块结构

修改后模型的主干网络的结构如表 1 所示。

表 1 YOLOv5s-MobileNetV2 主干网络

序号	输入	数量	参数	模块	张量信息
0	-1	1	928	Conv	[3, 32, 3, 2]
1	-1	1	896	InvertedResidual	[32, 16, 1, 1]
2	-1	1	5136	InvertedResidual	[16, 24, 2, 6]
3	-1	1	8832	InvertedResidual	[24, 24, 1, 6]
4	-1	1	10000	InvertedResidual	[24, 32, 2, 6]
5	-1	1	14848	InvertedResidual	[32, 32, 1, 6]
6	-1	1	14848	InvertedResidual	[32, 32, 1, 6]
7	-1	1	21056	InvertedResidual	[32, 64, 2, 6]
8	-1	1	54272	InvertedResidual	[64, 64, 1, 6]
9	-1	1	54272	InvertedResidual	[64, 64, 1, 6]
10	-1	1	54272	InvertedResidual	[64, 64, 1, 6]
11	-1	1	66624	InvertedResidual	[64, 96, 1, 6]
12	-1	1	118272	InvertedResidual	[96, 96, 1, 6]
13	-1	1	118272	InvertedResidual	[96, 96, 1, 6]
14	-1	1	155264	InvertedResidual	[96, 160, 2, 6]
15	-1	1	320000	InvertedResidual	[160,160,1, 6]
16	-1	1	320000	InvertedResidual	[160,160,1, 6]
17	-1	1	473920	InvertedResidual	[160,320,1, 6]

修改后的模型参数对比如表 2 所示。

表 2 网络模型对比

网络模型	层数	参数	计算量
YOLOv5s	283	7063542	16.4
YOLOv5s-MobileNetv2	325	4543798	9.8

模型的参数减少 35.7%， 计算量减少 40.2%。

(3)车辆跟踪算法

在十字路口前的车道场景下，车辆受到遮挡而导致跟踪失败的情况较少，因此采用简单的 SORT 目标跟踪算法[7]，降低跟踪模型的计算量，减少 CPU 占用率，节省计算资源，提高跟踪的实时性。SORT 的具体工作流程如图 11：

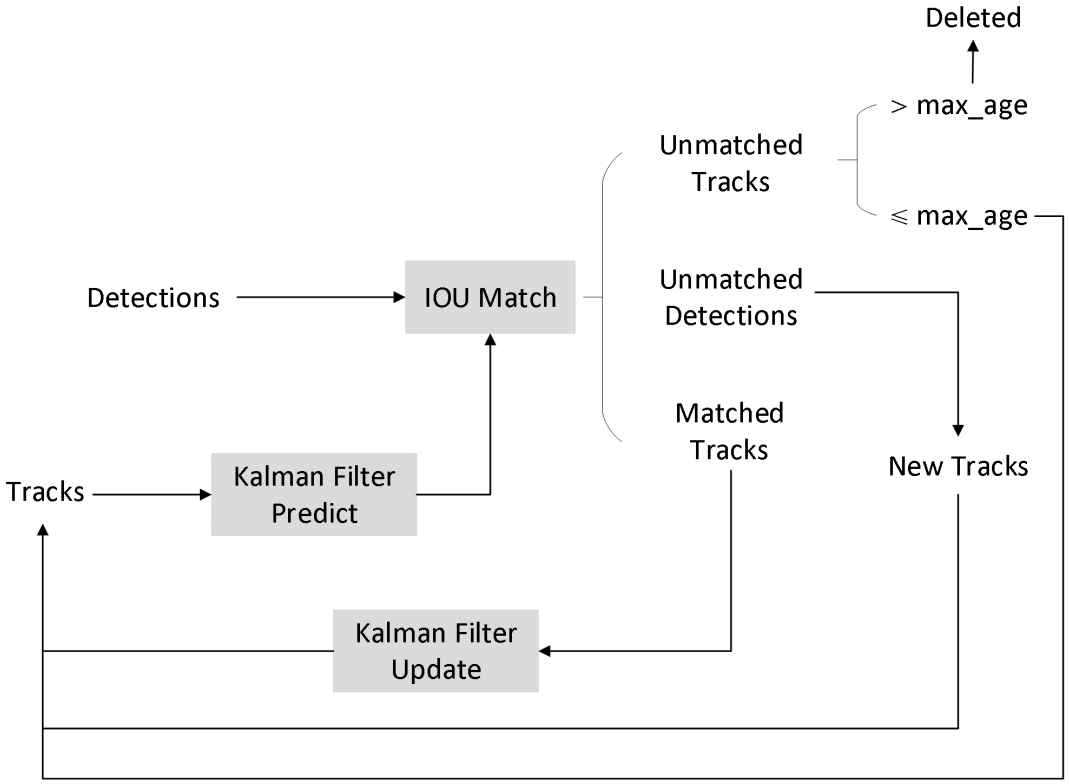


图 11 SORT 目标跟踪算法流程

其中，Detections 是通过 YOLOv5s-MobileNetv2 模型检测到的目标框，Tracks 是使用卡尔曼滤波形成的轨迹信息。整个算法的工作流程如下：

1）将第一帧检测到的结果创建其对应的 Tracks。将卡尔曼滤波的运动变量初始化，通过卡尔曼滤波预测其对应的框框。

2）将该帧目标检测得到的目标检测框和上一帧通过轨迹 Tracks 预测的框一一进行 IOU 匹配，再通过 IOU 匹配的结果计算其代价矩阵。

3）将 2）中得到的所有的代价矩阵作为匈牙利算法的输入，得到线性的匹配的结果，从而得到三种结果。第一种是轨迹 Tracks 失配（Unmatched Tracks），当该轨迹失配的次数大于设定的 max_age 时，我们直接将失配的 Tracks 删除；第二种是该帧的检测框 Detections 失配（Unmatched Detections），我们将这样的 Detections 初始化为一个新的 Tracks（new Tracks）；第三种是该帧的检测框和预测的框配对成功，这说明我们前一帧和后一帧追踪成功，将其对应的检测框 Detections 通过卡尔曼滤波更新其对应的轨迹 Tracks 变量。

4）反复循环 2）-3）步骤，直到视频帧结束。

3.4.1.3 分车道车辆统计功能

进行各车道区域坐标的计算与基于虚拟线圈的车辆统计。基于虚拟线圈的车流量统计方法与交通道路上传统的物理线圈类似，但由于物理线圈需要埋设在路面之下，产生安装维护费用高，路面破坏等问题。而采用基于视频的虚拟线圈的车辆计数方法可避免以上问题。虚拟线圈车辆计数法的原理是在采集到的交通流视频中，在需要进行车辆计数的道路或路段上设置一条或一条以上的检测线对通过车辆进行检测，从而完成计数工作。检测线的设置原则一般是在检测车道上设置一条垂直于车道线的虚拟线段，通过判断其与通过车辆的相对位置的变化，完成车流量统计的工作。如图 12 和 13 所示，黄色的线就是虚拟检测线。

判断车辆是否穿过虚拟检测线的方法如图 12 所示。P_1和 P_0分别为前一帧和当前帧画面的车辆检测框的中心点。A 和 B 分别为虚拟检测线的两个端点。当向量$\overrightarrow{AP_1}$ 和 向量$\overrightarrow{AB}$的叉乘的结果与向量$\overrightarrow{AP_0}$ 和 向量 $\overrightarrow{AB}$ 的叉乘的结果的符号为一正一负时，即可判断车辆穿过虚拟检测线。

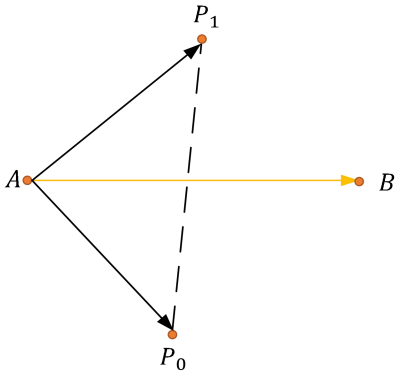


图 12 车辆撞线示意图

最终的计数方法为：

循环遍历每一个车道线检测点的横坐标：

当检测到车辆穿过虚拟检测线时，

1）当车辆中心点的横坐标小于车道线横坐标时，车道线横坐标对应的车道的车辆数目累加 1

2）当车辆中心点的横坐标大于车道线横坐标且循环已遍历到最后一条车道线时，最右边车道的车辆数目累加 1

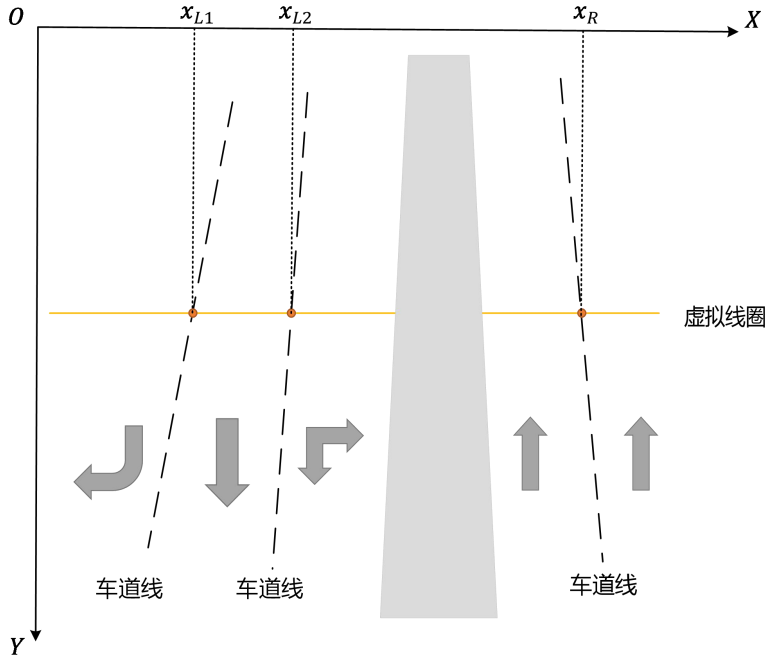


图 13 分车道车流量统计示意图

3.4.2 交通灯控制模块

3.4.2.1 交通灯控制算法

以相位灯相关车道车流量、人行道通行时间、最大绿灯时间等作为输入，对相应车道的交通灯切换时间进行计算。

针对每次相位灯变换之前的输入，调度算法算出合适的绿灯时长。选取相位灯相关车道中车流量最大的车道作为实际车流量，先将该实际车流量与路口可通行最大车流量进行比较。若实际车流量大于可通行最大车流量，则将绿灯时长设置为设定好的最大时间值。若实际车流量小于最大车流量，则将实际车流量与最小车流量进行比较。若实际车流量小于最小车流量，为了保证行人能够有足够的通行时间，则将绿灯时长设置为设定好的最小时间值；否则，按照“输出时间=车辆数*平均通过时间+车辆数*延误时间”对绿灯持续时长进行计算。

（1）模块基本说明

交通灯控制模块主要是基于交通灯时间的计算，即每个相位灯绿灯的持续时间的计算。如图 14 所示，首先是需要输入一系列的路口参数，其中包括了人行道时间，最大绿灯时间等等，详细参数表如表 3 所示。

循环开始，传入 Xi 相位灯相关车道的车辆数，相关车道比如 X1 相位灯的相关车道是 Y1 相位的左转车道,Y2 相位的直行车道。比较选出最多车辆的车道的车辆数，比较该车辆数是否达到路口通行最大车辆数，如果达到，则输出最大时间值，如果没有达到，则判断是否小于最小车辆数，如果小于，则直接输出最小时间值，保证行人的通过时间，否则则输出计算过后的时间值。输出的时间值为该相位灯的绿灯持续时间。i++，循环切换到下一个需要变换绿灯的相位灯。

每次调用该算法的时间设定在 Xi 相位灯即将转换成绿灯前的那个时刻。

（2）模块流程图和路口参数表

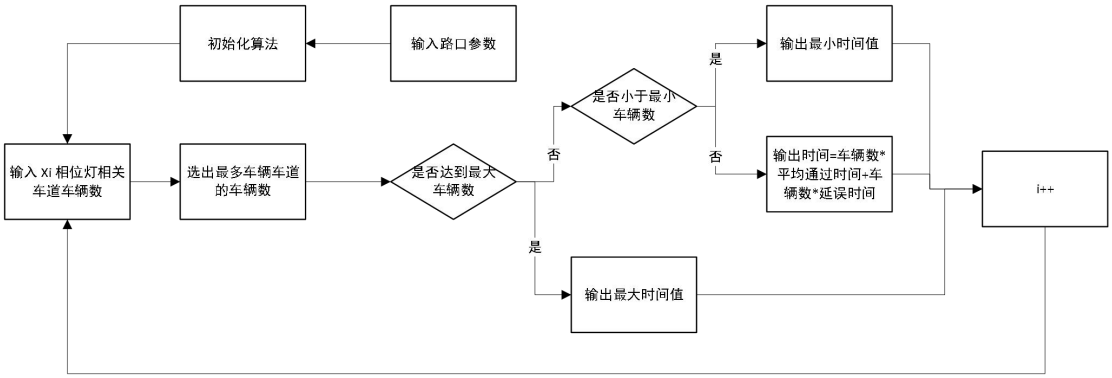


图 14 交通灯时间计算

表 3 路口参数表

参数名	参数等级	参数数据单位	参数简单说明
相位数量	1	个	一共有多少个相位灯
第 i 组相位最小通行时间	2	秒	人行横道通行时间
第 i 组相位最大通行时间	2	秒	最大绿灯时间
第 i 组相位关联信号灯数量	2	个	一个相位灯可能牵扯到两个甚至更多方向的信号灯。
第 i 组第 y 个信号灯序号和指示方向	3	Int，string	编号和方向，一般方向为“左转”和“直行”

第 i 组第 y 个信号灯所指示的车道数	3	个	这个信号灯所关注的车道数量
第 i 组第 y 个信号灯平均车辆通过速度	3	车/秒	平均一秒能通过多少部车
第 i 组第 y 个信号灯车辆的平均延误时间	3	秒	转换信号灯时车辆的起步损耗的时间以及变换成黄灯的时候所损耗的时间

第 4 章 系统实现

4.1 用户界面

用户界面使用 pyqt5 编写，界面内提供四个按钮。依次点击“打开图片”、“检测车道”、“打开视频”、“开始运行”，输入待检测的车道图片与视频，可以获得检测出来的车道线以及实时车流量信息。展示如下图 15。

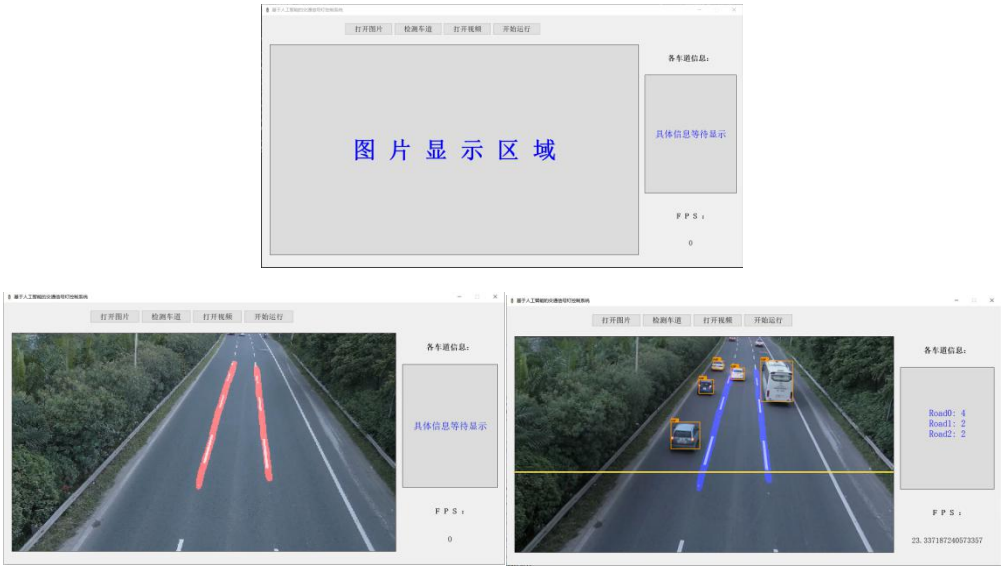


图 15 用户操作界面

其中，输入的图片与视频来源于交通道路上的监控摄像头。

4.2 运行环境

操作系统：　　Windows 10 64 位
Python 运行环境：　Python3.8
深度学习模型：　　Pytorch 1.9.0
CPU：　　Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz
GPU：　　NVIDIA GeForce GTX 1650 Ti 4G
CUDA 版本：　　10.2

4.3 系统部署

下载项目源码文件（已附作品压缩上传），打开 counting.py 文件，并运行，即可顺利进入该系统界面。项目文件目录如下图 16。

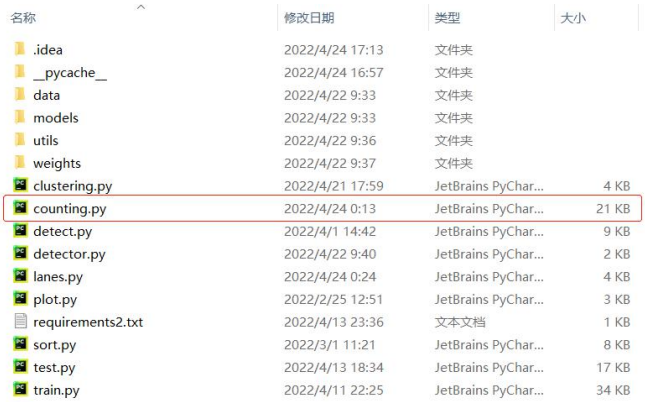


图 16 素材源码目录

第 5 章 测试分析

5.1 项目评测模型

5.1.1 车流量识别评测模型

（1）车道线检测模型性能分析

将数据集按 8：2 进行划分，在添加预训练模型的基础上，模型的训练损失值快速收敛，如图 17 所示，

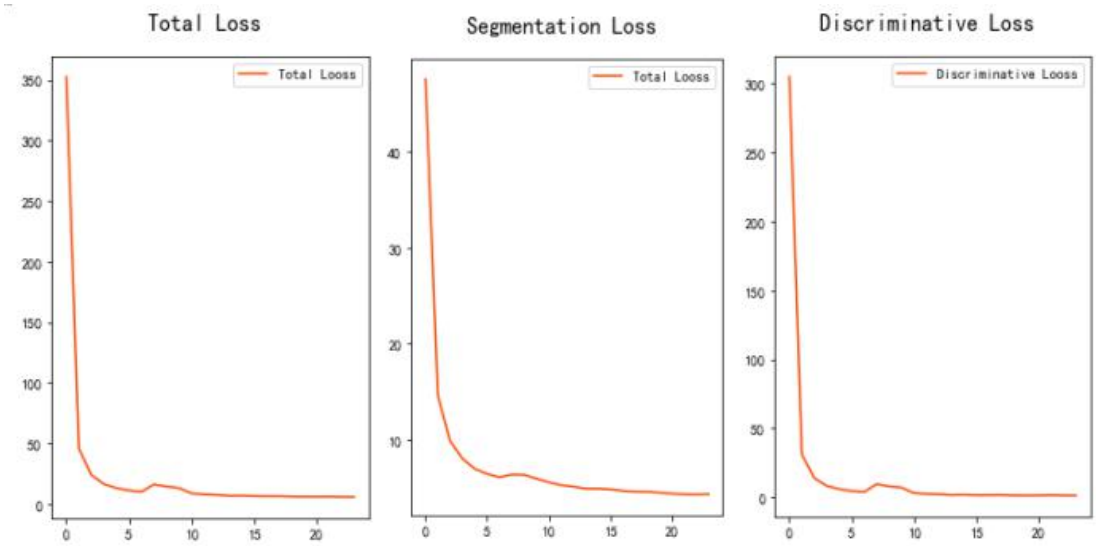


图 17 车道线检测模型训练 Loss 值

验证结果表明，模型预测正确的像素点的准确率可达 98.36%。

（2）目标检测模型性能分析

对 YOLOv5s-MobileNetv2 模型的训练设置 30 轮，训练过程中保存最佳模型权重，训练结果如图 18 所示。

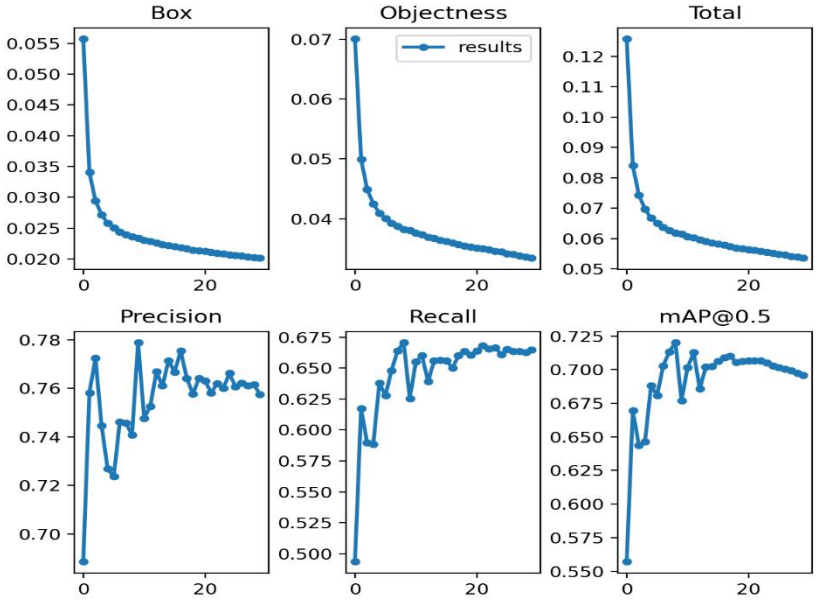


图 18 模型训练结果

设置 YOLOv5s 的模型作为纵向对比模型，使用相同的数据集设置相同的训练轮数进行训练，得到的精确率、召回率和 mAP@0.5、模型大小和 FPS 指标的对比如表 4 所示。

表 4 模型纵向对比

网络模型	精确率	召回率	mAP@0.5/%	模型大小(M)	FPS (GeForce GTX 1650 Ti, 4G)
YOLOv5s	78.8%	68.9%	74.3%	81.3	22.71
YOLOv5s-MobileNetv2	76.3%	66.1%	71.0%	52.6	25.79

结果显示，虽然 mAP@0.5 下降了约 3 个点，但模型压缩了 35.3%，计算量减少 40.2%，在 GeForce GTX 1650 Ti 的平台上，FPS 上升了 13.56%，满足实时性要求，更易于嵌入边缘设备。

5.1.2 交通灯控制算法评测模型

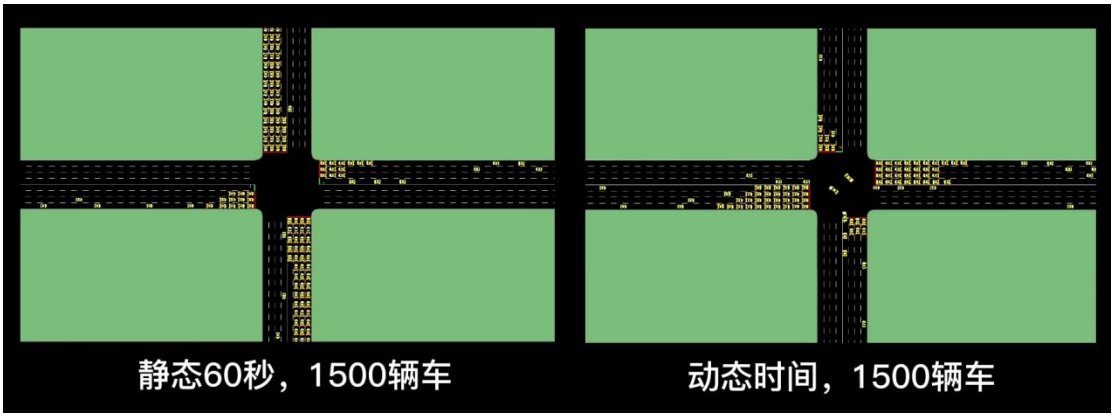


图 19 交通灯算法评测模型

为了清晰地对比传统交通灯时间分配算法和本项目所采用的交通灯时间分配算法，本项目通过使用 SUMO（Simulation of Urban Mobility）交通系统仿真软件构建了一个十字交叉路口来演示传统和智能交通灯两套算法的实现效果。传统交通灯算法采用每个相位灯持续 60 秒的标准，而智能交通灯算法是根据实时的车流量来确定绿灯持续时长。可以清晰地看到，两套算法在各道路车辆数目相同、模拟时间相同的环境下进行仿真模拟，智能算法在缓解交通拥堵方面明显优于传统算法。

在仿真模拟中通过统计道路上所有车辆的停车时间总和来对两套算法进行数据上的比较，可以看出在车辆数目相同的情况下传统算法的累计停车时间远远大于智能算法的动态绿灯时间。

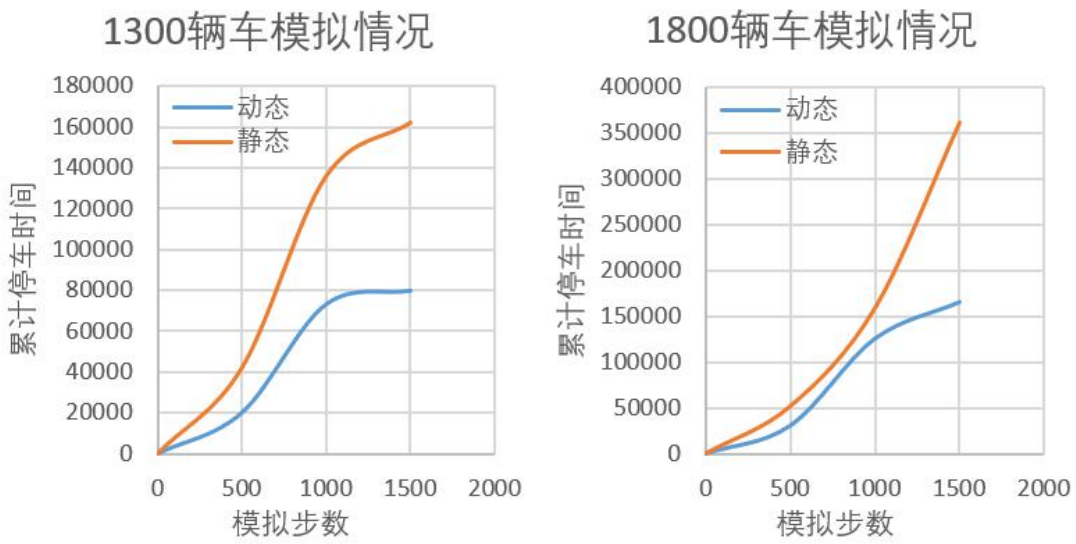


图 20 不同车辆数模拟情况对比

除此之外，本项目采用的智能算法提供两种方案。其中，方案一采用每个周期进行一次时间分配的方法，方案二采用每个相位灯变换之前进行一次时间分配的方法。对比多次随机车流量通过路口所耗费的平均时间，得出方案一和方案二都优于传统算法，且方案二优于方案一的结论。本项目采用方案二。

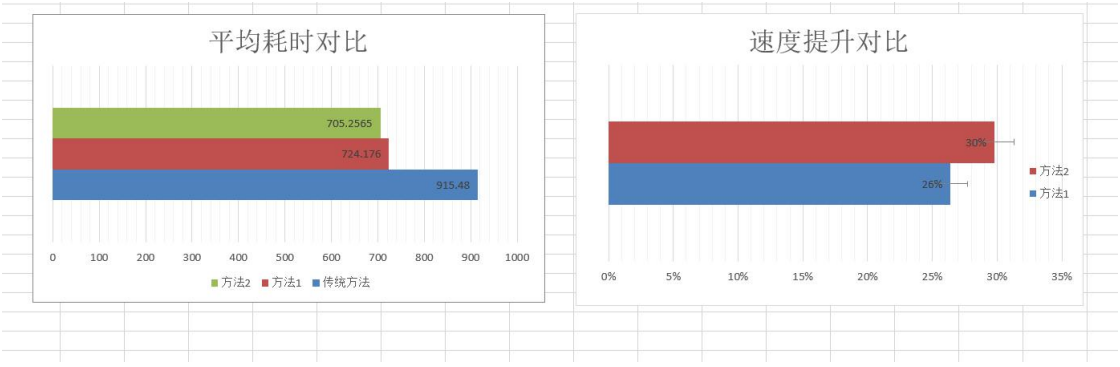


图 21 算法耗时对比和速度提升对比

第 6 章 作品总结

6.1 作品特色与创新点

能够对对输入系统的图片进行车道检测，实时呈现各个车道的车流量信息，并根据车流量进行交通灯时间灵活分配，使得智能算法的动态绿灯时间小于传统算法的累计停车时间，一定程度上有效缓解了道路交通拥堵问题。

本项目在性能上能够实现在车道线检测模型具有较高的准确率的条件下最大限度压缩目标检测模型模型大小，减小模型计算量，使其更易于嵌入边缘设备。

整个项目各模块内部内聚度高，相互之间耦合度低，便于开发、维护与修改。

6.2 应用推广

6.2.1 助力智能交通系统

中国的智能交通系统发展迅速，在北京、上海、广州等大城市已经建设了先进的智能交通系统;其中，北京建立了道路交通控制、公共交通指挥与调度、高速公路管理和紧急事件管理的 4 大 ITS 系统;广州建立了交通信息共用主平台、物流信息平台和静态交通管理系统的 3 大 ITS 系统。

本项目研究的基于人工智能的交通信号灯控制系统可参与 ITS 系统的构成，且基于系统本身具有的模型小、计算量小以及满足实时性的优点，其在现实背景下具有一定的指挥、调度与控制道路交通的潜能。

6.2.2 提供先进的交通信息服务

交通系统中车辆、乘客和行人元素不断进行着复杂的互动，使得高效的交通控制成为一个极具挑战性的问题。

本项目的系统在具有较高检测技术以收集精准的交通系统特性数据的同时，对边缘设备友好。其可结合控制优化和预测等算法等生成更高效安全的交通控制管理方案以及提供先进的交通信息服务，突破视频这一交通检测手段的局限性，加强在交通信号灯控制系统和交通调度控制系统中的应用。

系统功能可拓展至交通信号灯协调控制系统、高速公路管理优化系统和智能公共

交通调度系统，帮助人们解决出行的交通问题，提供更优质的服务。

6.3 作品展望

本作品完成了分车道车流量统计和交通灯算法三项主要功能，作品仍有以下几点不足：

1. 车辆检测模型中未增加针对小目标的研究。
2. 车辆检测模型的精度有待提高。
3. 交通灯算法设计目前只考虑了车流量。

参考文献

[1]百度地图.2019 年度中国城市交通报告.[DB/OL]. (2020-03-17) [2022-04-15].http://renqi.baidu.com/reports/landing?id=50.

[2]孙娟,陈韶钰,张洪森.基于 PLC 的智能交通系统设计的研究[J].内蒙古科技与经济,2021(03):100-101+107.

[3]丁文杰. 基于深度强化学习的交通信号自适应控制研究[D].天津职业技术师范大学,2020.DOI:10.27711/d.cnki.gtjgc.2020.000243.

[4]付振华,纪祥,赵坤旭.基于 YOLO 算法的智能交通灯控制系统模型[J].单片机与嵌入式系统应用,2019,19(09):12-13+17.

[5] Neven D , Brabandere B D , Georgoulis S , et al. Towards End-to-End Lane Detection: an Instance Segmentation Approach[J]. IEEE, 2018.

[6] Sandler M , Howard A , Zhu M , et al. MobileNetV2: Inverted Residuals and Linear Bottlenecks[J]. IEEE, 2018.

[7]A. Bewley, Z. Ge, L. Ott, F. Ramos and B. Upcroft, "Simple online and realtime tracking," 2016 IEEE International Conference on Image Processing (ICIP), 2016, pp. 3464-3468, doi: 10.1109/ICIP.2016.7533003.