AIGC检测·全文报告单

NO:CNKIAIGC2025FG_20250499029735　　检测时间：2025-04-21 18:23:04

篇名：基于YOLOv8的车辆检测与跟踪系统的设计与实现
作者：魏照轩
单位：
文件名：计算机学院本科毕业论文模板（30页以上正文，20篇以上参考文献，至少3篇外文参考文献）-2025

全文检测结果　知网AIGC检测　https://cx.cnki.net

AI特征值：9.0%
AI特征字符数：1490
总字符数：16523

AI特征显著（计入AI特征字符数）
AI特征疑似（未计入AI特征字符数）
未标识部分

AIGC片段分布图

前部20%
AI特征值：0.8%
AI特征字符数：27

中部60%
AI特征值：12.6%
AI特征字符数：1250

后部20%
AI特征值：6.4%
AI特征字符数：213

0　16523

AI特征显著　AI特征疑似　未标识部分

分段检测结果

序号	AI特征值	AI特征字符数 / 章节(部分)字符数	章节(部分)名称
1	9.0%	889 / 9925	基于YOLOv8的车辆检测与跟踪系统的设计与实现.doc_第1部分
2	9.1%	601 / 6598	基于YOLOv8的车辆检测与跟踪系统的设计与实现.doc_第2部分

1. 基于YOLOv8的车辆检测与跟踪系统的设计与实现.doc_第1部分

AI特征值：9.0%　　AI特征字符数 / 章节(部分)字符数：889 / 9925

片段指标列表

序号	片段名称	字符数	AI特征	
1	片段1	1085	疑似	10.9%
2	片段2	294	疑似	3.0%
3	片段3	236	疑似	2.4%
4	片段4	306	显著	3.1%
5	片段5	226	显著	2.3%
6	片段6	333	疑似	3.4%
7	片段7	357	显著	3.6%

原文内容

摘要：车辆的检测与跟踪系统是智能交通系统(ITS)的重要组成部分，其广泛应用于交通监控，自动驾驶，违章检测当中。针对目前复杂的交通监控环境下车辆的检测精度不足，速度缓慢的问题，提出了引进yolov8的车辆目标检测系统。

首先，介绍了yolov8的基本原理以及其在车辆检测环境中的应用，通过深度学习算法系统能够识别视频流中的车辆目标并且进行分类。其次，结合DeepSort多目标跟踪算法，系统实现了对于车辆的连续跟踪。实验结果表明，该系统在实际应用当中具有较高的鲁棒性和准确性，为智能交通跟踪和自动驾驶技术的发展提供了理论支持和技术参考。

关键词：YOLOV8；DeepSort；车辆检测；多目标跟踪

Abstract: Vehicle detection and tracking systems are crucial components of Intelligent Transportation Systems (ITS), widely applied in traffic monitoring, autonomous driving, and violation detection. To address the issues of insufficient detection accuracy and slow processing speeds in current complex traffic monitoring environments, this paper proposes a vehicle detection system that introduces YOLOv8.

Firstly, the fundamental principles of YOLOv8 and its application in vehicle detection environments are introduced. Through deep learning algorithms, the system can identify and classify vehicle targets in video streams. Secondly, by integrating the DeepSort multi-object tracking

10.9%(1085)

algorithm, the system achieves continuous vehicle tracking. Experimental results demonstrate that this system exhibits high robustness and accuracy in practical applications, providing theoretical support and technical reference for the development of intelligent transportation tracking and autonomous driving technologies.

Keywords: YOLOV8; DeepSort; Vehicle detection; Multi-object tracking

1. 绪论

随着城市化进程的加快和机动车持有量的快速增长，交通拥堵、交通事故等问题日益突出，对于城市的交通管理提出了更高的要求。智能交通系统(Intelligent Transportation System, ITS)作为解决以上问题的重要手段，它集成了先进的信息技术，通信技术和控制技术，能够有效的减少交通事故。在智能交通系统当中，车辆的检测与跟踪是核心组成部分，其被广泛的运用于交通流量监测，自动驾驶等领域[1]。

1.1 研究背景和意义

1.1.1 研究背景

近些年，随着计算机视觉技术的快速发展，目标检测算法和多目标跟踪算法在智能交通系统和自动驾驶领域得到了广泛应用。YOLO(You Only Look Once)系列算法作为目标检测领域的经典模型，它在精度和速度上有着显著的优势[2]。本文选用的YOLOv8作为较新一代的YOLO系列算法，相比于传统的目标检测，它在检测精度，速度和泛化能力上均有显著提升。

当下城市交通环境复杂且多变，大多时候会出现光照发生变化以及车辆之间相互重叠的情况，这些状况均会对车辆的检测造成严重影响，要是继续运用传统的目标检测算法，极易出现检测速度迟缓、检测精度不足的问题，开展基于YOLOv8的车辆检测与跟踪系统的研究有关键的现实价值。

1.1.2 研究意义

相和传统的目标检测相比较，基于YOLOv8的车辆检测与跟踪系统可提升检测的精度与速度，依靠引入多尺度的注意力机制[3]以及改进损失函数，YOLOv8在车辆目标检测任务里一般可达到更高的准确率与检测精度， 在复杂的交通环境中，YOLOv8借助改进骨干网络和特征融合模块，有效提高了对于小目标以及远处目标的检测能力。针对无人机视角下的车辆检测问题，YOLOv8借助优化特征提取以及损失函数的设计，提高了在背景干扰下小目标检测的鲁棒性，另外，YOLOv8还可应用于街道上的车道线和行人的多目标检测，基于无人机的多目标检测算法若结合YOLOv8模型，还可应用于智慧城市的监控和管理中。

1.2 国内外研究

1.2.1 国内研究

近些年，国内学者在把YOLOv8模型应用于车辆检测和跟踪领域取得了进展，国内

的研究成果主要是提高了检测精度，优化了算法性能并解决特定场景下的检测问题。

周飞的研究团队提出了一种轻量级的交通监控车辆检测算法，他们替换了YOLOv8的骨干特征提取网络，引入相似度注意力机制(Sim Attention)模块，这一改变提升了模型的检测精度和推理速度。他们针对密集车流和小尺度车辆的检测问题，增加了小目标检测头，并采用自适应权重调整的损失函数，优化了模型的泛化能力，实验结果说明，他们提高了检测精度(mAP)和检测速度[4]。

张利丰提出了改进的RBT-YOLOv8算法，他借助多尺度融合方式和Triplet Attention机制，提高了对小目标的检测能力[5]。引入SoftNMS代替传统非极大值抑制，提升了模型对重叠目标的处理能力，实验结果显示，该算法在PASCAL VOC数据集和COCO数据集上的性能明显强于原始的YOLOv8模型。

国内学者曾尝试将YOLOv8应用于无人机交通监控领域，史涛和崔杰所进行的研究对YOLOv8-CX算法做出改进，他们引入CF-2d模块，借此提高无人机在复杂背景下的检测精度以及实时性[6]，最终实验结果显示，该算法于Mapsai数据集上的平均精度实现了11.2%的提升。

3.0%(294)

1.2.2 国外研究

国外的学者在基于深度学习的车辆目标跟踪领域的研究成果不同于国内学者的提高检测精度和优化算法性能，他们的研究成果主要集中在多模态的数据融合，复杂场景的适应性优化以及轻量化部署等方面。

巴基斯坦和沙特阿拉伯联合团队提出的DeepSort-YOLOv8框架创新型的将FCM图像分割于ORB特征匹配相结合[7]。该系统通过卡尔曼滤波补偿无人机拍摄时的运动模糊问题，在VEDIA航拍数据集上实现了0.89的追踪精度。

希腊研究团队开发的SAR救援车载系统[8]通过重构C2f模块显著提升了处理效率，相比传统的YOLOv5降低23%推理延迟的同时保持了98%以上的手势控制准确率。他们创新性的采用无锚点设计和解耦头结构，适合用于救灾车辆等这种低算力场景下的实时跟踪需求。

沙特阿拉伯的研究团队所提出的WSA-YOLO架构通过加权空间注意力机制平衡计算效率和准确度[9]，这个方案在电力巡检无人机等移动平台上验证了可行性，它的自适应特征选择机制有效解决了林区道路等复杂背景下的跟踪漂移问题。

2.4%(236)

1.3 组织结构

本文围绕YOLOv8和DeepSort的车辆轨迹识别系统展开研究，全文分为五章，具体工作内容如下：

第一章为绪论，主要介绍了课题的研究背景和意义，以及国内外的有关于车辆检测及相关算法的研究现状。其中研究背景和意义部分，首先分析了随着城市化进程的加快和机动车持有量的快速增长，城市交通面临的交通拥堵和交通事故等问题日益突出，为了解决复杂交通环境下的车辆检测与跟踪精度不足、目标丢失等问题，本文提

3.1%(306)

出了基于YOLOv8的车辆检测与跟踪系统的研究意义和方法。国内外研究部分分别介绍了相较于传统的YOLOv8模型，国内和国外的学者们对于模型的改进。最后，组织结构中介绍了文章的脉络和各个章节的关联。

第二章阐述了达成本系统所需的目标检测以及追踪的相关技术，此部分着重讲解了YOLOv8目标检测算法和DeepSort轨迹跟踪那个算法的算法结构以及检测流程，以此为后续的实验构筑基础。

第三章说明了检测追踪系统的设计实现以及实验的结果剖析，先是介绍了本研究提出的车辆检测系统的架构，接着详尽描述了这个实验各层级的设计方案以及这个实验的最终的呈现效果，同时，也针对所实现的系统仍旧存在的问题给予一定程度的总结。

第四章为总结和展望，归纳了本文的研究内容，针对系统从设计直至实现分别基于总结，并且在末尾剖析了研究内容中的欠缺以及存在的问题，提出可行的解决方案并对未来的研究展开展望。

2.

目标检测技术相关原理

在这一章节当中主要对目标检测技术的相关理论基础展开介绍，覆盖了YOLOv8的基础架构以及DeepSort多目标检测算法这两个关键部分的架构原理与检测流程，目标检测作为计算机视觉领域的一项任务，其作用在于识别图像或者视频流里的目标类别，和图像分类任务存在差异，目标检测不只是要判断图像里是否存在特定对象，要是存在的话，还得准确识别物体的类别，像是人、小轿车或者公交车等，并且要在图像中标明该物体的位置，标注的呈现形式一般是边界框。。

图2-1 目标检测

早期阶段的目标检测方法主要是依靠人工设计的特征以及传统机器学习模型，像基于Haar特征与支持向量机的Viola-Jones方法，又或者是HOG特征结合线性分类器的方法，这些方法在特定场景之中表现还说得过去，然而普遍存在着鲁棒性欠佳、检测精度不高以及难以实现泛化的问题。随着深度学习技术不断发展，基于卷积神经网络的目标检测算法渐渐成为被选择较多的目标检测算法，其后续的YOLO系列推动了目标检测的研究与应用。。

2.1 YOLOv8基础架构

YOLO(You Only Look Once)系列是目前最具代表性的目标检测算法之一。他的核心理念是将目标检测问题转化为一个回归问题，通过单个神经网络在一次向前传播中同时预测出目标的类别和位置。

2.1.1 网络结构组成

YOLOv8的网络结构主要由Backbone，Neck和Head三大模块组成，它们共同完成图像的特征提取，特征融合和目标检测结果的输出。同前面的YOLOv5和YOLOv7相比较，YOLOv8都在结构设计上更加轻量化，适合在多种硬件平台上进行部署，尤其适合实

时检测检测的场景。

Backbone是用于从原始图像中提取深层次语义特征的主干网络。在YOLOv8中，Ultralytics团队引入了C2f(Cross Stage Partial Fusion)模块用以替代YOLO系列中常用的CSP(Cross Stage Partial)结构。

同时，YOLOv8的主干网络中还包括了多层卷积块(Conv)[11]，Bath Normalization(BN)和激活函数。

Neck网络位于Backbone和Head之间，主要功能是进行多尺度的特征融合，以便模型能够同时检测小，中，大尺寸的目标。YOLOv8的Neck网络综合了FPN和PAN的结构优势，通过这种双向的信息传递机制，TOLOv8能够更全面地理解不同尺寸的目标特征，显著提升检测精度，在小目标检测方面效果显著优于传统结构[12]。

2.3%(226)

Detection Head是YOLOv8的输出模块，负责将融合后的特征映射为具体的检测结果[13]。不同于过去的YOLOv5的基于锚框机制，YOLOv8支持的是Anchor-Free检测方式[14]，直接预测目标中心点坐标和边界框的大小，减少了对于超参数的依赖。

图2-2 YOLOv8结构

2.1.2 检测流程与输出

YOLOv8的检测流程是端到端的推理过程，从图像输入到目标检测结果输出，具有较强的实时性和部署便捷性。YOLOv8的检测输出流程可以分为图像预处理，特征提取，分类预测，非极大值抑制和结果输出可视化五个步骤。

在输入阶段，图像需经过统一的尺寸调整，一般设置为640×640。随后，图像会经过归一化操作[15]，将像素值缩放至0~1区间

经过预处理的图像被送入到YOLOv8的网络结构中。在主干部分，图像的边缘，颜色，纹理等低层特征会被逐层提取形成不同尺度的特征图。这些特征图被传入Neck模块，经过FPN和PAN的结构进行自顶向下和自底向上的双向融合，

融合过后的特征图将被输入到Head模块，YOLOv8通过Anchor-Free机制直接对目标的位置和类别进行预测。由于Head会在多个位置输出候选框，因此需要采用非极大值抑制(NMS)算法对重叠较大的候选框进行筛选。

最后，YOLOv8会输出一系列检测框，这些检测框包含类别标签和置信度。检测流程如图2-3所示。

图2-3 YOLOv8检测流程

2.2 DeepSort目标跟踪原理

DeepSort属于一种基于检测的多目标跟踪算法，该算法于SORT算法基础之上，引入深度学习的外观特征提取网络[16]，以此可让目标于复杂环境里维持ID的分配以及连续跟踪，在本项目当中，DeepSort会和YOLOv8相结合，达成对车辆的检测以及持续跟踪。

2.2.1 多目标跟踪任务概述

多目标跟踪，也就是Multi-Object Tracking，简称为MOT，是计算机视觉领域里

相当关键的一项任务，其应用范围极为广泛，涉及了智能监控、自动驾驶、交通分析以及行为识别等诸多场景，在这些场景中，多目标跟踪任务需要在连续不断的视频帧里面精确地识别出目标，同时要为每一个目标赋予一个持续且独一无二的ID，最终形成目标随时间变化的轨迹。

多目标跟踪任务一般依照先检测后跟踪的逻辑来开展，在多目标跟踪任务里，时常会出现像是多目标之间频繁地交叉或者重叠、外观较为相似、摄像头发生运动或者光照出现变化等诸多干扰因素，挑选一个高效的跟踪算法乃是达成可靠跟踪的必要前提，本文选用DeepSort算法来实施多目标跟踪。

2.2.2 算法原理与流程

DeepSort是在经典的SORT算法上发展而来的多目标跟踪算法，其最大的特点是结合了深度学习提取的外观特征[17]，使得算法在处理遮挡，目标相似等复杂情况时更加稳定和准确。DeepSort的整体工作流程分为：目标检测结果接收，卡尔曼滤波器预测目标状态，外观特征提取，多指标数据关联和轨迹更新。

DeepSort并不会进行目标检测，它依赖于外部的检测器（例如YOLO系列）在每一帧当中提供检测框的信息。每一个检测框包含边界框的坐标信息[x，y，w，h]，目标的置信度和类别标签。这些信息会作为DeepSort的输入基础。

DeepSort运用卡尔曼滤波器针对每个跟踪目标的状态开展建模以及预测工作，物体的状态向量囊括位置、速度以及加速度，DeepSort借助持续更新预测和观测结果之间的误差，达成目标的轨迹线路在帧与帧之间的连贯衔接，为处理目标外观相似以及重叠的问题，DeepSort会于每个目标检测框上提取目标的外观特征向量，这部分依靠一个预训练的Re-Identification(ReID)网络，一般是基于轻量级的CNN模型。。

DeepSort采用综合关联策略以达成准确稳定的目标匹配[18]，此策略主要把运动信息和外观特征两方面的距离度量相结合，运动信息借助马氏距离来计算，外观特征运用余弦距离来计算，最终利用线性加权和，并借助匈牙利算法将这两个指标结合起来[19]。

匹配完成后，DeepSort会更新每个轨迹的状态信息。如果检测目标与已有的轨迹相匹配，就更新他的状态；如果出现新的目标，就创建一条匹配的新的轨迹[20]。算法流程图如图2-4所示。

图2-4 DeepSort算法流程图

2.3 本章小结

本章介绍了本系统中采用的目标检测与跟踪的核心技术：YOLOv8和DeepSort。首先对目标检测的发展脉络进行了概述，随后详细阐述了YOLOv8的网络结构，检测流程与输出机制。接着，分析了DeepSort的跟踪原理与数据关联流程，突出其通过融合运动信息和外观特征实现多目标的持续跟踪的能力。两者的结合为系统提供了高效，稳定的车辆检测与跟踪的能力，为后续的系统设计和实现奠定了基础。

3.

车辆检测跟踪系统

本章采用YOLOv8架构和DeepSort多目标跟踪算法，设计并实现了对于车辆的检测跟踪系统。本章主要介绍系统的主要流程，方法，系统的实现过程和最后的结果。

3.1 系统硬件配置和软件环境

本系统硬件配置主要为AMD Ryzen 7 5800H with Radeon Graphics的处理器，16GB内存，显卡配置为NIVIDA GeForce RTX 3050 Ti。

本系统的软件使用Windows 11操作系统作为开发环境，版本型号3.9.21的python为开发语言，采用VS Code作为开发工具，Anaconda版本为25.3.0，基于Pytorch2.2.1深度学习框架和Opencv图像处理库进行系统开发与实现。系统界面采用PyQt构建，以实现用户操作与检测结果可视化，同时通过MySQL数据库实现监测数据的存储。

3.2 系统的总体设计

车辆的检测与跟踪系统针对上传的视频流进行分析检测，从而实现对于车辆的检测与追踪。

首先，系统会接收用户上传的图片或者视频资源。其次，用户根据系统左侧的选项列表选择想要进行的操作，如对于图片的分析，对于视频的检测，对于视频的追踪或越线计数。接着，用户对于检测的各项参数进行设置，比如选择检测的目标类别或设置检测的置信度。最后，点击分析按钮，系统将在后台对于图片或视频流进行检测或追踪，识别的结果将在系统完成操作后呈现在程序右侧的边框中。

该系统使用的是三层架构，分别为表示层，应用逻辑层和数据层。表示层主要负责用户的交互界面，交互界面基于PyQt实现，主要分为用户登录界面和用户操作界面。应用逻辑层承载了该系统的核心业务逻辑，主要包含了以下子模块：图像检测模块，视频检测模块，轨迹分析模块和越线计数模块。数据层主要负责系统运行过程中的数据存储和加载，包括用户的登录信息和系统分析的图像视频文件。

3.6%(357)

3.3 系统的详细设计与实现

本系统选用Python作为核心开发语言，采用模块化逐步搭建各种功能。在开发的过程中，系统遵循三层架构思想，从项目结构设计开始就进行了功能分区和文件组织，以保障后续开发的清晰。

3.3.1 系统环境与依赖

在本系统的开发过程中，为了保证各种功能模块的完整性和运行的稳定性，开发环境采用了基于Conda的虚拟环境进行构建，开发语言为Python 3.9.21，后续所有的依赖包都是通过Conda或pip安装。

本项目使用Anaconda环境管理工具创建了独立的虚拟环境，环境名称为venv，虚拟环境如图3-1所示。

图3-1 虚拟环境

在图形界面中，系统采用5.15.10版本的PyQT5构建图形用户界面，通过.ui文件实

现主窗口和功能模块的交互。关于数据处理方面，pandas使用2.2.1版本，numpy使用1.26.4版本，其余主要依赖见图3-2所示。

图3-2 依赖列表

数据库方面，本系统使用MySQL Workbench可视化软件进行操作，具体使用的数据库版本为MySQL 8.0.32，如图3-3所示。

图3-3 数据库操作台

本节详细介绍了系统开发与运行所依赖的环境，包括操作系统平台，Python开发环境，关键依赖库以及MySQL数据库的版本。通过调整开发环境和各个依赖项来确保系统可以稳定运行，为后续的系统设计和代码编写奠定基础。

3.3.2 表示层

表示层是系统与用户直接交互的部分，主要负责界面展示，用户输入数据的采集，可视化结果的呈现等功能。在本系统当中，表示层以PyQt5框架为基础搭建，采用Qt Designer设计静态界面，通过逻辑代码实现动态行为响应。

代码段当中，涉及到表示层的代码主要包括登录窗口LoginDialog(QDialog)和主窗口MyWindow(QWidget)。

```
class LoginDialog(QDialog):
def __init__(self, parent=None):
super(LoginDialog, self).__init__(parent)
uic.loadUi("dialog_new.ui", self)
self.loginButton.clicked.connect(self.handle_login)
class MyWindow(QWidget):
def __init__(self):
super(MyWindow, self).__init__()
uic.loadUi("main_new.ui", self)
self.pushButton_open.clicked.connect(self.load_image)
self.pushButton_detect.clicked.connect(self.detect_objects)
```

用户首次启动系统时，需要通过登陆界面完成注册登陆操作，界面包含用户名，密码输入框，注册按钮和登录按钮，以上所有的输入行为都通过槽函数和数据库校验逻辑绑定。登录界面如图3-4所示。

图3-4 登陆界面

没有使用过本系统的用户需要首先进行用户的注册，系统运行之后默认界面是登陆界面，需要先切换到注册界面，输入username和password之后点击register进行注册。显示注册成功后再切换回登陆界面使用用户名和密码进行登录，若全部输入正确会显示“登陆成功”，否则会显示“登陆失败，请重试”。认证界面如图3-5所示。

(a)

(b)

(c)

图3-5 认证界面。

主窗口是系统功能的主体，包含多个选项卡，如图像检测，视频分析，车辆轨迹追踪和越线计数。每个功能区域对应独立的按钮，滑块，文件选择控件等界面元素，用户可以通过点击或拖动来实现与系统的交互。主界面如图3-6所示。

图3-6 主窗口

本系统的表示层主要负责前端功能模块的展示和用户操作的收集。界面采用多标签结构，将不同的模块分别放在不同的标签页当中以此减少界面的混乱。整体上，表示层基本实现了预定的人机交互任务，用户可以在不涉及到底层代码的情况下操作系统的主要功能，但是在功能聚合和交互一致方面还有优化空间。

3.3.3 应用逻辑层

应用逻辑层是系统的核心处理单元，承担全部的业务逻辑，模型推理，行为分析和数据处理的智能。在系统运行的过程中，所有用户的请求操作都会通过表示层转发到应用逻辑层，由它来完成实际运算和处理任务。在本系统当中应用逻辑层主要包含以下的模块:YOLOv8检测模块，DeepSort多目标追踪模块和越线行为统计模块。其中YOLOv8检测模块用于完成图像目标检测和视频目标检测，加入DeepSort多目标追踪模块用于完成轨迹分析。后文中所系统所使用的视频为2025-03-31_01.mp4，取自BiliBili平台作品。下面对应用逻辑层当中主要模块的实现过程和最终结果进行分析和展示。

图像目标检测模块主要是针对静态图像中的车辆目标进行识别，实现过程包括模型加载，目标预测，检测结果绘制以及回传展示图像。本模块使用的YOLOv8模型是yolov8n.pt，主要代码如下。

2. 基于YOLOv8的车辆检测与跟踪系统的设计与实现.doc_第2部分

AI特征值：9.1%　　AI特征字符数 / 章节(部分)字符数：601 / 6598

片段指标列表

序号	片段名称	字符数	AI特征	
8	片段1	360	疑似	5.5%
9	片段2	388	显著	5.9%
10	片段3	244	疑似	3.7%
11	片段4	213	显著	3.2%
12	片段5	263	疑似	4.0%

13	片段6	204	疑似		3.1%

原文内容

```
def detect_objects(self):
model = YOLO('yolov8n.pt') # 加载YOLOv8模型
results = model.predict(self.image_path, conf=0.5)
res_plotted = results[0].plot() # 自动绘制检测框
output_path = "detect_output.jpg"
cv2.imwrite(output_path, res_plotted) # 保存检测结果
self.show_image(output_path) # 显示在界面上
```

该模块实现了从用户输入图像到检测结果可视化的完整流程，适用于单张图想的车辆识别任务，结果如图3-7所示。

图3-7 图像目标检测

视频目标检测模块对上传的整段视频的每一帧进行检测，其处理过程类似于图像检测，这个模块依旧使用了YOLOv8模型完成对车辆目标的识别任务，但增加了帧读取和视频输出的逻辑，检测结果会被写入新的文件夹当中。

5.5%(360)

系统首先通过OpenCV的VideoCapture接口逐帧读取视频内容，并将每一帧图像输入到YOLOv8模型当中进行推理分析。检测后的每帧图像会自动绘制出目标边界框与类别标签，形成标注图像，随后使用VideoWriter模块将所有带有检测结果的帧连续写入输出视频流当中。完成之后，系统会生成一段全新的检测视频，并将其保存在指定的文件夹当中。模型主要代码如下，效果展示如图3-8所示。

```
def showVideo(self):
cap = cv2.VideoCapture(self.video_path)
model = YOLO('yolov8n.pt')
output_path = "video_output/processed.avi"
out = cv2.VideoWriter(output_path, fourcc, fps, size)
while cap.isOpened():
ret, frame = cap.read()
if not ret:
break
results = model.predict(frame, conf=0.5)
annotated = results[0].plot()
out.write(annotated)
cap.release()
out.release()
```

图3-8 视频目标检测

在视频的目标检测当中，虽然可以识别出视频中每一帧的车辆目标，但缺乏对目标身份连续性的认识，无法判断某一辆车在多帧之间是否连续存在，也无法绘制他的行驶轨迹。故而在多目标追踪模块当中通过融合YOLOv8检测器和DeepSort跟踪算法来实现对检测目标的跨帧身份保持和轨迹绘制。系统在追踪过程中，保证persist为True，即保持连续跟踪状态。检测与追踪的结构化结果将通过results提供，分别是YOLO格式的中心点坐标(x，y，w，h)，追踪器分配的目标唯一的ID和YOLO模型预测的目标类别cls。为了记录每个目标在整个视频中多帧的连续运动，系统会在循环中不断地将每帧的坐标信息添加到对应ID的轨迹列表当中。最后，在处理过程中系统使用OpenCV的线段渲染，在视频上实时绘制每个目标的轨迹。在系统分析完成后可以选择保存并且导出这次分析的数据。使用到的关键代码如下，展示结果如图3-9所示。

```
boxes = result.boxes.xywh.cpu()
track_ids = result.boxes.id.int().cpu().tolist()
clss = result.boxes.cls.int().cpu().tolist()
for box, track_id, cls in zip(boxes, track_ids, clss):
x, y, w, h = box.tolist()
cx, cy = int(x), int(y)
self.locations[track_id].append({
"x": cx,
"y": cy,
"c": cls
})。
```

图3-9 多目标追踪轨迹绘制

越线计数模块也是目标行为检测逻辑的一部分，其主要功能是统计在上传的视频中越过虚拟线段的车辆数量，以此来实现对于交通流量的基本监测。该模块是建立在多目标追踪模块基础之上的，通过 DeepSort 追踪算法为检测到的每个目标分配唯一的ID，并记录其在视频中的运动轨迹。系统在每一帧中提取目标的中心坐标，并判断该中心点是否穿越了设定的逻辑检测线，一旦中心目标穿越了设定的逻辑检测线，就会显示越线数量加1。

为避免重复统计，系统会使用集合结构记录已越线的目标 ID，确保每个目标只在首次越线时计数一次。逻辑检测线为一条水平方向的虚拟线段，在检测过程中实时绘制在视频帧上，并同步更新越线总数。用户可以通过系统界面查看越线数量，检测线位置以及目标标注结果，从而对交通状况进行一个初步的评估。其关键实现代码将在下文展示，检测与统计结果如图 3-10 所示。

```
for box, track_id in zip(boxes, track_ids):
x, y, w, h = box
```

5.9%(388)

```
center_y = y + h / 2
if center_y < line_position and track_id not in crossed_vehicles:
crossed_line_count += 1
crossed_vehicles.add(track_id)
```

图3-10 越线计数

应用逻辑层中最后实现的功能是系统加载前文提到的保存的分析数据，在表示层当中显示出保存数据的内容，一般识别.xlsx文件。系统在加载完成之后可以选择目标的被分配唯一ID来绘制车辆的轨迹路线，同时系统将绘制轨迹检测识别视频文件当中所识别到的车辆疏密热力图。关键代码如下，结果如图3-11所示。

```
data = pd.DataFrame(columns=['x', 'y'])
data['x'] = self.df['X']
data['y'] = self.df['Y']
res = np.zeros((7, 7))
heatmap = (
HeatMap(init_opts=opts.InitOpts(width="650px", height="500px"))
.set_global_opts(
title_opts=opts.TitleOpts(title="热力图"),
visualmap_opts=opts.VisualMapOpts(
max_=150,
)
self.hotmap.setHtml(heatmap.render_embed())
```

(a)

（b）

图3-11 检测数据集和数据分析

应用逻辑层作为三层架构的中间处理层次，承担了检测和分析功能的具体实现。本系统中，应用逻辑层主要负责调用YOLOv8模型完成图像与视频的目标识别，并在视频的轨迹分析功能模块当中调用track()接口实现基于DeepSort的多目标追踪。总体上逻辑处理流程按照函数为单位划分，具备一定的结构清晰度，但是模块之间的耦合程度相对较高。

3.3.4 数据层

数据层是用来管理，读取和保存各类数据资源的后端模块，主要包含用户信息，输入的媒体（包括图片和视频），检测的输出结果以及可视化展示所需要的静态图像资源等。在本系统当中，数据层没有作为独立的模块进行封装，而是嵌套进表示层和应用逻辑层当中，承担着数据流的输入，中转和输出。下面对数据层起到的功能和具体作用进行分析和展示。

系统通过连接MySQL数据库完成用户的登录和注册功能，它依赖于封装的

MysqlTool工具类来实现数据库的增添，删除，查阅和改变功能。创建的数据库名称为pyqt_yolov8，数据表的设计如下表所示。用户在图形界面输入用户名和密码之后，系统会通过sql语句进行数据库验证。

表3-1 user数据表

字段名说明数据类型主键非空

Id 用户唯一标识 INT YES YES

Name 用户名 VARCHAR (50) YES YES

Pwd 用户登陆密码 VARCHAR(255) NO YES

Created_at 账户创建时间 TIMESTAMP NO NO

图像和视频资源的输入以及输出的资源管理都是通过数据层来实现的。输入数据需要通过表示层的文件对话框选择路径，在应用逻辑层通过OpenCV读取。图像检测结果的保存方式较为简单，会使用cv2.imwrite()函数直接通过图片的形式输出，视频的输出需要完整的视频写入流程检测，检测结果保存为.mp4格式文件存入指定路径的文件夹下面。

本系统在完成目标检测与多目标追踪之后，会对各个追踪目标在视频中的运动轨迹进行结构化记录，并支持将该轨迹数据导出为标准的 .csv 或 .xlsx 文件格式。这一模块的主要作用是实现检测结果的数据持久化，方便后续进行轨迹回溯、行为分析或统计建模等操作，同时也可作为用户进一步研究与结果验证的基础数据支持。

在系统运行过程中，追踪模块会为每个检测目标分配唯一的 Track ID，并持续记录其在各个视频帧中的空间坐标。系统最终将其导出为包含 Track ID、X、Y、Category 四个字段的 .xlsx 表格文件，分别对应目标编号、中心点坐标及类别信息。该结构简洁清晰，便于后续通过 Excel、Matplotlib 等工具进行热力图、柱状图等可视化分析。系统已在“检测数据集分析”模块中集成相关功能，实现从检测追踪到数据统计的完整流程。关键实现代码见下文，示意结果如图 3-12 所示。。

3.7%(244)

```
def export_location(self):
rows = []
for track_id, records in self.locations.items():
for r in records:
rows.append({
"Track ID": track_id,
"X": r["x"],
"Y": r["y"],
"Category": r["c"]
})
df = pd.DataFrame(rows)
df.to_excel("track_output.xlsx", index=False)
```

图3-12 保存数据

数据层在本系统当中承担了用户的信息管理，图像资源的读写，检测结果的输出和轨迹数据的导出，是支撑系统运行的核心。在用户验证方面，系统通过本地MySQL数据库来存储用户的基本信息。图像与视频资源的输入和输出主要通过OpenCV实现。但从系统的设计和实现情况来看，数据层仍存在着结构性和安全性上的不足。在当前版本中的数据层以简化逻辑为主，仅仅适合本地演示，在数据隔离，安全性和异常处理方面仍需要进一步完善。

3.4 本章小结

本章围绕车辆检测系统的功能结构和实现方式展开叙述，结合系统使用的三层架构，对表示层，应用逻辑层和数据层进行了逐一的分析。系统通过图形界面实现了图像与视频的加载和展示，应用逻辑层集成了YOLOv8和DeepSort完成目标检测与跟踪，数据层则负责用户的验证，文件的读写以及检测结果的输出。综合来看，本系统基本实现了设计时的基本功能，完成了具有基本的完整的分析能力的车辆检测与追踪功能。同时，本章也指出了本系统在数据安全性，路径管理和结构优化等各方面的不足，为后续的改进提供方向。

4. 总结与展望

4.1 研究内容总结

本文围绕“基于YOLOv8的车辆检测和跟踪系统的设计与实现”这一主题展开，本文希望可以构建一个具备图像和视频目标检测，跟踪和分析能力的多功能系统。目前，随着我国城市交通智能化程度的不断提高，车辆检测和跟踪作为智能交通系统当中的关键任务，已经逐渐成为分析研究的重点。传统的基于特征匹配或背景建模的检测方法在如今日益复杂的环境中已经开始表现出鲁棒性不足的问题，与之相对应的，基于深度学习的目标检测算法凭借着他的出色的检查速度和高精准度正在一步步的成为智能交通系统当中重要的一环。

在这种背景下，本文选择使用YOLOv8作为系统的目标检测主干网络。YOLOv8相较于以往的版本，在结构设计和推理效率等方面均有显著的提升，能够保持在具有较高的检测精度的同时实现足够的检测分析速度。同时，本文结合了DeepSort多目标追踪算法，为检测目标分配了唯一ID，实现了对于车辆的连续追踪和轨迹记录。YOLOv8和DeepSort的组合兼顾了检测和跟踪的效率以及稳定性，为构建轻量级的，可部署的系统平台提供了技术支持。

3.2%(213)

在系统的架构方面，本系统采用三层架构设计，将系统划分为表示层，应用逻辑层和数据层。表示层基于PyQT5框架构建图形化界面，支持图像和视频的加载，功能选择，参数调节和结果的展示。应用逻辑层是系统的功能核心，整合了检测，追踪，越线计数，轨迹统计和图表绘制等任务功能。数据层则提供了用户验证，媒体文件的读写和检测结果的导出等底层支撑功能。

系统所实现的主要内容包括静态图像目标检测，视频目标检测，视频实时轨迹绘

制，多目标越线计数和轨迹信息的结构化导出。本系统支持对于车辆的检测结果的多维度分析，为交通检测和流量统计可以提供基本的数据支撑能力。通过图形化界面，用户可以直观的选择参数，输入数据，并查看检测的结果和分析图表，一定程度上有效的降低了操作的门槛，提升了实际的操作效率。

在系统的实际开发过程中，实现了功能结构的基本设计和模块集成，确保了功能实现的合理性和逻辑性。同时，本文在撰写的过程中，也分了各模块的技术原理和设计依据，总结了本系统在功能实现和运行流程方面的优势和不足整体而言。本研究实现了一个功能相对完整，结构相对清晰，技术可控的车辆检测与跟踪系统，为相关领域的研究和应用提供了一定的技术参考。

4.0%(263)

4.2 未来工作展望

尽管本系统已经基本完成了车龄目标的检测，追踪和分析等主要功能，但从系统的扩展性，稳定性以及面向实际场景的部署的角度来看，本系统仍然具有相当多的值得深入改进与拓展的地方。

首先，在系统的功能层面，目前本系统仅仅支持用户上传图像或视频进行处理，缺少调用摄像头进行实时分析的功能模块。在未来的工作中，可将系统扩展为支持摄像头的实时监测平台。

其次，在用户的管理和系统安全方面，目前的用于认证机制仅仅通过铭文用户名和密码进行数据库匹配，缺少加密机制和权限等级划分，存在一定的安全风险。后续考虑引入哈希加密和权限控制机制，实现不同用户角色（如管理员和普通用户）下的功能限制和数据隔离。

3.1%(204)

在算法层面，虽然YOLOv8和DeepSort的组合能够满足一般检测与跟踪任务的需求，但是在面对一定程度上的高速移动，目标遮挡，光照变化等复杂场景的时候，仍然存在鲁棒性问题。后续考虑针对YOLOv8的架构进行改进，以提升它的检测精准度。

最后，本系统在UI交互流程和用户体验方面仍然存在较大的提升空间。未来可以引入响应式界面设计，批量的数据处理机制，拖拽式文件上传等功能。

综上所述，本论文所构建的车辆检测与跟踪系统具备一定的实用性和扩展能力，但是仍然处于功能集成的初级阶段。后续工作将围绕系统性能优化，功能扩展等方面展开，不断完善系统架构与实际场景的匹配能力，为智能交通分析和视频检测系统的进一步发展提供支撑。

致谢

在毕业论文的撰写和系统的开发过程中，我得到了导师的悉心指导和耐心帮助。导师不仅仅在选题，结构设计，技术实现方面提供了专业指导，也在我遇到困难时给予我鼓励，使我能够逐步完成系统的设计和开发。

同时，感谢我的同学在我开发过程中的支持和配合，大家在测试数据，功能优化方面提出了许多有价值的建议。此外，感谢家人在我的整个学习阶段给予的理解和支

持，使我能够心无旁骛地完成毕业设计任务。

附录

说明:

1、支持中、英文内容检测;

2、AI特征值=AI特征字符数/总字符数;

3、红色代表AI特征显著部分，计入AI特征字符数;

4、棕色代表AI特征疑似部分，未计入AI特征字符数;

5、检测结果仅供参考，最终判定是否存在学术不端行为时，需结合人工复核、机构审查以及具体学术政策的综合应用进行审慎判断。

cx.cnki.net