REAL-TIME INTELLIGENT SURVEILLANCE SYSTEM USING YOLO

A PROJECT REPORT

Submitted by

NAVEEN V

(2023179007)

submitted to the Faculty of

INFORMATION AND COMMUNICATION ENGINEERING

in partial fulfillment

for the award of the degree

of

MASTER OF COMPUTER APPLICATIONS

in

INFORMATION TECHNOLOGY

DEPARTMENT OF INFORMATION SCIENCE AND TECHNOLOGY

COLLEGE OF ENGINEERING, GUINDY

ANNA UNIVERSITY

CHENNAI 600 025

MAY 2025

ANNA UNIVERSITY

CHENNAI - 600 025

BONA FIDE CERTIFICATE

Certified that this project report titled **"REAL-TIME INTELLIGENT SURVEILLANCE SYSTEM USING YOLO"** is the bonafide work of **NAVEEN V (2023179007)** who carried out project work under my supervision. Certified further that to the best of my knowledge and belief, the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or an award was conferred on an earlier occasion on this or any other candidate.

S. Sridhar

PLACE: CHENNAI

DATE: 16-05-2025

Dr. S. SRIDHAR
PROFESSOR
PROJECT GUIDE
DEPARTMENT OF IST, CEG
ANNA UNIVERSITY
CHENNAI 600025

COUNTERSIGNED

15/5/25

DR. S. SWAMYNATHAN
HEAD OF THE DEPARTMENT
DEPARTMENT OF INFORMATION SCIENCE AND TECHNOLOGY
COLLEGE OF ENGINEERING, GUINDY
ANNA UNIVERSITY
CHENNAI 600025

ABSTRACT

Traditional traffic surveillance systems used for vehicle speed monitoring often rely on dedicated hardware such as infrared speed sensors, microwave radar sensors, and laser-based speed detectors. These advanced sensors, while relatively easy to install, access, and maintain, significantly increase the cost and complexity of deployment. To address these challenges, this project presents a real-time intelligent surveillance system capable of estimating vehicle speed and detecting helmet usage directly from traffic video without requiring any additional sensors other than a camera. The proposed system integrates a suite of advanced computer vision techniques: YOLO for object detection, perspective transformation for converting pixel displacement into real-world distances, DeepSORT for multi-object tracking, and a combination of Good Features to Track with the Lucas-Kanade optical flow method for motion estimation. For helmet detection, a YOLO-based classifier is trained to differentiate between helmeted and non-helmeted riders, while DeepSORT maintains object identity across frames. A Flask-based web dashboard, developed using HTML and CSS, provides real-time visualization of vehicle speeds, helmet compliance status, and tracking overlays. The system was evaluated using 30 recorded road surveillance videos, with ground-truth vehicle speeds obtained from an in-vehicle speedometer during actual driving sessions. Performance metrics for the speed estimation module include a Mean Absolute Error (MAE) of 1.50 km/h, a Root Mean Square Error (RMSE) of 1.84 km/h, and a Mean Absolute Percentage Error (MAPE) of 3.37%, demonstrating the system's high accuracy. These results validate the system as a cost-effective, sensor-free alternative to conventional speed enforcement methods, suitable for scalable deployment in real-world traffic surveillance and road safety monitoring.

திட்டப்பணிச்சுருக்கம்

பாரம்பரிய போக்குவரத்து கண்காணிப்பு அமைப்புகள் வாகன வேகத்தைக் கண்காணிக்க அகச்சிவப்பு வேக உணர்விகள், மைக்ரோவேவ் ரேடார் உணர்விகள் மற்றும் லேசர் அடிப்படையிலான வேக கண்டறிதல் போன்ற பிரத்யேக வன்பொருளைப் பயன்படுத்துகின்றன. இந்த வன்பொருட்கள் நிறுவ, பராமரிக்க மற்றும் அணுக எளிதாக இருந்தாலும், அவை செலவும் சிக்கலும் அதிகரிக்கும் தன்மையைக் கொண்டவை. இந்த சவால்களை சமாளிக்க, இந்த திட்டம் எந்தவொரு கூடுதல் சென்சார்களையும் பயன்படுத்தாமல், கேமரா மட்டும் கொண்டு போக்குவரத்து வீடியோவிலிருந்து நேரடியாக வாகன வேகத்தை மதிப்பீடு செய்வதற்கும், ஹெல்மெட் அணிந்திருக்கிறார்களா என கண்டறிவதற்குமான நிகழ்நேர அறிவார்ந்த கண்காணிப்பு அமைப்பை உருவாக்குகிறது. முன்மொழியப்பட்ட அமைப்பு பல முன்னோடியான கணினி பார்வை நுட்பங்களை ஒருங்கிணைக்கிறது: YOLO அடிப்படையிலான பொருள் கண்டறிதல், முன்னோக்கு மாற்றம் (perspective transformation), DeepSORT அடிப்படையிலான பல பொருள் கண்காணிப்பு, மற்றும் Lucas-Kanade optical flow முறையைச் சார்ந்த இயக்க மதிப்பீடு ஆகியவை இதில் அடங்கும். ஹெல்மெட் கண்டறிதலுக்காக YOLO மாதிரி, ஹெல்மெட் அணிந்த மற்றும் அணியாத இருசக்கர வாகன ஓட்டுநர்களை வேறுபடுத்த பயிற்சி பெறுகிறது. DeepSORT ஒவ்வொரு சட்டகத்திலும் தனித் தனிப் பொருட்களின் அடையாளத்தை தொடர்கிறது. HTML மற்றும் CSS மூலம் உருவாக்கப்பட்ட Flask அடிப்படையிலான வலை டாஷ்போர்டு, நிகழ்நேர வேகம், ஹெல்மெட் பயன்பாட்டு நிலை மற்றும் கண்காணிப்பு மேலடுக்குகளை காட்சிப்படுத்துகிறது. இந்த அமைப்பு 30 சாலை கண்காணிப்பு வீடியோக்களில் சோதனை செய்யப்பட்டது. உண்மையான வாகன ஓட்டம் அமர்வுகளில் ஸ்பீடோமீட்டரில் பதிவு செய்யப்பட்ட வேகங்கள் ஒப்பீட்டிற்காக பயன்படுத்தப்பட்டன. வேக மதிப்பீட்டுக்கான செயல்திறன் அளவீடுகளில் சராசரி முழுமையான

பிழை (MAE) 1.50 கிமீ/மணி, இருபடி மூல பிழை (RMSE) 1.84 கிமீ/மணி மற்றும் முழுமையான சதவீத பிழை (MAPE) 3.37% ஆகியவை அடங்கும், இது அமைப்பின் உயர் துல்லியத்தை உறுதிப்படுத்துகிறது. இந்த முடிவுகள், செலவு குறைந்த மற்றும் சென்சார் இல்லாத மாற்றாக, இந்த அமைப்பை உண்மையான உலக போக்குவரத்து கண்காணிப்பு மற்றும் சாலை பாதுகாப்பு பயன்பாடுகளுக்குத் தகுந்ததாக நிரூபிக்கின்றன.

ACKNOWLEDGEMENT

It is my privilege to express my deepest sense of gratitude and sincere thanks to **Dr. S. SRIDHAR**, Professor, Project Guide, Department of Information Science and Technology, College of Engineering, Guindy, Anna University, for her constant supervision, encouragement, and support in my project work. I greatly appreciate the constructive advice and motivation that was given to help me advance my project in the right direction.

I am grateful to **Dr. S. SWAMYNATHAN**, Professor and Head, Department of Information Science and Technology, College of Engineering Guindy, Anna University for providing us with the opportunity and necessary resources to do this project.

I would also wish to express my deepest sense of gratitude to the Members of the Project Review Committee: **Dr. P.VARALAKSHMI,** Professor, **Dr. K. KULOTHUNGAN,** Associate Professor, **Dr. S. BAMA SRINIVASAN,** Associate Professor, **Ms K. MOHANA BHINDU**, Teaching Fellow, Department of Information Science and Technology, College of Engineering Guindy, Anna University, for their guidance and useful suggestions that were beneficial in helping me improve my project.

I also thank the faculty member and non teaching staff members of the Department of Information Science and Technology, Anna University, Chennai for their valuable support throughout the course of our project work.

NAVEEN V

(2023179007)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

YOLO	You Only Look Once (a real-time object detection algorithm)
DeepSORT	Deep Simple Online and Realtime Tracking (object tracking algorithm using appearance descriptors)
GFT	Good Features to Track (feature detection algorithm)
LK	Lucas-Kanade (optical flow tracking algorithm)
FPS	Frames Per Second
MAE	Mean Absolute Error
RMSE	Root Mean Square Error
CNN	Convolutional Neural Network
API	Application Programming Interface
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
AJAX	Asynchronous JavaScript and XML
UI	User Interface
mAP	Mean Average Precision (used in object detection evaluation)
JSON	JavaScript Object Notation
CCTV	Closed-Circuit Television
GPU	Graphics Processing Unit
GT	Ground Truth
ML	Machine Learning
DL	Deep Learning
CV	Computer Vision
FID	Frechet Inception Distance (image quality metric)
IS	Inception Score (image quality metric)
PSNR	Peak Signal-to-Noise Ratio
MSE	Mean Squared Error
MAPE	Mean Absolute Percentage Error

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND OF SURVEILLANCE SYSTEMS

Surveillance systems have evolved significantly over the past few decades, driven by growing urbanization, increased vehicular density, and the rising demand for automated traffic management and public safety enforcement. These systems are now widely deployed at city intersections, highways, and sensitive public areas to monitor vehicular movement, detect violations, and gather statistical data. Traditionally, surveillance technologies are categorized into two major types: intrusive and non-intrusive systems. Intrusive systems involve physical sensors embedded into the roadway infrastructure such as inductive loops, piezoelectric sensors, and magnetic field detectors which provide highly accurate vehicle data but require costly installation, regular maintenance, and road surface disruption.

Non-intrusive systems, in contrast, rely on external sensors like video cameras, infrared sensors and microwave radars. Among these, camera-based surveillance systems have gained particular attention due to their lower cost and ability to capture complex visual information. With advancements in deep learning and real-time computer vision, such systems now support not just detection but also behavior analysis, including vehicle counting, lane tracking, speed measurement, and safety compliance monitoring. These developments have paved the way for intelligent surveillance solutions capable of detecting traffic violations and promoting road safety, especially in regions with high accident rates involving two-wheelers.

This project proposes a real-time intelligent surveillance system focused on two critical objectives: vehicle speed estimation and helmet detection for two-wheeler riders. The system leverages deep learning and computer vision models to process only video input no additional sensors are required. Object detection is performed using YOLO (You Only Look Once), while tracking is managed through DeepSORT (Deep Simple Online and Realtime Tracking). Lucas-Kanade optical flow and Good Features to Track are employed for motion estimation, and perspective transformation enables conversion from pixel displacement to real-world distance for accurate speed calculation. Helmet detection is performed by training YOLO to distinguish between helmeted and non-helmeted riders. Finally, a Flask-based web dashboard developed with HTML and CSS provides real-time visualization and analytics. This unified approach ensures a low-cost, scalable, and accurate surveillance solution that treats both helmet compliance and speed monitoring as equally vital components of traffic safety.

1.2 PROBLEM STATEMENT

Conventional speed monitoring systems depend heavily on physical sensors like radars or infrared detectors, which are expensive, infrastructure-dependent, and not suitable for flexible deployment. Additionally, enforcing helmet laws manually remains labor-intensive and error-prone. There is a clear need for a unified, camera-based surveillance system that can automatically detect helmets and accurately estimate vehicle speeds without the support of any additional external sensors.

1.3 BLOCK DIAGRAM

The architecture of the proposed real-time surveillance system is illustrated in Figure 1.1. It highlights the minimal and essential components involved in video-based speed estimation and helmet detection.

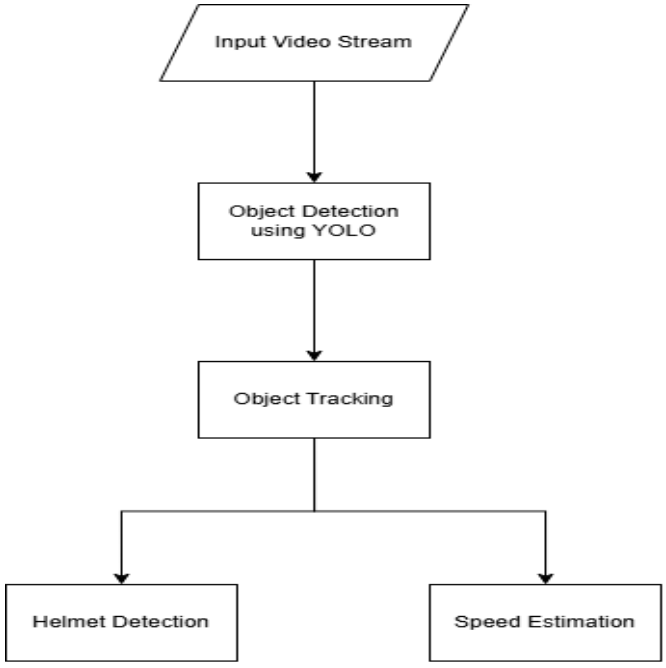

Figure 1.1: Block diagram of the proposed surveillance system.

1.4 OBJECTIVES

The primary goal of this project is to develop an intelligent, camera-based traffic monitoring system that utilizes computer vision techniques to enforce road safety regulations. The specific objectives of the system are as follows:

To develop a video-based system that accurately estimates the speed of moving vehicles using computer vision techniques, eliminating the need for physical speed sensors.

To implement an automated helmet detection mechanism using a deep learning model (YOLO) trained to identify helmet compliance among two-wheeler riders in real-time.To build a web-based dashboard interface that allows users to monitor traffic violations, including speed and helmet usage, with live updates and analytics visualization.

1.5 PROPOSED SYSTEM

The proposed solution begins by collecting 30 road surveillance videos, each recorded during real driving scenarios with concurrent ground-truth speed measurements obtained from a vehicle's speedometer. These recordings serve as the dataset for system development and validation. For the helmet detection component, a subset of video frames was manually annotated to label helmeted and non-helmeted riders, forming a labeled dataset for training a YOLO model.

The system pipeline integrates several computer vision components. YOLO is used to detect vehicles and riders, while DeepSORT is employed to track each detected object across frames using motion and appearance cues. Perspective transformation is applied to map pixel-based displacement into real-world distances, enabling accurate speed calculation. To enhance tracking stability and motion estimation, Good Features to Track and the Lucas-Kanade optical flow algorithm are used. These techniques collectively allow the system to estimate the velocity of each tracked object with high accuracy, while maintaining identity consistency.

Hyperparameters in DeepSORT such as the cosine similarity threshold, object lifespan, and minimum detection hits were tuned to optimize tracking reliability across crowded and occluded scenes. The entire system is integrated into a Flask-based web dashboard, developed using HTML and CSS,

which displays live results including vehicle speeds, helmet usage, and tracking visualizations. The model's accuracy was evaluated against ground-truth speed data using Mean Absolute Error (MAE) and Root Mean Square Error (RMSE), achieving best values of 1.50 km/h for MAE and 1.84 km/h for RMSE, respectively. These results confirm that the proposed approach can reliably estimate vehicle speed and detect helmet violations in real-time using only video input.

1.6 ORGANIZATION OF THE REPORT

This report is organized into 6 chapters, describing each part of the project with detailed illustrations and system design diagrams.

CHAPTER 1: Introduction presents an overview of the research background and motivation for intelligent surveillance in road environments. It outlines the challenges with traditional sensor-based systems and introduces the problem of vehicle speed estimation and helmet detection using camera-based, deep learning techniques. The chapter includes the problem statement, objectives, proposed solution, and the scope of the study.

CHAPTER 2: Literature Review reviews prior work and advancements in computer vision-based traffic monitoring systems. It covers traditional sensor-based speed estimation methods and recent deep learning approaches, including the use of YOLO for object detection, DeepSORT for object tracking, and optical flow for motion analysis. Helmet detection methodologies using visual data are also discussed.

CHAPTER 3: System Design describes the architecture and workflow of the proposed surveillance system. This chapter explains each module in detail: data collection, annotation for helmet detection, YOLO-based object detection, DeepSORT-based tracking, speed estimation using optical flow and perspective transformation, and the logic behind component integration. Design decisions, model configurations, and processing pipelines are elaborated.

CHAPTER 4: Implementation details the actual development process of the surveillance system. It covers the training and deployment of the YOLO model for vehicle and helmet detection, implementation of Good Features to Track for motion point selection, and the use of the Lucas-Kanade optical flow algorithm for estimating displacement between frames. The chapter also explains how DeepSORT is configured and optimized to maintain consistent object identities across video frames, facilitating speed estimation

CHAPTER 5: Evaluation and Results evaluates the performance of the system using quantitative metrics. For speed estimation, Mean Absolute Error (MAE) of 1.50 km/h and Root Mean Square Error (RMSE) of 1.84 km/h are calculated based on 30 recorded traffic videos with ground-truth speed from a speedometer. Helmet detection performance is assessed qualitatively. The results are presented with visualizations and discussed in terms of accuracy and system efficiency.

CHAPTER 6: Conclusion and Future Work summarizes the contributions of the research and the effectiveness of using video-only input for traffic surveillance. The chapter highlights the success of combining YOLO, DeepSORT, and optical flow methods for real-time, sensor-free analysis. It also suggests future directions such as expanding the system to include license plate recognition, integration with edge devices, and support for multi-camera setups in large-scale deployments.

CHAPTER 2

LITERATURE SURVEY

This chapter provides a concise review of existing research on vehicle speed estimation and traffic management using computer vision and machine learning. It focuses on the application of object detection models, such as YOLOv3, multi-object tracking algorithms like DeepSORT, and optical flow techniques for accurate speed measurement. The review also covers traffic monitoring systems, including vehicle classification and speed estimation frameworks, and explores challenges and advancements in monocular camera-based methods for real-time traffic analysis.

2.1 VIDEO-BASED VEHICLE SPEED ESTIMATION

Sangsuwan and Ekpanyapong [1] proposed an integrated vehicle speed estimation system that combines YOLOv3 for object detection, DeepSORT for multi-object tracking, and the Lucas-Kanade optical flow algorithm for analyzing object motion. They used virtual intrusion lines to compute speed based on the time it takes a vehicle to cross a known distance, and validated their system against a laser speed gun. The system achieved a Mean Absolute Error (MAE) of 3.38 km/h and a Root Mean Square Error (RMSE) of 4.69 km/h, while a supplementary neural network model using Multilayer Perceptron (MLP) improved results further to an MAE of 3.07 km/h and RMSE of 3.98 km/h. Their system processed 813 vehicles without the need for dedicated speed sensors, proving practical for real-time deployment in urban environments. The architecture of the proposed system is illustrated in Figure 2.1.

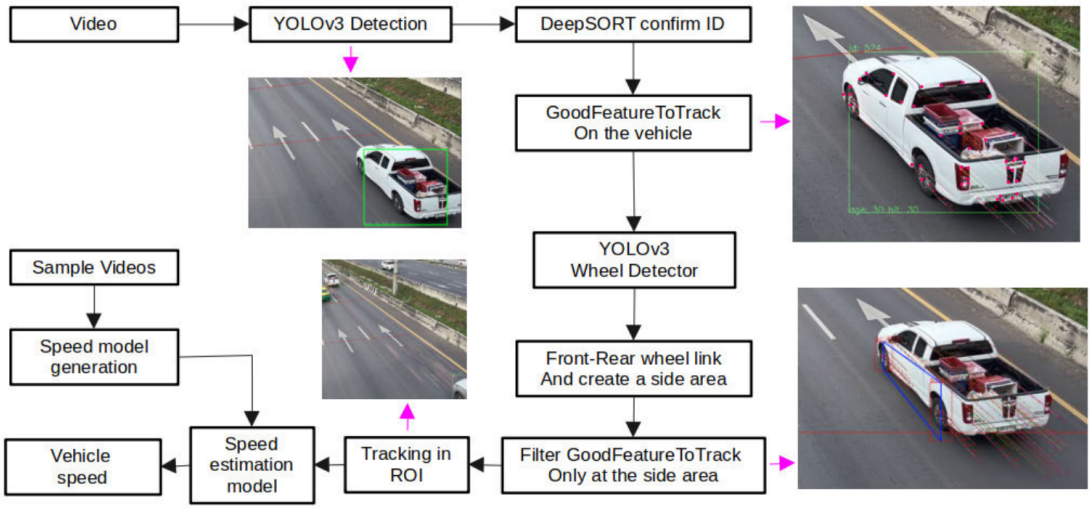

Figure 2.1: System Architecture Proposed by Sangsuwan and Ekpanyapong incorporating YOLOv3, DeepSORT, and Lucas-Kanade optical flow for speed estimation.

Wojke et al. [2] enhanced the SORT (Simple Online and Realtime Tracking) algorithm by introducing a deep association metric, giving rise to the DeepSORT algorithm. This model incorporates a pre-trained deep appearance descriptor that enables robust track association even during partial occlusions or visual ambiguities. The inclusion of this descriptor reduced identity switches by 45% compared to the original SORT, while retaining high processing speed necessary for real-time applications. The framework employs a Kalman Filter for motion prediction and uses cosine distance for visual matching, enabling reliable multi-object tracking in complex traffic scenes.

Bouguet [3] introduced the pyramidal implementation of the Lucas-Kanade optical flow method, which remains a cornerstone in motion estimation and tracking applications. The pyramidal approach allows for multi-resolution analysis, enhancing the algorithm's capability to track features across large displacements. It is computationally efficient and maintains robustness under varying lighting and noise conditions. This method forms the

basis for many later optical flow systems and has been widely integrated into real-time vehicle tracking modules due to its balance of speed and accuracy.

2.2 VEHICLES TRAFFIC MANAGEMENT

Kurdi [4] conducted an in-depth review of closed-circuit television (CCTV)-based traffic monitoring systems, underscoring their potential in real-time vehicle counting, speed measurement, and incident detection. The study evaluated both legacy and modern systems, concluding that video analytics can extract multiple traffic parameters without deploying additional hardware such as inductive loops or radar. Kurdi highlighted challenges such as occlusion handling and camera calibration inaccuracies but noted significant improvements in algorithmic accuracy with the adoption of AI-based methods. The paper also recommended enhancing video quality and leveraging cloud processing to enable large-scale urban traffic management.

Yabo et al. [5] presented a framework that performs both vehicle classification and speed estimation using feeds from monocular cameras. Their technique involved motion segmentation for detecting moving objects, followed by vehicle class recognition using shape and size heuristics. Speed was computed by measuring frame-by-frame displacement of the vehicle and mapping it to real-world distances using calibrated camera geometry. Their system operates in real-time and does not rely on dedicated roadside sensors, making it scalable for wide-area deployment. The authors validated their results against ground truth data, demonstrating its suitability for urban traffic monitoring and law enforcement use cases.

2.3 MONOCULAR CAMERA TECHNIQUES

Luvizon et al. [6] introduced a vehicle speed estimation system based solely on a single camera setup. Their approach utilizes perspective transformation to map image-plane coordinates to real-world distances, eliminating the need for stereo setups or GPS data. Object detection and motion estimation are conducted using video tracking algorithms, and vehicle speed is calculated using displacement over time in the corrected world plane. The system is cost-efficient and particularly suitable for developing countries with limited traffic infrastructure. Experiments showed accurate results with low deviation from ground truth across diverse lighting and traffic conditions.

Wu et al. [7] addressed the practical limitations of monocular cameras by proposing a speed estimation algorithm that uses object height estimation in combination with camera parameters to correct for scale. The system uses motion detection for object tracking and applies template matching techniques for frame-to-frame correlation. A critical enhancement in their work is the introduction of an adaptive height correction factor that dynamically adjusts speed estimates based on object dimensions. Evaluation against lidar-based ground truth showed that 95% of the speed estimates fell within a ±3% error range, marking a notable achievement for monocular systems.

Kumar et al. [8] proposed a semi-automated method that combines manual calibration using real-world markers with automated tracking algorithms for vehicle speed estimation from monocular videos. The pipeline involves homography estimation to derive the transformation between image and ground planes, followed by object detection and tracking using background subtraction techniques. The authors found that their hybrid method significantly improves accuracy in challenging environments such as curved roads and crowded intersections. Experiments on roadside videos demonstrated that their approach

maintained a low Mean Absolute Percentage Error (MAPE) while being easy to set up in new locations. The overall workflow of their system is illustrated in Figure 2.2.

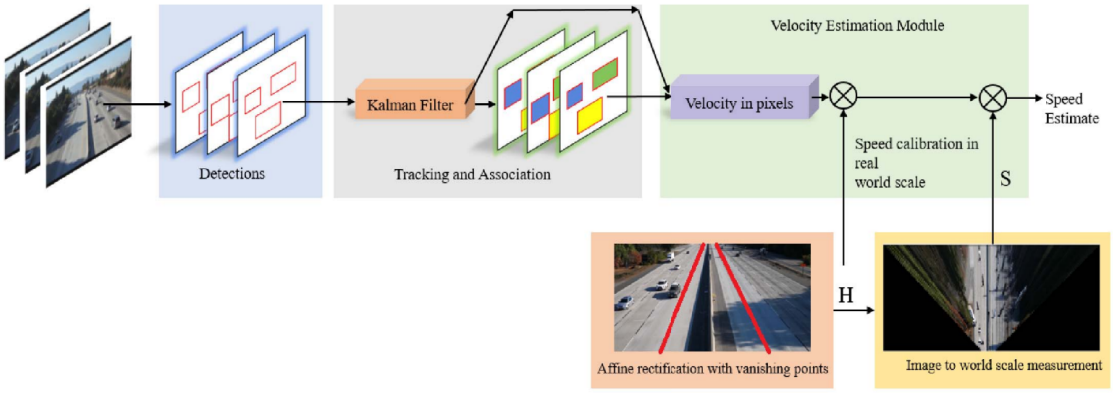

Figure 2.2: Pipeline of semi-automatic vehicle speed estimation using manual calibration and automated tracking.

2.4 ADVANCED ESTIMATION METHODS

Bell et al. [9] proposed a novel hybrid model that integrates deep learning-based feature extraction with traditional geometric modeling for vehicle speed estimation. Their approach uses a convolutional neural network (CNN) to detect vehicles and predict feature embeddings, which are then mapped onto real-world coordinates using a camera calibration matrix. The fusion of learning-based and model-based methods enables robust performance under diverse conditions, including varying lighting and partial occlusions. Extensive testing demonstrated that their system achieves higher accuracy than classical monocular methods, particularly in cluttered or occluded scenes.

Lin et al. [10] introduced the YOLOv8n-ASF-DH model in the context of safety helmet detection, but their architectural innovations have broader implications for vehicle tracking and speed estimation. The model

integrates Attentional Scale Fusion (ASF) to improve detection accuracy across different object scales and uses a Dynamic Head (DH) mechanism to adaptively weight feature importance. While focused on pedestrian safety applications, the model's improvements in detection speed and accuracy suggest that similar architectures could enhance vehicle detection in dense traffic scenes. These innovations contribute toward building more adaptive and accurate multi-object tracking systems, which are essential for high-precision speed estimation.

2.5 LIMITATIONS OF EXISTING SYSTEMS

Despite significant advancements in video-based vehicle speed estimation and traffic monitoring, several limitations persist across the reviewed systems. Many traditional methods rely on manual calibration or scene-specific tuning, such as those by Kumar et al. [8], which require physical markers and human intervention for homography estimation. This limits scalability and rapid deployment across different environments. Similarly, monocular camera techniques, while cost-effective, often suffer from accuracy degradation due to perspective distortions, occlusions, and lack of depth perception, as highlighted by Wu et al. [7].

Deep learning-based methods, like those proposed by Sangsuwan and Ekpanyapong [1], offer promising results but are heavily reliant on annotated training data and computational resources. Furthermore, real-time implementation on edge devices may be constrained due to the high processing demands of multi-stage pipelines involving YOLO, DeepSORT, and optical flow. In the context of advanced detection architectures such as YOLOv8n-ASF-DH discussed by Lin et al. [10], although attention mechanisms improve feature extraction, these models are still sensitive to environmental variations like lighting and weather conditions, impacting reliability.

Moreover, traffic management frameworks such as that by Kurdi [4] often lack integration with real-time feedback loops or adaptive controls, reducing their effectiveness in dynamically changing traffic scenarios. Many systems also fail to address long-term tracking consistency and identity switching, which remain significant challenges in congested environments. These limitations indicate the need for a more generalized, adaptive, and computationally efficient framework capable of operating robustly across diverse traffic scenes with minimal human calibration and infrastructure dependency.

2.6 SUMMARY

This chapter reviewed key developments in video-based vehicle speed estimation and traffic monitoring, covering both traditional computer vision methods and modern deep learning approaches. Early systems employed handcrafted techniques such as homography estimation, background subtraction, and optical flow, with reliable results in constrained scenarios using tools like Lucas-Kanade tracking. Recent research has shifted towards deep learning models like YOLO for object detection and DeepSORT for tracking, offering improved accuracy and real-time performance. Techniques such as perspective transformation, intrusion line analysis, and attention-based models have enabled accurate vehicle speed estimation from monocular video feeds with minimal hardware. These advancements form the foundation for the proposed system, which aims to combine the strengths of both paradigms to deliver a scalable, low-cost, and accurate solution for urban traffic environments.

CHAPTER 3

SYSTEM DESIGN

This chapter presents the system design of the proposed real-time surveillance system for vehicle speed estimation and helmet detection. It outlines the structural and functional components that make up the entire workflow. The design process focuses on creating a modular and efficient pipeline capable of operating solely on video input without requiring any external sensors.

3.1 SYSTEM ARCHITECTURE OVERVIEW

The system architecture diagram provides a high-level view of the key modules involved in the proposed real-time surveillance system. It illustrates the end-to-end workflow beginning with video input, followed by object detection, object tracking, helmet classification, and vehicle speed estimation. Each component in the pipeline is modular, ensuring real-time performance and ease of deployment using only a camera feed without any external sensors.

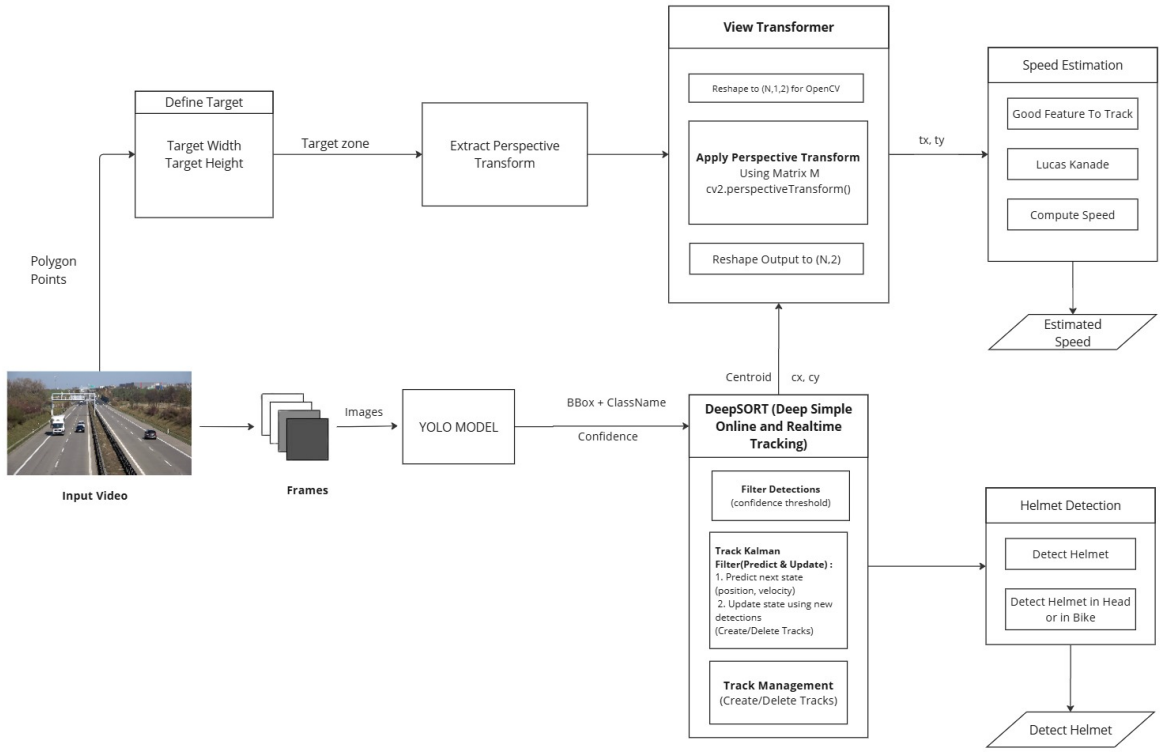

Figure 3.1: System Architecture of Vehicle Speed Estimation and Helmet Detection Framework

Figure 3.1 illustrates the architecture of the intelligent traffic surveillance system designed to detect helmet violations and estimate the speed of vehicles in road environments. The system starts with video input, which is typically captured from a stationary roadside or overhead camera. Each frame from the video is processed using a trained YOLO model, which performs object detection to identify vehicles and riders. Once detection is completed, the DeepSORT tracking algorithm is used to maintain consistent identities of detected objects across consecutive frames. This enables the system to track specific entities and associate movement patterns reliably.

For helmet compliance analysis, detected riders are classified as either helmeted or non-helmeted using the YOLO model fine-tuned on annotated helmet data. To estimate vehicle speed, the system applies the Good Features

to Track algorithm to extract stable keypoints from the vehicle's image region. These points are then tracked using the Lucas-Kanade optical flow method across successive frames. A perspective transformation is applied to convert pixel displacement into real-world distance, allowing accurate speed estimation based on the frame rate.

The results from both helmet detection and speed estimation are combined to provide a comprehensive understanding of traffic behavior. The modular architecture ensures that each component, whether for detection, tracking, classification, or speed estimation, can be optimized or updated independently without disrupting the overall data pipeline. This flexibility supports efficient deployment in varied traffic scenarios using only a standard camera setup.

3.2 DATA COLLECTION AND DATA PROCESSING

The effectiveness of the proposed surveillance system is strongly influenced by the quality and diversity of the datasets used for training and evaluation. This section outlines the data sources and preprocessing steps used for both the vehicle speed estimation and helmet detection components of the system. The collected data forms the basis for training object detection models, validating tracking accuracy, and measuring real-world performance metrics.

3.2.1 Speed Estimation Dataset

To build a reliable speed estimation module, a custom dataset was generated using 30 real-world traffic videos captured under various conditions. These videos were recorded using a fixed roadside camera setup in urban environments, including highways, narrow streets, and intersections. Each video

includes a clear view of moving vehicles captured at a consistent frame rate and resolution.

The camera is mounted on the helmet while driving, and the speed readings are recorded in the video through a call to generate ground truth speed data. The synchronized video and speedometer data allowed for accurate mapping of real-world speeds to visual motion in the footage. These videos also reflect variations in vehicle types, lighting conditions, and traffic density, enabling robust system evaluation across diverse scenarios. A sample frame from one of the recorded videos used for speed estimation is shown in Figure 3.2.

Figure 3.2: Sample Video Frame for Speed Estimation

3.2.2 Helmet Detection Dataset

For helmet detection, a pre-existing public dataset was used from Roboflow titled "Bike Helmet Detection"[1]. This dataset contains annotated images of motorcyclists under various real-world conditions and supports three distinct classes: helmet (regions showing helmets in general), with helmet (riders wearing helmets), and without helmet (riders without helmets).

The dataset consists of thousands of labeled images collected from different angles, backgrounds, and traffic situations, making it suitable for training deep learning models like YOLO. Preprocessing steps included resizing the images, normalizing the pixel values, and converting the annotations into YOLO-compatible format. The model was fine-tuned using transfer learning techniques to improve detection performance specific to helmet compliance enforcement.

3.3 USER INTERFACE MODULE

The user interface module forms the interaction layer between the end-user and the real-time surveillance system. It facilitates the submission of input data, configuration of processing parameters, and delivery of processed outputs through an integrated browser-based interface. This module is designed to support both asynchronous interactions for setup and configuration and continuous, low-latency communication for real-time video analysis.

The flow begins when the user interface is loaded into the browser, at which point initial configuration data and static assets are served from the backend. Users can upload a surveillance video file directly through the interface, which is then transmitted to the backend and queued into the analysis

[1] https://universe.roboflow.com/bike-helmets/bike-helmet-detection-2vdjo

pipeline. Configuration parameters for the processing modules—such as frame rate, detection thresholds, and tracking behavior—can be adjusted prior to execution. These settings are sent asynchronously from the frontend, ensuring seamless updates without interrupting the interface or data stream.

During processing, video frames augmented with detection overlays and structured metadata are streamed back to the frontend in real time. The system maintains a persistent, low-latency communication channel to synchronize the output with playback and support dynamic user interactions. The user module is designed for responsiveness and modularity, enabling live visualization, runtime reconfiguration, and uninterrupted monitoring. It integrates tightly with the backend analytics to deliver a coherent and efficient interaction framework for real-time surveillance tasks.

3.4 SUMMARY

This chapter presented the system design of the proposed real-time surveillance framework for vehicle speed estimation and helmet detection, emphasizing a modular architecture that operates solely on video input without external sensors. The system integrates YOLO for object and helmet detection, DeepSORT for tracking, and optical flow with perspective transformation for accurate speed estimation. It utilizes a custom dataset of traffic videos for speed analysis and a public Roboflow dataset for helmet classification, with appropriate preprocessing and model fine-tuning. Additionally, a user module was developed to enable browser-based interaction for video upload, parameter configuration, and real-time visualization, ensuring efficient and flexible deployment in varied traffic monitoring scenarios.

CHAPTER 4

IMPLEMENTATION

This chapter details the implementation of the real-time intelligent surveillance system, focusing on the integration of deep learning modules and the design of the user-facing web interface. The development pipeline encompasses various components, including object detection, multi-object tracking, speed estimation, and helmet classification. Each process from video frame acquisition and preprocessing to model inference and real-time visualization is discussed in a structured, step-by-step manner. The goal is to provide a comprehensive understanding of the system's architecture and the individual algorithms that contribute to its overall functionality.

4.1 OBJECT DETECTION

This module is responsible for detecting vehicles and riders in real-time using YOLOv11, the latest iteration in the YOLO object detection framework. YOLOv11 operates as a single-stage detector, performing both object classification and localization in a single pass, making it ideal for time-sensitive applications like traffic surveillance. The model introduces several enhancements over its predecessors, including multi-path attention mechanisms for better spatial understanding, dynamic anchor refinement for improved bounding box accuracy, and lightweight transformer-based detection heads for enhanced contextual reasoning [11].

Each video frame is resized and normalized before being passed into the YOLOv11 network, which outputs bounding boxes along with object class labels and confidence scores. The complete object detection process is formally

described in Algorithm 4.1. Post-processing is performed using Non-Maximum Suppression (NMS) to remove overlapping detections. The final output is a set of high-confidence detections for each frame, which are then passed to subsequent modules for tracking, speed estimation, and helmet compliance analysis. YOLOv11's balance of speed and accuracy makes it a robust backbone for real-time surveillance systems.

Algorithm 4.1 YOLOv11-Based Object Detection

Input: Raw input image frame $F \in \mathbb{R}^{H \times W \times 3}$
Output: Set of filtered object detections $D = \{(x, y, w, h, \hat{c}, \hat{s})\}$, where $\hat{c}$ is the predicted class and $\hat{s}$ the confidence score

procedure YOLOv11_ObjectDetection(F)
    $F' \leftarrow \text{resize}(F, 640 \times 640)$
    Normalize pixel values of F' to range $[0, 1]$
    Extract multi-scale feature maps from F' using YOLOv11 backbone
    for each detection head **do**
        Predict bounding box parameters (x, y, w, h)
        Predict objectness score $s \in [0, 1]$
        Predict class probability distribution $P = \{p_1, p_2, \ldots, p_k\}$
    end for
    Apply dynamic anchor refinement to bounding box proposals
    Fuse spatial features using multi-path attention blocks
    Aggregate outputs from detection heads
    Perform Non-Maximum Suppression (NMS) with IoU threshold θ
    return Filtered detections $D = \{(x_i, y_i, w_i, h_i, \hat{c}_i, \hat{s}_i)\}_{i=1}^{n}$
end procedure

The output of this module includes the class label, bounding box location, and a confidence score for each detected object. These detections serve as the input for subsequent modules in the system pipeline, including the multi-object tracking module (DeepSORT), speed estimation module, and helmet compliance classification. The efficiency and robustness of YOLOv11 make it a suitable detection engine for this end-to-end real-time surveillance system, providing the accuracy and responsiveness required for deployment in real-world traffic environments.

4.2 OBJECT TRACKING

The object tracking module is responsible for maintaining consistent identities of detected vehicles and riders across video frames. To accomplish this, the system uses DeepSORT, an enhancement of the Simple Online and Realtime Tracking (SORT) algorithm. DeepSORT integrates deep appearance features with motion-based predictions, making it robust in challenging scenarios such as occlusion or object proximity.

DeepSORT combines two key components: a Kalman filter for motion prediction and a deep neural network for extracting appearance embeddings. The complete tracking workflow is formally presented in Algorithm 4.2. The Kalman filter estimates the next state of each tracked object using its prior motion trajectory, while the appearance features provide a visual signature that helps distinguish between similar-looking objects.

Algorithm 4.2 DeepSORT Object Tracking

Input: Current frame detections $D_t = \{d_1, d_2, \ldots, d_n\}$, previous tracks T_{t-1}
Output: Updated object tracks T_t for current frame

procedure DeepSORT_Tracking(D_t, T_{t-1})
    Predict track positions $\hat{T}_t$ using Kalman filter
    for each detection $d_i \in D_t$ **do**
        Extract appearance feature $\hat{f}_i$
    end for
    Compute distance matrix $D_{\text{match}} = \text{ComputeDistance}(\hat{T}_t, \hat{f}_t)$
    Apply Hungarian algorithm for association
    for each matched pair (t_i, d_j) **do**
        Update track t_i with detection d_j
    end for
    Initialize new tracks for unmatched detections
    Remove tracks unmatched for a threshold duration
    return Updated tracks T_t
end procedure

For each frame, YOLOv11 provides detections in the form of bounding boxes, class labels, and confidence scores. These are passed to DeepSORT, where appearance features are extracted for each detection. A distance matrix is then computed between the predicted track positions and the new detection embeddings. The Hungarian algorithm is used to match detections to existing tracks based on this matrix. If no match is found, a new track is initialized; otherwise, the matched track is updated. Tracks are removed when they remain unmatched for a predefined number of frames.

4.3 SHI-TOMASI GOOD FEATURES TO TRACK

The Shi-Tomasi Good Features to Track algorithm Algorithm 4.3 is crucial for detecting stable and distinct points in an image that can be reliably tracked across video frames. The algorithm identifies corners where intensity varies significantly in multiple directions by calculating a cornerness measure based on the eigenvalues of the gradient matrix. These eigenvalues describe intensity variations, with higher values indicating more stable features that persist over time, making them ideal for tracking applications.

The selected features serve as anchor points for motion tracking, enabling the system to follow vehicle and rider movements even in challenging conditions with partial occlusions or lighting variations. The algorithm's robustness comes from its eigenvalue-based selection criterion, which prioritizes points with strong intensity variations in multiple directions while filtering out less distinctive features.

Algorithm 4.3 Shi-Tomasi Good Features to Track

Input: Single image frame $I \in \mathbb{R}^{H \times W}$ (grayscale or RGB)
Output: Set of top N corner feature points $\{(x_i, y_i)\}_{i=1}^{N}$

procedure ShiTomasi_GoodFeaturesToTrack(I)
    Convert image I to grayscale if not already
    Apply Gaussian smoothing to reduce noise
    Compute image gradients I_x and I_y
    Compute gradient products: $I_{xx} = I_x^2$, $I_{yy} = I_y^2$, $I_{xy} = I_x I_y$
    For each pixel, compute the structure tensor matrix:
    $M = \begin{bmatrix} \sum I_{xx} & \sum I_{xy} \\ \sum I_{xy} & \sum I_{yy} \end{bmatrix}$
    Calculate eigenvalues λ_1, λ_2 of M
    Compute cornerness score: $C = \min(\lambda_1, \lambda_2)$
    Select top N features based on C
    return Best N features $\{(x_i, y_i)\}_{i=1}^{N}$
end procedure

4.4 LUCAS-KANADE OPTICAL FLOW

Building upon the features detected by Shi-Tomasi, the Lucas-Kanade Optical Flow method Algorithm 4.4 estimates feature displacement between consecutive frames. This differential approach assumes small, consistent pixel motion and constant intensity over time, using first-order Taylor expansion to estimate motion vectors. The method combines spatial and temporal gradient information within local neighborhoods, making it particularly effective for tracking vehicle and rider motion.

The algorithm's window-based approach provides several advantages for traffic monitoring: it aggregates information from surrounding pixels to increase robustness, handles small appearance variations naturally, and maintains computational efficiency crucial for real-time operation. These characteristics make it ideal for estimating vehicle velocities and trajectories in surveillance applications where processing speed and accuracy are both critical requirements.

Algorithm 4.4 Lucas-Kanade Optical Flow

Input: Two consecutive grayscale image frames $I_1, I_2 \in \mathbb{R}^{H \times W}$
Output: Optical flow vectors $\mathbf{v} = (v_x, v_y)$ for tracked feature points

procedure LucasKanade_OpticalFlow(I_1, I_2)
    Compute spatial gradients: $I_x = \frac{\partial I}{\partial x}$, $I_y = \frac{\partial I}{\partial y}$
    Compute temporal gradient: $I_t = I_2 - I_1$
    for each point (x, y) in feature set **do**
        Define a window W around (x, y)
        Compute:
        $A = \begin{bmatrix} \sum_W I_x^2 & \sum_W I_x I_y \\ \sum_W I_x I_y & \sum_W I_y^2 \end{bmatrix}, b = \begin{bmatrix} -\sum_W I_x I_t \\ -\sum_W I_y I_t \end{bmatrix}$
        Solve $A\mathbf{v} = b$ for $\mathbf{v} = \begin{bmatrix} v_x \\ v_y \end{bmatrix}$
    end for
    return Flow vectors $\mathbf{v} = (v_x, v_y)$ for all tracked points
end procedure

4.5 USER INTERFACE MODULE IMPLEMENTATION

The user interface module facilitates interaction with the vehicle tracking system through three sequential stages, as illustrated in Figures 4.1, 4.2. The workflow begins with the upload and configuration interface, where users import traffic footage and define analysis parameters. This interface allows video file selection in multiple formats (MP4, MOV, AVI) and provides tools for spatial calibration, including manual region-of-interest (ROI) demarcation using four reference points and specification of real-world dimensions for accurate speed estimation. The interface also incorporates video rotation controls to correct orientation issues from misaligned camera installations.

Following configuration, the system processes the input through the detection and tracking pipeline, presenting results via the real-time monitoring dashboard Figure 4.1. This display shows the annotated video stream with overlaid vehicle bounding boxes, unique tracking IDs, and instantaneous speed

calculations. The visualization maintains color-coded indicators for different vehicle classes (private cars, motorcycles, commercial vehicles) and highlights potential traffic violations. Simultaneously, the dashboard's sidebar panel provides summary statistics including vehicle count, average speeds, and peak velocity recordings.

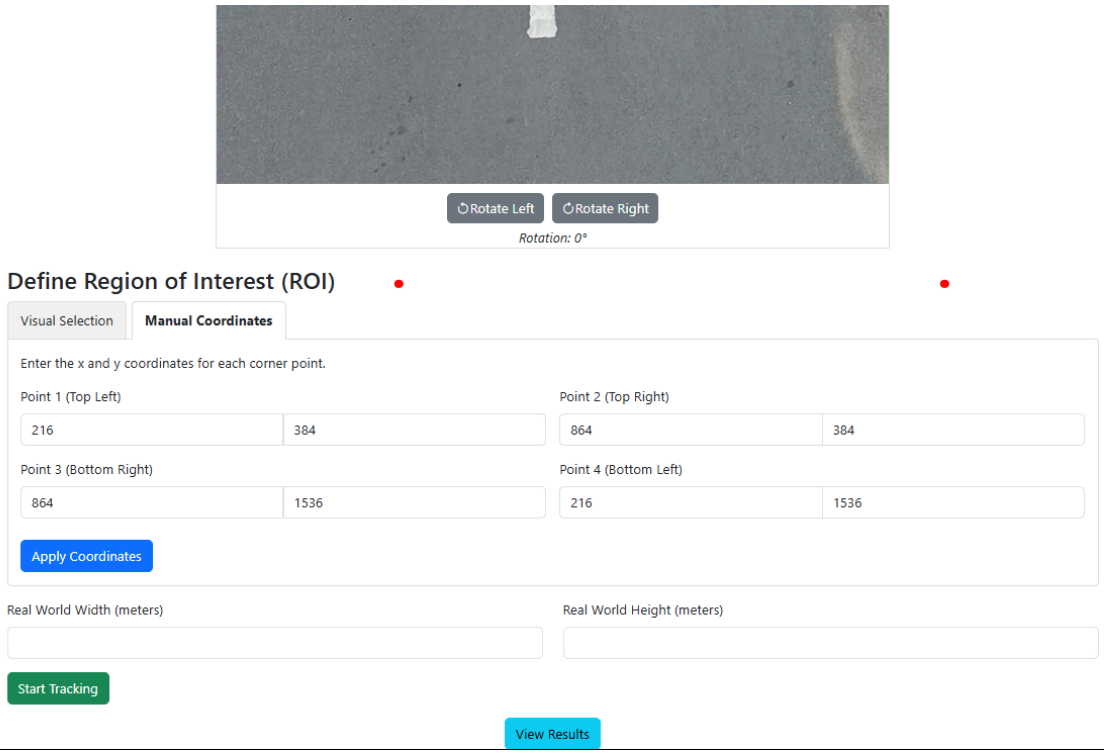

Figure 4.1: Input for interface: Polyline coordinates and frame dimensions defining the ROI for tracking and speed estimation.

The final output appears in the comprehensive results dashboard Figure 4.2, which aggregates processed data for operational use. This interface features a playback controller for reviewing tracking results, a gallery of captured vehicle images with timestamped speed records, and export functionality for evidentiary documentation. The dashboard's design emphasizes rapid identification of traffic violations through visual cues like speed threshold exceedance warnings and persistent tracking of individual vehicles across the surveillance area. Three distinct color schemes differentiate

vehicle categories while maintaining consistent ID labeling throughout the analysis session.For further details on the user interface implementation, refer to Section A.3 in Appendix A.

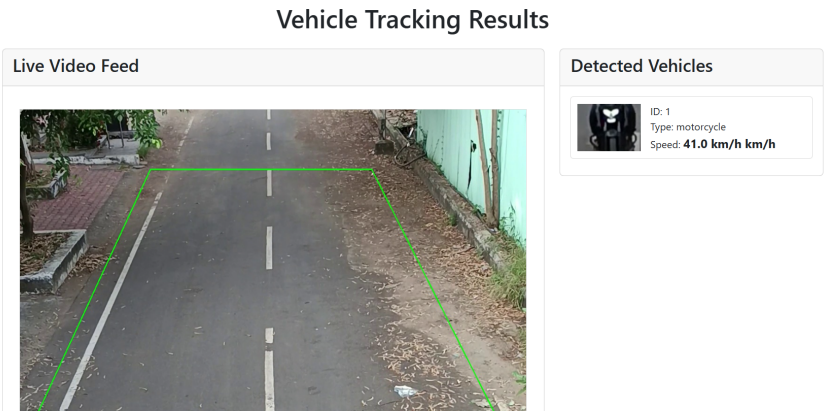

Figure 4.2: Analysis dashboard presenting vehicle tracking history, speed profiles, and evidence documentation features

4.6 SPEED ESTIMATION MODEL

The speed estimation module employs a vision-based pipeline to compute vehicle velocities from video, removing the need for physical sensors. Vehicles are first detected using the YOLO object detection model and consistently tracked across frames using DeepSORT. For each uniquely identified vehicle, a region of interest is defined and good feature points are extracted via the Shi-Tomasi corner detection algorithm. These points are tracked over time using the Lucas-Kanade optical flow method to estimate movement.

To translate pixel displacement into real-world speed, a manually defined perspective transformation is applied, mapping image coordinates to metric space. The centroid of tracked feature points in this transformed space is used to compute displacement over time. Final speed values, calculated in kilometers per hour from the average displacement and time interval, are

overlaid on the video above each vehicle. The complete workflow is detailed in Algorithm 4.5.

Algorithm 4.5 Vehicle Speed Estimation using Shi-Tomasi and Lucas-Kanade

procedure EstimateSpeed(Video Frames $F_1, F_2, ..., F_n$)
  for each frame F_t **do**
    Detect vehicles using YOLO: $B_t \leftarrow \text{YOLO}(F_t)$
    Track vehicles with DeepSORT: $ID_t \leftarrow \text{DeepSORT}(B_t)$
    for each detected vehicle v with ID i **do**
      if first appearance of v_i **then**
        Extract region of interest (ROI) around B_t^i
        Detect features: $F_0^i \leftarrow \text{Shi-Tomasi}(ROI)$
      else
        Track features using Lucas-Kanade Optical Flow: $F_t^i \leftarrow \text{LucasKanade}(F_{t-1}, F_t, F_{t-1}^i)$
        Compute centroid of features in real-world coordinates: $(x_t^i, y_t^i) \leftarrow \text{PerspectiveTransform}(F_t^i)$
        Store position and timestamp: $P_t^i \leftarrow (x_t^i, y_t^i, t)$
      end if
    end for
  end for
  for each vehicle v_i with trajectory $P_1^i, P_2^i, ..., P_k^i$ **do**
    Compute displacement: $d_i \leftarrow \sqrt{(x_k^i - x_1^i)^2 + (y_k^i - y_1^i)^2}$
    Compute time: $\Delta t \leftarrow t_k - t_1$
    Compute speed: $v_i \leftarrow \frac{d_i}{\Delta t} \times 3.6$ ▷ Convert m/s to km/h
  end for
end procedure

4.7 HELMET DETECTION MODULE

he helmet detection module aims to identify whether motorcycle riders are wearing helmets using a vision-based dual YOLO and DeepSORT tracking approach. Two YOLO models are employed: one trained for general object detection (detecting persons and motorcycles), and another specialized for helmet detection. DeepSORT is used to assign persistent identities to detected persons and motorcycles across video frames. In each frame, the module detects and tracks persons, motorcycles, and helmets, and then associates persons with nearby motorcycles based on Intersection-over-Union (IoU) overlap.

For each person-motorcycle pair, the algorithm checks if a detected helmet's centroid lies within the person's bounding box but not within the motorcycle's bounding box indicating that the person is wearing the helmet. If the helmet centroid is within both bounding boxes, it is marked as a helmet being carried on the motorcycle rather than worn. The system keeps track of riders and helmet usage per motorcycle, and visually annotates the video with this information in real time. The complete detection and association process is outlined in Algorithm 4.6.

Algorithm 4.6 Helmet Detection Using YOLO and DeepSORT

procedure DetectHelmet(Video Frames $F_1, F_2, ..., F_n$)
    Load general object detector (YOLO): detects persons and motorcycles
    Load helmet detector (YOLO): detects helmets
    Initialize DeepSORT trackers for persons and motorcycles
    for each frame F_t **do**
        $B_{p,t}, B_{m,t} \leftarrow \text{YOLO}_{\text{general}}(F_t)$ ▷ Persons, Motorcycles
        $B_{h,t} \leftarrow \text{YOLO}_{\text{helmet}}(F_t)$ ▷ Helmets
        $T_{p,t} \leftarrow \text{DeepSORT}(B_{p,t})$ ▷ Track persons
        $T_{m,t} \leftarrow \text{DeepSORT}(B_{m,t})$ ▷ Track motorcycles
        Compute centroids $C_{h,t}$ of helmet boxes $B_{h,t}$
        for each tracked person $P_t^i \in T_{p,t}$ **do**
            for each tracked motorcycle $M_t^j \in T_{m,t}$ **do**
                if $\text{IoU}(P_t^i, M_t^j) > \theta$ **then**
                    Associate P_t^i with M_t^j
                    Update rider count: $R_{M_t^j} \leftarrow R_{M_t^j} + 1$
                    for each helmet centroid $c_h \in C_{h,t}$ **do**
                        if $c_h \in P_t^i$ **and** $c_h \notin M_t^j$ **then**
                            Mark P_t^i as **Wearing Helmet**
                            $H_{M_t^j} \leftarrow H_{M_t^j} + 1$
                        else if $c_h \in P_t^i$ **and** $c_h \in M_t^j$ **then**
                            Mark P_t^i as **Helmet in Bike**
                        end if
                    end for
                end if
            end for
        end for
        Annotate F_t with bounding boxes and helmet status
        Save and display annotated frame
    end for
    Save output video
end procedure

4.8 SUMMARY

This chapter has presented the comprehensive implementation of the real-time intelligent surveillance system, detailing each core module from object detection using YOLOv11 and multi-object tracking with DeepSORT, to feature-based motion estimation using Shi-Tomasi and Lucas-Kanade algorithms, and concluding with specialized modules for speed estimation and helmet detection. The system architecture successfully integrates these components with an intuitive user interface, demonstrating a complete pipeline capable of accurate vehicle monitoring, speed calculation, and safety compliance verification in diverse traffic scenarios. Through careful algorithm selection and optimization, the implementation achieves the necessary balance between computational efficiency and detection accuracy required for real-world deployment in traffic management applications.

CHAPTER 5

RESULTS AND ANALYSIS

This chapter presents the results and analysis of the proposed real-time surveillance system for vehicle speed estimation and helmet detection using computer vision techniques. The system was evaluated using traffic videos captured in real-world driving conditions, with ground truth speed obtained from vehicle speedometers to assess the accuracy of the speed estimation module. The helmet detection component was validated using a pre-annotated public dataset, enabling the trained YOLO model to distinguish between riders with and without helmets. The analysis includes performance metrics such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Mean Absolute Percentage Error (MAPE) for speed prediction, providing comprehensive quantitative measures of accuracy. The system's output was further verified through visual results, showcasing the effectiveness of the integrated modules in detecting violations and tracking vehicles under varied environmental conditions. Overall, the results demonstrate that the system performs reliably in identifying helmet compliance and estimating vehicle speed from monocular video.

5.1 SPEED ESTIMATION MODULE

This section details the evaluation of the speed estimation module, a critical component of the proposed surveillance system. The module is designed to estimate the speed of moving vehicles directly from video input, without relying on external hardware sensors. The evaluation includes an overview of the dataset used and a quantitative performance analysis, comparing the predicted vehicle speeds with the recorded ground truth values.

Figure 5.1 illustrates the detection and tracking process of a bike, where the system assigns a unique ID to the vehicle, selects distinctive features for tracking, and applies the Lucas-Kanade method for motion estimation. This process is key to ensuring accurate speed predictions by reliably following the object's trajectory in the video stream.

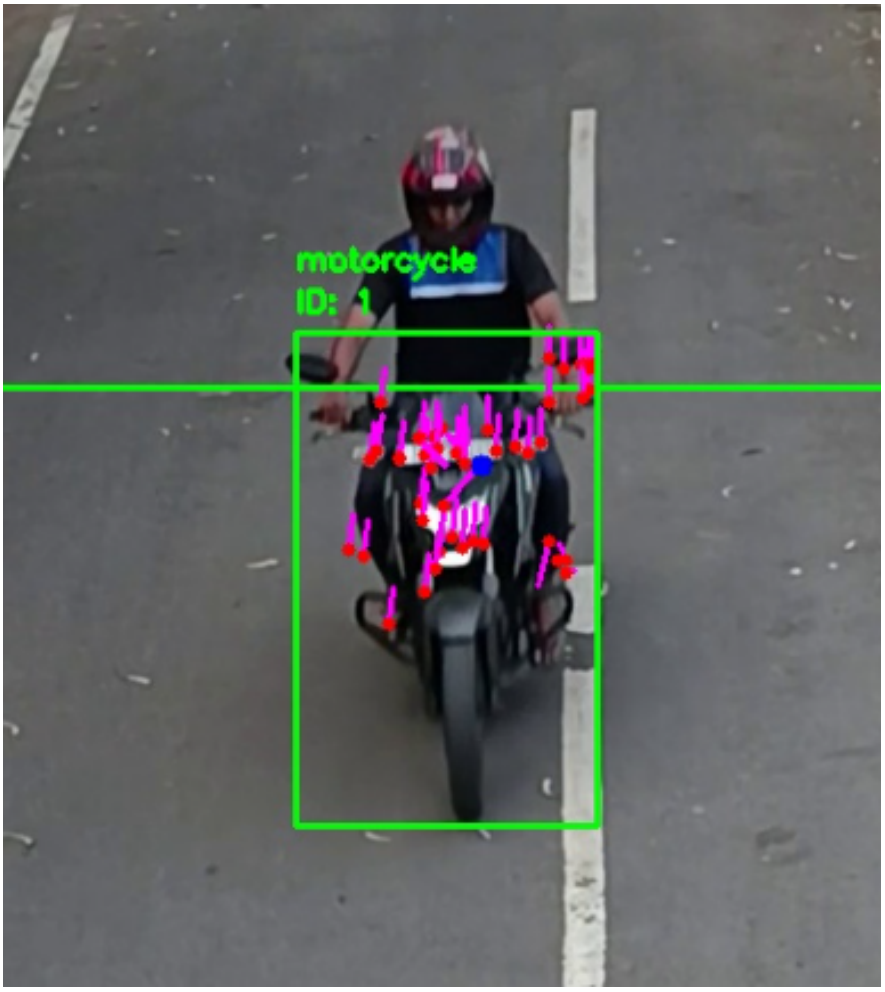

Figure 5.1: Detection and tracking of a moving vehicle, including ID assignment, feature selection, and application of the Lucas-Kanade optical flow method for motion estimation.

5.1.1 Dataset

The dataset utilized for evaluating the speed estimation module consists of 30 videos recorded under real-world traffic conditions. All recordings

were captured using a Redmi Note 13 Pro smartphone camera with a resolution of 1080p and a frame rate of 30 frames per second. The dataset primarily includes videos of motorcycles traveling on urban roads.

To obtain accurate ground truth speed values, the speed of each motorcycle was recorded using its digital odometer. Simultaneously, the rider verbally communicated the current speed during a phone call, which was captured in the audio of the video. This dual-recording approach provided a reliable verification mechanism for determining the actual speed during post-processing, improving the validity of the dataset used in the evaluation.

5.1.2 Evaluation

The speed estimation module was tested across a variety of speed levels ranging from 15 km/h to 63 km/h. The module's performance was assessed by comparing the estimated speeds to the actual recorded speeds, using standard error metrics including Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Mean Absolute Percentage Error (MAPE). These metrics provide insights into both the average magnitude and the percentage deviation of the predicted values from the actual ground truth.

The detailed results of the evaluation are presented in Table 5.1, which includes actual and estimated speeds, along with the corresponding absolute errors, squared errors, and MAPE values for each observation.

Table 5.1: Performance of the speed estimation module based on actual and estimated speeds with associated error metrics.

Actual Speed (km/h)	Estimated Speed (km/h)	Absolute Error (km/h)	Squared Error (km^2/h^2)	MAPE (%)
15	15.10	0.10	0.0100	0.67
20	20.20	0.20	0.0400	1.00
25	25.80	0.80	0.6400	3.20
30	28.12	1.88	3.5344	6.27
35	34.48	0.52	0.2704	1.49
40	42.50	2.50	6.2500	6.25
45	46.10	1.10	1.2100	2.44
50	52.20	2.20	4.8400	4.40
55	58.20	3.20	10.2400	5.82
60	62.69	2.69	7.2361	4.48
63	61.69	1.31	1.7161	2.08

From the data in Table 5.1, the speed estimation module achieved a Mean Absolute Error (MAE) of 1.50 km/h, indicating the average deviation of estimated speeds from the actual values. The Root Mean Square Error (RMSE) was calculated to be 1.84 km/h, reflecting the typical magnitude of error with increased sensitivity to larger deviations. Furthermore, the Mean Absolute Percentage Error (MAPE) was determined to be 3.37%, which quantifies the relative size of the errors in percentage terms. Interpreting this value as a measure of model accuracy, the system achieved an overall accuracy of 96.63%. For a detailed explanation of the evaluation metrics used, please refer to Section A.1 in Appendix A.

5.2 HELMET DETECTION MODULE

The helmet detection module is a core component of the proposed surveillance system, designed to automatically identify helmet usage among two-wheeler riders. Unlike basic detection systems that only check for helmet presence, this module is capable of distinguishing between riders who are wearing helmets correctly, riders who are not wearing helmets, and those who

may be carrying helmets improperly (e.g., hanging on the handle or placed on the bike). This fine-grained classification enables more effective monitoring of helmet compliance in real-world traffic environments. An example of the helmet detection output is shown in Figure 5.2, where bounding boxes are drawn around detected riders with corresponding class labels.

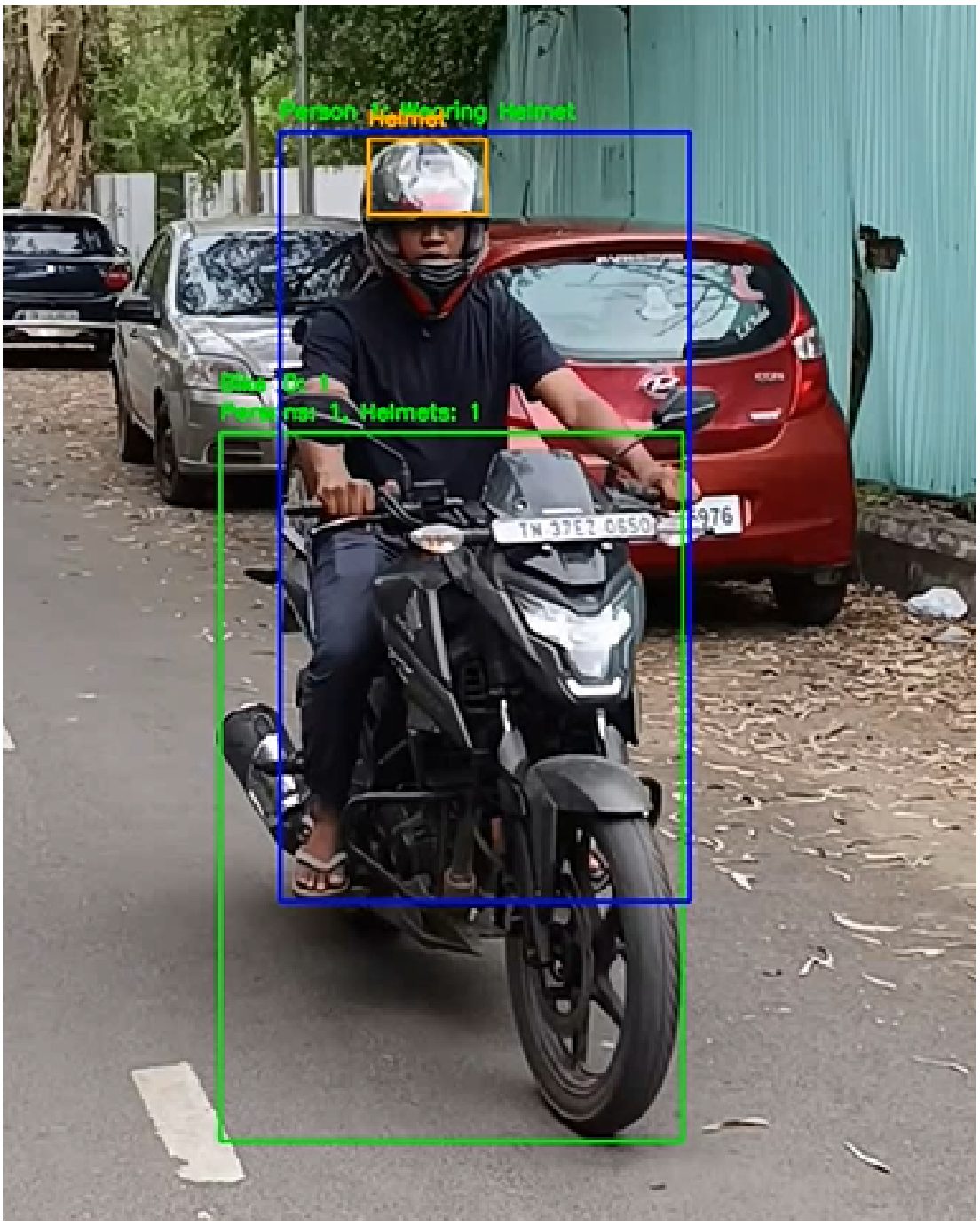

Figure 5.2: Example of Helmet Detection Output with Bounding Boxes

5.2.1 Dataset

To train the model, a publicly available dataset titled *Bike Helmet Detection* was utilized, sourced from the Roboflow platform. The dataset consists of annotated images labeled into three classes: helmet, with helmet, and without helmet. This categorization allows the model to learn detailed visual distinctions in helmet usage behavior. The YOLO (You Only Look Once) object detection architecture was used for this task, with training performed using a transfer learning approach. Preprocessing involved standard augmentation techniques to improve robustness under varying lighting, weather conditions, and rider poses. For further details, visit the dataset at [1].

Once trained, the model demonstrated strong performance in detecting helmet status across various traffic scenes. It successfully identifies riders even in crowded or occluded scenarios and distinguishes between riders wearing helmets and those who have merely carried them without wearing.

5.2.2 Evaluation

The trained YOLO model was evaluated on a separate validation set to measure detection accuracy. The model's performance was assessed using common object detection metrics, including mean Average Precision at Intersection over Union (IoU) 0.5 (mAP@50), Precision, and Recall. These metrics reflect the model's accuracy in localization (bounding box quality), class prediction confidence, and ability to detect all relevant instances, respectively.

[1] `https://universe.roboflow.com/bike-helmets/bike-helmet-detection-2vdjo`

Table 5.2: Helmet Detection Evaluation Metrics

Metric	Value
mAP@50	91.5%
Precision	65.1%
Recall	92.8%

The results of the evaluation are presented in Table 5.2. The mAP@50 value of 91.5% indicates high accuracy in detecting riders and minimizing overlap errors between predicted and ground-truth bounding boxes. Precision, at 65.1%, shows moderate effectiveness in predicting helmet presence, but there is room to reduce false positives. The Recall of 92.8% highlights the model's strength in identifying most relevant cases, even in challenging environments. Detailed information on the evaluation metrics is provided in Section A.2 of Appendix A.

5.3 SUMMARY

This chapter presented the results and analysis of the proposed real-time surveillance system designed for vehicle speed estimation and helmet detection. The performance of the speed estimation module was evaluated using real-world data, achieving accurate predictions with low error margins, demonstrating the effectiveness of the system in estimating vehicle speeds. Similarly, the helmet detection module, utilizing an advanced object detection framework, showed high accuracy in identifying helmet usage, with strong performance metrics. These results highlight the reliability and effectiveness of the system in real-time monitoring and enforcement, providing a comprehensive solution for traffic safety applications.

CHAPTER 6

CONCLUSION AND FUTURE WORK

This chapter presents the conclusion of the study and outlines potential future research directions.

6.1 CONCLUSION

This project proposes a real-time, camera-based intelligent surveillance system to improve road safety by estimating vehicle speed and detecting helmet usage without relying on intrusive or costly physical sensors. The integrated framework combines several computer vision and deep learning techniques, including YOLO for object detection, DeepSORT for multi-object tracking, the Shi-Tomasi and Lucas-Kanade optical flow for motion estimation, and perspective transformation to translate pixel displacement into real-world speed measurements. Key contributions include the development of a cost-effective speed estimation module achieving a Mean Absolute Error (MAE) of 1.50 km/h, a Root Mean Square Error (RMSE) of 1.84 km/h, and a Mean Absolute Percentage Error (MAPE) of 3.37

6.2 FUTURE WORK

Future research will aim to enhance the system's adaptability and performance under challenging real-world conditions. This includes improving detection and tracking under low light and occlusion using infrared imaging or low-light-adapted YOLO models. Further, the speed estimation module could be generalized to operate without reliance on lane markings, using techniques

like ground plane estimation or SLAM. For helmet compliance, future efforts may focus on verifying whether the helmet is properly worn rather than merely present, possibly through 3D pose estimation or head-helmet alignment analysis. To support dense, multi-lane urban traffic, enhancements in vehicle clustering and localized tracking will be necessary. Moreover, optimizing the entire pipeline for edge deployment on platforms like NVIDIA Jetson or Raspberry Pi will enable real-time, decentralized operation. The system could also be extended to support traffic anomaly detection such as wrong-way driving, triple riding, or overspeeding, with instant alerts. Finally, expanding the annotated dataset to include variations in lighting, weather, and camera angles will allow for more robust model training via transfer learning strategies.

6.3 SUMMARY

This chapter provided a comprehensive conclusion to the study, highlighting the development and evaluation of a real-time, camera-based intelligent surveillance system for vehicle speed estimation and helmet detection. The system successfully integrates advanced computer vision and deep learning techniques to deliver accurate, non-intrusive monitoring across varied traffic scenarios. Additionally, future research directions were outlined, focusing on improving system robustness, expanding capabilities to handle complex traffic conditions, and enabling deployment on low-power edge devices. These advancements will further contribute to enhancing road safety and enabling scalable, intelligent traffic surveillance solutions.

APPENDIX A

EVALUATION METRICS AND USER INTERFACE

A.1 SPEED ESTIMATION EVALUATION METRICS

The speed estimation module is evaluated using several performance metrics, focusing on the accuracy of speed predictions when compared to ground truth values. The key evaluation metrics include:

1. Mean Absolute Error (MAE)

The MAE measures the average magnitude of errors in the speed predictions, without considering their direction. It is calculated as shown in Equation 1:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|v_{\text{pred},i} - v_{\text{gt},i}| \tag{1}$$

where:

- $v_{\text{pred},i}$ is the predicted speed of the i-th vehicle.
- $v_{\text{gt},i}$ is the ground truth speed of the i-th vehicle.
- N is the total number of vehicles in the dataset.

2. Root Mean Squared Error (RMSE)

RMSE is used to evaluate the differences between predicted and actual speeds, with greater emphasis on larger errors. It is defined in Equation 2

as:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(v_{\text{pred},i} - v_{\text{gt},i})^2} \tag{2}$$

where the terms are as defined above.

3. Mean Absolute Percentage Error (MAPE)

MAPE measures the accuracy of speed predictions as a percentage error. It is defined in Equation 3 as:

$$\text{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{v_{\text{pred},i} - v_{\text{gt},i}}{v_{\text{gt},i}}\right| \times 100 \tag{3}$$

where:

- $v_{\text{pred},i}$ is the predicted speed of the i-th vehicle.
- $v_{\text{gt},i}$ is the ground truth speed of the i-th vehicle.
- N is the total number of vehicles in the dataset.

4. Accuracy

The accuracy metric is calculated as shown in Equation 4:

$$\text{Accuracy} = 1 - \text{MAPE} \tag{4}$$

where:

- MAPE is the Mean Absolute Percentage Error, as described above in Equation 3.

A.1.1 Overview of Speed Estimation Evaluation

The speed estimation module is evaluated on a dataset of moving vehicles, comparing the predicted speeds to the ground truth values. The key focus is on ensuring accurate tracking and motion estimation through the use of optical flow techniques and robust feature tracking, as described in the main section of this report.

5. Speed Estimation Formulas

To estimate the speed of a vehicle, the following formula shown in Equation 5 is used based on the displacement of the vehicle in pixels and the real-world distance:

$$\text{Speed (km/h)} = \frac{\text{Total Distance (m)}}{\text{Time (s)}} \times 3.6 \tag{5}$$

where:

- Total Distance is the sum of the Euclidean distances between consecutive vehicle positions.
- Time is the difference between timestamps of the recorded positions.

6. Perspective Transformation

To convert the 2D pixel coordinates of the vehicle to real-world coordinates, a perspective transformation is applied. This transformation is crucial for converting the measurements from the image plane to real-world distances. The transformation matrix M is computed using the region of interest (ROI) coordinates and the corresponding real-world coordinates as shown in

Equation 6:

$$M = \text{cv2.getPerspectiveTransform}(\text{roi\_coords}, \text{dst\_points}) \quad (6)$$

Once the transformation matrix is obtained, any point in the image can be transformed to real-world coordinates as shown in Equation 7:

$$\text{Real World Coordinates} = \text{cv2.perspectiveTransform}(\text{point}, M) \quad (7)$$

Using the transformation defined in Equation 6 and Equation 7, we can accurately convert image coordinates to real-world measurements for speed calculation as shown in Equation 5.

A.2 YOLO MODEL TRAINING AND EVALUATION METRICS

The YOLO (You Only Look Once) model is used for object detection, and its performance is evaluated using standard metrics in object detection tasks. These metrics assess the accuracy of the model's ability to detect and localize objects in the image frame.

1. Intersection over Union (IoU)

IoU is a metric used to evaluate the accuracy of object localization. It is defined in Equation 8 as the ratio of the intersection area of the predicted bounding box and the ground truth bounding box to the area of their union:

$$\text{IoU} = \frac{\text{Area of Intersection}}{\text{Area of Union}} \quad (8)$$

2. Precision and Recall

Precision and recall are used to evaluate the performance of the YOLO model in detecting objects:

- **Precision:** The ratio of true positive detections to the total number of detections made by the model as defined in Equation 9:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{9}$$

  where TP is the number of true positives and FP is the number of false positives.

- **Recall:** The ratio of true positive detections to the total number of actual objects (ground truth) as shown in Equation 10:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{10}$$

  where FN is the number of false negatives.

3. Mean Average Precision (mAP)

mAP is a key evaluation metric in object detection, especially in multi-class detection tasks. It measures the precision at various recall levels and then averages the result across all classes. The formula for mAP is given in Equation 11:

$$mAP = \frac{1}{N} \sum_{c=1}^{N} \text{AP}_c \tag{11}$$

where N is the number of object classes, and AP_c is the average precision for class c.

4. F1-Score

The F1-score is the harmonic mean of precision and recall and is particularly useful when the class distribution is imbalanced. It is calculated as shown in Equation 12:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{12}$$

A.3 USER INTERFACE

A.3.1 Helmet Detection

The user interface for the helmet detection module was developed using Flask as the backend framework and HTML/CSS for frontend rendering and styling. This web-based application enables users to upload traffic surveillance videos for analysis and provides real-time feedback on helmet compliance. The system is designed to be user-friendly and responsive, allowing seamless interaction for both technical and non-technical users.

Initially, the interface presents an input page where users can upload videos by dragging and dropping files into a designated area. This page provides visual cues and guides to ensure the upload process is intuitive. An example of the input interface is shown in Figure A.1.

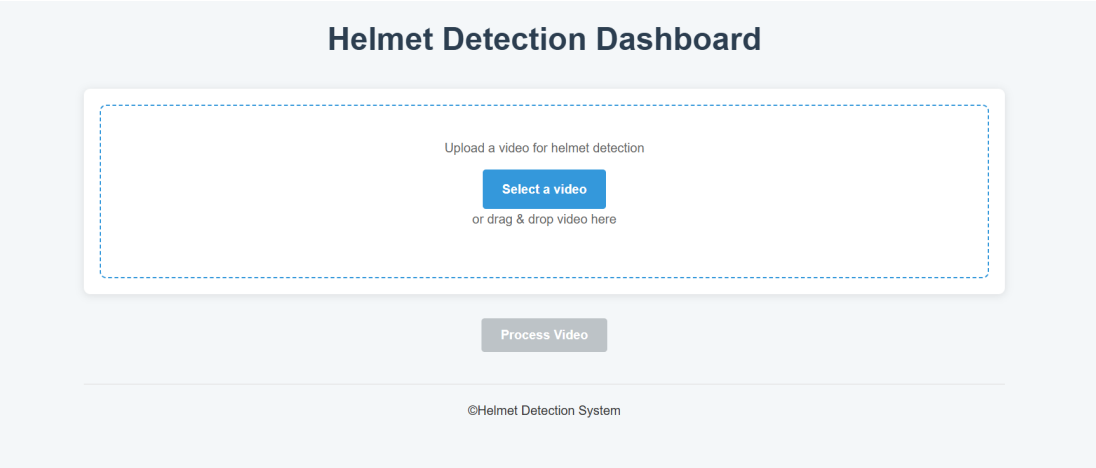

Figure A.1: Helmet detection input page with drag-and-drop video upload

Once a video is uploaded, the system begins processing the video frames in real time. Detected riders are highlighted using bounding boxes, and class labels such as "with helmet" or "without helmet" are overlaid to indicate compliance. These results are streamed back to the browser with minimal latency, allowing live monitoring as the video is analyzed. A snapshot of the real-time detection interface is shown in Figure A.2.

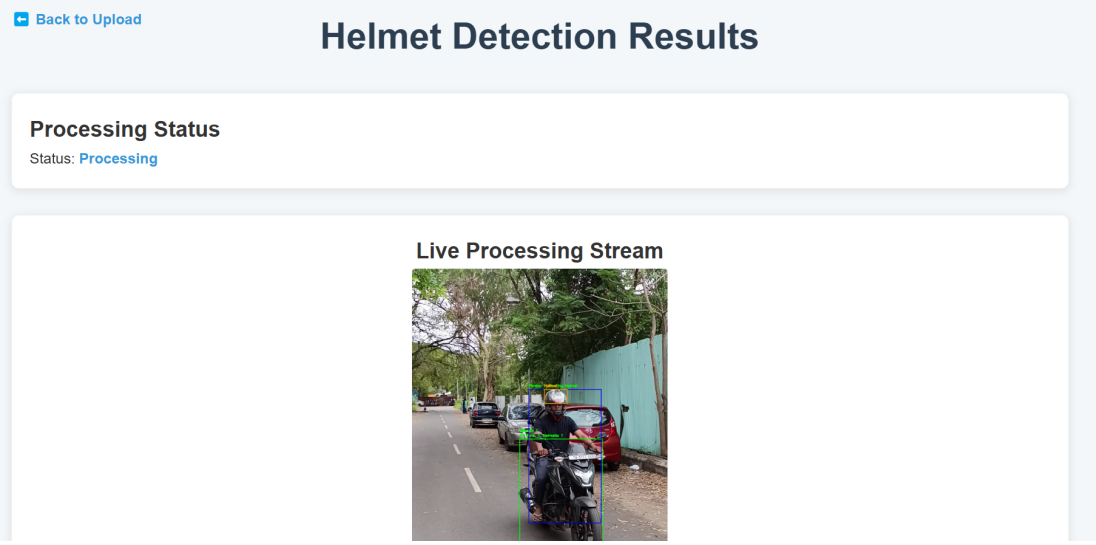

Figure A.2: Real-time helmet detection view with bounding boxes and class labels

In addition to real-time visual feedback, the interface includes a statistical summary panel that aggregates detection results. It presents key metrics such as vehicle IDs, total number of detected persons, and the number of helmeted riders. This feature supports enforcement and analytics by providing structured insights in tabular format. The statistics view is illustrated in Figure A.3.

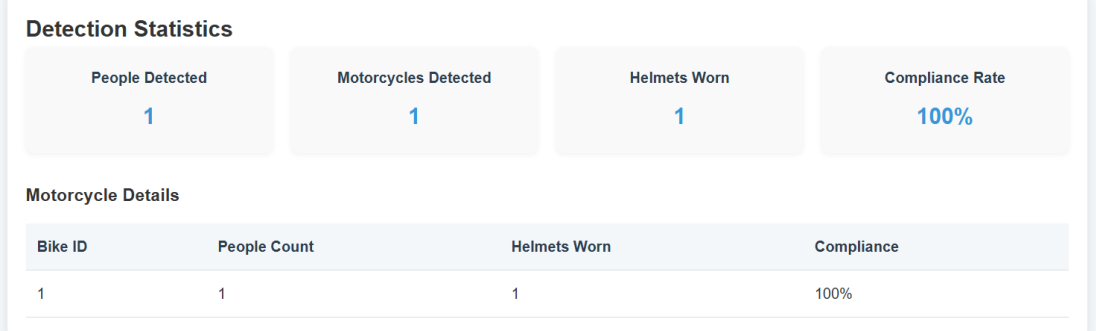

Figure A.3: Helmet compliance statistics displayed by vehicle ID and rider helmet status

REFERENCES

[1] Keattisak Sangsuwan and Mongkol Ekpanyapong. Video-based vehicle speed estimation using speed measurement metrics. *IEEE Access*, 12:4845–4858, 2024.

[2] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649, Beijing, China, 2017.

[3] Jean-Yves Bouguet. Pyramidal implementation of the lucas kanade feature tracker description of the algorithm. Technical report, Intel Corporation, Microprocessor Research Lab, 2000. Available online: `http://robots.stanford.edu/cs223b04/algo_affine_tracking.pdf`.

[4] H. A. Kurdi. Review of closed circuit television (cctv) techniques for vehicles traffic management. *International Journal of Computer Science and Information Technology*, 6(2):199–206, 2014.

[5] A. G. Yabo, S. I. Arroyo, F. Safar, and D. Oliva. Vehicle classification and speed estimation using computer vision techniques. In *Conference of the Argentina Association of Control and Automation (AADECA)*, pages 1–6, Buenos Aires, Argentina, 2016. Available online: `https://core.ac.uk/download/pdf/296389522.pdf`.

[6] Diogo C. Luvizon, Bruno T. Nassu, and Rodrigo Minetto. A video-based system for vehicle speed measurement in urban roadways. *IEEE Transactions on Intelligent Transportation Systems*, 18(6):1393–1404, 2017.

[7] W. Wu, V. Kozitsky, M. E. Hoover, R. Loce, and D. M. T. Jackson. Vehicle speed estimation using a monocular camera. In *Proceedings of SPIE*, volume 9407, pages 17–30, 2015.

[8] A. Kumar, P. Khorramshahi, W.-A. Lin, P. Dhar, J.-C. Chen, and R. Chellappa. A semi-automatic 2d solution for vehicle speed estimation from monocular videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 137–1377, Salt Lake City, UT, USA, 2018.

[9] D. Bell, W. Xiao, and P. James. Accurate vehicle speed estimation from monocular camera footage. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, V-2-2020:419–426, 2020.

[10] Bingyan Lin. Yolov8n-asf-dh: An enhanced safety helmet detection method. *IEEE Access*, 12:126313–126328, 2024.

[11] Rahima Khanam and Muhammad Hussain. Yolov11: An overview of the key architectural enhancements. *arXiv preprint arXiv:2410.17725*, 2024. Accessed: 2025-05-02.

[12] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[13] S. Javadi, M. Dahl, and M. I. Pettersson. Vehicle speed measurement model for video-based systems. *Computers & Electrical Engineering*, 76:238–248, June 2019.

[14] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, and R. Qu. A survey of deep learning-based object detection. *IEEE Access*, 7:128837–128868, 2019.

[15] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *arXiv preprint arXiv:1708.02002*, 2017.

[16] J. Xie, Y. Zheng, R. Du, W. Xiong, Y. Cao, Z. Ma, D. Cao, and J. Guo. Deep learning-based computer vision for surveillance in its: Evaluation of state-of-the-art methods. *IEEE Transactions on Vehicular Technology*, 70(4):3027–3042, April 2021.