A Project Report on

Integrated Traffic Monitoring System using YOLO For Speed Estimation and Vehicle Counting

A Dissertation submitted to JNTU Hyderabad in partial fulfillment of the academic requirements for the award of the degree.

Bachelor of Technology

In

COMPUTER SCIENCE AND ENGINEERING

Submitted by

A. Nithin Kumar Reddy
(20H51A0504)

Balaji Bhandare
(20H51A0531)

Gangula Sindhu
(20H51A0592)

Under the esteemed guidance of

Ms. A. Mounika Rajeswari
(Assistant Professor)

Department of Computer Science and Engineering

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(UGC Autonomous)

*Approved by AICTE *Affiliated to JNTUH *NAAC Accredited with A^+ Grade

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD - 501401.

2020 - 2024

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD – 501401

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the Major Project report entitled "Integrated Traffic Monitoring System using YOLO for Speed Estimation and Vehicle Counting" being submitted by A. Nithin Kumar Reddy (20H51A0504), Balaji Bhandare (20H51A0531), Gangula Sindhu (20H51A0592) in partial fulfillment for the award of Bachelor of Technology in COMPUTER SCIENCE AND ENGINEERING is a record of bonafide work carried out under my guidance and supervision.

The results embodies in this project report have not been submitted to any other University or Institute for the award of any Degree.

Ms. A. Mounika Rajeswari	**Dr.S.Siva Skandha**	
Assistant Professor	**Associate Professor and HOD**	**EXTERNAL EXAMINER**
Dept. Of CSE	**Dept. of CSE**	

ACKNOWLEDGEMENT

With great pleasure we want to take this opportunity to express our heartfelt gratitude to all the people who helped in making this project a grand success.

We are grateful to **Ms. A. Mounika Rajeswari, Assistant Professor**, Department of CSE for her valuable technical suggestions and guidance during the execution of this project work.

We would like to thank**, Dr. Siva Skandha Sanagala,** Head of the Department of CSE, CMR College of Engineering and Technology, who is the major driving forces to complete our project work successfully.

We are very grateful to **Dr. Ghanta Devadasu**, Dean-Academics, CMR College of Engineering and Technology, for his constant support and motivation in carrying out the project work successfully.

We are highly indebted to **Major Dr. V A Narayana,** Principal, CMR College of Engineering and Technology, for giving permission to carry out this project in a successful and fruitful way.

We would like to thank the **Teaching & Non- teaching** staff of Department of Dept Name for their co-operation

We express our sincere thanks to **Shri. Ch. Gopal Reddy**, Secretary& Correspondent, CMR Group of Institutions, and Shri Ch Abhinav Reddy, CEO, CMR Group of Institutions for their continuous care and support.

Finally, we extend thanks to our parents who stood behind us at different stages of this Project. We sincerely acknowledge and thank all those who gave support directly or indirectly in completion of this project work.

A. Nithin Kumar Reddy	20H51A0504
Balaji Bhandare	20H51A0531
Gangula Sindhu	20H51A0592

List of Figures

ABSTRACT

Modern urban environments face complex challenges in traffic management and safety. The primary aim of this project is to develop an intelligent and efficient traffic monitoring system capable of enhancing road safety, enforcing traffic regulations, and providing valuable insights for traffic management authorities. This project introduces a comprehensive traffic monitoring and management system that leverages the YOLO (You Only Look Once) object detection algorithm. The system addresses critical aspects of traffic control including speed estimation and vehicle counting within a single integrated pipeline. Speed estimation refers to the process of calculating the velocity of vehicles in real-time as they traverse a monitored area. Speed estimation, a cornerstone of traffic monitoring, is executed using advanced optical flow and tracking techniques empowered by YOLO. The system calculates vehicle speeds in real time, aiding in the identification of speed violations and the optimization of traffic flow. Realtime vehicle counting using YOLO is seamlessly integrated into the pipeline. The vehicle counts include incoming and outgoing provide valuable insights for traffic authorities, aiding in informed decisions about road capacity and congestion management.

Our system aggregates data from diverse sources, including cameras and other relevant data streams. This integration enables a comprehensive view of traffic conditions, allowing for better-informed decision-making by traffic management authorities. Real-time Processing: Operating in real-time, our system provides immediate information, enabling rapid responses to traffic incidents, accidents, or congestion. Beyond monitoring, our system also assists in enforcing traffic regulations by identifying and documenting violations. This capability contributes to improved road safety by discouraging risky behavior. The potential benefits of our project are numerous, including reduced accidents, improved traffic flow, reduced congestion, and better-informed traffic authorities. By optimizing traffic management and safety, our system contributes to a higher quality of urban life

CHAPTER 1
INTRODUCTION

CHAPTER 1
INTRODUCTION

1.1 Problem Statement

In contemporary urban environments, the management of traffic and ensuring road safety has evolved into a multifaceted challenge. The exponential growth in the number of vehicles on the road, coupled with the dynamic nature of urban traffic, has given rise to a range of interconnected issues. These include a high frequency of traffic accidents, widespread speed violations, recurring congestion, and a pressing need for data-driven insights in traffic management.

The fundamental problem at the heart of this project lies in the limitations of existing traffic monitoring and management systems. These systems typically lack the sophistication and comprehensiveness required to address the intricate web of issues that modern urban traffic presents. Existing systems often face challenges in providing comprehensive coverage and analysis of urban traffic patterns and behaviors. They may also struggle to offer real-time monitoring and analysis capabilities, hindering the ability to respond swiftly to changing traffic conditions.

Furthermore, scalability and adaptability are often limitations of traditional traffic management systems, making it challenging to accommodate the evolving needs of urban environments. The project's primary challenge is to overcome these limitations and develop a novel solution that combines real-time speed estimation, vehicle counting, traffic regulation enforcement, seamless data integration, scalability, and adaptability. This challenge calls for innovative technological interventions to create a system capable of transcending the constraints of traditional traffic management, offering a comprehensive approach that proactively addresses the multifaceted issues of modern urban traffic.

The project aims to develop a system that enhances road safety, reduces congestion, minimizes accidents, and empowers traffic management authorities with the tools they need to make informed decisions. By addressing this problem, the project strives to create urban environments where transportation is not only safer and more efficient but also contributes to an improved quality of life for residents and commuters alike.

1.2 Research Objective

The research objectives of the project encompass a comprehensive approach towards the development of an Intelligent Traffic Monitoring System aimed at enhancing traffic management and safety in urban environments. The key objectives are as follows:

- **Development of an Intelligent Traffic Monitoring System:** The primary research objective is to design, develop, and implement an intelligent traffic monitoring system that integrates state-of-the-art YOLO (You Only Look Once) object detection technology. This system will enable real-time traffic management and safety enhancement by accurately detecting and tracking various traffic elements, including vehicles, pedestrians, and cyclists.
- **Real-time Speed Estimation:** A critical aspect of the project involves developing algorithms and techniques to accurately estimate the speed of vehicles in real-time as they traverse monitored areas. This entails leveraging YOLO object detection capabilities to track vehicles and calculate their speeds, with a specific focus on identifying and addressing speed violations to improve road safety.
- **Real-time Vehicle Counting:** Another essential objective is to create a system capable of real-time vehicle counting using YOLO technology. This system will count both incoming and outgoing vehicles, providing essential data for traffic flow management, congestion reduction, and infrastructure planning.
- **Seamless Data Integration:** The project aims to develop mechanisms for the seamless integration of data from various sources, including cameras and other data streams. By integrating disparate data sources, the system will provide a unified and comprehensive view of traffic conditions, enabling more informed decision-making by traffic management authorities.
- **Traffic Regulation Enforcement:** To contribute to the enforcement of traffic regulations and overall road safety, the project will implement features and algorithms to identify and document traffic violations. By automatically detecting and recording violations such as speeding, illegal parking, and red-light running, the system will support law enforcement efforts and promote compliance with traffic regulations.
- **Visualization and Reporting:** User-friendly graphical interfaces and reporting tools will be developed to effectively communicate data and insights to traffic authorities and stakeholders. These visualization tools will enable authorities to interpret and analyze traffic

data more easily, facilitating data-driven decision-making and enhancing overall traffic management efficiency.

- **Integration of Advanced Predictive Analytics:** In addition to real-time monitoring and analysis, the project will explore the integration of advanced predictive analytics techniques. By analyzing historical traffic data and identifying patterns and trends, the system will be able to anticipate traffic congestion, identify potential bottlenecks, and proactively suggest optimized traffic management strategies. This integration of predictive analytics will further enhance the system's ability to anticipate and mitigate traffic-related issues, leading to more efficient traffic flow and reduced congestion.
- **Implementation of Machine Learning for Adaptive Traffic Control:** Leveraging machine learning algorithms, the project will focus on developing adaptive traffic control mechanisms that dynamically adjust traffic signal timings based on real-time traffic conditions. By continuously learning from traffic data and user feedback, the system will optimize traffic signal timings to minimize delays, reduce congestion, and improve overall traffic flow efficiency. This adaptive approach to traffic control will enable the system to respond dynamically to changing traffic patterns and optimize traffic flow in real-time, ultimately enhancing the effectiveness of urban traffic management efforts.

1.3 Project Scope and Limitations

The scope of project is

- **Real-time Speed Estimation:** The project aims to develop advanced algorithms and techniques for real-time speed estimation, with a focus on achieving high accuracy and reliability. By leveraging cutting-edge technologies and methodologies, the system will be capable of accurately identifying and addressing speed violations in real-time, thereby enhancing road safety and compliance with traffic regulations.
- **Real-time Vehicle Counting:** In addition to speed estimation, the system will be designed to perform real-time vehicle counting using YOLO (You Only Look Once) object detection technology. This feature will enable the system to count both incoming and outgoing vehicles accurately, providing valuable data for traffic flow management, congestion reduction, and infrastructure planning. By leveraging YOLO's robust detection capabilities, the system will ensure accurate and reliable vehicle counting across diverse urban environments and traffic conditions.
- **Scalability and Adaptability:** The system will be designed to be scalable and adaptable for deployment in diverse urban environments, accommodating different road types, traffic volumes, and infrastructure configurations. By adopting flexible architecture and modular design principles, the system will be capable of scaling seamlessly to meet the evolving needs of urban traffic management. Additionally, the system will be adaptable to varying traffic conditions, allowing for dynamic adjustments and optimizations to ensure optimal performance in different scenarios.
- **Real-time Processing:** One of the primary objectives of the project is to develop the system to operate in real-time, providing immediate responses to traffic incidents, accidents, and congestion. By leveraging high-speed processing capabilities and efficient data processing algorithms, the system will deliver timely and actionable insights to traffic management authorities, enabling proactive intervention and rapid response to emerging traffic situations. Real-time processing will enhance traffic management effectiveness, improve road safety, and minimize disruptions caused by traffic incidents.

Limitations:

- **Privacy Concerns:** While the system aims to enhance traffic management and safety, it must address potential privacy concerns related to data collection and surveillance. Ensuring compliance with privacy regulations and ethical considerations is paramount to safeguarding individuals' privacy rights and maintaining public trust. The project will implement robust privacy protection measures, such as data anonymization, encryption, and access control, to mitigate privacy risks and protect sensitive information.
- **Hardware Limitations:** The effectiveness of the system may be constrained by the quality and capabilities of the hardware components used for data collection, such as cameras and sensors. Factors such as resolution, field of view, and sensor accuracy can impact the system's performance and reliability. To address this limitation, the project will evaluate and select hardware components carefully, considering factors such as performance, durability, and compatibility with the system's requirements. Additionally, the system will incorporate mechanisms for hardware monitoring, maintenance, and upgrades to ensure optimal performance over time.
- **Resource Constraints:** The project's scope may be limited by budget, time, and resource constraints, which could affect the depth and complexity of system development. To mitigate resource constraints, the project will prioritize key objectives and milestones, focusing on delivering core functionalities and features that align with project goals and stakeholder requirements. Additionally, the project will explore collaborative opportunities, leverage existing resources and expertise, and adopt agile development methodologies to maximize efficiency and mitigate risks associated with resource limitations.
- **Regulatory and Legal Considerations:** The project must consider legal and regulatory factors related to traffic monitoring and data collection, ensuring compliance with relevant laws and regulations. This includes regulations governing surveillance, data privacy, data retention, and intellectual property rights. The project will conduct thorough legal research and consultation to identify applicable regulations and requirements and incorporate legal compliance measures into the system design and implementation. Additionally, the project will establish clear policies and procedures for data handling, access control, and compliance monitoring to mitigate legal risks and ensure regulatory compliance throughout the project lifecycle.

CHAPTER 2
BACKGROUND WORK

CHAPTER 2

BACKGROUND WORK

2.1 Intelligent Traffic System Using Machine Learning Techniques [4]

2.1.1 Introduction

Urban planning plays a critical role in ensuring effective traffic management, given that issues like congestion, accidents, and delays can have significant negative impacts on the economy, society, and the environment. However, managing traffic is becoming increasingly challenging with the rapidly growing global population. In a recent study, the authors proposed the use of machine learning (ML) technology to automate traffic management and assist ambulances in navigating through heavy traffic. ML involves the development of algorithms capable of learning from data and improving over time. To achieve this, the authors employed various algorithms, databases, and mathematical computations to create traffic management systems capable of handling high levels of traffic. Additionally, they implemented object detection techniques and processed photos and videos using Python, a popular programming language. This approach aims to leverage advanced technology to address the complexities of modern traffic management and improve overall efficiency and effectiveness in urban environments.

2.1.2 Merits, Demerits and Challenges

Merits

- **Real-time detection:** YOLO excels in real-time detection, allowing it to identify objects, including vehicles, swiftly and accurately. This real-time capability makes YOLO particularly suitable for applications where fast processing is crucial. For instance, in traffic monitoring systems, real-time detection enables immediate responses to traffic incidents, accidents, and congestion. This swift detection capability enhances overall system responsiveness and contributes to improved traffic management and safety.
- **Single-stage detection:** One of the key advantages of YOLO is its single-stage detection approach, which simplifies the object detection process and reduces computational complexity. Unlike traditional two-stage detection methods, which involve separate region proposal and object detection stages, YOLO performs object detection in a single pass. This streamlined approach not only accelerates the detection process but also reduces the computational resources required, making YOLO more efficient and practical for deployment in resource-constrained environments.

- **Good performance on small objects:** Another notable strength of YOLO is its ability to achieve good performance in detecting small objects, including vehicles, even in challenging scenarios. This capability is particularly advantageous in situations where vehicles may appear small or partially obscured, such as in crowded urban environments or adverse weather conditions. By reliably detecting small objects, YOLO enhances the overall accuracy and robustness of object detection systems, thereby improving their effectiveness in various real-world applications.

Demerits

- **Accuracy trade-off:** Despite its remarkable speed, YOLO may encounter a trade-off in accuracy compared to slower but more complex detection algorithms. This compromise is inherent in its single-stage detection process, which may lead to occasional inaccuracies or misclassifications, especially in scenarios with intricate or overlapping objects. While YOLO strives to achieve a balance between speed and accuracy, users must be mindful of potential compromises in detection precision, particularly in applications where high accuracy is paramount, such as medical imaging or surveillance in critical environments.
- **Limited generalization:** YOLO may struggle with detecting vehicles in challenging conditions, such as extreme lighting variations or occlusions. The model's performance may degrade when confronted with scenarios outside its training data distribution, leading to reduced accuracy or failure to detect objects accurately. This limitation highlights the importance of robust training datasets that encompass diverse environmental conditions and object variations to enhance the model's generalization capabilities. Additionally, fine-tuning or augmenting the training data can help mitigate this issue by exposing the model to a wider range of scenarios and enhancing its adaptability to real-world conditions.
- **Sensitivity to object size and aspect ratio:** Another potential limitation of YOLO is its sensitivity to object size and aspect ratio, particularly in instances where objects exhibit significant variations in scale or orientation. While YOLO's architecture is designed to detect objects of various sizes and aspect ratios, extreme deviations from the training data distribution may pose challenges. Objects that deviate significantly from the average size or aspect ratio may be missed or inaccurately detected, leading to performance degradation in certain scenarios.

Challenges

- **Accuracy and Precision:** YOLO may encounter difficulties in achieving high accuracy and precision in vehicle counting, particularly in challenging conditions such as low light, adverse weather, or complex traffic scenarios. These conditions can obscure vehicle details or introduce noise into the image data, leading to inaccuracies in detection and counting. Overcoming these challenges to maintain accurate counts is crucial for ensuring the reliability and effectiveness of traffic monitoring systems. Solutions may involve enhancing the robustness of the detection algorithm through advanced image processing techniques, optimizing model parameters, or incorporating additional sensor data to supplement visual information.
- **Vehicle Occlusion:** When vehicles overlap or partially obstruct each other, accurately detecting and counting them becomes challenging for YOLO. Occlusion scenarios are common in dense traffic environments or congested roadways, where vehicles may obstruct each other's view in images or videos. Addressing this challenge requires developing sophisticated object detection algorithms capable of accurately identifying and delineating individual vehicles, even in occluded regions. Techniques such as multi-object tracking, context-aware segmentation, or ensemble learning may be employed to improve the accuracy of vehicle counting in occlusion scenarios.
- **Size and Scale Variation:** Vehicles exhibit significant size and scale variation, ranging from motorcycles to trucks, which can pose challenges for object detection algorithms like YOLO. Ensuring that the system can accurately detect and count vehicles of different sizes is essential for comprehensive traffic monitoring and management. To address this challenge, the detection algorithm must be trained on diverse datasets that encompass a wide range of vehicle sizes and scales. Additionally, techniques such as multi-scale feature extraction, adaptive thresholding, or class-specific scaling factors may be employed to improve the algorithm's ability to detect and count vehicles of varying sizes accurately.

2.1.3 Implementation of Intelligent Traffic System Using Machine Learning Techniques

- Collect and annotate a diverse traffic dataset with bounding boxes.
- Choose between YOLO for vehicle counting.
- Preprocess data to suit the chosen model's input format.
- Train the selected model on annotated data.
- Validate and test model performance on new data.

- Implement post-processing (e.g., NMS) for accurate counting (YOLO).
- Deploy the model for real-time inference.
- Monitor accuracy and retrain as needed.
- Address privacy and ethical concerns.
- Scale the system for multiple cameras.
- Plan for regular system maintenance and monitoring.

2.2 Vehicle Counting and Traffic Congestion Detection Using Yolov3[6]

2.2.1 Introduction

This project endeavors to meet these evolving demands by harnessing the power of the OpenCV module in conjunction with the pre-trained YOLOv3 algorithm. Through the fusion of image processing techniques and the sophisticated capabilities of YOLOv3, the system is adept at accurately detecting and categorizing vehicles captured in both images and videos.

The core objective of this initiative lies in the accurate detection and classification of vehicles, facilitated by the robust features of the YOLOv3 algorithm. By meticulously analyzing input imagery and footage, the system excels at delineating vehicles from their surroundings and assigning them to specific classes based on their unique attributes. This process allows for precise identification of various vehicle types, spanning from cars and trucks to motorcycles, even amidst complex traffic scenarios.

2.2.2 Merits, Demerits and Challenges

Merits

- **Accurate Vehicle Counting:** YOLOv3 excels in providing accurate and efficient vehicle counting, enabling traffic management authorities to gather reliable data for congestion analysis and road management. By accurately detecting and categorizing vehicles in real-time, YOLOv3 ensures the integrity of traffic data, facilitating informed decision-making for effective traffic management strategies.
- **Real-Time Processing:** YOLOv3 is optimized for real-time object detection, making it well-suited for live traffic monitoring and congestion detection applications. Its efficient architecture and optimized algorithms enable rapid processing of incoming video streams from traffic cameras, ensuring timely detection and response to changing traffic conditions.
- **High-Speed Inference:** The model's architecture allows for high-speed inference, enabling it to process video streams from traffic cameras with minimal delay. This high-speed processing capability ensures that traffic data is analyzed promptly, allowing for timely interventions to mitigate congestion and optimize traffic flow.
- **Versatility:** YOLOv3 demonstrates versatility in detecting various types of vehicles, including cars, trucks, motorcycles, and buses, making it suitable for diverse traffic scenarios. Its ability to accurately identify and classify different vehicle types enhances the comprehensiveness and applicability of traffic monitoring systems across a wide

range of environments and conditions.

- **Object Localization:** In addition to vehicle counting, YOLOv3 also localizes the positions of detected objects, including vehicles. This localization capability provides valuable information for tracking and analyzing traffic flow dynamics, enabling traffic management authorities to gain insights into vehicle movements and congestion patterns for more effective decision-making and planning.

Demerits

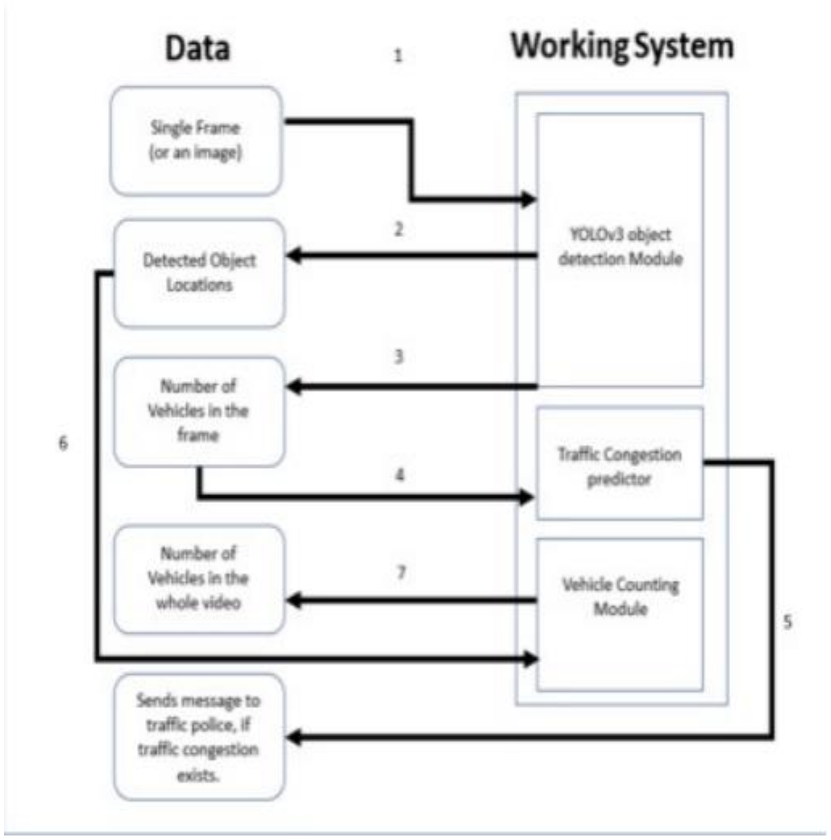
- **Resource-Intensive:** YOLOv3 imposes significant computational demands, necessitating powerful hardware for real-time processing. This requirement for high-performance hardware may not be feasible for all deployment scenarios, particularly in resource-constrained environments where access to such hardware may be limited or cost-prohibitive.
- **Complex Implementation:** Setting up and configuring YOLOv3 for vehicle counting and congestion detection can be intricate and may demand expertise in deep learning techniques. The complexity of implementation can pose challenges for users without sufficient experience in deploying and fine-tuning deep learning models, potentially hindering the adoption of YOLOv3 in practical applications.
- **Data Annotation:** Annotating large datasets for YOLOv3 training can be a time-consuming and labor-intensive process. Manual annotation of images or videos to create ground truth labels for training data requires significant effort, particularly for datasets with diverse traffic scenarios and vehicle types. This process adds overhead to the development pipeline and may delay project timelines.
- **Accuracy in Challenging Conditions:** YOLOv3's accuracy may degrade in challenging environmental conditions, such as heavy rain, low light, or extreme weather. These adverse conditions can obscure vehicle details and introduce noise into the input data, affecting the model's performance in congestion detection tasks. Addressing these challenges requires robust preprocessing techniques and model adaptation strategies to maintain accuracy under varying conditions.
- **Model Maintenance:** Continuous model maintenance and retraining are essential to adapt YOLOv3 to changing traffic patterns and ensure sustained accuracy over time. As traffic dynamics evolve and new scenarios emerge, the model must be regularly updated and fine-tuned to maintain optimal performance. This ongoing maintenance effort demands dedicated resources and expertise to manage effectively.

Challenges

- **Data Collection:** Acquiring and annotating diverse traffic datasets to train the YOLOv3 model is a labor-intensive task that demands a significant amount of data. This process involves sourcing a wide variety of traffic scenes captured under different conditions, such as varying weather, lighting, and traffic densities. Annotating these datasets with bounding boxes for vehicle detection and classification requires meticulous attention to detail and consumes considerable time and resources.
- **Privacy Concerns:** The deployment of traffic cameras for data collection raises legitimate privacy concerns among the public. Ensuring compliance with privacy regulations and safeguarding individuals' privacy rights is paramount. Implementing robust privacy-preserving measures, such as anonymizing sensitive data and restricting access to authorized personnel, is essential to address these concerns and foster public trust in the system.
- **Scalability:** Scaling the system to accommodate multiple cameras and cover a large urban area poses significant challenges. It requires additional infrastructure, such as high-performance servers and network bandwidth, to handle the increased data volume and processing load. Optimizing the system for scalability involves designing efficient distributed architectures and deploying load-balancing mechanisms to ensure seamless operation across multiple camera nodes.
- **Environmental Factors:** Environmental factors, such as glare, shadows, and reflections, can introduce complexities in congestion detection and vehicle tracking. Handling these variations in environmental conditions is crucial for ensuring the reliability and accuracy of the system. Advanced image processing techniques, including adaptive filtering and contrast enhancement, may be employed to mitigate the effects of environmental factors and improve detection performance.
- **Ethical Considerations:** Ethical considerations surrounding surveillance, data collection, and data usage must be carefully addressed throughout the project lifecycle. Respecting individuals' privacy rights, obtaining informed consent for data collection, and adhering to ethical guidelines are paramount. Transparency in the system's operation, clear communication of its purpose and implications, and proactive measures to mitigate potential ethical risks are essential for responsible implementation and societal acceptance.

2.2.3 Implementation of Vehicle Counting and Traffic Congestion Detection using Yolov3

- Takes a frame from the whole video as input.
- It is passed through the first convolutional layer with arbitrary padding and filter size(kernel). The size of the image might get shrunk based on the padding value and filter used.
- When using the pre-trained YOLOv3 model, the weights and biases of different layers are already set to detect the desired vehicles in the given frame. As the image passes through the YOLOv3 model, the activations are generated after each layer based on the pixel intensities of the frame and the neural network's weights and biases.
- These activations are sent as input to the next layer and the next layer in the next step and so on. Finally, in the last layer, the activations in terms of probabilities are evaluated based on all those previous different types of layers (convolutional, pooling, fully connected).


Data
Working System
1
Single Frame (or an image)
2
YOLOv3 object detection Module
Detected Object Locations
3
Number of Vehicles in the frame
6
4
Traffic Congestion predictor
Number of Vehicles in the whole video
7
Vehicle Counting Module
5
Sends message to traffic police, if traffic congestion exists.

2.3 Vehicle speed detection using image processing [5]

Introduction

Image processing has been widely applied to traffic analysis for a variety of purposes. As traffic research field is very wide and it has many goals that include detection of queue, detection of incident, classification of vehicles, and counting vehicles. One of the most important of these purposes is to estimate the speed of a vehicle, a vehicle. Traffic congestion poses lot of problems for people. Because of this, many accidents occur. To reduce this problem, new approach has been developed for estimating the speed of vehicle. A radar technology was used to determine the speed on highways. But it has a disadvantage of high cost. Then a lidar detector was designed to detect the infrared emissions of law enforcement agencies lidar speed detection devices and warn motorists that their speed is being measured. Its disadvantage is it has to be held or placed at a static point. These drawbacks of speed detection techniques motivated to develop new technique for that purpose.

2.3.1 Merits, Demerits and Challenges

Merits

- **Non-intrusive:** Image processing-based speed detection is non-intrusive as it doesn't require physical devices on the road. This non-invasive approach minimizes disruption to traffic flow, allowing vehicles to move freely without encountering additional obstacles or constraints.
- **Cost-Efficient:** Compared to physical sensors or radar-based systems, image processing for speed detection can be a cost-effective solution. It eliminates the need for expensive hardware installations and maintenance, making it suitable for deployment across various road types and locations without significant financial burden.
- **Versatility:** Image processing techniques can be applied to various types of cameras, including CCTV cameras commonly found in urban and highway environments. This versatility makes image processing adaptable to existing infrastructure, leveraging readily available camera networks for speed detection purposes.
- **High Accuracy:** When properly calibrated and configured, image processing can deliver high levels of accuracy in vehicle speed measurements. By analyzing detailed image data and employing sophisticated algorithms, image processing systems can provide precise speed readings, enhancing overall road safety by facilitating effective speed enforcement measures.

- **Real-time Monitoring:** Image processing enables real-time speed monitoring, allowing traffic authorities to promptly detect and respond to speed violations and traffic incidents. This real-time capability enhances situational awareness and facilitates rapid intervention to address speeding behavior, contributing to improved traffic management and enhanced road safety for all road users.

Demerits

- **Environmental Conditions:** Image processing-based speed detection systems may encounter challenges in adverse weather conditions such as heavy rain, fog, or low light. These environmental factors can degrade image quality and hinder the accuracy of speed measurements, potentially leading to inaccuracies in speed detection.
- **Calibration and Maintenance:** Ensuring the accuracy of speed measurements requires regular calibration and maintenance of the cameras used in image processing systems. This process can be resource-intensive, involving periodic checks, adjustments, and repairs to maintain the reliability and precision of speed detection equipment.
- **Privacy Concerns:** Vehicle speed detection using cameras raises legitimate privacy concerns, as it involves capturing and processing images of vehicles and their occupants. Adequate safeguards and privacy-enhancing measures are necessary to protect individuals' privacy rights and ensure compliance with data protection regulations.
- **Data Storage and Processing:** Processing a continuous stream of images from multiple cameras necessitates substantial storage and computational resources. Storing and managing large volumes of image data, as well as performing real-time processing and analysis, can impose significant demands on infrastructure and increase operational costs.
- **Accuracy Variation:** The accuracy of image processing-based speed detection systems may vary depending on factors such as camera placement, image quality, and the size of vehicles being monitored. Variations in these parameters can impact the reliability and consistency of speed measurements, leading to potential discrepancies in detection results across different locations or scenarios.

Challenges

- Image Quality: Ensuring consistent image quality across varying weather conditions and lighting environments poses a significant challenge. Poor-quality images, caused by factors such as fog, rain, or low light, can degrade the accuracy of speed detection systems, making it challenging to reliably identify and track vehicles.

- Data Annotation: Annotating data for model training and validation is a labor-intensive process. It involves manually labeling vehicle positions and speeds in large datasets, which can be time-consuming and resource-intensive. Ensuring accurate annotations is crucial for training effective speed detection models but requires significant human effort and expertise.
- Model Training: Training models for speed detection requires extensive datasets containing annotated images of vehicles at various speeds. This process can be computationally intensive, particularly for deep learning models, posing challenges for resource-constrained environments or organizations with limited computational resources. Additionally, ensuring the quality and diversity of training data is essential for building robust and accurate models.
- Real-Time Processing: Processing images in real time to detect vehicle speeds imposes demands on computational resources. Delays in image processing can affect the system's responsiveness to speed violations, potentially leading to missed detections or delayed enforcement actions. Optimizing real-time processing algorithms and deploying efficient hardware infrastructure are essential to mitigate processing delays and ensure timely speed monitoring.
- Legal Compliance: Complying with legal and regulatory requirements for speed enforcement is paramount. Speed detection systems must adhere to strict standards and guidelines to ensure the accuracy and reliability of speed measurements. Legal challenges may arise from inaccuracies in speed detection or disputes regarding the validity of speed enforcement measures, emphasizing the importance of robust validation and calibration procedures. Additionally, ensuring transparency and accountability in speed enforcement practices is essential to maintain public trust and confidence in the system.

2.3.2 Implementation of Vehicle speed detection using image processing

- In this approach 2 extracted images are selected to apply the motion estimation process in the developed MATLAB algorithm.
- Standalone images is segmented into 16 × 16 small blocks using the division technique. Then the blocks are extracted from video coding to be compared with its respective image in current image and the previous image.
- The blocks are compared to detect the changes in pixels which is used to estimate the

velocity of the respective moving vehicle.

- The successive steps are
  1. Video image processing: Block extraction and subtraction technique is applied into region of interest instead the complete video sequence or images.
  2. Vehicle velocity estimation: Once the image is extracted and segmented into blocks, the motion vector technique is applied to calculate the pixels changes among the two blocks to measure the speed of the moving vehicle. The motion vector is applied with the vector valued function to demonstrate the vehicle speed detection algorithm for the video from surveillance cameras.
  3. Vector-valued function for vehicle motion velocity: This is to verify the respective number of changes in blocks within two consecutive images. The input of a vector valued function may be a scalar or a vector where as the output is a vector. When vector valued function is applied, the vehicle speed can be determined.
  4. Digital Video Recorder (DVR) card setting estimation: If we want to associate the sequence images with vector valued function algorithms, a digital video recorder (DVR) card is required. It is installed to capture the video from the camera and save into a particular folder in the hard drive.

CHAPTER 3
PROPOSED SYSTEM

CHAPTER 3

PROPOSED SYSTEM

3.1. Objective of proposed model:

The primary goal of this project is to create an advanced system capable of accurately detecting, tracking, counting, and estimating the speed of vehicles in real-time using video streams. This system addresses the need for efficient traffic flow analysis, which is crucial for various applications including traffic management, urban planning, and surveillance.

To achieve this objective, the project is structured around a series of carefully designed steps. It begins with data acquisition, where individual frames are extracted from the video stream at an appropriate frame rate. These frames undergo preprocessing techniques such as resizing and normalization to ensure compatibility with the subsequent stages of the system.

The next phase focuses on vehicle detection, wherein a pre-trained YOLOv8 model optimized for vehicle detection is employed. This model processes each frame to identify vehicles by generating bounding boxes and class labels. Additionally, non-maximum suppression is implemented to eliminate redundant detections and improve accuracy.

Following vehicle detection, the system utilizes the Deep SORT algorithm for vehicle tracking. This algorithm facilitates robust feature tracking by identifying key points within the detected bounding boxes across consecutive frames. By estimating vehicle trajectories based on motion vectors, the system can accurately track the movement of vehicles over time.

In parallel with tracking, the system incorporates vehicle counting functionality. Virtual counting lines are defined across traffic lanes, and a counter is incremented each time a vehicle's trajectory crosses these lines. Moreover, the system integrates speed estimation capabilities to further enhance its traffic analysis capabilities. By calculating the distance traveled by each vehicle between frames using bounding box coordinates and known camera parameters, the system can estimate vehicle speeds in real-time.

The output and visualization component of the system play a crucial role in providing actionable insights. Real-time vehicle counts and estimated speeds are displayed on the screen, enabling immediate feedback on traffic conditions. Additionally, bounding boxes, trajectories, and speed information are overlaid on the video stream, enhancing visual confirmation of the system's analysis.

3.2. Algorithms Used for Proposed Model

3.2.1 YOLOv8 for Vehicle Detection:

- **Introduction to YOLO and YOLOv8:**

  YOLO, or "You Only Look Once," represents a breakthrough in object detection algorithms due to its exceptional speed and accuracy. YOLOv8 specifically denotes the eighth iteration of this algorithm, further refined and optimized for various object detection tasks, prominently including the detection of vehicles.

- **Operational Mechanism:**

  The operational mechanism of YOLO involves dividing the input image into a grid, enabling the algorithm to localize objects within the image. For each grid cell, YOLO predicts bounding boxes that encapsulate the objects along with class probabilities to determine the type of object present.

- **Utilization of Convolutional Neural Networks (CNNs):**

  YOLOv8 harnesses the power of convolutional neural networks (CNNs) to process image data and make predictions. CNNs are well-suited for tasks like object detection due to their ability to extract hierarchical features from images. YOLOv8's CNN architecture is designed to analyze the entire image efficiently and predict bounding box coordinates along with class probabilities for detected objects.

- **Training Process:**

  During the training phase, YOLOv8 learns to detect vehicles by optimizing a predefined loss function. This loss function quantifies the disparity between the predicted bounding boxes and class labels and their corresponding ground truth values. Through iterative adjustments of model parameters during training, YOLOv8 gradually improves its ability to accurately detect vehicles in various environmental conditions.

- **Performance Optimization and Evaluation:**

  YOLOv8 undergoes rigorous optimization and evaluation processes to ensure its effectiveness in real-world scenarios. Performance metrics such as precision, recall, and mean average precision (mAP) are commonly used to evaluate the accuracy of object detection algorithms.

3.2.2. Deep SORT for Vehicle Tracking:

- **Advanced Tracking Algorithm:**

  Deep SORT, known as "Simple Online and Realtime Tracking with a Deep Association Metric," is a sophisticated tracking algorithm designed to navigate through complex scenarios involving multiple objects. Its architecture seamlessly integrates detection and association methodologies to enable consistent tracking of objects across consecutive frames.

- **Utilization of Deep Learning:**

  Deep SORT leverages a deep learning-based appearance model as a fundamental component of its functionality. This model assigns unique identifiers (IDs) to detected objects, facilitating their tracking over time. By capturing intricate visual features, Deep SORT ensures accurate and reliable tracking even amidst challenging conditions such as occlusion or varying lighting.

- **Predictive Capability with Kalman Filtering:**

  Incorporating Kalman filtering, Deep SORT predicts the state of each object across subsequent frames. This predictive capability enhances tracking accuracy and robustness by anticipating object trajectories. Even in situations where detection may be obscured or compromised, Deep SORT maintains a coherent understanding of object movement.

- **Real-Time Track Management:**

  Deep SORT excels in managing and updating tracks in real-time. It maintains a dynamic set of active tracks, continuously updating their states based on new detections and associations. This adaptive approach ensures responsiveness to changes in the environment, enhancing overall effectiveness and reliability.

- **Comprehensive Track Management Tasks:**

  Equipped to handle various track management tasks seamlessly, Deep SORT streamlines the process from track initialization to deletion of obsolete tracks. By managing track histories effectively, it ensures the continuity and accuracy of object tracking over extended durations. This comprehensive approach underscores Deep SORT's versatility and effectiveness in real-world tracking applications.

3.2.3. Speed Estimation:

- **Advanced Tracking Algorithm:**

  Deep SORT, known as "Simple Online and Realtime Tracking with a Deep Association Metric," is a sophisticated tracking algorithm designed to navigate through complex scenarios involving multiple objects. Its architecture seamlessly integrates detection and association methodologies to enable consistent tracking of objects across consecutive frames.

- **Utilization of Deep Learning:**

  Deep SORT leverages a deep learning-based appearance model as a fundamental component of its functionality. This model assigns unique identifiers (IDs) to detected objects, facilitating their tracking over time. By capturing intricate visual features, Deep SORT ensures accurate and reliable tracking even amidst challenging conditions such as occlusion or varying lighting.

- **Predictive Capability with Kalman Filtering:**

  Incorporating Kalman filtering, Deep SORT predicts the state of each object across subsequent frames. This predictive capability enhances tracking accuracy and robustness by anticipating object trajectories. Even in situations where detection may be obscured or compromised, Deep SORT maintains a coherent understanding of object movement.

- **Real-Time Track Management:**

  Deep SORT excels in managing and updating tracks in real-time. It maintains a dynamic set of active tracks, continuously updating their states based on new detections and associations. This adaptive approach ensures responsiveness to changes in the environment, enhancing overall effectiveness and reliability.

- **Comprehensive Track Management Tasks:**

  Equipped to handle various track management tasks seamlessly, Deep SORT streamlines the process from track initialization to deletion of obsolete tracks. By managing track histories effectively, it ensures the continuity and accuracy of object tracking over extended durations. This comprehensive approach underscores Deep SORT's versatility and effectiveness in real-world tracking applications.

3.3. Designing:

3.3.1 UML Diagram:

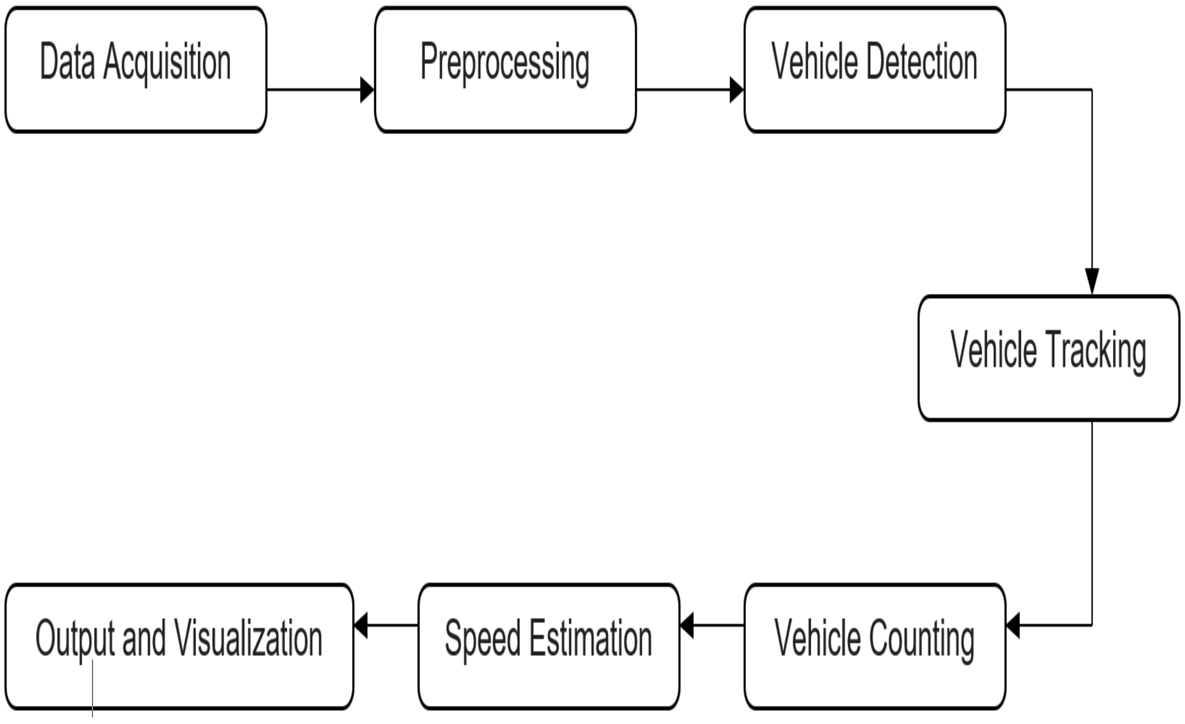


Fig 3.3.1 UML Diagram

- **Data Acquisition and Preprocessing:**
  The primary goal of this project is to create an advanced system capable of accurately detecting, tracking, counting, and estimating the speed of vehicles in real-time using video streams. This system addresses the need for efficient traffic flow analysis, which is crucial for various applications including traffic management, urban planning, and surveillance.
- **Vehicle Detection:**
  To achieve this objective, the project is structured around a series of carefully designed steps. It begins with data acquisition, where individual frames are extracted from the video stream at an appropriate frame rate. These frames undergo preprocessing techniques such as resizing and normalization to ensure compatibility with the subsequent stages of the system.
- **Vehicle Tracking:**

The next phase focuses on vehicle detection, wherein a pre-trained YOLOv8 model optimized for vehicle detection is employed. This model processes each frame to identify vehicles by generating bounding boxes and class labels. Additionally, non-maximum suppression is implemented to eliminate redundant detections and improve accuracy.

- **Vehicle Counting:**

  Following vehicle detection, the system utilizes the Deep SORT algorithm for vehicle tracking. This algorithm facilitates robust feature tracking by identifying key points within the detected bounding boxes across consecutive frames. By estimating vehicle trajectories based on motion vectors, the system can accurately track the movement of vehicles over time.

- **Speed Estimation:**

  In parallel with tracking, the system incorporates vehicle counting functionality. Virtual counting lines are defined across traffic lanes, and a counter is incremented each time a vehicle's trajectory crosses these lines. Moreover, the system integrates speed estimation capabilities to further enhance its traffic analysis capabilities. By calculating the distance traveled by each vehicle between frames using bounding box coordinates and known camera parameters, the system can estimate vehicle speeds in real-time.

3.4. Implementation and Code:

Predict.py:

```
# Ultralytics YOLO

import hydra
import torch
import argparse
import time
from pathlib import Path

import cv2
import torch
import torch.backends.cudnn as cudnn
from numpy import random
from ultralytics.yolo.engine.predictor import BasePredictor
from ultralytics.yolo.utils import DEFAULT_CONFIG, ROOT, ops
from ultralytics.yolo.utils.checks import check_imgsz
from ultralytics.yolo.utils.plotting import Annotator, colors, save_one_box

import cv2
from deep_sort_pytorch.utils.parser import get_config
from deep_sort_pytorch.deep_sort import DeepSort
from collections import deque
import numpy as np
palette = (2 ** 11 - 1, 2 ** 15 - 1, 2 ** 20 - 1)
data_deque = {}

deepsort = None

def init_tracker():
    global deepsort
    cfg_deep = get_config()
    cfg_deep.merge_from_file("deep_sort_pytorch/configs/deep_sort.yaml")

    deepsort= DeepSort(cfg_deep.DEEPSORT.REID_CKPT,
                        max_dist=cfg_deep.DEEPSORT.MAX_DIST,
min_confidence=cfg_deep.DEEPSORT.MIN_CONFIDENCE,
                        nms_max_overlap=cfg_deep.DEEPSORT.NMS_MAX_OVERLAP,
max_iou_distance=cfg_deep.DEEPSORT.MAX_IOU_DISTANCE,
                        max_age=cfg_deep.DEEPSORT.MAX_AGE,
n_init=cfg_deep.DEEPSORT.N_INIT, nn_budget=cfg_deep.DEEPSORT.NN_BUDGET,
                        use_cuda=True)
##########################################################################################
##############
def xyxy_to_xywh(*xyxy):
    """" Calculates the relative bounding box from absolute pixel values. """
    bbox_left = min([xyxy[0].item(), xyxy[2].item()])
    bbox_top = min([xyxy[1].item(), xyxy[3].item()])
    bbox_w = abs(xyxy[0].item() - xyxy[2].item())
    bbox_h = abs(xyxy[1].item() - xyxy[3].item())
    x_c = (bbox_left + bbox_w / 2)
    y_c = (bbox_top + bbox_h / 2)
    w = bbox_w
    h = bbox_h
    return x_c, y_c, w, h

def xyxy_to_tlwh(bbox_xyxy):
    tlwh_bboxs = []
    for i, box in enumerate(bbox_xyxy):
        x1, y1, x2, y2 = [int(i) for i in box]
        top = x1
        left = y1
        w = int(x2 - x1)
        h = int(y2 - y1)
        tlwh_obj = [top, left, w, h]
        tlwh_bboxs.append(tlwh_obj)
    return tlwh_bboxs

def compute_color_for_labels(label):
    """
    Simple function that adds fixed color depending on the class
    """
    if label == 0: #person
        color = (85,45,255)
    elif label == 2: # Car
        color = (222,82,175)
    elif label == 3:  # Motobike
        color = (0, 204, 255)
    elif label == 5:  # Bus
        color = (0, 149, 255)
    else:
        color = [int((p * (label ** 2 - label + 1)) % 255) for p in palette]
    return tuple(color)

def draw_border(img, pt1, pt2, color, thickness, r, d):
    x1,y1 = pt1
    x2,y2 = pt2
    # Top left
    cv2.line(img, (x1 + r, y1), (x1 + r + d, y1), color, thickness)
    cv2.line(img, (x1, y1 + r), (x1, y1 + r + d), color, thickness)
    cv2.ellipse(img, (x1 + r, y1 + r), (r, r), 180, 0, 90, color, thickness)
    # Top right
    cv2.line(img, (x2 - r, y1), (x2 - r - d, y1), color, thickness)
    cv2.line(img, (x2, y1 + r), (x2, y1 + r + d), color, thickness)
    cv2.ellipse(img, (x2 - r, y1 + r), (r, r), 270, 0, 90, color, thickness)
    # Bottom left
    cv2.line(img, (x1 + r, y2), (x1 + r + d, y2), color, thickness)
    cv2.line(img, (x1, y2 - r), (x1, y2 - r - d), color, thickness)
    cv2.ellipse(img, (x1 + r, y2 - r), (r, r), 90, 0, 90, color, thickness)
    # Bottom right
    cv2.line(img, (x2 - r, y2), (x2 - r - d, y2), color, thickness)
    cv2.line(img, (x2, y2 - r), (x2, y2 - r - d), color, thickness)
    cv2.ellipse(img, (x2 - r, y2 - r), (r, r), 0, 0, 90, color, thickness)

    cv2.rectangle(img, (x1 + r, y1), (x2 - r, y2), color, -1, cv2.LINE_AA)
    cv2.rectangle(img, (x1, y1 + r), (x2, y2 - r - d), color, -1, cv2.LINE_AA)

    cv2.circle(img, (x1 +r, y1+r), 2, color, 12)
    cv2.circle(img, (x2 -r, y1+r), 2, color, 12)
    cv2.circle(img, (x1 +r, y2-r), 2, color, 12)
    cv2.circle(img, (x2 -r, y2-r), 2, color, 12)

    return img

def UI_box(x, img, color=None, label=None, line_thickness=None):
    # Plots one bounding box on image img
    tl = line_thickness or round(0.002 * (img.shape[0] + img.shape[1]) / 2) + 1  # line/font
thickness
    color = color or [random.randint(0, 255) for _ in range(3)]
    c1, c2 = (int(x[0]), int(x[1])), (int(x[2]), int(x[3]))
    cv2.rectangle(img, c1, c2, color, thickness=tl, lineType=cv2.LINE_AA)
    if label:
        tf = max(tl - 1, 1)  # font thickness
        t_size = cv2.getTextSize(label, 0, fontScale=tl / 3, thickness=tf)[0]

        img = draw_border(img, (c1[0], c1[1] - t_size[1] -3), (c1[0] + t_size[0], c1[1]+3), color,
1, 8, 2)
```

```
        cv2.putText(img, label, (c1[0], c1[1] - 2), 0, tl / 3, [225, 255, 255], thickness=tf,
lineType=cv2.LINE_AA)




def draw_boxes(img, bbox, names,object_id, identities=None, offset=(0, 0)):
    #cv2.line(img, line[0], line[1], (46,162,112), 3)

    height, width, _ = img.shape
    # remove tracked point from buffer if object is lost
    for key in list(data_deque):
      if key not in identities:
        data_deque.pop(key)

    for i, box in enumerate(bbox):
        x1, y1, x2, y2 = [int(i) for i in box]
        x1 += offset[0]
        x2 += offset[0]
        y1 += offset[1]
        y2 += offset[1]

        # code to find center of bottom edge
        center = (int((x2+x1)/ 2), int((y2+y2)/2))

        # get ID of object
        id = int(identities[i]) if identities is not None else 0

        # create new buffer for new object
        if id not in data_deque:
          data_deque[id] = deque(maxlen= 64)
        color = compute_color_for_labels(object_id[i])
        obj_name = names[object_id[i]]
        label = '{}{:d}'.format("", id) + ":"+ '%s' % (obj_name)

        # add center to buffer
        data_deque[id].appendleft(center)
        UI_box(box, img, label=label, color=color, line_thickness=2)
        # draw trail
        for i in range(1, len(data_deque[id])):
            # check if on buffer value is none
            if data_deque[id][i - 1] is None or data_deque[id][i] is None:
                continue
            # generate dynamic thickness of trails
            thickness = int(np.sqrt(64 / float(i + i)) * 1.5)
            # draw trails
            cv2.line(img, data_deque[id][i - 1], data_deque[id][i], color, thickness)
    return img



class DetectionPredictor(BasePredictor):

    def get_annotator(self, img):
        return Annotator(img, line_width=self.args.line_thickness,
example=str(self.model.names))

    def preprocess(self, img):
        img = torch.from_numpy(img).to(self.model.device)
        img = img.half() if self.model.fp16 else img.float()  # uint8 to fp16/32
        img /= 255  # 0 - 255 to 0.0 - 1.0
        return img

    def postprocess(self, preds, img, orig_img):
        preds = ops.non_max_suppression(preds,
                                        self.args.conf,
                                        self.args.iou,
                                        agnostic=self.args.agnostic_nms,
                                        max_det=self.args.max_det)

        for i, pred in enumerate(preds):
            shape = orig_img[i].shape if self.webcam else orig_img.shape
            pred[:, :4] = ops.scale_boxes(img.shape[2:], pred[:, :4], shape).round()

        return preds

    def write_results(self, idx, preds, batch):
        p, im, im0 = batch
        all_outputs = []
        log_string = ""
        if len(im.shape) == 3:
            im = im[None]  # expand for batch dim
        self.seen += 1
        im0 = im0.copy()
        if self.webcam:  # batch_size >= 1
            log_string += f'{idx}: '
            frame = self.dataset.count
        else:
            frame = getattr(self.dataset, 'frame', 0)

        self.data_path = p
        save_path = str(self.save_dir / p.name)  # im.jpg
        self.txt_path = str(self.save_dir / 'labels' / p.stem) + ('' if self.dataset.mode == 'image' else
f'_{frame}')
        log_string += '%gx%g ' % im.shape[2:]  # print string
        self.annotator = self.get_annotator(im0)

        det = preds[idx]
```

```
all_outputs.append(det)
if len(det) == 0:
    return log_string
for c in det[:, 5].unique():
    n = (det[:, 5] == c).sum()  # detections per class
    log_string += f"{n} {self.model.names[int(c)]}{'s' * (n > 1)}, "
# write
gn = torch.tensor(im0.shape)[[1, 0, 1, 0]]  # normalization gain whwh
xywh_bboxs = []
confs = []
oids = []
outputs = []
for *xyxy, conf, cls in reversed(det):
    x_c, y_c, bbox_w, bbox_h = xyxy_to_xywh(*xyxy)
    xywh_obj = [x_c, y_c, bbox_w, bbox_h]
    xywh_bboxs.append(xywh_obj)
    confs.append([conf.item()])
    oids.append(int(cls))
xywhs = torch.Tensor(xywh_bboxs)
confss = torch.Tensor(confs)

outputs = deepsort.update(xywhs, confss, oids, im0)
if len(outputs) > 0:
    bbox_xyxy = outputs[:, :4]
    identities = outputs[:, -2]
    object_id = outputs[:, -1]

    draw_boxes(im0, bbox_xyxy, self.model.names, object_id,identities)

return log_string
```

```
@hydra.main(version_base=None, config_path=str(DEFAULT_CONFIG.parent),
config_name=DEFAULT_CONFIG.name)
def predict(cfg):
    init_tracker()
    cfg.model = cfg.model or "yolov8n.pt"
    cfg.imgsz = check_imgsz(cfg.imgsz, min_dim=2)  # check image size
    cfg.source = cfg.source if cfg.source is not None else ROOT / "assets"
    predictor = DetectionPredictor(cfg)
    predictor()


if __name__ == "__main__":
    predict()
```

Train.Py:

```
# Ultralytics YOLO

from copy import copy

import hydra
import torch
import torch.nn as nn

from ultralytics.nn.tasks import DetectionModel
from ultralytics.yolo import v8
from ultralytics.yolo.data import build_dataloader
from ultralytics.yolo.data.dataloaders.v5loader import create_dataloader
from ultralytics.yolo.engine.trainer import BaseTrainer
from ultralytics.yolo.utils import DEFAULT_CONFIG, colorstr
from ultralytics.yolo.utils.loss import BboxLoss
from ultralytics.yolo.utils.ops import xywh2xyxy
from ultralytics.yolo.utils.plotting import plot_images, plot_results
from ultralytics.yolo.utils.tal import TaskAlignedAssigner, dist2bbox, make_anchors
from ultralytics.yolo.utils.torch_utils import de_parallel


# BaseTrainer python usage
class DetectionTrainer(BaseTrainer):

    def get_dataloader(self, dataset_path, batch_size, mode="train", rank=0):
        # TODO: manage splits differently
        # calculate stride - check if model is initialized
        gs = max(int(de_parallel(self.model).stride.max() if self.model else 0), 32)
        return create_dataloader(path=dataset_path,
                                 imgsz=self.args.imgsz,
                                 batch_size=batch_size,
                                 stride=gs,
                                 hyp=dict(self.args),
                                 augment=mode == "train",
                                 cache=self.args.cache,
                                 pad=0 if mode == "train" else 0.5,
                                 rect=self.args.rect,
                                 rank=rank,
                                 workers=self.args.workers,
                                 close_mosaic=self.args.close_mosaic != 0,
                                 prefix=colorstr(f'{mode}: '),
                                 shuffle=mode == "train",
                                 seed=self.args.seed)[0] if self.args.v5loader else \
            build_dataloader(self.args, batch_size, img_path=dataset_path, stride=gs, rank=rank,
mode=mode)[0]

    def preprocess_batch(self, batch):
        batch["img"] = batch["img"].to(self.device, non_blocking=True).float() / 255
        return batch

    def set_model_attributes(self):
        nl = de_parallel(self.model).model[-1].nl  # number of detection layers (to scale hyps)
        self.args.box *= 3 / nl  # scale to layers
        # self.args.cls *= self.data["nc"] / 80 * 3 / nl  # scale to classes and layers
        self.args.cls *= (self.args.imgsz / 640) ** 2 * 3 / nl  # scale to image size and layers
        self.model.nc = self.data["nc"]  # attach number of classes to model
        self.model.args = self.args  # attach hyperparameters to model
        # TODO: self.model.class_weights = labels_to_class_weights(dataset.labels,
nc).to(device) * nc
        self.model.names = self.data["names"]

    def get_model(self, cfg=None, weights=None, verbose=True):
        model = DetectionModel(cfg, ch=3, nc=self.data["nc"], verbose=verbose)
        if weights:
            model.load(weights)

        return model

    def get_validator(self):
        self.loss_names = 'box_loss', 'cls_loss', 'dfl_loss'
        return v8.detect.DetectionValidator(self.test_loader,
                                            save_dir=self.save_dir,
                                            logger=self.console,
                                            args=copy(self.args))

    def criterion(self, preds, batch):
        if not hasattr(self, 'compute_loss'):
            self.compute_loss = Loss(de_parallel(self.model))
        return self.compute_loss(preds, batch)

    def label_loss_items(self, loss_items=None, prefix="train"):
        """
        Returns a loss dict with labelled training loss items tensor
        """
        # Not needed for classification but necessary for segmentation & detection
        keys = [f"{prefix}/{x}" for x in self.loss_names]
        if loss_items is not None:
            loss_items = [round(float(x), 5) for x in loss_items]  # convert tensors to 5 decimal
place floats
            return dict(zip(keys, loss_items))
        else:
            return keys

    def progress_string(self):
        return ('\n' + '%11s' *
                (4 + len(self.loss_names))) % ('Epoch', 'GPU_mem', *self.loss_names, 'Instances',
'Size')

    def plot_training_samples(self, batch, ni):
        plot_images(images=batch["img"],
                    batch_idx=batch["batch_idx"],
                    cls=batch["cls"].squeeze(-1),
                    bboxes=batch["bboxes"],
                    paths=batch["im_file"],
                    fname=self.save_dir / f"train_batch{ni}.jpg")

    def plot_metrics(self):
        plot_results(file=self.csv)  # save results.png



# Criterion class for computing training losses
class Loss:

    def __init__(self, model):  # model must be de-paralleled

        device = next(model.parameters()).device  # get model device
        h = model.args  # hyperparameters

        m = model.model[-1]  # Detect() module
        self.bce = nn.BCEWithLogitsLoss(reduction='none')
        self.hyp = h
        self.stride = m.stride  # model strides
        self.nc = m.nc  # number of classes
        self.no = m.no
        self.reg_max = m.reg_max
        self.device = device
        self.use_dfl = m.reg_max > 1
        self.assigner = TaskAlignedAssigner(topk=10, num_classes=self.nc, alpha=0.5, beta=6.0)
        self.bbox_loss = BboxLoss(m.reg_max - 1, use_dfl=self.use_dfl).to(device)
        self.proj = torch.arange(m.reg_max, dtype=torch.float, device=device)

    def preprocess(self, targets, batch_size, scale_tensor):
        if targets.shape[0] == 0:
            out = torch.zeros(batch_size, 0, 5, device=self.device)
        else:
            i = targets[:, 0]  # image index
            _, counts = i.unique(return_counts=True)
            out = torch.zeros(batch_size, counts.max(), 5, device=self.device)
            for j in range(batch_size):
                matches = i == j
                n = matches.sum()
                if n:
                    out[j, :n] = targets[matches, 1:]
            out[..., 1:5] = xywh2xyxy(out[..., 1:5].mul_(scale_tensor))
        return out

    def bbox_decode(self, anchor_points, pred_dist):
        if self.use_dfl:
            b, a, c = pred_dist.shape  # batch, anchors, channels
            pred_dist = pred_dist.view(b, a, 4, c //
4).softmax(3).matmul(self.proj.type(pred_dist.dtype))
            # pred_dist = pred_dist.view(b, a, c // 4,
4).transpose(2,3).softmax(3).matmul(self.proj.type(pred_dist.dtype))
            # pred_dist = (pred_dist.view(b, a, c // 4, 4).softmax(2) *
self.proj.type(pred_dist.dtype).view(1, 1, -1, 1)).sum(2)
        return dist2bbox(pred_dist, anchor_points, xywh=False)
    def __call__(self, preds, batch):
        loss = torch.zeros(3, device=self.device)  # box, cls, dfl
        feats = preds[1] if isinstance(preds, tuple) else preds
        pred_distri, pred_scores = torch.cat([xi.view(feats[0].shape[0], self.no, -1) for xi in feats],
2).split(
            (self.reg_max * 4, self.nc), 1)

        pred_scores = pred_scores.permute(0, 2, 1).contiguous()
        pred_distri = pred_distri.permute(0, 2, 1).contiguous()

        dtype = pred_scores.dtype
        batch_size = pred_scores.shape[0]
        imgsz = torch.tensor(feats[0].shape[2:], device=self.device, dtype=dtype) * self.stride[0]
# image size (h,w)
        anchor_points, stride_tensor = make_anchors(feats, self.stride, 0.5)

        # targets
        targets = torch.cat((batch["batch_idx"].view(-1, 1), batch["cls"].view(-1, 1),
batch["bboxes"]), 1)
        targets = self.preprocess(targets.to(self.device), batch_size, scale_tensor=imgsz[[1, 0, 1,
0]])
        gt_labels, gt_bboxes = targets.split((1, 4), 2)  # cls, xyxy
        mask_gt = gt_bboxes.sum(2, keepdim=True).gt_(0)

        # pboxes
        pred_bboxes = self.bbox_decode(anchor_points, pred_distri)  # xyxy, (b, h*w, 4)

        _, target_bboxes, target_scores, fg_mask, _ = self.assigner(
            pred_scores.detach().sigmoid(), (pred_bboxes.detach() *
stride_tensor).type(gt_bboxes.dtype),
            anchor_points * stride_tensor, gt_labels, gt_bboxes, mask_gt)
        target_bboxes /= stride_tensor
        target_scores_sum = target_scores.sum()

        # cls loss
        # loss[1] = self.varifocal_loss(pred_scores, target_scores, target_labels) /
target_scores_sum  # VFL way
        loss[1] = self.bce(pred_scores, target_scores.to(dtype)).sum() / target_scores_sum  # BCE

        # bbox loss
        if fg_mask.sum():
            loss[0], loss[2] = self.bbox_loss(pred_distri, pred_bboxes, anchor_points,
target_bboxes, target_scores,
                                              target_scores_sum, fg_mask)

        loss[0] *= self.hyp.box  # box gain
        loss[1] *= self.hyp.cls  # cls gain
        loss[2] *= self.hyp.dfl  # dfl gain

        return loss.sum() * batch_size, loss.detach()  # loss(box, cls, dfl)


@hydra.main(version_base=None, config_path=str(DEFAULT_CONFIG.parent),
config_name=DEFAULT_CONFIG.name)
def train(cfg):
    cfg.model = cfg.model or "yolov8n.yaml"
    cfg.data = cfg.data or "coco128.yaml"  # or yolo.ClassificationDataset("mnist")
    # trainer = DetectionTrainer(cfg)
    # trainer.train()
    from ultralytics import YOLO
    model = YOLO(cfg.model)
    model.train(**cfg)
```

```
if __name__ == "__main__":
    """
    CLI usage:
    python ultralytics/yolo/v8/detect/train.py model=yolov8n.yaml data=coco128 epochs=100
imgsz=640

    TODO:
    yolo task=detect mode=train model=yolov8n.yaml data=coco128.yaml epochs=100
    """
    train()
```

CHAPTER 4
RESULTS AND DISCUSSION

CHAPTER 4

RESULTS AND DISCUSSION

4.1. Performance metrices:

- The accuracy of the model is 94.8%.
- The precision achieved by the model is 94.8%.
- The recall of the model is 100%.
- The F1 score of the model is 97.3%.

The integration of YOLO for detection and the deep sort algorithm for tracking represents a significant advancement in vehicle counting accuracy. By leveraging YOLO's robust object detection capabilities and deep sort's precise tracking of vehicles across frames, the system can achieve highly accurate vehicle counts. This integration minimizes error rates commonly associated with traditional methods of vehicle counting, such as manual counting or simple detection algorithms. With improved accuracy, the system provides more reliable data for traffic analysis, contributing to better-informed decision-making in transportation management.

In addition to accurate vehicle counting, the system offers reliable speed estimation capabilities. Speed estimations are derived from precise distance and time calculations, facilitated by the robust tracking provided by the deep sort algorithm. These estimations are valuable for traffic analysis and potential enforcement applications, as they provide insights into vehicle speed distributions and compliance with speed limits. By combining accurate vehicle counts with reliable speed estimations, the system enhances the overall understanding of traffic dynamics, facilitating targeted interventions to improve road safety and efficiency.

The system's ability to process video streams in real-time is a key feature that enables timely insights into traffic patterns and conditions. Real-time data acquisition allows operators and decision-makers to monitor traffic flow dynamics as they unfold, facilitating immediate responsiveness to changing conditions. This capability is particularly beneficial for managing traffic congestion, identifying traffic incidents, and optimizing traffic signal timing to improve overall traffic flow efficiency.

Visualized insights provided by overlaying tracking information and speed measurements on the video stream enhance the system's usability and interpretability.

Operators and researchers can observe traffic flow dynamics and individual vehicle behavior directly from the video feed, gaining valuable insights into traffic patterns and congestion hotspots. This visual feedback facilitates rapid decision-making and allows for the identification of trends or anomalies that may require further investigation or intervention.

Furthermore, the collected data can be leveraged for data-driven analysis to inform traffic management strategies. By analyzing the collected data, stakeholders can identify peak traffic times, understand individual vehicle behavior, and optimize traffic flow strategies accordingly. This data-driven approach enables evidence-based decision-making in transportation management, leading to more effective allocation of resources and improved overall traffic efficiency. Overall, the system's capabilities for accurate vehicle counting, reliable speed estimation, real-time data acquisition, visualized insights, and data-driven analysis collectively contribute to more efficient and informed traffic management practices.

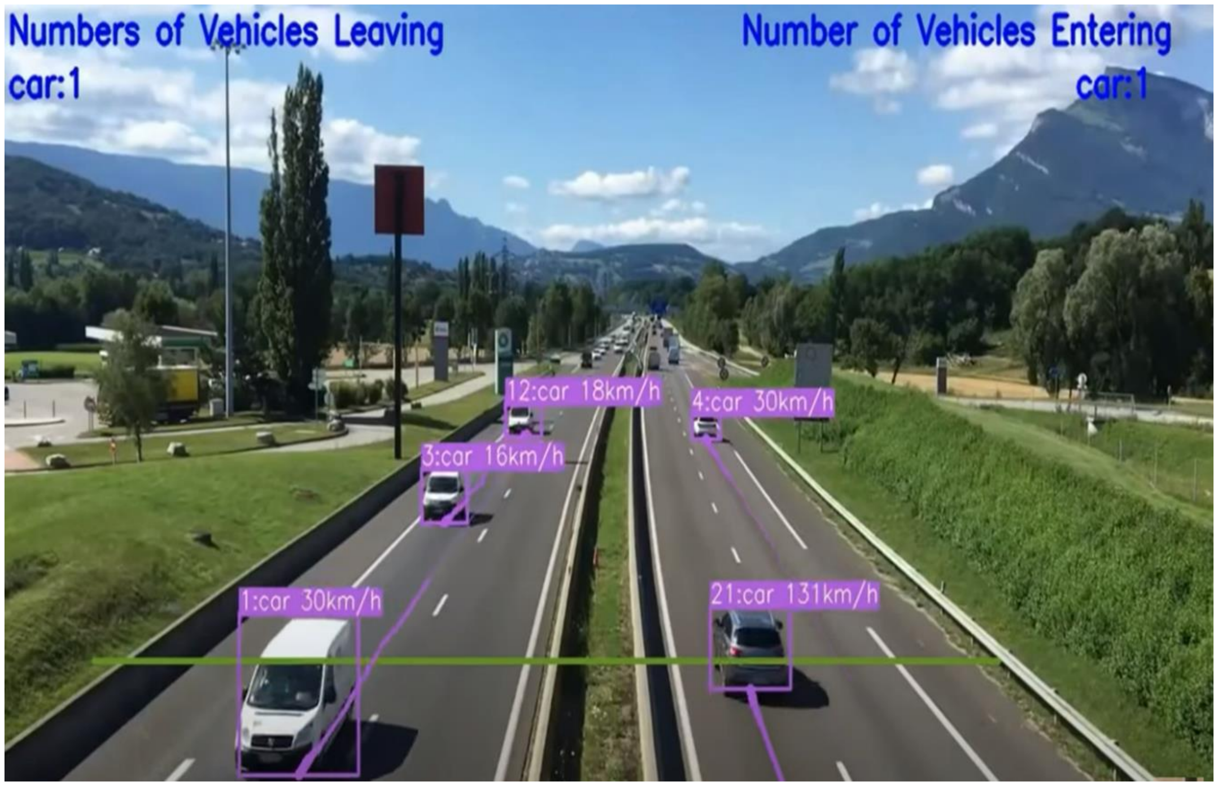


Fig 4.1.1 Detection 1

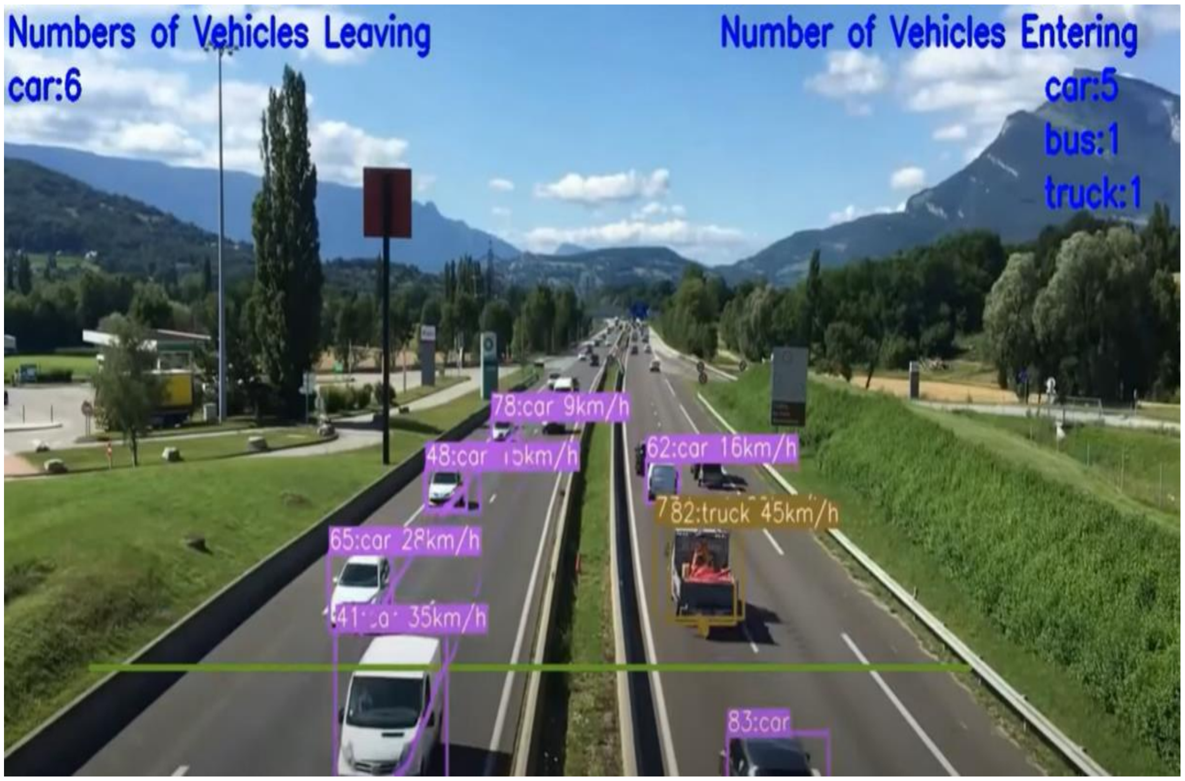


Fig 4.1.2 Detection 2

CHAPTER 5
CONCLUSION

CHAPTER 5

CONCLUSION

The integration of YOLO object detection and deep SORT algorithms in this project has yielded promising results for vehicle counting and speed estimation in traffic scenarios. By leveraging the capabilities of these advanced AI techniques, we have achieved highly accurate vehicle count data and reliable speed estimation, providing valuable insights into traffic flow dynamics.

Firstly, by employing YOLO for object detection, we achieve a high level of accuracy in identifying and tracking vehicles within a traffic scene. YOLO's real-time processing capabilities enable efficient detection of vehicles even in complex environments, ensuring that no vehicles are missed in the counting process. This accuracy forms the foundation for reliable vehicle count data, which is crucial for various traffic management tasks, such as congestion analysis, infrastructure planning, and traffic flow optimization.

Secondly, the integration with deep SORT enhances the tracking process by assigning unique identities to each detected vehicle and maintaining consistent tracking across frames. This not only facilitates accurate vehicle counting but also enables precise speed estimation by analyzing the displacement of vehicles over time. The combination of YOLO and deep SORT ensures that speed estimates are robust and trustworthy, providing valuable insights into the dynamics of traffic flow.

The potential benefits of this integration extend beyond mere data collection. With the ability to integrate seamlessly into existing traffic management systems, our solution opens up avenues for enhanced traffic monitoring and analysis. Furthermore, the insights gained from this project offer exciting possibilities for both practical applications in traffic management and avenues for further research in transportation studies.

In essence, this project lays a robust foundation for a powerful tool in traffic monitoring and analysis. By harnessing the potential of AI-powered solutions, we are poised to revolutionize the way we understand and manage traffic, ultimately contributing to smarter and safer transportation systems. As we continue to refine and expand upon this work, the future holds great promise for the integration of AI technologies in enhancing the efficiency and safety of our roadways.

Future Scope

- **Improved Accuracy and Robustness**: Developing more accurate and robust vehicle detection models, possibly through the integration of advanced deep learning architectures, attention mechanisms, or ensemble methods.
- **Small Object Detection**: Enhancing the capability of detection models to accurately detect small vehicles or vehicles at a distance.
- **Adaptation to Dynamic Environments:** Designing systems capable of adapting to dynamic environments with changing traffic patterns, road conditions, and environmental factors.
- **Deployment in Autonomous Vehicles**: Investigating the integration of vehicle detection, counting, and speed estimation algorithms into autonomous vehicle systems.
- **Behavior Analysis**: Going beyond simple counting and speed estimation to analyze complex vehicle behaviors, such as lane-changing, merging, turning, and interactions with other road users for accident prevention.

REFERENCES

REFERENCES

[1]. Ammar Awni Abbass University of Baghdad "Estimating vehicle speed using image processing", AL-Mansour Journal / No.14/ Special Issue /(Part Two) 2010

[2]. Osman Ibrahim, Hazem ElGendy, and Ahmed M. ElShafee, Member, IEEE " Speed Detection Camera System using Image Processing Techniques on Video Streams ", International Journal of Computer and Electrical Engineering, Vol. 3, No. 6, December 2011

[3]. Siddharth Jhumat," Vehicle Speed Estimation in Accident Prone Areas using Image Processing ", International Journal of Advanced Research in Computer and Communication Engineering Vol. 3, Issue 5, May 2014

[4]. https://ijrpr.com/uploads/V4ISSUE5/IJRPR12846.pdf

[5]. https://www.researchgate.net/publication/327337312_A_Review_on_Vehicle_Speed_Detection_using_Image_Processing

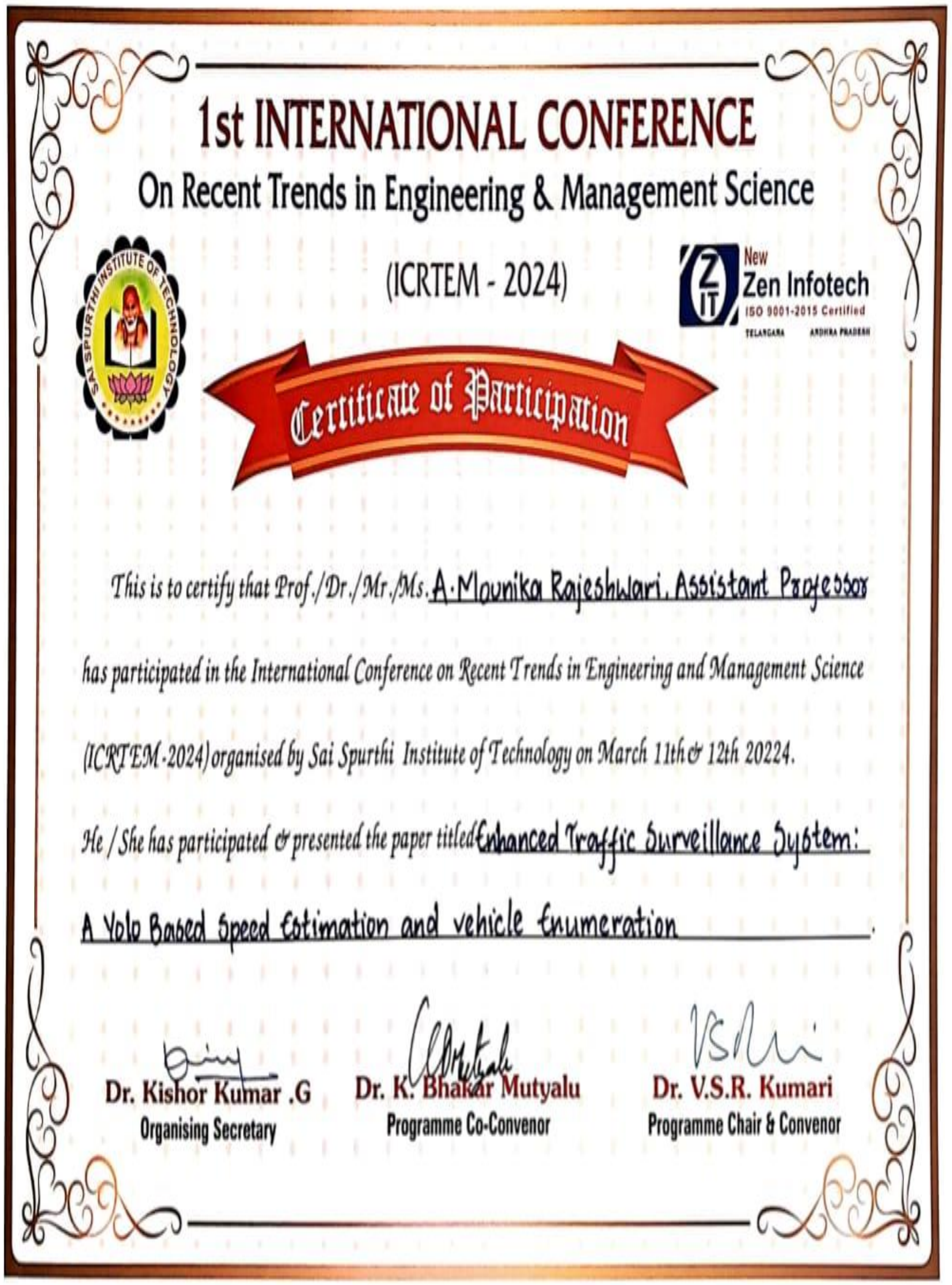
[6]. https://www.ijfmr.com/papers/2023/4/4616.pdf

[7]. S. S. S. Ranjit, S. A. Anas, S. K. Subramaniam, K. C. Lim, A. F. I. Fayeez, A. R. Amirah," Real-Time Vehicle Speed Detection Algorithm using Motion Vector Technique ", Proc. of Int. Conf. on Advances in Electrical & Electronics 2012

GITHUB LINK

https://github.com/Nithin1572/MajorProject

PUBLICATION CERTIFICATES



1st INTERNATIONAL CONFERENCE

On Recent Trends in Engineering & Management Science

(ICRTEM - 2024)

SAI SPURTHI INSTITUTE OF TECHNOLOGY
New
Zen Infotech
ISO 9001-2015 Certified
TELANGANA ANDHRA PRADESH

Certificate of Participation

This is to certify that Prof./Dr./Mr./Ms. A·Mounika Rajeshwari, Assistant Professor

has participated in the International Conference on Recent Trends in Engineering and Management Science

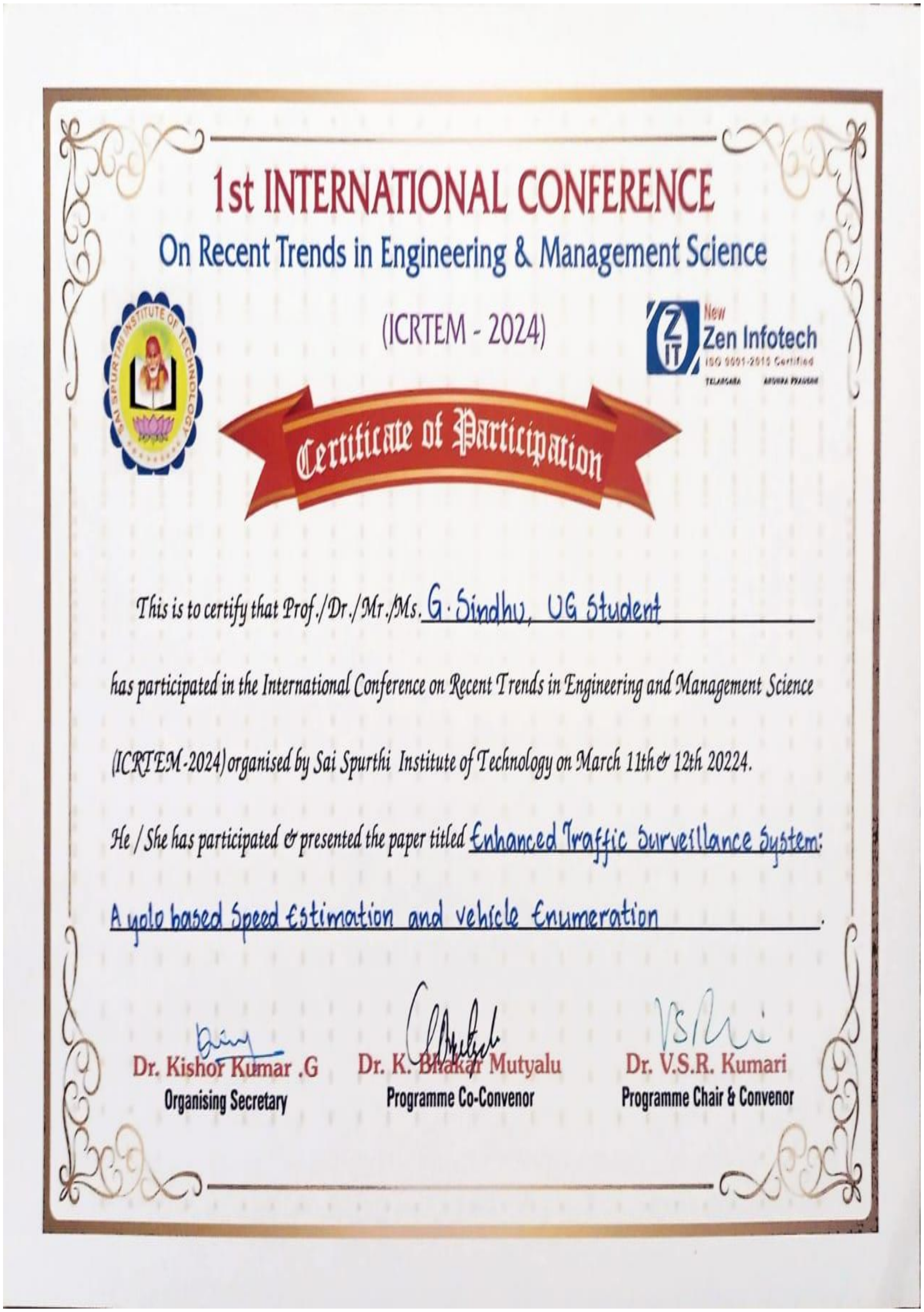
(ICRTEM-2024) organised by Sai Spurthi Institute of Technology on March 11th & 12th 20224.

He / She has participated & presented the paper titled Enhanced Traffic Surveillance System: A Yolo Based Speed Estimation and vehicle Enumeration.

[illegible]
Dr. Kishor Kumar .G
Organising Secretary

[illegible]
Dr. K. Bhakar Mutyalu
Programme Co-Convenor

[illegible]
Dr. V.S.R. Kumari
Programme Chair & Convenor



1st INTERNATIONAL CONFERENCE

On Recent Trends in Engineering & Management Science

(ICRTEM - 2024)

SAI SPURTHI INSTITUTE OF TECHNOLOGY
ZIT
New Zen Infotech
ISO 9001-2015 Certified

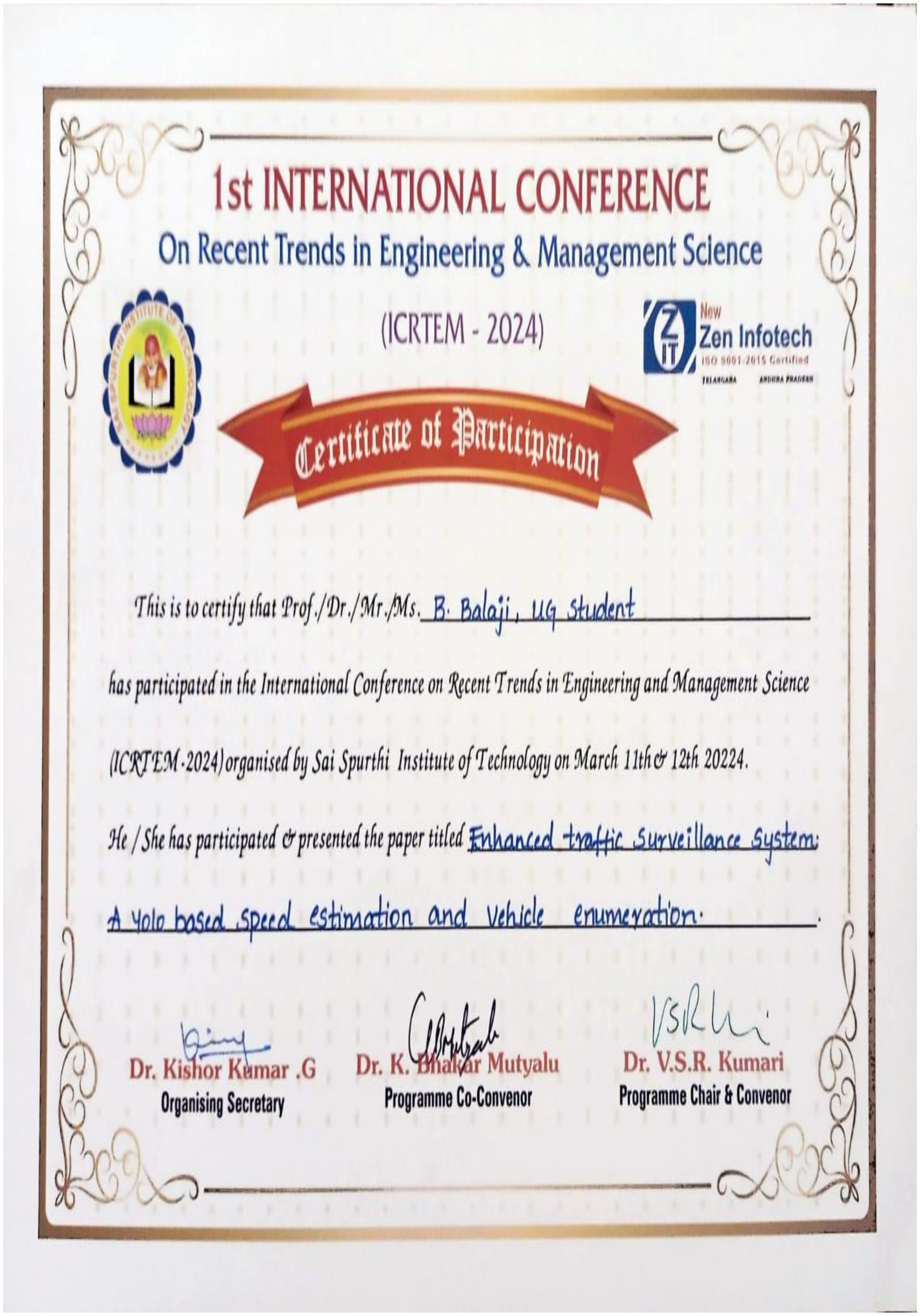
Certificate of Participation

This is to certify that Prof./Dr./Mr./Ms. G. Sindhu, UG Student

has participated in the International Conference on Recent Trends in Engineering and Management Science (ICRTEM-2024) organised by Sai Spurthi Institute of Technology on March 11th & 12th 20224.

He / She has participated & presented the paper titled Enhanced Traffic Surveillance System: A yolo based Speed Estimation and vehicle Enumeration.

[illegible]	[illegible]	[illegible]
Dr. Kishor Kumar .G	Dr. K. Bhakar Mutyalu	Dr. V.S.R. Kumari
Organising Secretary	Programme Co-Convenor	Programme Chair & Convenor



1st INTERNATIONAL CONFERENCE

On Recent Trends in Engineering & Management Science

(ICRTEM - 2024)

New Zen Infotech ISO 9001-2015 Certified TELANGANA ANDHRA PRADESH

Certificate of Participation

This is to certify that Prof./Dr./Mr./Ms. B. Balaji, UG Student

has participated in the International Conference on Recent Trends in Engineering and Management Science (ICRTEM-2024) organised by Sai Spurthi Institute of Technology on March 11th & 12th 20224.

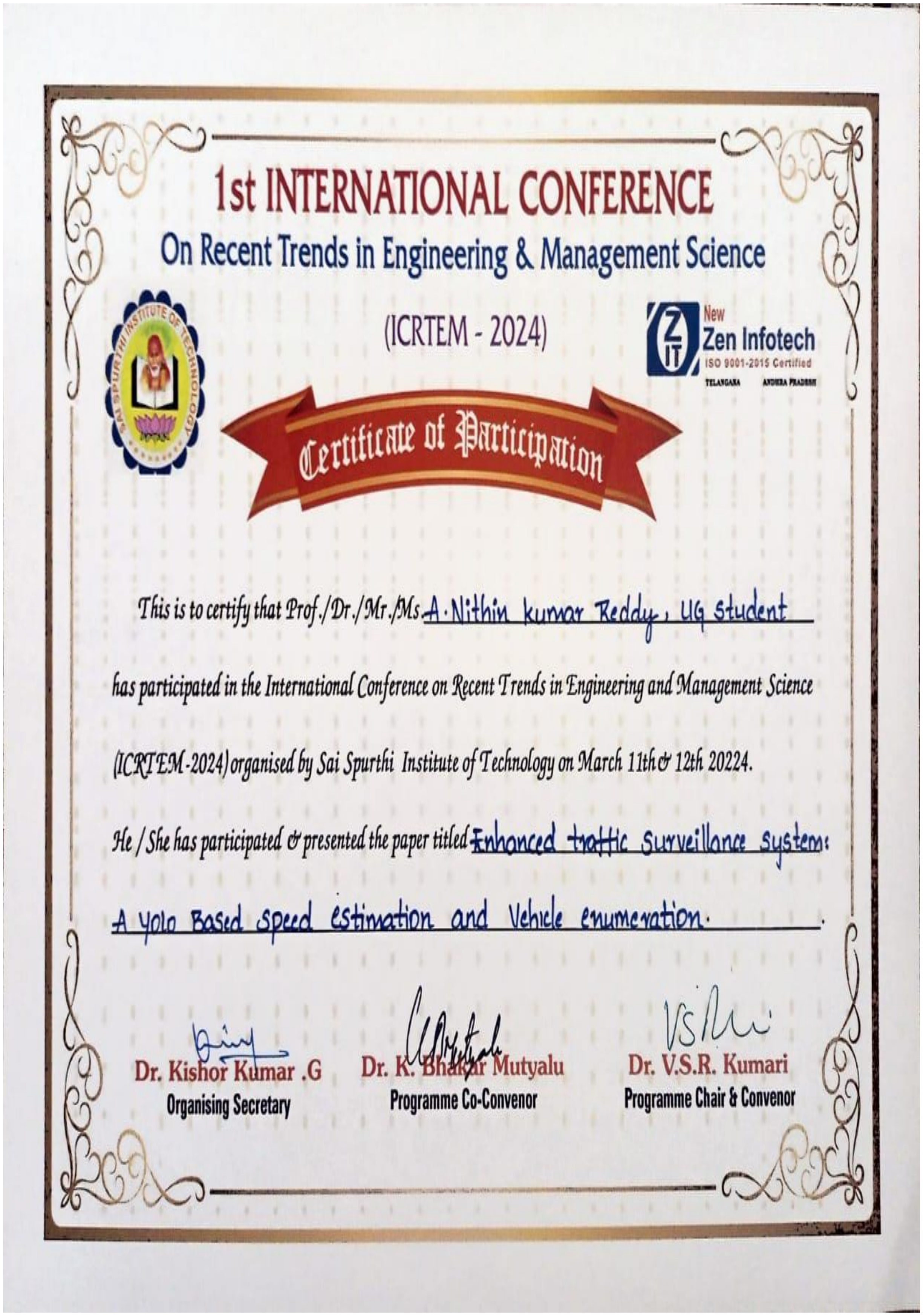
He / She has participated & presented the paper titled Enhanced traffic Surveillance System: A yolo based speed estimation and vehicle enumeration.

[illegible] [illegible] [illegible]

Dr. Kishor Kumar .G
Organising Secretary

Dr. K. Bhakar Mutyalu
Programme Co-Convenor

Dr. V.S.R. Kumari
Programme Chair & Convenor



1st INTERNATIONAL CONFERENCE

On Recent Trends in Engineering & Management Science

(ICRTEM - 2024)

SAI SPURTHI INSTITUTE OF TECHNOLOGY

New Zen Infotech ISO 9001-2015 Certified TELANGANA ANDHRA PRADESH

Certificate of Participation

This is to certify that Prof./Dr./Mr./Ms. A. Nithin kumar Reddy, UG student

has participated in the International Conference on Recent Trends in Engineering and Management Science

(ICRTEM-2024) organised by Sai Spurthi Institute of Technology on March 11th & 12th 20224.

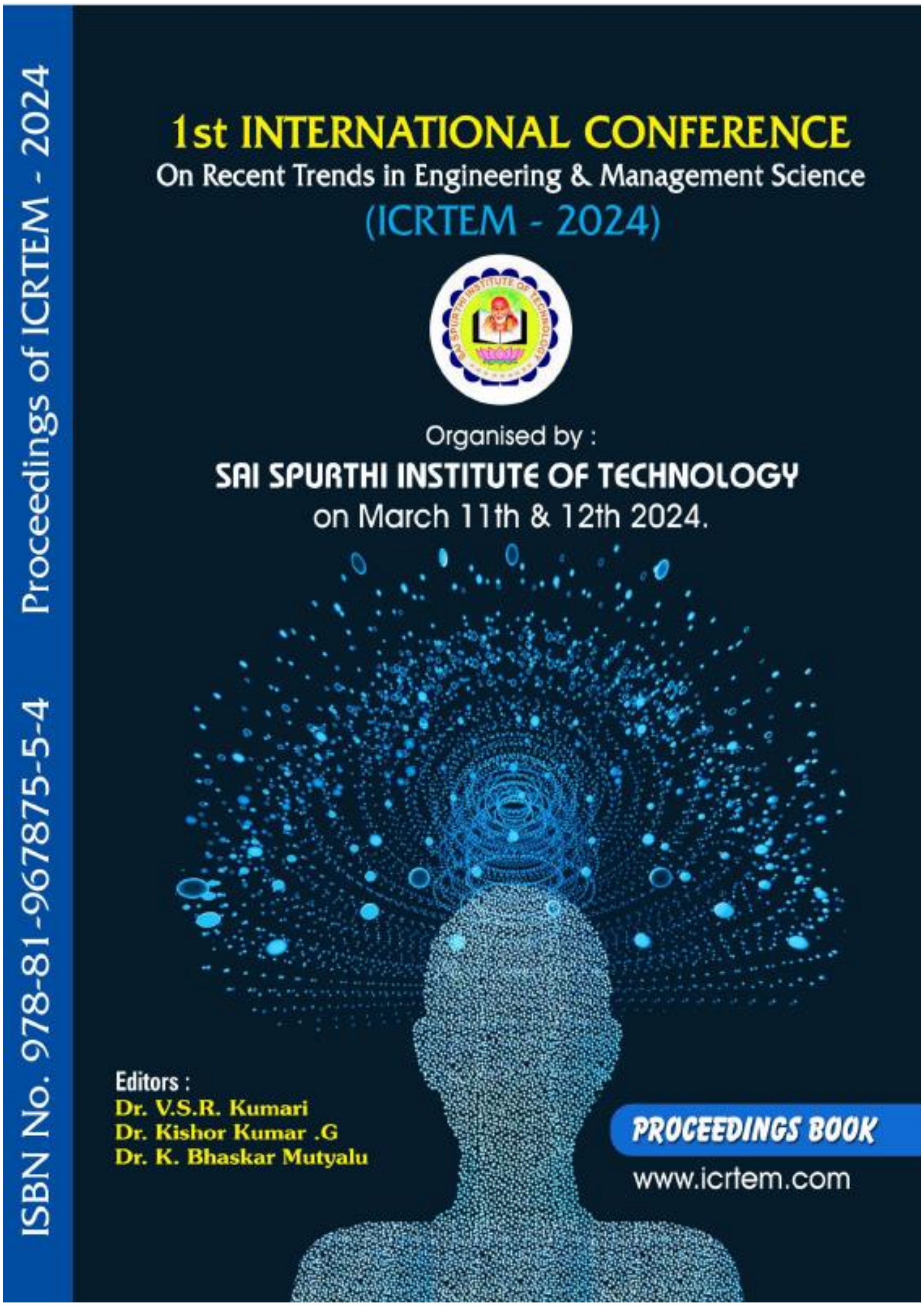
He / She has participated & presented the paper titled Enhanced traffic surveillance system: A yolo Based speed estimation and Vehicle enumeration.

Dr. Kishor Kumar .G
Organising Secretary

Dr. K. Bhaskar Mutyalu
Programme Co-Convenor

Dr. V.S.R. Kumari
Programme Chair & Convenor

PUBLICATION PAPER



1st INTERNATIONAL CONFERENCE

On Recent Trends in Engineering & Management Science

(ICRTEM - 2024)

Organised by :

SAI SPURTHI INSTITUTE OF TECHNOLOGY

on March 11th & 12th 2024.

Editors :

Dr. V.S.R. Kumari

Dr. Kishor Kumar .G

Dr. K. Bhaskar Mutyalu

PROCEEDINGS BOOK

www.icrtem.com

ISBN No. 978-81-967875-5-4 Proceedings of ICRTEM - 2024

1st International Conference
on

Recent Trends in Engineering &

Management Science

(ICRTEM-2024)

11th &12th March 2024, Hybrid Mode

PROCEEDINGS BOOK

Organized by

SAI SPURTHI INSTITUTE OF TECHNOLOGY

SATHUPALLY

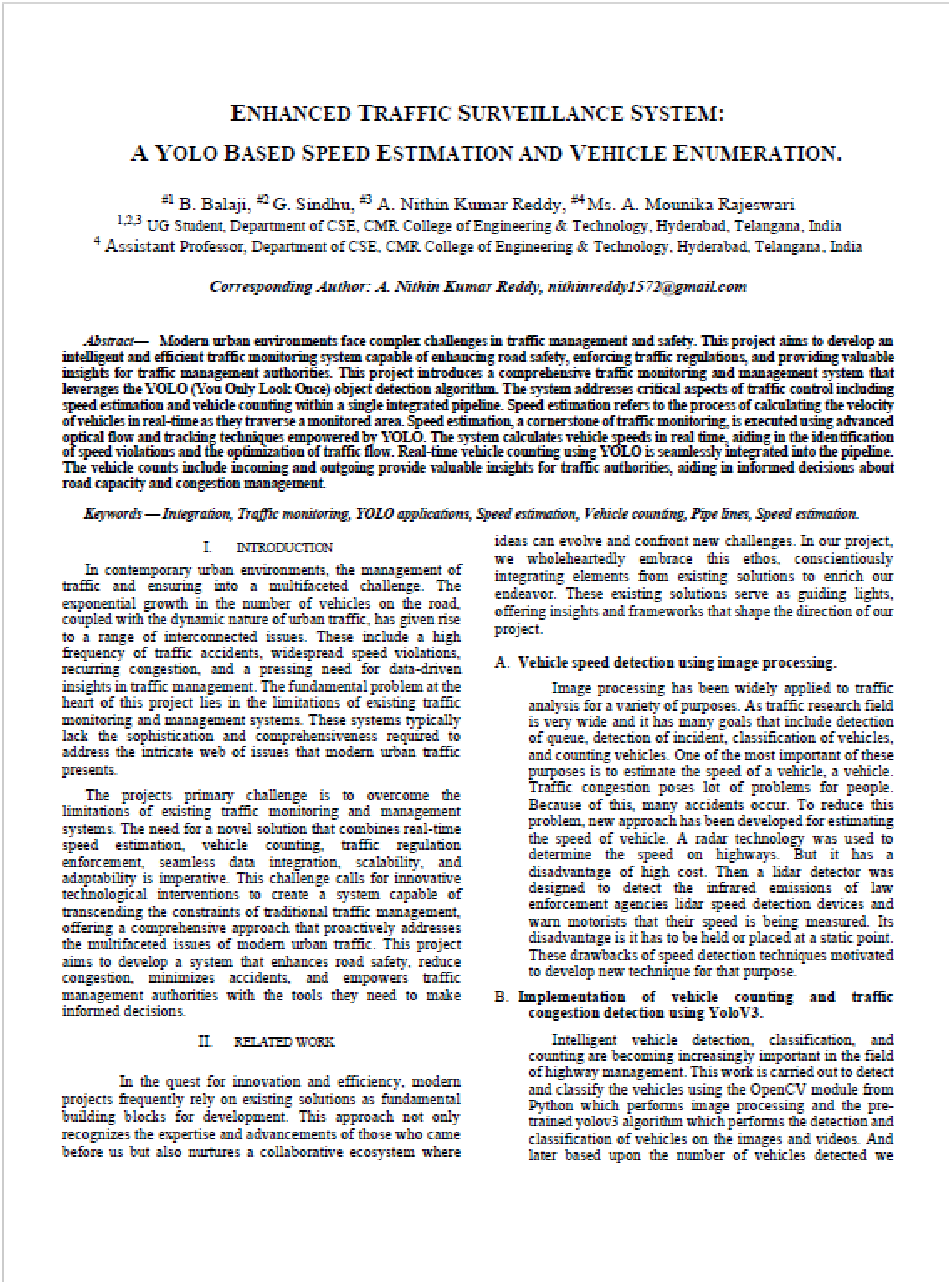
In association with

New Zen Infotech
ISO9001:2015Certified
TELANGANA | ANDHRA PRADESH

NEWZEN INFOTECH, HYDERABAD

www.icrtem.com | www. saispurthi.ac.in

ISBN No: 978-81-967875-5-4



ENHANCED TRAFFIC SURVEILLANCE SYSTEM: A YOLO BASED SPEED ESTIMATION AND VEHICLE ENUMERATION.

#1 B. Balaji, #2 G. Sindhu, #3 A. Nithin Kumar Reddy, #4 Ms. A. Mounika Rajeswari
1,2,3 UG Student, Department of CSE, CMR College of Engineering & Technology, Hyderabad, Telangana, India
4 Assistant Professor, Department of CSE, CMR College of Engineering & Technology, Hyderabad, Telangana, India

Corresponding Author: A. Nithin Kumar Reddy, nithinreddy1572@gmail.com

***Abstract*— Modern urban environments face complex challenges in traffic management and safety. This project aims to develop an intelligent and efficient traffic monitoring system capable of enhancing road safety, enforcing traffic regulations, and providing valuable insights for traffic management authorities. This project introduces a comprehensive traffic monitoring and management system that leverages the YOLO (You Only Look Once) object detection algorithm. The system addresses critical aspects of traffic control including speed estimation and vehicle counting within a single integrated pipeline. Speed estimation refers to the process of calculating the velocity of vehicles in real-time as they traverse a monitored area. Speed estimation, a cornerstone of traffic monitoring, is executed using advanced optical flow and tracking techniques empowered by YOLO. The system calculates vehicle speeds in real time, aiding in the identification of speed violations and the optimization of traffic flow. Real-time vehicle counting using YOLO is seamlessly integrated into the pipeline. The vehicle counts include incoming and outgoing provide valuable insights for traffic authorities, aiding in informed decisions about road capacity and congestion management.**

***Keywords* — Integration, Traffic monitoring, YOLO applications, Speed estimation, Vehicle counting, Pipe lines, Speed estimation.**

I. INTRODUCTION

In contemporary urban environments, the management of traffic and ensuring into a multifaceted challenge. The exponential growth in the number of vehicles on the road, coupled with the dynamic nature of urban traffic, has given rise to a range of interconnected issues. These include a high frequency of traffic accidents, widespread speed violations, recurring congestion, and a pressing need for data-driven insights in traffic management. The fundamental problem at the heart of this project lies in the limitations of existing traffic monitoring and management systems. These systems typically lack the sophistication and comprehensiveness required to address the intricate web of issues that modern urban traffic presents.

The projects primary challenge is to overcome the limitations of existing traffic monitoring and management systems. The need for a novel solution that combines real-time speed estimation, vehicle counting, traffic regulation enforcement, seamless data integration, scalability, and adaptability is imperative. This challenge calls for innovative technological interventions to create a system capable of transcending the constraints of traditional traffic management, offering a comprehensive approach that proactively addresses the multifaceted issues of modern urban traffic. This project aims to develop a system that enhances road safety, reduce congestion, minimizes accidents, and empowers traffic management authorities with the tools they need to make informed decisions.

II. RELATED WORK

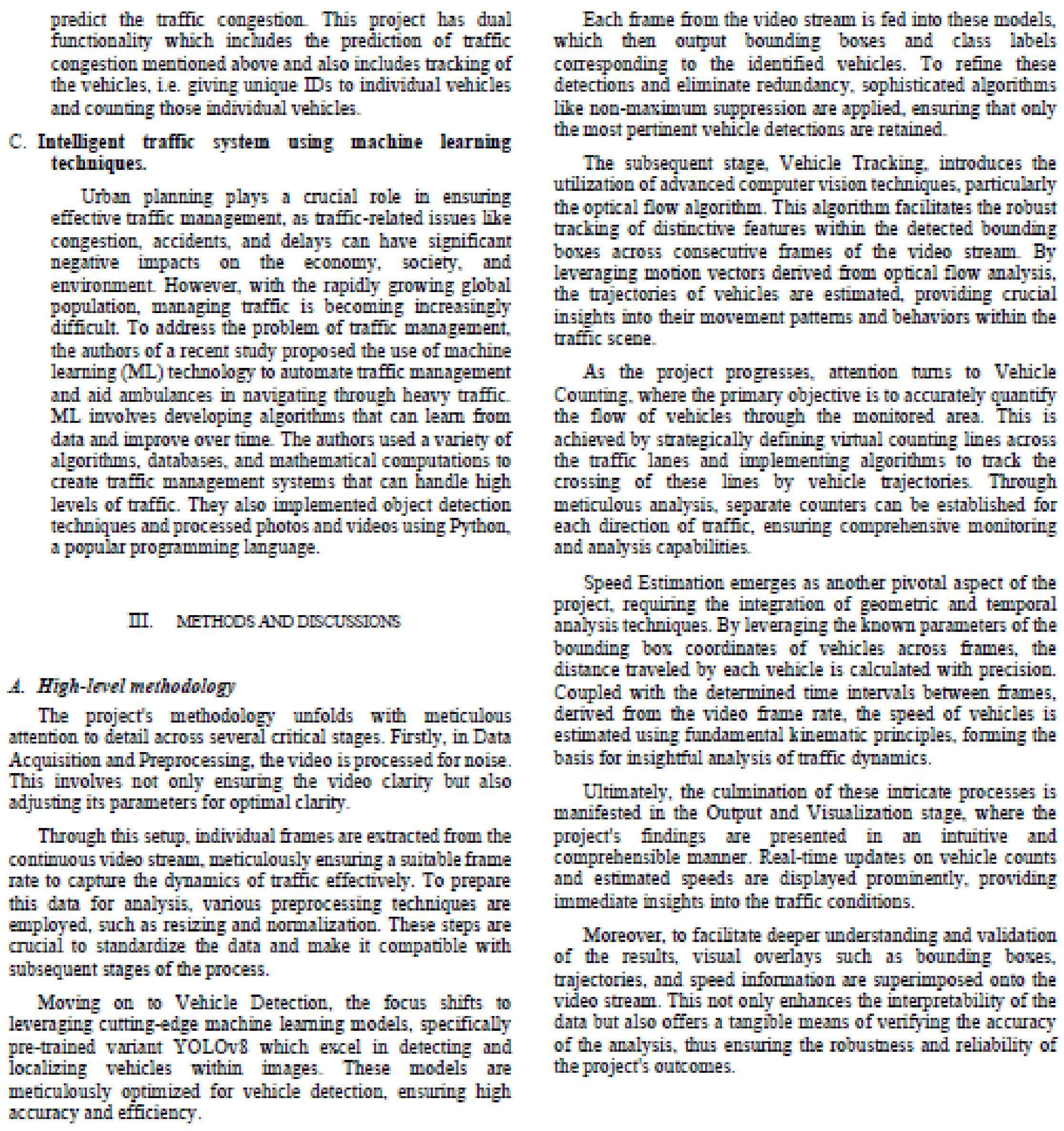
In the quest for innovation and efficiency, modern projects frequently rely on existing solutions as fundamental building blocks for development. This approach not only recognizes the expertise and advancements of those who came before us but also nurtures a collaborative ecosystem where ideas can evolve and confront new challenges. In our project, we wholeheartedly embrace this ethos, conscientiously integrating elements from existing solutions to enrich our endeavor. These existing solutions serve as guiding lights, offering insights and frameworks that shape the direction of our project.

A. Vehicle speed detection using image processing.

Image processing has been widely applied to traffic analysis for a variety of purposes. As traffic research field is very wide and it has many goals that include detection of queue, detection of incident, classification of vehicles, and counting vehicles. One of the most important of these purposes is to estimate the speed of a vehicle, a vehicle. Traffic congestion poses lot of problems for people. Because of this, many accidents occur. To reduce this problem, new approach has been developed for estimating the speed of vehicle. A radar technology was used to determine the speed on highways. But it has a disadvantage of high cost. Then a lidar detector was designed to detect the infrared emissions of law enforcement agencies lidar speed detection devices and warn motorists that their speed is being measured. Its disadvantage is it has to be held or placed at a static point. These drawbacks of speed detection techniques motivated to develop new technique for that purpose.

B. Implementation of vehicle counting and traffic congestion detection using YoloV3.

Intelligent vehicle detection, classification, and counting are becoming increasingly important in the field of highway management. This work is carried out to detect and classify the vehicles using the OpenCV module from Python which performs image processing and the pre-trained yolov3 algorithm which performs the detection and classification of vehicles on the images and videos. And later based upon the number of vehicles detected we



predict the traffic congestion. This project has dual functionality which includes the prediction of traffic congestion mentioned above and also includes tracking of the vehicles, i.e. giving unique IDs to individual vehicles and counting those individual vehicles.

C. **Intelligent traffic system using machine learning techniques.**

Urban planning plays a crucial role in ensuring effective traffic management, as traffic-related issues like congestion, accidents, and delays can have significant negative impacts on the economy, society, and environment. However, with the rapidly growing global population, managing traffic is becoming increasingly difficult. To address the problem of traffic management, the authors of a recent study proposed the use of machine learning (ML) technology to automate traffic management and aid ambulances in navigating through heavy traffic. ML involves developing algorithms that can learn from data and improve over time. The authors used a variety of algorithms, databases, and mathematical computations to create traffic management systems that can handle high levels of traffic. They also implemented object detection techniques and processed photos and videos using Python, a popular programming language.

III. METHODS AND DISCUSSIONS

A. *High-level methodology*

The project's methodology unfolds with meticulous attention to detail across several critical stages. Firstly, in Data Acquisition and Preprocessing, the video is processed for noise. This involves not only ensuring the video clarity but also adjusting its parameters for optimal clarity.

Through this setup, individual frames are extracted from the continuous video stream, meticulously ensuring a suitable frame rate to capture the dynamics of traffic effectively. To prepare this data for analysis, various preprocessing techniques are employed, such as resizing and normalization. These steps are crucial to standardize the data and make it compatible with subsequent stages of the process.

Moving on to Vehicle Detection, the focus shifts to leveraging cutting-edge machine learning models, specifically pre-trained variant YOLOv8 which excel in detecting and localizing vehicles within images. These models are meticulously optimized for vehicle detection, ensuring high accuracy and efficiency.

Each frame from the video stream is fed into these models, which then output bounding boxes and class labels corresponding to the identified vehicles. To refine these detections and eliminate redundancy, sophisticated algorithms like non-maximum suppression are applied, ensuring that only the most pertinent vehicle detections are retained.

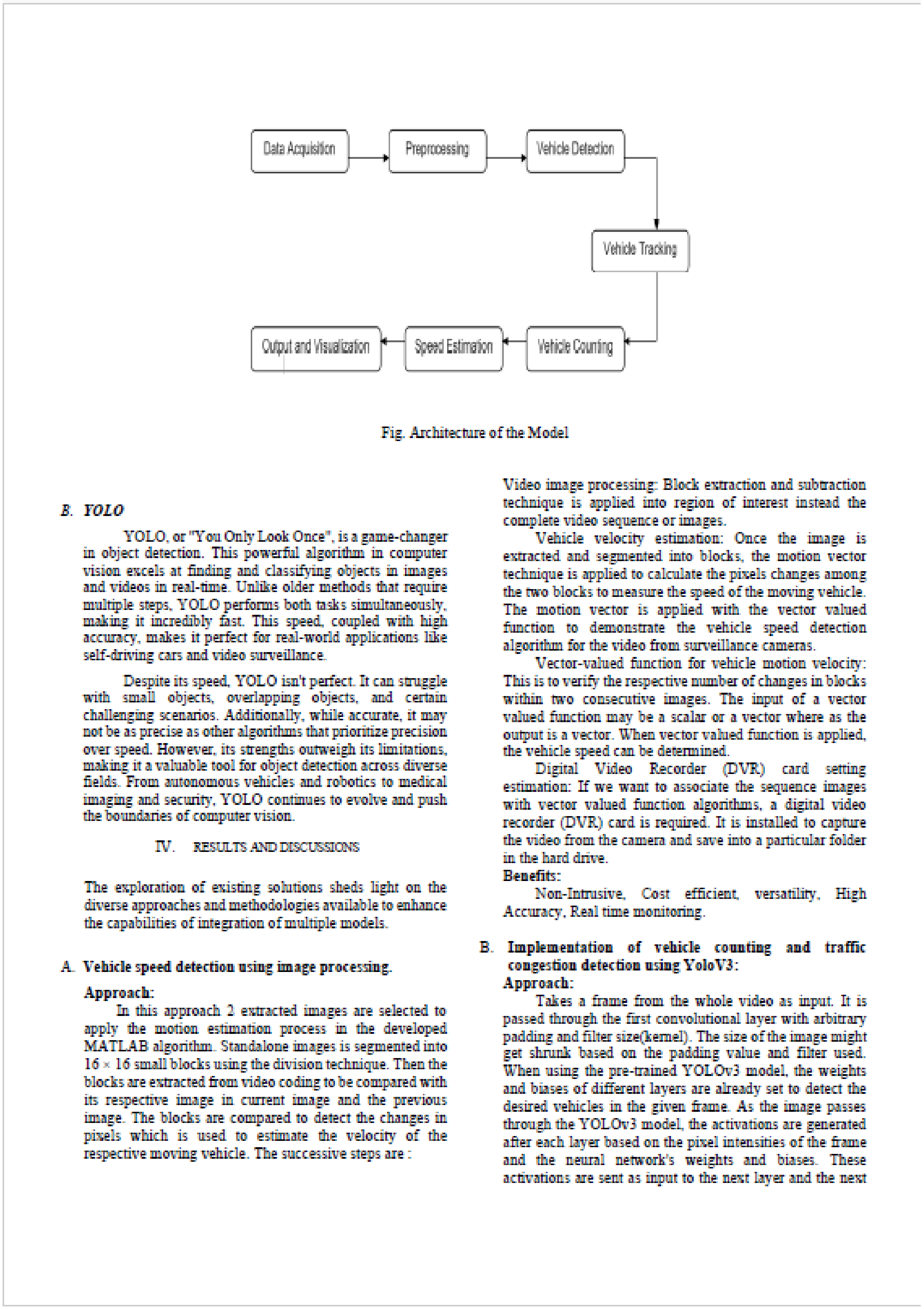
The subsequent stage, Vehicle Tracking, introduces the utilization of advanced computer vision techniques, particularly the optical flow algorithm. This algorithm facilitates the robust tracking of distinctive features within the detected bounding boxes across consecutive frames of the video stream. By leveraging motion vectors derived from optical flow analysis, the trajectories of vehicles are estimated, providing crucial insights into their movement patterns and behaviors within the traffic scene.

As the project progresses, attention turns to Vehicle Counting, where the primary objective is to accurately quantify the flow of vehicles through the monitored area. This is achieved by strategically defining virtual counting lines across the traffic lanes and implementing algorithms to track the crossing of these lines by vehicle trajectories. Through meticulous analysis, separate counters can be established for each direction of traffic, ensuring comprehensive monitoring and analysis capabilities.

Speed Estimation emerges as another pivotal aspect of the project, requiring the integration of geometric and temporal analysis techniques. By leveraging the known parameters of the bounding box coordinates of vehicles across frames, the distance traveled by each vehicle is calculated with precision. Coupled with the determined time intervals between frames, derived from the video frame rate, the speed of vehicles is estimated using fundamental kinematic principles, forming the basis for insightful analysis of traffic dynamics.

Ultimately, the culmination of these intricate processes is manifested in the Output and Visualization stage, where the project's findings are presented in an intuitive and comprehensible manner. Real-time updates on vehicle counts and estimated speeds are displayed prominently, providing immediate insights into the traffic conditions.

Moreover, to facilitate deeper understanding and validation of the results, visual overlays such as bounding boxes, trajectories, and speed information are superimposed onto the video stream. This not only enhances the interpretability of the data but also offers a tangible means of verifying the accuracy of the analysis, thus ensuring the robustness and reliability of the project's outcomes.



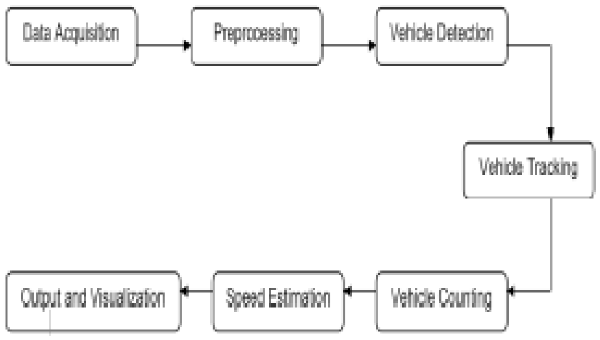


Fig. Architecture of the Model

B. *YOLO*

YOLO, or "You Only Look Once", is a game-changer in object detection. This powerful algorithm in computer vision excels at finding and classifying objects in images and videos in real-time. Unlike older methods that require multiple steps, YOLO performs both tasks simultaneously, making it incredibly fast. This speed, coupled with high accuracy, makes it perfect for real-world applications like self-driving cars and video surveillance.

Despite its speed, YOLO isn't perfect. It can struggle with small objects, overlapping objects, and certain challenging scenarios. Additionally, while accurate, it may not be as precise as other algorithms that prioritize precision over speed. However, its strengths outweigh its limitations, making it a valuable tool for object detection across diverse fields. From autonomous vehicles and robotics to medical imaging and security, YOLO continues to evolve and push the boundaries of computer vision.

IV. RESULTS AND DISCUSSIONS

The exploration of existing solutions sheds light on the diverse approaches and methodologies available to enhance the capabilities of integration of multiple models.

A. Vehicle speed detection using image processing.

Approach:

In this approach 2 extracted images are selected to apply the motion estimation process in the developed MATLAB algorithm. Standalone images is segmented into 16 × 16 small blocks using the division technique. Then the blocks are extracted from video coding to be compared with its respective image in current image and the previous image. The blocks are compared to detect the changes in pixels which is used to estimate the velocity of the respective moving vehicle. The successive steps are :

Video image processing: Block extraction and subtraction technique is applied into region of interest instead the complete video sequence or images.

Vehicle velocity estimation: Once the image is extracted and segmented into blocks, the motion vector technique is applied to calculate the pixels changes among the two blocks to measure the speed of the moving vehicle. The motion vector is applied with the vector valued function to demonstrate the vehicle speed detection algorithm for the video from surveillance cameras.

Vector-valued function for vehicle motion velocity: This is to verify the respective number of changes in blocks within two consecutive images. The input of a vector valued function may be a scalar or a vector where as the output is a vector. When vector valued function is applied, the vehicle speed can be determined.

Digital Video Recorder (DVR) card setting estimation: If we want to associate the sequence images with vector valued function algorithms, a digital video recorder (DVR) card is required. It is installed to capture the video from the camera and save into a particular folder in the hard drive.

Benefits:

Non-Intrusive, Cost efficient, versatility, High Accuracy, Real time monitoring.

B. Implementation of vehicle counting and traffic congestion detection using YoloV3:

Approach:

Takes a frame from the whole video as input. It is passed through the first convolutional layer with arbitrary padding and filter size(kernel). The size of the image might get shrunk based on the padding value and filter used. When using the pre-trained YOLOv3 model, the weights and biases of different layers are already set to detect the desired vehicles in the given frame. As the image passes through the YOLOv3 model, the activations are generated after each layer based on the pixel intensities of the frame and the neural network's weights and biases. These activations are sent as input to the next layer and the next layer in the next step and so on. Finally, in the last layer, the activations in terms of probabilities are evaluated based on all those previous different types of layers (convolutional, pooling, fully connected).

Benefits:

Accurate vehicle counting, Real time processing, High speed interference, versatility, Object localization.

C. **Intelligent traffic system using machine learning techniques:**

Approach:

Collect and annotate a diverse traffic dataset with bounding boxes. Choose between YOLO for vehicle counting. Preprocess data to suit the chosen model's input format. Train the selected model on annotated data. Validate and test model performance on new data. Implement post-processing (e.g., NMS) for accurate counting (YOLO). Deploy the model for real-time inference. Monitor accuracy and retrain as needed. Address privacy and ethical concerns. Scale the system for multiple cameras. Plan for regular system maintenance and monitoring.

Benefits: Real time detection, Single stage detection, God performance on small objects.

Comparison:

Accurate Vehicle counting: The integration of yolo for detection and optical flow for tracking could lead to highly accurate counts, minimizing error rates compared to traditional methods.

Reliable speed estimation: Speed estimations derived from distance and time calculations combined with robust tracking should offer reliable values, valuable for traffic analysis and potential enforcement applications.

Real time data acquisition: The system's ability to process the video streams in real time provides timely insights into traffic patterns and enables immediate responsiveness to changing conditions.

Visualized Insights: Overlaying tracking information and speed measurements on the video stream offers valuable visual feedback for operators and researchers to observe traffic flow dynamics and individual vehicle behavior.

Data-driven Analysis: Collected data can be further analyzed to identify peak traffic times, understand individual vehicle behavior, and inform traffic flow optimization strategies.

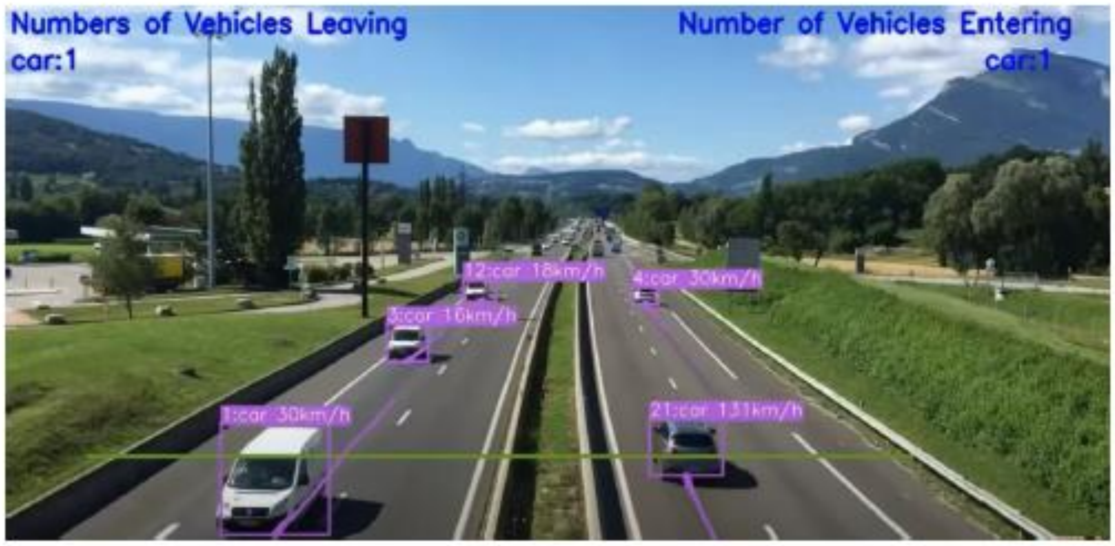


Fig. Detection image 1

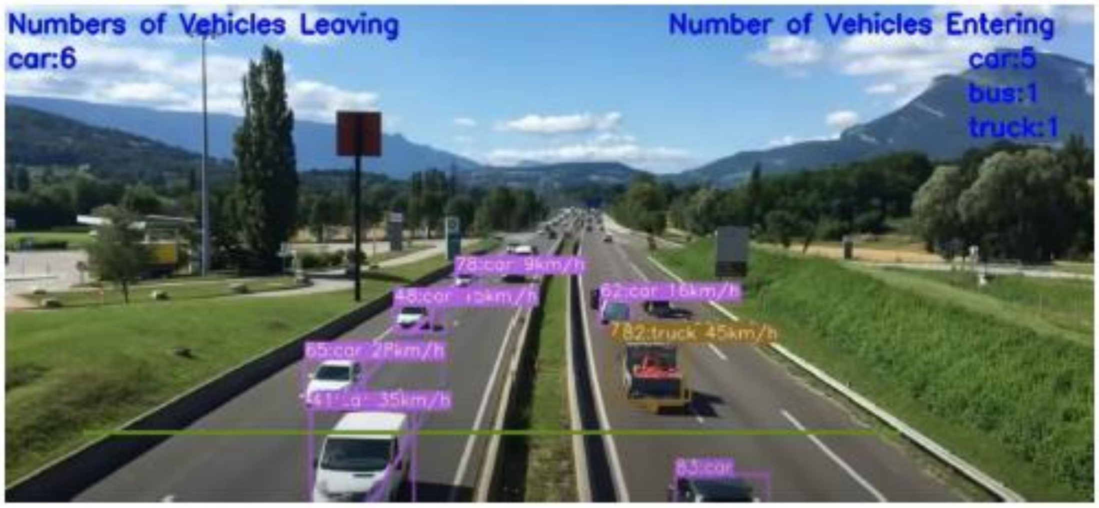


Fig 2: Detection image 2

V. CONCLUSION

This project represents a comprehensive exploration into the synergy between YOLO object detection and optical flow algorithms, with the primary aim of facilitating vehicle counting and speed estimation in traffic scenarios. By combining these advanced technologies, the project endeavors to achieve a multifaceted approach to traffic analysis, encompassing both quantitative metrics and qualitative visual insights.

The integration of YOLO object detection enables precise identification and localization of vehicles within the traffic scene. Leveraging state-of-the-art deep learning techniques, YOLO facilitates the accurate detection of vehicles of various shapes, sizes, and orientations. This capability forms the cornerstone of the project's ability to generate highly accurate vehicle count data, providing stakeholders with a reliable foundation for traffic analysis and management.

Moreover, the incorporation of optical flow algorithms enhances the project's analytical capabilities by enabling robust feature tracking and motion estimation. By tracking key points within the detected vehicle bounding boxes across consecutive frames, the project can discern the trajectories of individual vehicles with high fidelity. This not only contributes to the accuracy of vehicle counting but also enables the estimation of vehicle speeds based on their motion patterns over time.

The potential benefits of this integrated approach are manifold. Firstly, the project promises to deliver highly accurate vehicle count data, which is essential for traffic management, urban planning, and infrastructure development. By providing stakeholders with precise insights into traffic volume and flow patterns, the project empowers decision-makers to implement targeted interventions to alleviate congestion and enhance safety on roadways.

Furthermore, the reliable speed estimation capabilities afforded by the project offer valuable insights into traffic dynamics and behavior. By quantifying the speed of individual vehicles, as well as aggregating speed data across the traffic network, stakeholders can identify potential bottlenecks, assess the impact of interventions, and optimize traffic flow for efficiency and safety.

Beyond its immediate applications, the project holds promise for integration with existing traffic management systems and further data analysis efforts. By seamlessly integrating with established infrastructure and workflows, the project can extend its utility and impact within real-world traffic environments. Additionally, ongoing data analysis and refinement of algorithms offer exciting possibilities for advancing the state-of-the-art in traffic monitoring and research.

In conclusion, this project represents a significant step forward in the quest for smarter and safer transportation systems. By harnessing the power of AI-powered solutions, such as YOLO object detection and optical flow algorithms, the project lays the groundwork for a powerful tool in traffic monitoring and analysis. With its potential to deliver accurate vehicle count data, reliable speed estimation, and valuable visual insights, the project stands poised to make a meaningful contribution to the enhancement of urban mobility and road safety.

VI. REFERENCES

[1] Ammar Awni Abbass University of Baghdad "Estimating vehicle speed using image processing", AL-Mansour Journal / No.14/ Special Issue (Part Two) 2010.

[2] Osman Ibrahim, Hazem ElGendy, and Ahmed M. ElShafee, Member, IEEE " Speed Detection Camera System using Image Processing Techniques on Video Streams ", International Journal of Computer and Electrical Engineering, Vol. 3, No. 6, December 2011.

[3] S. S. S. Ranjit, S. A. Anas, S. K. Subramaniam, K. C. Lim, A. F. I. Fayeez, A. R. Amirah," Real-Time Vehicle Speed Detection Algorithm using Motion Vector Technique ", Proc. of Int. Conf. on Advances in Electrical & Electronics 2012.

[4] Siddharth Jhmnat," Vehicle Speed Estimation in Accident Prone Areas using Image Processing ", International Journal of Advanced Research in Computer and Communication Engineering Vol. 3, Issue 5, May 2014.

[5]https://www.researchgate.net/publication/327337312_A_Review_on_Vehicle_Speed_Detection_using_Image_Processing.