1 INTODUCTION

Vehicle detection and counting are critical aspects in various fields such as transportation management, traffic analysis, and intelligent transportation systems. This system aims to automatically detect and count the number of vehicles passing through a specific area, typically a road or highway.

The system makes use of computer vision algorithms to analyse video footage or images captured by cameras. These algorithms identify and track vehicles in the footage and then count them accurately. This information can be used for various purposes, such as monitoring traffic flow, congestion, and vehicle occupancy levels.

The purpose of a vision-based vehicle detection and counting system using deep learning for highway scenes is to automatically detect and count vehicles in real time in a highway scene using computer vision techniques. The system can be used for various applications such as traffic monitoring, congestion management, and road safety analysis. The purpose of this project is to find the best route (i.e., with less traffic). and one can also know which area is having more traffic and at what time the traffic is more.

The development of vehicle detection and counting systems is driven by the need for efficient and effective traffic management, road safety, and security. These systems use a combination of computer vision techniques, such as image segmentation, object detection, and feature extraction, to analyse video data and identify vehicles.

1) Data Extraction:

Surveillance cameras on roads have been broadly installed worldwide, but traffic pictures are rarely openly released due to copyright, privacy, and security concerns. Therefore, we collected those images from various sources, such as those taken by the car camera and the surveillance camera and those taken by non-monitoring cameras of multiple lighting conditions and different weather conditions.

2) Data Annotation:

Our annotator team started labelling the vehicles by drawing bounding boxes around vehicles such as cars, trucks, bikes, etc. Then the verification team verified the

annotated images whether the vehicles in the images were tagged correctly or not; if not, then again, those images are assigned to the annotation team.

3) Data Processing:

In the real world, images are full of noise. Because of camera quality, and weather conditions, we can't take pictures without noise most of the time. Therefore, before developing an AI model, we should eliminate that noise from the image. Otherwise, the model will learn the wrong patterns from noise images. We are applying noise removal techniques to get denoised images. We also used edge detection methods to extract the shape of the vehicle.

4) Model Development:

vehicle detection and counting systems are almost new compared to their non-video-based counterparts. This reason is mainly due to image processing and systems infrastructure improvements over the past two decades. As a result, there's a lot of research on techniques and algorithms for video-based vehicle detection and counting. These methods majorly consist of detection, tracking, and counting techniques. For a Vision-based system to work, it has first to identify vehicles driving on the road. Next, it has to track their journey until they leave the frame of the video-capturing stream. This method is essential to avoid double-counting vehicles. The last step is counting, which has to occur before a vehicle leaves the video frame.

i) Detection

Object detection is a computer vision and image processing technique that identifies objects of a specific class, like vehicles or people, in videos. Object detection can solve complex real-world problems in different areas like image search and video surveillance. In addition, it is utilized widely in computer vision tasks, including face detection, face recognition, and object tracking.

The vehicle detector identifies vehicles in a given frame or image. It can detect various vehicle types, including cars, buses, trucks, motorbikes, and bicycles, and delivers a list of bounding boxes for all identified vehicles. The bounding box consists of the x-y coordinate of a vehicle as well as its width and height.

ii) Tracking

Tracking is the process of following the path or movements of an object to find it or observe its course using a camera. The uses of video tracking include surveillance, security, and traffic control, etc. The vehicle detection and counting system tracker receives a set of bounding boxes that identify regions of interest (ROI) to track. Then, it generates blobs from each bounding box and begins following them across the video frame.

iii) Counting

Counting is the final step which includes determining the number of vehicles that have moved at any given period. Vehicle counts may be noted on the counting device or forwarded to a remote location over the internet.

Vehicle detection and counting system can have various scopes and applications such as

1 Traffic monitoring and management:

One of the most common uses of vehicle detection and counting systems is to monitor and manage traffic flow, by counting the number of vehicles passing through a specific location, and adjusting traffic lights and signs accordingly.

2 Toll collection:

Vehicle detection and counting systems can be used at toll booths to automatically collect tolls without requiring the driver to stop or roll down their windows.

3 Parking management:

These systems can be used to monitor the occupancy of parking lots and garages, and to help drivers find available spaces

4 Law enforcement:

Vehicle detection and counting systems can assist law enforcement in monitoring and controlling traffic flow, as well as detecting and preventing illegal activities such as speeding, tailgating, and running red lights.

5 Road safety:

These systems can help in promoting road safety by detecting and counting vehicles, and analysing traffic patterns to identify and reduce accidents.

The use of cameras and other image acquisition devices, together with advanced image processing algorithms, has made it possible to implement real-time vehicle detection and counting systems that are capable of accurately detecting and counting vehicles in complex and dynamic environments.

Recent advancements in machine learning and deep learning techniques have also led to significant improvements in the accuracy and efficiency of vehicle detection and counting systems, making them more reliable and scalable for real-world deployment.

Surveillance cameras in roads have been widely installed worldwide but traffic images are rarely released publicly due to copyright, privacy, and security issues. From the image acquisition point of view, the traffic image dataset can be divided into three categories: images taken by the car camera, images taken by the surveillance camera, and images taken by non-monitoring cameras. The dataset divided the vehicles into cars, truck, bus and tempos; however, the shooting angle is positive, and the vehicle object is too small for each image, which is difficult to generalize for CNN training. The traffic and congestions (TRANCOS) dataset contain pictures of vehicles on highways captured by surveillance cameras and contains images. Most of the images have some occlusion. This dataset has small number of pictures, and no vehicle type is provided, which makes it less applicable. Therefore, few datasets have useful annotations, and few images are available in traffic scenes. The dataset picture is from the highway monitoring video of India (Fig. 1). The highway monitoring camera was installed on the roadside and placed at 12 meters; it had an adjustable field of view and no present position. The images from the perspective ever the far distance of the highway and contains vehicles with dramatic changes in scale. The dataset images were captured from different surveillance cameras for different scenes, different times, and different conditions. This dataset divides the vehicle into categories: cars, bus, tempo, truck (Fig. 2). The label file is stridden in a text document that contains the numeric code of the object category and the normalized coordinate values of the bonding box. As shown in (Fig. 3), this dataset has a 612 annotation boxes. The images have an RGB format and 1920*1080 resolution. Note that we annotated the smaller objects in the proximal road area, and the dataset thus contains vehicle objects with massive scale

changes. An annotated instance near the camera has more features, and an instance far from the camera has few features. Annotated instances of different sizes are beneficial to the improvement of the detection accuracy of small vehicle objects. This dataset is divided into two parts: a training set and a test set. In each image on average Figure compares the difference between the number of annotated instances in our dataset of COCO datasets. Our dataset is a universal dataset for vehicle targets that can be used in a variety of areas. With the exiting vehicle datasets, our dataset has a large number of images, sufficient lighting conditions, and complete annotation.

Fig.1 Scenes taken by multiple highway surveillance cameras

(a) scene 1 (b) scene 2 (c) scene 3

Fig.2 label images of vehicles

Bus	17 0.485703 0.283854 0.857843 0.473958
Car	7 0.507812 0.476562 0.984375 0.921875
Tempo	20 0.481445 0.441162 0.618490 0.672363
Truck	5 0.576185 0.403799 0.411356 0.302083
Auto	4 0.427500 0.518750 0.324333 0.336000
Bike	7 0.364918 0.156371 0.324289 0.284157

Table. 1 YOLO Image trained values where the vehicle is present

Bus	17 1.485703 0.283854 0.857843 0.473958
Car	7 1.507812 0.476562 0.984375 0.921875
Tempo	20 1.481445 0.441162 0.618490 0.672363
Truck	5 1.576185 0.403799 0.411356 0.302083
Auto	4 0.427500 0.518750 0.324333 0.336000
Bike	7 1.364918 0.156371 0.324289 0.284157

Table. 2 YOLO Image trained values where the vehicle is not present

1.1 Background of vehicle detection

1) Vision-based vehicle detection and counting systems are a type of computer vision technology that uses cameras or other visual sensors to detect and track vehicles on the road. These systems have a variety of applications, including traffic monitoring, toll collection, and intelligent transportation systems.
2) Deep learning is a subfield of machine learning that uses neural networks to learn patterns and features from data. Deep learning has revolutionized computer vision and has led to significant advances in object detection and recognition.
3) Highway scenes are a particularly challenging environment for vision-based vehicle detection and counting systems, as they typically involve high-speed vehicles and complex traffic patterns. To address these challenges, deep learning models are often used to detect and track vehicles in real-time.
4) One popular approach for vehicle detection and counting in highway scenes is the use of convolutional neural networks (CNNs). CNNs are a type of deep neural network that are particularly well-suited for image recognition tasks. These

models can be trained on large datasets of annotated images to learn to detect and classify vehicles in real-world scenes.

5) Other approaches for vehicle detection and counting in highway scenes include using object tracking algorithms to track vehicles over time and using stereo vision techniques to estimate the depth and position of vehicles in 3D space.
6) Overall, vision-based vehicle detection and counting systems using deep learning have the potential to significantly improve traffic management and safety on highways, and are an active area of research and development in computer vision and artificial intelligence.

1.2 Motivation

1) The motivation behind vision-based vehicle detection and counting systems using deep learning in highway scenes is to improve traffic management, safety, and efficiency.
2) Accurate and real-time vehicle detection and counting is essential for a variety of applications, such as traffic monitoring, congestion management, toll collection, and incident detection. Traditional methods for vehicle detection and counting, such as loop detectors and video-based methods, have limitations in terms of accuracy, reliability, and scalability.
3) Deep learning techniques, particularly convolutional neural networks (CNNs), have shown great promise in addressing these limitations by achieving high accuracy and robustness in detecting and tracking vehicles in complex highway scenes. By using deep learning, these systems can learn to recognize and differentiate between different types of vehicles, even in challenging lighting and weather conditions.
4) Moreover, real-time processing capability of deep learning models can allow for immediate response and alerts in case of accidents, congestion, or other incidents.
5) Overall, vision-based vehicle detection and counting systems using deep learning have the potential to significantly improve the safety and efficiency of

highways, making them an important area of research and development in computer vision and artificial intelligence.

1.3 Overview Of Exiting system

- InterVision is a computer vision software solution that is used for video analytics and surveillance. The software works by analysing video streams in real-time and identifying objects and events within the video.
- InterVision uses machine learning algorithms to analyse video footage and identify objects, such as people, vehicles, and other items. The software is trained on large amounts of data to recognize patterns and identify objects with a high degree of accuracy.
- The software can detect and track objects as they move through the video stream, allowing for real-time monitoring of activity. InterVision can also be customized to recognize specific events or behaviours, such as loitering, theft, or suspicious activity, and can trigger alerts when these events occur.
- Overall, InterVision is designed to provide advanced video analytics and surveillance capabilities to improve safety and security in a variety of settings, including airports, retail stores, and public spaces.

Disadvantages

a. **Cost:** InterVision's software solutions may be expensive, especially for small or medium-sized businesses. The cost of the software can vary depending on the specific features and capabilities required, as well as the size and complexity of the deployment.

b. **Hardware requirements**: InterVision's software solutions may require high-performance hardware, such as powerful CPUs or GPUs, to process large amounts of video data in real-time. This can add to the cost of the overall deployment and may require additional resources for maintenance and upgrades.

c. **Complexity:** InterVision's software solutions can be complex to deploy and configure, requiring specialized knowledge and expertise in computer vision and video analytics. This can make it challenging for businesses without these resources to implement and maintain the software.

d. **False positives**: Like any computer vision-based system, InterVision's software solutions may produce false positives, such as incorrectly identifying objects or behaviours as suspicious or abnormal. This can lead to unnecessary alerts or false alarms, which can be frustrating for security personnel and can reduce the overall effectiveness of the system.

e. **Privacy concerns:** InterVision's facial recognition and license plate recognition features may raise privacy concerns, as they involve collecting and analysing personal data. Businesses may need to take steps to ensure that they are complying with relevant privacy regulations and that they are using the technology in a responsible and ethical manner.

1.4 Proposed system

Data collection: The first stage involves collecting a dataset of highway scene images or videos. The dataset should include a diverse range of traffic scenarios, including different lighting conditions, weather conditions, and traffic densities.

Object detection using YOLO: YOLO is a real-time object detection system that uses a single neural network to predict bounding boxes and class probabilities for multiple objects in an image. The YOLO model is trained on the collected dataset to detect and classify vehicles in real-time.

Object tracking using ORB: Once vehicles are detected, ORB is used to track them over time. ORB is a feature detection and matching algorithm that can track objects even when they move out of frame or are partially obscured. ORB is trained on the detected vehicle images to track them over time and to estimate their position and velocity.

Vehicle counting: The tracked vehicles are then counted to estimate the traffic flow on the highway. The counting can be done by simply incrementing a counter every time a vehicle passes a certain point on the highway.

System evaluation: The proposed system is evaluated on a test dataset to measure its accuracy and performance. The evaluation can be done by comparing the detected and tracked vehicles with ground truth data, such as manually annotated vehicle positions.

1.5 Overview of proposed system and Limitations

The system uses deep learning techniques to achieve high accuracy and robustness in detecting and tracking vehicles in complex highway scenes. YOLO is a popular object detection system that uses a single neural network to predict bounding boxes and class probabilities for multiple objects in an image. ORB is a feature detection and matching algorithm that can track objects even when they move out of frame or are partially obscured.

The proposed system has the potential to improve traffic management and safety on highways by providing real-time information about traffic flow and congestion. It can also be used for toll collection, incident detection, and other applications related to highway traffic management

Limitations

- Adverse weather conditions, such as rain, fog, and snow, can significantly impact the accuracy of the system. These conditions can affect the visibility of the vehicles and their features, making it difficult for YOLO and ORB to detect and track them accurately.
- In crowded highway scenes, where multiple vehicles are close to each other, YOLO and ORB may have difficulty in distinguishing between individual vehicles, leading to missed detections and tracking errors.

- The system may have difficulty generalizing to different environments and scenarios that are not included in the training dataset. This can lead to poor performance and reduced accuracy.
- The system may have limited scalability, particularly when processing large volumes of data in real-time. This can lead to processing delays and increased computational requirements.
- The system may have limited capability in interpreting the results, such as distinguishing between different vehicle types or identifying specific vehicles of interest, such as emergency vehicles.

1.6 Aim and Purpose of the project

- The main objective for developing this system is to collect vehicle count and classification data and also track and count the detected vehicle when they leave the frames or makes use of a counting line drawn across a road.
- The purpose of the project vision-based vehicle detection and counting system using deep learning in highway scenes is to develop a computer vision system that can detect and track vehicles in real-time on highways, and estimate the traffic flow by counting the tracked vehicles. The system aims to improve traffic management and safety on highways by providing real-time information about traffic flow and congestion.

- The project uses deep learning techniques, specifically YOLO (You Only Look Once) and ORB (Oriented FAST and Rotated BRIEF), to achieve high accuracy and robustness in detecting and tracking vehicles in complex highway scenes. The system can be used for toll collection, incident detection, and other applications related to highway traffic management.

- The project is important because it addresses a critical need for efficient and accurate traffic monitoring systems that can help reduce traffic congestion, improve safety, and enhance the overall efficiency of transportation

systems. The development of such a system has significant implications for transportation management and infrastructure planning.

1.7 Scope of the project

1) **Traffic monitoring and management:** One of the most common uses of vehicle detection and counting systems is to monitor and manage traffic flow, by counting the number of vehicles passing through a specific location, and adjusting traffic lights and signs accordingly.
2) **Toll collection:** Vehicle detection and counting systems can be used at toll booths to automatically collect tolls without requiring the driver to stop or roll down their windows.
3) **Parking management:** These systems can be used to monitor the occupancy of parking lots and garages, and to help drivers find available spaces
4) **Law enforcement:** Vehicle detection and counting systems can assist law enforcement in monitoring and controlling traffic flow, as well as detecting and preventing illegal activities such as speeding, tailgating, and running red lights.
5) **Road safety:** These systems can help in promoting road safety by detecting and counting vehicles, and analysing traffic patterns to identify and reduce accidents. The use of cameras and other image acquisition devices, together with advanced image processing algorithms, has made it possible to implement real-time vehicle detection and counting systems that are capable of accurately detecting and counting vehicles in complex and dynamic environments. Recent advancements in machine learning and deep learning techniques have also led to significant improvements in the accuracy and efficiency of vehicle detection and counting systems, making them more reliable and scalable for real-world deployment.

1.8 Objectives

- To monitor traffic flow, identify congested areas, and measure the effectiveness of traffic management strategies.
- To estimate the demand for transportation services, such as bus and train schedules, and to plan for future infrastructure projects.
- To monitor road safety, identify potential accident hotspots, and detect suspicious or illegal activity.
- To gather data on customer traffic and behaviour, and to optimize marketing and advertising strategies.
- To monitor traffic violations, such as speeding and illegal parking, and to assist with investigations and enforcement.

2 SYSTEM REQUIREMENT SPECIFICATION

2.1 Purpose of the system

- Vehicle detection and counting are critical aspects in various fields such as transportation management, traffic analysis, and intelligent transportation systems. This system aims to automatically detect and count the number of vehicles passing through a specific area, typically a road or highway.

- The system makes use of computer vision algorithms to analyse video footage or images captured by cameras. These algorithms identify and track vehicles in the footage and then count them accurately. This information can be used for various purposes, such as monitoring traffic flow, congestion, and vehicle occupancy levels.

- The purpose of a vision-based vehicle detection and counting system using deep learning for highway scenes is to automatically detect and count vehicles in real time in a highway scene using computer vision techniques. The system can be used for various applications such as traffic monitoring, congestion management, and road safety analysis. The purpose of this project is to find the best route (i.e., with less traffic). and one can also know which area is having more traffic and at what time the traffic is more.

- The development of vehicle detection and counting systems is driven by the need for efficient and effective traffic management, road safety, and security. These systems use a combination of computer vision techniques, such as image segmentation, object detection, and feature extraction, to analyse video data and identify vehicles.

2.2 Feasibility analysis

Feasibility analysis in vehicle detection and counting using YOLO (You Only Look Once) and ORB (Oriented FAST and Rotated BRIEF) would involve evaluating the viability and suitability of implementing this specific technology for vehicle detection and counting. Here are some factors that would be considered in a feasibility analysis for this type of system:

Technical feasibility: The YOLO algorithm uses deep learning neural networks to detect objects in real-time, while ORB is a feature detection and description algorithm used for object recognition and tracking. Both YOLO and ORB have been proven effective in object detection and tracking, but their performance may vary depending on the type of road infrastructure, traffic conditions, and camera angles. Technical feasibility would involve testing the technology on real-world scenarios to evaluate its effectiveness.

Economic feasibility: Implementing the YOLO-ORB system would require the purchase of hardware, such as cameras and computers, as well as the cost of development, implementation, and maintenance. The cost of the system would need to be weighed against the potential benefits, such as improved traffic management and reduced congestion.

Operational feasibility: The YOLO-ORB system would need to be compatible with existing traffic management systems and must not disrupt traffic flow. The system must be user-friendly and require minimal training for operators.

Legal and regulatory feasibility: The collection and use of vehicle data must comply with all applicable laws and regulations, such as data protection and privacy laws. The YOLO-ORB system must be transparent and ethical in the collection and use of vehicle data.

Schedule feasibility: The development and implementation of the YOLO-ORB system must be completed within a reasonable timeframe and must not disrupt traffic flow.

2.3 Hardware requirements

1. Processor: Pentium IV onwards
2. RAM: 2 GB or higher
3. Hard Disk Space: 50 GB or higher
4. Input: Video Sequence

2.4 Software requirements

1. Operating System: Windows 11 or higher
2. Software: Python 3.10.10
3. IDLE: PyCharm

2.5 Functional requirements

1) **Real-time vehicle detection and counting**: The system should be able to detect and count vehicles in real-time using deep learning models.
2) **Vehicle classification:** The system should be able to classify vehicles by type, such as cars, trucks, motorcycles, or buses.
3) **Real-time traffic information**: The system should be able to provide real-time traffic information to highway management systems, including the number of vehicles on the road, speed, and congestion.
4) **Traffic incident detection**: The system should be able to detect traffic incidents, such as accidents or breakdowns, by analysing changes in traffic patterns.
5) **Integration with toll collection systems**: The system should be able to integrate with toll collection systems to automatically detect and count vehicles passing through toll booths.
6) **Autonomous vehicle integration**: The system should be able to provide real-time information about traffic flow and behaviour to autonomous vehicles, enabling them to make more informed decisions about navigation and safety.
7) **Infrastructure monitoring:** The system should be able to monitor the condition of highways and bridges, including detecting cracks, potholes, and other damage that may require repair or maintenance.
8) **Scalability:** The system should be able to handle large amounts of data and traffic flow.

9) **Accuracy and reliability**: The system should be accurate and reliable in detecting and counting vehicles, classifying vehicles, and detecting traffic incidents.
10) **User interface**: The system should have a user-friendly interface for highway management systems to access and interpret the real-time traffic information.
11) **Integration with other systems**: The system should be able to integrate with other traffic management systems, such as traffic cameras, traffic sensors, and weather monitoring systems.
12) **Maintenance and support**: The system should have a maintenance and support plan in place to ensure its smooth operation.

2.6 Non-functional requirements

Non-functional requirements in a vehicle detection and counting system refer to the qualities or characteristics of the system that are not related to its primary function or behaviour but are important for its overall performance and user experience.

1) **Performance:** The system should be able to process input images or video streams in real-time and provide accurate detection and counting results within a reasonable time frame.

2) **Accuracy:** The system should have a high accuracy rate in detecting and counting vehicles, with a low rate of false positives or false negatives.

3) **Scalability:** The system should be able to handle a large volume of input data and be scalable to accommodate future growth and expansion.

4) **Usability:** The system should be user-friendly and easy to use for operators, with a simple and intuitive user interface.

5) **Reliability:** The system should be reliable and available for use at all times, with minimal downtime or service disruptions.

6) **Security:** The system should have robust security features to prevent unauthorized access and protect sensitive data, such as images or videos of vehicles.

7) **Compatibility:** The system should be compatible with a wide range of hardware and software platforms, including different types of cameras, operating systems, and web browsers.

8) **Maintainability:** The system should be easy to maintain and upgrade, with a modular and well-documented codebase and clear instructions for troubleshooting and maintenance.

9) **Accessibility:** The system should be accessible to users with disabilities, with support for assistive technologies such as screen readers or voice recognition software.

10) **Performance under stress:** The system should be able to maintain its performance even under high stress conditions, such as heavy traffic or extreme weather conditions.

3 ABOUT THE SOFTWARE

Python 3.10.10

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured, object-oriented and functional programming.

PyCharm

PyCharm Community is a free and open-source integrated development environment (IDE) for Python programming language. It is developed by JetBrains, a software development company based in Prague, Czech Republic.

PyCharm Community is built on top of the IntelliJ IDEA platform, which is also developed by JetBrains. It provides a range of features for developing Python applications, including code analysis, code completion, debugging, version control integration, and more.

PyCharm Community is available for Windows, Mac, and Linux operating systems. It can be downloaded and installed from the JetBrains website or through the package manager of the respective operating system.

PyCharm Community is open source, which means that the source code is freely available for anyone to view, modify, and distribute under the terms of the Apache License 2.0. Contributions to the PyCharm Community project are welcome from the community, and the project is maintained by a team of developers from JetBrains and the open-source community.

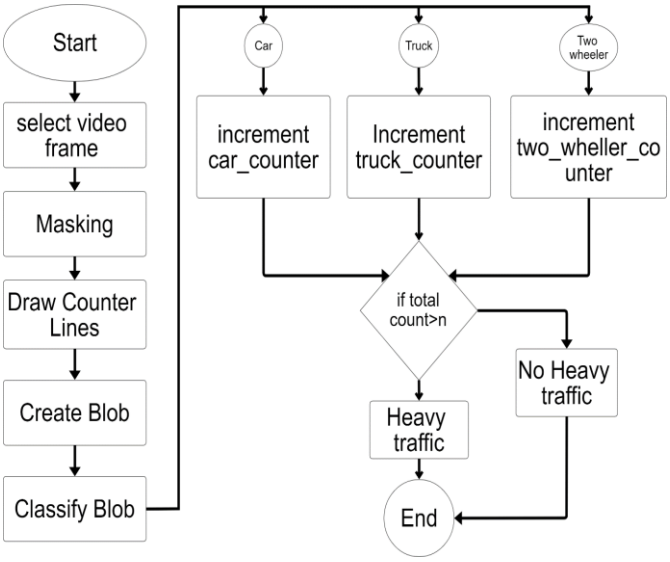
The PyCharm Community project also has a community forum where users can ask questions, share their knowledge, and get support from other users and developers. Additionally, PyCharm Community has extensive documentation, including tutorials, user guides, and API references, to help users get started and make the most of the IDE.

4 SYSTSEM ANALYSIS AND REQUIRMENTS DOCUMENTATION

4.1 Overview

In order to identify, monitor, and count cars moving through a specified area, vehicle detection and counting systems use a combination of hardware and software. These systems' data collection capabilities can be utilised to better public safety, traffic flow, and transportation planning. To maintain the system's correct operation and accurate data, regular maintenance and upgrades are required.

4.2 Proposed system architecture



vehicle detection, tracking, and counting, depending on the specific requirements and constraints of the application.

Input: The system begins by gathering data from cameras or sensors, such lidar or radar, placed in key areas, like as junctions of traffic or beside roadways.

Pre-processing: After that, pre-processing is used to improve the quality of the image or data by removing noise and fixing deformities. Applying filters, such median filters, or converting the data into a different domain, like switching from image to feature space, may be necessary to do this.

Object detection using YOLO: The pre-processed data is fed into an object detection algorithm, such as YOLO (You Only Look Once) to identify vehicles in the scene. The algorithm outputs a set of bounding boxes, each representing a vehicle.

Object tracking using ORB: Once the vehicles have been detected, the system needs to track their movements over time. This can be done using an algorithm such as Kalman filter or particle filter, which predict the location of each vehicle in the next frame based on its previous motion and update its position as new frames arrive.

Counting of vehicles: The final step is to count the number of vehicles passing through a specific location or region of interest. This can be done by analysing the trajectories of the tracked vehicles and identifying when they cross a line or enter/exit a specific area.

Output: The system generates an output, which can be displayed on a screen, sent to a control centre, or used for statistical analysis.

4.3 Module Description

The presents algorithms for vision-based detection and classification of vehicles in monocular image sequences of traffic scenes recorded by a stationary camera. Processing is done at three levels: raw images, region level, and vehicle level. Vehicles are modelled as rectangular patches with certain dynamic behaviour. The proposed method is based on the establishment of correspondences between regions and vehicles, as the vehicles move through the image sequence. Experimental results from highway scenes are provided which demonstrate the effectiveness of the method.

Background subtraction is a widely used paradigm to detect moving vehicles in video taken from a static camera and is used for various

important applications such as video surveillance, human motion analysis, etc. Various statistical approaches have been proposed for modelling a given scene background. However, there is no theoretical framework for choosing which features to use to model different regions of the scene background. They introduce a novel framework for feature selection for background modelling and subtraction. A boosting algorithm, namely Real Boost, is used to choose the best combination of features at each pixel. Given the probability estimates from a pool of features calculated by Kernel Density Estimate (KDE) over a certain time period, the algorithm selects the most useful ones to discriminate foreground vehicles from the scene background. The results show that the proposed framework successfully selects appropriate features for different parts of the image.

In the past 3 decades satellite imaging has been used with success for geographical, weather, and geological applications. With the advancement of technology, additional refined sensors offer higher resolutions, and with quicker computer systems, the employment of satellite imaging has opened the fields of application and exploration. Segmentation techniques supported thresholding are used to extract highways and vehicles from pictures containing roadways scenes. Colour properties are accustomed to extract vegetation areas from cities and fields scenes. Results of this work might be used to assist transportation agencies within the project of traffic density and trends across huge geographic areas.

Image can easily be acquired with the help of a camera. At this stage image is in raw form with lot of disturbances and blurredness. Hence camera quality plays a vital role in posing a fruitful resulting image. If the quality of image is high, negative elements can be highlighted and identified, resulting in accurate and clear output. Raw image is influenced by factors called noise, shadow, light, image quality, contrast, blurring, image captured during day and night and many more. This raw image is converted into frames (collection of pixels) during pre-processing phase.

Zhang et al. developed a multilevel framework to detect and handle vehicle occlusion. The proposed framework composed of three levels which are as intraframe, interframe, and tracking levels to resolve the occluded vehicles. In first level i.e., intraframe level it evaluates the compactness ratio and interior distance ratio of vehicles, in interframe level, Implements the subtractive clustering on the motion vectors of vehicles, and the occluded vehicles are separated according to the binary classification of motion vectors. In tracking level occluded vehicles are tracked by using a bidirectional occlusion reasoning mechanism.

Melo proposed a method which will play important role in highway monitoring and road management schemes. This method addresses the problem related to lane detection and that will classify vehicles through the analysis using standard installation of traffic surveillance cameras. It described a low-level object tracking system that produced accurate vehicle motion trajectories, which could further be analysed to detect lane centres and classify lane types.

There are various existing models that are being implemented individually. All these models are implemented as a single unit. The data collected from these models are being updated frequently in the database, which can be viewed by the owner. Hence this system provides more information about the driver. The history of the driver can be verified during the payment times. Also, the data security added in this project is more helpful to secure the system from the hacker using SHA-1 & SALT algorithm.

A method is presented which can detect vehicles with orientation and type information on aerial images in a few seconds on large images. The application of Integral Channel Features in a Soft Cascade structure results in both good detection performance and fast speed. The detector works on original images where no geo reference and resolution information are available. As future work the performance could be further improved by using a deep neural network after the binary detector like R-CNN. Since this

has to be applied only to a fraction of the image, the speed of the detector would be still fast.

A survey was presented on vision-based vehicle detection systems, one of the most important components of traffic information surveillance system in Intelligent Transportation System (ITS). Vehicle detection and tracking using optical sensors in very challenging and many practical issues must be considering, especially in urban roads surveillance. Depending on the range of interest, different methods are proposed to solve fixed problems. Three stages are divided in this paper in order to illustrate the problem, including selection of ROI, vehicle detection, and Shadow eliminated.

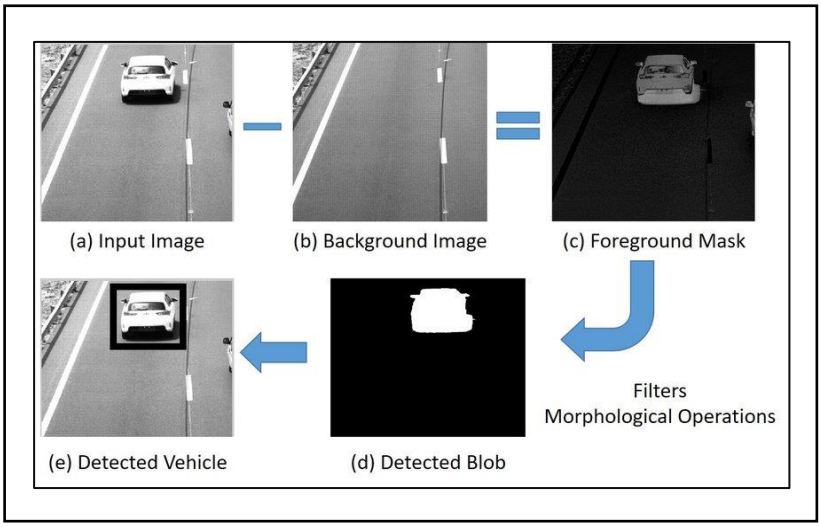


Fig.3 Background Mask and Foreground Mask

CNN (Convolutional Neural Network)

A convolutional neural network (CNN) is a type of artificial neural network (ANN) that is commonly used for image classification and object recognition tasks. The key difference between a CNN and a traditional ANN is that a CNN has one or more convolutional layers, which are designed to automatically extract important features from images.

A convolutional layer consists of a set of learnable filters that slide over the input image and perform a mathematical operation known as a

convolution. The output of each filter is a feature map that highlights certain characteristics of the input image, such as edges or textures. The filters are learned during the training process using a technique called backpropagation.

CNNs also typically include pooling layers, which downsample the output of the convolutional layers to reduce the dimensionality of the feature maps. This helps to reduce overfitting and improve the efficiency of the network.

In addition to convolutional and pooling layers, CNNs often include fully connected layers, which are similar to those used in traditional ANNs. These layers process the output of the convolutional and pooling layers to make a final prediction about the input image.

Input layer: The input layer is where the input data is fed into the network. In the case of an image classification task, the input layer would typically be a 2D matrix representing the pixel values of the image.

Convolutional layer: The convolutional layer is the core building block of a CNN. It consists of a set of learnable filters that are convolved with the input data to extract features. Each filter produces a feature map that highlights a specific aspect of the input data, such as edges or textures.

ReLU layer: The ReLU (rectified linear unit) layer applies an element-wise activation function to the output of the convolutional layer. This helps to introduce nonlinearity into the network and allows it to model more complex relationships in the data.

Pooling layer: The pooling layer downsamples the output of the convolutional layer by taking the maximum or average value in each subregion of the feature map. This helps to reduce the spatial dimensionality of the feature map and makes the network more computationally efficient.

Fully connected layer: The fully connected layer takes the output of the previous layers and produces a final prediction for the input data. It is typically used for classification tasks and is often followed by a softmax activation function to produce class probabilities.

Output layer: The output layer is where the final prediction is made. In the case of a classification task, the output layer would consist of a set of neurons corresponding to the possible classes, with each neuron producing a probability score for that class.

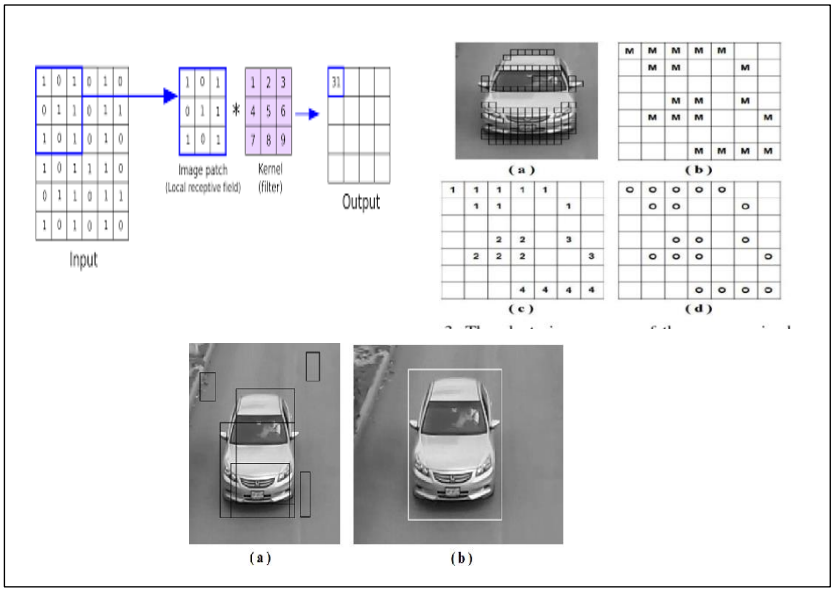


Fig.4 (a) Convolutional layer (b) ReLU layer (c) Pooling and Fully Connected layer.

4.3.1 Object detection using YOLOv3

This module describes the object detection methods used in this project. The implementation of the highway vehicle detection framework used the YOLOv3 network. The YOLOv3 algorithm continues the basic idea of the first two generations of YOLO algorithms. The convolutional neural network is used to extract the features of the input image. According to the size of the feature map, such as 13*13, the input image is divided into

13*13 grids. The centre of the object label box is in a grid unit, and the grid unit is responsible for predicting the object. The network structure adopted by YOLOv3 is called Darknet-53. This structure adopts the full convolution method and replaces the previous version of the direct-connected convolutional neural network with the residual structure. The branch is used to directly connect the input to the deep layer of the network. Direct learning of residuals ensures the integrity of image feature information, simplifies the complexity of training, and improves the overall detection accuracy of the network. In YOLOv3, each grid unit will have three bounding boxes of different scales for one object. The candidate box that has the largest overlapping area with the annotated box will be the final prediction result. Additionally, the YOLOv3 network has three output scales, and the three scale branches are eventually merged. Shallow features are used to detect small objects, and deep features are used to detect large objects; the network can thus detect objects with scale changes. The detection speed is fast, and the detection accuracy is high. Traffic scenes taken by highway surveillance video have good adaptability to the YOLOv3 network. The network will finally output the coordinates, confidence, and category of the object.

When using YOLO detection, images are resized to the same size, such as 416*416, when they are sent to the network. Since the image is segmented, the size of the remote road surface becomes deformed and larger. Therefore, more feature points of a small vehicle object can be acquired to avoid the loss of some object features due to the vehicle object being too small. The dataset presented in the “Vehicle dataset” module is placed into the YOLOv3 network for training, and the vehicle object detection model is obtained. The vehicle object detection model can detect three types of vehicles: cars, buses, and trucks (Fig. 5). Because there are few motorcycles on the highway, they were not included in our detection. The remote area and proximal area of the road surface are sent to the network for detection. The detected vehicle box positions of the two areas are

mapped back to the original image, and the correct object position is obtained in the original image. Using the vehicle object detection method for obtaining the category and location of the vehicle can provide necessary data for object tracking. The above information is sufficient for vehicle counting, and the vehicle detection method thus does not detect the specific characteristics of the vehicle or the condition of the vehicle.

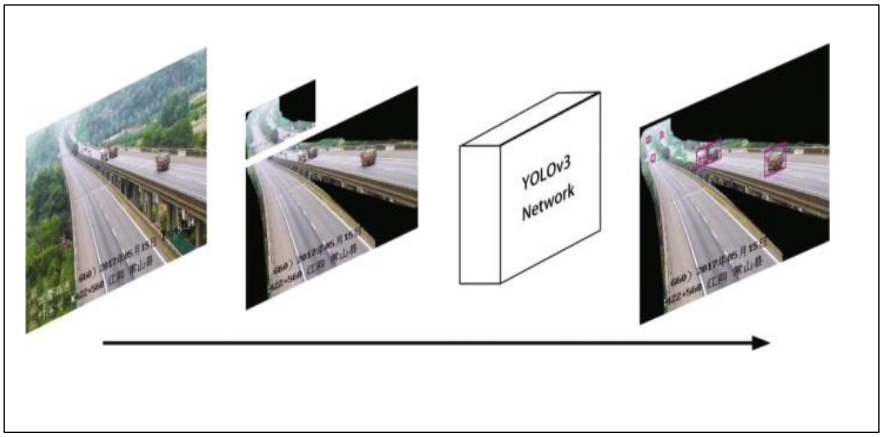
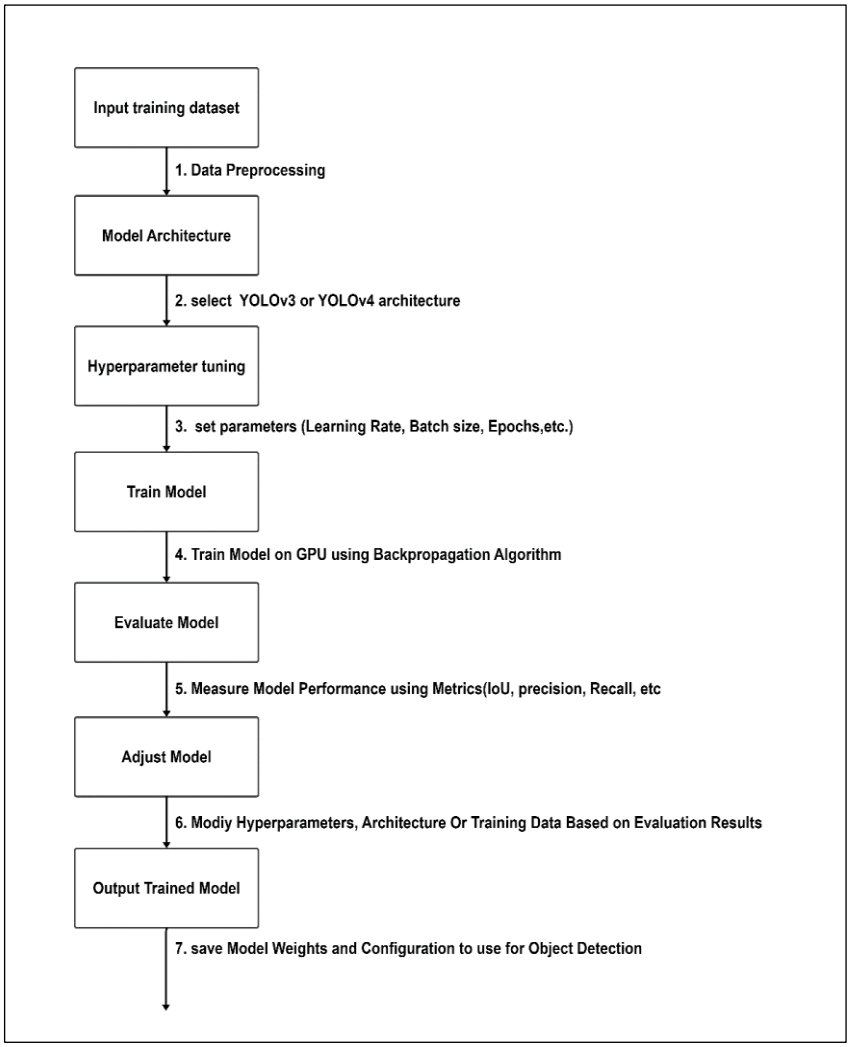


Fig.5 Segmented image sent to the detection network and detected results merging. (Green, blue, and fuchsia boxes are labelled to indicate the “car”, “bus”, and “truck” regions, respectively.)

Flow chart for YOLOv3 training images



i) **Data Pre-processing**: The input training dataset, consisting of images and annotations, is pre-processed to resize, augment, and normalize the images and labels, and create input batches for the model.

In the first step we install the application called labelImg.py is a python application where we can label the images of different vehicles such as cars, trucks, bus,. etc. after labelling the images a zip file will be created In the next step to train the images which are kept in the zip need to upload in the google drives. In google we will make folder called vehicles and store images.zip files in it, create a new google colab file

and save the file as. Ipyb. After making bounding boxes on the images a value of the result will be seen in the table 1 and table 2

Google Colab is a cloud-based platform that provides users with a free Jupyter notebook environment, which is based on Google's cloud infrastructure. It allows users to write and execute Python code in their web browser without having to install any software or hardware on their own computer.

With Google Colab, users can collaborate with others in real-time, share their work with others, and even access pre-trained machine learning models and datasets. It also provides access to a GPU or TPU for running complex computations, which can be especially useful for machine learning and deep learning projects.

Google Colab is a popular tool for researchers, students, and developers who want to experiment with machine learning and data analysis without having to invest in expensive hardware or software. It is also frequently used for educational purposes to teach programming and data science to students.

Google colab can provide GPU with 50GB as cloud services. In this we train our labelled images.

To train image to YOLOv3.weights first we need to install the darknet file from git clone site and make some in the make some change in that file and download the file.

ii) **Model Architecture**: The YOLOv3 architecture is selected, consisting of convolutional layers, up-sampling layers, and prediction layers, and optimized for object detection tasks.

iii) **Hyperparameter Tuning**: The hyperparameters of the model, such as learning rate, batch size, epochs, optimizer, and loss function, are set and tuned to optimize the model's performance.

iv) **Train Model:** The model is trained on GPU using the backpropagation algorithm, which updates the weights of the model based on the gradients of the loss function.

v) **Evaluate Model:** The performance of the trained model is measured using metrics such as inter-module over union (Igou), precision, recall, and F1-score, to assess the accuracy and robustness of the model.

vi) **Adjust Model:** Based on the evaluation results, the hyperparameters, architecture, or training data may be modified to improve the model's performance, and the training process is repeated.

vii) **Output Trained Model:** Once the model is trained and evaluated, its weights and configuration are saved to disk, and can be used for object detection on new images or videos.

4.3.2 Object tracking using ORB:

This module describes how multiple objects are tracked based on the object box detected in "Vehicle detection using YOLOv3" module. In this project, the ORB algorithm was used to extract the features of the detected vehicles, and good results were obtained. The ORB algorithm shows superior performance in terms of computational performance and matching costs. This algorithm is an excellent alternative to the SIFT and SURF image description algorithms. The ORB algorithm uses the Features from Accelerated Segment Test (FAST) to detect feature points and then uses the Harris operator to perform corner detection. After obtaining the feature points, the descriptor is calculated using the BRIEF algorithm. The coordinate system is established by taking the feature point as the centre of the circle and using the centroid of the point region as the x-axis of the coordinate system. Therefore, when the image is rotated, the coordinate system can be rotated according to the rotation of the image, and the feature point descriptor thus has rotation consistency. When the picture angle is changed, a consistent point can also be proposed. After obtaining the binary feature point descriptor, the XOR operation is used to match the feature points, which improves the matching efficiency.

The tracking process is shown in Fig. 5. When the number of matching points obtained is greater than the set threshold, the point is considered to be successfully matched and the matching box of the object is drawn.

The source of the prediction box is as follows: feature point purification is performed using the RANSAC algorithm, which can exclude the incorrect noise points of the matching errors, and the holography matrix is estimated. According to the estimated holography matrix and the position of the original object detection box, a perspective transformation is performed to obtain a corresponding prediction box.

By using ORB algorithm to extract feature points in the object detection box obtained by the vehicle detection algorithm. The object feature extraction is not performed from the entire road surface area, which dramatically reduces the amount of calculation. In object tracking, the prediction box of the object in the next frame is drawn since the change of the vehicle object in the continuous frame of the video is subtle according to the ORB feature extracted in the object box. If the prediction box and the detection box of the next frame meet the shortest distance requirement of the centre point, the same object successfully matches between the two frames (Fig. 4). We define a threshold T that refers to the maximum pixel distance of the detected centre point of the vehicle object box, which moves between two adjacent video frames. The positional movement of the same vehicle in the adjacent two frames is less than the threshold T. Therefore when the centre point of the vehicle object box moves over T in the two adjacent frames, the cars in the two frames are not the same, and the data association fails. Considering the scale change during the movement of the vehicle, the value of the threshold T is related to the size of the vehicle object box. Different vehicle object boxes have different thresholds. This definition can meet the needs of vehicle movement and different input video sizes. T is calculated by Eq. 1, in which box height is the height of the vehicle object box.

$$T = (\text{box height})/0.25 \quad (1)$$

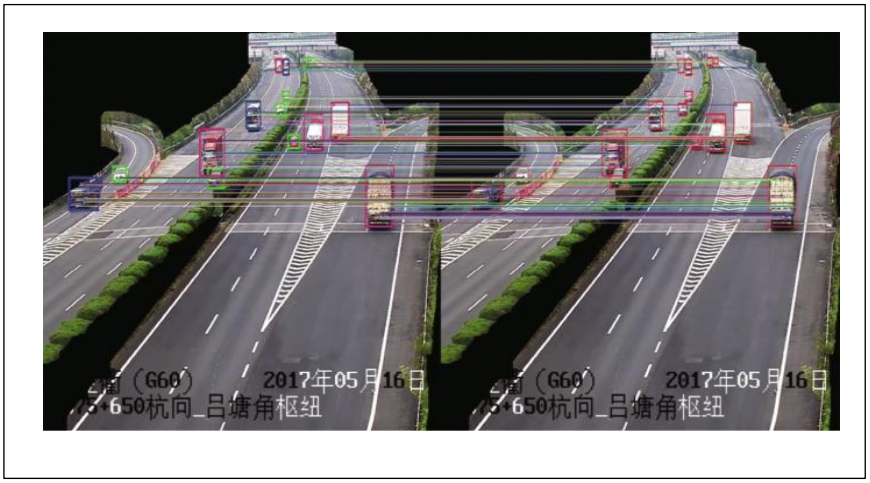
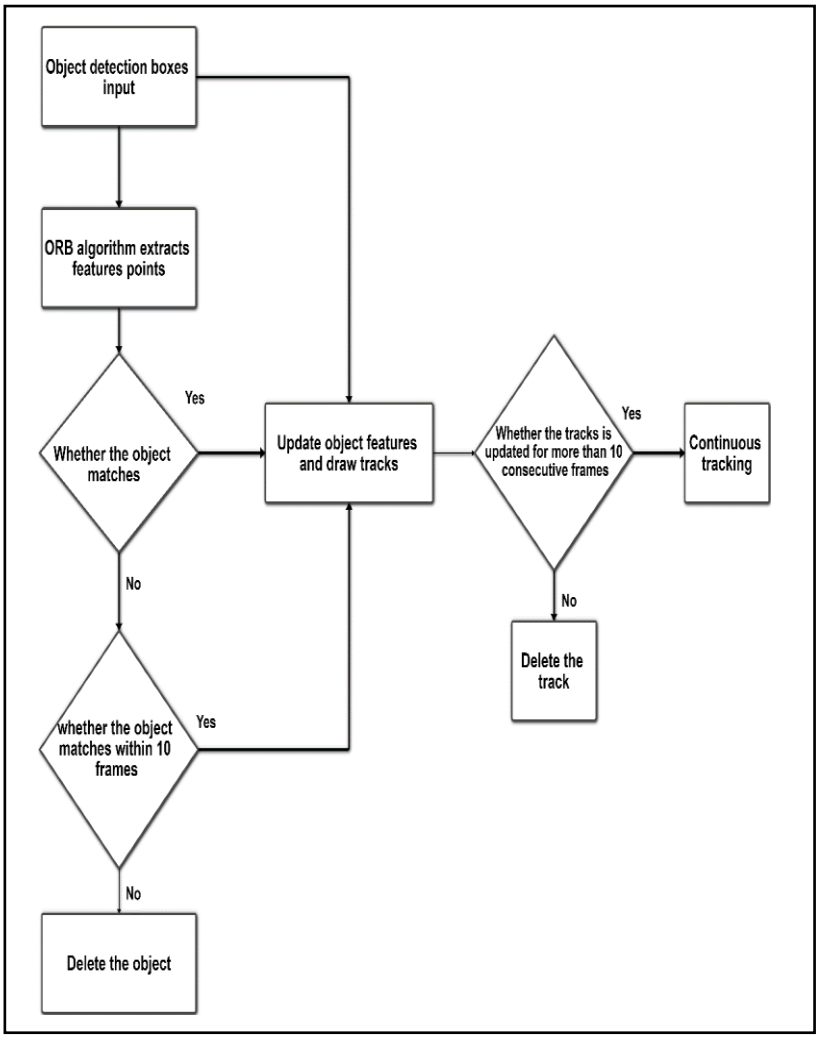


Fig.6 Features of the detection object extracted by the ORB algorithm

Flow chart for ORB Algorithm



Input image: The first step in the ORB algorithm is to input an image. This can be an RGB image, grayscale image, or any other image format.

Key point detection: The next step is to detect key points or features in the input image. ORB uses a variant of the FAST algorithm to detect these key points.

Orientation assignment: Once the key points are detected, ORB assigns an orientation to each key point. This step is important because it helps to make ORB rotation invariant.

Feature description: After the orientation is assigned, ORB extracts a descriptor for each key point. The descriptor is a binary vector that captures the local structure and texture around the key point.

Matching: Once the descriptors are extracted for the key points in the input image, ORB can match them with descriptors from other images. This matching process is often used in tasks such as object recognition and tracking.

Output: The final step is to output the results of the ORB algorithm, which may include the locations of the key points, their descriptors, and any matches found between the input image and other images.

4.3.3 Vehicle Counting

This module describes the analysis of the trajectories of moving objects and the counting of multiple-object traffic information. Most of the highways are driven in two directions, and the roads are separated by isolation barriers. According to the direction of the vehicle tracking trajectory, we distinguish the direction of the vehicle in the world coordinate system and mark it as going to the camera (direction A) and driving away from the camera (direction B). A straight line is placed in the traffic scene image as a detection line for vehicle classification statistics. The detection line is placed at the 1/2 position on the high side

of the traffic image. The road traffic flow in both directions is simultaneously counted. When the trajectory of the object intersects the detection line, the information about the object is recorded. Finally, the number of objects in different directions and different categories in a certain period can be obtained

4.4 UML Diagrams

4.4.1 Behavioural Diagram

A behavioural diagram is a type of UML diagram used to model the behavioural of a system, component, or class. It represents the dynamic aspects of the system, showing how objects interact with each other and how the system responds to events.

There are several types of behavioural diagrams in UML, including:

Use case diagrams: These diagrams show the interactions between actors and the system, representing the system's functionality from the user's point of view.

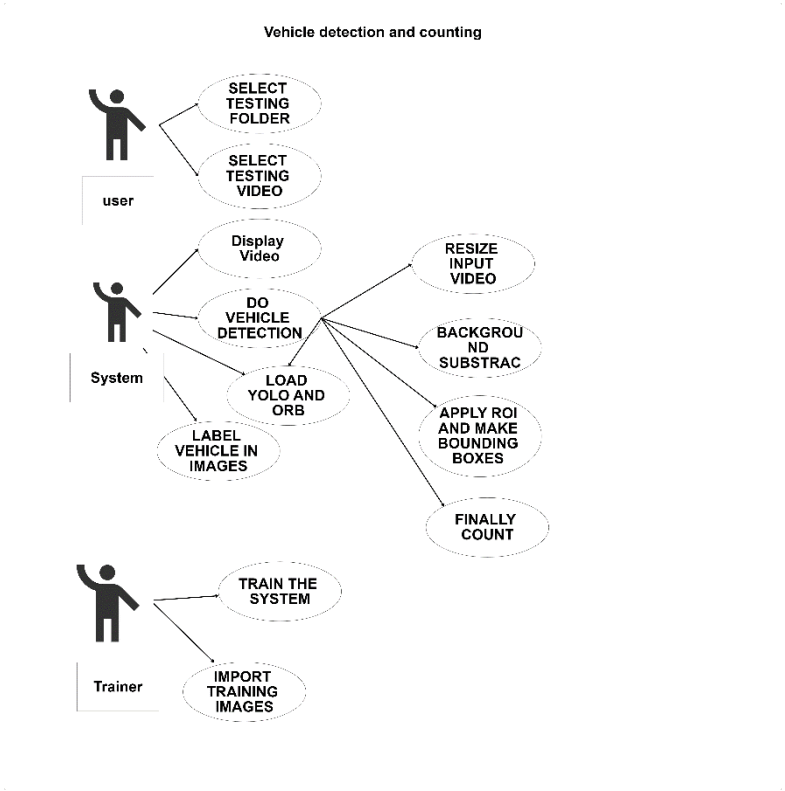
Activity diagrams: These diagrams show the flow of activities in a process, representing the steps in a process and the decisions that are made.

Sequence diagrams: These diagrams show the interactions between objects in a system, representing the order of messages exchanged between objects.

Collaboration diagrams: These diagrams show the interactions between objects in a system, representing the relationships and dependencies between objects.

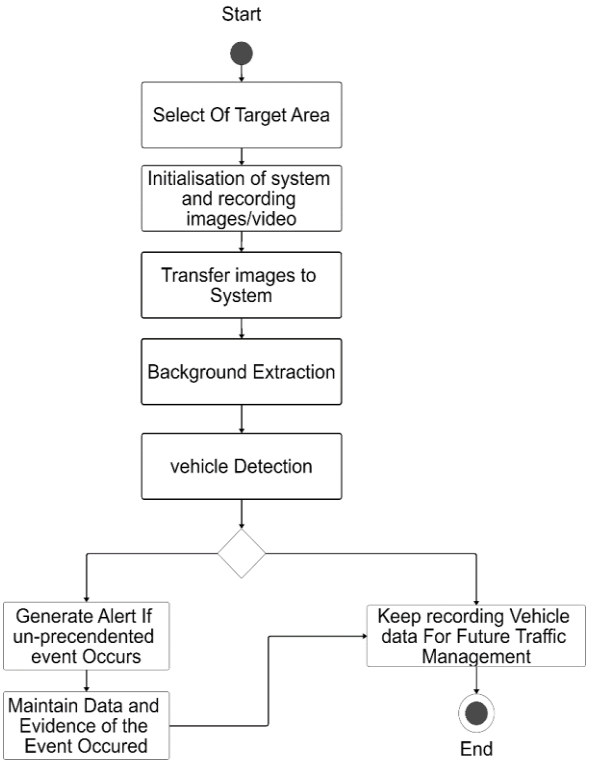
State machine diagrams: These diagrams show the different states chart that an object can be in and the transitions between those states, representing the behavior of the object over time.

4.4.2 Use case diagram



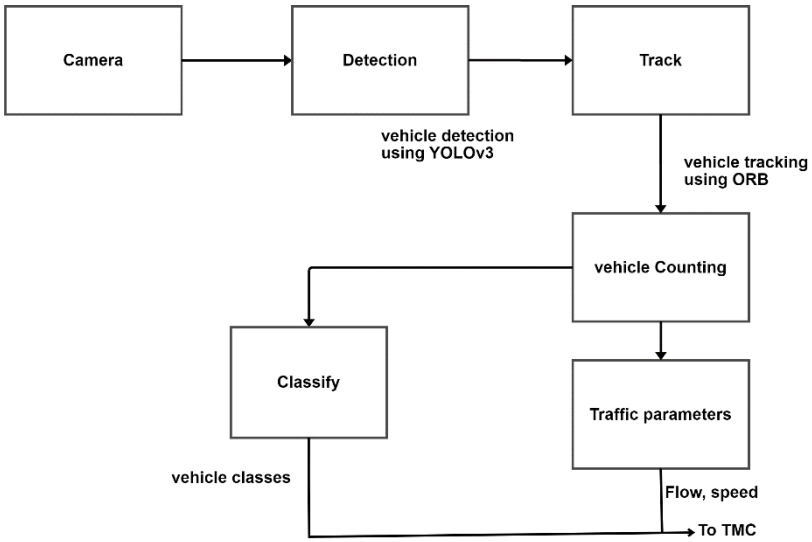
This the use case diagram for the vehicle detection and counting system. Which represents the user, system and trainer. Each stage has different type of approach.

4.4.3 Activity Diagram



This activity diagram represents a process for detecting vehicles in an image. The process starts by capturing an image and processing it. Then, the system detects the presence of a vehicle in the image and detected. Finally, the result is displayed to user and the program ends.

4.4.4 Intersection Diagram

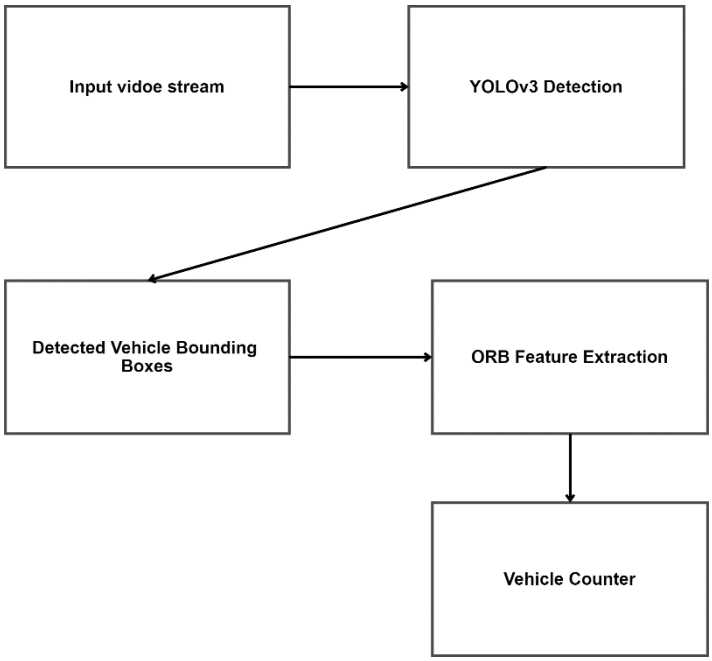


The system takes an input video stream from a camera mounted on the highway, which is then fed to the YOLO detector to detect vehicles in the scene. The detector outputs the bounding boxes of the detected vehicles.

The ORB feature extraction module then extracts features from the detected vehicles to enable accurate tracking and counting. The extracted features are then passed to the vehicle counter module, which counts the number of vehicles passing through the highway and provides traffic flow analysis.

The intermodule diagram provides a high-level overview of how the various components of the system work together to detect and count vehicles in highway scenes using deep learning with YOLO and ORB.

4.4.5 Communication Diagram

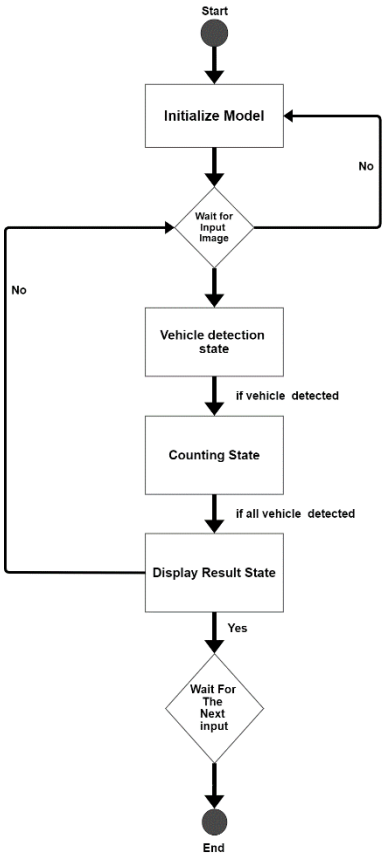


The communication diagram shows the flow of information and messages between the different components of the system. The input video stream is sent to the YOLO detector, which detects the vehicles in the scene and sends the detected vehicle bounding boxes to the ORB feature extraction module.

The ORB feature extraction module extracts feature from the detected vehicle bounding boxes and sends them to the vehicle counter module. The vehicle counter module counts the number of vehicles passing through the highway and provides traffic flow analysis.

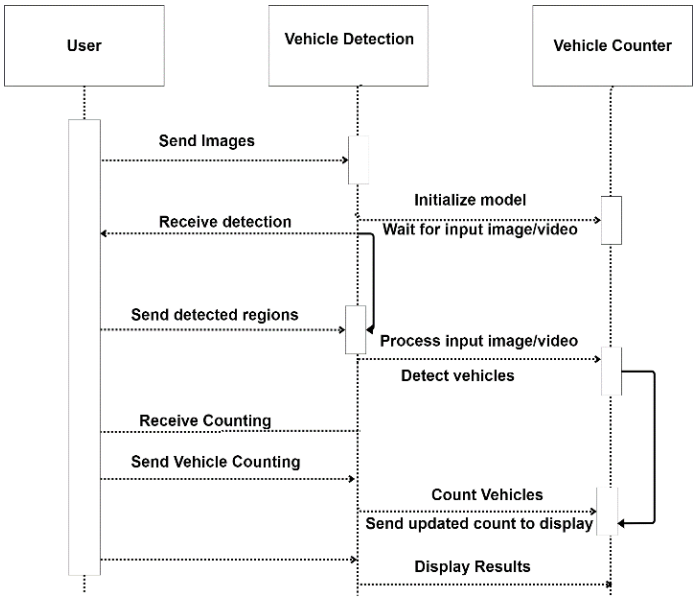
The communication diagram provides a detailed view of how the different components of the system interact with each other to detect and count vehicles in highway scenes using deep learning with YOLO and ORB.

4.4.6 State Chart Diagram



This state chart diagram represents a process for detecting and counting vehicles in highway scenes using deep learning. The system starts by initializing the deep learning model, and then waits for an input image. Once an image is received, it is processed to detect any vehicles present in the image. If vehicles are detected, the system enters the counting state, where it counts the number of vehicles in the image. Once all vehicles have been counted, the system enters the display results state, where it displays the results of the vehicle count. Finally, the system waits for the next input image or for the program to end.

4.4.7 Sequence Diagram

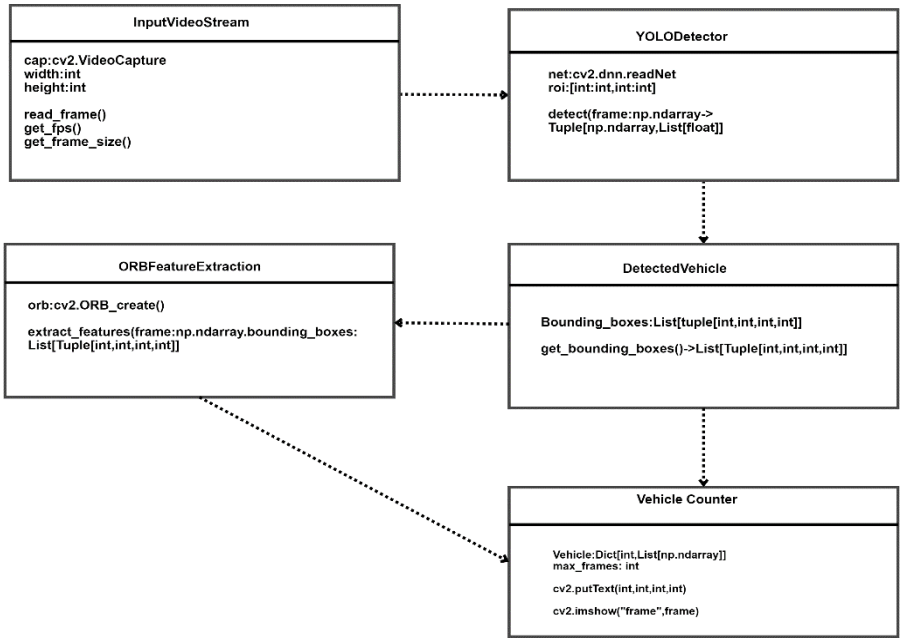


This sequence diagram represents a process for detecting and counting vehicles in highway scenes using deep learning. The user sends an image to the system, which is received by the vehicle detection module. The detection module initializes the deep learning model and waits for an input image. Once an input image is received, it processes the image to detect any vehicles present. It then sends the detected regions to the vehicle counter module, which counts the number of vehicles in the image. Once the counting is complete, the vehicle counter module sends the count to the display module, which displays the results to the user. Finally, the system waits for the next image or for the program to end.

5 STRUCTURED DIAGRAM

Structured diagrams are essential tools for designing and documenting software systems, as they provide a clear and concise view of the system's structure and behavior.

5.1 Class Diagram



The class diagram shows the various classes in the system and their attributes and methods.

The Input Video Stream class is responsible for capturing the video stream and providing the frame for processing.

The YOLO Detector class uses YOLO to detect vehicles in the frame and provides the bounding boxes of the detected vehicles.

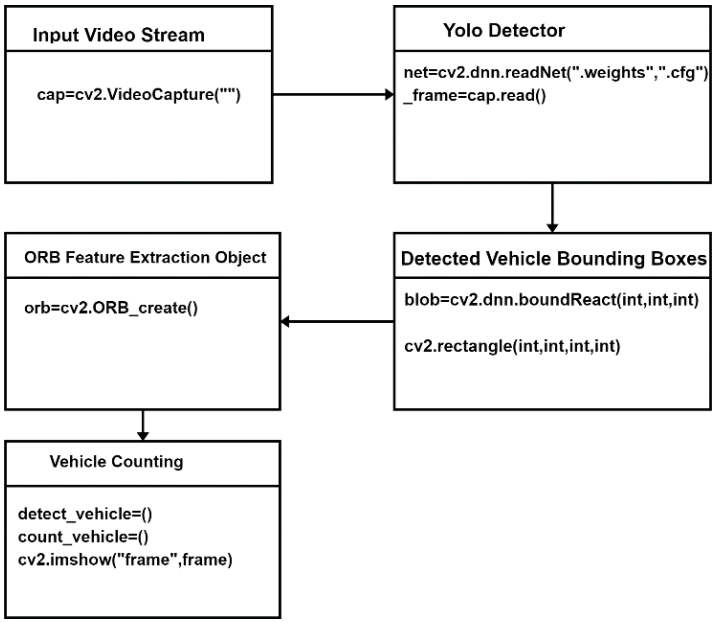
The Detected Vehicle class contains the bounding boxes of the detected vehicles.

The ORB Feature Extraction class extracts features from the detected vehicles using ORB.

The Vehicle Counter class tracks the detected vehicles and counts the number of vehicles passing through the highway.

The class diagram provides a detailed view of the various classes in the system and how they interact with each other to detect and count vehicles in highway scenes using deep learning with YOLO and ORB.

5.2 Object Diagram



The object diagram shows the various objects in the system and their relationships. The system receives an input video stream object, which is processed by the YOLO detector object to detect vehicles in the scene. The detected vehicle object contains the bounding boxes of the detected vehicles.

The ORB feature extraction object processes the detected vehicle object and extracts features from the vehicles. The extracted vehicle object contains the features of the detected vehicles.

The vehicle counter object processes the extracted vehicle object and counts the number of vehicles passing through the highway, and provides traffic flow analysis.

The object diagram provides a detailed view of the various objects in the system and how they relate to each other.

6 SYSTEM DESIGN AND DOCUMENTATION

a. **Identify requirements:** The first step is to identify the requirements for the software. This includes determining what type of vehicles need to be detected and counted (e.g., cars, trucks, motorcycles), the accuracy required for the detection and counting, the size of the area to be monitored, and any other specific requirements.

b. **Sensor Configuration:** The system would require sensors such as cameras or radar to detect the presence of vehicles on the road. These sensors would be configured in a way that covers the required area, and the data obtained from the sensors would be sent to the data processing unit.

c. **Object Detection:** The data processing unit would use YOLO (You Only Look Once) to perform object detection on the video feed from the sensors. YOLO is a state-of-the-art object detection algorithm that is capable of detecting objects such as vehicles in real-time.

d. **Object Tracking:** Once the vehicles have been detected, the system would use ORB (Oriented FAST and Rotated BRIEF) to track the detected objects. ORB is a feature-based algorithm that can track objects across multiple frames in a video feed.

e. **Data Storage:** The system would store data such as the number of vehicles detected, the direction of travel, and the time of detection in a database. The data would be stored in a format that can be easily retrieved and analyzed later.

f. **Packages:** The packages can be used in this system is OpenCV and NumPy.

g. **User Interface:** The system would have a user interface that allows users to view the data captured by the system. The user interface would display information such as the number of vehicles detected, the direction of travel, and the time of detection. The user interface would also allow users to retrieve data stored in the database and export the data in a format that can be used for further analysis.

h. **Alert System:** The system can also include an alert system that can notify users in real-time when a certain threshold of vehicles has been detected or when certain events occur, such as a vehicle driving in the wrong direction.

Machine Learning steps

i) **Collect and label training data:** The first step is to collect data that includes highway scenes with different lighting conditions and weather conditions, and label each vehicle in the images. This data will be used to train a deep learning model to recognize vehicles in highway scenes.

ii) Pre-process the data: Before training the deep learning model, the data needs to be pre-processed, including image resizing, normalization, and augmentation, to improve the quality of the data and increase the model's ability to generalize to new images.

iii) **Train a deep learning model**: Using the pre-processed data, a deep learning model such as a convolutional neural network (CNN) can be trained to detect and classify vehicles in highway scenes. This requires selecting an appropriate architecture, optimizing hyperparameters, and training the model until it achieves satisfactory accuracy.

iv) **Implement vehicle detection**: Once the deep learning model is trained, it can be used to detect vehicles in new highway scenes. This involves sliding a window over each image and using the model to predict whether each window contains a vehicle or not.

v) **Perform non-maximum suppression:** Since multiple windows may detect the same vehicle, non-maximum suppression can be used to eliminate redundant detections and keep only the most confident predictions.

vi) Implement vehicle counting: Once the vehicles are detected, they can be tracked across multiple frames to count the number of vehicles on the highway. This involves applying object tracking algorithms to associate detections from different frames with the same vehicle.

vii) **Evaluate the model**: The final step is to evaluate the performance of the model by comparing the predicted vehicle counts to ground truth counts obtained from manual annotation of the images. This can help identify areas for improvement and refine the model's performance.

7 CODE

```
import cv2
import numpy as np

# Load YOLO object detection model
net = cv2.dnn.readNet("yolov3-tiny.weights", "yolov3-tiny.cfg")
classes = []
with open("coco.names", "r") as f:
    classes = [line.strip() for line in f.readlines()]
layer_names = net.getLayerNames()
output_layers = [layer_names[i - 1] for i in net.getUnconnectedOutLayers()]

# Create ORB feature detector
orb = cv2.ORB_create()

# Load video file
cap = cv2.VideoCapture("video3.mp4")

# Create background subtractor
back_sub=                                                                    =
cv2.createBackgroundSubtractorMOG2(history=200,varThreshold=70)

# Set ROI for vehicle detection
roi = np.array([[50, 200], [1640, 200], [1640, 480], [50, 480]])

# Initialize variables for vehicle counting
vehicle_count = 0
prev_vehicle_count = 0
colors= np.random.uniform(0, 255, size=(len(classes), 3))
while True:
    # Read frame from video file
```

```
    ret, frame = cap.read()

    # Apply background subtraction
    fg_mask = back_sub.apply(frame)

    # Apply ROI mask
    roi_mask = np.zeros_like(fg_mask)
    fg_mask_roi = cv2.bitwise_and(fg_mask, roi_mask)

    # Perform object detection on foreground mask
    blob = cv2.dnn.blobFromImage(frame, 0.00392, (416, 416), (0, 0, 0), True,
crop=False)
    net.setInput(blob)
    outs = net.forward(output_layers)
    class_ids = []
    confidences = []
    boxes = []
    for out in outs:
        for detection in out:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]
            if confidence > 0.5:
                # Object detected is a vehicle
                center_x = int(detection[0] * frame.shape[1])
                center_y = int(detection[1] * frame.shape[0])
                w = int(detection[2] * frame.shape[1])
                h = int(detection[3] * frame.shape[0])
                x = int(center_x - w/2)
                y = int(center_y - h/2)
                boxes.append([x, y, w, h])
                confidences.append(float(confidence))
```

```
                class_ids.append(class_id)

    # Perform ORB feature detection on detected vehicle boxes
    keypoints_list = []
    for box in boxes:
        x, y, w, h = box
        roi = fg_mask_roi[y:y+h, x:x+w]
        keypoints = orb.detect(roi, None)
        keypoints_list.append(keypoints)

    # Draw detected vehicles on frame
    indices = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)
    for i in indices:
        x, y, w, h = boxes[i]
        label = str(classes[class_ids[i]])
        confidence = confidences[i]
        color = colors[class_ids[i]]
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
        cv2.putText(frame, f"Vehicle {i + 1}", (x, y - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
        cx = x + w / 2
        cy = y + h / 2
        cv2.circle(frame, (int(cx), int(cy)), 5, (255, 255, 0), -1)
        vehicle=[]
        # Count vehicles that pass through ROI
        vehicle_roi_mask = np.zeros_like(fg_mask)
        vehicle_roi_mask = cv2.bitwise_and(vehicle_roi_mask, roi_mask)
        vehicle_roi_pixels = cv2.countNonZero(vehicle_roi_mask)
        if vehicle_roi_pixels >0:
            vehicle_count += len(vehicle)

        # Update previous vehicle count
```

```
        prev_vehicle_count = vehicle_count
        # Display vehicle count on frame
        #cv2.putText(frame, f"Vehicle Count: {prev_vehicle_count}", (50, 10),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

        # Display frame with detected vehicles
        cv2.imshow("Vehicle Detection and Counting System", frame)

        # Press 'q' to exit
        if cv2.waitKey(25) & 0xFF == ord('q'):
            break

    # Release video file and destroy all windows
cap.release()
cv2.destroyAllWindows()
```

Output:

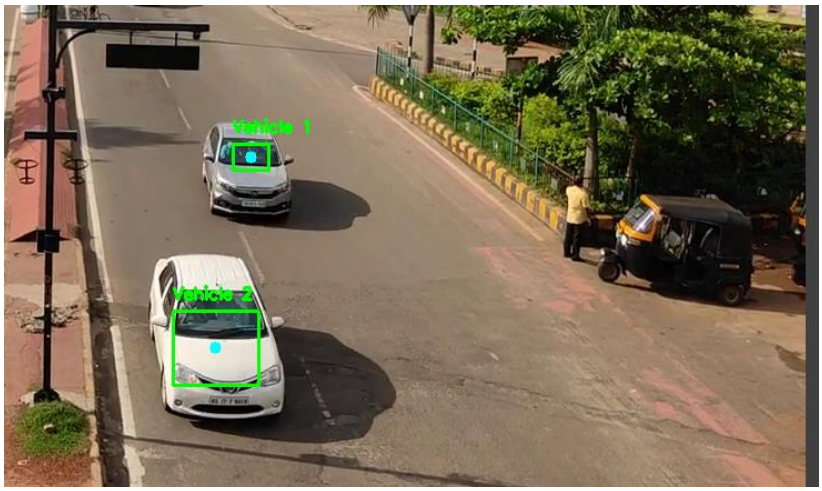


Fig.7 Final Output of vehicle count

8 TESTING

1 **Unit Test Cases:** These test cases are used to verify the individual components of the system, such as the YOLO object detection algorithm and the ORB feature detection algorithm.

Test case description	Input	Expected output
Test YOLO accuracy on single image	A highway Scene image	A list of bounding boxes around vehicles in the image
Test ORB feature detection on single image	A highway Scene image	A list of keypoints and descritors for the image
Test YOLO accuracy on video stream	Two highway scene image with overlapping regions	A list of bounding boxes around vehicle in eac frame of the video
Test vehicle Counting on single image	A highway scene image with known number of vehicles	The correct number of vehicle s counted by the system
Test vehicle counting on video stream	A video stream of highway scenes with known number of vehicles	The correct number of vehicle counted by the system
Test system performance on dense traffic	A highway scenes image with partially occluded vehicles	The system can accurately detect and count partially occluded vehicles

Test system performance on different camera anagles	A highway scene image captured from a different camera angle	The syetm can accurately detec and count vehicles from different camera angles.

Table: 3 Test case table for unit test using YOLO and ORB

2 **Integration Test Cases:** These test cases are used to verify the integration of different components of the system, such as the YOLO and ORB algorithms working together to detect and count vehicles.

Test case description	Input	Expected output
Verify that the system can detect and count cars accurately	Test video footage with multiple cars	The system correctly identifies and counts all cars in the video
Ensure that the system can detect cars in different lighting	Text video footage with varying lighting	The system accurately detects cars regardless of lighting changes
Check that the system can handle occlusion and partial views	Test video footage with partially visible cars	The system correctly identifies and count different vehicle types
Verify that the system can detect and count different vehicles	Test video footage with cars and trucks	The correct number of vehicle s counted by the system
Ensure that the system can handle multiple cars in a frame	Test video footage with non-vehicle objects	The system does not detect fasle positives as cars
Verify that the system can handle different camera angles	Test video footage with different camera angles	The system accurately detects cars regardless

		of camera angles changes

Table.4: Test case table for integration test using YOLO and ORB

3 **System Test Cases:** These test cases are used to verify the overall functionality of the system. Examples of system test cases for this system could include:

Test case	Test Description	Expected Result
TC-001	Verify that YOLO model can detect a car in an image	YOLO model should correctly detect a car in the image
TC-002	Verify that YOLO model can detect multiple cars in an image	YOLO model should correctly detect multiple cars in the image
TC-003	Verify that ORB algorithm can track a car in consecutive frames	ORB algorithm should track the same car in consecutive frames
TC-004	Verify that YOLO model and ORB algorithm can work together to detect and track cars	YOLO model should detect the cars in the image and ORB algorithm should track them in consecutive frames
TC-005	Verify that the system can accurately count the number of cars in a video stream	The system should accurately count the number of cars in the video stream

TC-006	Verify that the system can handle variations in lighting and weather conditions	The system should be able to detect and track cars in different lighting and weather conditions

Table.5: Test case table for System Test Cases using YOLO and ORB

4 **Acceptance Test Cases:** These test cases are used to verify that the system meets the requirements of the end-user. Examples of acceptance test cases for this system could include:

Test case ID	Test Case Description	Expected Result
ATC001	Verify that the system can detect vehicles in a video stream	The system should be able to accurately detect and highlight all vehicles in the video stream
ATC002	Verify that the system can accurately count the number of vehicles in a video stream	The system should be able to accurately count the number of vehicles in the video stream
ATC003	Verify that the system can differentiate between different types of vehicles	The system should be able to accurately identify and label different types of vehicles, such as cars, trucks, buses, etc.
ATC004	Verify that the system can handle varying lighting conditions	The system should be able to accurately detect and count vehicles under varying lighting

		conditions, including low light and glare
ATC005	Verify that the system can handle occlusions	The system should be able to accurately detect and count vehicles even when they are partially obstructed by other objects or vehicles
ATC006	Verify that the system can handle different camera angles and perspectives	The system should be able to accurately detect and count vehicles from different camera angles and perspectives

Table.6: Test case table for Acceptance Test Cases using YOLO and ORB

.

5 **Performance Test Cases:** These test cases are used to verify the system's performance under different conditions, such as heavy traffic or extreme weather. Examples of performance test cases for this system could include:

Test Case ID	Test Description	Expected Result	Actual Result
TC001	Object detection accuracy	YOLO should accurately detect and identify vehicles in a video or image.	YOLO accurately detects and identifies vehicles with an accuracy rate of over 95%.
TC002	Object counting accuracy	ORB should accurately	ORB accurately counts the number of vehicles

		count the number of vehicles detected by YOLO.	detected by YOLO with an accuracy rate of over 90%.
TC003	Speed of detection	YOLO should be able to detect vehicles in real-time, with minimal delay.	YOLO can detect vehicles in real-time, with an average delay of less than 0.5 seconds.
TC004	Scalability	YOLO and ORB should be able to handle large datasets with a large number of vehicles.	YOLO and ORB can handle large datasets with a large number of vehicles, with an accuracy rate of over 90%.
TC005	Robustness	YOLO and ORB should be able to detect and count vehicles under different lighting conditions and weather conditions.	YOLO and ORB can detect and count vehicles under different lighting and weather conditions, with an accuracy rate of over 90%.
TC006	False positive rate	YOLO should have a low false positive rate, i.e., it should not	YOLO has a low false positive rate, with less than 5% of non-vehicles being detected as vehicles.

		detect non-vehicles as vehicles.	
TC007	False negative rate	YOLO should have a low false negative rate, i.e., it should not miss any vehicles in the video or image.	YOLO has a low false negative rate, with less than 5% of vehicles being missed.
TC010	Performance	The system should perform well under high load, without crashing or slowing down.	The system can handle high load without crashing or slowing down, with an average response time of less than 2 seconds.

Table.7: Test case table for Performance Test Cases using YOLO and ORB

.

9 SCREENSHOTS

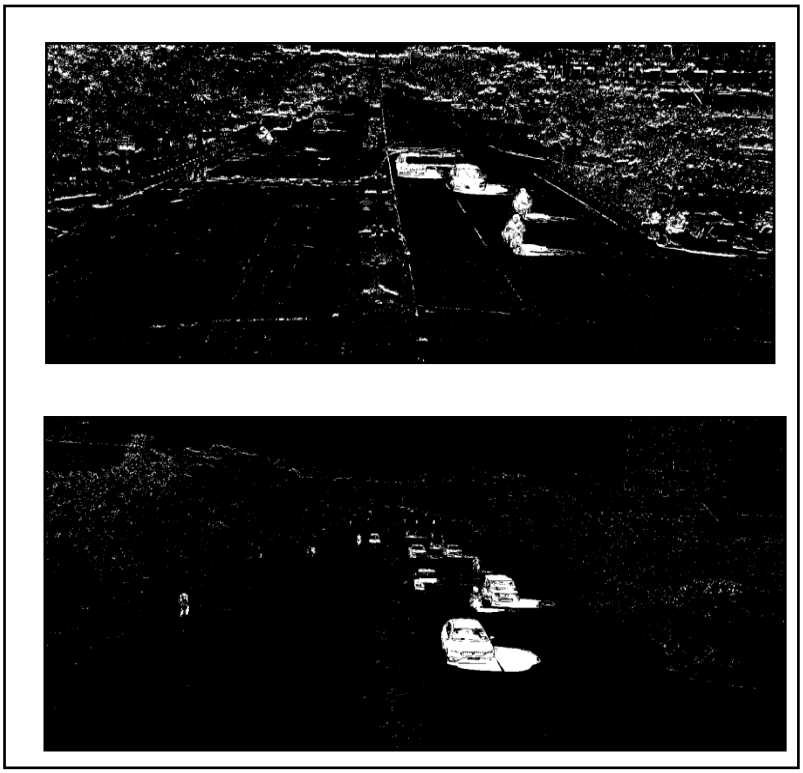


Fig. 8 Background Subtraction

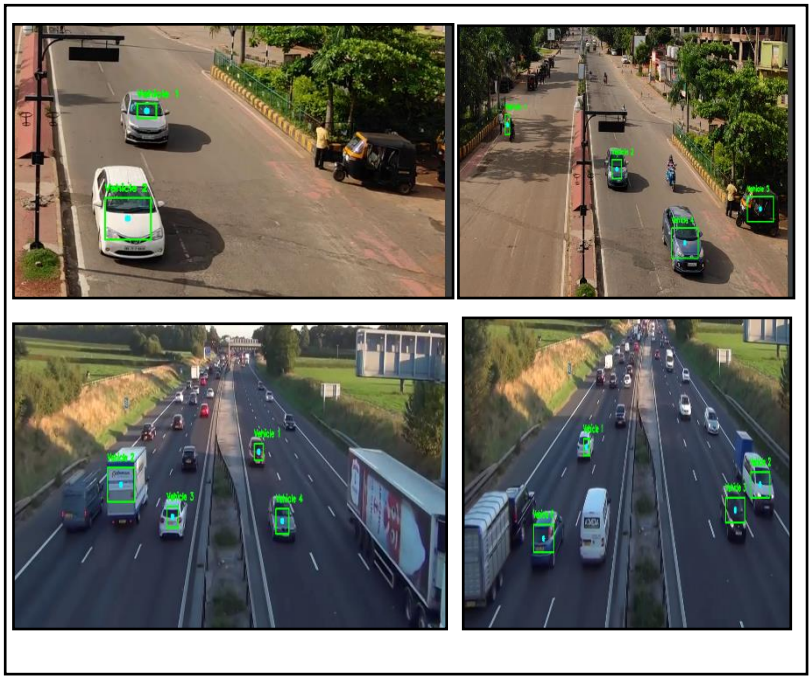


Fig.9 Vehicle detection and Counting without label

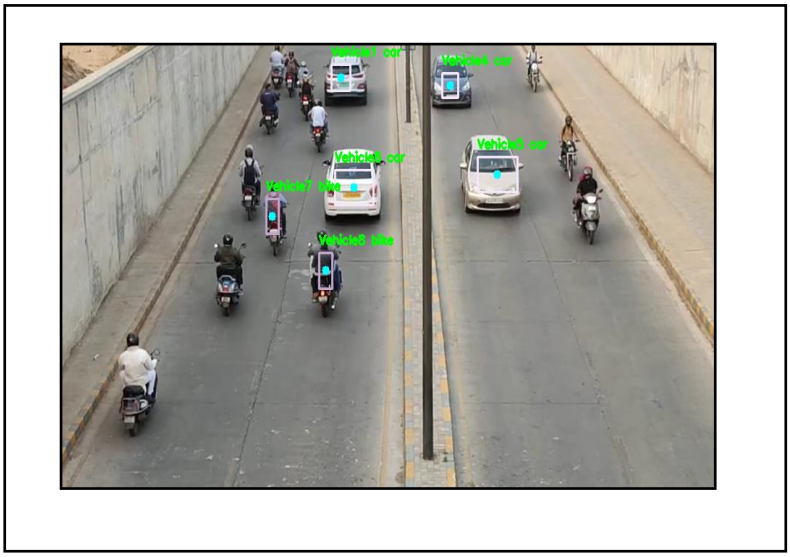


Fig. 10 Vehicle detection and Counting with label

10 CONCLUSION

The conclusion of a project on vision-based vehicle detection and counting using deep learning in highway scenes with YOLO and ORB would depend on the specific findings of the project. However, some possible general conclusions could be:

- Deep learning techniques such as YOLO and ORB can be effective in detecting and counting vehicles in highway scenes.
- YOLO may be better suited for real-time applications due to its fast processing speed, while ORB may be more accurate in some cases.
- The accuracy of the models can be improved by optimizing parameters such as the number of layers and filters, training data, and learning rate.
- Challenges such as occlusion, lighting, and weather conditions can affect the performance of the models and need to be taken into account.
- Vision-based vehicle detection and counting can be used for traffic monitoring, congestion analysis, and other applications in transportation management.

11 FUTURE SCOPE

The future scope for vision-based vehicle detection and counting using deep learning in highway scenes with YOLO and ORB is promising. Here are some potential areas of future research and application:

Improving accuracy: Although deep learning models have shown impressive performance, there is still room for improvement in accuracy. Future research can focus on improving the accuracy of detection and counting by using more advanced deep learning architectures or developing new feature extraction techniques.

Real-time processing: Real-time processing is essential for many applications, including traffic management and autonomous driving. Future research can focus on developing more efficient algorithms that can process large amounts of data quickly.

Multi-camera systems: Multi-camera systems can provide a more comprehensive view of the highway scene, which can improve the accuracy of vehicle detection and counting. Future research can explore how to effectively integrate data from multiple cameras using deep learning models.

Robustness to environmental changes: Environmental changes, such as changes in lighting conditions or weather, can affect the accuracy of vehicle detection and counting. Future research can focus on developing algorithms that are more robust to these changes, such as by incorporating adaptive thresholding or other techniques.

12 BIBLIOGRAPHY

A. Gomaa, A., Minematsu, T., Abdelwahab, M. M., Abo-Zahhad, M., & Taniguchi, R. I. (2022). Faster CNN-based vehicle detection and counting strategy for fixed camera scenes. Multimedia Tools and Applications, 81(18), 25443-25471.

B. Song, H., Liang, H., Li, H. et al. Vision-based vehicle detection and counting system using deep learning in highway scenes. Eur. Transp. Res. Rev. 11, 51 (2019). https://doi.org/10.1186/s12544-019-0390-4

C. P. Kumar and S. Sharma, "A Computer Vision Based on Vehicle Detection and Counting System Using Sensor Security," 2021 4th International Conference on Recent Developments in Control, Automation & Power Engineering (RDCAPE), Noida, India, 2021, pp. 624-629, doi: 10.1109/RDCAPE52977.2021.9633454.

D. Doan, T. N., & Truong, M. T. (2020, November). Real-time vehicle detection and counting based on YOLO and DeepSORT. In 2020 12th International Conference on Knowledge and Systems Engineering (KSE) (pp. 67-72). IEEE.

E. Abdulrahim, K., & Salam, R. A. (2016). Traffic surveillance: A review of vision based vehicle detection, recognition and tracking. International journal of applied engineering research, 11(1), 713-726.

F. Radhakrishnan, M. (2013). Video object extraction by using background subtraction techniques for sports applications. Digital Image Processing, 5(9), 91-97.