

A Project Report On

**SMART TRAFFIC LIGHT CONTROL SYSTEM BY  
VEHICAL DETECTION USING  
MACHINE LEARNING**

Submitted in partial fulfillment of the requirement for the award of the degree in

**BACHELOR OF TECHNOLOGY**

In

**COMPUTER SCIENCE AND ENGINEERING**

Submitted By

<b>P.VISHNU VARDHAN REDDY</b>	<b>198X1A05I4</b>
<b>E.VENKATESH</b>	<b>198X1A05H4</b>
<b>SK.MAHAMMAD RASOOL</b>	<b>198X1A05H3</b>
<b>Y.KANUKA LOURDHU RESHMA</b>	<b>198X1A05G6</b>

**Under the esteemed Guidance of  
Dr K Venkata Subba Reddy**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**KALLAM HARANADHAREDDY INSTITUTE OF TECHNOLOGY**

Approved by AICTE- New Delhi, Accredited by NAAC A Grade and NBA Accredited

Permanently Affiliated to Jawaharlal Nehru Technological University, Kakinada

NH-5, Chowdavaram, Guntur, Andhra Pradesh, India

2022-2023

**KALLAM HARANADHAREDDY INSTITUTE OF TECHNOLOGY**  
**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**CERTIFICATE**

This is to certify that the project report entitled SMART TRAFFIC CONTROL SYSTEM BY VEHICAL DETECTION USING MACHINE LEARNING being submitted by

**P.VISHNU VARDHAN REDDY**

**198X1A05I4**

**E.VENKATESH**

**198X1A05H4**

**SK.MAHAMMAD RASOOL**

**198X1A05H3**

**Y.KANUKA LOURDHU RESHMA**

**198X1A05G6**

in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science and Engineering to the Jawaharlal Nehru Technological University, Kakinada is a record of bonafied work carried out under my guidance and supervision.

The result embedded in this thesis has not been submitted to any other university /institute for the award of any degree / diploma.

**PROJECT SUPERVISOR**

Dr K Venkata Subba Reddy  
B.Tech,M.Tech,Ph.D  
Professor, Department of CSE, Khit

**HEAD OF THE DEPARTMENT**

Prof V Rajeev Jetson  
M.Tech,(Ph.D)  
Head of the Department

**EXTERNAL EXAMINER**

## DECLARATION

We hereby declare that the work described in the project report, entitled “**SMART TRAFFIC CONTROL SYSTEM BY VEHICAL DETECTION USING MACHINE LEARNING**” which is submitted by us in partial fulfilment for the award of Bachelor of Technology in the Department of **Computer Science and Engineering**, KHIT, Andhra Pradesh is the record original and independent research work done by us during the academic year 2022–2023 under the supervision of **Dr K Venkata Subba Reddy**. The work is original and has not been submitted for the award of any Degree or Diploma of associate ship or Fellowship or any other similar title to this or any other university.

<u>Name of the Student</u>	<u>Roll No</u>	<u>Signature</u>
<b>PALAPRATHI VISHNU VARDHAN REDDY</b>	<b>198X1A05I4</b>	
<b>ELASAGARAM VENKATESH</b>	<b>198X1A05H4</b>	
<b>SHAIK MAHAMMAD RASOOL</b>	<b>198X1A05H3</b>	
<b>YAMPATI KANUKA LOURDHU RESHMA</b>	<b>198X1A05G6</b>	

## ACKNOWLEDGMENT

We profoundly grateful to express our deep sense of gratitude and respect towards our honorable chairman, **Sri KALLAM MOHAN REDDY**, Chairman of Kallam group for his precious support in the college.

We are thankful to **Dr. M. UMA SANKAR REDDY**, Director, KHIT, GUNTUR for hisencouragement and support for the completion of the project.

We are much thankful to **Dr. B. SIVA BASIVI REDDY**, Principal, KHIT, GUNTUR for his support during and until the completion of the project.

We are greatly indebted to **Prof. V. Rajeev Jetson** Professor, & Head of the department, Computer Science and Engineering, KHIT, GUNTUR for providing the laboratory facilities fully as and when required and for giving us the opportunity to carry the project work in the college.

We are also thankful to our Project Coordinators D. Vinay Kumar, G. Mantru Naik and Mr. Ramprasad Mathi who helped us in each step of our Project.

We extend our deep sense of gratitude to our Internal **Dr. K Venkat Subba Reddy**, Professor, & Head of the department and other Faculty Members & Support staff for their valuable suggestions, guidance and constructive ideas in each and every step, which was indeed of great help towards the successful completion of our project.

### Team Members

<b>P.VISHNU VARDHAN REDDY</b>	<b>198X1A05I4</b>
<b>E.VENKATESH</b>	<b>198X1A05H4</b>
<b>SK.MAHAMMAD RASOOL</b>	<b>198X1A05H3</b>
<b>Y.KANUKA LOURDHU RESHMA</b>	<b>198X1A05G6</b>

## **TABLE OF CONTENTS**

<b>SNO</b>	<b>CHAPTER NAME</b>	<b>Page No</b>
	<b>ABSTRACT</b>	<b>VII</b>
	<b>LIST OF TABLES</b>	<b>VIII</b>
	<b>LIST OF FIGURES</b>	<b>IX</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 Problem Statement	3
	1.2 Scope	3
	1.3 Objective of the Project	4
<b>2</b>	<b>LITERATURE SURVEY</b>	<b>6</b>
	2.1 Literature Survey	6
<b>3</b>	<b>SYSTEM ANALYSIS</b>	<b>15</b>
	3.1 Existing System	15
	3.2 Drawbacks	15
	3.3 Proposed System	16
	3.4 Advantages	16
	3.5 Methodology	17
	3.6 Approaches used in this project	17
<b>4</b>	<b>SYSTEM DESIGN</b>	<b>19</b>
	4.1 System Architecture	19
	4.2 UML Diagrams	20
	4.2.1 Class Diagram	20
	4.2.2 Use case Diagram	21
	4.2.3 Activity Diagram	22
	4.2.4 Sequence Diagram	23
<b>5</b>	<b>SYSTEM REQUIREMENT SPECIFICATION</b>	<b>24</b>
	5.1 Functional Requirements	24
	5.1.1 Hardware Requirements	24

	5.1.2 Software Requirements	24
	5.2 Non- Functional Requirements	24
<b>6</b>	<b>MODULES</b>	<b>26</b>
	6.1 Video Processing	26
	6.2 Vehicle Detection and Classification	26
	6.3 Vehicle Tracking	28
	6.4 Event Processing	28
<b>7</b>	<b>IMPLEMENTATION</b>	<b>29</b>
	7.1 Sample Source Code	29
<b>8</b>	<b>TESTING</b>	<b>36</b>
	8.1 Testing Introduction	36
	8.1.1 Unit Testing	37
	8.1.2 Integration Testing	37
	8.1.3 User Interface Testing	38
	8.2 Test Results	39
<b>9</b>	<b>OUTPUT SCREENS</b>	<b>30</b>
	9.1 Comparison of YOLO V3 with Deep learning models	53
	9.2 Comparison of YOLO V3 with existing algorithm while predicting the traffic density	54
	9.3 A Chart representation for Analysis of Multiple Machine Learning Models	55
<b>10</b>	<b>CONCLUSION</b>	<b>56</b>
<b>11</b>	<b>REFERENCES</b>	<b>57</b>

## **ABSTRACT**

At present, the traffic control frameworks in India, need insight and go about as an open-loop control framework, with no input or detecting system. Present technologies use Inductive loops and sensors to detect the number of vehicles passing by. It is a very inefficient and expensive way to make traffic lights adaptive. Using a simple CCTV camera can improve the conditions. The visual tracking of objects is amongst the most critical areas of computer vision and deep learning. The objective of this work was to develop the traffic control framework by presenting a detecting system, which gives an input to the current system, with the goal that it can adjust the changing traffic density patterns and provides a vital sign to the controller in a continuous activity. Using this method, improvement of the traffic signal switching expands the street limit, saves time for voyaging, and prevents traffic congestion. The framework additionally goes for consolidating exceptional arrangements for clearing the path for emergency vehicles. In this paper, we will detect and track vehicles on a video stream and count those going through a defined line and to ultimately give an idea of what the real-time on-street situation is across the road network. Our real target is to advance the deferral in the travel of vehicles in odd hours of the day. It uses the YOLO (“You Only Look Once”) object detection technique to detect objects on each of the video frames and track those objects over different frames. Once the objects are detected and tracked over different frames, a simple mathematical calculation is applied to count the intersections between the vehicle's previous and current frame positions with a defined line. and as per the count the signals light triggered.

**Keywords** - Traffic control, traffic clog, traffic signal algorithm, Vehicle Detection, Machine Learning.

## **LIST OF TABLES**

<b>TABLE N.O</b>	<b>DESCRIPTION</b>	<b>PAGE N.O</b>
Table 1	Comparison of Deep learning models.	53
Table 2	Comparison of YOLO V3 with other models.	54



## LIST OF FIGURES

FIGURE NO	DESCRIPTION	PAGE NO
Fig 1	Architecture for Traffic light control system.	18
Fig 2	Classification Diagram for RTLMS	19
Fig 3	Use case Diagram for RTLMS	20
Fig 4	Activity Diagram for RTLMS	21
Fig 5	Sequence Diagram for RTLMS	22
Fig 6	Grid cell Division	26
Fig 7	Input video 1	41
Fig 8	Output 1 after Processing	42
Fig 9	Input video 2	43
Fig 10	Output 2 after Processing	44
Fig11	Input video 3	45
Fig 12	Output 3 after Processing	46
Fig 13	Input video 4	47
Fig 14	Output 4 after Processing	48
Fig 15	Input video 5	49
Fig 16	Output 5 after Processing	50
Fig 17	Input video 6	51
Fig 18	Output 6 after Processing	52
Fig 19	Chart for analysis of Multiple Machine Learning Algorithms	55

# 1.INTRODUCTION

Many traffic lights signals operate on the basis of timing mechanism which is optimized to alter the traffic lights after a given period of time resulting the traffic to wait for long. Most of the traffic light controllers are designed in such a way that they have a fixed-cycle and do not take into account the density of traffic coming from each direction.

A relentless increment in the population mainly in India, the number of vehicles and autos builds quickly, which prompts the congested road issue. This proposed framework will have a compelling job to evade the crowded road. Under common conditions, traffic signals control, for the most part, has two deformities:

The consideration of emergency cars is not there. As traffic signal control frameworks are the lack of crisis measures, the intersection dependably meets congested driving conditions and prompts un-necessary financial losses.

Overwhelming Traffic Congestions: With an expanding fleet of automobiles out on the road, overwhelming transit clogs have generously expanded in significant urban communities. This happened ordinarily at the primary intersections regularly toward the beginning of the day that is in office hour and at night after office hours, after the available time. The significant impact of this issue is the expanded time wastage of the general population.

No traffic yet standing still: Since the traffic signal stays red for the current timespan, the street clients have to hold up until the traffic light goes to green. On jumping a red light, they need to pay fines. The answer to this issue is building up a framework that recognizes traffic streams on every street and sets the time for changing the signals automatically. Also, the effective interaction of traffic flags in neighboring intersections are likewise crucial.

Innovation for smart traffic management was undertaken at Carnegie Mellon University for deployment in a test project in Pittsburgh, targeting minimization of road traffic emanations arising from the targeted area.

Dissimilar to other unique control flags that update the plans and stages of traffic signals by the values hardcoded in the controller, the proposed framework consolidates the latest technologies with human-made brainpower. The signs interact with each other, thereby adjusting to the evolving traffic scenarios to minimize the time vehicles waste waiting. Benefiting from the fibre-optic camcorders similar to the ones used efficiently in unique controlling frameworks, this proposed method checks the number of vehicles. It makes changes progressively to stay away from blockages as much as possible. Initial outcomes from the pilot project are encouraging: the amount of time wasted by drivers lingering at traffic signals was reduced by 40, and to traverse the targeted area reduced by 25%.

This proposed model will detect and track vehicles on a video stream and count those going through a defined line and to ultimately give an idea of what the real-time on-street situation is across the road network. Our real target is to advance the deferral in the travel of vehicles in odd hours of the day. It uses the YOLO object detection technique to detect objects on each of the video frames and SORT (Simple Online and Real-time Tracking algorithm) to track those objects over different frames. If we have less traffic or there is more traffic than usual, our model will optimize the light by increasing or decreasing the duration of the light. Traffic situation can be further optimized once we have the real-time count of the vehicles by just applying previous dynamic scheduling algorithms.

## **1.1 PROBLEM STATEMENT**

Vehicle tracking is the technique of utilizing a camera to track a moving vehicle. To increase tracking performance, capturing a vehicle in a video sequence from a security camera is a challenging application. The number of applications for this technology is growing, including traffic control, traffic monitoring, traffic flow, security, and so on. Using this technique, the expected cost will be quite low. Many cities and metropolitan regions have employed video and image processing for traffic surveillance, analysis, and monitoring. In recent years, several approaches for estimating speed have been presented.

This project aims to reduce traffic congestion and unwanted long-time delays and provides a better approach to this by calculating the density of the traffic at each part of the road and simultaneously provides the best solution in order to reduce the congestion and in some cases to stop giving a greenlight indication to some roads where there are less number of vehicles

## **1.2 SCOPE**

Nowadays, many countries suffer from the traffic congestion problems that affect the transportation system in cities and cause serious dilemma. In this way, assuming a continuous population growth, the number of vehicles in large cities will increase as well, but much faster than transportation infrastructure; consequently, traffic congestion will become a pressing issue.

It creates several negative concerns for the environment and society such as increasing number of traffic accidents, economic development, increase in greenhouse gas emission, time spent and health issues.

Our real target is to advance the deferral in the travel of vehicles in odd hours of the day.

Due to continuous growth of population in the world, it is a great challenge for the upcoming generation to manage the traffic system. Much improvement will come in the future. To manage the conventional transport system we should think of the intelligent and automatic way of controlling the system. As the population increases, it increases the number of vehicles also. To control the huge number of vehicles the intelligent methods should be adopted. In future purpose we can use the image sensor or imager. It does it work by

producing the image of the roads. It creates the image by converting the variable attenuation of light into signal that conveys the picture. Imagers used both digital and analog electronics devices.

### **1.3 Objective of Project**

The traffic congestion problem is increasing day by day everywhere around us and must be solved more intelligently considering the continuous increase in traffic in not-so distant future. Among mostproposed solutions like building more infrastructure or improving citizen commute patterns, creatingan intelligent control system to manage the traffic flow comes to be a better and most efficient solution. The more intelligent control system part over here means to use more sophisticated timing algorithms with the help of machine learning algorithms to solve the problem autonomously. Insteadof using fixed time intervals system should be able to set variable timers to each lane.

These variable timers should be computed by the algorithms based on different traffic conditions in real-time. Using such solutions in real life can positively affect various characteristics of a daily commute, especially in metropolitan cities. It may also help in saving many valuable resources like time and fuel while managing the smooth traffic flow with the least stops possible.

The primary motivation for this kind of solution can be said to be the increasing transportation in major cities. Fast transportation systems and rapid transit systems are vital for economic developments for any state. More population means more vehicles on the streets day-by-day. Traditional traffic management systems are not built with the consideration of increasing traffic at this high pace. Thus, it cannot keep up with the variable changing traffic conditions daily.

In the past few years, fields like machine learning and computer vision have developed to be so betterand easy to implement these days. Using concepts like deep learning to detect the vehicles and automatically adjust accordingly helps in many ways for better transportation flow. An algorithm can be created to detect the patterns of the traffic flow and react accordingly. This not only makes the system robust but also more efficient in managing the traffic.

The main goal of this work is to develop a fast and reliable solution to the traffic congestion

problem. The solution must be intelligent and has decision-making capabilities depending on the different situations in the different lanes. There are different phases of the proposed model. The first phase is video processing. The continuous video stream from the traffic camera situated at the traffic signal junction is fed to the model. During the video processing, the frames are extracted from the video

input. There are hundreds of frames present in a stream of few seconds, so it is not possible to process each frame and it is not required because there is no such significant change in the vehicle density between two consecutive frames. Therefore, each frame after a particular time interval is taken. Then shrinking the size of the frame is done from the actual size of the frame to 416 x 416 pixels. After this phase, the processed frame is fed to the model for object detection.

There are two separate models developed for object detection using each framework i.e., YOLOv3 and YOLOv5. The neural network extracts the features and identifies the object using bounding boxes. After object detection, the model predicts the class of the vehicle from a car, bike, truck, etc. The total vehicle count of a lane is passed to the dynamic traffic signal timer algorithm. The algorithm considers the density of all the other lanes and calculates the relativity between them. Depending on which, it classifies the lane in either low, medium, or high vehicle density class, it decides the green signal timer for a particular lane.

## **2. LITERATURE SURVEY**

### **2.1 LITERATURE SURVEY:**

The research intends to develop the vehicle detection and counting system using image processing. Overall works are software development of a system that requires a video stream and capture to a video frame. They consist of the following components: background road without any moving vehicle and the frame with moving vehicles. The system is designed to find the differentiation which is the moving vehicles and find the number of moving vehicles from the video frame.

The vehicle detection and counting system consists of four major components: 1) image acquisition,

2) image analysis, 3) object detection and counting, and 4) Display Result. The experiment has been conducted in order to access the following qualities: 1) Usability, to prove that the system can determine vehicle detection and counting under the specific condition layout. 2) Efficiency, to show that the system can work with high accuracy. [1]

Vehicle detection and statistics in highway monitoring video scenes are of considerable significance to intelligent traffic management and control of the highway. With the popular installation of traffic surveillance cameras, a vast database of traffic video footage has been obtained for analysis. Generally, at a high viewing angle, a more-distant road surface can be considered. The object size of the vehicle changes greatly at this viewing angle, and the detection accuracy of a small object far away from the road is low. In the face of complex camera scenes, it is essential to effectively solve the above problems and further apply them. In this article, we focus on the above issues to propose a viable solution, and we apply the vehicle detection results to multi-object tracking and vehicle counting. [1]

The traffic flow is usually estimated to evaluate the traffic state in traffic management, and vehicle counting is a key method for estimating traffic flow. With wide deployment of cameras in urban transportation systems, the surveillance video becomes an important data source to conduct vehicle counting. However, the efficiency and accuracy of vehicle counting are seriously affected by the complexity of traffic scenarios. In this paper, we employ the virtual loop method to improve the quality of video-based vehicle counting

method. As details, the expectation-maximization (EM) algorithm is fused with the Gaussian mixture model (GMM) for improving the segmentation quality of moving vehicles.

In addition, a restoration method is designed to remove noise and fill holes for obtaining a better object region. Finally, a morphological feature and the color-histogram are utilized to solve occlusion issues. The effectiveness and efficiency experiments show that the proposed approach can improve the vehicle segmentation result and the vehicle occlusion detection. The accuracy of vehicle counting can also be improved significantly and reach 98%. Highlights EM-based Gaussian mixture model is proposed for better vehicle segmentation. A restoration method is designed to restore the vehicle region in the difference image. A morphological feature and color histogram are combined for detecting occlusion. [2].

Intelligent Traffic Flow Analysis (TFA) systems include sensing and data processing techniques that attempt to improve the traditional traffic systems [1]. They integrate additional hardware of sensors, digital traffic signs and cameras and apply advanced techniques to process the data provided by the hardware to intelligently improve vehicles and traffic flow [2]. For example, efficiently adjusting the lights phasing and timing of a traffic light system in such a way that improves the traffic flow and reduce the delay [3]. In a video-based vehicle counting, the TFA system makes use of video stream that contains a sequence of images [4]. This is essential when digital image processing is the only practical technology for counting, feature extraction, pattern recognition, projection and multi-scale signal analysis of the system which can know the vehicle counting, the system also needs to identify the exact location to be more accurate in counting vehicles [5]. Basically, the video is divided into frames, the frames are further transformed into colored or gray level frames and the frames are given to the system as inputs. Then the system applies different types of algorithms for preprocessing, object detection, identification and tracking to perform vehicle counting [6]. The tracking is focusing on a particular region of an image which is considered to be important for the collection of data called as Region of Interest (ROI) and after the system collects the data it will ensure not to reconsider processing/counting the same object again. Vehicle tracking is an entire measurement process in many defined frames for the same object location [7].

A camera-based scheme is proposed for detecting vehicles at user-defined virtual loops,



simulating the operation of inductive loops. False vehicular detections are minimized by a combination of efficient edge detection and color information. The experimental results suggest that the proposed scheme potentially can detect and count vehicles at user-defined virtual loops accurately (with more than 98% correct detection rate, in average), besides being more robust to cast shadows and sudden illumination changes than comparable methods that represent the state of the art [3].

Traffic monitoring systems are used to capture important data for traffic management, and are based on a range of different sensors technologies. For example, GPS-based systems can be used. Traffic monitoring systems are used to capture important data for traffic management, and are based on a range of different sensors technologies. For example, GPS-based systems can be used to predict a vehicle location [3].

Traffic monitoring systems are used to capture important data for traffic management, and are based on a range of different sensors technologies. For example, GPS-based systems can be used to predict a vehicle location

Multiple sensors, such as cameras and laser, can be combined to provide even more accurate estimates of a vehicle localization. Also, camera-based systems can be used in vehicle classification and in license plate detection. Besides, data such as vehicle count, speed and street occupancy provide relevant information and can assist in urban planning. [3].

Nowadays, traffic monitoring systems often rely on inductive loop sensors or magnetic sensors to obtain traffic information. Nevertheless, traffic monitoring systems based on video cameras have advantages over other types of sensors. However, camera-based systems can be used for traffic monitoring and for detecting vehicles at user defined virtual loops, which can provide more information about the vehicular traffic (e.g., types of vehicles in circulation, vehicle size and speed) in comparison to other sensor technologies. [3].

This paper presents a new method to track and count vehicles in video traffic sequences. The proposed method uses image processing, particle filtering, and motion coherence to group particles in videos, forming convex shapes that are analyzed for potential vehicles. This analysis takes into consideration the convex shape of the objects and background information to merge or split the groupings. [4]

After a vehicle is identified, it is tracked using the similarity of color histograms on windows

centered at the particle locations.

Information about traffic conditions can be used to synchronize traffic lights, help users selecting the best routes and help governments to plan the expansion of the traffic system (e.g., building new roads). Data such as the number of vehicles, speed and track occupation are very important for traffic management. Normally, to obtain such information methods based on inductive loops, ultrasonic sensors and microwave are used. But in recent years, the use of computer vision in obtaining this information has been widely investigated because videos provide much more information than the above-mentioned methods, including the possibility of using cameras already installed. [4].

Robust and reliable traffic surveillance system is an urgent need to improve traffic control and management. Vehicle flow detection appears to be an important part in surveillance system. The traffic flow shows the traffic state in fixed time interval and helps to manage and control especially when there's a traffic jam. In this paper presents an effective traffic surveillance system for detecting and tracking moving vehicles in various nighttime environments.

The proposed algorithm is composed of four steps: headlight segmentation and detection, headlight pairing, vehicle tracking, vehicle counting and detection. First, a fast segmentation process based on an adaptive threshold is applied to effectively extract bright objects of interest. The extracted bright objects are then processed by a spatial clustering and tracking procedure that locates and analyzes the spatial and temporal features of vehicle light patterns, and identifies and classifies moving cars and motorbikes in traffic scenes. The experimental results show that the proposed system can provide real-time and useful information for traffic surveillance [5].

In recent years, the traffic surveillance system is put forward extensively to be discussed and studied because it can provide meaningful and useful information such as traffic flow density, the length of queue, average traffic speed and the total vehicle in fixed time interval. Generally, the traffic surveillance system requires more sensors. The common traffic sensors include (1) push button (detecting pedestrian demand), (2) loop detectors (detecting vehicle presence at one point), (3) magnetic sensors (magnetometers), (4) radar sensors, (5)

microwave detectors, (6) video cameras. Video camera is a promising traffic sensor because of its low cost and its potential ability to collect a large amount of information (such as the number of vehicles, vehicles speed/acceleration, vehicle class, vehicles track) which can also infer higher-level information (incidents, speeding, origin- destination of vehicles, macroscopic traffic statistics, etc.). The video cameras (CCD or CMOS) are connected to a computer that performs images/video processing, object recognition and object tracking. Numerous research projects aiming to detect and track vehicle from stationary rectilinear cameras have been carried out in terms of measuring traffic performance during the past decades [1]-[3]. It is widely recognized that vision-based are flexible and versatile in traffic monitoring applications if they can be made sufficiently reliable and robust [4],[5]. As the key goal for a traffic surveillance system, the evaluation of traffic conditions can be represented by the following parameters: traffic flow rate, average traffic speed, the length of queue and traffic density.

Vehicle detection and tracking plays an effective and significant role in the area of traffic surveillance system where efficient traffic management and safety is the main concern. In this paper, we discuss and address the issue of detecting vehicle / traffic data from video frames. Although various research has been done in this area and many methods have been implemented, still this area has room for improvements. With a view to do improvements, it is proposed to develop a unique algorithm for vehicle data recognition and tracking using Gaussian mixture model and blob detection methods [6].

First, we differentiate the foreground from background in frames by learning the background. Here, foreground detector detects the object, and a binary computation is done to define rectangular regions around every detected object. To detect the moving object correctly and to remove the noise some morphological operations have been applied. Then the final counting is done by tracking the detected objects and their regions. The results are encouraging, and we got more than 91% of average accuracy in detection and tracking using the Gaussian Mixture Model and Blob Detection methods.

This paper presents an intelligent vehicle counting method based on blob analysis in traffic surveillance. The proposed algorithm is composed of three steps: Processing is done by three main steps: moving object segmentation, blob analysis, and tracking. A vehicle is

modeled as a rectangular patch and classified via blob analysis. By analyzing the blob of vehicles, the meaningful features are extracted. Tracking moving targets is achieved by comparing the extracted features and measuring the minimal distance between two temporal images [7].

In addition, the velocity of each vehicle and the vehicle flow through a predefined area can be calculated by analyzing blobs of vehicles. The experimental results show that the proposed system can provide real-time and useful information for traffic surveillance.

There exist several vehicle counting methods such as Particle Filtering or Headlight Detection, among others. Although Principal Component Pursuit (PCP) is considered to be the state-of-the-art for video background modeling, it has not been previously exploited for this task. This is mainly because most of the existing PCP algorithms are batch methods and have a high computational cost that makes them unsuitable for real-time vehicle counting.

In this paper, we propose to use a novel incremental PCP-based algorithm to estimate the number of vehicles present in top-view traffic video sequences in real-time. We test our method against several challenging datasets, achieving results that compare favorably with state-of-the-art methods in performance and speed: an average accuracy of 98% when counting vehicles passing through a virtual door, 91% when estimating the total number of vehicles present in the scene, and up to 26 fps in processing time [8].

Recently, the intelligent transportation system is monitoring traffic flow with a high-definition camera and improving its performance by integrating Doppler method-based radar sensor and detection data. A radar has more advantages in terms of night, bad weather and speed accuracy [9].

However, its performance is much reduced when it runs at a crowding low speed, which occurs frequently. As a result, it is required to develop a video-based algorithm that can detect and count vehicles even at a crowding low speed driving and measure approximate speed.

This paper presents a solution to solve the car detection and counting problem in images

acquired by means of unmanned aerial vehicles (UAVs). UAV images are characterized by a very high spatial resolution (order of few centimeters), and consequently by an extremely high level of details which calls for appropriate automatic analysis methods. The proposed method starts with a screening step of asphalted zones in order to restrict the areas where to detect cars and thus to reduce false alarms. Then, it performs a feature extraction process based on scalar invariant feature transform thanks to which a set of key points is identified in the considered image and opportunely described. Successively, it discriminates between key points assigned to cars and all the others, by means of a support vector machine classifier.

The last step of our method is focused on the grouping of the key points belonging to the same car in order to get a “one key point-one car” relationship. Finally, the number of cars present in the scene is given by the number of final key points identified. The experimental results obtained on a real UAV scene characterized by a spatial resolution of 2 cm show that the proposed method exhibits a promising car counting accuracy [10].

SNO	AUTHORS	TECHNIQUES	ADVANTAGES	LIMITATIONS
1.	Pornpanomchai , C.,Liamsanguan, T.,&Vannakosit,V.	Vehicle detection and counting from a video frame (2008)	The system can determine vehicle detection and counting under the specific condition layout.	It is not efficient at detection of occlusion of the vehicles which affects the accuracy of the counting as well as classification.
2.	Xia, Y., Shi, X., Song, G., Geng, Q., & Liu, Y	Towards improving quality of video- based vehicle counting method for traffic flow estimation (2016)	The effectiveness and efficiency experiments show that the proposed approach can improve the vehicle segmentation result and the vehicle occlusion detection.	The efficiency and accuracy of vehicle counting are seriously affected by the complexity of traffic scenarios
3.	Barcellos, P., Bouvié, C., Escouto, F. L., & Scharcanski, J.	Image-based approach for detecting vehicles in user-defined virtual inductive loops (2015)	False vehicular detections are minimized by a combination of efficient edge detection and color information.	It suffers from performance problems.
4.	Bouvie, C., Scharcanski, J., Barcellos, P., & Escouto, F. L	Tracking and counting vehicles in traffic video sequences using particle filtering (2013)	This analysis takes into consideration the convex shape of the objects and background information to merge or split the groupings.	It uses large amount of data.
5.	Salvi, G.	An Automated Nighttime Vehicle Counting and Detection System for Traffic Surveillance (2014)	The traffic flow shows the traffic state in fixed time interval and helps to manage and control especially when there's a traffic jam	Only helpful in certain intervals of time period.

6.	Bhaskar, P. K., & Yong, S. P.	Image processing-based vehicle detection and tracking method (2014)	Foreground detector detects the object and a binary computation is done to define rectangular regions around every detected object.	Unwanted objects are detected.
7.	Chen, T. H., Lin, Y. F., & Chen, T. Y.	Intelligent Vehicle Counting Method Based on Blob Analysis in Traffic Surveillance (2007)	Tracking moving targets is achieved by comparing the extracted features and measuring the minimal distance between two temporal images.	It is very much time-consuming.
8.	Quesada, J., & Rodriguez, P.	Automatic vehicle Counting Method Based on Principal Component Pursuit Background Modeling (2016)	It estimates the number of vehicles present in the current frame through a spatiotemporal processing.	Maximum error of 4 units per frame.
9.	Jang, H., Won, I. S., & Jeong, D. S.	Automatic Vehicle Detection and Counting Algorithm (2014)	A radar has more advantages in terms of night, bad weather and speed accuracy	High Maintenance cost is required.
10.	Moranduzzo, T., & Melgani, F.	Automatic Car Counting Method for Unmanned Aerial Vehicle Images (2014)	The number of cars present in the scene is given by the number of final key points identified.	Used only for detecting cars.

## **3.SYSTEM ANALYSIS**

### **3.1 Existing System**

The traffic lights used in India are basically pre-timed wherein the time of each lane to have a green signal is fixed. In a four-lane traffic signal one lane is given a green signal at a time. Thus, the traffic light allows the vehicles of all lanes to pass in a sequence. So, the traffic can advance in either straight direction or turn by 90 degrees as shown in Fig.1. So even if the traffic density in a particular lane is the least, it has to wait unnecessarily for a long time and when it gets the green signal it unnecessarily makes other lanes wait for even longer durations.

Another Existing method based on OpenCV with background subtraction and blob tracking method achieve good results for fixed cameras only. However, it produces abnormal results for unbalanced handheld cameras.

### **3.2 DRAWBACKS**

The problem with the existing traffic system is that for every minute the vehicles at the 4-way junction (cross-roads) will be large in number due to this there is always a lot of congestion on such roads. Even though there are no vehicles at a particular side, the traffic signal will still glow green for a fixed time.

This causes a lot of problems such as:

- The vehicles on the other part of the road have to keep waiting for their turn.
- The congestion on the roads after a point gets even worse.
- In some cases, most of the ambulances cannot move and get stuck in the congestion



### **3.3 PROPOSED SYSTEM**

A typical traffic monitoring system consists of a network of traffic cameras, hardware, and software, which processes live video images captured from traffic scenes on roadways and transmits the extracted parameters in real-time to a control room where managers can monitor the traffic events.

The framework is divided into four different modules: Video processing; Event processing, Signal time calculation and Visualization.

The video processing module receives a video input from a source, such as a traffic camera, an online video stream or any other video source. Each frame is then forwarded into a state-of-the-art vehicle detection and classification model. The model returns the parameters for the location, class, and confidence score for each of the vehicles detected in the frame. For vehicle tracking first, we identify ROIs in the scene. For each new vehicle detection within a ROI, we start to track its position until it leaves this region. For the other vehicles detected within a ROI, that are being tracked from previous frames, we update their tracking positions.

All the results from the previous module are sent to the event processor module where all the information is analyzed to extract significant events. One subtask of the event processing module is vehicle counting. Within the ROI there is a virtual counting zone where the vehicles are counted by class type. In this module, other subtasks can be added to assist in traffic management, such as accident detection, speed monitoring, traffic density measurement or traffic flow prediction.

### **3.4 ADVANTAGES**

- Reduce the traffic congestion.
- Avoid roadblocks and prevent accidents from happening by effectively analyzing vehicles every now and then.
- Reduce waiting period for the vehicles and people and also save fuel

## 3.5 METHODOLOGY

**Step 1:** The video processing module receives a video input from a source, such as a traffic camera, an online video stream or any other video source.

**Step 2:** Each frame is then forwarded into a state-of-the-art vehicle detection and classification model.

**Step 3:** It identifies ROIs in the scene. For each new vehicle detection within a ROI, it starts to track its position until it leaves this region.

**Step 4:** All the results from the previous module are sent to the event processor module where all the information is analyzed to extract significant events.

**Step 5:** Vehicles are counted according to their classes. Once the vehicle leaves the counting zone, the vehicle stops being tracked, and we update its status to *tracking = False*.

**Step 6:** Based on the count, the traffic signal light will be changed dynamically.

## 3.6 Approaches used in this Project

### YOLO Algorithm

YOLOv3 [RF18] is a state-of-the-art, real-time object detection system. This model is extremely fast and accurate, has several advantages over classifier-based systems such as Faster R-CNN. At inference time, this model looks at the whole image, so its predictions are informed by the global context in the image. Also, this version of YOLO (v3) is very recent and that could explain why the other implementations did not perform as well (considerable slower) as the Darknet framework.

YOLOv3 uses Darknet-53, a new 53-layer CNN for feature extraction. To run the object detection model, we needed some weights for the feature extraction layers (convolutional layers). We used the weights of a pre-trained model trained on the ImageNet dataset with 1000 object classes and then fine-tuned on the COCO dataset [LMB+14] with 80 object classes.

One of the advantages of YOLO is that it makes predictions with a single network evaluation.

YOLO divides the input image into a  $S \times S$  grid. Each grid cell predicts only one object, e.g., in Figure 3.3 the green cells try to predict the vehicles whose center (orange dot) fall within the same. To locate the objects, each cell predicts a fixed number  $B$  of bounding boxes and each box includes a confidence score, four parameters for the bounding box and  $C$  classes probabilities, but detects only one object regardless of the number of boxes  $B$  (Figure 3.4).

YOLOv3 uses multi-label classification meaning that same objects can have more than one label e.g., the model may label a pick-up van with two labels (car and truck). As we are using a pre-trained model, we made an adaptation of the non-maximal suppression used in YOLOv2, so we remove the duplicated labels and bounding boxes with the lowest confidence score, as we don't want to have more than one label per vehicle detected.

On the video processing module once an image is forwarded to the vehicle detection and classification model, we will obtain the data of each detection as a result and not an annotated image. The data obtained from the model is the  $x$  and  $y$  coordinates of the top left corner of the bounding box, the width the height of the bounding box, a class label and a confidence score for the class of each detection.

YOLOv3 is a state-of-the-art object detector. It's fast and accurate. It has been improved to detect smaller objects and is able to predict bounding boxes at three different scales using a similar concept to feature pyramid networks. Even using a pre-trained model on generic objects proved to be an accurate vehicle detector in images of traffic scenes.

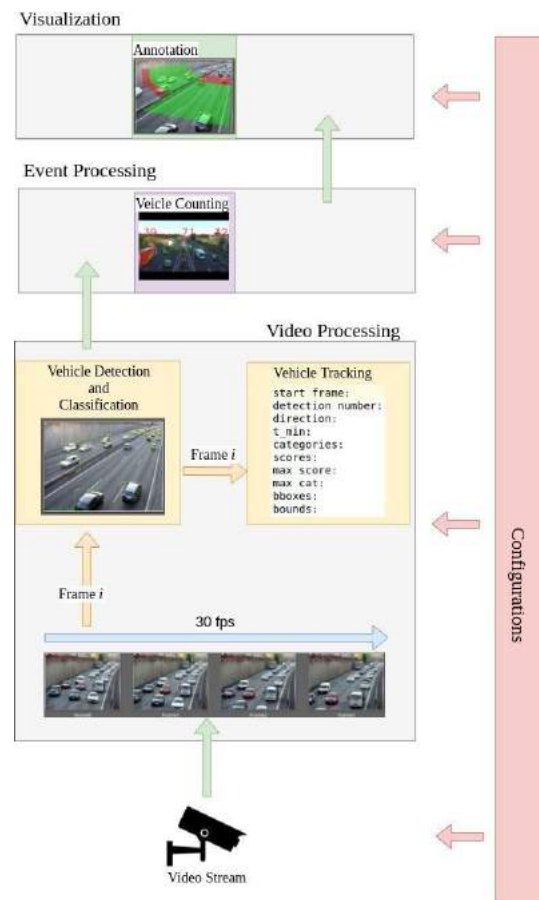
### **ROI-Based Processing**

A region of interest (ROI) is a portion of an image that you want to filter or operate on in some way. You can represent an ROI as a binary mask image. In the mask image, pixels that belong to the ROI are set to 1 and pixels outside the ROI are set to 0. The toolbox offers several options to specify ROIs and create binary masks.

The toolbox supports a set of objects that you can use to create ROIs of many shapes, such circles, ellipses, polygons, rectangles, and hand-drawn shapes. After you create the objects, you can modify their shape, position, appearance, and behaviour. For more information about the ROI shapes, see Create ROI Shapes.

## 4.System Design

### 4.1 System Architecture:



**Fig:1** Architecture for Traffic light control system.

The system architecture has following stages:

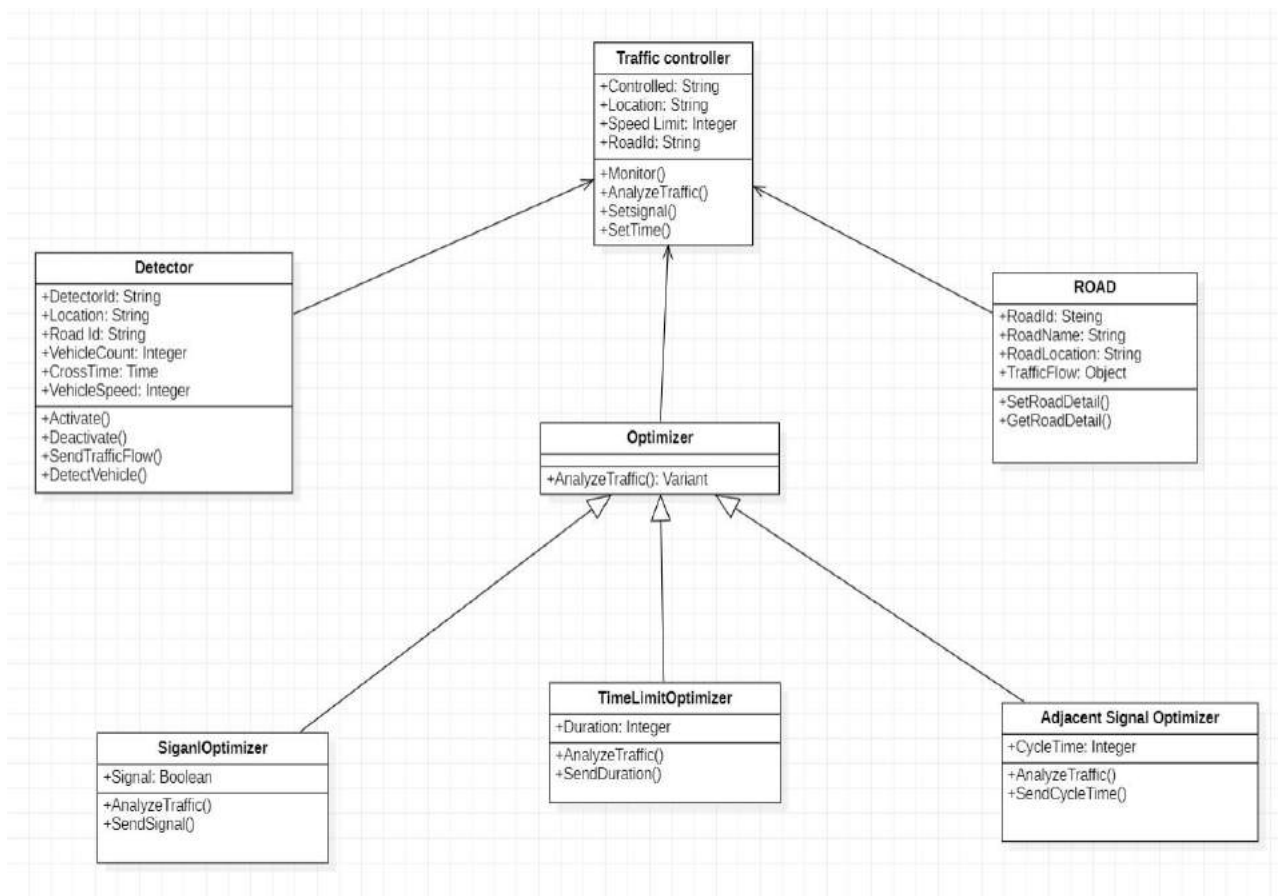
**Visualization:** The object detected in a frame may correspond to one or more tracked boundingboxes. The correlation filters stop updating when the vehicle disappears from the camera view. Subsequently, they are added to the passive trackers' list after eliminating from the active trackers' list.

**Event Processing:** In this condition, the tracked object is isolated from the detected object by an adequate distance. This state is identified as the detected state. There after the detected object is assigned to a new correlation filter-based tracker to initiate the tracking process. The corresponding tracker is added to active trackers' list.

**Video Processing:** If the coordinates of tracker corresponding to each vehicle  $CF_t^i$  reaches the border of the frame, then the condition is defined as terminated. If the object is occluded, it reappears in the scene after a short period. However, object disappears completely as the result of vehicle moving out of camera view.

## 4.2 UML Diagrams

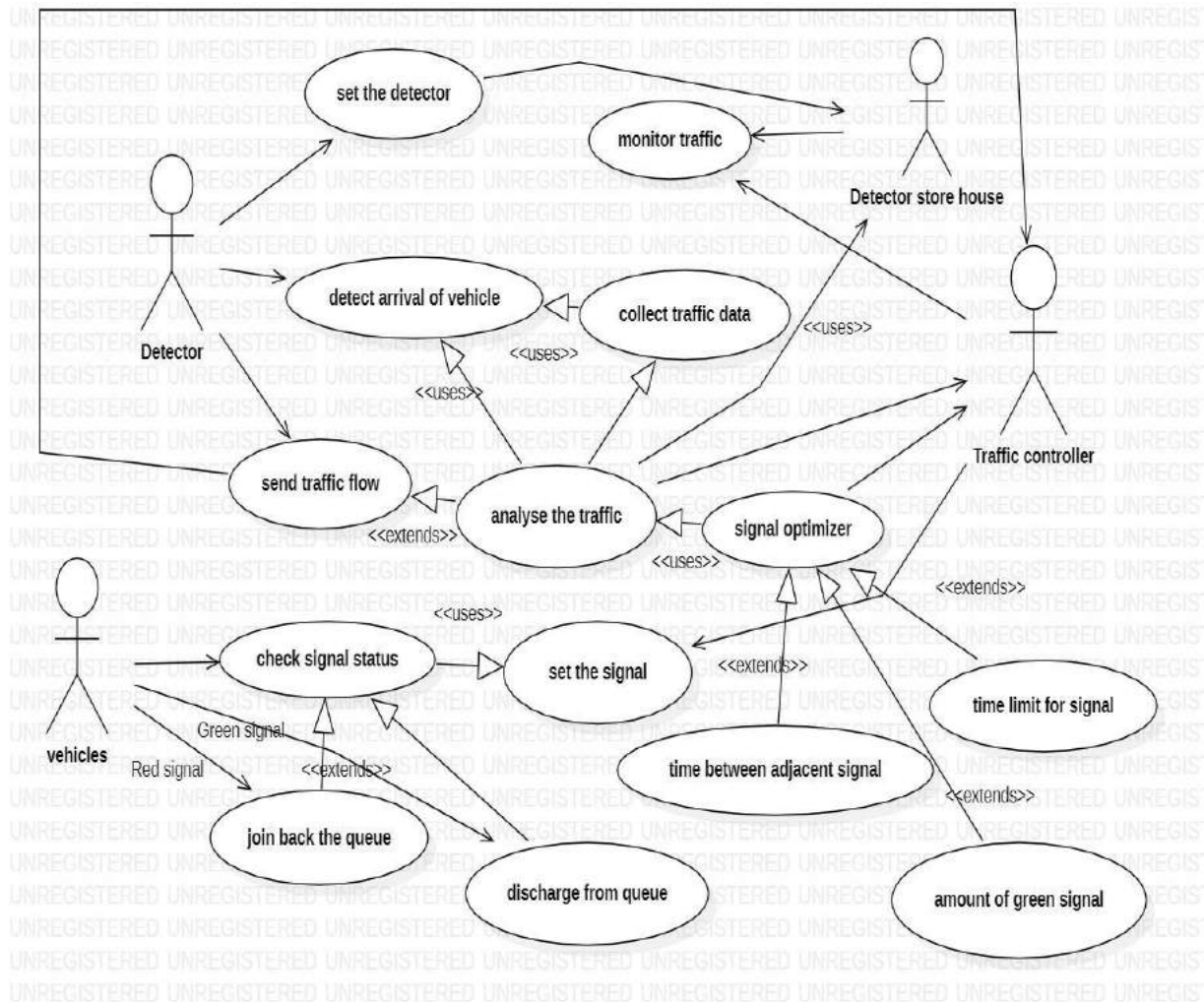
### 4.2.1 Class Diagram:



**Fig 2:** Class diagram for RTLMS (Road Traffic Light Management System).

Class diagram describes the static designs which in turn help us to understand the functional requirements of the system, how the system is composed from the description of classes. A class diagram shows a set of classes, interfaces collaborations and their relationships

## 4.2.2 Use case Diagram:



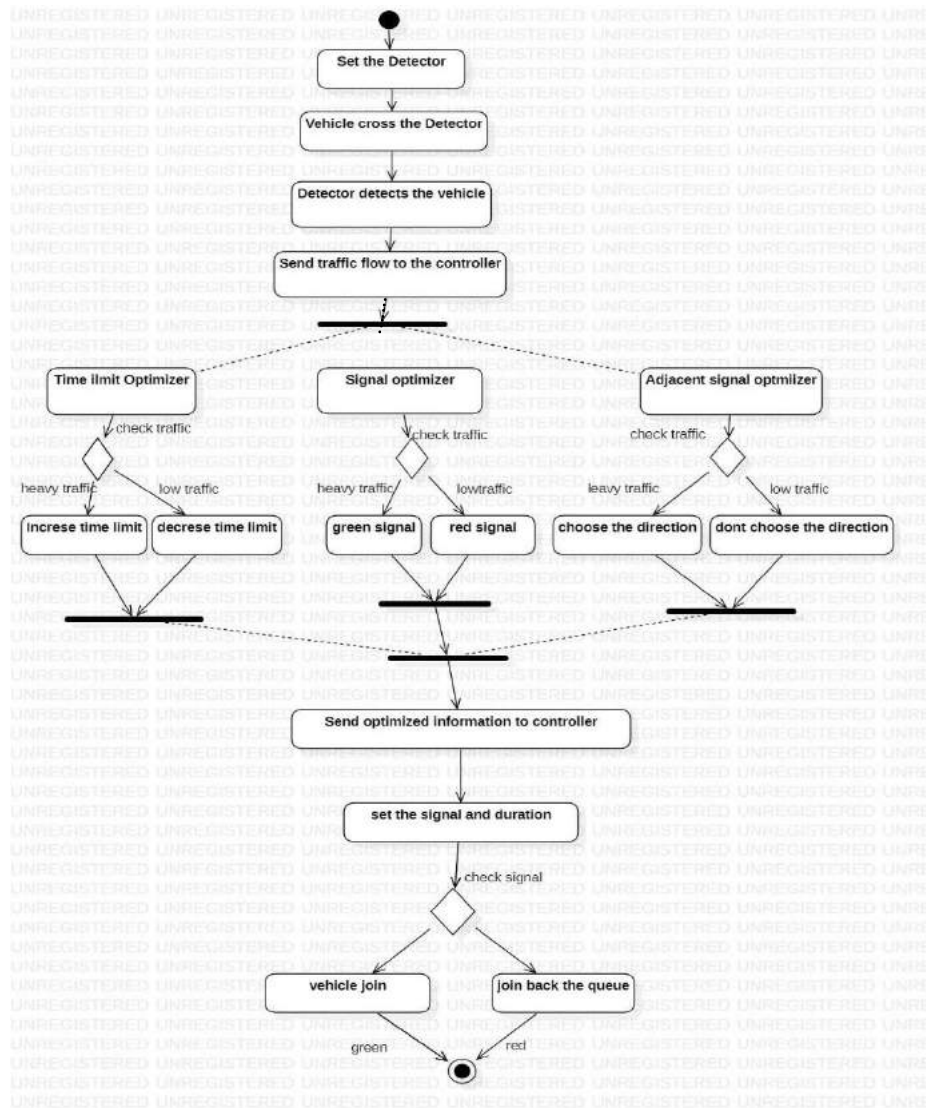
**Fig 3:** Use case Diagram for RTLMS (Road Traffic Light Management System).

Use case diagram captures the functional requirement of the system and it's the interaction between the actor the system.

The purpose of a use case is to:

- Manage scope
- Establish requirements
- Outline the ways a user will interact with the system
- Visualize system architecture

### 4.2.3 Activity Diagram:



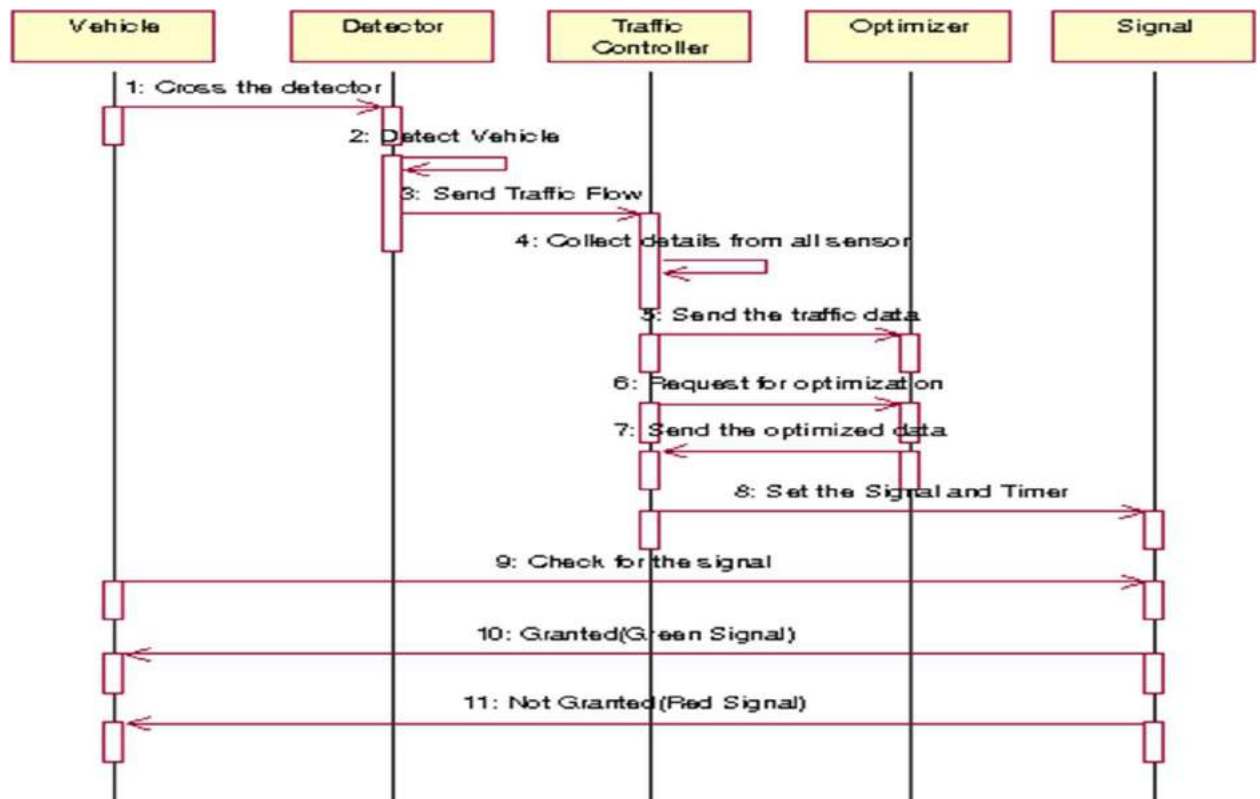
**Fig 4:** Activity Diagram for RTLMS (Road Traffic Light Management System).

It visually presents a series of actions or flow of control in a system similar to a flowchart or a data flow diagram.

Activities contain several activity nodes and activity edges that represent the sequence of tasks in a workflow that result in a behavior.

In activity diagrams, an object node is an abstract activity node that helps to define the object flow in an activity. An object node indicates that an instance of a classifier might be available at a particular point in the activity.

#### 4.2.4 Sequence diagram:



**Fig 5:** sequence diagram for RTLMS (Road Traffic Light Management System).

Sequence diagrams, commonly used by developers, model the interactions between objects in a single use case. They illustrate how the different parts of a system interact with each other to carry out a function, and the order in which the interactions occur when a particular use case is executed.

A system sequence diagram should specify and show the following:

- External actors
- Messages (methods) invoked by these actors
- Return values (if any) associated with previous messages
- Indication of any loops or iteration area



## **5.SOFTWARE REQUIREMENTS SPECIFICATIONS**

### **5.1 FUNCTIONAL REQUIREMENTS 5.1.1**

#### **HARDWARE REQUIREMENTS:**

- Processor : Intel Core i3 or Newer
- Storage : 256 GB
- Ram : 4GB
- Monitor : 15 VGA Colour6

#### **5.1.2 SOFTWARE REQUIREMENTS:**

- Operating system : - Windows XP/7/10/11.
- Coding Language : Python
- Compiler/Shell : IDLE

### **5.2 NON-FUNCTIONAL REQUIREMENTS**

Non-functional requirements are requirements which specify criteria that can be used to judge the operation of a system, rather than specific behaviors. This should be contrasted with functional requirements that specify specific behavior or functions. Typical non-functional requirements are reliability, scalability, and cost. Non-functional requirements are often called the ileitis of a system. Other terms for non-functional requirements are "constraints", "quality attributes" and "quality of service requirements".

**Reliability:** If any exceptions occur during the execution of the software, it should be caught and thereby prevent the system from crashing

**Scalability:** The system should be developed in such a way that new modules and functionalities can be added, thereby facilitating system evolution.

**Cost:** The cost should be low because of the free availability of software package.

## **6.MODULES**

**The modules listed in this project are as follows:**

- 1.Video Processing
- 2.Vehicle detection and classification
- 3.Vehicle Tracking
- 4.Event Processing

### **6.1 Video Processing:**

Video processing covers most of the image processing methods but also includes methods where the temporal nature of the video data is exploited. The goal of the video processing module is to analyse video images from traffic monitoring cameras with the purpose of first finding vehicles and then extract some parameter of these vehicles, e.g., location, type of vehicle, travelling direction, etc.

In this work, it has been used to capture videos from traffic surveillance cameras that are located on motorways. The video is then fed into the video processing module for processing. The first step is to consider the video frames as images and process those images using image processing techniques. To read the video, frame by frame, we used OpenCV which is an open-source computer vision library that was specially developed for video and images processing. In this work video processing can be considered as a combination of three tasks: Vehicle detection; Vehicle classification; and Vehicle tracking.

### **6.2 Vehicle Detection and Classification:**

The vehicle detection and classification tasks are the most important in this framework, as traffic surveillance applications for vehicle detection relies upon fast and accurate vehicle detection capabilities. As stated, before in the literature review in the last years CNN have shown great advantages in object detection and classification. For this task in our framework, we used YOLOv3 as a state-of-the-art, real-time object detection system.

YOLOv3 uses Darknet-53 a new 53-layer CNN for feature extraction. To run the object detection model, we needed some weights for the feature extraction layers (convolutional layers). We used the weights of a pre-trained model trained on the ImageNet dataset with 1000 object classes and then fine-tuned on the COCO dataset [LMB<sup>+</sup>14] with 80 object classes.

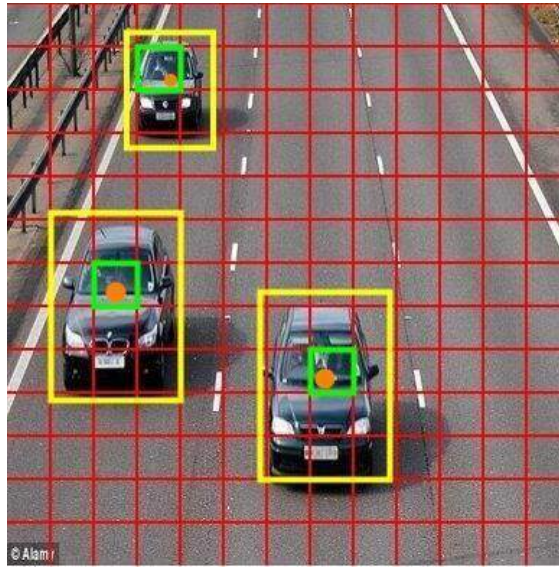


Fig 6: Grid cell Divisio

## 6.3 Vehicle Tracking

YOLO has excellent object detection and classification capabilities; however, its design is to detect objects in a single image. At each frame repeats the same steps, detection, classification and probability scores of every object detected. Its advantage over similar object detection models is that it can repeat these steps over many frames quickly. YOLOv3 trained on the COCO dataset can process 30 fps.

As there are several vehicles in an image that we have to track, we have reviewed works in the literature that could do multi-object tracking. Multi-object tracking requires both an estimation of an unknown number of objects in a video and their respective path. A common approach to multi-object tracking is tracking-by-detection where first an object detector is applied to each frame and second a tracker is used to associate the detections in the video sequences. This approach suited the architecture of our framework perfectly. A typical problem for this approach is the limited performance of the underlying object detector. YOLOv3 is a state-of-the-art object detector so it should be able to handle the problems presented by this approach.

## 6.4 Event Processing

Its function is to automatically detect any events, generate useful data related to traffic status, and create or manage alarms that might be required for the traffic surveillance/control system to operate safely and reliably. Such module plays an important role in the system as it helps to monitor and control events. In this work, we implemented the vehicle counting sub-module, but in the future other modules can be added.

It has been calculated and track the centroid of the bounding box of each vehicle. When a vehicle enters the tracking zone, its status is updated to *tracking = True* and *counted = False*. Then its position is tracked on the following frames. If it reaches the counting zone, we update its status *counted = True*, indicating that the vehicle was counted. Vehicles are counted according to their classes.

# 7.IMPLEMENTATION

## 7.1 SAMPLE SOURCE CODE

```
# import the necessary packages
# python yolo_video.py --input input_videos/bridge.mp4 --output output_videos/bridge.avi --yolo
yolo-coco
import numpy as np
import imutils
import time
from scipy import spatial
import cv2
from input_retrieval import *

#All these classes will be counted as 'vehicles'
list_of_vehicles = ["bicycle","car","motorbike","bus","truck", "train"]
# Setting the threshold for the number of frames to search a vehicle for
FRAMES_BEFORE_CURRENT = 10
inputWidth, inputHeight = 416, 416

#Parse command line arguments and extract the values required
LABELS, weightsPath, configPath, inputVideoPath, outputVideoPath,\
    preDefinedConfidence, preDefinedThreshold, USE_GPU= parseCommandLineArguments()

# Initialize a list of colors to represent each possible class label
np.random.seed(42)
COLORS = np.random.randint(0, 255, size=(len(LABELS), 3),
    dtype="uint8")

# PURPOSE: Displays the vehicle count on the top-left corner of the frame
# PARAMETERS: Frame on which the count is displayed, the count number of vehicles
# RETURN: N/A
def displayVehicleCount(frame, vehicle_count):
    ##      cv2.putText(
    ##          frame, #Image
    ##          'Detected Vehicles: ' + str(), #Label
    ##          (20, 20), #Position
    ##          cv2.FONT_HERSHEY_SIMPLEX, #Font
    ##          0.8, #Size
    ##          (0, 0xFF, 0), #Color
    ##          2, #Thickness
    ##          cv2.FONT_HERSHEY_COMPLEX_SMALL,
    ##          )
    i=vehicle_count
    cv2.imwrite('test.jpg',frame)
    cv2.waitKey(1)
    while(i>0):
        frame=cv2.imread('test.jpg')
        cv2.waitKey(1)
        i=i-1
        cv2.putText(
            frame, #Image
            'GREEN TIME: ' + str(i), #Label
```

```

        (20, 60), #Position
        cv2.FONT_HERSHEY_SIMPLEX, #Font
        0.8, #Size
        (0, 0xFF, 0), #Color
        2, #Thickness
        cv2.FONT_HERSHEY_COMPLEX_SMALL,
    )
    cv2.imshow('Frame', frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
    time.sleep(1)
i=10
while(i>0):
    frame=cv2.imread('test.jpg')
    cv2.waitKey(1)
    i=i-1
    cv2.putText(
        frame, #Image
        'RED TIME: ' + str(i), #Label
        (20, 60), #Position
        cv2.FONT_HERSHEY_SIMPLEX, #Font
        0.8, #Size
        (0, 0, 0xFF), #Color
        2, #Thickness
        cv2.FONT_HERSHEY_COMPLEX_SMALL,
    )
    cv2.imshow('Frame', frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
    time.sleep(1)

# PURPOSE: Determining if the box-mid point cross the line or are within the range of 5 units
# from the line
# PARAMETERS: X Mid-Point of the box, Y mid-point of the box, Coordinates of the line
# RETURN:
# - True if the midpoint of the box overlaps with the line within a threshold of 5 units
# - False if the midpoint of the box lies outside the line and threshold
def boxAndLineOverlap(x_mid_point, y_mid_point, line_coordinates):
    x1_line, y1_line, x2_line, y2_line = line_coordinates #Unpacking

    if (x_mid_point >= x1_line and x_mid_point <= x2_line+5) and\
        (y_mid_point >= y1_line and y_mid_point <= y2_line+5):
        return True
    return False

# PURPOSE: Displaying the FPS of the detected video
# PARAMETERS: Start time of the frame, number of frames within the same second
# RETURN: New start time, new number of frames
def displayFPS(start_time, num_frames):
    current_time = int(time.time())
    if(current_time > start_time):

```

```

        os.system('clear') # Equivalent of CTRL+L on the terminal
        print("FPS:", num_frames)
        num_frames = 0
        start_time = current_time
    return start_time, num_frames

# PURPOSE: Draw all the detection boxes with a green dot at the center
# RETURN: N/A
def drawDetectionBoxes(idxs, boxes, classIDs, confidences, frame):
    # ensure at least one detection exists
    if len(idxs) > 0:
        # loop over the indices we are keeping
        for i in idxs.flatten():
            # extract the bounding box coordinates
            (x, y) = (boxes[i][0], boxes[i][1])
            (w, h) = (boxes[i][2], boxes[i][3])

            # draw a bounding box rectangle and label on the frame
            color = [int(c) for c in COLORS[classIDs[i]]]
            cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)
            text = "{:} : {:.4f}".format(LABELS[classIDs[i]],
                                         confidences[i])
            cv2.putText(frame, text, (x, y - 5),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)
            # Draw a green dot in the middle of the box
            cv2.circle(frame, (x + (w//2), y + (h//2)), 2, (0, 0xFF, 0), thickness=2)

# PURPOSE: Initializing the video writer with the output video path and the same number
# of fps, width and height as the source video
# PARAMETERS: Width of the source video, Height of the source video, the video stream
# RETURN: The initialized video writer
def initializeVideoWriter(video_width, video_height, videoStream):
    # Getting the fps of the source video
    sourceVideofps = videoStream.get(cv2.CAP_PROP_FPS)
    # initialize our video writer
    fourcc = cv2.VideoWriter_fourcc(*"MJPG")
    return cv2.VideoWriter(outputVideoPath, fourcc, sourceVideofps,
                           (video_width, video_height), True)

# PURPOSE: Identifying if the current box was present in the previous frames
# PARAMETERS: All the vehicular detections of the previous frames,
#             the coordinates of the box of previous detections
# RETURN: True if the box was current box was present in the previous frames;
#        False if the box was not present in the previous frames
def boxInPreviousFrames(previous_frame_detections, current_box, current_detections):
    centerX, centerY, width, height = current_box
    dist = np.inf #Initializing the minimum distance
    # Iterating through all the k-dimensional trees
    for i in range(FRAMES_BEFORE_CURRENT):
        coordinate_list = list(previous_frame_detections[i].keys())
        if len(coordinate_list) == 0: # When there are no detections in the previous frame
            continue
        # Finding the distance to the closest point and the index

```



```

temp_dist, index = spatial.KDTree(coordinate_list).query([(centerX, centerY)])
if (temp_dist < dist):
    dist = temp_dist
    frame_num = i
    coord = coordinate_list[index[0]]

if (dist > (max(width, height)/2)):
    return False

# Keeping the vehicle ID constant
current_detections[(centerX, centerY)] = previous_frame_detections[frame_num][coord]
return True

def count_vehicles(idxs, boxes, classIDs, vehicle_count, previous_frame_detections, frame):
    current_detections = { }
    # ensure at least one detection exists
    if len(idxs) > 0:
        # loop over the indices we are keeping
        for i in idxs.flatten():
            # extract the bounding box coordinates
            (x, y) = (boxes[i][0], boxes[i][1])
            (w, h) = (boxes[i][2], boxes[i][3])

            centerX = x + (w//2)
            centerY = y + (h//2)

            # When the detection is in the list of vehicles, AND
            # it crosses the line AND
            # the ID of the detection is not present in the vehicles
            if (LABELS[classIDs[i]] in list_of_vehicles):
                current_detections[(centerX, centerY)] = vehicle_count
                if (not boxInPreviousFrames(previous_frame_detections, (centerX, centerY,
w, h), current_detections)):
                    vehicle_count += 1
                    # vehicle_crossed_line_flag += True
            # else: #ID assigning
            # Add the current detection mid-point of box to the list of detected items
            # Get the ID corresponding to the current detection

            ID = current_detections.get((centerX, centerY))
            # If there are two detections having the same ID due to being too close,
            # then assign a new ID to current detection.
            if (list(current_detections.values()).count(ID) > 1):
                current_detections[(centerX, centerY)] = vehicle_count
                vehicle_count += 1

            #Display the ID at the center of the box
            cv2.putText(frame, str(ID), (centerX, centerY),\
cv2.FONT_HERSHEY_SIMPLEX, 0.5, [0,0,255], 2)

    return vehicle_count, current_detections

# load our YOLO object detector trained on COCO dataset (80 classes)

```

```

# and determine only the *output* layer names that we need from YOLO
print("[INFO] loading YOLO from disk...")
net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)

#Using GPU if flag is passed
if USE_GPU:
    net.setPreferableBackend(cv2.dnn.DNN_BACKEND_CUDA)
    net.setPreferableTarget(cv2.dnn.DNN_TARGET_CUDA)

ln = net.getLayerNames()
ln = [ln[i - 1] for i in net.getUnconnectedOutLayers()]

# initialize the video stream, pointer to output video file, and
# frame dimensions
videoStream = cv2.VideoCapture(inputVideoPath)
video_width = int(videoStream.get(cv2.CAP_PROP_FRAME_WIDTH))
video_height = int(videoStream.get(cv2.CAP_PROP_FRAME_HEIGHT))

# Specifying coordinates for a default line
x1_line = 0
y1_line = video_height//2
x2_line = video_width
y2_line = video_height//2

#Initialization
previous_frame_detections = [{(0,0):0} for i in range(FRAMES_BEFORE_CURRENT)]
# previous_frame_detections = [spatial.KDTree([(0,0)])]*FRAMES_BEFORE_CURRENT #
Initializing all trees
num_frames, vehicle_count = 0, 0
writer = initializeVideoWriter(video_width, video_height, videoStream)
start_time = int(time.time())
# loop over frames from the video file stream
while True:
    print("=====NEW FRAME=====")
    num_frames+= 1
    print("FRAME:\t", num_frames)
    # Initialization for each iteration
    boxes, confidences, classIDs = [], [], []
    vehicle_crossed_line_flag = False

    #Calculating fps each second
    start_time, num_frames = displayFPS(start_time, num_frames)
    # read the next frame from the file
    (grabbed, frame) = videoStream.read()

    # if the frame was not grabbed, then we have reached the end of the stream
    if not grabbed:
        break

    # construct a blob from the input frame and then perform a forward
    # pass of the YOLO object detector, giving us our bounding boxes
    # and associated probabilities
    blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (inputWidth, inputHeight),

```

```

        swapRB=True, crop=False)
net.setInput(blob)
start = time.time()
layerOutputs = net.forward(ln)
end = time.time()

# loop over each of the layer outputs
for output in layerOutputs:
    # loop over each of the detections
    for i, detection in enumerate(output):
        # extract the class ID and confidence (i.e., probability)
        # of the current object detection
        scores = detection[5:]
        classID = np.argmax(scores)
        confidence = scores[classID]

        # filter out weak predictions by ensuring the detected
        # probability is greater than the minimum probability
        if confidence > preDefinedConfidence:
            # scale the bounding box coordinates back relative to
            # the size of the image, keeping in mind that YOLO
            # actually returns the center (x, y)-coordinates of
            # the bounding box followed by the boxes' width and
            # height
            box = detection[0:4] * np.array([video_width, video_height, video_width,
video_height])

            (centerX, centerY, width, height) = box.astype("int")

            # use the center (x, y)-coordinates to derive the top
            # and and left corner of the bounding box
            x = int(centerX - (width / 2))
            y = int(centerY - (height / 2))

            #Printing the info of the detection
            #print('\nName:\t', LABELS[classID],
            #      '\t\tBOX:\t', x,y)

            # update our list of bounding box coordinates,
            # confidences, and class IDs
            boxes.append([x, y, int(width), int(height)])
            confidences.append(float(confidence))
            classIDs.append(classID)

## Changing line color to green if a vehicle in the frame has crossed the line
# if vehicle_crossed_line_flag:
#     cv2.line(frame, (x1_line, y1_line), (x2_line, y2_line), (0, 0xFF, 0), 2)
## Changing line color to red if a vehicle in the frame has not crossed the line
# else:
#     cv2.line(frame, (x1_line, y1_line), (x2_line, y2_line), (0, 0, 0xFF), 2)

# apply non-maxima suppression to suppress weak, overlapping
# bounding boxes
idxs = cv2.dnn.NMSBoxes(boxes, confidences, preDefinedConfidence,

```

```

preDefinedThreshold)

# Draw detection box
drawDetectionBoxes(idxs, boxes, classIDs, confidences, frame)

vehicle_count, current_detections = count_vehicles(idxs, boxes, classIDs, vehicle_count,
previous_frame_detections, frame)

# Display Vehicle Count if a vehicle has passed the line
displayVehicleCount(frame, vehicle_count)

# write the output frame to disk
writer.write(frame)

cv2.imshow('Frame', frame)
if cv2.waitKey(1) & 0xFF == ord('q'):
    break

# Updating with the current frame detections
previous_frame_detections.pop(0) #Removing the first frame from the list
# previous_frame_detections.append(spatial.KDTree(current_detections))
previous_frame_detections.append(current_detections)

# release the file pointers
print("[INFO] cleaning up...")
writer.release()
videoStream.release()

```

# 8.TESTING

## 8.1 TESTING INTRODUCTION

Software testing can be stated as the process of verifying and validating whether a software or application is bug-free, meets the technical requirements as guided by its design and development, and meets the user requirements effectively and efficiently by handling all the exceptional and boundary cases.

The process of software testing aims not only at finding faults in the existing software but also at finding measures to improve the software in terms of efficiency, accuracy, and usability. It mainly aims at measuring the specification, functionality, and performance of a software program or application.

### Need for Testing

Testing was essential for the following reasons: -

Existence of program defects of inadequacies

- The software behavior as intended by its designer
- Conformance with requirement specification/user needs.
- Assess the operational reliability of the system.
- Reflect the frequency of actual user inputs.
- Find the fault, which caused the output anomaly.
- Checks for detect flaws and deficiencies in the requirements.
- Check whether the software is operationally useful.
- Exercise the program using data like the real data processed by the program.

### **8.1.1 Unit Testing**

Unit testing is a software testing method where individual units or components of a software application are tested in isolation from the rest of the system. A unit refers to the smallest testable part of an application, such as a function, method, or class. The purpose of unit testing is to validate that each unit of the software performs as expected and to identify and fix any defects or bugs early in the development process.

Unit tests are typically automated and run in a testing framework that allows developers to create, run, and analyse the results of tests. Unit tests should be independent, meaning that they should not rely on other units or external resources, and should be repeatable and predictable.

By performing unit testing, developers can ensure that each unit of their code is functioning as intended and that any issues are caught early in the development process, when they are easier and less expensive to fix. This approach can also improve the overall quality and reliability of the software, as well as make it easier to maintain and modify over time.

### **8.1.2 Integration Testing**

Integration testing is a software testing method that involves testing the interaction between different components or modules of an application to ensure that they work together as expected. This type of testing is performed after unit testing and before system testing.

Integration testing can be performed using different approaches, such as top-down, bottom-up, or a combination of both. In top-down integration testing, the higher-level modules are tested first, followed by the lower-level modules. In contrast, bottom-up integration testing tests the lower-level modules first, followed by the higher-level modules. In both approaches, the goal is to ensure that the integration between the modules is seamless and that the system as a whole function correctly.

By performing integration testing, developers can identify any issues that may arise when different components of the application are combined, and ensure that the system functions as intended. This can help reduce the risk of errors or bugs in the final product, improve the overall quality and reliability of the software, and ensure that the system meets the requirements and expectations of end-users.

### **8.1.3 User Interface Testing**

User Interface (UI) testing is a software testing method that focuses on testing the user interface or the front-end of a software application to ensure that it functions correctly and meets the requirements of end-users. The purpose of UI testing is to verify that the graphical user interface (GUI) elements such as buttons, menus, icons, and other visual components of the application work as intended, are displayed correctly, and are responsive to user input.



UI testing can be performed manually or with the help of automated testing tools. In manual UI testing, testers perform tests by using the software application and interacting with the user interface to verify that it behaves as expected. Automated UI testing involves using software tools to simulate user interactions and validate that the UI elements are displayed correctly, and that they respond to user input in the expected manner.

Some common UI testing scenarios include verifying that the application is responsive to different screen resolutions and orientations, checking that the user interface elements are aligned correctly, testing that the application is compatible with different browsers, and ensuring that the user interface is accessible to users with disabilities.



By performing UI testing, developers can ensure that the user interface of their software application functions correctly and meets the expectations of end-users. This can help improve the user experience, reduce the risk of errors or bugs, and increase the overall quality and reliability of the software application.

## 8.2 Test Results



### Test case-1:

Input	Expected output	Actual output	Result
	15 sec	Green signal time is 15 sec 	Pass

### Test case-2:



Input	Expected output	Actual output	Result
	6 sec	Green signal time is 6sec 	Pass

### Test case-3:

Input	Expected output	Actual output	Result
	17 sec	Green signal time is 17 sec 	Pass



**Test case-4:**

Input	Expected output	Actual output	Result
	21 sec	Green signal time is 21 	Pass

## 9.OUTPUT SCREENS

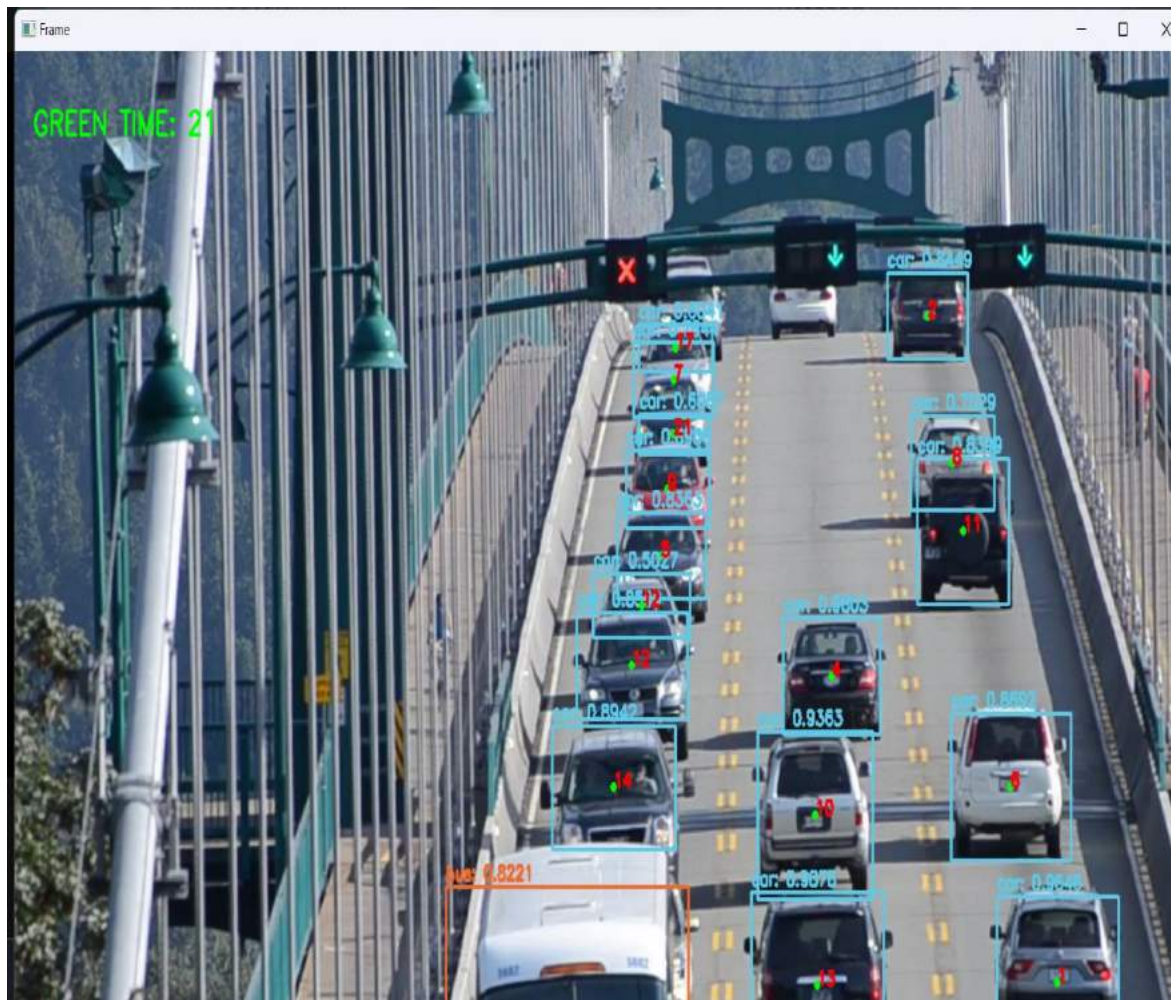
### Input video 1:



**Fig 7: Input video 1**

The input for this scenario is a video of traffic on a road. The purpose of the video is to analyze the traffic on the road in order to gain insights and make data-driven decisions related to traffic management. The video will be processed using machine learning techniques to detect and track vehicles, estimate their densities, and classify them by type. The information gathered from the video analysis will be used to identify traffic patterns, predict congestion, and develop strategies for improving traffic flow and reducing accidents. The analysis will provide valuable insights into the traffic behavior, helping to optimize traffic management and enhance overall traffic efficiency.

## Output 1:



**Fig 8: Output 1 after processing**

The image is a view of a road and it displays several vehicles, each represented by a box drawn around them. In addition to the boxes, the image also includes a vehicle count and the green signal time, which is

21 seconds.

The boxes around the vehicles indicate that computer vision techniques were used to identify and track each vehicle in the image. This information can be used to estimate vehicle speeds, identify traffic patterns, and predict potential collisions or congestion.

## **Input video 2:**

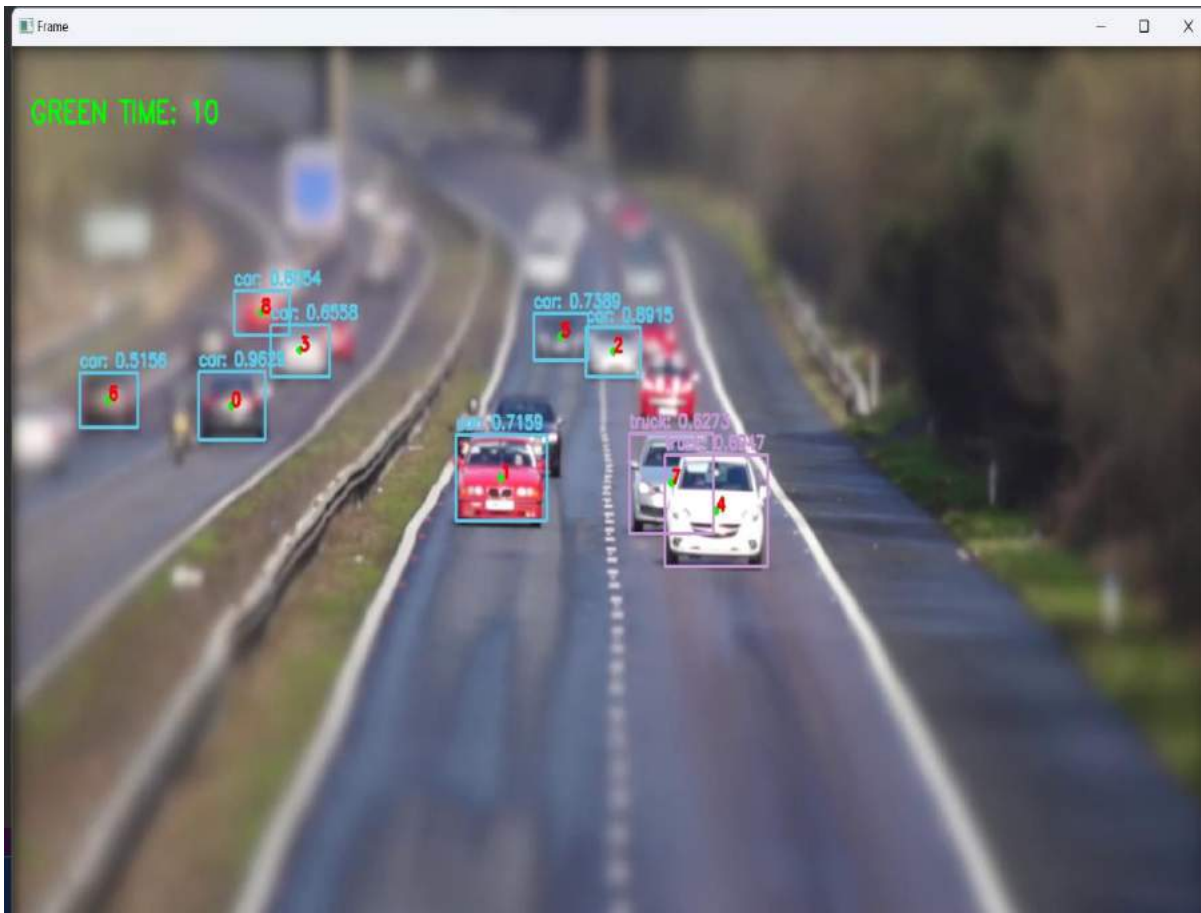


**Fig 9: Input video 2**

The input given is a video recording of traffic on a two-way road. The video captures vehicles moving in both directions on a road that is designed to accommodate traffic in two lanes, one for each direction. The video could include cars, motorcycles, bicycles, and other vehicles traveling at varying speeds. The purpose of recording such a video could be for traffic analysis, safety measures, or studying traffic behavior on two-way roads. The video could also be used for training machine learning algorithms to detect traffic patterns, estimate traffic volume, and predict traffic congestion on two-way roads. Overall, the video provides a detailed snapshot of the traffic situation on a particular two-way road at a specific time.



## Output 2:



**Fig 10: Output 2 after processing**

The image appears to depict a traffic scene with several vehicles in motion. The number of vehicles present in the scene has been indicated by a count displayed on the top left corner of the image. It seems that the current traffic signal is displaying a green light, with a timer set for 10 seconds. The green light implies that vehicles are allowed to move forward, and the timer likely indicates the time remaining for the green light signal.

The vehicles are represented in the image as boxes, each of them labeled with an identification number or symbol. This could be a useful feature for traffic management and surveillance systems to identify and track specific vehicles. Overall, the image seems to be a snapshot of a typical traffic scenario, with information about vehicle count and signal timing provided for further analysis and interpretation.

### **Input video 3:**

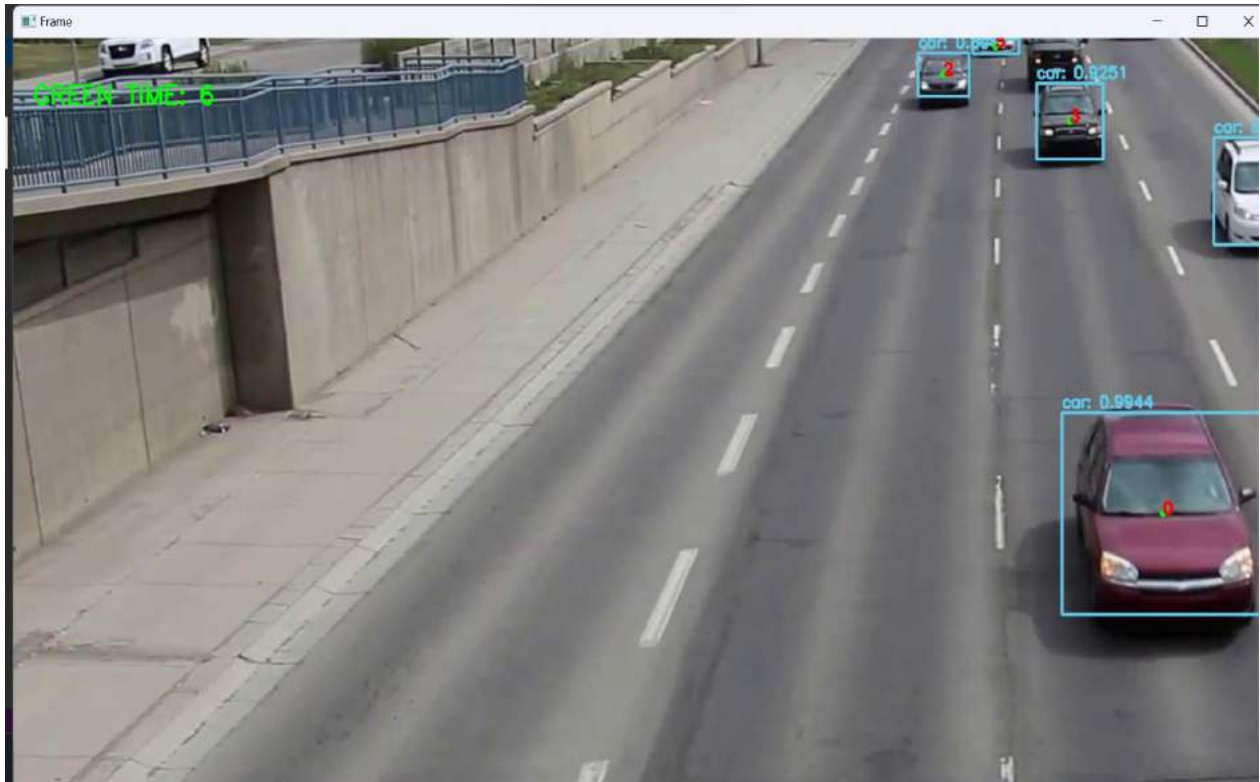


**Fig 11: Input video 3**

The video shows a bustling intersection on a multi-lane road, with heavy traffic moving in various directions. The intersection appears to be regulated by a traffic signal system, with different colored lights indicating when vehicles can proceed or stop. As the video progresses, vehicles can be seen approaching the intersection from different lanes, making turns, or moving straight ahead. The traffic flow is constant, with vehicles merging into different lanes, changing direction, or stopping for pedestrians at designated cross walks.

The video also captures the different types of vehicles present, ranging from cars and trucks to motorcycles and bicycles. The sound of honking horns and screeching brakes can be heard throughout, adding to the chaotic and busy atmosphere of the intersection. Overall, the video provides a glimpse into the complex and dynamic nature of traffic management in a highly populated area, requiring constant monitoring and regulation to ensure safety and efficient flow.

### Output 3:



**Fig 12: Output 3 after processing**

The image appears to depict a traffic scenario with several vehicles present, each represented by a box with an identification number or symbol. The total number of vehicles present has been indicated by a count displayed on the top left corner of the image. It seems that the current traffic signal is displaying a green light, with a timer set for 6 seconds. The green light indicates that vehicles are allowed to proceed, while the timer likely indicates the time remaining for the green signal.

Overall, the image provides a snapshot of a traffic situation, providing useful information about vehicle count and signal timing for analysis and interpretation. The image could be beneficial for traffic engineers or city planners in analyzing traffic flow and making decisions on traffic control strategies.

#### **Input video 4:**



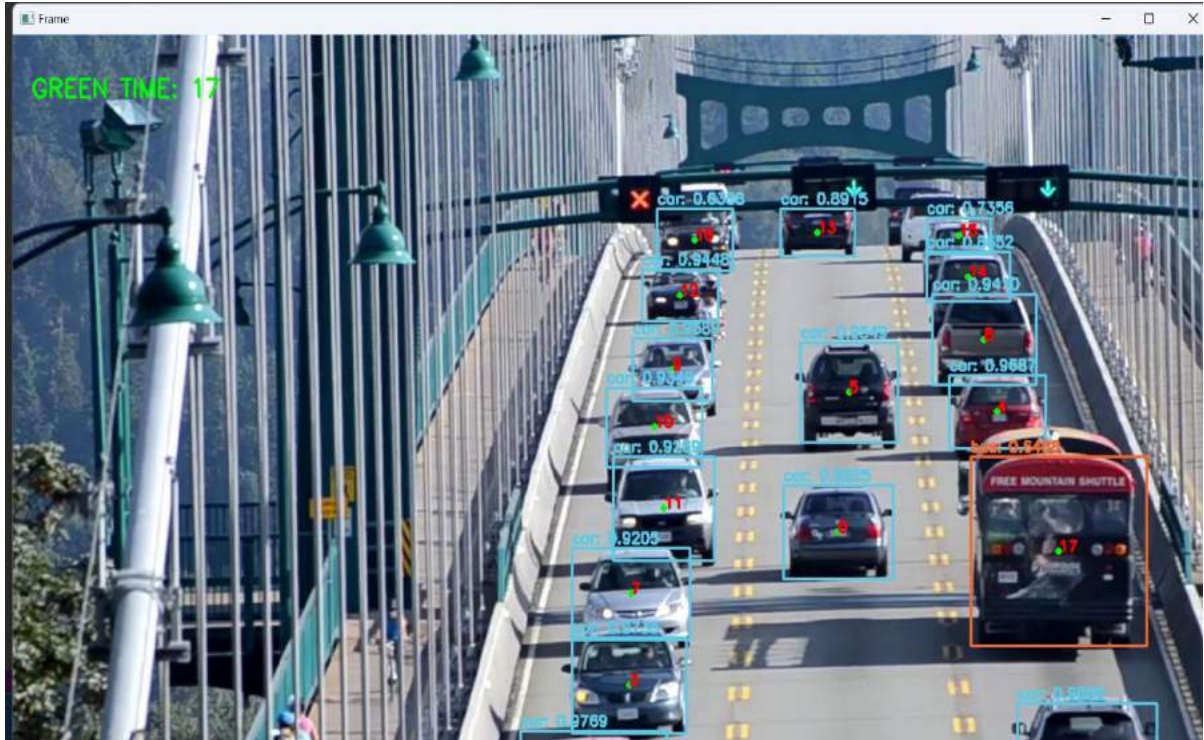
**Fig 13: Input video 4**

The video shows a three-lane traffic scenario on a bridge, with vehicles moving in both directions. The bridge appears to be a major thoroughfare, with heavy traffic flowing constantly. The traffic is regulated by a traffic signal system, with different colored lights indicating when vehicles can proceed or stop. Throughout the video, vehicles can be seen moving in different lanes, changing direction, and merging into other lanes.

The video also shows pedestrians walking along the sidewalk on either side of the bridge, with some crossing the road at designated crosswalks. The sound of honking horns and the occasional screeching of brakes can be heard in the background, highlighting the busy and dynamic nature of the traffic. Overall, the video offers a glimpse into the complexity of traffic management on a major bridge, requiring careful regulation and attention to ensure safety and efficient flow.



## Output4:



**Fig 14: Output 4 after processing**

The image appears to depict a traffic scenario with several vehicles present, each represented by a box with an identification number or symbol. The total number of vehicles present has been indicated by a count displayed on the top left corner of the image. It seems that the current traffic signal is displaying a green light, with a timer set for 17 seconds. The green light indicates that vehicles are allowed to proceed, while the timer likely indicates the time remaining for the green signal.

The vehicles depicted in the image include different types, such as cars, trucks, and motorcycles, with each of them identifiable through their unique box label.

Overall, the image provides a snapshot of a traffic situation, providing useful information about vehicle count and signal timing for analysis and interpretation. The image could be beneficial for traffic engineers or city planners in analyzing traffic flow and making decisions on traffic control strategies. The longer green signal time of 17 seconds could also suggest a lower traffic density or lighter traffic flow.

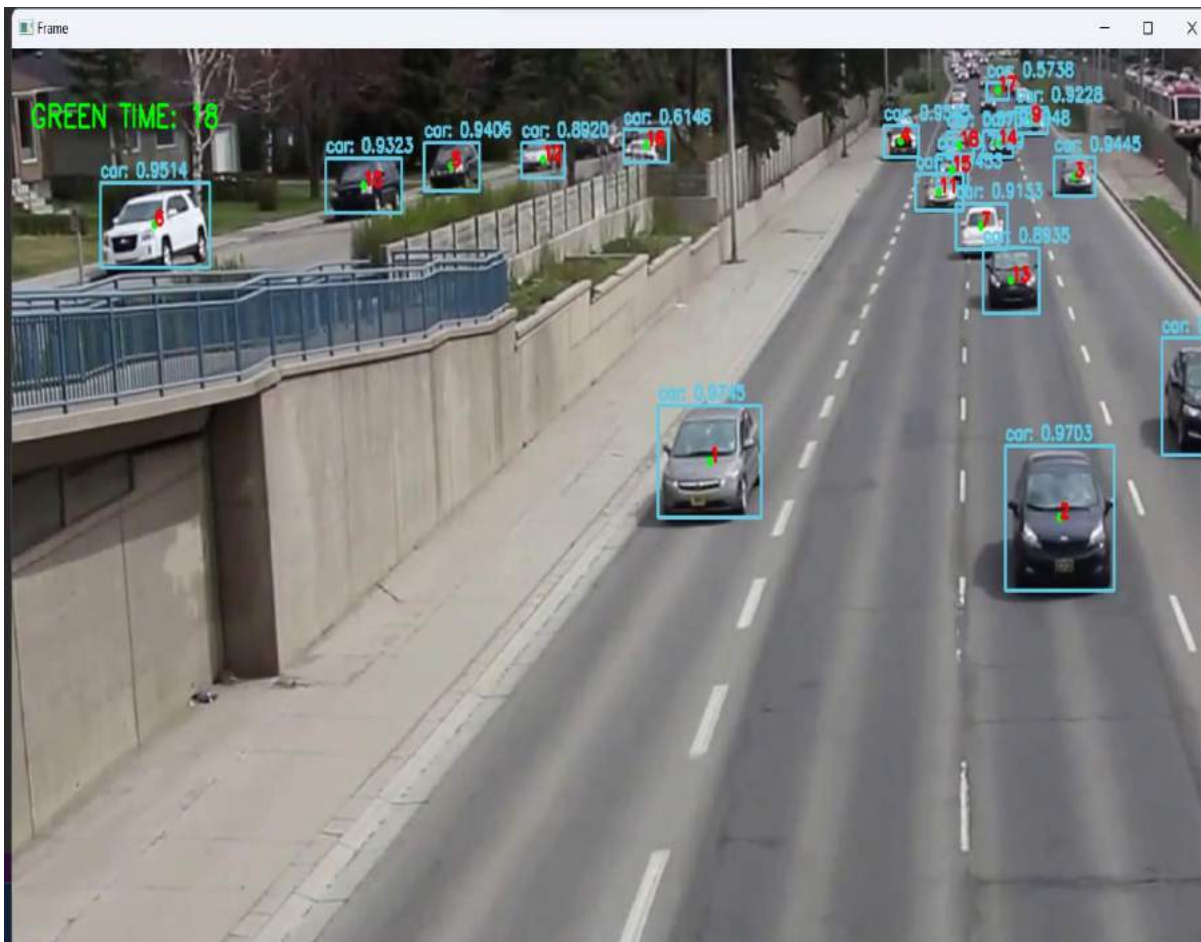
### **Input video 5:**



**Fig 15: Input video 5**

The video shows a single-lane road with traffic flowing in one direction. The road appears to be a local street, with relatively low traffic density and speed. Throughout the video, vehicles can be seen moving in a single file, with few instances of overtaking or merging. The sound of vehicles can be heard in the background, with the occasional honking horn. The absence of traffic in the opposite direction allows for a smoother flow of traffic and reduces the risk of accidents. The video provides a snapshot of a straightforward traffic situation, where careful attention and adherence to traffic rules are required to ensure safety on the road.

## Output 5:



**Fig 16: Output 5 after processing**

The image represents a traffic situation where vehicles are present and their counts are displayed. The green signal time of 18 seconds suggests that the intersection is busy and requires careful regulation. The vehicles are identified and represented in boxes, indicating the presence of an automated system that tracks and records vehicle movement. This system can help monitor traffic flow and ensure that vehicles adhere to traffic regulations.

The image provides valuable data for traffic management, helping authorities to optimize traffic flow, reduce congestion, and minimize the risk of accidents. Overall, the image highlights the importance of effective traffic management and the role of technology in regulating and monitoring traffic on busy roads.



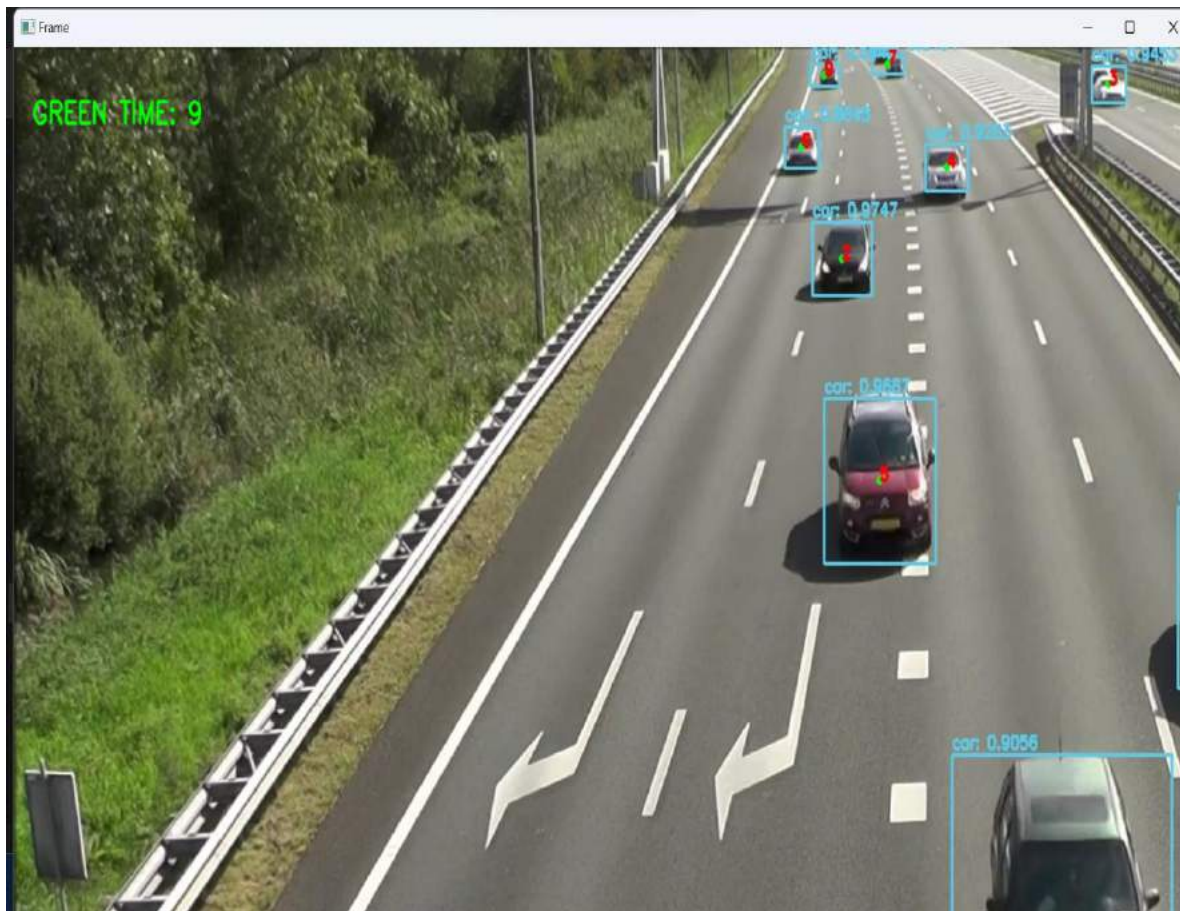
### **Input video 6:**



**Fig 17: Input video 6**

The video depicts traffic on a road where vehicles are changing direction. The road appears to be a multi-lane highway or major thoroughfare, with high traffic density and speed. Throughout the video, vehicles can be seen changing lanes, turning left or right at intersections. This creates a complex traffic situation that requires careful attention and adherence to traffic rules. The video provides a snapshot of a challenging traffic situation, where effective traffic management and driver awareness are critical to ensuring safety on the road. Drivers must be vigilant, patient, and alert to changes in traffic flow, and follow traffic regulations to prevent accidents and maintain a smooth flow of traffic.

## Output 6:



**Fig 18: Output 6 after processing**

The image shows a traffic scene where vehicles are present, and a vehicle counting system is in place. The green signal time is set to 9 seconds, which determines the time allowed for vehicles to move through the intersection. This means that vehicles must move quickly and efficiently to clear the intersection during the green light phase.

The image also shows the presence of vehicle identification boxes, which are used to track and monitor traffic flow. These boxes are used to identify different types of vehicles, including cars, buses, trucks, and motorcycles, and count the number of vehicles passing through the intersection.

Overall, this image captures a traffic management system in action, where the green signal time and vehicle counting system are key elements in ensuring the smooth flow of traffic.

## 9.1 Comparison of YOLO V3 with Deep learning models:

S.no	Model	True detection rate	False detection rate
1.	Fast-RCNN (pre-trained)	90%	0.7%
2.	Faster-RCNN (pre-trained)	92%	0.6%
3.	Mask-RCNN (pre-trained)	92%	0.5%
4.	YOLO (pre-trained)	92%	0.4%
5,	YOLO V3(pre-trained)	95%	0.3%

**Table 1: Comparison of Deep learning models.**

The table presents the performance evaluation of five popular object detection models: Fast-RCNN, Faster-RCNN, Mask-RCNN, YOLO, and YOLO V3. The models were evaluated on their true detection rate (TDR) and false detection rate (FDR) metric.

The results show that all the models performed relatively well, with TDR ranging from 90% to 95%. However, there were some variations in the FDR, with the best model having an FDR of 0.3% and the worst model having an FDR of 0.7%.

Overall, the YOLO V3 model performed the best, with the highest TDR and the lowest FDR. This indicates that it can detect most of the objects accurately and has a low false detection rate. On the other hand, the Fast-RCNN model had the lowest TDR and the highest FDR, indicating that it may not be the best model for object detection.

It is important to note that these models were pre-trained, meaning that they were trained on a large dataset and can be fine-tuned to improve their performance on specific tasks. Therefore, the results may vary depending on the dataset and task at hand.

## 9.2 Comparison of YOLO V3 with existing algorithm while predicting the traffic density:

Classifier	Accuracy (%)	Sensitivity (%)	Specificity (%)	ACU
Navie Bayesian	76.68	56.90	88.70	0.815
SVM	82.16	66.10	91.90	0.790
Logistic Regression	82.16	69.80	89.90	0.860
C4.5 decision tree	78.15	68.50	84.60	0.770
Random Forest	84.30	71.40	92.20	0.904

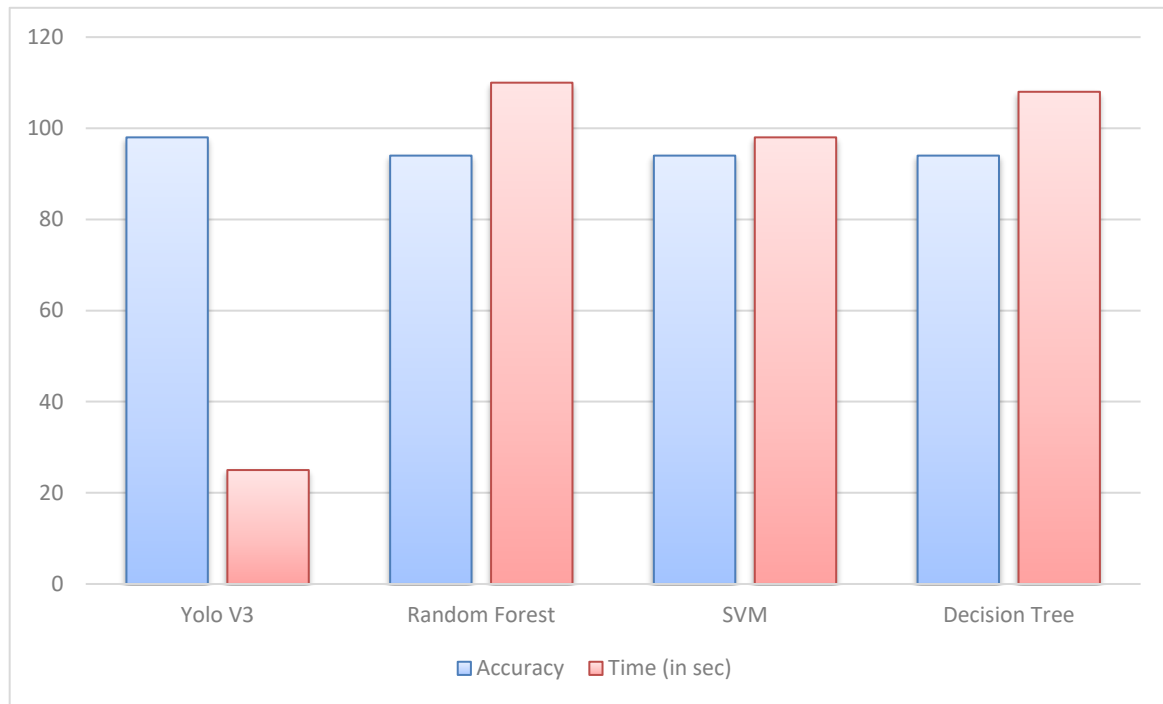
**Table 2: Comparison of YOLO V3 with other models.**

The table presents the performance evaluation of five popular machine learning classifiers: Navie Bayesian, SVM, Logistic Regression, C4.5 decision tree, and Random Forest. The models were evaluated on four metrics: accuracy, sensitivity, specificity, and area under the curve (AUC).

The results show that all the models performed relatively well, with accuracy ranging from 76.68% to 84.30%. However, there were some variations in the sensitivity and specificity, with the best model having a sensitivity of 71.40% and a specificity of 92.20%, and the worst model having a sensitivity of 56.90% and a specificity of 88.70%. The Logistic Regression and Random Forest models performed the best, with the highest accuracy and AUC scores. This indicates that they can classify the data accurately and have a good discriminative ability. On the other hand, the Navie Bayesian and C4.5 decision tree models had lower accuracy and AUC scores, indicating that they may not be the best models for classification.

It is important to note that the performance of these models may vary depending on the dataset and task at hand. Therefore, it is recommended to evaluate multiple models and choose the one that performs best on the specific task.

### 9.3 A Chart representation for Analysis of Multiple Machine Learning Models:



**Fig 19: chart for analysis of Multiple Machine Learning Algorithms**

This chart compares the accuracy and speed of four different algorithms: Yolo V3, Random Forest, SVM, and Decision Tree. The X-axis shows the names of the algorithms, and the Y-axis shows the time it takes for each algorithm to complete, measured in seconds. The bars or data points on the chart represent the accuracy of each algorithm.

By looking at this chart, you can easily compare the performance of each algorithm in terms of both accuracy and speed. For example, you can see which algorithm is the most accurate, but also takes the most time to complete. Or, you can identify which algorithm is the fastest, but sacrifices some accuracy compared to the others. This chart can be useful for making informed decisions about which algorithm to use for a particular task, balancing the trade-offs between accuracy and speed.



## 10.CONCLUSION

This new system facilitates the movement of cars in intersections, resulting in reducing congestion, less CO2 emissions, etc. The richness that video data provides highlights the importance of advancing the state-of-the-art in object detection, classification and tracking for real-time applications. There has been a steady progression of image detection techniques beginning with feature descriptors like HOG and, more recently, deep network-based approaches like Faster R-CNN and YOLO. YOLO provides extremely fast inference speed with slight compromise in accuracy, especially at lower resolutions and with smaller objects.

While real-time inference is possible, applications that utilize edge devices still require improvements in either the architecture's design or edge device's hardware. Finally, we have proposed new algorithm taking this real-time data from YOLO and optimizing phases in order to reduce cars waiting time.

# 11.REFERENCES

- [1] Pornpanomchai, C., Liamsanguan, T., & Vannakosit, V., Vehicle detection and counting from a video frame. In Wavelet Analysis and Pattern Recognition, ICWAPR'08. International Conference on, vol. 1, pp. 356-361, 2008
- [2] Xia, Y., Shi, X., Song, G., Geng, Q., & Liu, Y, towards improving quality of video-based vehicle counting method for traffic flow estimation. Signal Processing, vol. 120, pp. 672-681,2016.
- [3] Barcellos, P., Bouvié, C., Escouto, F. L., & Scharcanski, J., A novel video-based system for detecting and counting vehicles at user-defined virtual loops. Expert Systems with Applications, vol. 42 no. 4, pp. 1845- 1856, 2015.
- [4] Bouvie, C., Scharcanski, J., Barcellos, P., & Escouto, F. L, Tracking andcounting vehicles intraffic video sequences using particle filtering. In Instrumentation and Measurement Technology Conference (I2MTC), 2013 IEEE International, pp. 812-815, 2013.
- [5] Salvi, G., An automated nighttime vehicle counting and detection system for traffic surveillance. In Computational Science and Computational Intelligence (CSCI), 2014 International Conference on, vol. 1, pp. 131-136, 2014.
- [6] Bhaskar, P. K., & Yong, S. P., Image processing-based vehicle detectionand tracking method. In Computer and Information Sciences (ICCOINS), 2014 International Conference on, pp. 1-5, 2014.
- [7] Chen, T. H., Lin, Y. F., & Chen, T. Y., Intelligent vehicle counting method based on blob analysis in traffic surveillance. In Innovative Computing, Information and Control, 2007. ICICIC'07. Second International Conference on, pp. 238-238, 2007.
- [8] Quesada, J., & Rodriguez, P., Automatic vehicle counting method based on principal component pursuit background modeling. In Image Processing (ICIP), 2016 IEEE International Conference on, pp. 3822- 3826, 2016.
- [9] Jang, H., Won, I. S., & Jeong, D. S., Automatic Vehicle Detection and Counting Algorithm. International Journal of Computer Science andNetwork Security (IJCSNS), vol. 14, no. 9, pp. 99, 2014.
- [10] Moranduzzo, T., & Melgani, F., Automatic car counting method for unmanned aerial

- vehicle images. *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 3, pp. 1635-1647, 2014.
- [11] Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on* (Vol. 1, pp. I-I). IEEE.
  - [12] Dalal, N., & Triggs, B. (2005, June). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on* (Vol. 1, pp. 886- 893). IEEE.
  - [13] Felzenszwalb, P. F., Girshick, R. B., & McAllester, D., Cascade object detection with deformable part models. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pp. 2241-2248, 2010.
  - [14] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91-99).
  - [15] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 779-788).
  - [16] Danelljan, M., Häger, G., Khan, F., & Felsberg, M. (2014). Accurate scale estimation for robust visual tracking. In *British Machine Vision Conference, Nottingham, September 1-5, 2014*. BMVA Press.
  - [17] Yang, Y., & Bilodeau, G. A. (2016). Multiple Object Tracking with Kernelized Correlation Filters in Urban Mixed Traffic. *arXiv preprint arXiv:1611.02364*.
  - [18] Van Pham, H., & Lee, B. R., Front-view car detection and counting with occlusion in dense traffic flow. *International Journal of Control, Automation and Systems*, vol. 13, no. 5, pp. 1150-1160, 2015.
  - [19] Wolf, C., & Jolion, J. M., Object count/area graphs for the evaluation of object detection and segmentation algorithms. *International Journal of Document Analysis and Recognition (IJDAR)*, vol. 8, no. 4, pp. 280- 296, 2006.



## COPY RIGHT



ELSEVIER  
SSRN

**2023 IJIEMR.** Personal use of this material is permitted. Permission from IJIEMR must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. No Reprint should be done to this paper, all copy right is authenticated to Paper Authors

IJIEMR Transactions, online available on 31<sup>st</sup> Mar 2023. Link

[:http://www.ijiemr.org/downloads.php?vol=Volume-12&issue=Issue 03](http://www.ijiemr.org/downloads.php?vol=Volume-12&issue=Issue 03)

**10.48047/IJIEMR/V12/ISSUE 03/112**

Title **Automated Traffic Light with Yolo V3 and Machine learning**

Volume 12, ISSUE 03, Pages: 805-812

Paper Authors

**P.Vishnu Vardhan Reddy, E.Venkatesh, SK.Mahammad Rasool, Y.Kanuka Lourdhu Reshma**



USE THIS BARCODE TO ACCESS YOUR ONLINE PAPER

To Secure Your Paper As Per **UGC Guidelines** We Are Providing A Electronic Bar Code

## Automated Traffic Light with Yolo V3 and Machine learning

P.Vishnu Vardhan Reddy, E.Venkatesh, SK.Mahammad Rasool, Y.Kanuka Lourdhu Reshma

UG Students, Dept of CSE, Kallam Haranadhareddy Institute of Technology, Ap, India.

### ABSTRACT

One of the major problems in India is traffic congestion, which is especially common in the country's major cities. For example, congested roadways serve as a stark reminder of the lodge's awfulness. Since streets are frequently free for the taking, there is little financial incentive for cars to use them responsibly, leading to traffic jams whenever demand exceeds the ability to pay. Both street estimating and the privatization of interstates have been suggested as possible solutions to reduce traffic through financial disincentives and rewards. Blockage can also happen as a result of one-time parkway events, like a mishap or road construction, which may reduce the street's capacity below normal levels. While congestion is a possibility for all modes of transportation, the majority of the systems focused on vehicle obstruction on open streets. Techniques for image analysis have been widely used in traffic framework management and control. This paper suggests an alternative methodology, an algorithm, that would help distribute traffic fairly while controlling the signal by utilizing HERE maps API in order to eliminate the excess and impracticality of these image preparation frameworks.

**Keywords**—Machine learning, traffic signal algorithm, traffic management, and congestion.

### 1. INTRODUCTION

India takes pride in being the second-largest social organization in the world. The staggering 5.4 million km still exist in the Indian street networks! As a result, it forms a massive final recommendation for the Indian Government to provide impeccable streets at each stage. Passing through the Indian alleys is undoubtedly a problem that no Indian, whether traditional or modern, might want to experience. Extreme traffic congestion develops as demand approaches a street's limit (or the intersections along the street).

A traffic jam or (informally) a traffic growl up occurs when all moving vehicles come to a full stop for an extended period of time. Drivers who encounter traffic obstructions may become perplexed and engage in street anger. In terms of numbers, a clog is typically thought of as the number of cars that pass through a particular location during a specific period of time or a stream. Some of the common transportation problems include Poor Street quality due to excessive traffic - The outrageous congestion of urban streets due to heavily used

Private vehicles contribute to the degradation of the street environment. This frequently results in constant transportation problems.

Air pollution, particularly in urban areas, and noise other health-harming problems like air pollution and noise pollution are also brought on by the mere size of the traffic problems.

Therefore, we suggest a system that dynamically manages traffic based on a number of key variables, including the time of day, the weather, the state of the roads, etc. The system makes it possible to equally spread the area's traffic congestion.

### 2. LITERATURE SURVEY

The goal of the study is to create an image processing-based system for vehicle recognition and counting. The overall works consist of creating software for a system that needs to record a video



frame and stream. The backdrop road is devoid of any moving vehicles, and the frame is filled with moving vehicles. The device is made to distinguish between moving vehicles and determine the quantity of moving vehicles from a video frame. There are four main parts to the system for detecting and numbering vehicles: image acquisition, image processing, object detection and counting, and display result are the first three steps. The following characteristics have been accessed through the experiment: 1) Usability, to demonstrate that the system is capable of detecting and numbering vehicles under the predetermined conditions. 2) Effectiveness, to demonstrate the system's great degree of accuracy. For intelligent traffic management and supervision of the highway, vehicle detection and statistics in highway surveillance video scenes are extremely important. The widespread installation of traffic security cameras has resulted in the collection of a sizable database of video footage for analysis. A farther-off road surface can typically be taken into account from an elevated viewing angle. At this viewing angle, the vehicle's object size varies significantly, and a small object far from the road is less likely to be detected accurately. Effectively resolving the aforementioned issues and then applying them is crucial when dealing with intricate camera situations. The purpose of this article is to address the aforementioned problems and offer a workable solution. We also apply the vehicle detection findings to multi-object tracking and vehicle counting.

Vehicle counting is a crucial technique for estimating traffic flow, which is typically estimated to assess the condition of the traffic in traffic management. With the widespread installation of cameras in metropolitan transportation networks, surveillance footage develops into a significant data source for conducting vehicle counting. The complexity of traffic situations, however, has a significant impact on the effectiveness and accuracy of vehicle counting. In this study, we use the virtual loop technique to enhance the accuracy of the video-based vehicle counting approach. As an example, to enhance the quality of segmenting moving cars, the expectation-maximization (EM) algorithm is combined with the Gaussian mixture model (GMM).

In order to get a better object area, a restoration technique is also intended to remove noise and fill in any holes. The final step is to address occlusion problems using a morphological feature and the color-histogram. The proposed approach can enhance the results of vehicle segmentation and vehicle occlusion detection, according to effectiveness and efficiency experiments. It is also possible to significantly increase and achieve 98% accuracy in vehicle counting. Highlights Better car segmentation is suggested using a Gaussian mixture model based on EM. The vehicle area in the difference image is intended to be restored by a restoration technique. Occlusion is detected by combining a morphological trait with a color histogram. In an effort to advance the current traffic systems, intelligent traffic flow analysis (TFA) systems use sensing and data processing methods. In order to actively improve vehicles and traffic flow, they incorporate additional hardware, such as sensors, digital traffic signs, and cameras, and use cutting-edge processing methods to process the data the hardware provides. For instance, effectively improving the traffic flow and decreasing the delay by changing the timing and phasing of the lights in a traffic light system. The TFA system uses a video stream that includes an image sequence to perform a video-based vehicle counting. This is crucial because the only feasible technology for counting, feature extraction, pattern recognition, projection, and multi-scale signal analysis is digital image processing. To count vehicles more accurately, the system also needs to know their precise position. In essence, the video is split into frames, which are then changed to colored or grayscale frames, and the frames are then provided to the system as inputs. To conduct vehicle counting, the system then uses various kinds of preprocessing, object detection, identification, and tracking algorithms. The tracking focuses on a specific area of an image known as the Region of Interest (ROI), which is thought to be crucial for the gathering of data. Once the system has collected the data, it will make sure not to process or count the same object again. For the same object location, vehicle tracking entails a comprehensive measurement procedure in numerous defined frames.

For the purpose of simulating the operation of inductive loops, a camera-based scheme is suggested for the detection of vehicles at user-defined virtual loops. Combining effective edge recognition with color information reduces false vehicle detections. The experimental results imply that the proposed method may be capable of accurately detecting and counting vehicles at user-defined virtual loops (with an average correct detection rate of more than 98%) while also being more resistant to cast shadows and abrupt changes in illumination than comparable methods that currently represent the state of the art.

Traffic monitoring systems, which are built on a variety of sensors technologies, are used to collect crucial data for traffic management. Systems dependent on GPS, for instance, can be utilized.

Traffic monitoring systems, which are built on a variety of sensors technologies, are used to collect crucial data for traffic management. For instance, a vehicle's position can be predicted using GPS-based systems. Traffic monitoring systems, which are built on a variety of sensors technologies, are used to collect crucial data for traffic management. For instance, GPS-based systems can be utilized to forecast the position of a vehicle.

It is possible to combine several sensors, including cameras and lasers, to produce approximations of a vehicle's localization that are even more precise. Detecting registration plates and classifying vehicles are two additional applications for camera-based systems. Additionally, information provided by statistics on the number of vehicles, their speeds, and the occupancy of streets can help with municipal planning. These days, inductive loop sensors or magnetic sensors are frequently used by traffic monitoring devices to collect traffic data.

However, video camera-based traffic monitoring systems have an edge over other kinds of sensors. In contrast to other sensor technologies, camera-based systems can be used for traffic monitoring and for detecting vehicles at user-defined virtual loops, which can provide more details about the vehicular traffic (such as the kinds of vehicles in circulation, vehicle size, and speed). This study introduces a novel technique for counting and tracking moving objects in video traffic patterns. The suggested technique groups particles in videos using image processing, particle filtering, and motion coherence to create convex shapes that are then examined for possible vehicles. In order to combine or separate the groupings, this analysis takes into account the convex shape of the items and contextual information.

Using the similarity of color histograms on windows centered at the particle locations, a vehicle is recognized and tracked after that. To synchronize traffic lights, assist users in choosing the best routes, and assist governments in planning the growth of the traffic system, information about traffic conditions can be used (e.g., building new roads). For traffic management, information on the number of vehicles, speed, and track occupancy is crucial. Typically, inductive loops, ultrasonic sensors, and microwave are used to acquire such information. Video provides much more information than the methods listed above, including the potential for using already-installed cameras, so computer vision has recently been the subject of extensive research. To enhance traffic control and management, a robust and trustworthy traffic surveillance system is essential. The detection of vehicle movement seems to be a crucial component of the surveillance system. The traffic flow aids in management and control, particularly when there is a traffic jam, by displaying the traffic status at regular intervals. This study describes an efficient traffic surveillance system for locating and monitoring moving vehicles in different low-light conditions. Four stages make up the proposed algorithm: headlight segmentation and detection, headlight pairing, vehicle tracking, vehicle counting, and vehicle detection. To quickly extract bright objects of interest, a segmentation method based on an adaptive threshold is first used.



The findings of the experiments demonstrate that the suggested system can deliver timely and beneficial information for traffic surveillance. The traffic surveillance system has received a lot of attention in recent years because it can deliver important and practical data like traffic flow density, queue length, average traffic speed, and total vehicle in a set time period. The traffic surveillance device typically needs more sensors. Push buttons (used to detect pedestrian demand), loop detectors (used to detect the presence of a vehicle at a specific location), magnetic sensors (magnetometers), radar sensors, microwave detectors, and cameras are some of the prevalent traffic sensors. A video camera is a promising traffic sensor because of its low cost and potential to gather a lot of data (including the number of vehicles, their speed and acceleration, class of vehicles, and their tracks), as well as the ability to infer higher-level data (incidents, speeding, origin-destination of vehicles, etc.).

A computer that handles image/video processing, object recognition, and object tracking is linked to the video cameras (CCD or CMOS). The past few decades have seen a plethora of research projects focused on measuring traffic performance using stationary rectilinear cameras to identify and track vehicles. If they can be made to be sufficiently dependable and robust, vision-based systems are generally acknowledged to be flexible and versatile in traffic monitoring applications. Traffic flow rate, average traffic speed, queue duration, and traffic density can all be used to evaluate traffic conditions, which is the main objective of a traffic surveillance system. When it comes to traffic surveillance systems, where efficient traffic management and safety are the primary concerns, vehicle detection and tracking is effective and important. Detecting vehicle and traffic statistics from video frames is a topic we cover in this paper.

There is still room for improvement in this field, despite the extensive study and numerous methods that have been used. To make changes, it is suggested to create a special algorithm for vehicle data tracking and recognition using blob detection techniques and the Gaussian mixture model. By studying the background, we can first tell the foreground from the background in frames. Here, the object is found using the foreground detector, and the area surrounding each found object is defined using a binary calculation. Some morphological operations have been used to accurately identify the moving object and remove the noise. The ultimate tally is then calculated by following the regions and objects that were discovered. The findings are promising; using the Gaussian Mixture Model and Blob Detection methods, we achieved average detection and tracking accuracy of over 91%. This paper introduces a traffic surveillance-based intelligent vehicle counting technique. The three stages of the proposed algorithm are as follows: Moving object segmentation, fragment analysis, and tracking are the three primary stages of processing. Through the use of blob analysis, a car is modelled as a rectangular patch.

The important characteristics are taken out by analyzing the blob of vehicles. Monitoring the minimal distance between two temporal images and comparing the extracted features allows one to track moving objects. Additionally, by examining blobs of vehicles, it is possible to determine each vehicle's velocity as well as the flow of vehicles through a designated region. The outcomes of the experiments demonstrate the capability of the suggested system to deliver timely and beneficial information for traffic monitoring.

There are various techniques for numbering vehicles, including headlight detection and particle filtering. Despite being the state-of-the-art for modelling video backgrounds, Principal Component Pursuit (PCP) hasn't been used for this job yet. This is primarily due to the fact that the majority of the PCP algorithms currently in use are batch techniques with large computational costs, making them unsuitable for real-time vehicle counting. In this study, we suggest using an innovative incremental

PCP-based algorithm to determine the number of vehicles in top-view traffic video sequences in real-time.

We put our method to the test on a number of difficult datasets, and the results compare favorably to state-of-the-art techniques in terms of both performance and speed: an average accuracy of 98% when counting vehicles passing through a virtual door, a 91% estimate of the total number of vehicles in the scene, and processing speeds of up to 26 frames per second.

### 3. PROBLEM STATEMENT

This project aims to reduce traffic congestion and unwanted long time delays and provides a better approach to this by calculating the density of the traffic at each part of the road and simultaneously provides the best solution in order to reduce the congestion and in some cases to stop giving a green light indication to some roads where there are less number of vehicles.

### 4. EXISTING SYSTEM

Currently, traffic lights operate on a fixed cycle with predetermined time delays between each signal change. However, this system can lead to significant traffic congestion, particularly in areas with high volumes of vehicles. This is because the traffic signal sequence may allocate a green light to a road section with minimal traffic, while a more congested area remains at a standstill.

### 5. PROPOSED SYSTEM

The proposed project presents a density-based traffic control system using reinforced learning to solve this problem. We bring in a slight change to the traffic signal system by making it priority based when there is a huge amount of traffic and then switching it back to the normal sequence after there is less amount of traffic. The system counts the number of vehicles on each part of the road and after the analysis the system takes an appropriate decision as to how much time is to be given the highest priority and the longest delay for the corresponding traffic light.

This algorithm is used to count, detect, and track the different types of vehicles. It determines the vehicle count earlier and suggests alternative routes to the vehicles.

The object detection algorithm operates in every frame. Finally counting the entire vehicle. If vehicle count is less than the threshold it is normal traffic signal switching otherwise the vehicle count is more suggest alternative routes to reduce the time spent.

The complete block diagram representation:

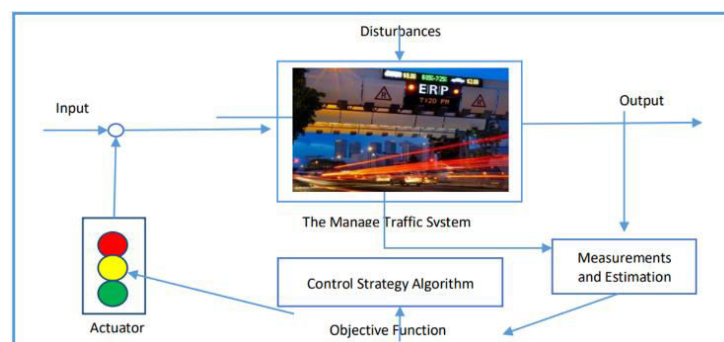


Figure 1: Overall block diagram of proposed system

For each of the vehicles identified in the frame, the model gives the parameters for the location, class, and confidence score. We first locate ROIs in the area for vehicle tracking. We begin tracking a vehicle's position when it is first detected within a ROI and continue to do so until it departs the area. We update the tracking positions of any additional vehicle detections within a ROI that are already being monitored from earlier frames. The events processor module receives all of the results from the preceding module and analyses them to identify important events. Vehicle

enumeration is one of the event processing module's subtasks.

There is a virtual counting zone within the ROI where the vehicles are counted according to class classification. Other subtasks, like traffic density measurement, accident detection, speed tracking, or traffic flow prediction, can be added to this module to help with traffic management.

## 6. METHODOLOGIES

The majority of image processing techniques apply to video processing, as well as techniques that take advantage of the temporal structure of video data. The objective of the video processing module is to analyze video images from traffic monitoring cameras in order to identify vehicles and then extract information about them, such as their position, class, and direction of travel. It has been used in this study to record videos from on-highway traffic surveillance cameras. The video processing module is then fed the footage to be processed. The first stage is to approach the video frames as images and apply image processing methods to them.

We used OpenCV, an open-source computer vision library created specifically for processing images and video, to read the movie frame by frame. Vehicle detection, Vehicle classification, and Vehicle tracking are the three tasks that make up video processing in this study.

As traffic surveillance applications for vehicle detection rely on quick and precise vehicle detection capabilities, the vehicle detection and classification duties are the most crucial in this framework. As was previously mentioned in the literature analysis, CNN has demonstrated significant improvements in object detection and classification in recent years. In our framework, we used the cutting-edge, real-time object detection system YOLOv3 for this job.

For feature extraction, YOLOv3 employs Darknet-53, a brand-new 53-layer CNN. We required some weights for the feature extraction layers in order to execute the object detection model (convolutional layers). We made use of the weights from a pre-trained model that had been optimized on the COCO dataset [LMB+14] with 80 object classes after being trained on the ImageNet dataset with 1000 object classes.

Although YOLO is very good at classifying and detecting items, it is designed to do so in a single image. The same processes for object detection, classification, and probability scores are repeated for each frame. Its ability to rapidly repeat these steps over a large number of frames gives it an edge over comparable object detection models. The COCO dataset-trained YOLOv3 can analyse 30 frames per second.

We have looked at works in the literature that could perform multi-object tracking because there are multiple vehicles in an image that we need to monitor. The estimation of an unknown number of objects in a film, as well as their individual paths, are both necessary for multi-object tracking.

In order for the traffic surveillance/control system to run securely and dependably, it must automatically detect any events, produce useful data about traffic status, and create or handle alarms. Such a module is crucial to the system's ability to observe and manage events. The vehicle counting sub-module was implemented in this effort, but additional modules may be added in the future.

The centroid of each vehicle's bounding area has been computed and tracked. A vehicle's status is changed from counted = False to tracking = True when it reaches the tracking zone. The subsequent frames track its position after that. Its state is updated to counted = True if it enters the counting zone. a message stating that the car was tallied. Counting of vehicles is done based on type. The vehicle ceases being tracked once it exits the counting zone, and we update its status to tracking = False.

## 7. RESULTS

YOLOv3 is a distinct model created for object recognition using each framework. Bounding boxes are used by the neural network to extract the features and identify the item. The model determines the class of the vehicle from a car, bike, truck, etc. after object recognition. The dynamic traffic signal timer algorithm receives the entire number of vehicles in a lane as input.

The algorithm determines the relativity between the lanes while taking into account the density of every other lane. The green signal timer for a specific lane is determined by the classification of the lane as low, middle, or high vehicle density.

S.No	Number of vehicles	Green Signal time
1	15	17 sec
2	17	21 sec
3	5	6 sec
4	20	30 sec
5	23	35 sec
6	30	43 sec

## 8. CONCLUSION

This new method makes it easier for cars to move through intersections, which reduces traffic and CO2 emissions, among other benefits. Because video data is so rich, it is crucial to keep up with technological advancements in object detection, classification, and tracking for real-time apps. The development of picture detection methods has been steady, starting with feature descriptors like HOG and ending with deep network-based methods like Faster R-CNN and YOLO. YOLO offers incredibly quick inference speed with a minimal accuracy loss, particularly for lower resolutions and smaller object sizes. Even though real-time inference is feasible, apps that use edge devices still need hardware upgrades for either the architecture or the edge devices themselves. Finally, by using this real-time data from YOLO and sequentially optimizing phases, we have suggested a new algorithm.

## REFERENCES

- [1] Pornpanomchai, C., Liamsanguan, T., & Vannakosit, V., Vehicle detection and counting from a video frame. In Wavelet Analysis and Pattern Recognition, ICWAPR'08. International Conference on, vol. 1, pp. 356-361, 2008.
- [2] Xia, Y., Shi, X., Song, G., Geng, Q., & Liu, Y., towards improving quality of video-based vehicle counting method for traffic flow estimation. Signal Processing, vol. 120, pp. 672-681, 2016.



- [3] Barcellos, P., Bouvié, C., Escouto, F. L., & Scharcanski, J., A novel video-based system for detecting and counting vehicles at user-defined virtual loops. *Expert Systems with Applications*, vol. 42 no. 4, pp. 1845-1856, 2015.
- [4] Bouvie, C., Scharcanski, J., Barcellos, P., & Escouto, F. L., Tracking and counting vehicles in traffic video sequences using particle filtering. In *Instrumentation and Measurement Technology Conference (I2MTC)*, 2013 IEEE International, pp. 812-815, 2013.
- [5] Salvi, G., An automated nighttime vehicle counting and detection system for traffic surveillance. In *Computational Science and Computational Intelligence (CSCI)*, 2014 International Conference on, vol. 1, pp. 131-136, 2014.
- [6] Bhaskar, P. K., & Yong, S. P., Image processing-based vehicle detection and tracking method. In *Computer and Information Sciences (ICCOINS)*, 2014 International Conference on, pp. 1-5, 2014.
- [7] Chen, T. H., Lin, Y. F., & Chen, T. Y., Intelligent vehicle counting method based on blob analysis in traffic surveillance. In *Innovative Computing, Information and Control, 2007. ICICIC'07. Second International Conference on*, pp. 238-238, 2007.
- [8] Quesada, J., & Rodriguez, P., Automatic vehicle counting method based on principal component pursuit background modeling. In *Image Processing (ICIP)*, 2016 IEEE International Conference on, pp. 3822-3826, 2016.
- [9] Jang, H., Won, I. S., & Jeong, D. S., Automatic Vehicle Detection and Counting Algorithm. *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 14, no. 9, pp. 99, 2014.
- [10] Moranduzzo, T., & Melgani, F., Automatic car counting method for unmanned aerial vehicle images. *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 3, pp. 1635-1647, 2014.
- [11] Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on* (Vol. 1, pp. I-I). IEEE. Dalal, N., & Triggs, B. (2005, June). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on* (Vol. 1, pp. 886-893). IEEE.