

# Intelligent Traffic System for Urban Condition using Real-time Vehicle Tracking

Dr. S. V. Ramana<sup>1</sup>, G. Saaketh<sup>2</sup>, D. Sohith Sai Neelesh<sup>3</sup>, G. G. S. S Ramanjaneyulu<sup>4</sup>, A.N.V.Subbaraju<sup>5</sup>

Professor, Department Of Information Technology<sup>1</sup>

Students, Department Of Information Technology<sup>2,3,4,5</sup>

S.R.K.R Engineering College, Bhimavaram, Andhra Pradesh, India

**Abstract:** Congestion in traffic is a major problem these days. Despite the fact that it appears to pervade all over, urban cities are the ones most influenced by it. And it's ever increasing nature makes it imperative to know the road traffic density in real time for better signal control and effective traffic management. There can be different causes of congestion in traffic like insufficient capacity, unrestrained demand, large Red Light delays etc. While insufficient capacity and unrestrained demand are somewhere interrelated, the delay of respective light is hard coded and not dependent on traffic. Therefore, the need for simulating and optimizing traffic control to better accommodate this increasing demand arises. In recent years, image processing and surveillance systems have been widely used in traffic management for traveller's information, ramp metering and updates in real time. The traffic density estimation can also be achieved using Image Processing. This project presents the method to use live images feed from the cameras at traffic junctions for real time traffic density calculation using image processing. It also focuses on the algorithm for switching the traffic lights according to vehicle density on road, thereby aiming at reducing the traffic congestion on roads which will help lower the number of accidents. In turn it will provide safe transit to people and reduce fuel consumption and waiting time. It will also provide significant data which will help in future road planning and analysis. In further stages multiple traffic lights can be synchronized with each other with an aim of even less traffic congestion and free flow of traffic. The vehicles are detected by the system through images instead of using electronic sensors embedded in the pavement. A camera will be placed alongside the traffic light. It will capture image sequences. Image processing is a better technique to control the state change of the traffic light. It shows that it can decrease the traffic congestion and avoids the time being wasted by a green light on an empty road. It is also more reliable in estimating vehicle presence because it uses actual traffic images. It visualizes the practicality, so it functions much better than those systems that rely on the detection of the vehicles' metal content.

**Keywords:** Vehicle Detection, Deep learning, yolo Algorithm, Traffic Optimization

## I. INTRODUCTION

Over the last decade, the adoption and use of technologies like Mobility, Cloud and Social Platforms has made it possible for common, middle class users to use small, focused applications for making their life easier and comfortable. Whether it is simply paying your utility bills using mobile banking or getting that favourite movie ticket by just clicking couple of buttons, use of technology has really changed the way we live, play and work. Though we have been referring to Smart Cities and communities for some time now, let us look at how use of Information and data available to us can be used to really create some smart services, which in a true sense provide us with better living. We shall look at a key case, which impacts us almost daily: traffic management. Use of technology and real time analysis can actually lead to a smooth traffic management. The common reason for traffic congestion is due to poor traffic prioritization, where there are such situations some lane has less traffic than the other. Vehicular congestion is increasing at an exponential rate. Let us take the case study of Chandigarh, one of the Union Territories of India. Chandigarh has the largest number of vehicles per capita in India. According to Chandigarh Transport Undertaking, more than 45,000 vehicles were registered this year in Chandigarh making the total count of more than 8 lakhs vehicles on the

road. While the number of vehicles are increasing at a fast pace, the infrastructure in the city is not being able to match this growth. Traffic jams during rush hours are becoming a routine affair, especially in the internal sectors where long queues of vehicles can be seen stranded. Therefore, we have tried to address the problem with the help of our project wherein the focus would be to minimize the vehicular congestion. We have achieved this with the help of image processing that can be obtained from surveillance cameras and eventually to deploy a feedback mechanism in the working of the traffic lights where the density of the traffic would also be factored in the decision making process.

## II. PURPOSE

Road transport is one of the primitive modes of transport in many parts of the world today. The number of vehicles using the road is increasing exponentially every day. Due to this reason, traffic congestion in urban areas is becoming unavoidable these days. Inefficient management of traffic causes wastage of invaluable time, pollution, wastage of fuel, cost of transportation and stress to drivers, etc. Our research on density based traffic drivers, etc. Our research is control. So, it is very much necessary to design a system to avoid the above casualties thus preventing accidents, collisions, and traffic jams. Connecting Smart Traffic Management System of the city and using the power of analytics is a key to smooth traffic management. Using real time analytics of data from these sources and linking them to some trends, we can manage traffic flow much better.

## III. PROBLEM STATEMENT

In urban areas, the management of traffic poses a significant challenge, necessitating the implementation of intelligent traffic management systems. The problem lies in the complexities of urban traffic patterns, which often lead to congestion, increased travel times, and elevated risks of accidents. Traditional traffic management systems fall short in addressing these issues effectively. Therefore, this study focuses on developing an intelligent traffic management system that utilizes real-time image processing. The aim is to leverage advanced computer vision techniques to analyze live images from surveillance cameras strategically placed across urban road networks. By extracting relevant information, such as vehicle density, flow, and anomalies, in real-time, the proposed system seeks to dynamically adapt traffic signal timings and optimize overall traffic management, with the ultimate goal of enhancing efficiency, reducing congestion, and improving safety in urban transportation.

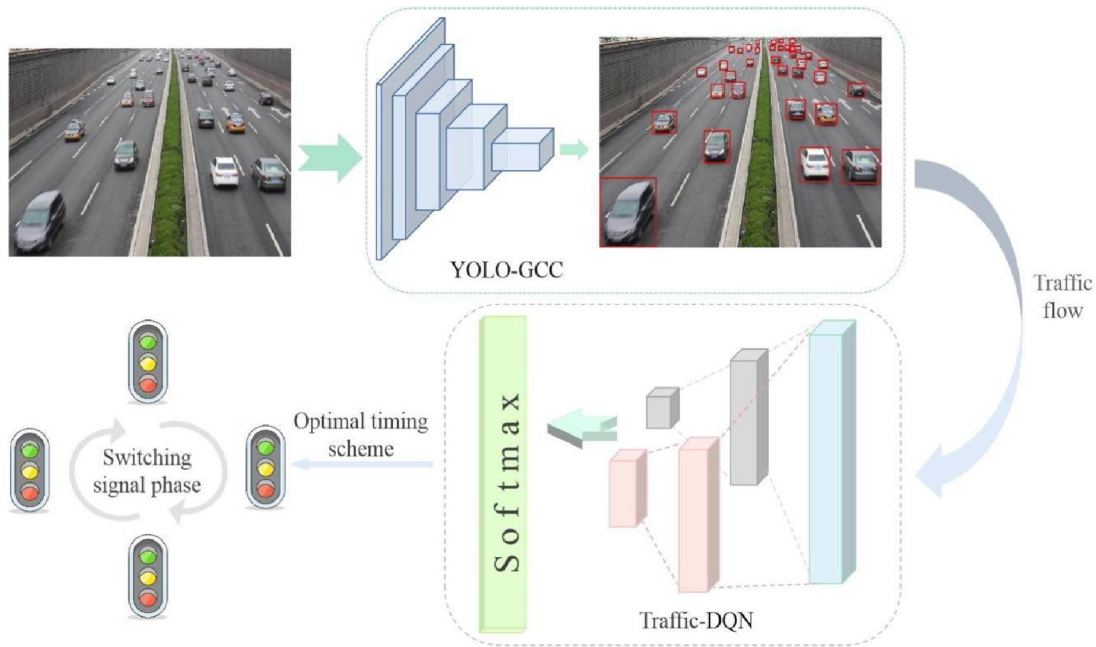
## IV. PROPOSED SYSTEM

It consists of pre-trained YOLO model algorithm to predict the traffic congestion of vehicles. 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. It requires only a single neural network to the full image. The input video sequence is given as input to convolutional neural network. The training process was implemented using convolutional neural network topology of the YOLO algorithm. A spatial detection of the object in a video-frame is necessary as a first input of most tracking algorithms, in our case, the object is segmented by using a Rectangular Region of Interest (ROI), in our implementation the frame rate of the videos was 45 FPS. Then the frames are given to YOLO model for counting, detecting and tracking purposes. 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 final step is to detect the wrong way vehicle. In our system, we defined that if the vehicle moves away from the camera, it will be detected as a wrong way vehicle. Suppose the vehicle is coming towards the camera and is in the right way. A wrong way vehicle after its detection, an image of the frame will be captured automatically. By using captured image further inception will be handled for wrong way vehicle.

## V. ARCHITECTURE

### Input Layer:

The input layer takes traffic images captured by cameras positioned across urban road networks. These images serve as the raw input for the YOLO system.



### Image Preprocessing:

Prior to feeding the images into the YOLO model, preprocessing steps may be applied, including resizing and normalization, to ensure uniformity and optimize model performance

### YOLO Model Backbone:

The YOLO architecture typically employs a deep neural network backbone, such as Darknet or YOLOv3, which consists of multiple convolutional layers. These layers are responsible for extracting hierarchical features from the input images.

### Object Detection Head:

The network's object detection head processes the features extracted by the backbone and generates bounding boxes, class probabilities, and confidence scores for each detected object. YOLO uses a grid system to efficiently detect objects at different scales within the image.

### Bounding Box Regression:

The bounding box regression layer refines the initially generated bounding boxes, improving the accuracy of object localization. This step is crucial for precise positioning of detected objects in the traffic scene.

### Non-Maximum Suppression (NMS):

Post-processing involves applying non-maximum suppression to filter out redundant boundingboxes and retain the most confident predictions. NMS ensures that only the most relevant and accurate detections are retained.

### Output Layer:

The output layer provides the final results, including the coordinates of bounding boxes, associated class labels (vehicle, pedestrian, etc.), and confidence scores indicating the certainty of the predictions.

**Integration with Traffic Management System:**

The processed results from YOLO are then integrated into the larger traffic management system. This integration enables real-time decision-making based on the detected objects, such as adjusting traffic signal timings, identifying congestion, or responding to emergencies

**VI. IMPLEMENTATION**

**Image Acquisition**

Generally an image is a two-dimensional function  $f(x,y)$  (here  $x$  and  $y$  are plane coordinates). The amplitude of image at any point say  $f$  is called intensity of the image. It is also called the grey level of image at that point. We need to convert these  $x$  and  $y$  values to finite discrete values to form a digital image. The input image is a fundus taken from stare data base and drive data base. The image of the retina is taken for processing and to check the condition of the person. We need to convert the analog image to digital image to process it through digital computer. Each digital image composed of a finite elements and each finite element is called a pixel.

**Formation of Image**

We have some conditions for forming an image  $f(x,y)$  as values of image are proportional to energy radiated by a physical source. So  $f(x,y)$  must be nonzero and finite.  
i.e.  $0 < f(x,y) < \infty$ .  
3. Image Pre-Processing

**Image Resizing:**

Formula:

`blob = cv2.dnn.blobFromImage(image, scalefactor=1.0/255, size=(416, 416), swapRB=True, crop=False)`

Explanation: Resizing the input image to a fixed size of 416x416 pixels. YOLO requires images to be in a specific format and size for detection.

**Normalization:**

Formula:

`blob = cv2.dnn.blobFromImage(image, scalefactor=1.0/255, size=(416, 416), swapRB=True, crop=False)`

Explanation: Dividing pixel values by 255 to scale them to the range [0, 1]. Normalization helps in faster convergence during training and better performance during inference.

**Color Channel Swapping:**

Formula:

`blob = cv2.dnn.blobFromImage(image, scalefactor=1.0/255, size=(416, 416), swapRB=True, crop=False)`

Explanation: Swapping the Red and Blue color channels. OpenCV loads images in BGR format by default, whereas YOLO expects images in RGB format. Swapping ensures the correct input format for YOLO.

**Non-maximum Suppression (NMS):**

Formula:

`idxs = cv2.dnn.NMSBoxes(boxes, confidences, args["confidence"], args["threshold"])`

Explanation: After detecting objects (vehicles in this case), NMS is applied to suppress multiple bounding box detections for the same object. It keeps only the most confident detection and removes overlapping boxes.

**Bounding Box Drawing:**

Formula: `cv2.rectangle(image, (x, y), (x + w, y + h), color, 2)`

**RGB to GRAY Conversion**

Humans perceive color through wavelength-sensitive sensory cells called cones. There are three different varieties of cones, each has a different sensitivity to electromagnetic radiation (light) of different wavelength. One cone is mainly

sensitive to green light, one to red light, and one to blue light. By emitting a restricted combination of these three colors (red, green and blue), and hence stimulate the three types of cones at will, we are able to generate almost any detectable color. This is the reason behind why color images are often stored as three separate image matrices; one storing the amount of red (R) in each pixel, one the amount of green (G) and one the amount of blue (B). We call such color images as stored in an RGB format. In grayscale images, however, we do not differentiate how much we emit of different colors, we emit the same amount in every channel. We will be able to differentiate the total amount of emitted light for each pixel; little light gives dark pixels and much light is perceived as bright pixels. When converting an RGB image to grayscale, we have to consider the RGB values for each pixel and make as output a single value reflecting the brightness of that pixel. One of the approaches is to take the average of the contribution from each channel:  $(R+B+C)/3$ . However, since the perceived brightness is often dominated by the green component, a different, more "human-oriented", method is to consider a weighted average, e.g.:  $0.3R + 0.59G + 0.11B$ .

### Image Enhancement

Image enhancement is the process of adjusting digital images so that the results are more suitable for display or further analysis. For example, we can eliminate noise, which will make it easier to identify the key characteristics. In poor contrast images, the adjacent characters merge during binarization. We have to reduce the spread of the characters before applying a threshold to the word image. Hence, we introduce "Power- Law Transformation" which increases the contrast of the characters and helps in better segmentation. The basic form of power-law transformation is

$$s = cr^\gamma,$$

where  $r$  and  $s$  are the input and output intensities, respectively;  $c$  and  $\gamma$  are positive constants. A variety of devices used for image capture, printing, and display respond according to a power law. By convention, the exponent in the power-law equation is referred to as gamma. Hence, the process used to correct these power-law response phenomena is called gamma correction. Gamma correction is important, if displaying an image accurately on a computer screen is of concern. In our experimentation,  $\gamma$  is varied in the range of 1 to 5. If  $c$  is not equal to '1', then the dynamic range of the pixel values will be significantly affected by scaling. Thus, to avoid another stage of rescaling after power-law transformation, we fix the value of  $c = 1$ . With  $\gamma = 1$ , if the power-law transformed image is passed through binarization, there will be no change in the result compared to simple binarization. When  $\gamma > 1$ , there will be a change in the histogram plot, since there is an increase of samples in the bins towards the gray value of zero. Gamma correction is important if displaying an image accurately on computer screen is of concern.

### Edge Detection

Edge detection is the name for a set of mathematical methods which aim at identifying points in a digital image at which the image brightness changes sharply or, more technically, has discontinuities or noise. The points at which image brightness alters sharply are typically organized into a set of curved line segments termed edges. Edge detection is a basic tool in image processing, machine vision and computer envisage, particularly in the areas of feature reveal and feature extraction.

Following are list of various edge-detection methods:-

Sobel Edge Detection Technique Perwitt Edge Detection

Roberts Edge Detection Technique Zerocross Threshold Edge Detection Technique Canny Edge Detection Technique

In our project we use "Canny Edge Detection Technique" because of its various advantages over other edge detection techniques.

### Canny Edge Detection

The Canny Edge Detector is one of the most commonly used image processing tools detecting edges in a very robust manner. It is a multi-step process, which can be implemented on the GPU as a sequence of filters.

Canny edge detection technique is based on three basic objectives.

**Low error rate:**

All edges should be found, and there should be no spurious responses. That is, the edges must be as close as possible to the true edges.

**Edge point should be well localized:**

The edges located must be as close as possible to the true edges. That is, the distance between a point marked as an edge by the detector and the center of the true edge should be minimum.

**Single edge point response:**

The detector should return only one point for each true edge point. That is, the number of local maxima around the true edge should be minimum. This means that the detector should not identify multiple edge pixels where only a single edge point exists.

The Canny edge detection algorithm consist of the following basic steps;

- Smooth the input image with Gaussian filter.
- Compute the gradient magnitude and angle images.
- Apply non-maxima suppression to the gradient magnitude image.
- Use double thresholding and connectivity analysis to detect and link edges.

**Image Matching**

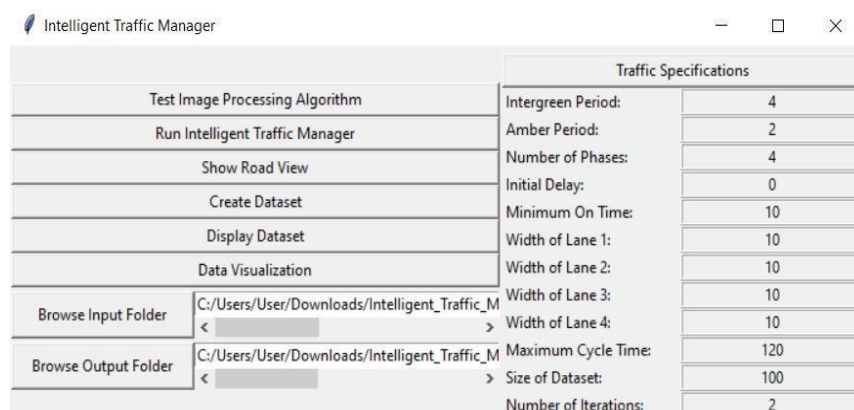
Recognition techniques based on matching represent each class by a prototype pattern vector. An unknown pattern is assigned to the class to which is closest in terms of predefined metric. The simplest approach is the minimum distance classifier, which, as its name implies, computes the (Euclidean) distance between the unknown and each of the prototype vectors. It chooses the smallest distance to make decision. There is another approach based on correlation, which can be formulated directly in terms of images and is quite intuitive. We have used a totally different approach for image matching.

Comparing a reference image with the real time image pixel by pixel. Though there are some disadvantages related to pixel based matching but it is one of the best techniques for the algorithm which is used in the project for decision making. Real image is stored in matrix in memory and the real time image is also converted in the desired matrix. For images to be same their pixel values in matrix must be same. This is the simplest fact used in pixel matching. If there is any mismatch in pixel value it adds on to the counter used to calculate number of pixel mismatches. Finally, percentage of matching is expressed as

**VII. SAMPLE OUTPUT**

**USER INTERFACE**

User interface is created using flask framework in python. It enables the user to select input files from the file system and upload in the project.



**COMPONENTS OF THE INTERFACE**





**Test Image Processing Algorithm:**

It will take random image from the input folder and run the image processing algorithm and find the number of vehicles detected in the image.



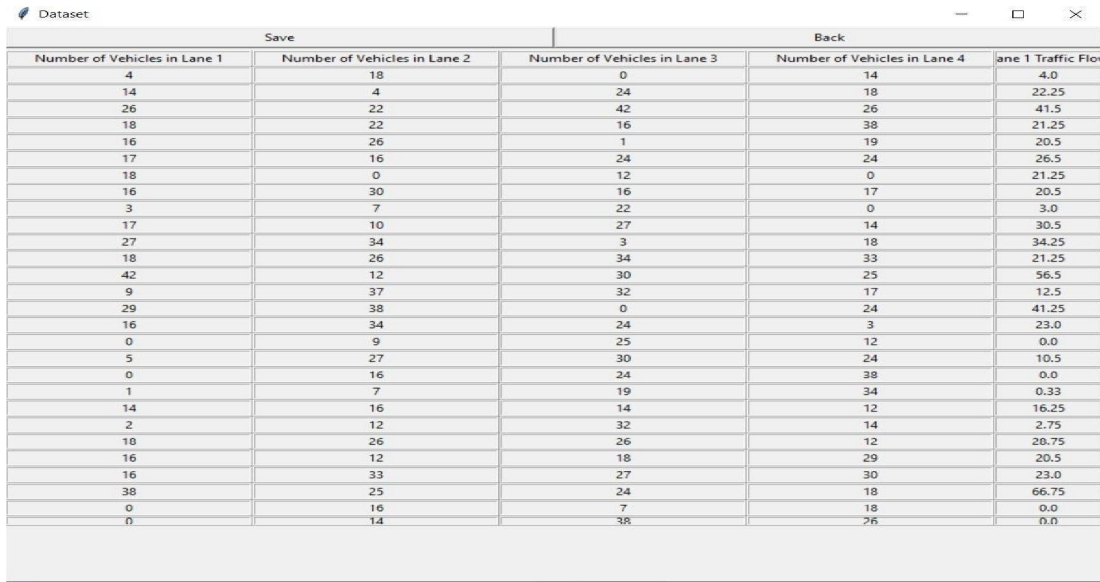
**Run Intelligent Traffic Manager:**

This will run the traffic light simulation based on the vehicle density obtained in each lane and allocate the green signal time based on vehicle density.

 <p>Width of Lane 1 = 10 Number of Vehicles = 18 Traffic Flow = 70.5 Saturation Flow = 180.0 On Time = 34</p>	 <p>Width of Lane 4 = 10 Number of Vehicles = 0 Traffic Flow = 0 Saturation Flow = 180.0 On Time = 10</p>
 <p>Width of Lane 2 = 10 Number of Vehicles = 0 Traffic Flow = 0</p>	 <p>Width of Lane 3 = 10 Number of Vehicles = 0 Traffic Flow = 0</p>

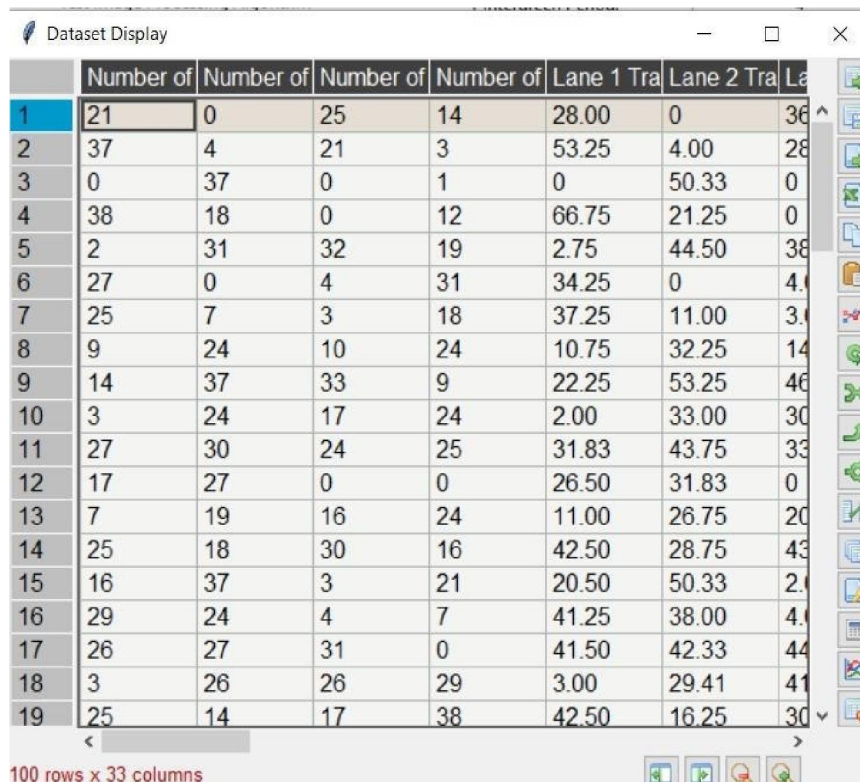
**Create Dataset:**

In this create dataset option a dataset is created to give as input to the algorithm for traffic analysis.



Number of Vehicles in Lane 1	Number of Vehicles in Lane 2	Number of Vehicles in Lane 3	Number of Vehicles in Lane 4	Lane 1 Traffic Flow
4	18	0	14	4.0
14	4	24	18	22.25
26	22	42	26	41.5
18	22	16	38	21.25
16	26	1	19	20.5
17	16	24	24	26.5
18	0	12	0	21.25
16	30	16	17	20.5
3	7	22	0	3.0
17	10	27	14	30.5
27	34	3	18	34.25
18	26	34	33	21.25
42	12	30	25	56.5
9	37	32	17	12.5
29	38	0	24	41.25
16	34	24	3	23.0
0	9	25	12	0.0
5	27	30	24	10.5
0	16	24	38	0.0
1	7	19	34	0.33
14	16	14	12	16.25
2	12	32	14	2.75
10	26	26	12	20.75
16	12	18	29	20.5
16	33	27	30	23.0
38	25	24	18	66.75
0	16	7	18	0.0
0	14	38	26	0.0

**Display Dataset:**



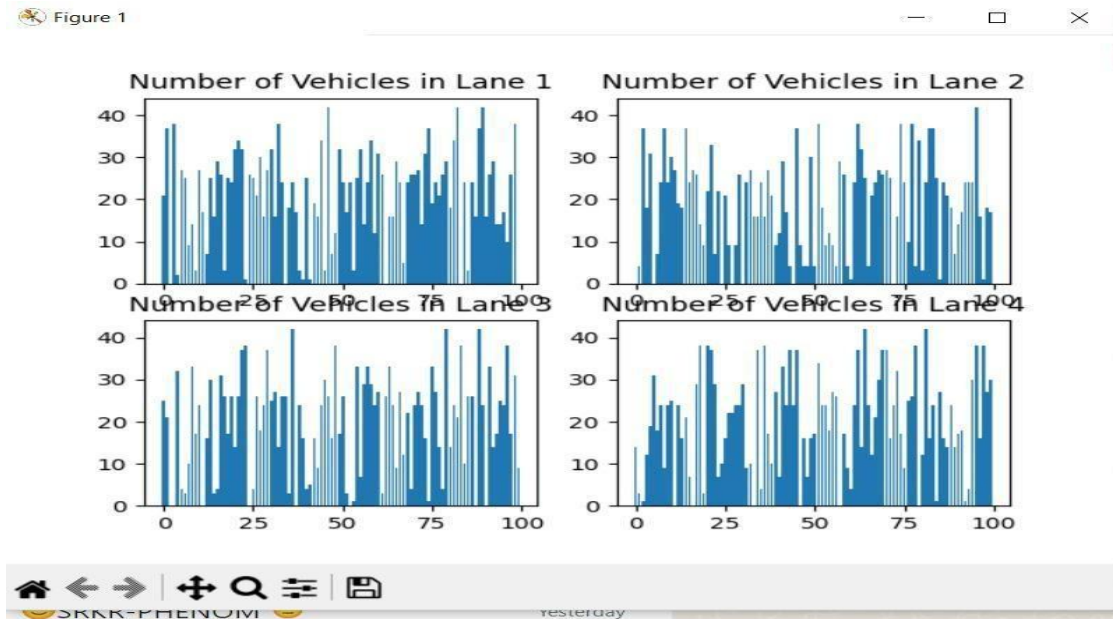
	Number of	Number of	Number of	Number of	Lane 1 Tra	Lane 2 Tra	Lane 3 Tra	Lane 4 Tra
1	21	0	25	14	28.00	0	36	28
2	37	4	21	3	53.25	4.00	28	28
3	0	37	0	1	0	50.33	0	0
4	38	18	0	12	66.75	21.25	0	0
5	2	31	32	19	2.75	44.50	38	38
6	27	0	4	31	34.25	0	4	4
7	25	7	3	18	37.25	11.00	3	3
8	9	24	10	24	10.75	32.25	14	14
9	14	37	33	9	22.25	53.25	46	46
10	3	24	17	24	2.00	33.00	30	30
11	27	30	24	25	31.83	43.75	33	33
12	17	27	0	0	26.50	31.83	0	0
13	7	19	16	24	11.00	26.75	20	20
14	25	18	30	16	42.50	28.75	43	43
15	16	37	3	21	20.50	50.33	2	2
16	29	24	4	7	41.25	38.00	4	4
17	26	27	31	0	41.50	42.33	44	44
18	3	26	26	29	3.00	29.41	41	41
19	25	14	17	38	42.50	16.25	30	30

100 rows x 33 columns

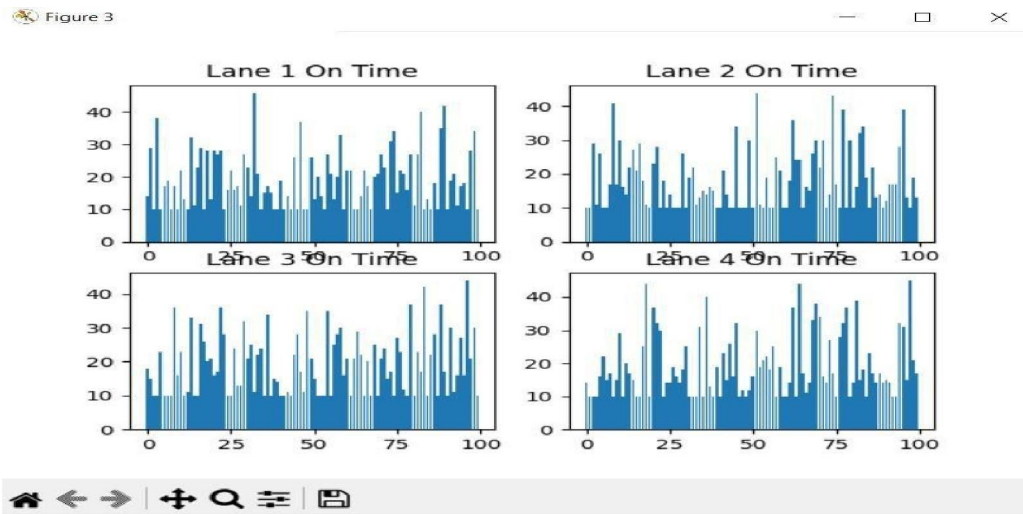


**Data Visualization:**

Number of Vehicles at each lane:



**Vehicles on time:**



**VIII. RESULTS**

**Traffic Density Analysis:** We can analyze the traffic density data obtained from the image processing module and discuss the distribution of traffic across different lanes. This analysis can include visualizations such as histograms or heatmaps to illustrate the variation in traffic density.

**Traffic Flow Management:** We can discuss the effectiveness of the traffic flow management system implemented in your code. This could involve comparing the actual traffic flow with the saturation flow and discussing any discrepancies.

**Traffic Light Optimization:** We can evaluate the performance of the traffic light optimization algorithm by examining the on-time duration of each traffic light phase

This analysis can include comparisons between the computed on-time durations and the minimum/maximum on-time limits.

Cycle Parameters: We can discuss the cycle parameters computed for the traffic signal control, such as the intergreen period, amber period, and total cycle time. This analysis can include the impact of these parameters on traffic flow efficiency and congestion management.

Dataset Analysis: If applicable, we can analyze any datasets collected during the simulation or testing phase of your traffic management system. This analysis can provide insights into traffic patterns, vehicle counts, and other relevant metrics.

## IX. CONCLUSION

- [1]. Improved Traffic Flow: By leveraging real-time data from cameras, sensors, and connected vehicles, a smart traffic management system can optimize traffic signal timings, lane assignments, and route planning to minimize congestion and delays. This leads to smoother traffic flow and reduced travel times for commuters.
- [2]. Enhanced Safety: With advanced object detection and monitoring capabilities, the system can detect and respond to potential safety hazards such as accidents, pedestrians crossing, or vehicles violating traffic rules. Timely alerts and interventions can help prevent accidents and improve overall road safety.
- [3]. Reduced Environmental Impact: By optimizing traffic flow and reducing idling time at intersections, a smart traffic management system can help reduce vehicle emissions and fuel consumption, contributing to lower air pollution and carbon footprint in urban areas.
- [4]. Dynamic Adaptability: Unlike traditional traffic control systems, which operate on fixed schedules, a smart traffic management system can dynamically adjust signal timings and lane configurations based on real-time traffic conditions, events, and emergencies. This flexibility allows for more adaptive and responsive traffic control strategies.
- [5]. Data-Driven Decision Making: The system generates large volumes of data on traffic patterns, vehicle movements, and environmental conditions. This data can be analyzed to identify trends, predict future traffic patterns, and optimize infrastructure investments and transportation policies.
- [6]. Integration with Emerging Technologies: A smart traffic management system can seamlessly integrate with emerging technologies such as autonomous vehicles, connected infrastructure, and mobility-as-a-service (MaaS) platforms. This integration fosters a more holistic approach to urban mobility and transportation planning.
- [7]. Improved Public Transit Integration: By coordinating traffic signal timings with public transit schedules and prioritizing buses or trams at intersections, the system can encourage greater use of public transportation and reduce reliance on private vehicles, leading to more sustainable urban mobility solutions.
- [8]. Scalability and Modularity: Smart traffic management systems are designed to be scalable and modular, allowing for easy deployment and expansion across different urban areas and transportation networks. They can also integrate with existing infrastructure and legacy systems, minimizing disruption and cost.
- [9]. Public Engagement and Transparency: By providing real-time traffic information, alerts, and updates to commuters through mobile apps or digital signage, the system promotes public engagement and awareness of traffic conditions. This transparency enhances public trust and confidence in urban mobility management.
- [10]. Continuous Innovation and Improvement: As technology evolves and new data sources become available, smart traffic management systems can continuously evolve and improve. Machine learning algorithms can be used to optimize traffic control strategies and adapt to changing urban dynamics over time.

## REFERENCES

- [1] <http://opencv.org/about.html>
- [2] [https://thingspeak.com/pages/learn\\_more](https://thingspeak.com/pages/learn_more)
- [3] <https://www.slideshare.net/louiseantonio58/image-processing-based-intelligent-traffic-control-systemmatlab-gui>
- [4] Khekare, G.S.; Sakhare, A.V., "A smart city framework for intelligent traffic system using VANET," Automation, Computing, Communication, Control and Compressed Sensing (iMac4s), 2013

- [5] Badura, S.; Lieskovsky, A., "Intelligent Traffic System: Cooperation of MANET and Image Processing," Integrated Intelligent Computing (ICIIC), 2010 First International Conference on, vol., no., pp.119,123, 5-7 Aug. 2010
- [6] [http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny\\_detector/canny\\_detector.htm](http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.htm) l
- [7] Salama, A.S.; Saleh, B.K.; Eassa, M.M., "Intelligent cross road traffic management system (ICRTMS)," Computer Technology and Development (ICCTD), 2010 2nd International Conference on, vol., no., pp.27,31, 2-4 Nov. 2010
- [8] Haimeng Zhao; Xifeng Zheng; Weiya Liu, "Intelligent Traffic Control System Based on DSP and Nios II," Informatics in Control, Automation and Robotics, 2009. CAR '09. International Asia Conference on, vol., no., pp.90,94, 1-2 Feb. 2009