International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 09 Issue: 10 | Oct 2022 www.irjet.net p-ISSN: 2395-0072

Control of Traffic Signals by AI based Image Processing

Mareddy Pragnya Bharathi[1], P. Venkat Narayana[2]

Dept. of Electrical and Electronics Engineering, JNTU College of Engineering, Hyderabad, India

---

***Abstract*— The blockage of traffic in urban cities is one of the major issues in the cities with the population growth and increase in automobiles. In the present-day Traffic system, wastage of fuel consumption in non-peak hours, as the ongoing traffic signal gives green light for longer duration, even there are few vehicles on the road, and in some cases, the user must wait as for signal to turn green while there are no other vehicles in the junction's other lanes, which adds to the driver's delay, increases fuel consumption, and raises transportation expenses, and increase air pollution. Regulating traffic flow in peak hours for vehicles to pass is the crucial factor. The ongoing traffic management systems that are in place are generally static, means that they do not adjust according to needs of the traffic flow. The suggested solution aims to provide a computer vision-based traffic controller that can adjust to the flow of traffic. By identifying the passing vehicles at the signal and adjusting green signal period accordingly, it employs a live feed from CCTV footage at traffic intersections to evaluate the traffic density in real-time. The automobiles are categorized as cars, motorbike, bus, truck, or tricycle to determine the green signal time with greater accuracy. For detecting the objects, techniques like YOLO algorithm are used and also counts number of automobiles for each direction. The timers of these traffic signals are set according to density of vehicle in each lane and hence the system becomes adaptive. This helps for clearing the traffic and optimizing the signal as green at a faster rate compared to present system, thus reduces the unnecessary delays, blockage, and waiting time. This reduces consumption of fuel and pollution. Pygame is used for developing simulation.**

Keywords—Traffic management, Artificial Intelligence, Object detection with YOLO algorithm, Computer Vision.

I. INTRODUCTION

With an increase in traffic, many road networks in urban areas experience issues with road capacity and associated service levels. Numerous traffic-related problems arise at the intersection control systems as a result of constant signal timers. They continue with the exact phase sequence as well as its duration. The need for cutting-edge control solutions will be identified in the field of smart transportation systems as the need for road capacity rises.

From the analysis of Traffic in Hyderabad from "The Times of India" The problem, according to officials of the police department and Road Transport Authority (RTA), has been triggered by a huge jump in private vehicular population in the city during the pandemic.

An estimated 10 lakh new vehicles were registered within the city, including just 4.5 lakh vehicles in the Cyberabad limits alone, resulting in an explosion within the population of vehicles which together with the present ones are criss-crossing the town, as per RTA authorities

There are conventional techniques for control that are getting used currently:

1)*Controlling with static timers at traffic signals*: In these fixed timers are loaded into the program. Even when there's no traffic on the lane the green signal is going to be present supported the timer set within the program.

This causes delay for people waiting at junction, and also increases fuel consumption, transportation cost and increase pollution.

2)*Controlling with Manpower*: They are managed by a Traffic Police standing at junction controlling the vehicles by hand gestures.

This method requires lot of manpower, when traffic is heavy and increasing day by day.

3)*Detecting the Traffic with air-sensors*: These devices detect the presence of pollution in environment. If pollution reaches particular level, depending on the extent of pollution, traffic density is detected using a sensor.

Since the vehicles on opposite lane also produces pollution, this is drawback for calculating the traffic density.

4)*Sensors on the road*: Adding some loop detectors or proximity sensors to the road is another advanced technique. This sensor transmits data on the movement of traffic. Collected sensor data is used to manage various traffic signals.

This contains a drawback of precision and scope of high-quality information i.e., usually supported high-cost technologies; hence a tight budget will cut back on the number of facilities. Additionally, a lot of sensors are typically needed to provide complete coverage across a facility network due to the limited optimal range for the most sensors.

Proposed system: The suggested system in this paper is meant to detect the volume of traffic and timers are set for every lane based on the traffic density calculated, this density may easily be adapted to current state of the traffic. Here, calculating traffic density can be done by taking live video at the junction, and by identifying the number of automobiles for individual lane approaching the junction and adjusting the duration of the green light. To determine a precise estimation of a green traffic time, the automobiles are identified as being a car, motorbike, bus, truck, or tricycle. It uses YOLO, so as to identify the number of vehicles and so set the timer of particular light in step with vehicle amount within the appropriate direction.

This reduces unneeded delays, overcrowding, and waiting times, which successively reduces the amount of fuel used and pollutants produced. Additionally, traffic is moved much more quickly than it would be under a static system.

II. LITERATURE REVIEW

A traffic management system supported image processing utilizing MATLAB code that modifies the timing of the green, yellow, red light depending on the volume of traffic. One Arduino UNO is used to control the green and yellow lights, and the other is used to control the red light and this is often endless process.[6]

An adaptive stoplight controller supported symbolic rationale for enhancing traffic flow at a remote crossroads. Using the data gathered from traffic detectors, a set of fuzzy rules has been created (queue length, arrival flow, and exit flow), evaluates the time for the lane. Two components make up the suggested fuzzy system: one with the primary driveway (which will receive the next volume of vehicles), and one for the second driveway (with a lower volume of vehicles). Different scenarios with varied traffic demands are used to compare proposed controller to a set signal programmed in order to verify the performance of the proposed decision rules.[9]

Before being submitted to the databases, the live video feed is edited. A C++-based algorithms is then used to generate the results. The techniques of hard coding and dynamic coding are contrasted, and the dynamic approach demonstrated a 35% improvement for controlling the signal timers.[3]

PIC microcontroller that implements dynamic time slots with various levels and measures traffic volume using Infrared sensor. Additionally, conveyable controller is designed to resolve traffic density and set the timers for overcrowded roads.[4]

A camera installed with light accustomed captures the live road images. Then the acquired pictures are loaded into digital image processor to seek out the traffic density on the road then the traffic lights are managed. By avoiding time lost on empty roads, the suggested strategy makes optimal use of time and fuel resources.[5]

Using image and video processing, actual traffic volume may be calculated using live video stream from the webcams at traffic intersections. It also works upon that algorithm for adjusting traffic signals in accordance with traffic volume on the road, intending to lessen congestion on roads that can reduce the number of accidents and fuel consumption will be reduced over time. Additionally, it'll still offer important information that can support analysis and planning for upcoming road projects. To further reduce traffic snarls and improve traffic flow, moreover signal lights are frequently synced with one another.[6]

Photos are captured by the camera using a Microcontroller based system, which then processes the photographs in MATLAB. Vehicle density is determined after the images are converted to criterion images by reducing saturation and colors. Through the USB and preconfigured simulation packages, MATLAB and Arduino are coupled. The Arduino determines the length of each lane's green light based on traffic volume and density. But there is a problem with this approach. The vehicles frequently collide one another and it's difficult to induce correct number of vehicles on the road. Additionally, since diverse things were also turned to white and black, they interfered with detection and is not possible to distinguish between common things like poles, trees, and billboards with vehicles.[7]

Algorithm for vector machines together using image processing techniques. in real-time video, the method is used because images in short frames are recorded.

Prior to using SVM, OpenCV is used to complete the image processing and convert the images to grayscale. Using this technique, red light violations and traffic density are both detected.[8]

A traffic density- and image-processing-based adaptive light timing control. This method uses a high-resolution image sensor, MATLAB, an Arduino signal timing, and transmission based on UART principles. But, neither the permitted ambulances nor accidents at the intersection are prioritized by this system.[9]

III. SUGGESTIVE SYSTEM

1) Overview of Proposed System:

In Proposed system live video is taken at traffic intersection as a source for current traffic density calculation. Here, density calculation utilizes image processing with python object detection is carried out utilizing YOLO Algorithm. This image is given to the vehicle identification system, which employs YOLO, as seen in Fig. 1. The YOLO method counts the number of each sort of vehicle, including cars, bicycles, buses, and trucks, to evaluate the traffic density. This density, along with a few other variables, is used by the signal changing algorithm to fix the signal time as green for each lane. Accordingly, the signal as red are modified. In order to prevent a particular lane from going without traffic, the green signal timer is limited to a minimum and maximum value. Additionally, simulation is created to show the precision and contrast it with the current static system.

2. Module for detecting vehicle:

The suggested system leverages "You Only Look Once" (YOLO) for object classification, which offers the requisite precision and processing speed. Vehicle detection was taught utilizing a customized YOLO model, which can identify a number of vehicles including automobiles, bikes, large vehicles (such as trucks and buses), and motorbikes

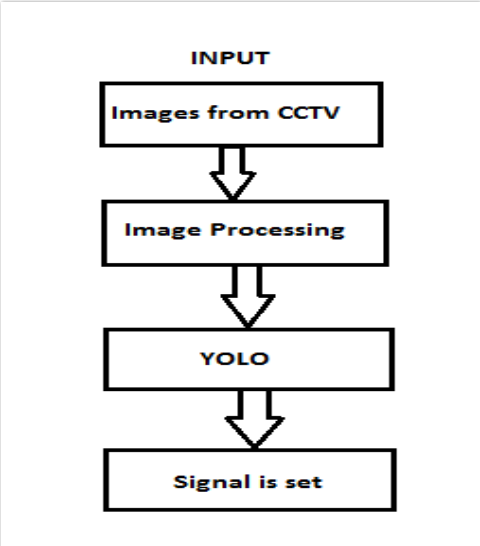
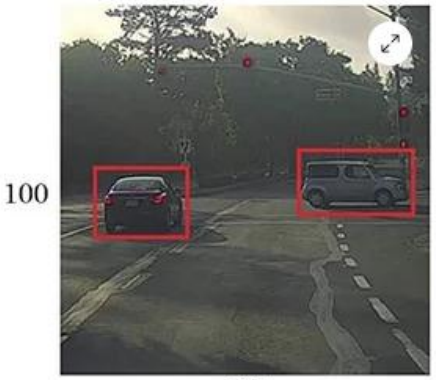


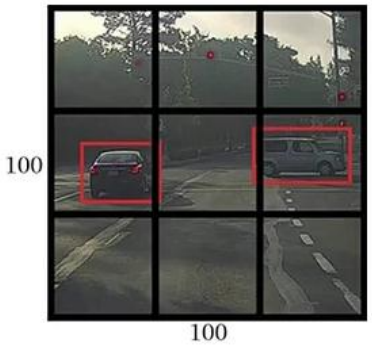
Fig. 1. Proposed System Model

The YOLOv3 algorithm first separates a picture into a grid. a grid cell each predicts some number of boundary boxes (sometimes stated as anchor boxes) around objects that score highly with the aforementioned predefined classes.

Yolo takes an input image initially:



The framework creates grids from input image (say a 3 X 3 grid):



Each grid employs image localization and classification. The object's bounding boxes and the associated class probabilities are then predicted by YOLO. Suppose we've got divided the image into a 3 x 3 grid and have created a total of three categories into which we want the object to be categorised. Consider that the categories are, respectively, Motorcycle, Car, and Pedestrian. So, for every grid, the label y is going to be an eight-dimensional vector:

Here,

- pc determines whether or not an object is included in the grid (it is the probability)
- by, bx, bw, and bh specifies the bounding box if any object is present

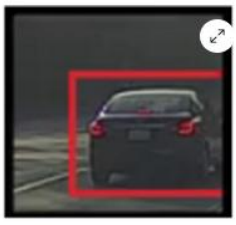
- c1, c2, and c3 stand in for the classes. In the event that the object is an automobile, then c2 will be 1 and both c1, c3 will be 0, and the process continues.

Suppose we choose a first grid as from example above:

Because this grid is empty, pc will be 0 and the label y of grid be:

y =	pc
	bx
	by
	bh
	bw
	c1
	c2
	c3

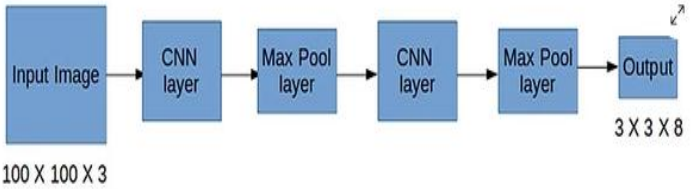
The grid is empty, therefore the bx, by, bh, bw, c1, c2, and c3 contain. Take another grid where an automobile is present (c2 = 1)



YOLO determines whether an actual object is present within the grid. If there are two objects (two automobiles) in the image YOLO takes midpoint among those two items and assign those objects to the grid that contains the centroid of those objects. Then car's center-left grid's y label reads

y =	1
	bx
	by
	bh
	bw
	0
	1
	0

Since we have an object in this grid, pc is enough to 1. then by, bx, bw, and bh will be calculated in relation to the specific grid cell that we are working with. Cars being of the second class, c2 = 1 & c3 and c1 = 0, respectively. We will therefore obtain an eight dimension output vector for each of the nine grids. The output will be three by three by eight in size. So now we have a target vector and an input image. Our model will be trained using the example above (input image: 100 X 100 X 3, output: 3 X 3 X 8):



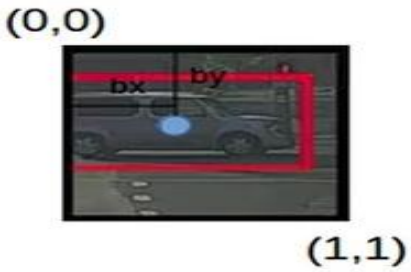
Encoding bounding boxes:

In relation to the grid cell that we are working with, the by, bx, bw, and bh are calculated. Take a look at the right-center grid, which has an automobile in it:

Therefore, just this grid will be used to calculate by, bx, bw, and bh. This grid's label y will be:

y =	1
	bx
	by
	bh
	bw
	0
	1
	0

pc = 1 Since a car is present in this grid and there's an



object, so c2 = 1. The coordinates for any or all of the grids (by, bx, bw, and bh) in YOLO are:

The by, bx are the x and y coordinates of the midpoint of the article with relevancy this grid. during this case:

The ratio of the height of the bounding box (the red box in the example above) to the height of the associated grid cell is known as bh, and in our case, it is approximately 0.9. So, bh = 0.9. BW is the proportion of the bounding box's width to the width of a grid cell. So, bw = 0.5 (approximately). This grid's y label will be:

The midpoint will always be inside the grid, hence the by and bx will always fall between 0 and 1. While bh & bw are frequently equal to 1, just in instance the bounding box's dimensions are equal to the grid's dimensions.

y =	1
	0.4
	0.3
	0.9
	0.5
	0
	1
	0

Implementation:

Step-1: Required libraries are imported

Step-2: Create a method to filter individual boxes depending on their criteria and chances

Step-3: Develop an algorithm to determine the IOU among two boxes

Step-4: Implement a Non-Max Reduction function

Step-5: Estimate the bounding box coordinates after creating a random volume with the size (19,19,5,85).

Step-6: Finally, we will build a function that will accept the CNN outputs as input and produce the suppressed boxes.

Step-7: Make a prediction for a random space using the yolo eval utility

Step-8: Test a previously trained YOLO algorithm on recent photos.

Step-9: Create a method to anticipate the bounding boxes, then save the photos with them.

Step-10: Using the estimate function, read a picture and estimation

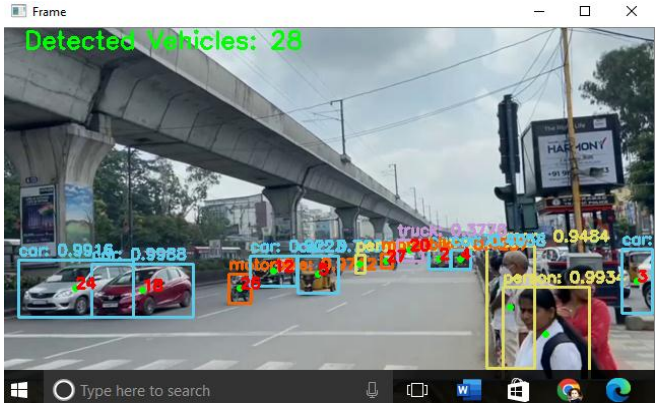
Step-11: Plot the estimations

Here videos of 4 lanes at a junction are taken and detected vehicles for 5 sec each on each side of the lane for calculation of vehicle density This is a picture of detected vehicles in the lane's right side

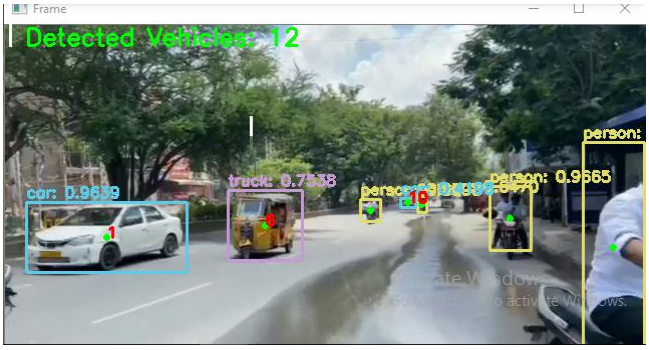
Detected vehicles on right lane = 57

Similarly, front lane detected vehicles =16

Left lane detected vehicles= 28



Back lane detected vehicles= 12



Considering the detected vehicles, Vehicle Traffic flow and vehicle can be calculated:

1)**Vehicle Traffic Flow**

The vehicle traffic flow on a road can be expressed q=n*3600/t

where,

q = vehicle flow (no. vehicles per hour)

n = no. vehicles passing in t seconds

t = time for passing vehicles (s)

2) Vehicle density:

K=(n*1000)/l

Where,

k = vehicles per km

n = no. of vehicles occupying a length l on the road

l = the length of road occupied by vehicles (m)

Considering traffic density calculated, the timer will be determined by the density on the road.

1) Module of signal switching

The vehicle detecting module's traffic density data is used by the Signal switching algorithm to set green traffic signal timer and update red traffic signal timers of all other signals. Additionally, it cycles back and forth among the signals in accordance with timers.

The detection module's vehicle detection information, as described in the preceding section, serves as the algorithm's input. This one is frequently in JSON format, also with accuracy and dimensions as the values and the label of the object detected as the key. After this data has been analyzed, the total number of automobiles in each class is determined.

Following that, the signal's green time is estimated and assigned, and also the signal's red time of all other signals are altered correspondingly. The method can be adjusted for any multiple of signals at a junction, either up or down.

The following elements are taken into account when creating the algorithm:

a) The time interval of algorithm to estimate traffic density and the length of green signal light, this gives information of image to be acquired at particular time

b) Count of lanes

c) Total number of each type of vehicle, including automobiles, trucks, motorbikes, etc.,

d) Using the aforementioned variables, traffic density was determined

e) Each car has lag at startup, and also the non-linear rise in lag faced by the vehicles in the rear results in additional time.

f) The avg speed of every class of car whenever the green light turns on, or the average amount of time each type of vehicle needs to cross the intersection.

g) The maximum and minimum cut-off time for the duration of a green signal, to stop blockage of traffic

The standard time for the principal signal of an initial cycle is ready when the algorithm is run for the first time, and as a result, the algorithm sets the times for all other indicators of the initial cycle and all subsequent signals. Every direction's vehicle detection is handled by a different thread, while the signal's timer is handled by the main thread. The detecting threads take a picture of the following direction whenever the green traffic timer of the present signal (or the red-light timer of the next green light) reaches 5 seconds. The timer for the succeeding green signal is then set when the result has been parsed. All of this occurs within background as the majority of threads continue to count down the remaining seconds on the green signal timer. This makes it possible for the timer to be assigned seamlessly, avoiding any latency. The succeeding signal turns green for number of seconds set by the algorithm whenever the green timer of this signal reaches zero.

When the signal that will turn into green next is 5 seconds away, the picture is taken. This gives the system a full 10 seconds to analyze the image, count the number of vehicles in every class that are visible there, determine the green traffic time, and set the duration of this signal as well as the red signal period of the next signal correspondingly. The normal speeds of cars at start and their acceleration times have been used, based on which an estimation of the typical time each class of vehicle takes to cross a junction was established, to get the ideal green signal time supported the number of vehicles of each class at a signal. Then, the green traffic time is determined using (1)

$$GST = \frac{\sum_{vehicleclass}^{n} (\text{Noofvehicles} * \text{Average time})}{(\text{NoOfLanes} + 1)}$$

Here:

• GST means Green Signal Time

• NoofVehiclesofClass means number of cars of each vehicle class on the signal detected by the module of vehicle detection.

• AverageTimeofClass means average time for vehicles of that class to cross the intersection.

• NoofLanes means Number of lanes in the junction.

The average-time it takes for each class of vehicles to cross an intersection is set according to the placement. i.e., By region, by city, by site, or perhaps by intersection-wise, assisted by the property of the intersection to facilitate traffic management. For this purpose, data from each transport company are often evaluated.

Instead of always pointing in the direction closest to the start, the signal periodically changes. This frequently occurs when the present system is coordinated, so people don't have to reroute or cause confusion and traffic lights turn green one at a time in a set pattern. Additionally, because of the existing system, traffic lights are in the same order and yellow traffic lights are regarded the same way.

Order of signal: Red to Green to Yellow to Red

4) Module of simulation

To model actual traffic, Pygame was used to create simulation from scratch. It helps with system visualisation and contrast with the current static system. A 4-way junction with 4 traffic lights is present there. Each signal has a timer on its top that indicates how long it will take for the light to change from green--yellow, red--green, or from yellow--red. The number of vehicles that have passed through the intersection is also shown next to each light. Automobiles, bicycles, buses, lorries, and rickshaws are all readily available. A few of the automobiles in the rightmost lane communicate cross the junction to make the simulation seem more realistic. When a vehicle is generated, the ability to rotate or not is set up using random data. It also has a timer that shows how much time has passed since the simulation started. A snapshot of the simulation's final result is shown in Fig. 2.

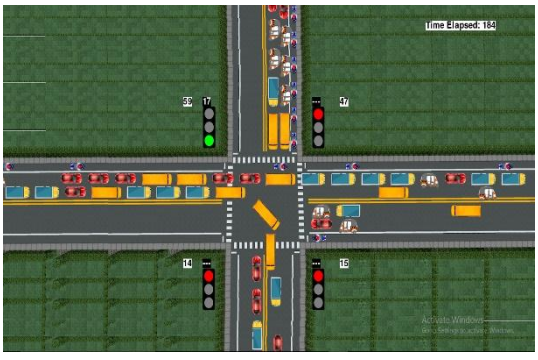


Fig 2: simulation output

Lane-wise Vehicle Counts

Lane 1: 92

Lane 2: 105

Lane 3: 26

Lane 4: 27

Total vehicles passed: 250

Total time passed: 300

No. of automobiles passed per unit time: 0.833333334

A Python module set called Pygame is cross-platform and made for creating video games. This provides music and graphics libraries created specifically for use with Python artificial language. On top of the superb SDL library, Pygame adds capabilities. This enables Python programmers to develop multimedia applications and fully functional games. Pygame could also be used on almost any platform and package and is fairly portable.

In the below image 4 sides of the junction is taken and vehicle detection is done in each lane, according to the vehicle detection count, timer is set green for more time on the side of lane with more vehicles than the other side with less vehicles.

Figure 3: 4 lanes output

Lane-wise Vehicle Counts

Lane 1 : 57

Lane 2 : 16

Lane 3 : 28

Lane 4 : 12

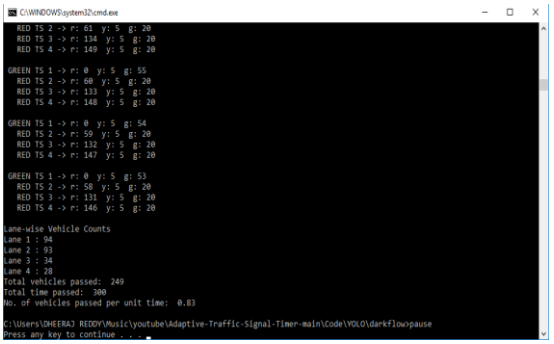


Fig4: Results of simulation

IV) ANALYSIS AND OUTCOMES

A. Evaluation of auto Recognition Module

Applying various test photos comprising varied numbers of cars, the vehicle detection-module was put to the test, and as a result, it was determined that between 75% and 80% of time, detection accuracy was accurate. In above Fig. 4, test result is displayed. This is frequently acceptable but not ideal. The absence of an accurate dataset is the primary cause of low accuracy. To build on this, model is frequently trained using actual traffic camera footage, increasing the system's accuracy.

B. Assessment of the suggested adaptive system

No matter of distribution, the suggested system always outperforms the static system. The improve in efficiency is based on how unevenly traffic is distributed among the lanes. The performance improves as the traffic distribution becomes more skewed.

The suggested approach only marginally outperforms the current system when traffic is evenly or near to evenly distributed throughout the 4 lanes. Here, performance has improved by around 9%.

The suggested system outperforms current method by a large margin when the traffic distribution is considerably skewed. Here, performance has improved by roughly 22%. This is often the nature of traffic dispersion that occurs in real-world settings.

The suggested system offers a vast performance gain than the exsisting system when the dispersion of traffic was significantly skewed. where the red and mete are separated by an excessive space and the line descends quickly. Here, performance has improved by roughly 36%.

The simulations are conducted for 5 min for every distribution under identical simulation conditions, including the dispersion of traffic, vehicle

speeds, that vehicles would turn, the distance between vehicles, etc. It was identifiably demonstrated that the suggested system performed 23% better on average than the existing system with set times. This also suggests a decrease in the length of time that green light stays idle because of the vehicles' waiting period

V. EPILOGUE

In conclude, the suggested system provides the signal as green for the lane with more traffic for a extended period of time than the lane with less traffic by setting the green signal time adaptive with the traffic density at the junction. This can lessen unneeded delays, traffic, and waiting times, which will ultimately result in less fuel usage and pollution.

In terms of the quantity of vehicles passing the junction, the system appears to perform around 23% better than the current method, which might be a considerable improvement. This method will be enhanced to achieve even greater results with further calibration utilizing real-world CCTV data for the model training.

REFERENCES

[1] M. M. Gandhi, D. S. Solanki, R. S. Daptardar and N. S. Baloorkar, "Smart Control of Traffic Light Using Artificial Intelligence," 2020 5th IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE), 2020, pp. 1-6, doi: 10.1109/ICRAIE51050.2020.9358334.

[2] https://www.analyticsvidhya.com/blog/2018/12/practical-guide-object-detection-yolo-framewor-python/

[3] https://www.researchgate.net/publication/224106688_Design_of_Intelligent_Traffic_Light_Controller_Using_Embedded_System

[4] A. Vogel, I. Oremović, R. Šimić and E. Ivanjko, "Improving Traffic Light Control by Means of Fuzzy Logic," 2018 International Symposium ELMAR, 2018, pp. 51-56, doi: 10.23919/ELMAR.2018.8534692.

[5] J. Hosur, R. Rashmi and M. Dakshayini, "Smart Traffic light control in the junction using Raspberry PI," 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC), 2019, pp. 153-156, doi: 10.1109/ICCMC.2019.8819695.

[6] Khushi, "Smart Control of Traffic Light System using Image Processing," 2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC), 2017, pp. 99-103, doi: 10.1109/CTCEEC.2017.8454966.

[7] .https://timesofindia.indiatimes.com/city/hyderabad/only-40-tech-firms-open-yet-traffic-jams-plague-it corridor/articleshow/92760259

[8] A. A. Zaid, Y. Suhweil and M. A. Yaman, "Smart controlling for traffic light time," 2017 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT), Aqaba, 2017, pp. 1-5, doi: 10.1109/AEECT.2017.8257768.

[9] Renjith Soman "Traffic Light Control and Violation Detection Using Image Processing"." IOSR Journal of Engineering (IOSRJEN), vol. 08, no. 4, 2018, pp. 23-27

[10] Siddharth Srivastava, Subhadeep Chakraborty, Raj Kamal, Rahil, Minocha, "Adaptive traffic light timer controller" , IIT KANPUR, NERD MAGAZINE.

[11] A. Kanungo, A. Sharma and C. Singla, "Smart traffic lights switching and traffic density calculation using video processing," 2014 Recent Advances in Engineering and Computational Sciences (RAECS), Chandigarh, 2014, pp. 1-6, doi: 10.1109/RAECS.2014.6799542.

[12] Ms. Saili Shinde, Prof. Sheetal Jagtap, Vishwakarma Institute Of Technology, Intelligent traffic management system:a Review, IJIRST 2016

[13] Arkatkar, Shriniwas & Mitra, Sudeshna & Mathew, Tom. "India" in Global Practices on Road Traffic Signal Control, ch.12, pp.217-242

[14] J. Hui, 'Real-time Object Detection with YOLO, YOLOv2 and now YOLOv3', 2018. [Online]. Available: https://medium.com/ @jonathan_hui/ real-time-object-detection-with-yolo-yolov2-28b1b93e2088

[15] J. Redmon, 'Darknet: Open Source Neural Networks in C', 2016. [Online]. Available: https://pjreddie.com/darknet/

[16] Tzutalin, 'LabelImg Annotation Tool', 2015. [Online]. Available: https://github.com/tzutalin/labelImg

[17] Li, Z., Wang, B., and Zhang, J. "Comparative analysis of drivers' start- up time of the first two vehicles at signalized intersections", 2016 J. Adv. Transp., 50: 228– 239. doi: 10.1002/atr.1318

[18] 'Pygame Library', 2019. [Online]. Available:https://www.pygame.org/wiki/about

[19] Open Data Science, 'Overview of the YOLO Object Detection Algorithm', 2018. [Online]. Available: https://medium.com/@ODSC/ overview-of-the-yolo-object-detection-algorithm-7b52a745d3e0