5th IEEE International Conference on Recent Advances and Innovations in Engineering- ICRAIE 2020 (IEEE Record#51050)

Smart Control of Traffic Light Using Artificial Intelligence

Mihir M. Gandhi
Dept. of Computer Engineering
K.J. Somaiya College of Engineering,
Vidyavihar University of Mumbai
Mumbai, India
mihir.mg@somaiya.edu

Devansh S. Solanki
Dept. of Computer Engineering
K.J. Somaiya College of Engineering,
Vidyavihar University of Mumbai
Mumbai, India
devansh.solanki@somaiya.edu

Rutwij S. Daptardar
Dept. of Computer Engineering
K.J. Somaiya College of Engineering,
Vidyavihar University of Mumbai
Mumbai, India
rutwij.d@somaiya.edu

Nirmala Shinde Baloorkar
Dept. of Computer Engineering
K.J. Somaiya College of Engineering,
Vidyavihar University of Mumbai
Mumbai, India
nirmalashinde@somaiya.edu

2020 5th IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE) | 978-1-7281-8867-6/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICRAIE51050.2020.9358334

***Abstract*—Traffic congestion is becoming one of the critical issues with increasing population and automobiles in cities. Traffic jams not only cause extra delay and stress for the drivers, but also increase fuel consumption and air pollution. Although it seems to pervade everywhere, megacities are the ones most affected by it. And its ever-increasing nature makes it necessary to calculate the road traffic density in real-time for better signal control and effective traffic management. The traffic controller is one of the critical factors affecting traffic flow. Therefore, the need for optimizing traffic control to better accommodate this increasing demand arises. Our proposed system aims to utilize live images from the cameras at traffic junctions for traffic density calculation using image processing and AI. It also focuses on the algorithm for switching the traffic lights based on the vehicle density to reduce congestion, thereby providing faster transit to people and reducing pollution.**

Keywords—Traffic control, Traffic light system, Traffic management, Intelligent transport systems, Smart surveillance, Computer Vision, Machine Learning, Object detection, YOLO.

I. Introduction

With the increasing number of vehicles in urban areas, many road networks are facing problems with the capacity drop of roads and the corresponding Level of Service. Many traffic-related issues occur because of traffic control systems on intersections that use fixed signal timers. They repeat the same phase sequence and its duration with no changes. Increased demand for road capacity also increases the need for new solutions for traffic control that can be found in the field of Intelligent Transport Systems.

Let us take the case study of Mumbai and Bangalore. Traffic flow in Bangalore is the worst in the world while Mumbai is close behind in fourth position, according to a report detailing the traffic situation in 416 cities across 57 countries. In Bangalore, a journey during rush-hour takes 71% longer. In Mumbai, it is 65% longer [1].

There are three standard methods for traffic control that are being used currently:

1) Manual Controlling: As the name suggests, it requires manpower to control the traffic. The traffic police are allotted for a required area to control traffic. The traffic police carry signboard, sign light, and whistle to control the traffic.

2) Conventional traffic lights with static timers: These are controlled by fixed timers. A constant numerical value is loaded in the timer. The lights are automatically switching to red and green based on the timer value.

3) Electronic Sensors: Another advanced method is placing some loop detectors or proximity sensors on the road. This sensor gives data about the traffic on the road. According to the sensor data, the traffic signals are controlled.

These conventional methods face certain drawbacks. The manual controlling system requires a large amount of manpower. As there is poor strength of traffic police, we cannot have them controlling traffic manually in all areas of a city or town. So a better system to control the traffic is needed. Static traffic controlling uses a traffic light with a timer for every phase, which is fixed and does not adapt according to the real-time traffic on that road. While using electronic sensors i.e., proximity sensors or loop detectors, the accuracy and coverage are often in conflict because the collection of high-quality information is usually based on sophisticated and expensive technologies, and thus limited budget will reduce the number of facilities. Moreover, due to the limited effective range of most sensors, the total coverage on a network of facilities usually requires a lot of sensors.

In recent years, video monitoring and surveillance systems have been extensively used in traffic management for security, ramp metering, and providing information and updates to travelers in real-time. The traffic density estimation and vehicle classification can also be achieved using video monitoring systems, which can then be used to control the timers of the traffic signals so as to optimize traffic flow and minimize congestion. Our proposed system aims to design a traffic light controller based on Computer Vision that can adapt to the current traffic situation. It uses live images from the CCTV cameras at traffic junctions for real-time traffic density calculation by detecting the number of vehicles at the

978-1-7281-8867-6/20/$31.00 ©2020 IEEE

signal and setting the green signal time accordingly. The vehicles are classified as a car, bike, bus/truck, or rickshaw to obtain an accurate estimate of the green signal time. It uses YOLO in order to detect the number of vehicles and then set the timer of the traffic signal according to vehicle density in the corresponding direction. This helps to optimize the green signal times, and traffic is cleared at a much faster rate than a static system, thus reducing the unwanted delays, congestion, and waiting time, which in turn will reduce the fuel consumption and pollution.

II. Litrature Review

Reference [2] proposes a solution using video processing. The video from the live feed is processed before being sent to the servers where a C++ based algorithm is used to generate the results. Hard code and Dynamic coded methodologies are compared, in which the dynamic algorithm showed an improvement of 35%.

Reference [3] proposes an Arduino-UNO based system that aims to reduce traffic congestion and waiting time. This system acquires images through the camera and then processes the image in MATLAB, where the image is converted to a threshold image by removing saturation and hues, and traffic density is calculated. Arduino and MATLAB are connected using USB and simulation packages, which are preinstalled. Depending on traffic count and traffic density, the Arduino sets the duration of green light for each lane. But this method has several flaws. The cars often overlap with each other and it is difficult to get a proper count of how many vehicles are on the road. Moreover, different objects interfered with the detection as they too were converted to black and white and there was no way of making a distinction between regular objects like billboards, poles, and trees with vehicles.

Reference [4] proposes a fuzzy logic-controlled traffic light that can be adapt to the current traffic situations. This system makes use of two fuzzy controllers with 3 inputs and one output for primary and secondary driveways. A simulation was done using VISSIM and MATLAB and for low traffic density, it improved traffic conditions.

Reference [5] proposes a smart traffic light system using ANN and fuzzy controller. This system makes use of images captured from cameras installed at traffic site. The image is first converted to a grayscale image before further normalization. Then, segmentation is performed using sliding window technique to count the cars irrespective of size and ANN is run through the segmented image, the output of which is used in fuzzy controller to set timers for red and green light using crisp output. Results had an average error of 2% with execution time of 1.5 seconds.

Reference [6] makes use of a support vector machine algorithm along with image processing techniques. From live video. images in small frames are captured and the algorithm is applied. Image processing is done using OpenCV and the images are converted to grayscale images before SVM is applied. This system not only detects traffic density but also detects red light violations.

Reference [7] proposes the use of adaptive light timer control using image processing techniques and traffic density. This system consists of microcontroller-controlled traffic light timer, high image sensing devices, MATLAB and transmission using UART principles. However, this system fails to prioritize the authorized emergency vehicles nor to detect accidents on the intersection.

Reference [8] reviews various techniques used for traffic light management system. This paper observes that each technique has a common architecture: choose input data, acquire traffic parameters from input data, process it, determine density, and update parameters.

- In the first method, VANETS are used to get information and location of every vehicle, which in turn is passed on to the nearest Intelligent Traffic light with the help of installed GPS Further, these ITLs will update the statistics and sent it to nearby vehicles. In case of accidents, the information would be sent to drivers to choose an alternate route to avoid congestion. However, this technique is not feasible as its deployment is quite expensive.
- In the second method, infrared sensor-based microcontrollers are used, which capture the unique ID of every car using transmitter and receiver. In case of an emergency situation, vehicle's radio frequency tags can be used to identify them and let other vehicles move. This method detects red light violations. However, this technique is not flexible due to the fact that infrared sensors need to be in sight.
- In the third method, fuzzy logic technique is used in which two fuzzy logic controllers are used – one is to optimize the signal and the other controller is used to extend the green phase of a road in an intersection. The sensors used to collect input data are video cameras that are placed at incoming and outgoing lines. The controller then utilizes the information collected through these sensors to make optimal decisions and minimize the goal function.
- In the fourth method, fuzzy logic is used, and the system takes in the number of vehicles and the average speed of traffic flow in each direction as the input parameters. The number of vehicles and the average speed of traffic flow can be determined using sensors placed on the road.
- In the fifth method, photoelectric sensors are used, which are set at some distance apart, that capture data and send it to the traffic cabinet, which calculates the weight of each road and then set the traffic light accordingly. However, the maintenance cost is quite high.
- In the sixth method, video imaging is used to capture the data. Dynamic background subtraction and various morphological operations are performed to capture a clear image of the vehicle. Every time a new vehicle enters the area of interest, a new rectangle is drawn and vehicle count is incremented. The algorithm is easy to implement but does not handle occlusion and shadow overlapping.

III. Proposed System

A. Proposed System Overview

Our proposed system takes an image from the CCTV cameras at traffic junctions as input for real-time traffic density calculation using image processing and object detection. As shown in Fig. 1, this image is passed on to the

vehicle detection algorithm, which uses YOLO. The number of vehicles of each class, such as car, bike, bus, and truck, is detected, which is to calculate the density of traffic. The signal switching algorithm uses this density, among some other factors, to set the green signal timer for each lane. The red signal times are updated accordingly. The green signal time is restricted to a maximum and minimum value in order to avoid starvation of a particular lane. A simulation is also developed to demonstrate the system's effectiveness and compare it with the existing static system.

B. Vehicle Detection Module

The proposed system uses YOLO (You only look once) for vehicle detection, which provides the desired accuracy and processing time. A custom YOLO model was trained for vehicle detection, which can detect vehicles of different classes like cars, bikes, heavy vehicles (buses and trucks), and rickshaws.

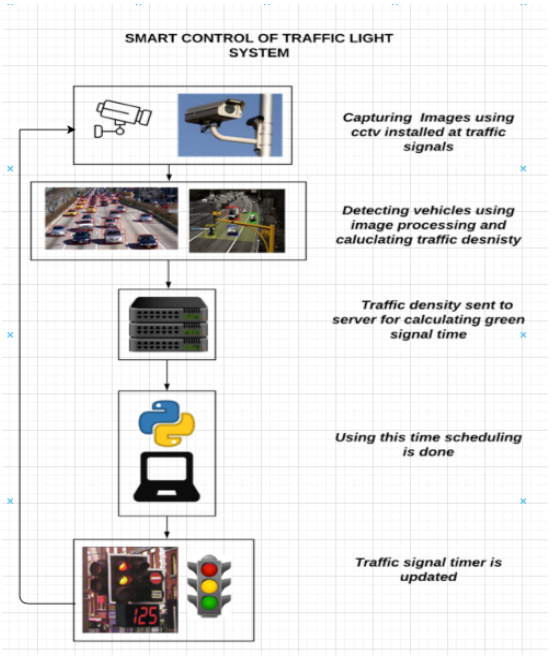

Fig. 1. Proposed System Model

YOLO is a clever convolutional neural network (CNN) for performing object detection in real-time. The algorithm applies a single neural network to the full image, and then divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities. YOLO is popular because it achieves high accuracy while also being able to run in real-time. The algorithm "only looks once" at the image in the sense that it requires only one forward propagation pass through the neural network to make predictions. After non-max suppression (which makes sure the object detection algorithm only detects each object once), it then outputs recognized objects together with the bounding boxes. With YOLO, a single CNN simultaneously predicts multiple bounding boxes and class probabilities for those boxes [9].

The backbone CNN used in YOLO can be further simplified to improve processing time. Darknet is an open source neural network framework written in C and CUDA. It is fast, easy to install, and supports CPU and GPU computation [11]. With DarkNet, YOLO achieves 72.9% top-1 accuracy and 91.2% top-5 accuracy on ImageNet. Darknet uses mostly 3 × 3 filters to extract features and 1 × 1 filters to reduce output channels. It also uses global average pooling to make predictions [10].

The dataset for training the model was prepared by scraping images from google and labelling them manually using LabelIMG, a graphical image annotation tool [12]. Then the model was trained using the pre-trained weights downloaded from the YOLO website. The configuration of the .cfg file used for training was changed in accordance with the specifications of our model. The number of output neurons in the last layer was set equal to the number of classes the model is supposed to detect by changing the 'classes' variable. In our system, this was 4 viz. Car, Bike, Bus/Truck, and Rickshaw. The number of filters also needs to be changed by the formula 5*(5+number of classes), i.e., 45 in our case. After making these configuration changes, the model was trained until the loss was significantly less and no longer seemed to reduce. This marked the end of the training, and the weights were now updated according to our requirements. These weights were then imported in code and used for vehicle detection with the help of OpenCV library. A threshold is set as the minimum confidence required for successful detection. After the model is loaded and an image is fed to the model, it gives the result in a JSON format i.e., in the form of key-value pairs, in which labels are keys, and their confidence and coordinates are values. Again, OpenCV can be used to draw the bounding boxes on the images from the labels and coordinates received.

Fig. 2 shows test images on which our vehicle detection model was applied. The left side of the figure shows the original image and the right side is the output after the vehicle detection model is applied on the image, with bounding boxes and corressponding labels

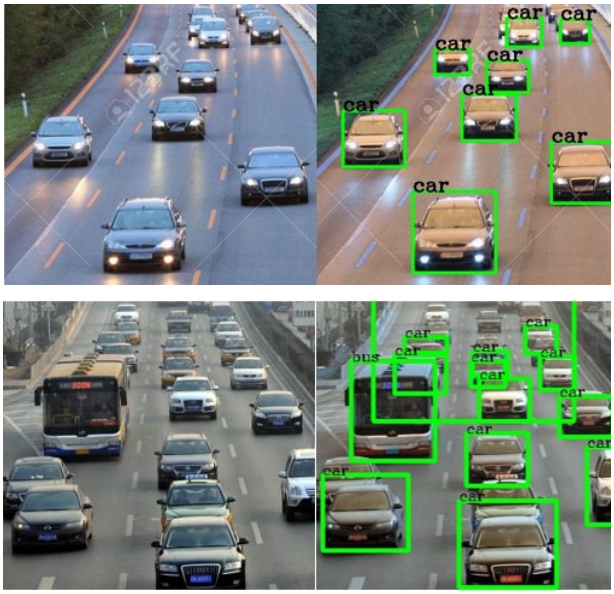

Fig. 2. Vehicle Detection Results

C. Signal Switching Module

The Signal Switching Algorithm sets the green signal timer according to traffic density returned by the vehicle detection module, and updates the red signal timers of other signals accordingly. It also switches between the signals cyclically according to the timers.

The algorithm takes the information about the vehicles that were detected from the detection module, as explained in the previous section, as input. This is in JSON format, with the label of the object detected as the key and the confidence and coordinates as the values. This input is then parsed to calculate the total number of vehicles of each class. After this, the green signal time for the signal is calculated and assigned to it, and the red signal times of other signals are adjusted accordingly. The algorithm can be scaled up or down to any number of signals at an intersection.

The following factors were considered while developing the algorithm:

1) The processing time of the algorithm to calculate traffic density and then the green light duration – this decides at what time the image needs to be acquired
2) Number of lanes
3) Total count of vehicles of each class like cars, trucks, motorcycles, etc.
4) Traffic density calculated using the above factors
5) Time added due to lag each vehicle suffers during start-up and the non-linear increase in lag suffered by the vehicles which are at the back [13]
6) The average speed of each class of vehicle when the green light starts i.e. the average time required to cross the signal by each class of vehicle [14]
7) The minimum and maximum time limit for the green light duration - to prevent starvation

When the algorithm is first run, the default time is set for the first signal of the first cycle and the times for all other signals of the first cycle and all signals of the subsequent cycles are set by the algorithm. A separate thread is started which handles the detection of vehicles for each direction and the main thread handles the timer of the current signal. When the green light timer of the current signal (or the red light timer of the next green signal) reaches 5 seconds, the detection threads take the snapshot of the next direction. The result is then parsed and the timer of the next green signal is set. All this happens in the background while the main thread is counting down the timer of the current green signal. This allows the assignment of the timer to be seamless and hence prevents any lag. Once the green timer of the current signal becomes zero, the next signal becomes green for the amount of time set by the algorithm.

The image is captured when the time of the signal that is to turn green next is 5 seconds. This gives the system a total of 10 seconds to process the image, to detect the number of vehicles of each class present in the image, calculate the green signal time, and accordingly set the times of this signal as well as the red signal time of the next signal. To find the optimum green signal time based on the number of vehicles of each class at a signal, the average speeds of vehicles at startup and their acceleration times were used, from which an estimate of the average time each class of vehicle takes to cross an intersection was found [14]. The green signal time is then calculated using (1).

$$GST = \frac{\sum_{vehicleClass} (NoOfVehicles_{vehicleClass} * AverageTime_{vehicleClass})}{(NoOfLanes + 1)} \quad (1)$$

where:

- GST is green signal time
- noOfVehiclesOfClass is the number of vehicles of each class of vehicle at the signal as detected by the vehicle detection module,
- averageTimeOfClass is the average time the vehicles of that class take to cross an intersection, and
- noOfLanes is the number of lanes at the intersection.

The average time each class of vehicle takes to cross an intersection can be set according to the location, i.e., region-wise, city-wise, locality-wise, or even intersection-wise based on the characteristics of the intersection, to make traffic management more effective. Data from the respective transport authorities can be analyzed for this.

The signals switch in a cyclic fashion and not according to the densest direction first. This is in accordance with the current system where the signals turn green one after the other in a fixed pattern and does not need the people to alter their ways or cause any confusion. The order of signals is also the same as the current system, and the yellow signals have been accounted for as well.

Order of signals: Red → Green → Yellow → Red

D. Simulation Module

A simulation was developed from scratch using Pygame to simulate real-life traffic. It assists in visualizing the system and comparing it with the existing static system. It contains a 4-way intersection with 4 traffic signals. Each signal has a timer on top of it, which shows the time remaining for the signal to switch from green to yellow, yellow to red, or red to green. Each signal also has the number of vehicles that have crossed the intersection displayed beside it. Vehicles such as cars, bikes, buses, trucks, and rickshaws come in from all directions. In order to make the simulation more realistic, some of the vehicles in the rightmost lane turn to cross the intersection. Whether a vehicle will turn or not is also set using random numbers when the vehicle is generated. It also contains a timer that displays the time elapsed since the start of the simulation. Fig. 3 shows a snapshot of the final output of the simulation.

Pygame is a cross-platform set of Python modules designed for writing video games. It includes computer graphics and sound libraries designed to be used with the Python programming language. Pygame adds functionality on

Fig. 3. Simulation output

top of the excellent SDL library. This allows users to create fully featured games and multimedia programs in the python language. Pygame is highly portable and runs on nearly every platform and operating system. It is free and licensed under LGPL [15].

IV. Results And Analysis

A. Evaluation of Vehicle Detection Module

The vehicle detection module was tested with a variety of test images containing varying amounts of vehicles, and the accuracy of detection was found to be in the range of 75-80%. Some test results are shown above in Fig. 3. This is satisfactory, but not optimum. The primary reason for low accuracy is the lack of a proper dataset. To improve upon this, real-life footage from traffic cameras can be used to train the model, so that accuracy of the system can be improved.

B. Evaluation of the proposed adaptive system

To measure how the proposed adaptive system compares to the existing static system, 15 simulations of both the systems were run for a period of 5 minutes each, with varying traffic distributions across the 4 directions. Performance was measured in terms of the number of vehicles that were able to pass the intersection per unit of time. In other words, the idle time of the signal i.e. the time when the signal is green but no car passes the intersection is compared. This has an impact on the waiting time of vehicles and queue lengths of the other signals.

The distribution [a,b,c,d] means that the probability of a vehicle being in lane 1, lane 2, lane 3, and lane 4 is a/d, (b-a)/d, (c-b)/d, and (d-c)/d, respectively. For example, in simulation 1, the distribution is [300,600,800,1000] which means probabilities of 0.3, 0.3, 0.2, and 0.2. The results obtained were tabulated in the form of number of vehicles passed lane-wise and the total number of vehicles passed.

TABLE I. Simulation Results of Current Static System

No.	Distribution	Lane1	Lane 2	Lane 3	Lane 4	Total
1	[300,600,800,1000]	70	52	52	65	239
2	[500,700,900,1000]	112	49	48	31	240
3	[250,500,750,1000]	73	53	63	62	251
4	[300,500,800,1000]	74	44	65	71	254
5	[700,800,900,1000]	90	32	25	41	188
6	[500,900,950,1000]	95	71	15	14	195
7	[300,600,900,1000]	73	63	69	24	229
8	[200,700,750,1000]	54	89	10	67	220
9	[940,960,980,1000]	100	10	8	4	122
10	[400,500,900,1000]	81	29	88	37	235
11	[200,400,600,1000]	42	47	54	86	229
12	[250,500,950,1000]	39	52	93	22	206
13	[850,900,950,1000]	74	10	13	17	114
14	[350,500,850,1000]	49	46	69	50	214
15	[350,700,850,1000]	51	64	37	43	195

TABLE II. Simulation Results of Proposed Adaptive System

No.	Distribution	Lane1	Lane 2	Lane 3	Lane 4	Total
1	[300,600,800,1000]	87	109	41	50	287
2	[500,700,900,1000]	128	55	49	25	257
3	[250,500,750,1000]	94	50	60	58	262
4	[300,500,800,1000]	89	46	69	59	263
5	[700,800,900,1000]	185	25	23	28	261
6	[500,900,950,1000]	94	118	11	16	239
7	[300,600,900,1000]	87	68	70	33	258
8	[200,700,750,1000]	56	108	19	78	261
9	[940,960,980,1000]	193	6	5	7	211
10	[400,500,900,1000]	97	29	100	34	260
11	[200,400,600,1000]	26	52	67	99	244
12	[250,500,950,1000]	52	75	101	7	235
13	[850,900,950,1000]	154	17	12	18	201
14	[350,500,850,1000]	64	53	80	47	244
15	[350,700,850,1000]	66	82	40	48	236

As it can be seen in fig. 4, the proposed adaptive system always performs better than the current static system, regardless of the distribution. The improvement in performance depends on how skewed the distribution of traffic is across the lanes. More the skewness of the distribution of traffic, more improved is the performance.

- When the distribution of traffic among the 4 lanes is equal or almost equal, then the proposed system performs only slightly better than the current system. This is the case in simulation numbers 1, 2, 3, and 4. The performance improvement is about 9% here.
- When the distribution of traffic is moderately skewed, then the proposed system performs significantly better than the current system. This is the case in simulation numbers 5, 6, 7, 8, 14, and 15. The performance improvement is about 22% here. Usually, this is the kind of traffic distribution seen in real life scenarios.
- When the distribution of traffic is sharply skewed, then the proposed system has a huge performance improvement as compared to the current system. This is the case in simulation numbers 9 and 13, where the red line drops sharply and there is a large gap between the red and green line. The performance improvement is about 36% here.

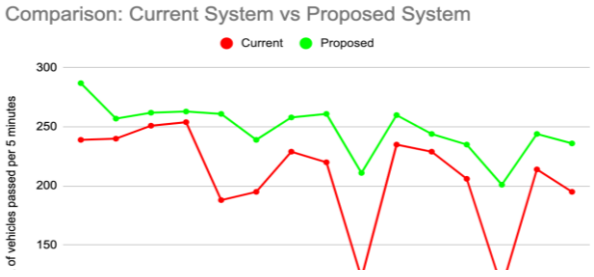

Fig. 4. Comparison of current static system and proposed adaptive system

With all simulation conditions same i.e distribution of traffic, speeds of vehicles, probability of vehicles turning, the gap between vehicles, and so on, the simulations were run for a total period of 1 hour 15 minutes, with 300 seconds i.e. 5 minutes for each distribution and it was found out that the proposed system, on an average, increased the performance by about 23% as compared to the current system with fixed times. This implies a reduction in idle green signal time as well as the waiting time of the vehicles.

On comparing these results with some alternative adaptive system, it was found that the proposed system performs better than some of those. For example, [2] gives an accuracy of 70% as compared to 80% of the proposed system. Reference [3] achieves an average performance improvement of 12% as compared to static systems while the proposed system achieves 23% improvement.

V. Conclusion And Future Work

In conclusion, the proposed system sets the green signal time adaptively according to the traffic density at the signal and ensures that the direction with more traffic is allotted a green signal for a longer duration of time as compared to the direction with lesser traffic. This will lower the unwanted delays and reduce congestion and waiting time, which in turn will reduce fuel consumption and pollution.

According to simulation results, the system shows about 23% improvement over the current system in terms of the number of vehicles crossing the intersection, which is a significant improvement. With further calibration using real-life CCTV data for training the model, this system can be improved to perform even better.

Moreover, the proposed system possesses certain advantages over the existing intelligent traffic control systems prevalent such as Pressure Mats and Infrared Sensors. The cost required to deploy the system is negligible as footage from CCTV cameras from traffic signals is used, which requires no additional hardware in most cases, as intersections with heavy traffic are already equipped with such cameras. Only minor alignment may need to be performed. The maintenance cost also goes down as compared to other traffic monitoring systems such as pressure mats that normally suffer wear and tear due to their placement on roads where they are subjected to immense pressure constantly. Thus, the proposed system can thus be integrated with the CCTV cameras in major cities in order to facilitate better management of traffic.

The project can be further expanded to include the following functionalities to enhance traffic management and bring down congestion:

1) Identification of vehicles violating traffic rules: The vehicles running red lights can be identified in an image or a video stream by defining a violation line and capturing the number plate of the image if that line is crossed when the signal is red. Lane changing can also be identified similarly. These can be achieved by background subtraction or image processing techniques.

2) Accident or breakdown detection: Intersections also tend to experience severe crashes due to the fact that several types of injurious crashes, such as angle and left-turn collisions, commonly occur there. Therefore, accurate and prompt detection of accidents at intersections offers tremendous benefits of saving properties and lives and minimizing congestion and delay. This can be achieved by identifying the vehicles that remain stationary for a long time in an inappropriate position such as in the middle of the road, so that parked vehicles are not included in this.

3) Synchronization of traffic signals across multiple intersections: Synchronizing signals along a street can benefit the commuters as once a vehicle enters the street, it may continue with minimal stopping. [16].

4) Adapting to emergency vehicles: Emergency vehicles such as an ambulance need to be given quicker passage through the traffic signals. The model can be trained to detect not just vehicles but also be able to recognize that it is an emergency vehicle and accordingly adapt the timers such that the emergency vehicle is given priority and is able to cross the signal at the earliest.

References

[1] TomTom.com, 'Tom Tom World Traffic Index', 2019. [Online]. Available: https://www.tomtom.com/en_gb/traffic-index/ranking/

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

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

[4] 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.

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

[6] 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.

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

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

[9] 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

[10] 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

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

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

[13] 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

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

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

[16] 'Traffic Signal Synchronization'. [Online]. Available: https://www.cityofirvine.org/signal-operations-maintenance/traffic-signal-synchronization