Smart Traffic Control System Using Deep Learning

Abhijit Gadge[1], Hardik Bhawsar[2], Piyush Gondkar[3], Praveen Chourey[4], Geeta Navale[5]

[1-5]Department of Computer Engineering, Sinhgad Institute of Technology and Science, Savitribai Phule Pune University, Pune

---

Abstract - *Traffic congestion is a daunting problem that affects the daily lives of many people in many countries across the world. Traffic congestion cannot be mitigated just by increasing the infrastructure. It is necessary to control the traffic sequence with the help of an intelligent system for transportation management. Moreover, the traditional systems cannot make critical decisions in certain events. The use of modern computer technology with the help of deep learning can help to solve the problems mentioned above with improved and efficient ways to manage traffic and reduce congestion along the heaviest flowing roads in major cities. This paper proposes a vehicle counting model and comparison between different versions of YOLO along with a dynamic traffic signal timer setting algorithm. After video processing, the vehicles of all the different lanes are counted and depending on the count of a particular lane, it is classified in one of the three classes of the density of vehicles such as low, medium, and high. While classifying a particular lane, the other 3 lanes are also taken into consideration and depending on the number of vehicles present in those lanes the green signal timer is set dynamically. The model will also identify the category of a particular vehicle.*

***Key Words*: Deep Learning, Video Processing, Image Processing, Computer Vision, Traffic, Object Detection, Convolutions Neural Network.**

1.INTRODUCTION

The traffic congestion problem is increasing day by day everywhere around us and must be solved more intelligently considering the continuous increase in traffic in not-so-distant future. Among most proposed solutions like building more infrastructure or improving citizen commute patterns, creating an intelligent control system to manage the traffic flow comes to be a better and most efficient solution. The more intelligent control system part over here means to use more sophisticated timing algorithms with the help of machine learning algorithms to solve the problem autonomously. Instead of using fixed time intervals system should be able to set variable timers to each lane. These variable timers should be computed by the algorithms based on different traffic conditions in real-time. Using such solutions in real life can positively affect various characteristics of a daily commute, especially in metropolitan cities. It may also help in saving many valuable resources like time and fuel while managing the smooth traffic flow with the least stops possible.

The primary motivation for this kind of solution can be said to be the increasing transportation in major cities. Fast transportation systems and rapid transit systems are vital for economic developments for any state. More population means more vehicles on the streets day-by-day. Traditional traffic management systems are not built with the consideration of increasing traffic at this high pace. Thus, it cannot keep up with the variable changing traffic conditions daily. In the past few years, fields like machine learning and computer vision have developed to be so better and easy to implement these days. Using concepts like deep learning to detect the vehicles and automatically adjust accordingly helps in many ways for better transportation flow. An algorithm can be created to detect the patterns of the traffic flow and react accordingly. This not only makes the system robust but also more efficient in managing the traffic.

The main goal of this work is to develop a fast and reliable solution to the traffic congestion problem. The solution must be intelligent and has decision-making capabilities depending on the different situations in the different lanes. There are different phases of the proposed model. The first phase is video processing. The continuous video stream from the traffic camera situated at the traffic signal junction is fed to the model. During the video processing, the frames are extracted from the video input. There are hundreds of frames present in a stream of few seconds, so it is not possible to process each frame and it is not required because there is no such significant change in the vehicle density between two consecutive frames. Therefore, each frame after a particular time interval is taken. Then shrinking the size of the frame is done from the actual size of the frame to 416 x 416 pixels. After this phase, the processed frame is fed to the model for object detection. There are two separate models developed for object detection using each framework i.e., YOLOv3 and YOLOv5. The neural network extracts the features and identifies the object using bounding boxes. After object detection, the model predicts the class of the vehicle from a car, bike, truck, etc. The total vehicle count of a lane is passed to the dynamic traffic signal timer algorithm. The algorithm considers the density of all the other lanes and calculates the relativity between them. Depending on which, it classifies the lane in either low, medium, or high vehicle density class, it decides the green signal timer for a particular lane.

2. RELATED WORK

2.1 Literature Review

Surging traffic levels and increasingly busier roads are common issues across the globe. Consequently, there is an increasing requirement to develop intelligent traffic surveillance systems that can play a crucial role in highway monitoring and road management systems. Author Jose Melo et al. have addressed challenges of lane detection with the help of segmentation and clustering methods [1]. The algorithm they have used is the Public Transportation facilities and Equipment Management System (PTMS) algorithm, which helps in resulting input to higher-level traffic monitoring systems like estimating traffic speed, frequency of lane changes, and accident detection. Alvin Abdagic et al.. proposed a solution to memorize stationary vehicles, detecting turning movement using an optical flow algorithm [2]. Instead of relying heavily on the sensors, their solution uses CPU processing. Thus, reducing the costs of both development and maintenance. The process contains a video capture using infrared (IR) cameras converted into frames of intensities of traffic. Using these intensities optical flow is calculated and a magnitude is generated.

The main problem which arises while detecting the vehicles is identifying the lane and vehicles present in that lane. A solution to this problem is given by Prashant Jadhav et al. using image processing in MATLAB [4]. In the proposed study, a system to count the number of vehicles on roads is discussed. The method involves analyzing a sequence of road images to determine the flow of vehicles for the given time and place. An approach proposed by authors Zhe Dai et al. in which the solution revolves around developing a video-based vehicle counting framework [7]. The proposed method used an ANN (Artificial Neural Network) model to estimate vehicle densities, traffic flow rates, and vehicle trajectories on different video captures of road intersections.

Surojit Dey et al. proposed a solution that is efficient and fast which uses image processing, data mining, and artificial neural network [6]. The proposed method measures the average speed of traffic and traffic density from cameras in real time. The processing speed was faster as well. However, only prediction and estimation of the traffic system were done. There is no proper solution to the traffic congestion problem. To give an optimal solution regarding the traffic congestion, Jingwei Cao et al. [8] used a YOLO (You Only Look Once) and SSD (Single Shot multi-box Detector) model which are one of the current mainstream object detection frameworks based on deep learning. A Faster R-CNN (reign-based convolutional neural network) and regression idea of YOLO realizes the detection and classification of multiple bounding boxes based on a simple end-to-end network that works on the KITTI dataset.

Until the present time, advanced traffic control research in urban areas has been largely restricted to freeways. This does not mean that urban areas have been ignored. Author Carter et al. gave traffic flow patterns by applying traffic flow and network theories [10]. The advantage of using this is the significant use of mathematical models and their simulations. A tremendous amount of time and power is wasted due to a green traffic light with no cars passing in its lane. Many solutions were proposed before to solve this, but installation cost and their cost of maintenance were too high. To replace these high-cost installation techniques Cheung et al. proposed a traffic surveillance technology system based on wireless sensors [11]. He proposed a prototype of a wireless sensor network for the Intelligent Transportation System (WITS) which helped him gather data and transfer data.

2.2 Dataset

The details of the dataset used in training the object detection models in this proposed solution can be found in this sub-section. The dataset used is Microsoft Common Objects in Context image (MS-COCO) dataset. This dataset is a well-known dataset developed by Microsoft [13]. This is a large-scale object detection dataset containing up to 1.5 million instances. There are around 330,000 images of 80 different classes and more than 220,000 labeled images. There are 5 classes of interest which are used in the proposed model which are motorbike, bicycle, truck, car and bus.

Some images from the dataset used in training object detection are presented with their basic description text as mentioned on the MS-COCO website. In the first image, figure 1 crowded traffic on a large one-way road can be seen.

Fig - 1: Dataset Sample Image 1

In the second image, figure 2 a large group of bikers and some big vehicles waiting on a signal can be seen. These types of scenes are helpful to train the model on highly dense traffic with primarily bikes present.

Fig - 2: Dataset Sample Image 2

In the third image as seen in figure 3, a wide view of a large intersection is seen. This type of images can be used to train the model on a bird's eye view of all the ways the traffic can move.

Fig - 3: Dataset Sample Image 3

3. PROPOSED SYSTEM ARCHITECTURE

The proposed system architecture can be referred from figure 4 mentioned later in this article. The figure gives a detailed data flow design of the complete system. The first stage of the system starts with input from a video capturing device or a video from storage as a video feed to the algorithm. This video feed is then passed over to the graphical user interface to be displayed on the dashboard. Along with that, the video feed is also passed to the main algorithm of the system to select frames at certain intervals. The extracted frames can then be passed for image processing like scaling to the correct resolution and with correct color formats. This must be done in order to make all the input from each lane uniform before sending them into the neural network for the next task. Once the inputs are all in the same format, they can be passed to the object detection algorithm where with the help of Convolutional Neural Network (CNN) the vehicle types, counts and positions can be found out. This detection data can then be stored in a local database for any future use. The vehicle counts are then converted to a magnitude of densities on each lane and passed over to the timing algorithm which will give the resultant timers to be set for the next lane to go with a green signal. All the data generated so far, like the vehicles' coordinates, their types, timers will be sent to the graphical user interface for monitoring on the dashboard. All the functions mentioned above will be explained more briefly in the upcoming section of this article.

3.2 Software Modules

The Software Modules required in development of this solution are mentioned in the following list.

- Python 3.5 (or Higher).
- TensorFlow.
- Keras.

4. METHODOLOGY

The proposed system consists of a total of three main phases. It starts with the inputs of the video feeds from a camera unit. Processing the inputs also comes in this phase. In the next phase, the process of object detection occurs on the inputs. In this phase, the data describing the current traffic will appear. In the last phase, earlier data will be used to compute the correct timers for each lane.

4.1 Video Processing

In this first part of the traffic control system, the inputs will be accepted in separate four individual videos of each lane in focus. These videos might come in any resolution or color format. The first task in this segment of the solution will be to update the resolution of the input videos to make all four inputs uniform and consistent for the detection model. The videos are adjusted to a specific resolution of 416 by 416 pixels each and the color formats of the video will be adjusted in the RGB (Red, Green, Blue) color format. Any videos which might get sent in other color schemas like CMYK (Cyan, Magenta, Yellow, blacK) or HSV (Hue, Saturation, Value) will be converted to RGB in a 3-dimensional array structure, containing 3, 2-dimensional matrices of each color component value in the video frames. Finally, these videos will be chopped down to some selected frames based on a certain interval.

4.2 Object Detection

The next phase of the proposed solution is to apply object detection to the received frames from the previous stage. Here the frames received will be passed to the object detection model of the user's choice in a multi-threaded environment to concurrently get the detections of all properly visible vehicles in each scene. To achieve this object detection there are two proposed choices of well-known object detectors trained on the MS-COCO dataset, as explained in the previous section. The detailed description of both the selected models has explained hereafter.

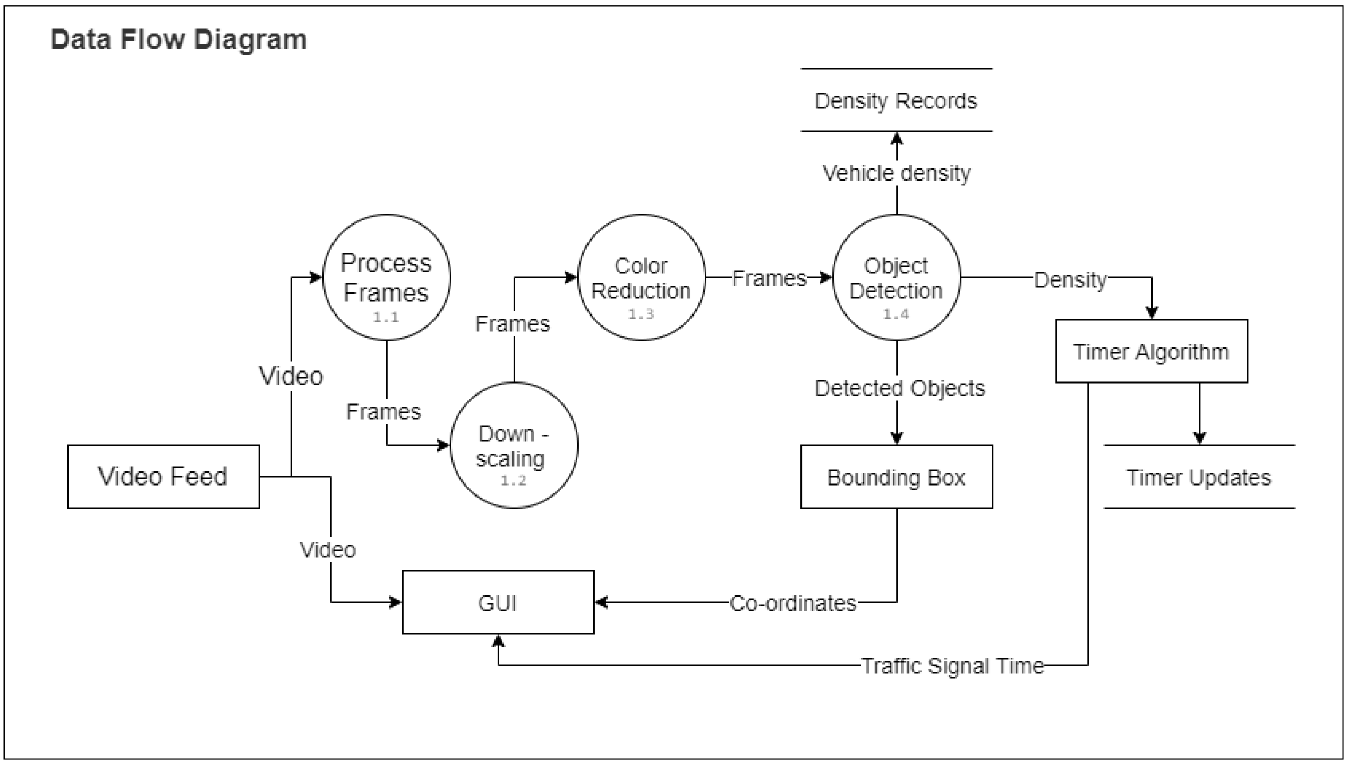


Fig -4: Data Flow Diagram for the Proposed System Architecture.

YOLOv3 is the third version of its kind of detection architecture [10]. YOLOv3 is one of the many real-time object detection algorithms which can identify a set of different classes of objects in a given image or a video. YOLOv3 starts with separating the images into smaller grids which are further used to predict anchor boxes or also called the boundary boxes around the object being detected. It does this by applying the 1x1 detection kernel on the feature map at multiple different places in the network and multiple resolutions. The predictions are made in three different scales while down sampling the image resolutions in the factors of 32, 16, and 8 respectively. The shape of the detection kernel can be described using the following expression,

$$1 \times 1 \times (B \times 5 + C)$$

where B denotes the maximum count of anchor box a cell on a feature map can predict, C denotes the total number of classes present in the training dataset and 5 is split into the number of bounding box attributes and one object confidence. When the expression is used in the MS-COCO dataset with a total of 80 classes (C=80) and considering bounding boxes per feature map to 3 (B=3) we get the shape of the detection kernel as 1x1x255. Every scale uses three anchor bounding boxes per layer, here the three largest boxes are in the first scale, the next three of the medium sizes used in the second scale, and the last three of small sizes in the third scale. This way the algorithm can detect every size of an object image from small to large objects. Finally, the regions with high confidence scores are considered for detection.

YOLO version 5 is the latest update in the YOLO object detection neural networks family [11]. Being the successor of version 4, it was released right after version 4 was released in the second half of the year 2020. At the time of writing this paper, it has been proven to be the most successful object detection complete deep learning model as compared to its predecessors as well as some other detection models like EfficientDet. The YOLOv5 shows a remarkable improvement in the architecture over the older version 3 YOLO architecture. The authors have used the backbone of CSPNet which stands for Cross Stage Partial Network and helps in enhancing the CNN being developed. It contains the head of the YOLOv3 but with the Generalized Intersection over Union Loss metric (GIoU-Loss). For the COCO dataset, the best fit count for the anchor-boxes using K-means comes up to 5 which significantly reduces the training time and increases the accuracy of the network.

The implementation of YOLOv5 is achieved in the PyTorch machine learning framework which is built over the Python programming language. In this proposed solution, among the five different sizes of models, the large network i.e., “yolo5L” was used as it fit the requirements based on the diversity of objects which were being detected. Once the object detection phase is complete, there will be a set of results from detecting each lane containing the classes of vehicles, count, locations of detection in the frame (bounding boxes). These results will be computed every certain interval to get new updates of each lane to decide the next timer values, which come in the next phase of the solution.

4.3 Dynamic traffic signal timer algorithm

To overcome the flaws of static timers, it becomes an essential requirement to replace them with Dynamic timers by considering the current scenario of traffic congestion problems in metropolitan cities. Taking advantage of traffic data from cameras situated at the intersection, our algorithm focuses on optimizing the timer by providing Optimal Flow and Minimum waiting time at the intersection.

Initially, the algorithm calculates the threshold value based on the current traffic scenario, so that it can be used for determining the class of each lane. The threshold is determined using the mean of all the densities. The calculation of the mean is based on the densities from lanes where the traffic signal is red. The classes are divided into three categories namely Low, Medium and High.

Low	Density is lower than the threshold value with more than a marginal difference.
Medium	Density is relatively close to the Threshold value.
High	Density is higher than the threshold value with more than a marginal difference.

However, there are some assumptions and presets which must be kept in mind.

Asumptions:

1. Allocation of the Green signal will work in a Round Robin fashion irrespective of traffic density.
2. At any particular moment in time, only one signal is Green while the remaining three signals are Red.
3. At any particular lane, there cannot be more than a hundred numbers of vehicles.
4. The signal state is unchangeable once it is in Green State.
5. The minimum and maximum value of the Green signal is 30 seconds and 120 seconds respectively.

Presets:

1. All the lanes belong to the medium class.
2. Green and Red signal timers are set to 60 seconds.

After calculating the densities of individual lanes, the algorithm updates the timer signal based on the following conditions:

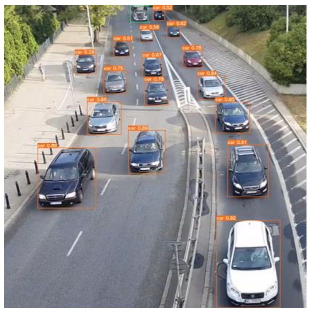
1. If the densities of all four lanes belong to the same class, then no need to update the timers.
2. If any one of the lanes belongs to a "High" class and the rest of the lanes belong to a "Low" class, then the lane with "High" density will be provided with extra time by reducing an equal amount of time from the rest of the lanes.
3. If some of the lanes belong to the "High" class and others belong to the "Medium" class, then priority will be given to the "High" class.
4. If lanes belong to all three classes, then more time is reduced from the "Low" class and provided to the "High" and "Medium" classes.

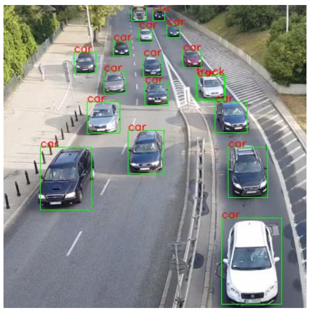
The manipulation of the timer of the subsequent lane does not depend directly on the class which is assigned to them by the algorithm because the amount of traffic from each lane is relatively considered. Although at any given time the algorithm is capable of handling different possibilities of traffic densities on each lane. On the basis of relativity between the traffic at different lanes, it decides the class and green signal timer of that particular lane.

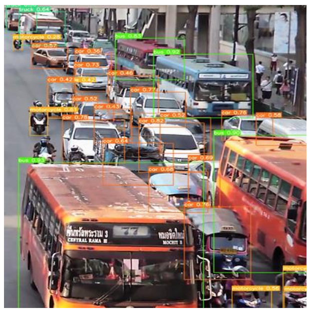
5. RESULTS

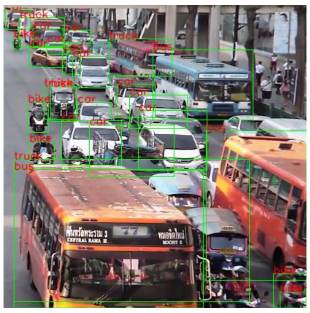
The main purpose of the proposed work is to choose an optimal and fast solution for the problem of traffic congestion. In order to design a dynamic and smart traffic control system, the method of deep learning is selected. Two models are design one using the YOLO version 3 with 23 layers and another using Yolo version 5 having 24 layers. Then trained with the MS COCO dataset as mention in section 4. After training the model is tested on different traffic scenarios. Comparison of results obtained by both the model is shown in figure 5. On comparing the results, version 5 has given more accurate and fast results as compared to the version 3 model. The model predicts the number of vehicles in a particular lane as well as the class of each detected vehicle.

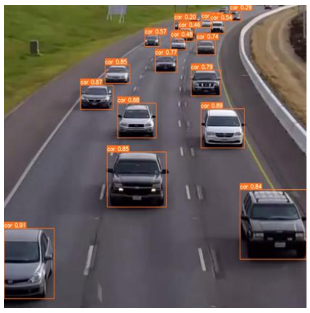
In the figure 5 below, different scenarios are demonstrated, such as low, medium and high densities of vehicles. The detected vehicles are shown inside the bounding boxes and the classes are also mentioned on top of the bounding boxes like motorbike, car, truck and bus.


a. YOLO V3 - 1


b. YOLO V5 - 1


c. YOLO V3 - 2


d. YOLO V5 – 2


e. YOLO V3 - 3

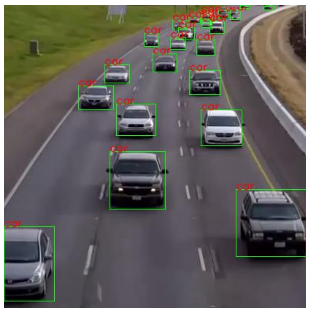
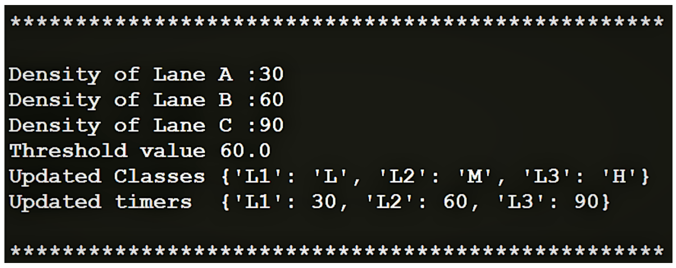

f. YOLO V5 - 3

Fig -5: Comparison of different traffic scenes with help YOLO version 3 and 5.

Following are some outputs from the algorithm when passed different values of vehicle count from each lane which are in stop state i.e., under red signa. Considering the practical condition of traffic, the algorithm can handle some special cases. The updated timer shown in the figure is a green signal timer.



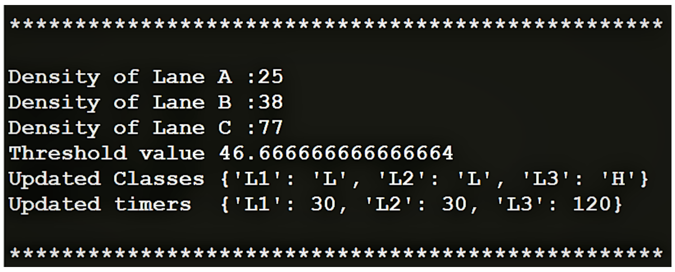
```
*************************************************

Density of Lane A :30
Density of Lane B :60
Density of Lane C :90
Threshold value 60.0
Updated Classes {'L1': 'L', 'L2': 'M', 'L3': 'H'}
Updated timers  {'L1': 30, 'L2': 60, 'L3': 90}

*************************************************
```

Fig - 6: Timer Algorithm results on lane densities of Low, Medium and High.

In the above case (figure 6) where lane D is currently running. The other three lanes remain blocked. With varying vehicle counts from each lane, the timer set by the algorithm for each lane was 30, 60, and 90 seconds each.



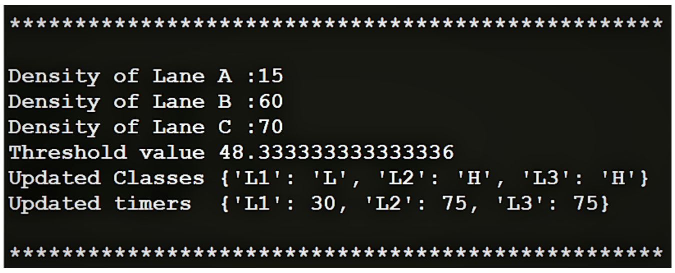
```
*************************************************

Density of Lane A :25
Density of Lane B :38
Density of Lane C :77
Threshold value 46.666666666666664
Updated Classes {'L1': 'L', 'L2': 'L', 'L3': 'H'}
Updated timers  {'L1': 30, 'L2': 30, 'L3': 120}

*************************************************
```

Fig - 7: Timer Algorithm on densities of Low, Low and High.

In the above case in figure 7, considering the opened land as D and other lanes currently on hold. The input vehicle densities were categorized in classes of Low, Low, and High and the resultant times for the next opening were 30, 30, and 120 seconds respectively.



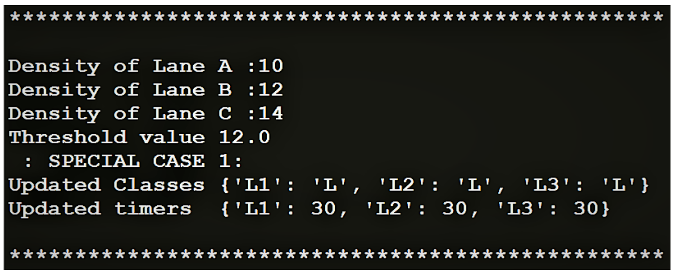
```
*************************************************

Density of Lane A :15
Density of Lane B :60
Density of Lane C :70
Threshold value 48.333333333333336
Updated Classes {'L1': 'L', 'L2': 'H', 'L3': 'H'}
Updated timers  {'L1': 30, 'L2': 75, 'L3': 75}

*************************************************
```

Fig - 8: Timer Algorithm results in scenario where one road has very less traffic.

As seen in figure 8, the inputs to the timer algorithm were passes in such a way that one road among the three had very less traffic as compared to other two. Here the algorithm accordingly identifies lane B and C with densities 60 and 70 is pretty close and thus assigns both with high class while the first lane gets low class assigned. Thus, the times of 30, 75 and 75 respectively.



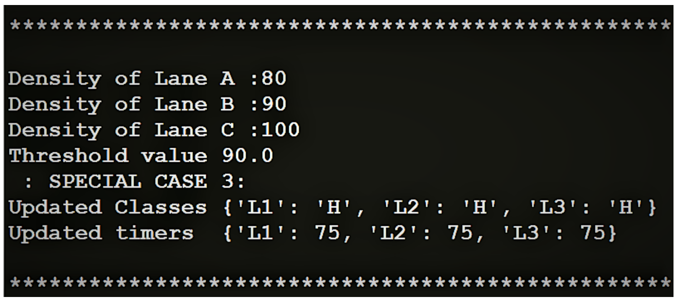
```
*************************************************

Density of Lane A :10
Density of Lane B :12
Density of Lane C :14
Threshold value 12.0
 : SPECIAL CASE 1:
Updated Classes {'L1': 'L', 'L2': 'L', 'L3': 'L'}
Updated timers  {'L1': 30, 'L2': 30, 'L3': 30}

*************************************************
```

Fig - 9: Timer Algorithm Special Case 1

In the special case where all the road densities come to be in low class (figure 9). The algorithm reduces the timer for all lanes irrespective of the threshold value. While calculating the time relativity with varying vehicle counts from each lane the timer set by the algorithm for each lane was 30, 30, and 30 seconds each.



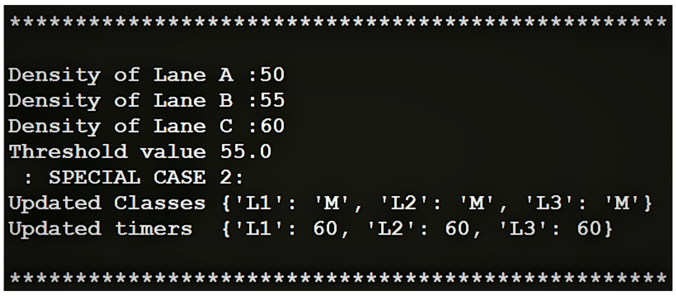
```
****************************************************

Density of Lane A :80
Density of Lane B :90
Density of Lane C :100
Threshold value 90.0
 : SPECIAL CASE 3:
Updated Classes {'L1': 'H', 'L2': 'H', 'L3': 'H'}
Updated timers  {'L1': 75, 'L2': 75, 'L3': 75}

****************************************************
```

Fig - 10: Timer Algorithm Special Case 2

As shown in above figure 10, the next special condition when the density of all lanes is very high causing high traffic congestion at intersections the algorithm increases the timer of all lanes with a minimal extension to the static timer so that the rotation of the green signal is fast. With varying vehicle counts from each lane, the timer set by the algorithm for each lane were 75, 75, and 75 seconds each.



```
****************************************************

Density of Lane A :50
Density of Lane B :55
Density of Lane C :60
Threshold value 55.0
 : SPECIAL CASE 2:
Updated Classes {'L1': 'M', 'L2': 'M', 'L3': 'M'}
Updated timers  {'L1': 60, 'L2': 60, 'L3': 60}

****************************************************
```

Fig - 4: Timer Algorithm Special Case 3

In this next special case where lane D is currently running and other lanes have been blocked. When the density of all lanes is relatively the same and does not require any severe changes in the predefined timer. With varying vehicle counts from each lane the timer set by the algorithm for each lane were 60, 60 and 60 seconds each.

6. CONCLUSIONS

In this article, a smart traffic control system is proposed which can be widely used in a smart city application for optimal traffic flow of vehicles. For smooth flow of traffic, it is necessary to handle traffic congestion properly. The system takes the real-time traffic data, captured from the pre-installed cameras at the intersection as input for the object detection phase and calculates the number of vehicles for density estimation. These densities are given to the timer algorithm as input for manipulating the green signal timer of the consecutive lanes. The timer algorithms are capable of handling multiple scenarios based on the practical condition which removes the flaws of the traditional static timer. The main advantage of the proposed method is the use of video processing over sensors which reduces the setup cost, low maintenance, and relatively more durability and accuracy.

REFERENCES

[1] Jose Melo and Andrew Naftel, "Detection and Classification of Highway Lanes Using Vehicle Motion Trajectories", IEEE transaction on intelligent transportation system, Vol.7, No.2, June 2006.

[2] Alvin Abdagic, Omer Tanovic, Abdulah Aksamovic and Senad Huseinbegovic "Counting traffic using optimal flow algorithm on video footage of a complex crossroad", IEEE Elmar 2010, Oct 2010.

[3] Prashant Jadhav, Pratiksha Kelkar,Kunal Patil and Snehal Thorat, "Smart Traffic Control System Using Image Processing", International research journal of engineering and technology(IRJET), March 2016.

[4] Surojit Dey and Mirzanur Rahman "Application of Image Processing and Data Mining Techniques for Traffic Density Estimation and Prediction", ICACCP, 2019.

[5] Zhe Dai, Huansheng Song, Xuan Wang, Yong Fang and Xu Yun, "Video-Based Vehicle Counting Framework", IEEE, May 2019.

[6] JingweiCao, Chunxue Song, Shixin Song, Silun Peng,DaWang, Yulong Shao and Feng Xiao, "Front Vehicle Detection Algorithm for Smart Car Based on improved SSD Model", Sensors, Aug 2020.

[7] Alexey Bochkovskiy, Chien-Yao Wang and Hong-Yuan Mark Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection", ARXIV, Apr 2020.

[8] A.A. Carter, J.W. Hess, E.A.Hodgkins and J. Raus, "Traffic Surveillance and Control Research", IEEE Vol.56, No.4, 1968.

[9] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick and Piotr Dollár "Microsoft COCO: Common Objects in Context", ARXIV, Feb 2015.

[10] Joseph Redmon and Ali Farhadi, "YOLOv3: An Incremental Improvement". CoRR abs/1804.02767. (2018).

[11] Glenn Jocher, "YOLO version 5", unpublished.