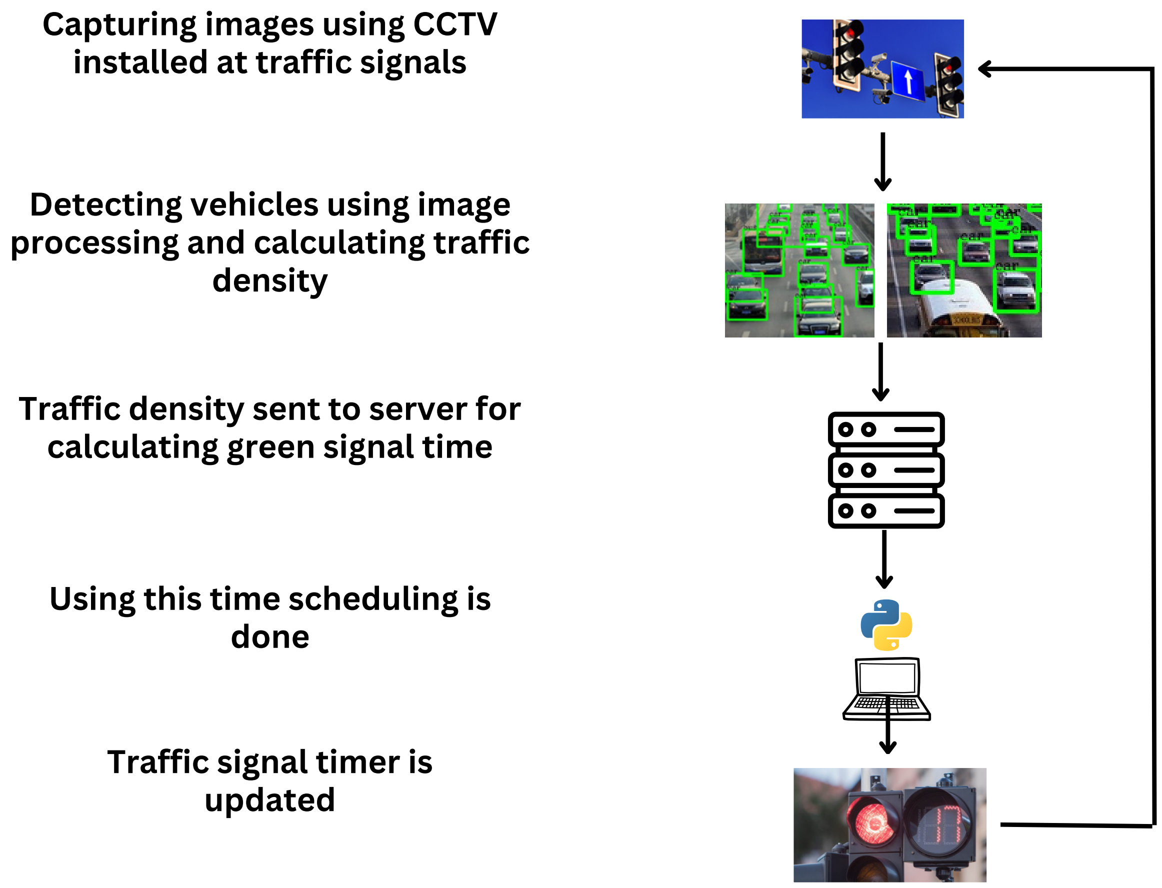
Traffic Flow Optimization And Congestion Management

Implementation Details

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. This system can be broken down into 3 modules: Vehicle Detection module, Signal Switching Algorithm, and Simulation module. As shown in the figure below, 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.


Capturing images using CCTV installed at traffic signals
Detecting vehicles using image processing and calculating traffic density
Traffic density sent to server for calculating green signal time
Using this time scheduling is done
Traffic signal timer is updated

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.

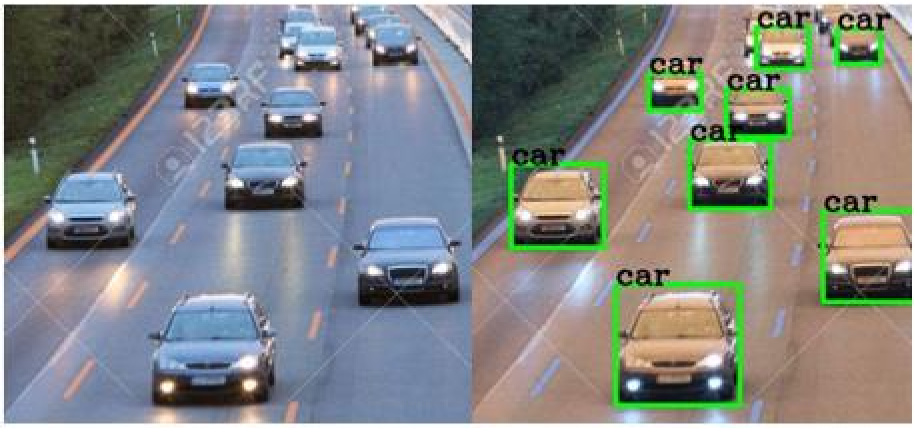
-The dataset for training the model was prepared by scraping images from google and labelling them manually using LabelIMG, a graphical image annotation tool.

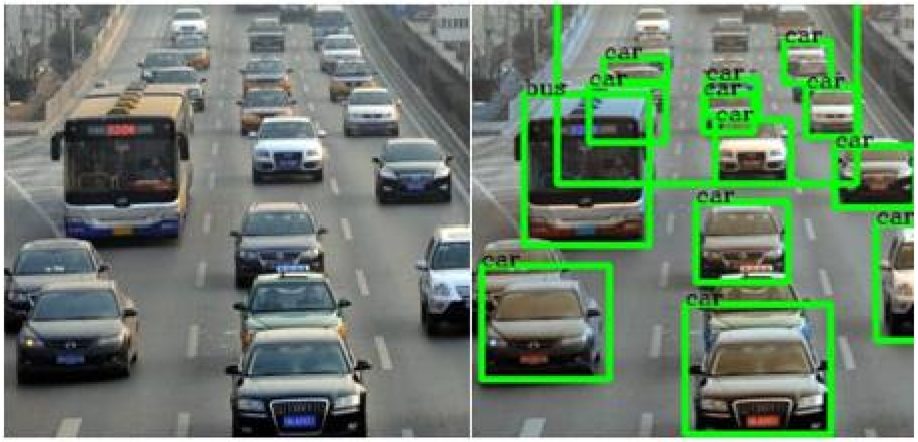
-Then the model was trained using the pre-trained weights downloaded from the YOLO website. The configuration of the .cfg file used for training was changed in accordance with the specifications of our model. The number of output neurons in the last layer was set equal to the number of classes the model 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.

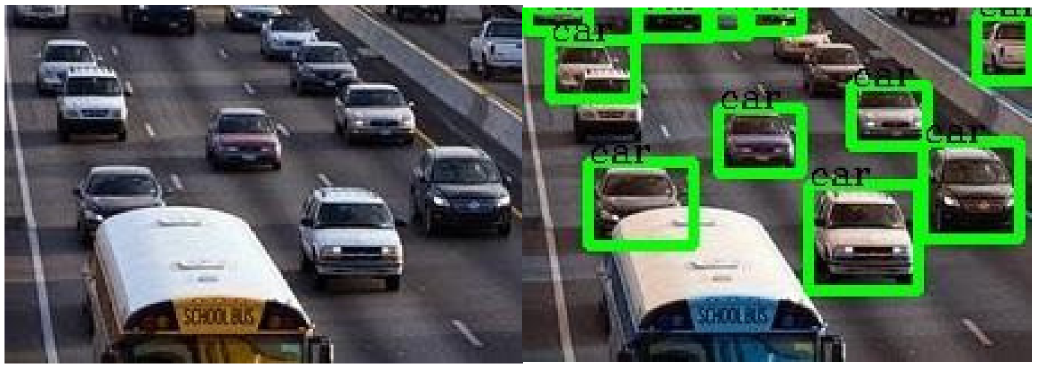
-After making these configuration changes, the model was trained until the loss was significantly less and no longer seemed to reduce. This marked the end of the training, and the weights were now updated according to our requirements.

-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.

Following are some images of the output of the Vehicle Detection Module:







Signal Switching Algorithm

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

Working of the algorithm
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 0 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 0 seconds. This gives the system a total of 5 seconds (equal to value of yellow signal timer) 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. The green signal time is then calculated using the formula below.

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

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.

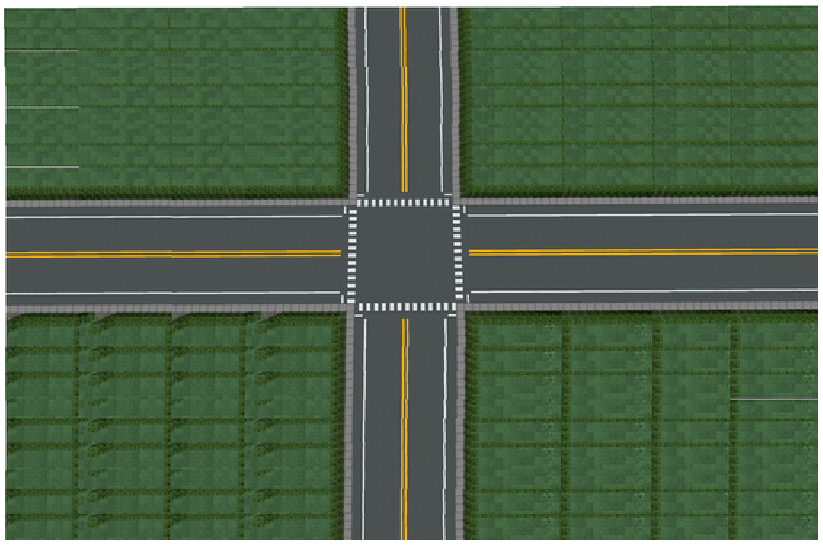
Order of signals: Red → Green → Yellow → Red

Simulation Module

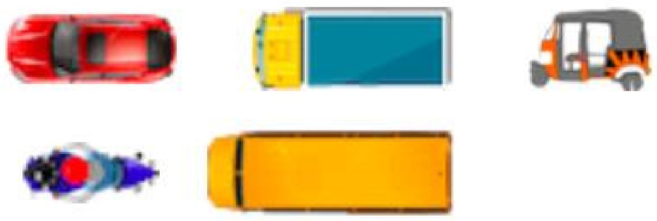
A simulation was developed from scratch using Pygame to simulate real-life traffic. It assists in visualizing the system and comparing it with the existing static system. It contains a 4-way intersection with 4 traffic signals. Each signal has a timer on top of it, which shows the time remaining for 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

Key steps in development of
simulation

1. Took an image of a 4-way intersection as background.

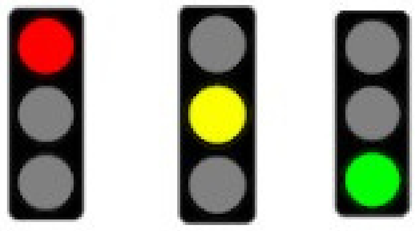


2. Gathered top-view images of car, bike, bus, truck, and
3. rickshaw. Resized them.



4. Rotated them for display along different directions.

5. Gathered images of traffic signals - red, yellow, and green.



6. Code: For rendering the appropriate image of the signal depending on whether it is red, green, or yellow.

7. Code: For displaying the current signal time i.e. the time left for a green signal to turn yellow or a red signal to turn green or a yellow signal to turn red. The green time of the signals is set according to the algorithm, by taking into consideration the number of vehicles at the signal. The red signal times of the other signals are updated accordingly.

8. Generation of vehicles according to direction, lane, vehicle class, and whether it will turn or not all set by random variables. Distribution of vehicles among the 4 directions can be controlled. A new vehicle is generated and added to the simulation after every 0.32 seconds.

9. Code: For how the vehicles move, each class of vehicle has different speed, there is a gap between 2 vehicles, if a car is following a bus, then its speed is reduced so that id does not crash into the bus.

10. Code: For how they react to traffic signals i.e. stop for yellow and red, move for green. If they have passed the stop line, then continue to move if the signal turns yellow.

11. Code: For displaying the number of vehicles that have crossed the signal.

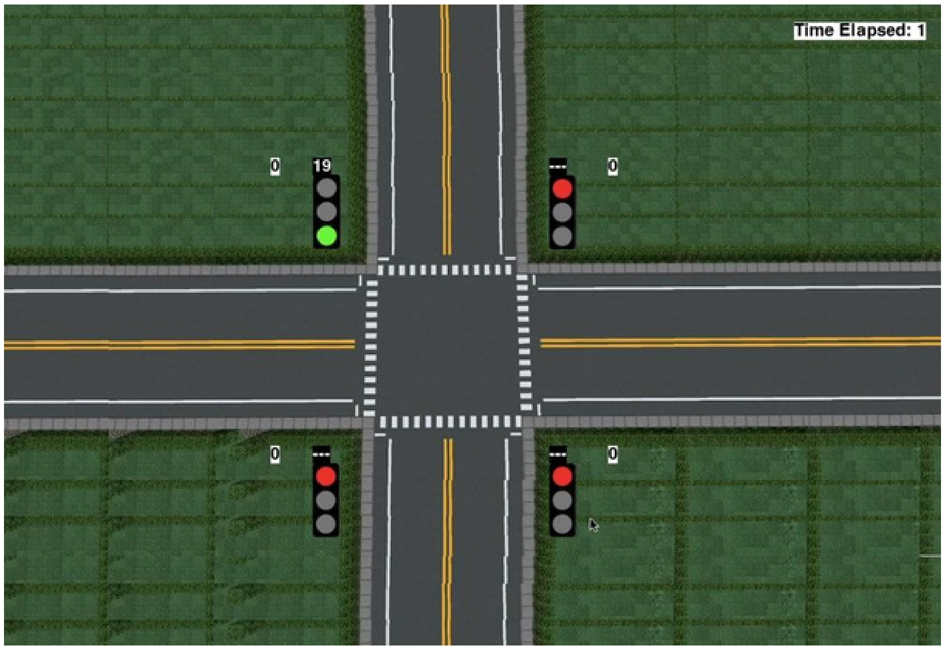
12. Code: For displaying the time elapsed since the start of the simulation.

13. Code: For updating the time elapsed as simulation progresses and exiting when the time elapsed equals the desired simulation time, then printing the data that will be used for comparison and analysis.

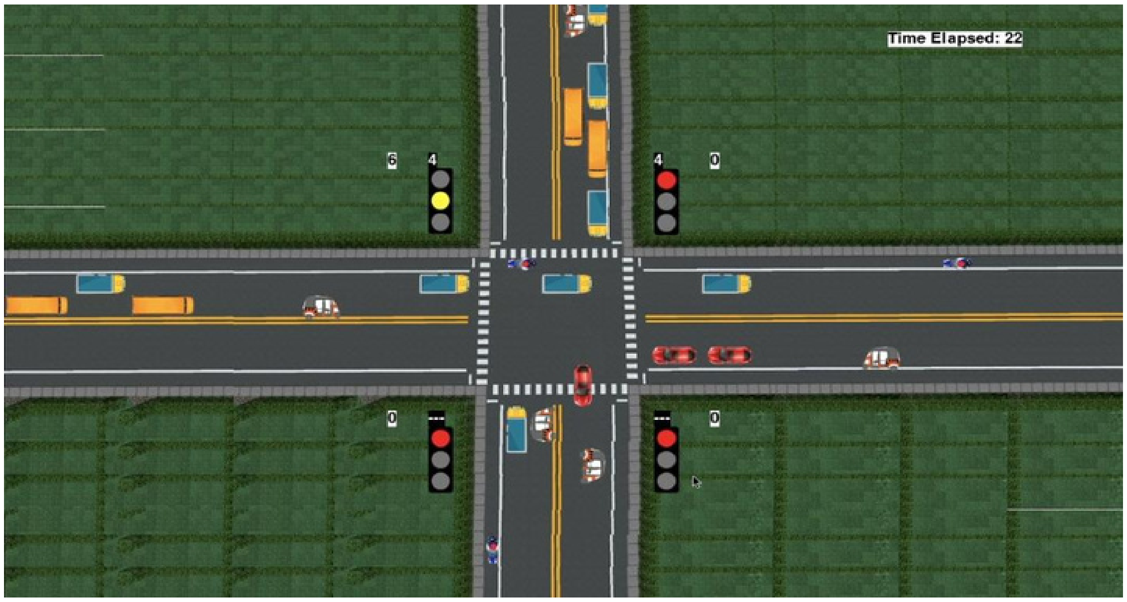
14. To make the simulation closer to reality, even though there are just 2 lanes in the image, we add another lane to the left of this which has only bikes, which is generally the case in many cities.

15. Vehicles turning and crossing the intersection in the simulation to make it more realistic.

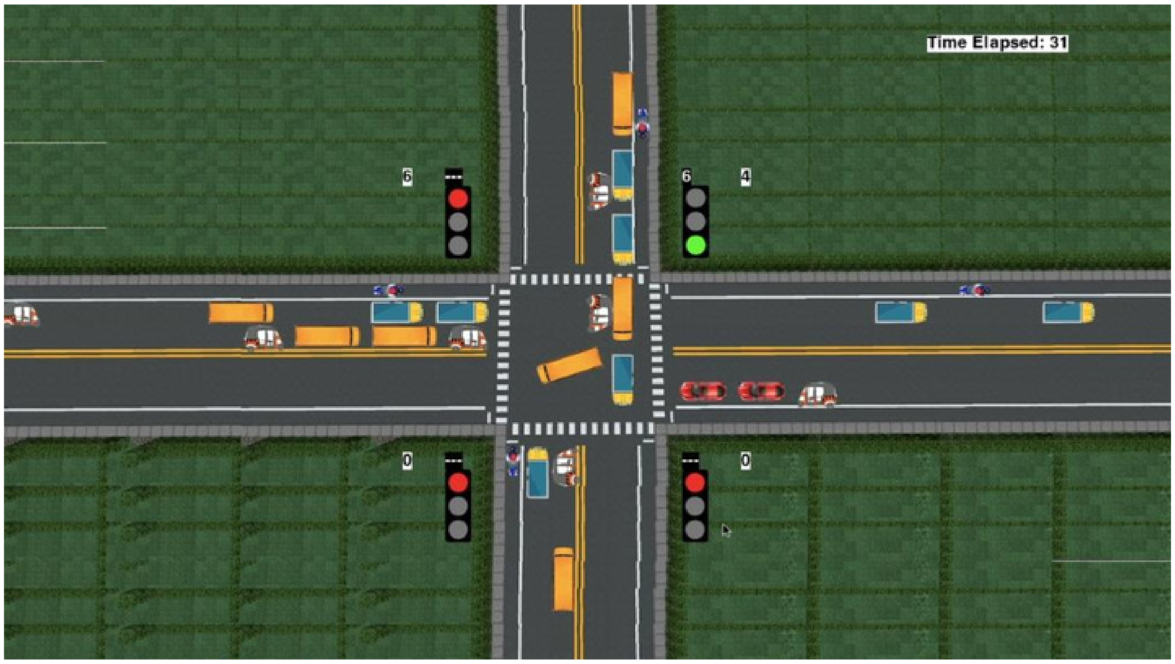
Following are some images of the demo simulation:



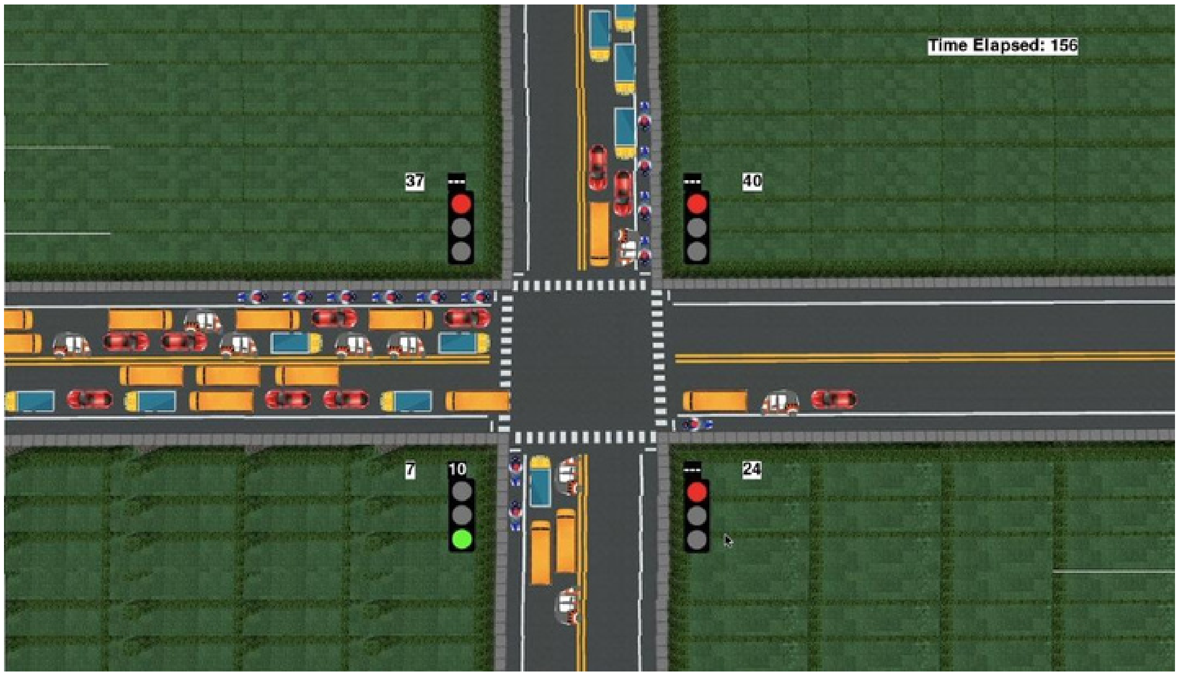
(i): Simulation just after start showing red and green lights, green signal time counting down from a default of 20 and red time of next signal blank. When the signal is red, we display a blank value till it reaches 10 seconds. The number of vehicles that have crossed can be seen beside the signal, which are all 0 initially. The time elapsed since the start of simulation can be seen on top right.



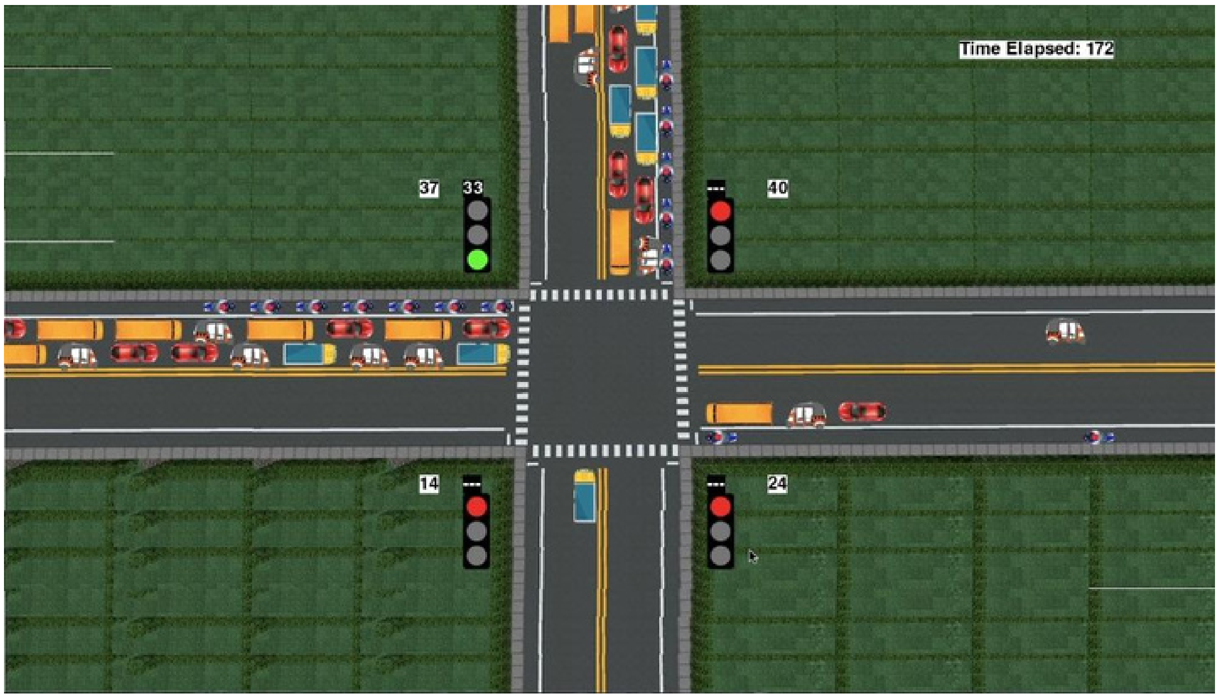
(ii): Simulation showing yellow light and red time for next signal. When red signal time is less than 10 seconds, we show the countdown timer so that vehicles can start up and be prepared to move once the signal turns green.



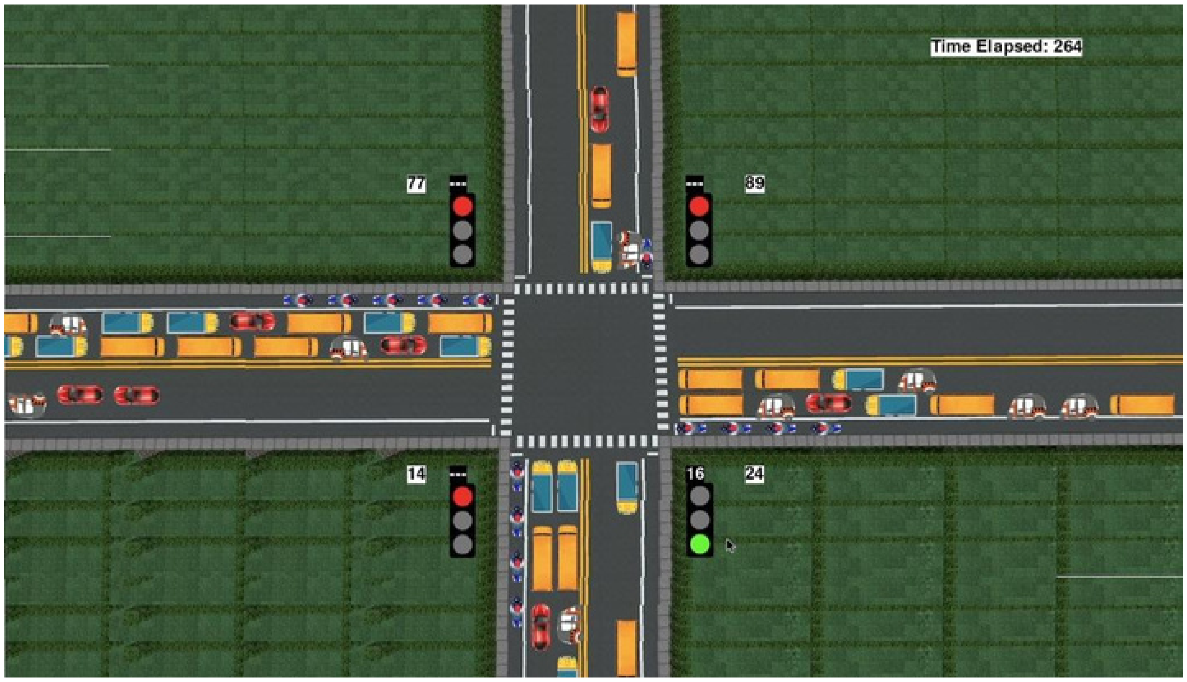
(iii): Simulation showing vehicles turning



(iv): Simulation showing green time of signal for vehicles moving up set to 10 seconds according to the vehicles in that direction. As we can see, the number of vehicles is quite less here as compared to the other lanes. With the current static system, the green signal time would have been the same for all signals, like 30 seconds. But in this situation, most of this time would have been wasted. But our adaptive system detects that there are only a few vehicles, and sets the green time accordingly, which is 10 seconds in this case.



(v): Simulation showing green time of signal for vehicles moving right set to 33 seconds according to the vehicles in that direction.

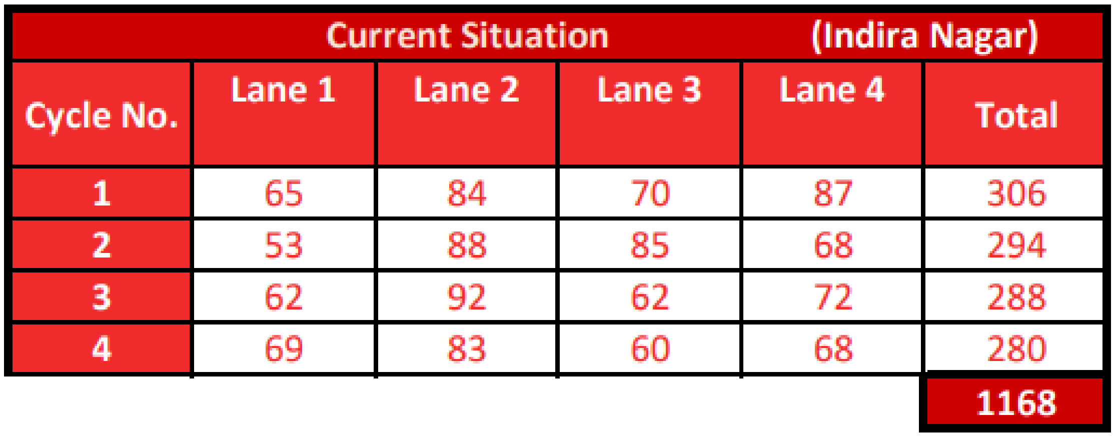


(vi): Simulation showing green time of signal for vehicles moving left set to 24 seconds according to the vehicles in that direction.

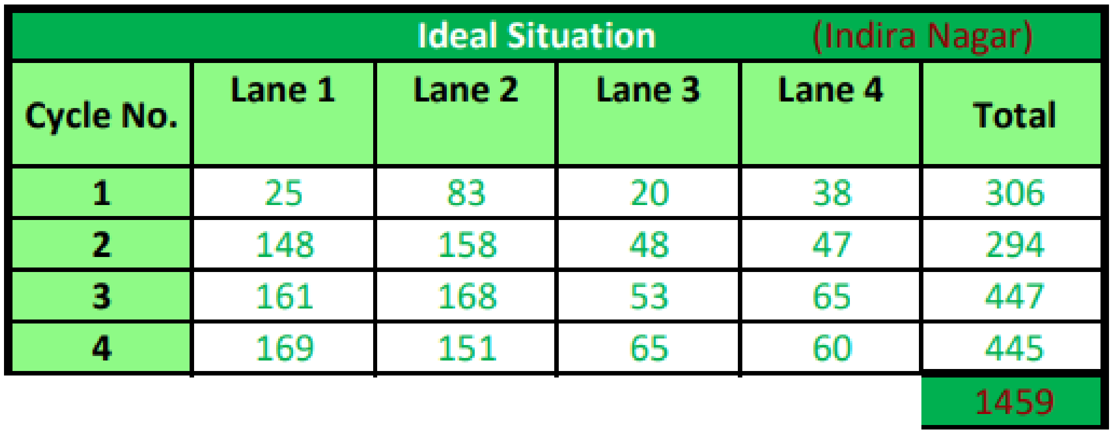
Results :

We compare the total number of vehicles that cross the intersection over a period of 12 minutes in the current system and the proposed adaptive system(Ideal Situation) with the same distribution of traffic.

The results obtained are tabulated in the form of number of vehicles lane-wise and the total number of vehicles passed.



Current Situation			(Indira Nagar)		
Cycle No.	Lane 1	Lane 2	Lane 3	Lane 4	Total
1	65	84	70	87	306
2	53	88	85	68	294
3	62	92	62	72	288
4	69	83	60	68	280
					1168



Ideal Situation			(Indira Nagar)		
Cycle No.	Lane 1	Lane 2	Lane 3	Lane 4	Total
1	25	83	20	38	306
2	148	158	48	47	294
3	161	168	53	65	447
4	169	151	65	60	445
					1459

As we can see, with all conditions alike, the adaptive system was able to pass 1459 vehicles while the current static system could pass only 1168 vehicles in 12 minutes, which means 291 more vehicles.
Thus, the proposed adaptive system improves the performance by over 20% (approx.).

This implies a reduction in idle green signal time (signal is green but no vehicle passes) as well as the waiting time of the vehicles.

Conclusion

Thus, 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 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 the fuel consumption and pollution.

According to simulation results, the system shows about 20% improvement over the current system in terms of the number of vehicles crossing the intersection, which is a significant improvement. This system can thus be integrated with the CCTV cameras in major cities in order to facilitate better management of traffic.