HACKATHON: INOVATE OR EVAPORATE

Problem Statement- 20 Design a system to optimize traffic flow and reduce congestion in urban areas.

Team Members

Ayushi Rajput

Chetan Pandey

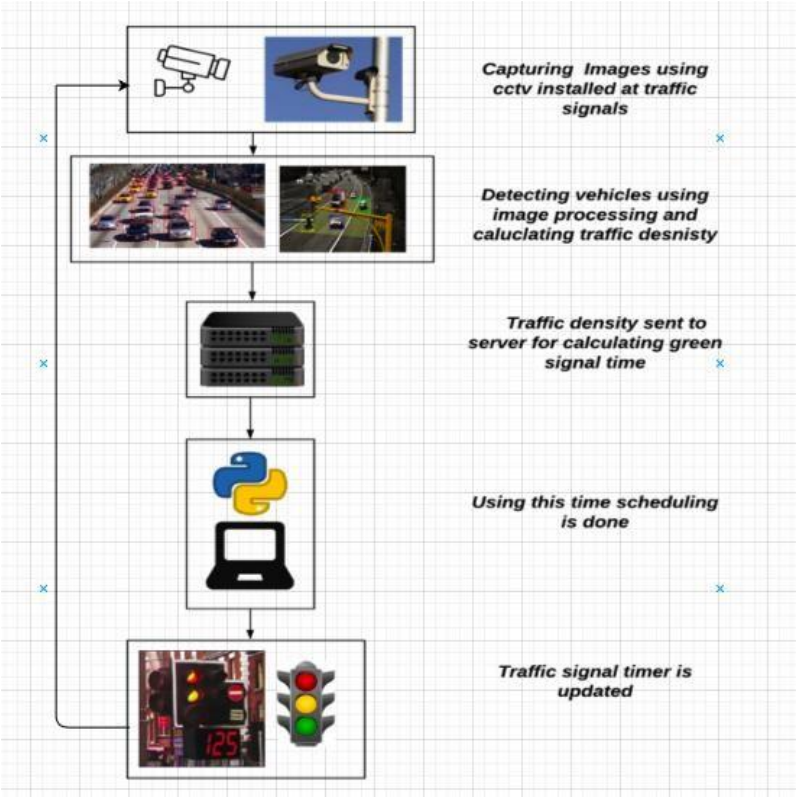
Deepak Kumar

Shwet Raj

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 to avoid congestion of a particular lane. A simulation is also developed to demonstrate the system's effectiveness and compare it with the existing static system.



Vehicle Detection Module

YOLO (You Only Look Once) is used for vehicle detection due to its balance between accuracy and speed.

Model Training: A custom YOLO model is trained to identify specific vehicle classes (cars, bikes, heavy vehicles, rickshaws).

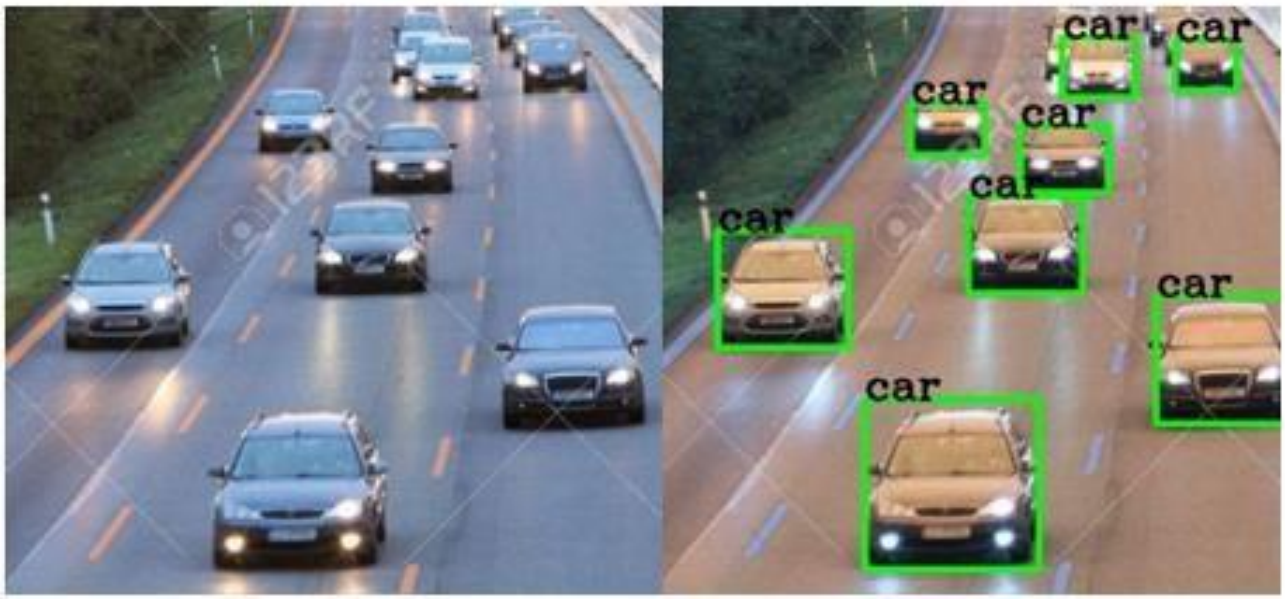
Data Collection: Training data is created by collecting images from Google and manually labelling them using a tool called LabelIMG.

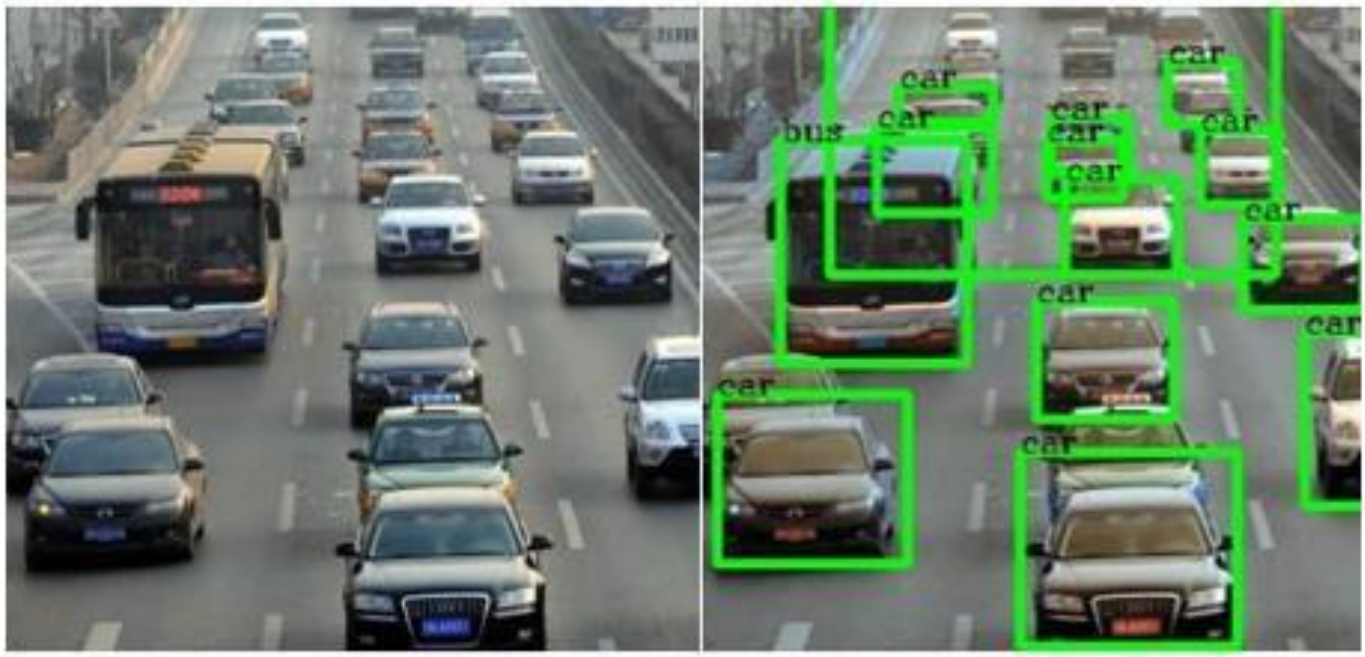
Model Configuration: A pre-trained YOLO model is fine-tuned by adjusting the configuration file (.cfg) to match the desired number of classes (4 in this case) and filters.

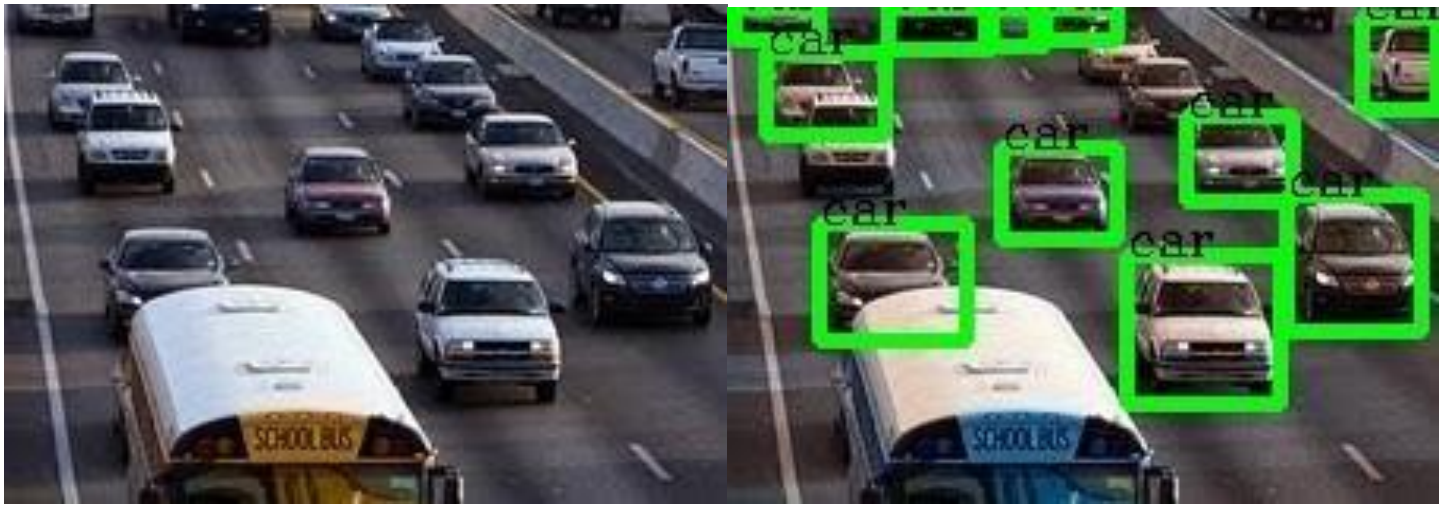
Training Process: The model is trained until it achieves a low loss value, indicating it has learned effectively.

Detection and Output: The trained model is used with OpenCV to detect vehicles in images. Detected vehicles are identified by class along with confidence score and bounding box coordinates (location in the image). OpenCV can also be used to visually display these bounding boxes on the image.

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




car
bus
car
car
car
car
car
car
car
car
car
car


SCHOOL BUS
car
car
car
car
car
car
SCHOOL BUS

Signal Switching Algorithm

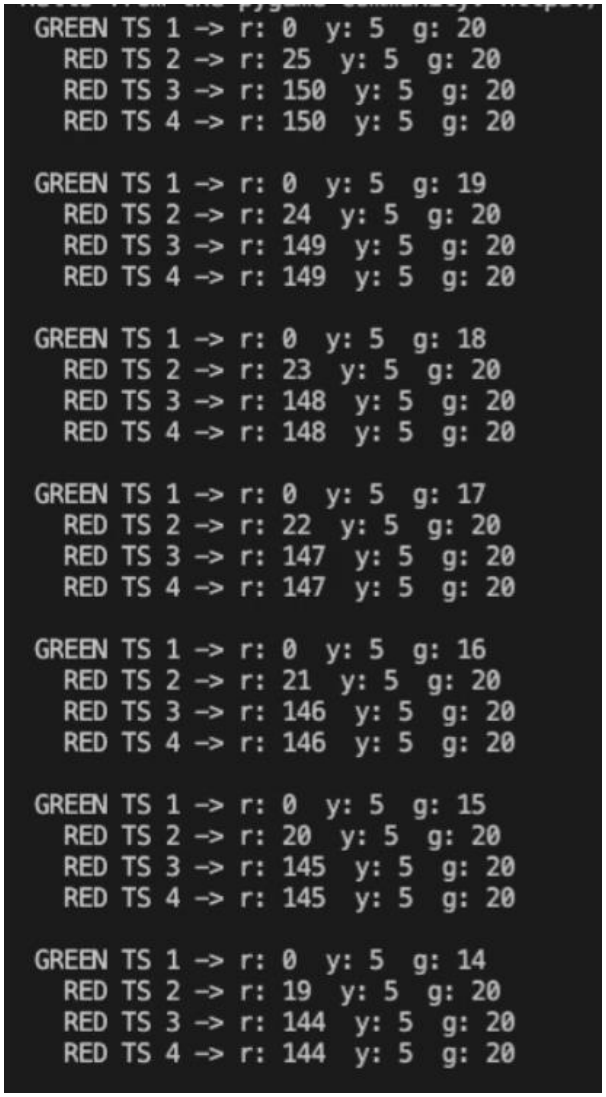
The Signal Switching Algorithm sets the green signal timer according to traffic density returned by the vehicle detection module and updates the red signal timers of other signals accordingly. It also switches between the signals cyclically according to the timers.

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 congestion.

Following are some images of the output of the Signal Switching Algorithm:

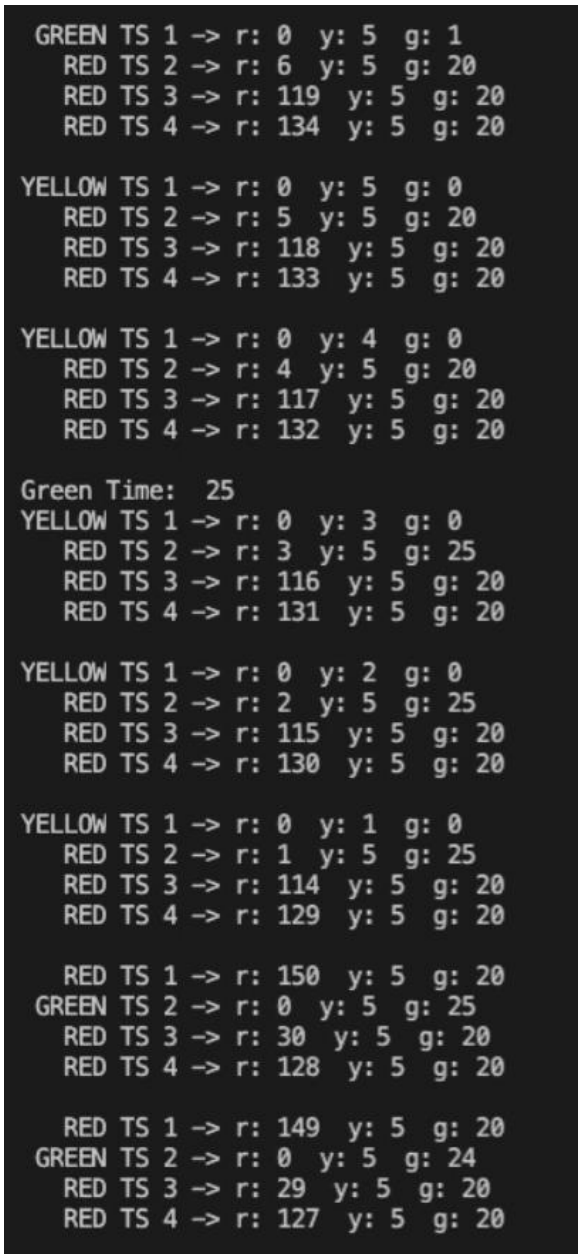


(i):
Initially, all signals are loaded with default values, only the red signal time of the second signal is set according to green time and yellow time of first signal.

(ii):

The leftmost column shows the status of the signal i.e. red, yellow, or green, followed by the traffic signal number, and the current red, yellow, and green timers of the signal. Here, traffic signal 1 i.e. TS 1 changes from green to yellow. As the yellow timer counts down, the results of the vehicle detection algorithm are calculated, and a green time of 9 seconds is returned for TS 2. As this value is less than the minimum green time of 10, the green signal time of TS 2 is set to 10 seconds. When the yellow time of TS 1 reaches 0, TS 1 turns

red and TS 2 turns green, and the countdown continues. The red signal time of TS 3 is also updated as the sum of yellow and green times of TS 2 which is 5+10=15.



(iii):

After a complete cycle, again, TS 1 changes from green to yellow. As the yellow timer counts down, the results of the vehicle detection algorithm are processed, and a green time of 25 seconds is calculated for TS 2. As this value is more than the minimum green

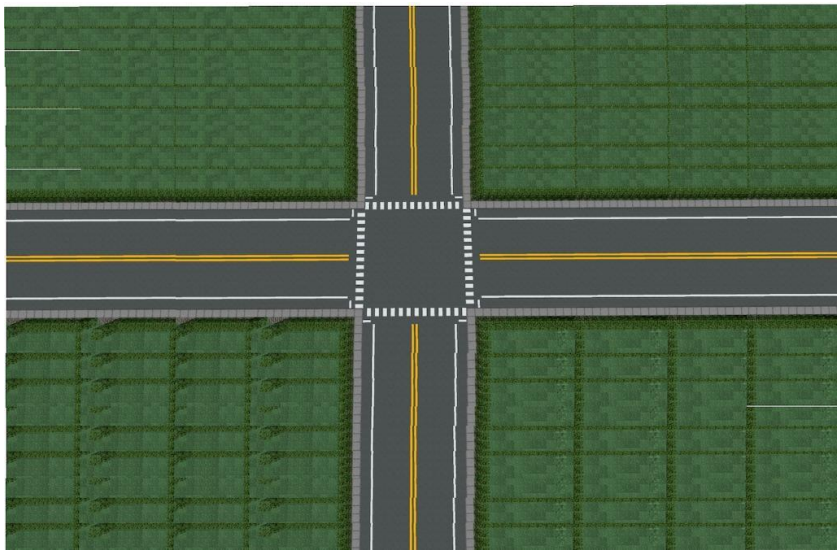
time and less than maximum green time, the green signal time of TS 2 is set to 25 seconds. When the yellow time of TS 1 reaches 0, TS 1 turns red and TS 2 turns green, and the countdown continues. The red signal time of TS 3 is also updated as the sum of yellow and green times of TS 2 which is 5+25=30.

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. 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 rickshaw.

3. 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 0.75 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 it 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.

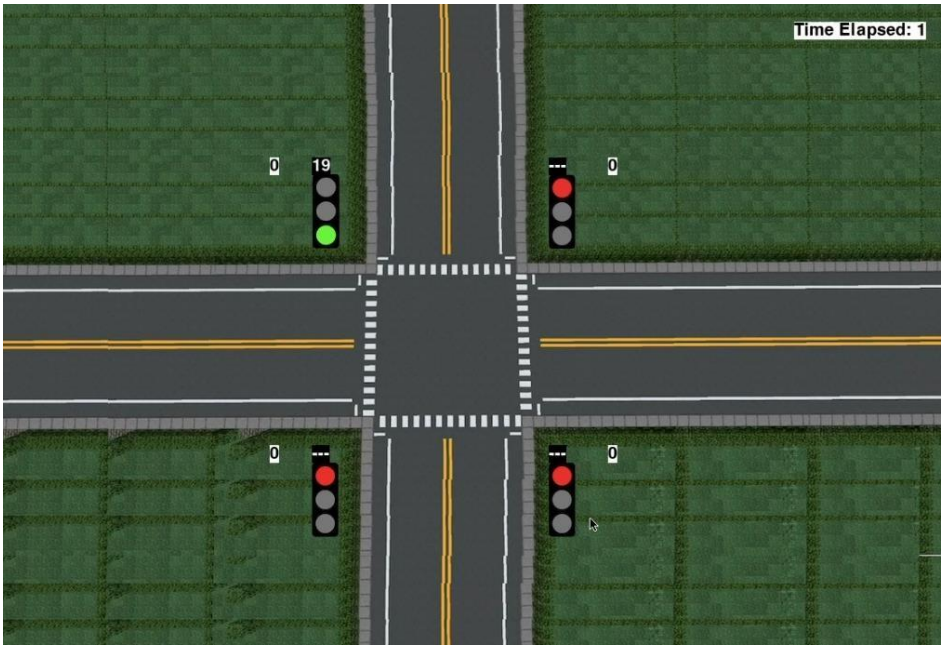
12. Code: For displaying the time elapsed since the start of the simulation.

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 final simulation:

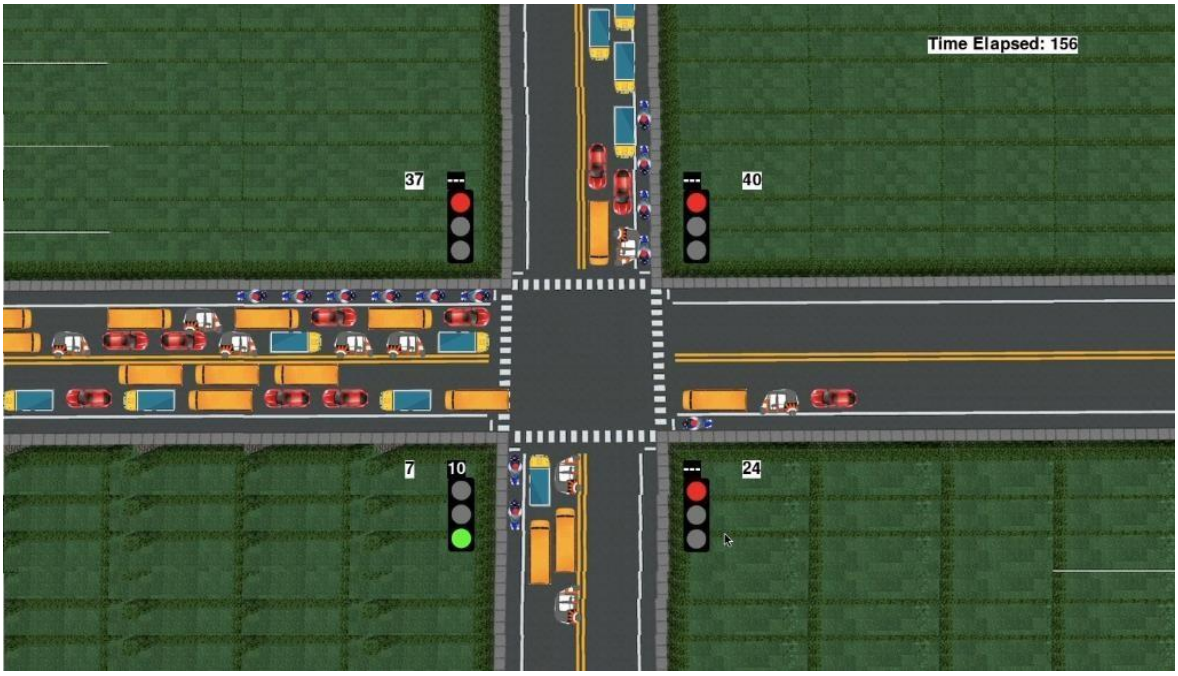


(i): Simulation just after start showing red and green lights, green signal time counting down from a default of 20 and red time of next signal blank. When the signal is red, we display a blank

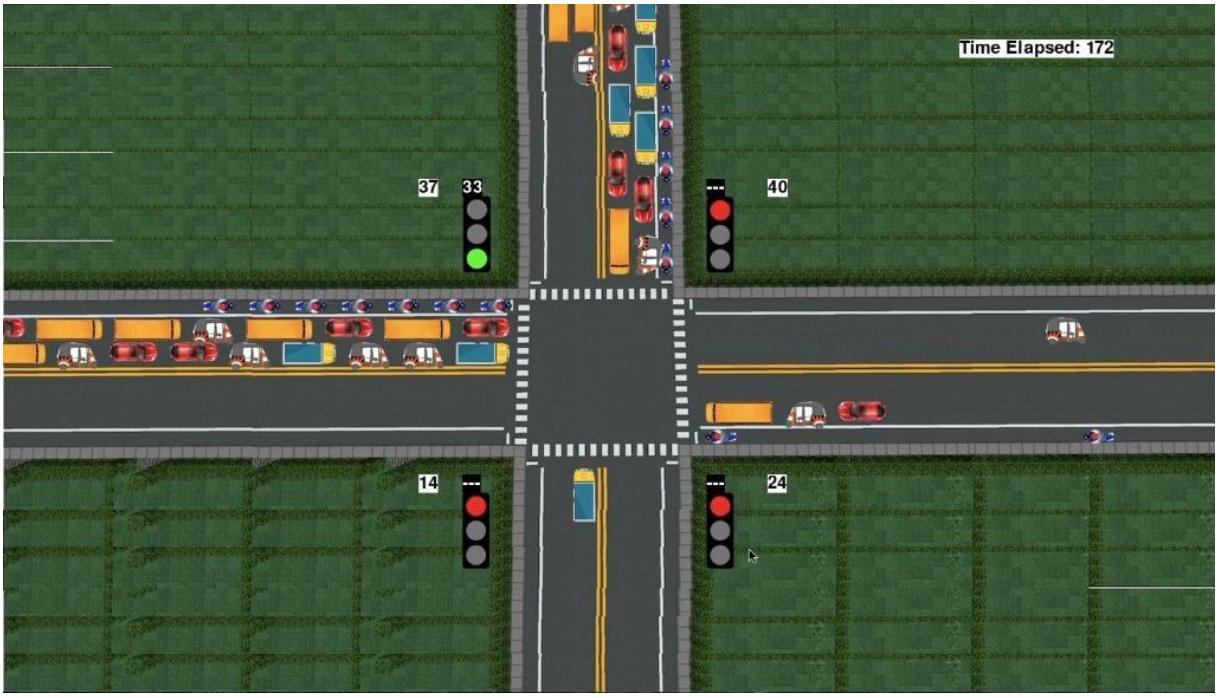
value till it reaches 10 seconds. The number of vehicles that have crossed can be seen beside the signal, which are all 0 initially. The time elapsed since the start of simulation can be seen on top right.

(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.

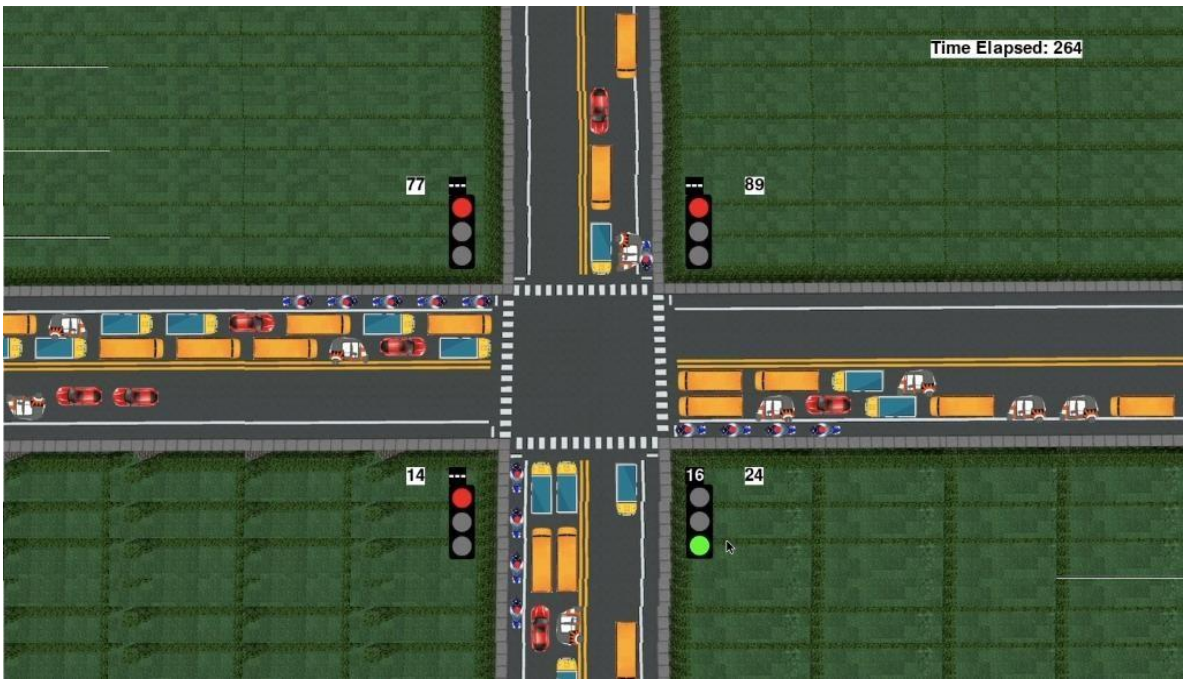
(iii): Simulation showing vehicles turning.



(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.

Ideas for Further Implementation

- **Adjusting traffic wait timer based on Traffic Density**

  Traffic congestion is a persistent challenge in urban areas, especially during peak hours. The existing traffic control systems often rely on fixed-time signal timers, leading to unnecessary delays even when traffic density is low. To address this issue, we propose an intelligent and adaptive traffic signal timer control system.

  Objectives:

  1. Continuous Adaptation: Our system aims to continuously adjust traffic signal timers based on real-time traffic density variations.
  2. Reduced Congestion: By dynamically modifying signal timings, we intend to significantly reduce traffic congestion levels.

- **Vehicle speed detection**

  Vehicle Speed Detection and Control System. This system aims to monitor vehicle speeds and, if necessary, regulate them to enhance road safety. Below, I'll provide an abstract outlining the objectives, methodology, and significance of such a system.

  Objectives:

1. Enhanced Road Safety: The primary goal is to reduce accidents caused by speeding vehicles.
2. Efficient Traffic Management: By controlling vehicle speeds, we contribute to smoother traffic flow.

- **Traffic Rule Violation Detection**

  Detecting traffic rule violations is crucial for road safety and efficient traffic management. Here's an abstract you can include in your presentation regarding the Traffic Rule Violation Detection System

Objectives:

Enhanced Road Safety:

1. Detect and penalize traffic violations promptly to reduce accidents caused by non-compliance with traffic rules.
2. Improve overall safety for pedestrians, cyclists, and other road users.

Efficient Traffic Management:

1. Optimize traffic flow by identifying and addressing rule violations.
2. Minimize congestion and ensure smoother movement of vehicles.

Automated Enforcement:

1. Shift from manual enforcement to automated systems for consistent and real-time monitoring.
2. Reduce reliance on human patrols and enhance efficiency.

Violation Notification:

1. Notify offenders immediately when a violation occurs.
2. Use visual displays, alarms, or communication channels to inform drivers.

Statistical Analysis:

1. Collect data on violations for statistical analysis.
2. Identify patterns, high-risk areas, and trends to inform policy decisions.