Rush Hour

Major Project Report

Submitted By:

Deepak Sharma	19103003
Rahul Jindal	19103033
Mayank	19103042
Raman Ailawadhi	19103061

Under the Supervision of:

Prof. Sudesh Rani
Faculty

Department of Computer Science and Engineering

Punjab Engineering College

(Deemed to be University)

Chandigarh

PEC
EXPLORE INNOVATE EXCEL

DECLARATION

We hereby declare that the project work entitled “Rush Hour” is an authentic record of our own work carried out at Punjab Engineering College (Deemed to be University), as a requirement of Major Project for the award of degree of BTech (Computer Science and Engineering), under the guidance of Prof. Sudesh Rani (Faculty, Department of Computer Science and Engineering) during August to December 2022.

Deepak Sharma	19103003
Rahul Jindal	19103033
Mayank	19103042
Raman Ailawadhi	19103061

ACKNOWLEDGEMENT

We have taken a lot of deliberations on this venture. But it wouldn't have been possible without the help and backing of numerous people. We want to extend our true appreciation and thank them. We take this opportunity to express our profound gratitude and deep regards to our mentor Prof. Sudesh Rani for her exemplary guidance, monitoring and constant encouragement throughout the course of this project.

This project truly wouldn’t have been possible without her mentorship.

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

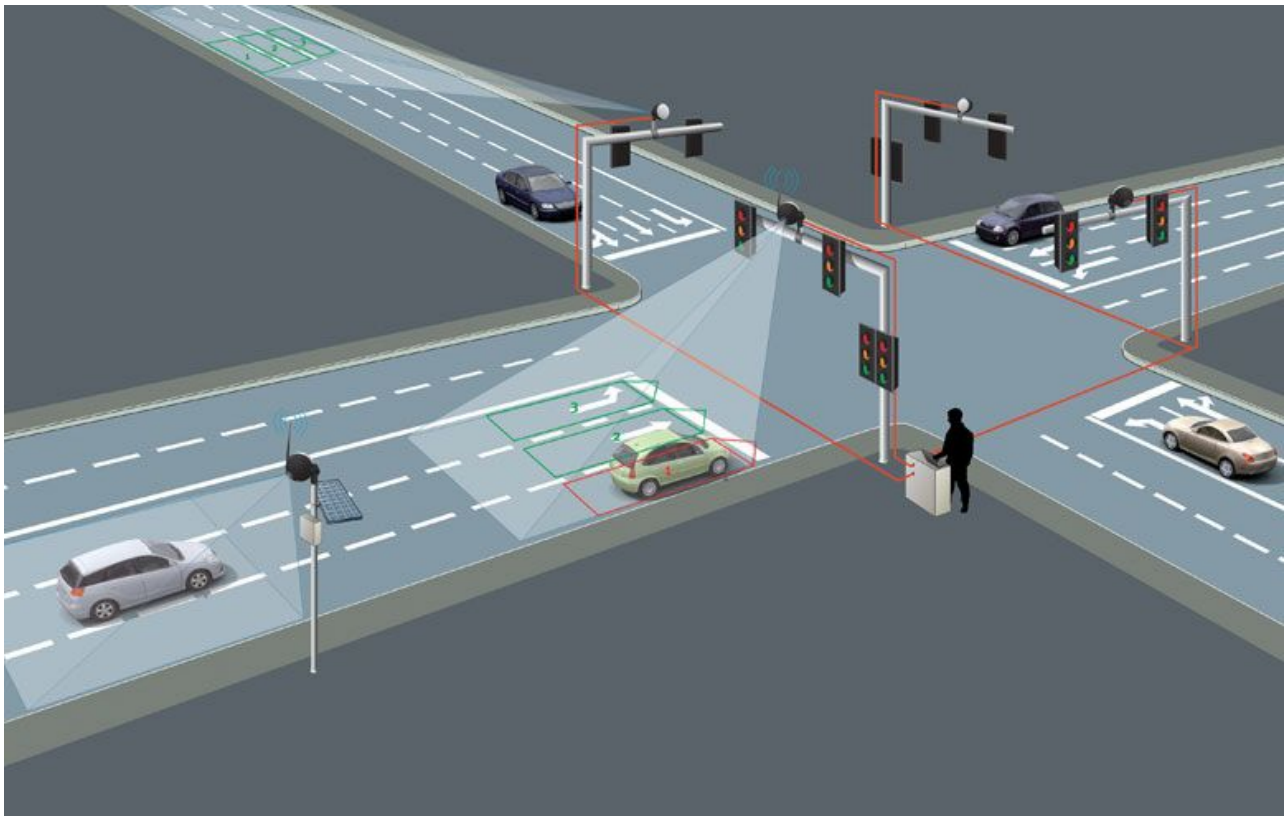

Fig 1: Smart Traffic Management Visualization

TABLE OF CONTENTS

DECLARATION 2

ACKNOWLEDGEMENT 3

ABSTRACT 4

ABBREVIATIONS 7

LIST OF FIGURES 8

CHAPTER 1 : INTRODUCTION 10

1.1 Introduction 10

1.2 ITCS - Improved Traffic Control System 10

1.3 YOLO Model 10

1.4 Pygame Simulator 11

1.5 Web Application 11

CHAPTER 2: PROBLEM STATEMENT 12

2.1 Research 12

2.2 Existing Systems 13

CHAPTER 3: PROPOSED WORK 15

3.1 Vehicle Detection Module 15

3.2 Signal Switching Algorithm 16

3.3 Simulation Module 17

CHAPTER 4: IMPLEMENTATION 18

4.1 Vehicle Detection Module 18

4.2 Signal Switching Algorithm 20

4.3 Simulation Module 22

4.4 Technologies and Tools used 26

CHAPTER 5: RESULTS AND DISCUSSION 32

5.1 Results 32

5.2 Discussion 44

CHAPTER 6: TIMELINE **45**

CHAPTER 7: CONCLUSION AND FUTURE WORK **49**

CHAPTER 8: REFERENCES **50**

ABBREVIATIONS

ATCS	Adaptive Traffic Control System
CCTV	Closed Circuit Television
CNN	Convolutional Neural Network
CSS	Cascading Style Sheets
DOM	Document Object Model
GST	Green Signal Time
HTML	Hypertext Markup Language
I/O	Input Output
ITCS	Improved Traffic Control System
JSON	Javascript Object Notation
SQL	Structured Query Language
YOLO	You Only Look Once

LIST OF FIGURES

Fig. No.	Description	Page No.
1	Smart Traffic Management Visualization	4
2	Workflow	16
3	Traffic Example	19
4	Traffic Detection with YOLO	19
5	Pygame Simulator Background for 3 Lanes	22
6	Pygame Simulator Background for 4 Lanes	22
7	Pygame Simulator Vehicles	23
8	Pygame Simulator Traffic Lights	23
9	Pygame Simulator Initiation	24
10	Simulation showing change of lights	24
11	Simulation showing predicted green timer for a light	25
12	An intermediate state of a crossroad	25
13	Simulation for 3-Way traffic system	26
14	HTML CSS Logo	26
15	JavaScript Logo	27
16	NodeJS Logo	27
17	MongoDB Logo	28
18	React Logo	28
19	Python Logo	29
20	PyGame Logo	30

21	Visual Studio Logo	**30**
22	Jupyter Notebook Logo	**30**
23	3-way Simulation	**32**
24	4-way Simulation	**33**
25	YOLO object detection example	**33**
26	YOLO object detection example	**34**
27	Data points for 3-way System	**35**
28	Data points for 4-way System	**35**
29	Analysis of Data points in 3-way System (Static v/s Dynamic)	**36**
30	Analysis of Case no. 3 in 3-way System	**37**
31	Analysis of Traffic for Case no. 5 in 3-way System	**38**
32	Analysis of Traffic for Case no. 11 in 3-way System	**39**
33	Analysis of Traffic for Case no. 15 in 3-Way System	**40**
34	Analysis of Traffic for 15 data points in 4-Way (Static v/s Dynamic)	**41**
35	Analysis of Traffic for Case no. 9 in 4-way System	**42**
36	Analysis of Traffic for Case no. 10 in 4-way System	**43**
37	Analysis of Traffic for Case no. 11 in 4-way System	**44**

CHAPTER 1 : INTRODUCTION

1.1 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.

1.2 ITCS - Improved Traffic Control System

ITCS is an Improved Traffic Control System that aims to reduce the traffic jams and clear the congestion and save time of the commuters stuck in traffic. We have developed a *Signal Switching algorithm* to regulate traffic that gives green signal time by analyzing the *number and type of vehicles* on the arm of the crossroad which will consider heterogeneity of the traffic.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 input. 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.

1.3 YOLO Model

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.

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.

With YOLO, a single CNN simultaneously predicts multiple bounding boxes and class probabilities for those boxes. The backbone CNN used in YOLO can be further simplified to improve processing time.

1.4 Pygame Simulator

A simulator with ITCS (Improved Traffic Control System) embedded in it has been developed using python's Pygame library to simulate real time traffic. It gives a demo of the entire model, and how traffic moves dynamically after green light timer changes for lanes are calculated using ITCS.

1.5 Web Application

A web application can be used to efficiently deliver functionality to the end user without burdening their systems. Processing can be done on the server to prevent data and memory leaks. A web application will be used to control the Rush Hour system from anywhere.

It uses React, HTML, CSS, JavaScript, BootStrap for the front-end and Nodejs, Python and MongoDB for the server-side scripting needs.

CHAPTER 2: PROBLEM STATEMENT

Problem Statement Formulation:

- Our project's prime intention is to build a system that can regulate the real time heterogeneous traffic so as to let more vehicles cross the crossroad and save time for the commuters, which in turn will reduce the fuel consumption and pollution.

2.1 Research

S. No.	Research Paper Title	Publication	Review
1	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	● Proposes an Arduino-UNO based system that processes the image in MATLAB, where the image is converted to a threshold image by removing saturation and hues, and traffic density is calculated. ● Depending on traffic count and traffic density, the Arduino sets the duration of green light for each lane.
2	Renjith Soman "Traffic Light Control and Violation Detection Using Image Processing"	IOSR Journal of Engineering (IOSRJEN), vol. 08, no. 4, 2018, pp. 23-27	● Makes use of a support vector machine algorithm along with image processing techniques. ● Image processing is done using OpenCV and the images are converted to grayscale images before SVM is applied. This system detects traffic density and red light violations.

3	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	• Proposes a smart traffic light system using ANN and fuzzy controller. This system makes use of images captured from cameras installed at traffic site. • 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.
4	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	• Proposes a fuzzy logic-controlled traffic light that can be adapt to the current traffic situations. • 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.
5	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	• Makes use of live video footage, images in small frames are captured and the road with most traffic density is given a green signal. • MATLAB image processing tools along with C++ algorithms are used.

2.2 Existing Systems

We contacted Smart City Chandigarh Project Manager to know about the existing Traffic Control System in Chandigarh and gathered the information regarding the same. We got to know that there is already an existing intelligent traffic control system known as ATCS i.e.

Adaptive traffic control system which measures the traffic queue length and predict the green time.

The average length between two crossroad junctions is 600 meters. For the first 20 meters of an arm of the crossroad, they have employed the traditional inductive loop technology to measure the traffic and after 20m, they have deployed a camera that will measure queue length on the basis of which the system will predict the green time. Then they will add the time to get the resultant green time.

If no vehicle crosses the inductive loop when there is green light for over 5 sec, they will turn the signal to red which will help them to save the time and that saved green signal time will be propagated to other junctions to reduce there respective red signal times.

But this system has few limitations as given below:

1. They are calculating the queue length rather than considering the number of the vehicles which leads to inaccurate approximation of the green time.
2. Heterogeneity of traffic is not considered i.e. type of vehicles like bus, car, truck, etc.
3. They are using traditional inductive loop technology to sense the presence of vehicles in the 20m vicinity of the traffic lights before skipping the green signal time for that junction.

CHAPTER 3: PROPOSED WORK

(Algorithm/Model/Approach)

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 2 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 used 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.

3.1 Vehicle Detection Module

- A pretrained YOLO model is used for vehicle detection which can detect vehicles of different classes like cars, motorbikes, heavy vehicles (buses and trucks).
- Further training of the YOLO model can be done to detect custom vehicles like auto-rickshaws and priority vehicles like ambulances, fire brigades and police vans.

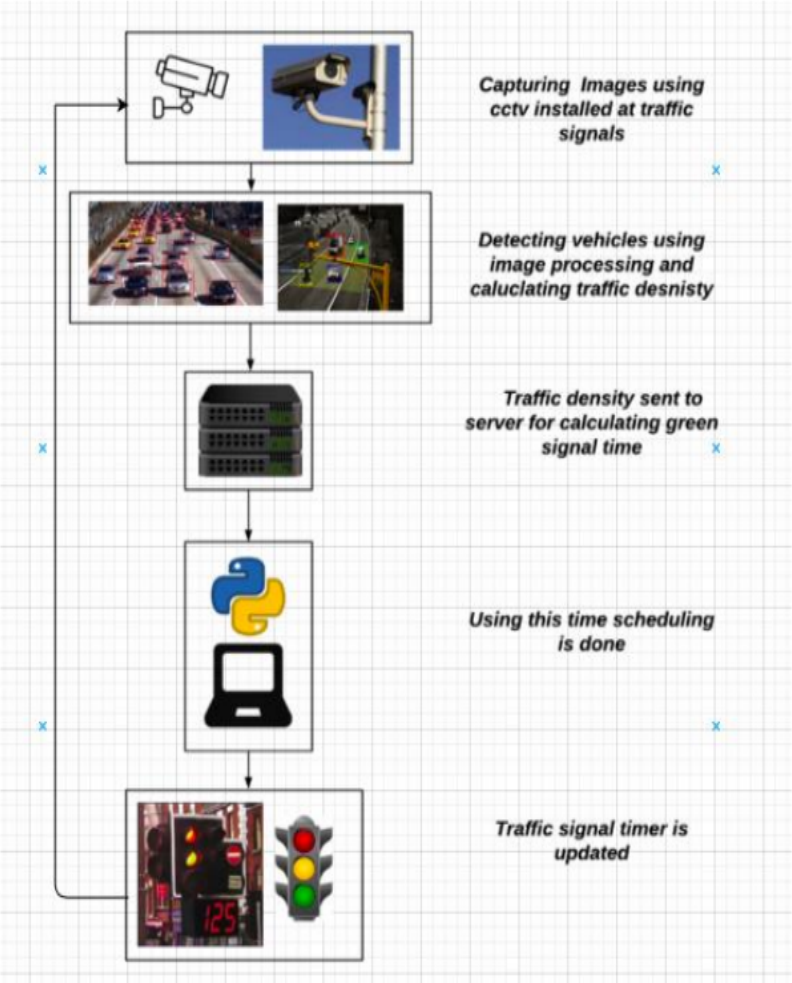

Fig 2: Workflow

3.2 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.

3.3 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.

CHAPTER 4: IMPLEMENTATION

4.1 Vehicle Detection Module

- 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 nonmax suppression (which makes sure the object detection algorithm only detects each object once), it then outputs recognized objects together with the bounding boxes.
- The model runs on pre-trained weights downloaded from the YOLO website. Further configuration of the .cfg file used for training can be changed in accordance with the specifications of our model. The number of output neurons in the last layer can be set to be equal to the number of classes the model is supposed to detect by changing the 'classes' variable. In our system, this was 5 viz. Car, Bike, Bus,Truck, and Rickshaw. Further enhancements to the model can be done by training on high priority vehicles like ambulances, fire brigades and police vans.

Fig 3: Traffic Example

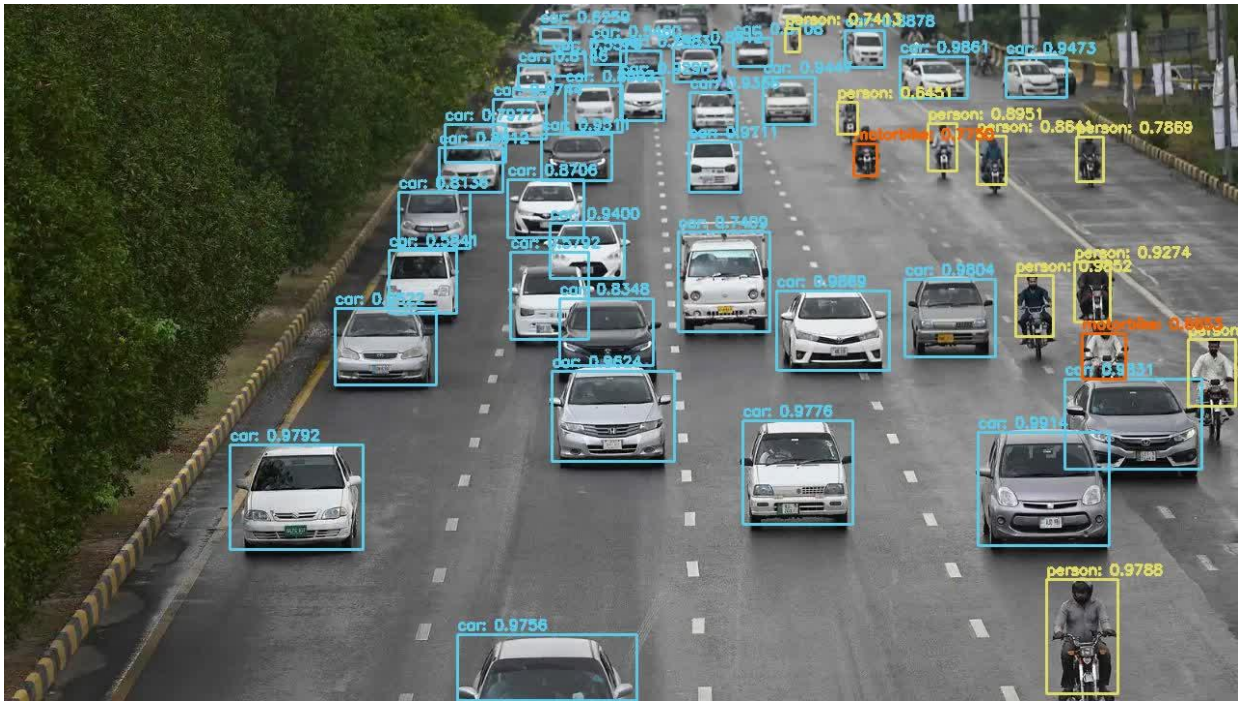

Fig 4: Traffic Detection with YOLO

4.2 Signal Switching Algorithm

Signal Switching algorithm dynamically changes the green light time for an arm of the crossroad.

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.
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
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. The green signal time is then calculated using the formula below.

GST = ($\sum_{vehicleClass} NoOfVehicles * AverageTime$ **) /** $(NoOfLanes + 1)$

where:

- GST is green signal time
- NoOfVehicles is the number of vehicles of each class of vehicle at the signal as detected by the vehicle detection module.
- averageTime is the average time the vehicles of that class take to cross an intersection.
- 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 the same.

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 **(Red → Green → Yellow → Red)**, and the yellow signals have been accounted for as well.

4.3 Simulation Module

Key Steps in development of simulation:

1. Below Figure 5 and 6 shows the background image considered for intersection.

3-way intersection

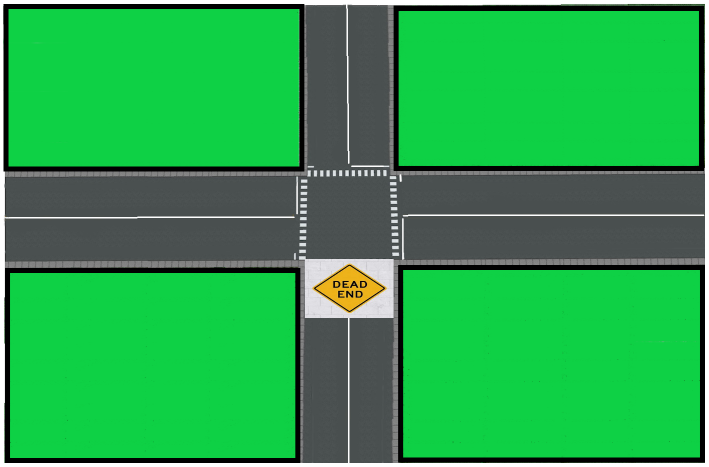

Fig 5: Pygame Simulator Background for 3 Lanes

4-way intersection

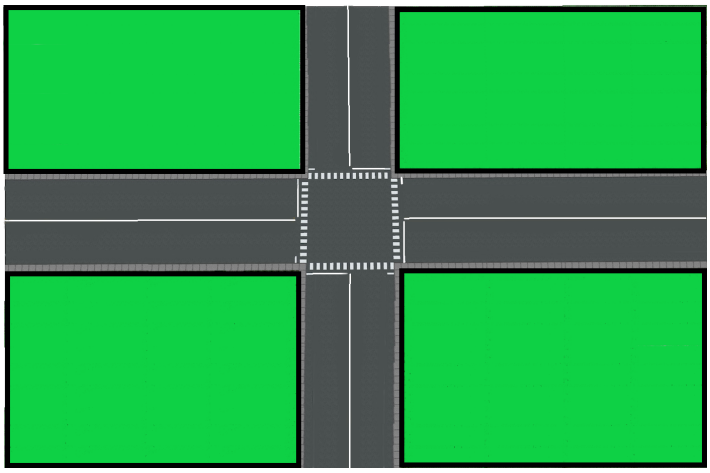

Fig 6: Pygame Simulator Background for 4 Lanes

2. Top-view images of car, truck, bus, rickshaw and motorcycle was gathered.

3. Aforementioned images resized.

Fig 7: Pygame Simulator Vehicles

4. Gathered images of traffic signals.

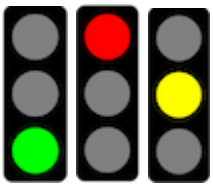

Fig 8: Pygame Simulator Traffic Lights

5. Code developed for rendering the appropriate image of the signal depending on whether it is red, green, or yellow.
6. Worked on 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.
7. 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 1 second.
8. Configured 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.
9. Code developed for how vehicles 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.
10. Displaying the number of vehicles that have crossed the signal and the total time elapsed since the start of the simulation.

11. 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 data analysis.
12. Worked on vehicle turning and crossing the intersection in the simulation to make it look more realistic.

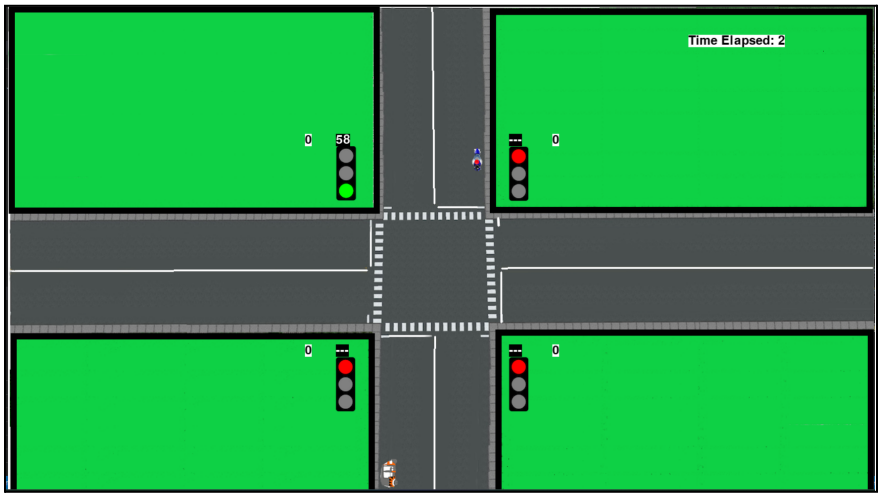

Fig 9: Pygame Simulator Initiation

13. Above Figure 9 shows the simulation just after it starts showing red and green lights, green signal time counting down from a default of 60 and red time of next signal is blank (--). When the signal is red, we display a blank value till it reaches 15 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.

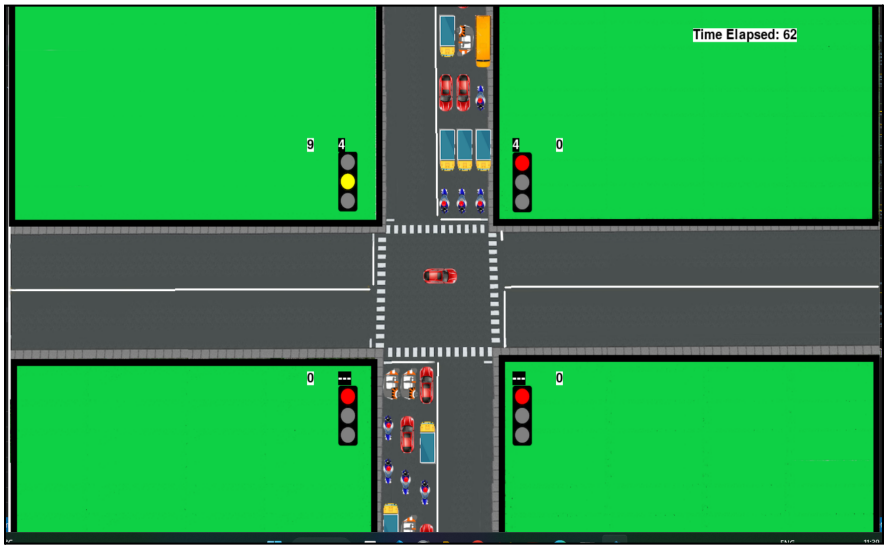

Fig 10: Simulation showing change of lights

14. Above Figure 10 shows the simulation showing yellow light and red time for next signal. When red signal time is less than 15 seconds, we show the countdown timer so that vehicles can start up and be prepared to move once the signal turns green.

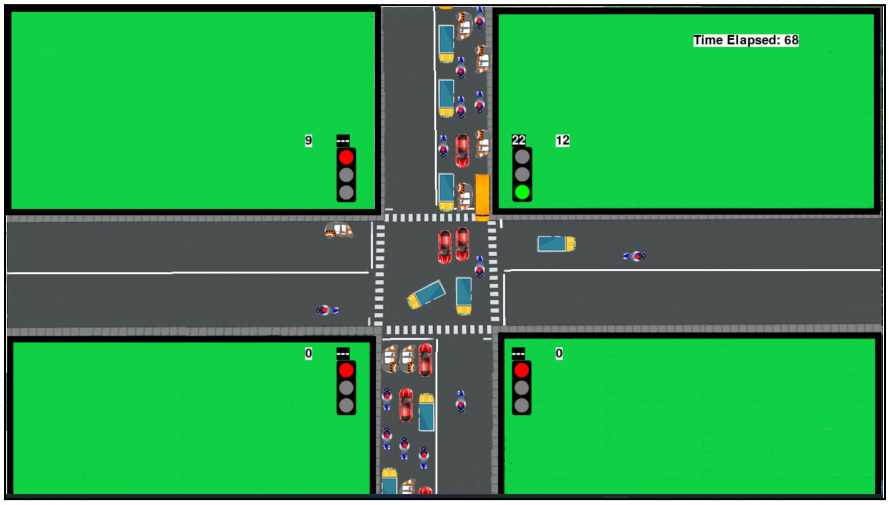

Fig 11: Simulation showing predicted green timer for a light

15. Above Figure 11 shows the instance when green time predicted by YOLO model is less than 60s on the basis of the number and type of vehicles and hence saving time of the commuters which is going to propagate for other arms of the crossroads too.

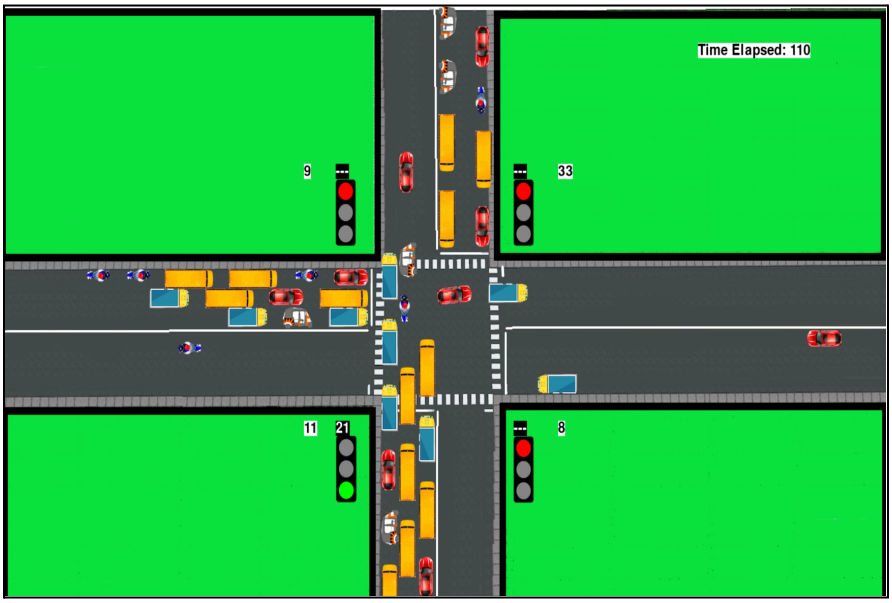

Fig 12: An intermediate state of a crossroad

16. Above figure 14 shows an intermediate state of a 4-Way crossroad and below figure 13 shows the intermediate state of a 3-Way Crossroad

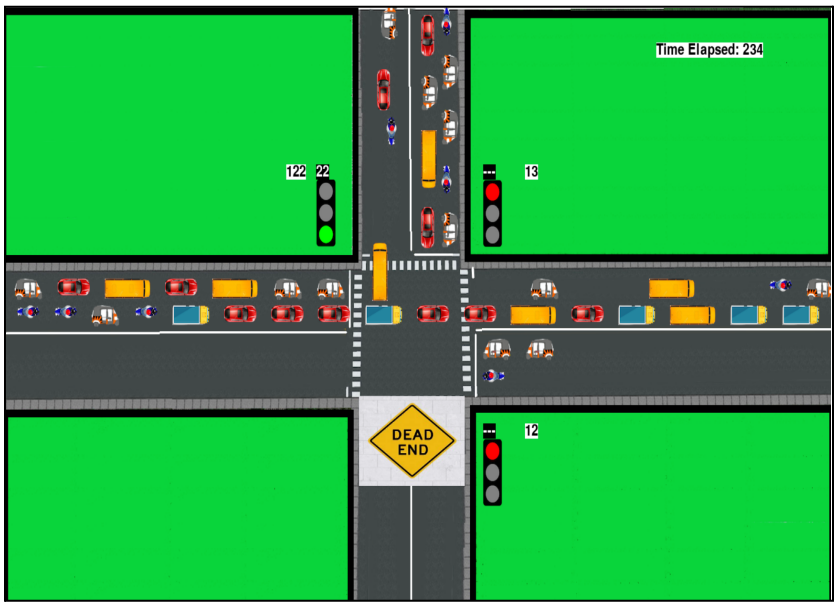

Fig 13: Intermediate Simulation for 3-Way traffic system

4.4 Technologies and Tools used

4.4.1 HTML & CSS

Fig 14: CSS HTML Logo

HTML (the Hypertext Markup Language) and CSS (Cascading Style Sheets) are two of the core technologies for building Web pages. HTML provides the *structure* of the page, CSS the (visual and aural) *layout,* for a variety of devices. Along with graphics and scripting, HTML and

CSS are the basis of building Web pages and Web Applications.

4.4.2 Javascript

Fig 15: Java Script Logo

JavaScript is a lightweight, interpreted programming language. It is designed for creating network-centric applications. It is complementary to and integrated with Java. JavaScript is very easy to implement because it is integrated with HTML. It is open and cross-platform.

4.4.3 Node.js

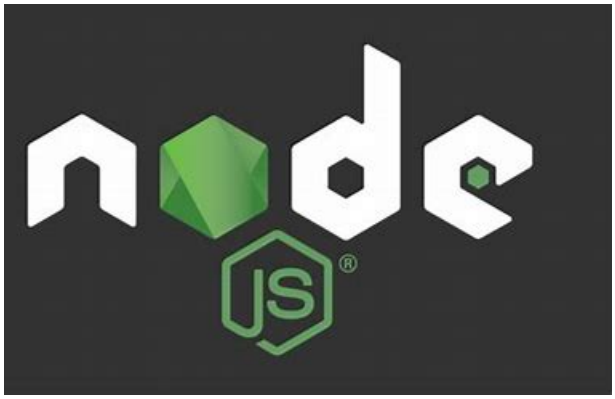

Fig 16: NodeJS Logo

Node.js is an open-source and cross-platform JavaScript runtime environment. Node.js runs the V8 JavaScript engine, the core of Google Chrome, outside of the browser. This allows Node.js to be very performant. Node.js provides a set of asynchronous I/O primitives in its standard library that prevent JavaScript code from blocking and generally, libraries in Node.js

are written using non-blocking paradigms, making blocking behavior the exception rather than the norm.

4.4.4 MongoDB

Fig 17: MongoDB Logo

MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and licensed under the Server Side Public License.

4.4.5 React

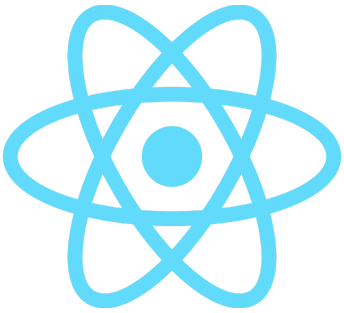

Fig 18: React Logo

React is a JavaScript library for building user interfaces.

- Declarative: React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right

components when your data changes. Declarative views make your code more predictable, simpler to understand, and easier to debug.

- Component-Based: Build encapsulated components that manage their own state, then compose them to make complex UIs. Since component logic is written in JavaScript instead of templates, you can easily pass rich data through your app and keep the state out of the DOM.
- Learn Once, Write Anywhere: We don't make assumptions about the rest of your technology stack, so you can develop new features in React without rewriting existing code. React can also render on the server using Node and power mobile apps using React Native.

4.4.6 Python

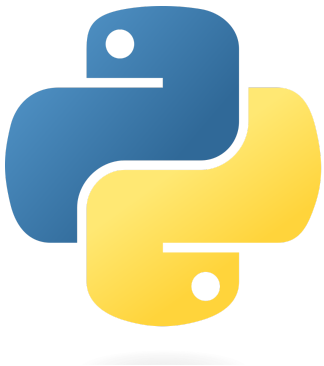

Fig 19: Python Logo

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

4.4.6 Pygame

Fig 20: Pygame Logo

Pygame is a cross-platform set of Python modules which is used to create video games. It consists of computer graphics and sound libraries designed to be used with the Python programming language. Pygame was officially written by Pete Shinners to replace PySDL. Pygame is suitable to create client-side applications that can be potentially wrapped in a standalone executable.

4.4.7 Visual Studio Code

Fig 21: Visual Studio Code Logo

Visual Studio Code is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.

4.4.8 Jupyter Notebook

Fig 22: Jupyter Notebook Logo

The Jupyter Notebook is an open-source web application that allows users to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, statistical modeling, data visualization, machine learning, and much more. We used Jupyter Notebook for writing code, compiling it and getting the results in modern fashion.

CHAPTER 5: RESULTS AND DISCUSSION

5.1 Results

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 which also took heterogeneity of vehicles into consideration. 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 **24.52%** improvement for 4 lane roads and **13.61%** for 3 lane roads over the current system in terms of the number of vehicles crossing the intersection in a given time frame, 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.

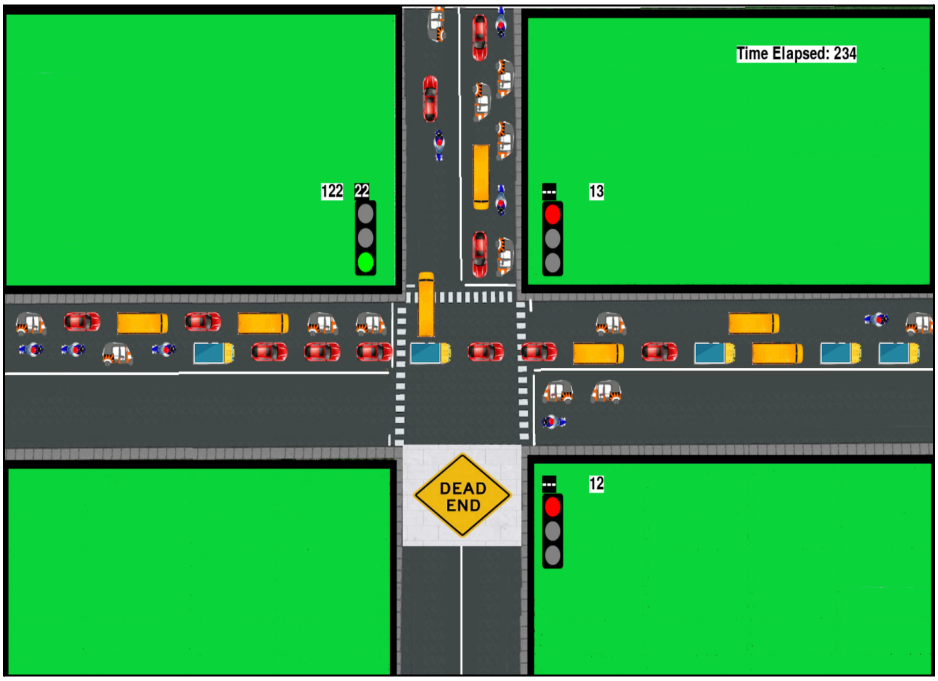

Fig 23: 3-Way System Simulation

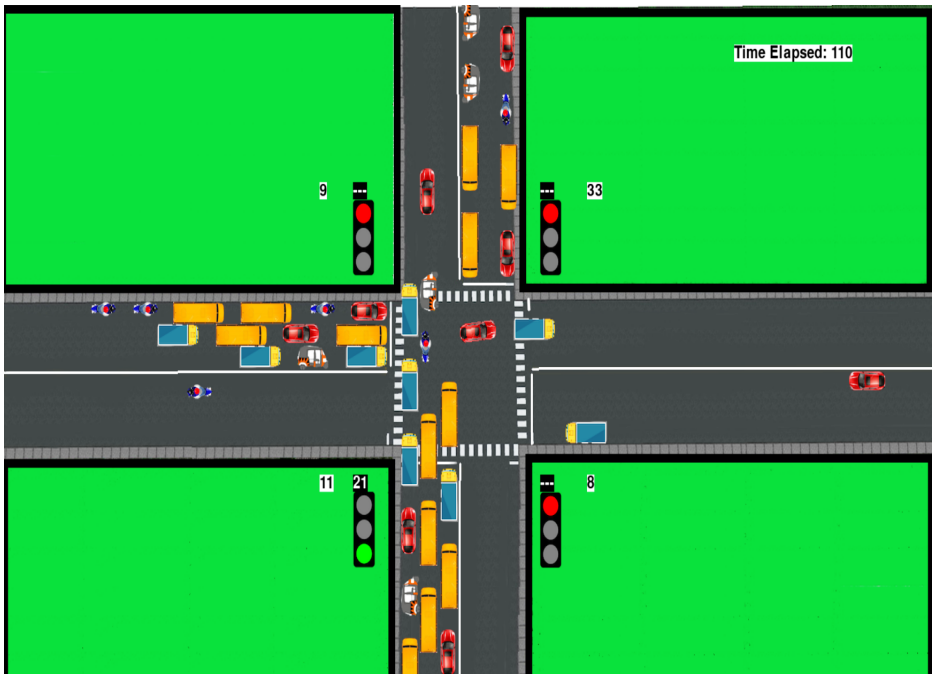

Fig 24: 4-way System Simulation

Fig 25: YOLO Model Detection

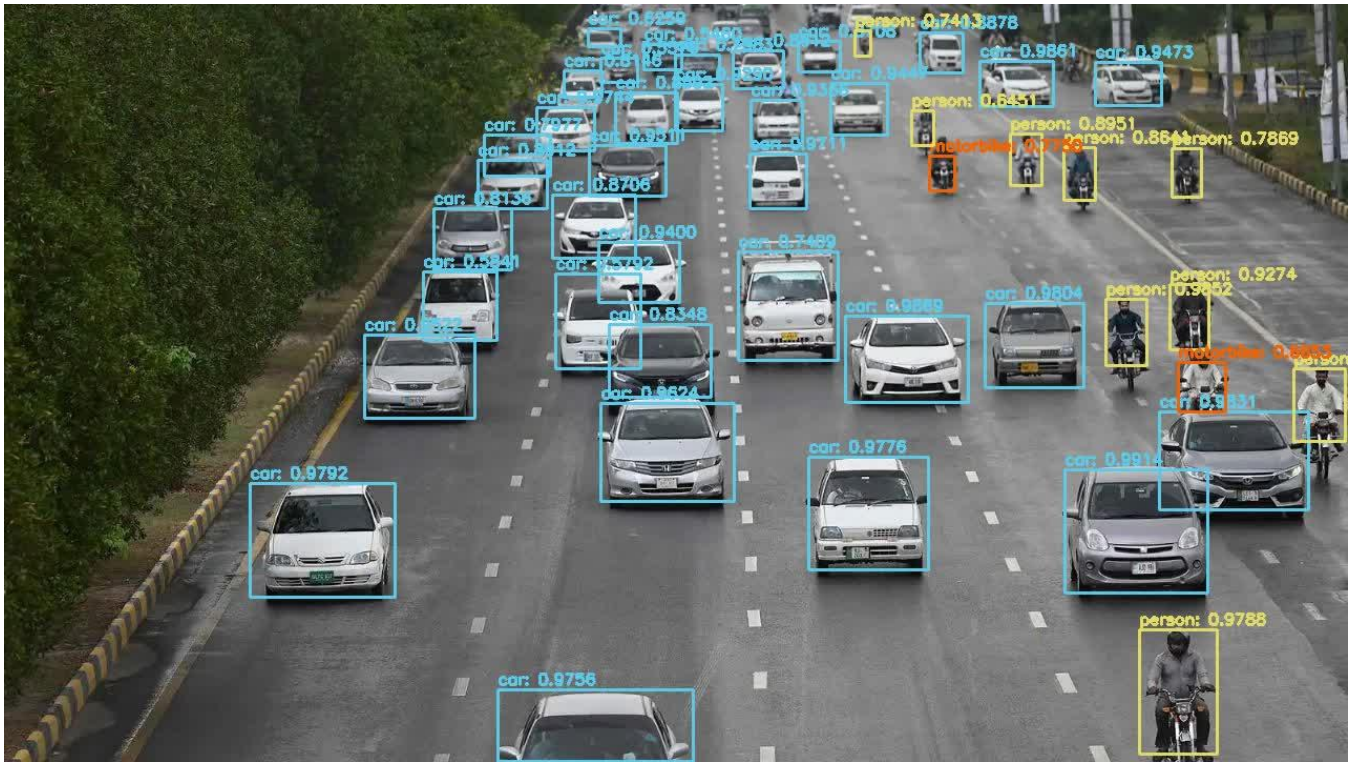

Fig 26: YOLO Model Detection

The below figures contain the following variables as columns:

- P(i) indicates the traffic density, where the sum of all the P(i) is equal to one.
- Lane(i) indicates the number of vehicles crossed from that lane.
- Total(S) indicates the total number of vehicles crossed from all the lanes considering the static system.
- Total(D) indicates the total number of vehicles crossed from all the lanes considering the dynamic system.
- Improvement indicates the percentage improvement considering the dynamic system over the static system. The red circled regions indicate that on these cases the improvement has been less than 10%.

	P1	P2	P3	Lane1	Lane2	Lane3	Total(D)	P1	P2	P3	Lane1	Lane2	Lane3	Total(S)	Improvement(%Age)
1	0.000	0.000	1.000	0	0	257	257	0.000	0.000	1.000	0	0	133	133	93.233083
2	0.000	0.500	0.500	0	113	113	226	0.000	0.500	0.500	0	126	76	202	11.881188
3	0.000	0.600	0.400	0	119	97	216	0.000	0.600	0.400	0	134	66	200	8.000000
4	0.333	0.333	0.334	74	93	61	228	0.333	0.333	0.334	68	90	55	213	7.042254
5	0.300	0.500	0.200	65	126	42	233	0.300	0.500	0.200	74	126	28	228	2.192982
6	0.500	0.400	0.100	100	113	29	242	0.500	0.400	0.100	100	109	14	223	8.520179
7	0.500	0.300	0.200	107	80	40	227	0.500	0.300	0.200	108	81	32	221	2.714932
8	0.700	0.200	0.100	185	38	29	252	0.700	0.200	0.100	142	51	13	206	22.330097
9	0.500	0.450	0.050	123	111	11	245	0.500	0.450	0.050	109	115	10	234	4.700855
10	0.300	0.600	0.100	73	161	18	252	0.300	0.600	0.100	66	144	16	226	11.504425
11	0.200	0.550	0.250	53	109	59	221	0.200	0.550	0.250	48	131	43	222	-0.450450
12	0.350	0.150	0.500	80	51	89	220	0.350	0.150	0.500	79	37	83	199	10.552764
13	0.350	0.500	0.150	85	134	33	252	0.350	0.500	0.150	79	123	25	227	11.013216
14	0.940	0.040	0.020	200	13	5	218	0.940	0.040	0.020	168	13	7	188	15.957447
15	0.850	0.100	0.050	193	18	17	228	0.850	0.100	0.050	194	31	15	240	-5.000000

Fig 27: Data points for 3-way System

	P1	P2	P3	P4	Lane1	Lane2	Lane3	Lane4	Total(D)	P1	P2	P3	P4	Lane1	Lane2	Lane3	Lane4	Total(S)	Improvement(%Age)
1	0.00	0.00	0.00	1.00	0	0	0	269	269	0.00	0.00	0.00	1.00	0	0	0	188	188	43.085106
2	0.00	0.00	0.50	0.50	0	0	146	158	304	0.00	0.00	0.50	0.50	0	0	109	117	226	34.513274
3	0.00	0.30	0.30	0.40	0	102	80	120	302	0.00	0.30	0.30	0.40	0	120	69	81	270	11.851852
4	0.25	0.25	0.25	0.25	94	86	74	82	336	0.25	0.25	0.25	0.25	72	92	58	60	282	19.148936
5	0.30	0.30	0.20	0.20	101	114	57	66	338	0.30	0.30	0.20	0.20	88	99	45	55	287	17.770035
6	0.50	0.20	0.20	0.10	177	77	76	27	357	0.50	0.20	0.20	0.10	139	75	53	24	291	22.680412
7	0.30	0.20	0.30	0.20	94	76	111	65	346	0.30	0.20	0.30	0.20	76	75	84	52	287	20.557491
8	0.70	0.10	0.10	0.10	229	32	37	32	330	0.70	0.10	0.10	0.10	180	46	36	17	279	18.279570
9	0.50	0.40	0.05	0.05	148	130	26	17	321	0.50	0.40	0.05	0.05	133	131	20	16	300	7.000000
10	0.30	0.30	0.30	0.10	99	99	90	38	326	0.30	0.30	0.30	0.10	87	110	71	32	300	8.666667
11	0.20	0.50	0.05	0.25	69	150	17	92	328	0.20	0.50	0.05	0.25	63	157	20	61	301	8.970100
12	0.35	0.15	0.35	0.15	124	49	107	53	333	0.35	0.15	0.35	0.15	93	57	59	38	247	34.817814
13	0.35	0.35	0.15	0.15	122	134	49	39	344	0.35	0.35	0.15	0.15	91	129	39	31	290	18.620690
14	0.94	0.02	0.02	0.02	274	4	14	6	298	0.94	0.02	0.02	0.02	165	8	8	7	188	58.510638
15	0.85	0.05	0.05	0.05	231	21	27	22	301	0.85	0.05	0.05	0.05	163	18	21	8	210	43.333333

Fig 28: Data points for 4-way System

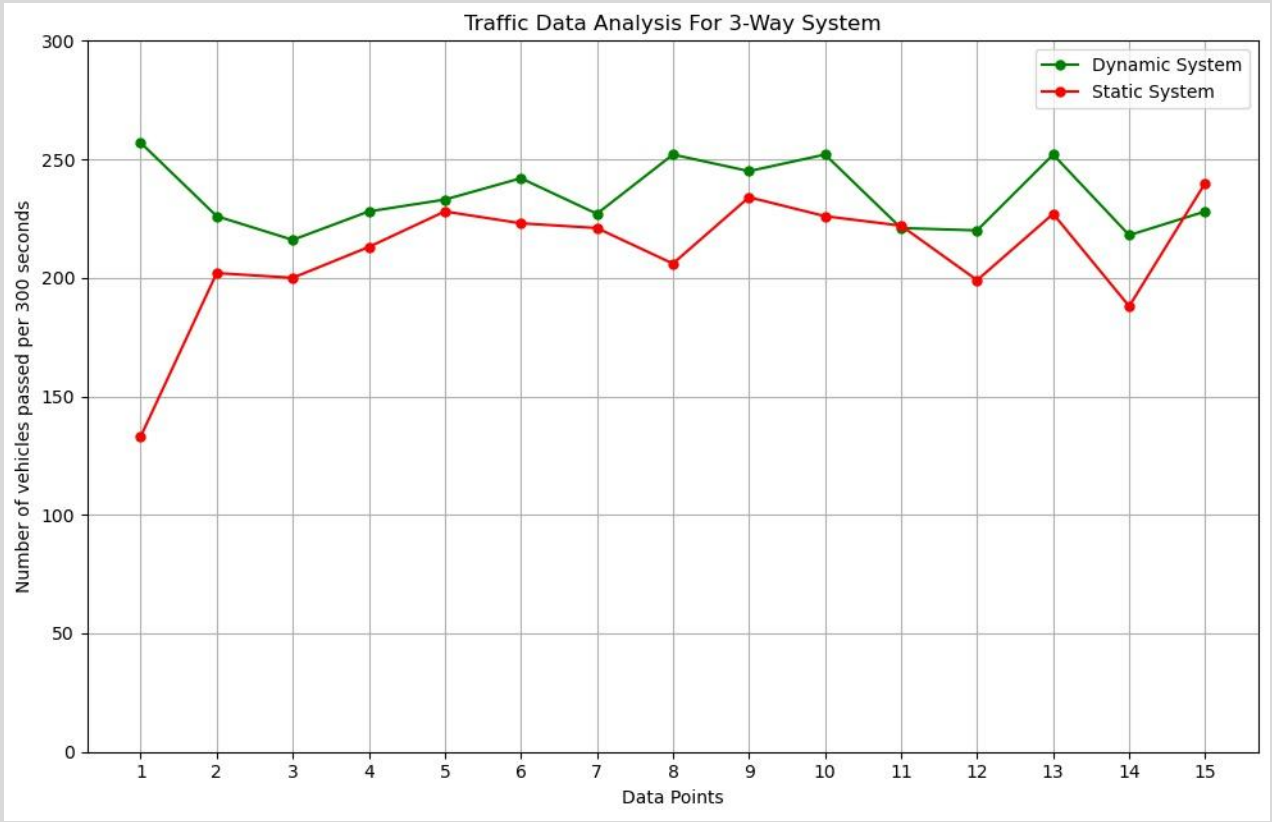

Fig 29: Analysis of Traffic for 15 data points in 3-Way System(Static v/s Dynamic)

1. Plot shows the traffic data analysis for 3-Way System for both dynamic and static systems. The x-axis indicates the test case number and the y-axis indicate the number of vehicles passed per 300 seconds. There is a small amount of gap between the static and the dynamic case indicating that the dynamic and the static system give near to same results for 3-Way System.

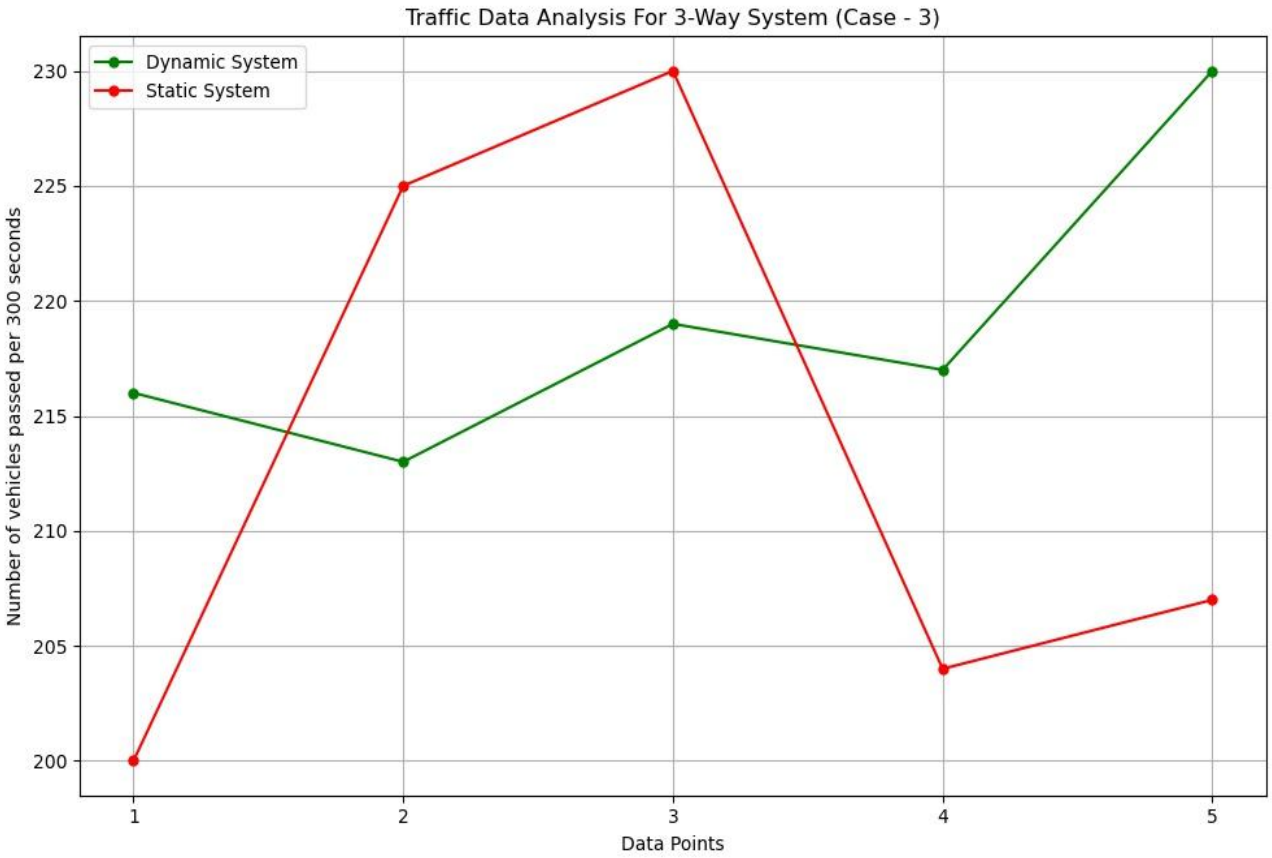

Fig 30: Analysis of Traffic for Case no. 3 in 3-Way System

2. Plot shows the Traffic data analysis for 3-Way System for test case number 3 for both dynamic and static system. Dynamic system gives almost uniform results and static system is having abrupt changes.

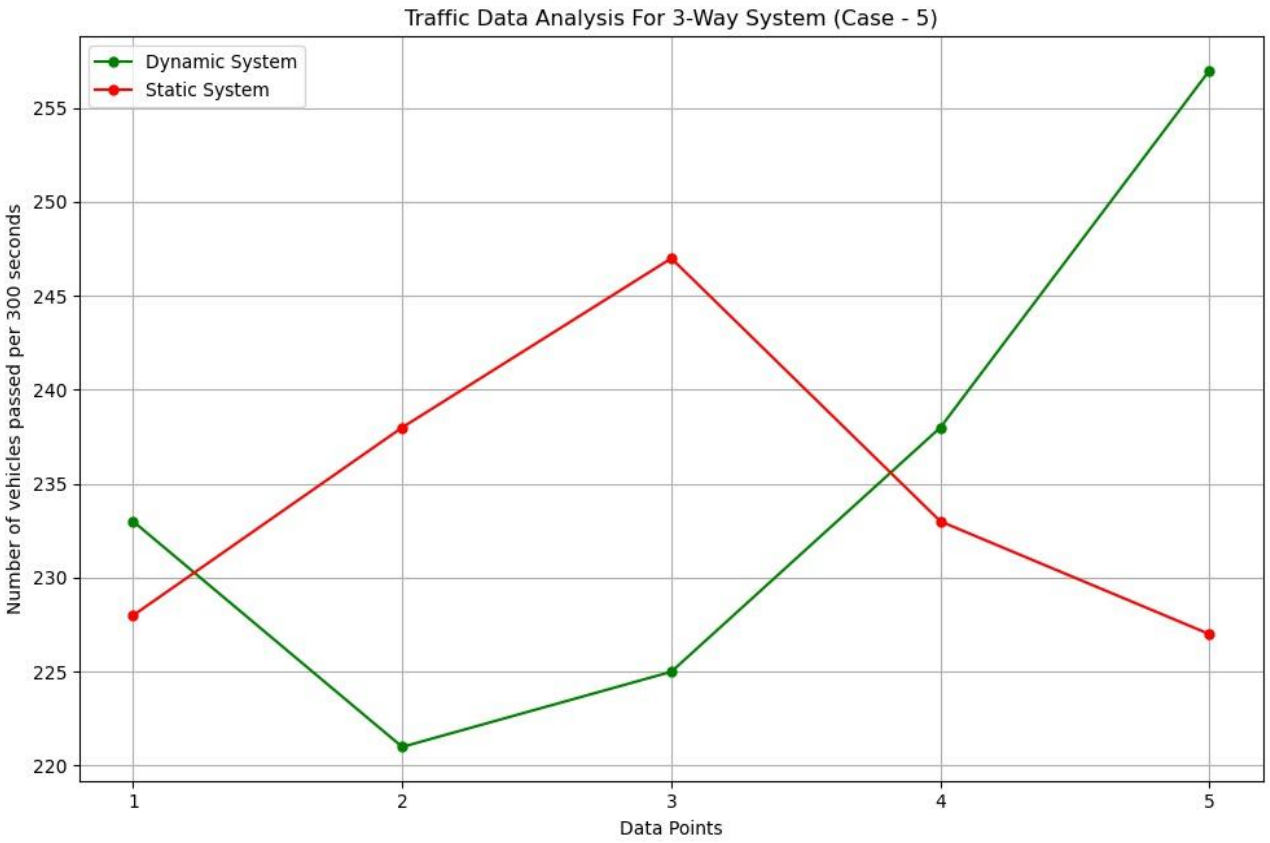

Fig 31: Analysis of Traffic for Case no. 5 in 3-Way System

3. Plot shows the Traffic data analysis for 3-Way System for test case number 5 for both dynamic and static system. Dynamic system first decreases and then increases at a faster rate and Static system first increases and then decreases.

Fig 32: Analysis of Traffic for Case no. 11 in 3-Way System

4. Plot shows the Traffic data analysis for 3-Way System for test case number 11 for both dynamic and static systems. Dynamic and static system is having a large gap between them with dynamic system showing good results.

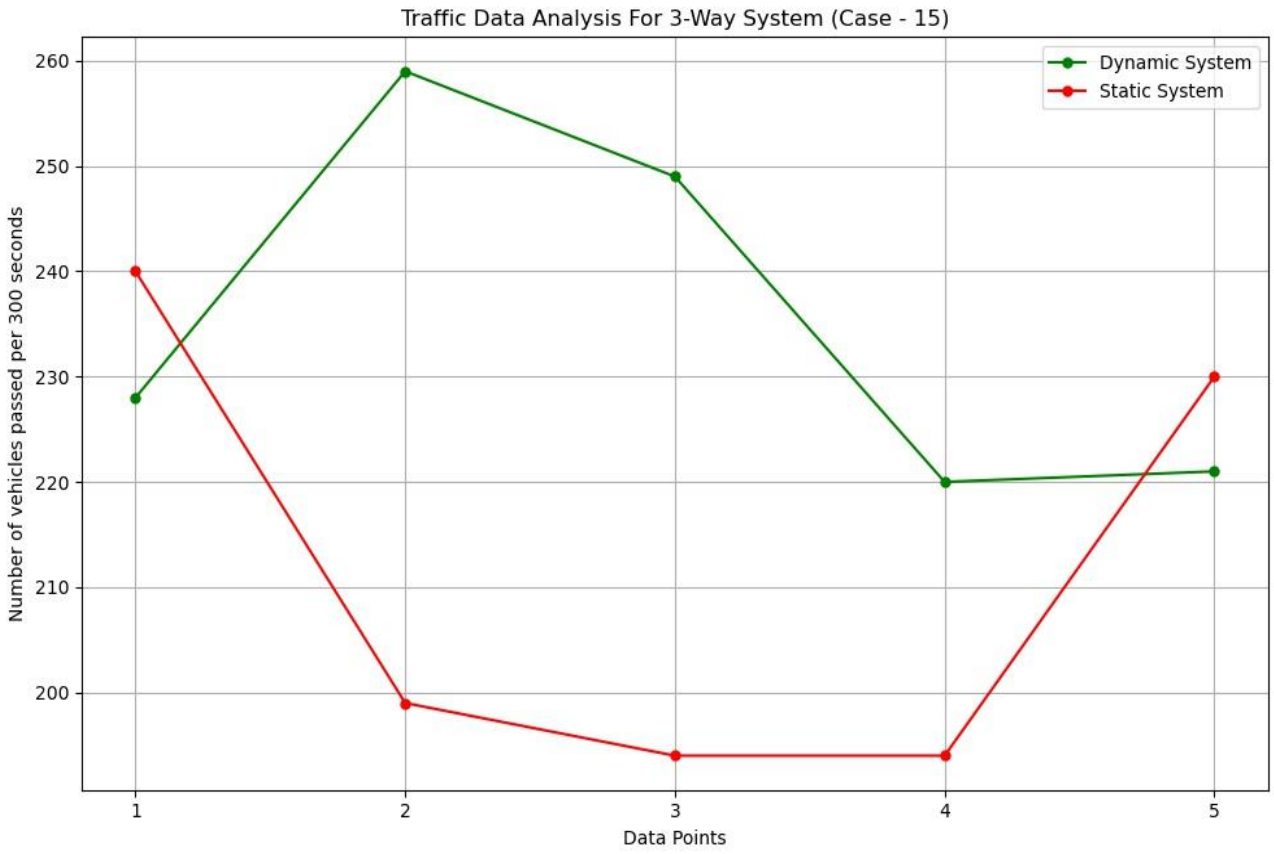

Fig 33: Analysis of Traffic for Case no. 15 in 3-Way System

5. Plot shows the Traffic data analysis for 3-Way System for test case number 15 for both dynamic and static system. Both the systems are having a large gap between them with the dynamic system having a good response time for the vehicles to pass.

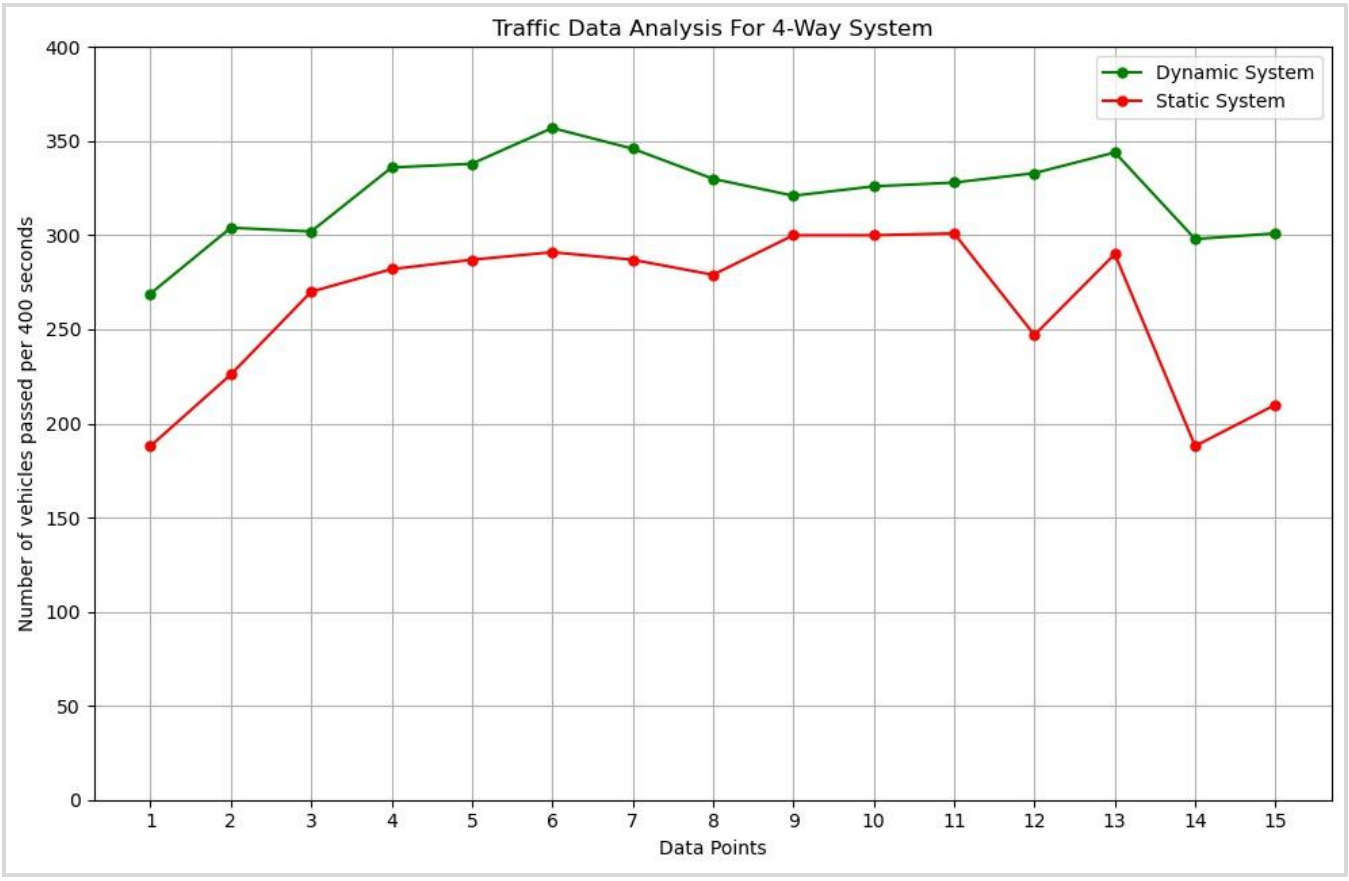

Fig 34: Analysis of Traffic for 15 data points in 4-Way (Static v/s Dynamic)

6. Plot shows the traffic data analysis for 4-Way System for both dynamic and static systems. The x-axis indicates the test case number and the y-axis indicate the number of vehicles passed per 400 seconds. There is a significant amount of gap between the static and the dynamic case indicating that the dynamic cases respond well on the random input traffic density.

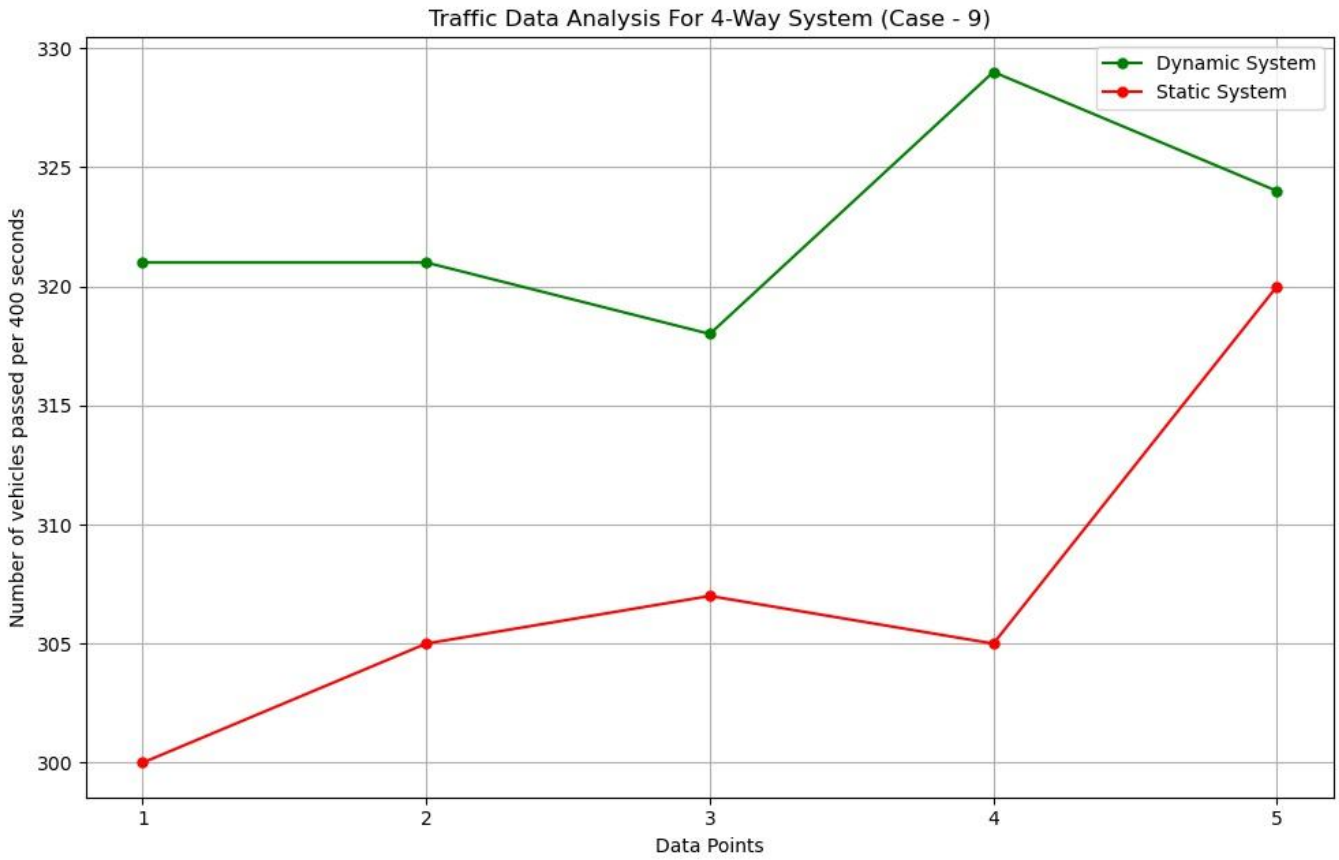

Fig 35: Analysis of Traffic for Case no. 9 in 4-Way System

7. Plot shows the Traffic data analysis for 4-Way System for test case number 9 for both dynamic and static system. Dynamic system shows good result with a large gap between both the systems.

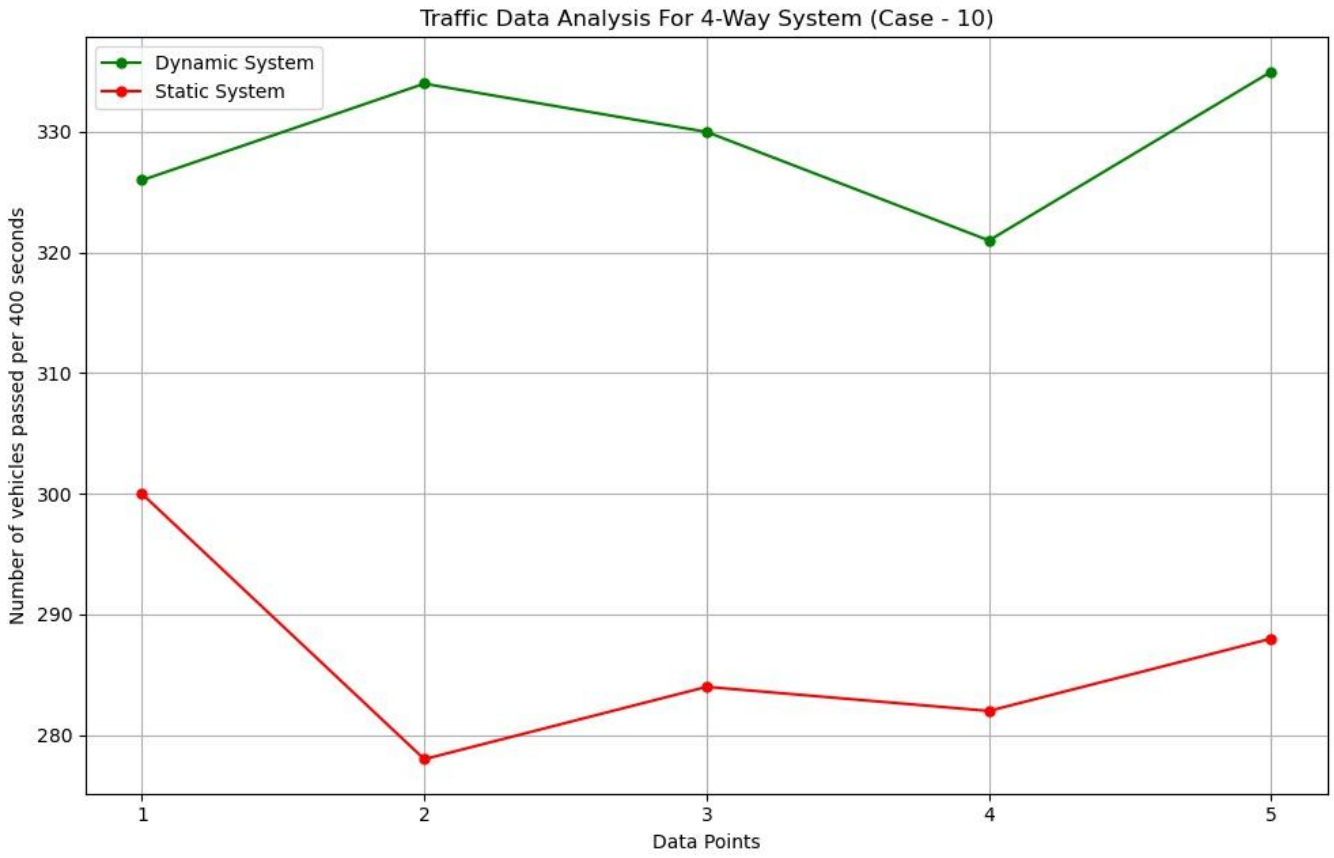

Fig 36: Analysis of Traffic for Case no. 10 in 4-Way System

8. Plot shows the Traffic data analysis for 4-Way System for test case number 10 for both dynamic and static system. Dynamic system shows good result with a large gap between both the systems.

Fig 37: Analysis of Traffic for Case no. 11 in 4-way System

9. Plot shows the Traffic data analysis for 4-Way System for test case number 11 for both dynamic and static system. Dynamic system shows good result with a large gap between both the systems.

5.2 Discussion

Through this project, we have made a system that has the potential to give more accurate green time dynamically by analyzing the number and type of vehicles using the YOLO model thus helping commuters to save their time and saving nature by the reduction of fuel consumption and air pollution.

CHAPTER 6: TIMELINE

Week 0

(12 September 2022 - 18 September 2022)

Team formation and Mentor Selection:

- Team Member Formation.
- Choosing the field/technology of interest after a series of meetings with the team members.
- Selecting the appropriate mentor best suited for the technology to be used.

Week 1

(19 September 2022 - 25 September 2022)

Project Idea Discussion and Synopsis Drafting:

Decided on three project ideas:

- NGO Donation System / Laundry System
- Depression Analysis and Guide System
- Smart Traffic Management System

Finally, we decided to take up the third idea as we found it to be a more practical and useful idea. It was also more feasible according to our knowledge and skill set.

Week 2

(26 September 2022 - 2 October 2022)

Discovering Project Requirements:

- Looking onto the existing work in this related field, if done.
- Deciding the further contributions we can make.
- Looking for availability of existing research papers on this topic

Week 3

(3 October 2022 - 9 October 2022)

Meeting with Chandigarh Smart City Official:

- Fixed a meeting with a Chandigarh Smart City official.
- Met the aforementioned official and discussed the existing problems in the domain with them.

Week 4

(10 September 2022 - 16 October 2022)

Modification of Problem Statement:

- Meeting with the Chandigarh Smart City official gave us more clarity on the idea and the problems, and did research on how to improve the existing system deployed in Chandigarh.
- Changes made to problem statement accordingly after discussion with our mentor.

Week 5

(17 October 2022 - 23 October 2022)

Technology Learning and Selection:

- Explored various methods of making a simulator such as Pygame, Sumo Simulator, Veins and OpenStreetMap.
- Also referred to some online articles and tutorials regarding YOLO.

Week 6

(24 October 2022 - 30 October 2022)

Started developing the Frontend & Backend for Rush Hour:

- Started developing the website's front end using HTML, CSS and JavaScrip and React.
- Website Layout designed before the start of actual implementation by team discussions and brainstorming.

Week 7

(31 October 2022 - 6 November 2022)

YOLO Model Development:

- Developed YOLO Model for accurate classification of images.
- Worked on the website side by side.

Week 8

(7 Novemeber 2022 - 13 November 2022)

Dynamic Green Timer Algorithm Development:

- Brainstorming done on algorithm to dynamically manage green signal timer.

Week 9

(14 November 2022 - 20 November 2022)

Improved Traffic Control System Algorithm Development:

- Changes made to the Dynamic Green Timer Algorithm, improved algorithm ITCS developed.
- Coded the algorithm.

Week 10

(21 November 2022 - 27 November 2022)

Pygame Simulator Development:

- Explored various simulators for traffic management, settled for Pygame.
- Pygame Simulation Development started.

Week 11-12

(28 November 2022 - 4 December 2022)

<u>Testing & Integration:</u>

- Merged the simulator with ITCS.
- Simulator testing done along with minor improvements.

CHAPTER 7: CONCLUSION AND FUTURE WORK

7.1 Conclusion

Improved Traffic Control System (ITCS) is a dynamic timer incorporating Machine Learning techniques. The deliverable will be a python code that will give us the green light time for that particular arm of the crossroad on the basis of number and type of the vehicles that will help to avoid traffic jams, commuters to save their valuable time using the latest cutting edge technology over traditional ones like inductive loop.

7.2 Future Scope

- Hardware implementation of Improved Traffic Control System (*ITCS*) using Raspberry Pi. We will have X number of roads with cameras installed that will capture video/image which will be further processed by Raspberry Pi that will generate the Green time and will automatically set red/green time for that road.
- The YOLO model that we are using to classify the vehicles doesn't classify a few types of vehicles like Auto Rickshaw, ambulances, fire brigades etc. So, we are planning to add those classes for classification too.
- After adding the above classes into our YOLO model, we will work on improving ITCS by adding more features such as providing GREEN corridor to high priority vehicles like ambulances, fire brigade etc so as to minimize the human intervention that is traditionally done by manually setting the lights red or green respectively.
- Adding new feature of traffic congestion control during the night using infrared technology or by training the YOLO Model for low light images if possible.
- Building an user interface where we can have visuals of real time analysis of dynamic timer and gather more data that is going to help in improving our system. We can have some kind of integration to change the traffic lights from the user interface itself.

CHAPTER 8: REFERENCES

[1] 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.

[2] 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.

[3] 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.

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

[5] 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.

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

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

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

[9] Open Data Science, ‘Overview of the YOLO Object Detection Algorithm’, 2018. [Online]. Available: https://odsc.medium.com/overview-of-the-yolo-object-detection-algorithm-7b52a745d3e0