University of Mumbai

Adaptive Traffic Signal Timer Based On Traffic Density

Submitted at the end of semester VI in partial fulfillment of requirements

For the degree of

Bachelors in Technology

by

KEYUR VINAY KULKARNI
Roll No: 1913026
CHINMAY ASHOK NAYAK
Roll No: 1913032
PARAS JITENDRA SHAH
Roll No: 1913036
ARYAMAN ABHIJIT PARKHI
Roll No: 1913037

Guide

Prof. Maruti Zalte and Prof. Mahesh Warang

Department of Electronics and Telecommunication Engineering
K. J. Somaiya College of Engineering, Mumbai-77
(Autonomous College Affiliated to University of Mumbai)

Batch 2019 -2023

K. J. Somaiya College of Engineering, Mumbai-77

(Autonomous College Affiliated to University of Mumbai)

Certificate

This is to certify that this report entitled **Adaptive Traffic Signal Timer Based On Traffic Density** submitted by Keyur Kulkarni, Chinmay Nayak, Paras Shah and Aryaman Parkhi at the end of semester VI of TY B. Tech is a bonafide record for partial fulfillment of requirements for the degree of Bachelors in Technology in Electronics and Telecommunication Engineering of University of Mumbai

___________________	________________________
Guide/Examiner1	Examiner2

Date: / /

Place: Mumbai-77

K. J. Somaiya College of Engineering, Mumbai-77

(Autonomous College Affiliated to University of Mumbai)

DECLARATION

We declare that this written report submission represents the work done based on our and / or others' ideas with adequately cited and referenced the original source. We also declare that we have adhered to all principles of intellectual property, academic honesty and integrity as we have not misinterpreted or fabricated or falsified any idea/data/fact/source/original work/ matter in my submission.

We understand that any violation of the above will be cause for disciplinary action by the college and may evoke the penal action from the sources which have not been properly cited or from whom proper permission is not sought.

Signature of the Student **Roll No.: 1913026**	**Signature of the Student** **Roll No.: 1913032**
Signature of the Student **Roll No.: 1913036**	**Signature of the Student** **Roll No.: 1913037**

Date: / /

Place: Mumbai-77

Abstract

With the growing population and number of automobiles in cities, traffic congestion is becoming a major issue. Traffic congestion not only wastes time and adds stress to driver's lives, but they also increase fuel consumption and pollution. Megacities are the ones who are most affected by it, and its ever-increasing nature necessitates the calculation of real-time road traffic density for better signal control and traffic management.

One of the most important variables affecting traffic flow is the traffic controller. As a result, there is a need to improve traffic controller in order to better meet this growing demand. The goal of our proposed system is to use live footages from traffic junction cameras to calculate traffic density using image processing and traffic switching algorithm. Thus, people will be able to travel more quickly and pollution will be reduced.

***Key words*: Traffic Control, Traffic Light System, Traffic Management, Intelligent Transport Systems, Smart Surveillance, Object Detection, OpenCV, Pygame, Numpy.**

Contents

List of Figures

1 Introduction

1.1 Background

In the urban areas, the routing of road traffic is governed by Traffic signals placed at intersections. The conventional traffic signal displays Red signal for a fixed time followed by a few seconds of Yellow signal and then the Green signal for a fixed time. This fixed amount of time is set by the Traffic regulatory authorities by their own experience and knowledge of the average amount vehicles passing through that specific intersection. The management and routing of traffic is of great importance because of limited, and sometimes constrained, road infrastructure and also because of large number of commuters and correspondingly large number of vehicles present on urban roads. Inefficiencies and mismanagement result not only in unpleasant commute experience and needless wastage of precious time, but also, in increased amount of emissions and strain on the existing road infrastructure.

The pattern of traffic flow is so dynamic that it changes from time to time within a day. For example, in the morning, the traffic flow might be towards workplaces, market areas, shopping places, etc., while in the evening the traffic flow maybe directed towards residential areas. Even the number of vehicles on the road changes w.r.t. time i.e., the Traffic density is higher during ‘peak hours’ or ‘rush hours’ of that area and lower otherwise.

This time estimation by traffic regulatory authorities, which eases the operation of traffic signals at the cost of ‘logical rigidity’, may be convenient for roads where the traffic density doesn’t change very drastically. But, for urban areas, which host innumerable vehicles throughout the day despite of constrained road infrastructure, it becomes an important issue. The amount of time wasted in road transport also hampers the business hours and, hence, the productivity of the region. Besides, the increased carbon emissions, caused by the unnecessary time spent by the vehicles in transport, not only disturbs the quality-of-life of people living in the city but also interferes with the national and international mission of reducing the carbon footprint.

It might be natural to assume that waiting in traffic jams is psychologically frustrating to the commuters. Clearly, we need to follow the traffic rules set by the relevant traffic regulatory authorities on the road. The, above mentioned, frustration may even be justified when one of the roads (approaching the conventional traffic signal) has to needlessly wait longer than other road(s) despite having greater traffic volume than the compared road(s) at a particular time of the day. This situation is the example of the failure of the conventional traffic signals.

1.2 Motivation

A conventional traffic signal, with constant time setting for Red and Green signal, may not cater to these dynamics of the Traffic flow. Judging from the simplicity of the conventional traffic signals, it is safe for us to assume that, these conventional traffic signals are incapacitated and unequipped to adapt to the dynamics of traffic flow. We can simply validate this by our own experience of traffic jams caused in the presence of traffic signals. In such critical cases, traffic signals are overridden by intervention of traffic police because it can’t comprehend the traffic flow and cannot change its routing strategy. Also, people stop following the traffic signals because of its unreliability in case of high volume of traffic.

Traffic signals need not be too complicated and sophisticated means of controlling flow of vehicles, because the traffic, which we refer to, is composed of human-driven vehicles (yet). Its working must still be intuitive and understanding enough for the common people so that they can believe in its working and maintain order on roads.

The core drawback of conventional signals is their inability to adjust the time of Red and Green signal in a single or multiple traffic signal cycles for the changing patterns of traffic flow.

1.3 Scope of the project

We believe that, answering the problem of traffic flow mismanagement, caused by conventional traffic signals, involves addressing the core drawback of the conventional traffic signals (as mentioned above) and deriving a solution for it. The immediate solution that we can think of is dynamically adjusting the Red signal (stop) time and Green signal (go) time in accordance with the number of vehicles present on the road. This problem statement divides our solution in 2 distinct stages,

1. Sensing the number of vehicles on the road,
2. Algorithm that takes the number of vehicles on the road as input and outputs the Green signal time,

Since actual hardware implementation is infeasible (not in current scope of the project), in order to demonstrate the working of the algorithm we have to use simulation environment which is capable enough to facilitate the traffic elements and their behaviour. This leads us to the 3rd stage,

3. Simulation environment for algorithm,

It is fair to assume that the traffic intersections, in urban areas of developed and in some developing countries, will be equipped with CCTV surveillance cameras for efficient governance. By this assumption we plan to sense the traffic and count the vehicles on the road by using computer vision / image processing techniques.

1.4 Brief description of the project

As mentioned earlier, our main aim for this project is to make the efficient use of the existing and also upcoming traffic signals and traffic systems. This can be achieved by using the above mentioned algorithm which helps in Dynamic Time allocation for each traffic signal and for each and every cycle.

The outcome of this project is that we will able to establish a traffic system which will ensure better flow of vehicular traffic. Moreover, few things such as safety of passengers, safety of pedestrians, noise air pollution will be considerably maintained.

2 Literature Survey

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

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

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

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

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

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

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

3 Project Design

3.1 Introduction

After taking into consideration all the factors and parameters regarding this project, we arrive at a three-step Project Design process. It as follows:-

OpenCV technology is used for processing the real-time data from the CCTV footages to obtain the real-time vehicular density count. This is achieved by taking the real-time footage video and then subtracting vehicles from the video background. Two OpenCV lines are used for getting the actual vehicular count which are present in between them at the given instant.

After getting the raw data of the vehicle density, the algorithm is constructed with the help of Weighted Average concept. This helps us to get the information regarding the current signal by taking into account the vehicular density on the remaining traffic signals.

After applying our algorithm on the raw data achieved from the OpenCV step, we can get the final data containing the valuable information of the vehicular density for a given instant of time which will help us determine the dynamic time for each every signal in a single traffic cycle.

3.2 Problem Statement

Efficient regulation of Traffic at traffic signals by taking into consideration the traffic density at every intersection of signals to provide appropriate green signal time (according to the density of vehicles) to corresponding intersections of signals.

3.3 Block Diagram of Adaptive Traffic System

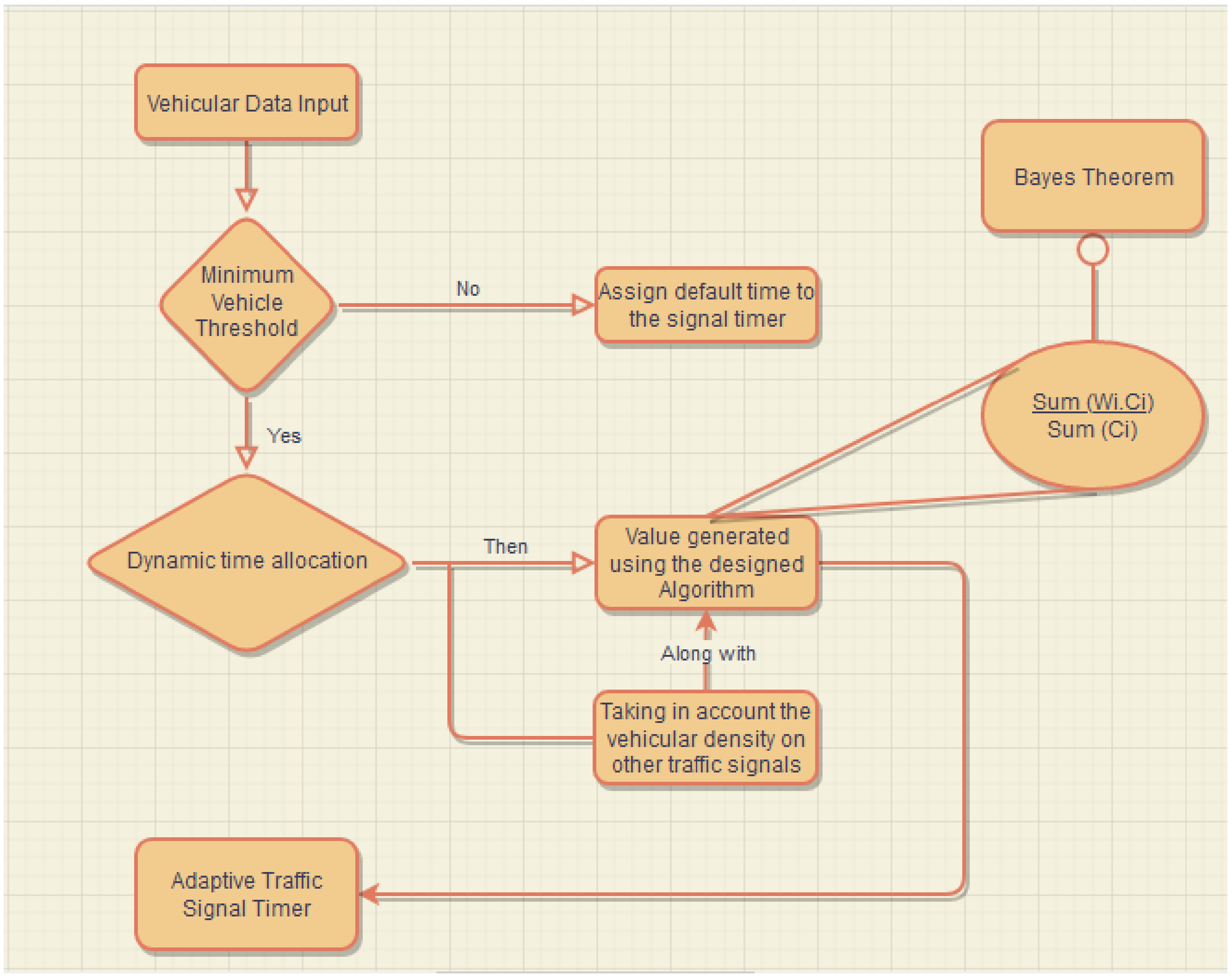

Vehicular Data Input
Minimum Vehicle Threshold
No
Assign default time to the signal timer
Yes
Dynamic time allocation
Then
Value generated using the designed Algorithm
Along with
Taking in account the vehicular density on other traffic signals
Bayes Theorem
Sum (Wi.Ci)
Sum (Ci)
Adaptive Traffic Signal Timer

Figure 1 – Project Flowchart

3.4 Block Diagram of our Algorithm

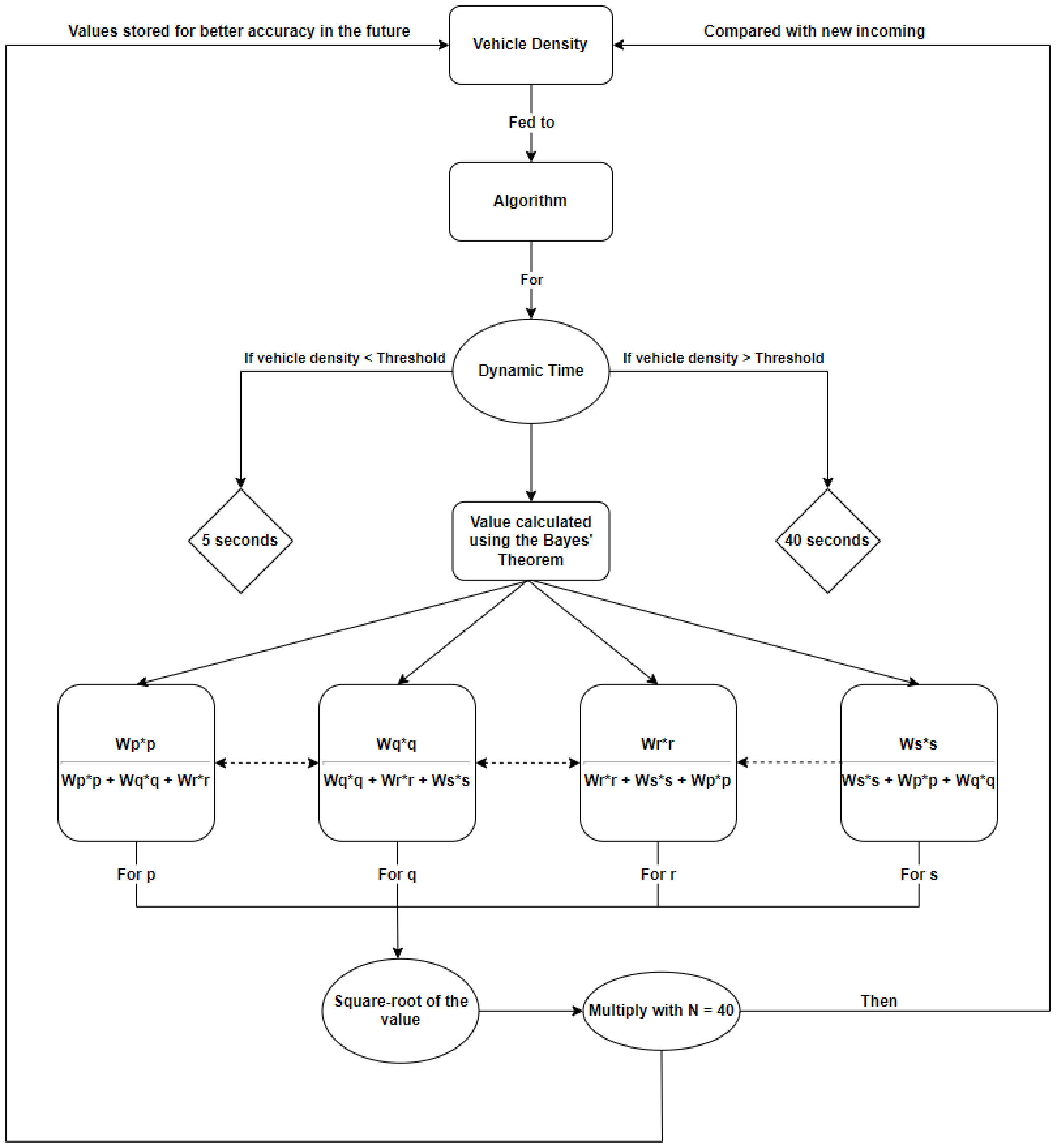


Figure 2 – Flowchart of Algorithm

4 Implementation and Experimentation

4.1 OpenCV Implementation

What is OpenCV?

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications.

Why are we using OpenCV?

We are using it to detect vehicles in motion and increment counter simultaneously. It is used to mark the moving vehicle with a rectangle and draw a line on the road; when the marked vehicle crosses the line the counter will increment. This helps in noting the count of the vehicle which is then fed to the algorithm which in turn gives the green signal time.

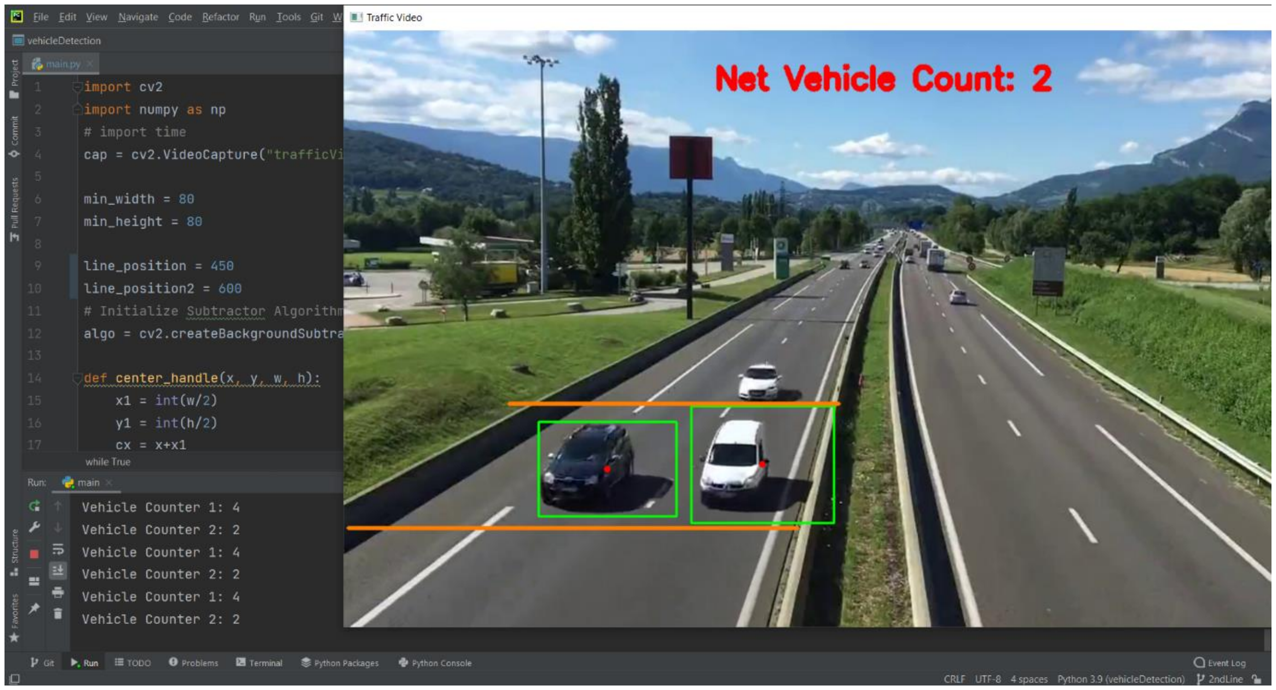
Working of the OpenCV:-

Basically the logic is to increment the count of vehicles when it enters the road; and decrement its count when it exits the signal.

The total count will be given by the formula:-

Count = Incoming vehicles – Outgoing vehicles

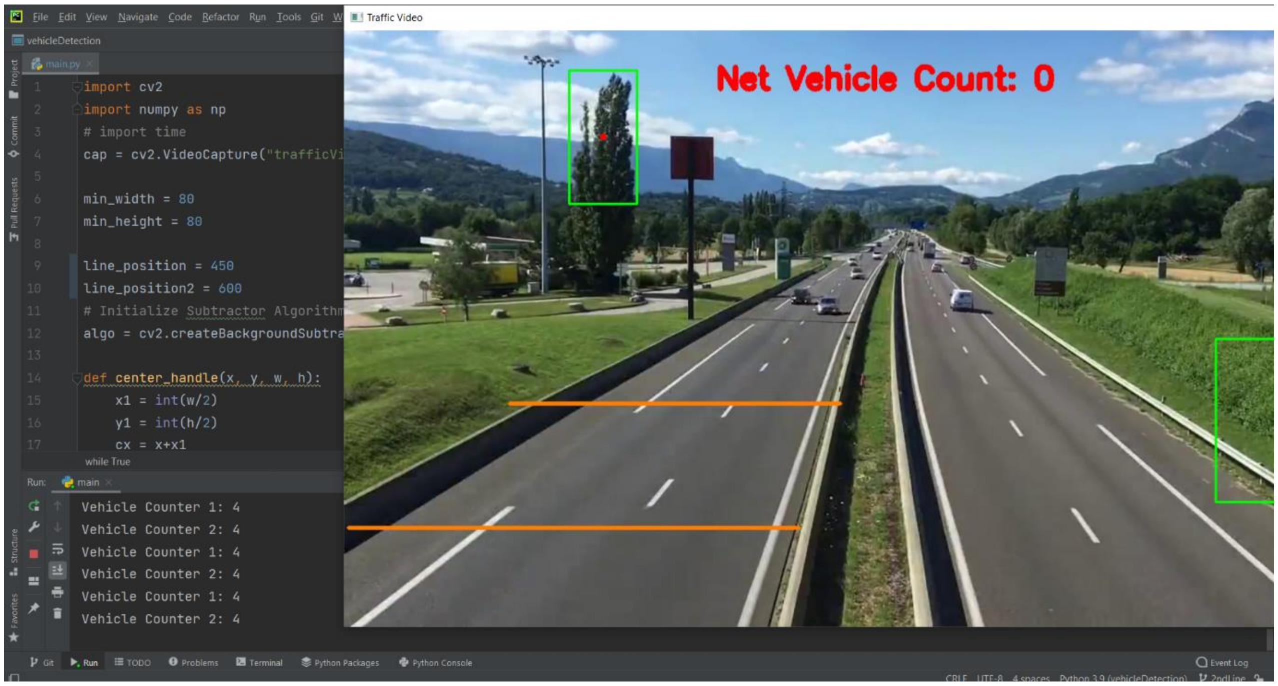
Now when this count is fed to the algorithm it will give us accurate results.


File Edit View Navigate Code Refactor Run Tools Git W
vehicleDetection
main.py
Traffic Video
Net Vehicle Count: 2
Run: main
Git Run TODO Problems Terminal Python Packages Python Console Event Log
CRLF UTF-8 4 spaces Python 3.9 (vehicleDetection) 2ndLine

```
import cv2
import numpy as np
# import time
cap = cv2.VideoCapture("trafficVi

min_width = 80
min_height = 80

line_position = 450
line_position2 = 600
# Initialize Subtractor Algorithm
algo = cv2.createBackgroundSubtra

def center_handle(x, y, w, h):
    x1 = int(w/2)
    y1 = int(h/2)
    cx = x+x1
```

```
Vehicle Counter 1: 4
Vehicle Counter 2: 2
Vehicle Counter 1: 4
Vehicle Counter 2: 2
Vehicle Counter 1: 4
Vehicle Counter 2: 2
```

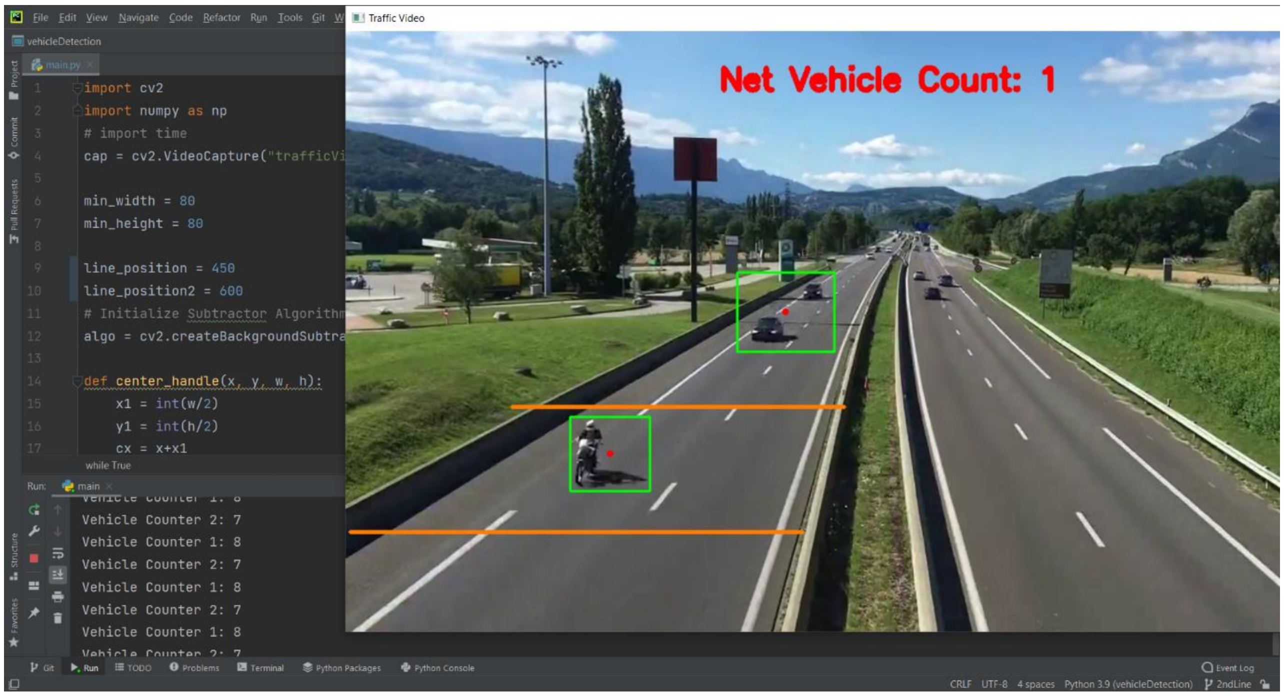
Figure 3 – When there are two vehicles between 2 lines


File Edit View Navigate Code Refactor Run Tools Git W
vehicleDetection
main.py
Traffic Video
Net Vehicle Count: 0
Run: main
Git Run TODO Problems Terminal Python Packages Python Console Event Log
CRLF UTF-8 4 spaces Python 3.9 (vehicleDetection) 2ndLine

```
import cv2
import numpy as np
# import time
cap = cv2.VideoCapture("trafficVi

min_width = 80
min_height = 80

line_position = 450
line_position2 = 600
# Initialize Subtractor Algorithm
algo = cv2.createBackgroundSubtra

def center_handle(x, y, w, h):
    x1 = int(w/2)
    y1 = int(h/2)
    cx = x+x1
```

```
Vehicle Counter 1: 4
Vehicle Counter 2: 4
Vehicle Counter 1: 4
Vehicle Counter 2: 4
Vehicle Counter 1: 4
Vehicle Counter 2: 4
```

Figure 4 – When there are zero vehicles between 2 lines

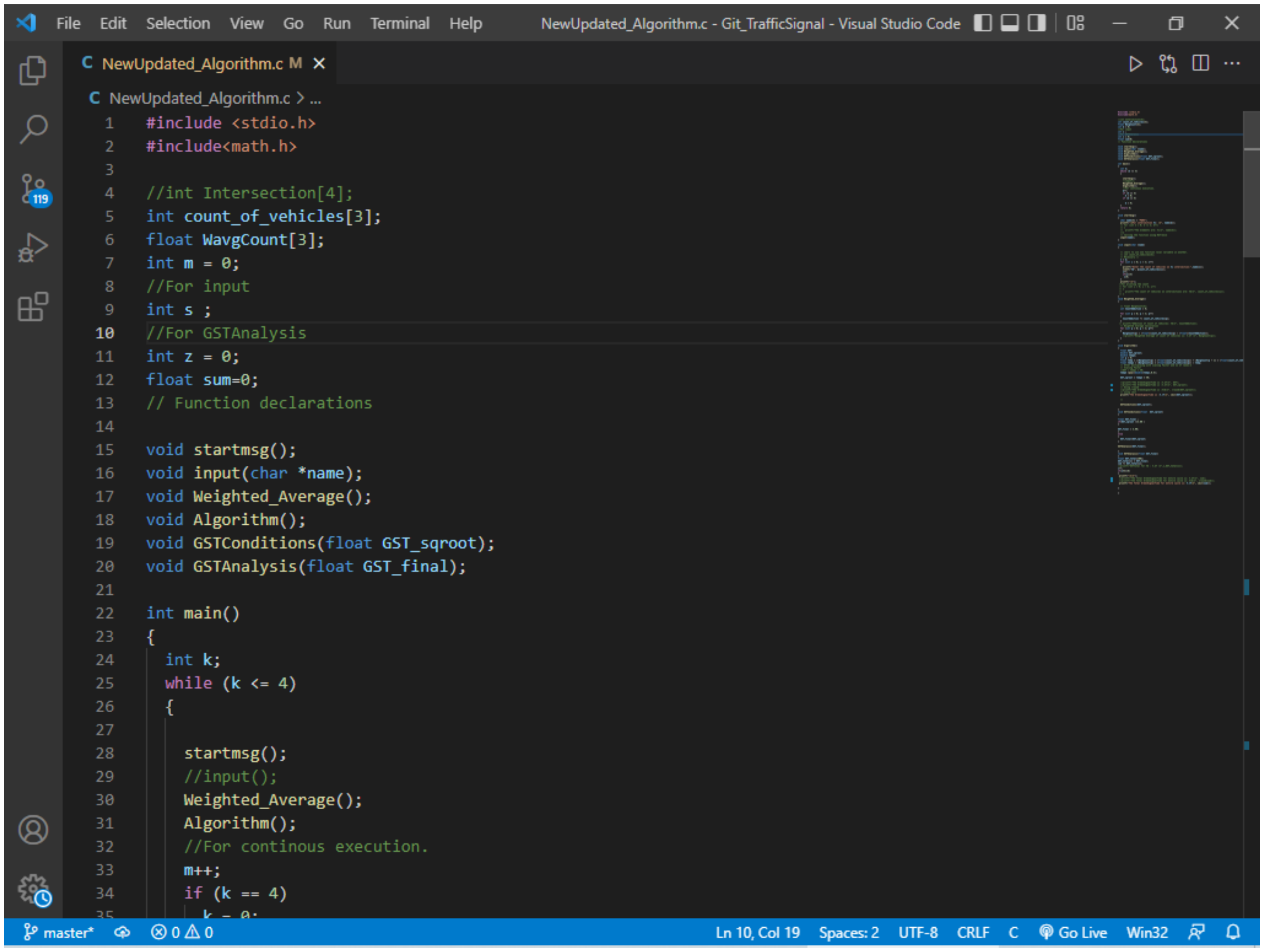

File Edit View Navigate Code Refactor Run Tools Git
vehicleDetection
main.py
Traffic Video
Net Vehicle Count: 1
while True
Run: main
Git Run TODO Problems Terminal Python Packages Python Console
Event Log
CRLF UTF-8 4 spaces Python 3.9 (vehicleDetection) 2ndLine

```
import cv2
import numpy as np
# import time
cap = cv2.VideoCapture("trafficVi

min_width = 80
min_height = 80

line_position = 450
line_position2 = 600
# Initialize Subtractor Algorithm
algo = cv2.createBackgroundSubtra

def center_handle(x, y, w, h):
    x1 = int(w/2)
    y1 = int(h/2)
    cx = x+x1
```

```
Vehicle Counter 2: 7
Vehicle Counter 1: 8
Vehicle Counter 2: 7
Vehicle Counter 1: 8
Vehicle Counter 2: 7
Vehicle Counter 1: 8
Vehicle Counter 2: 7
```

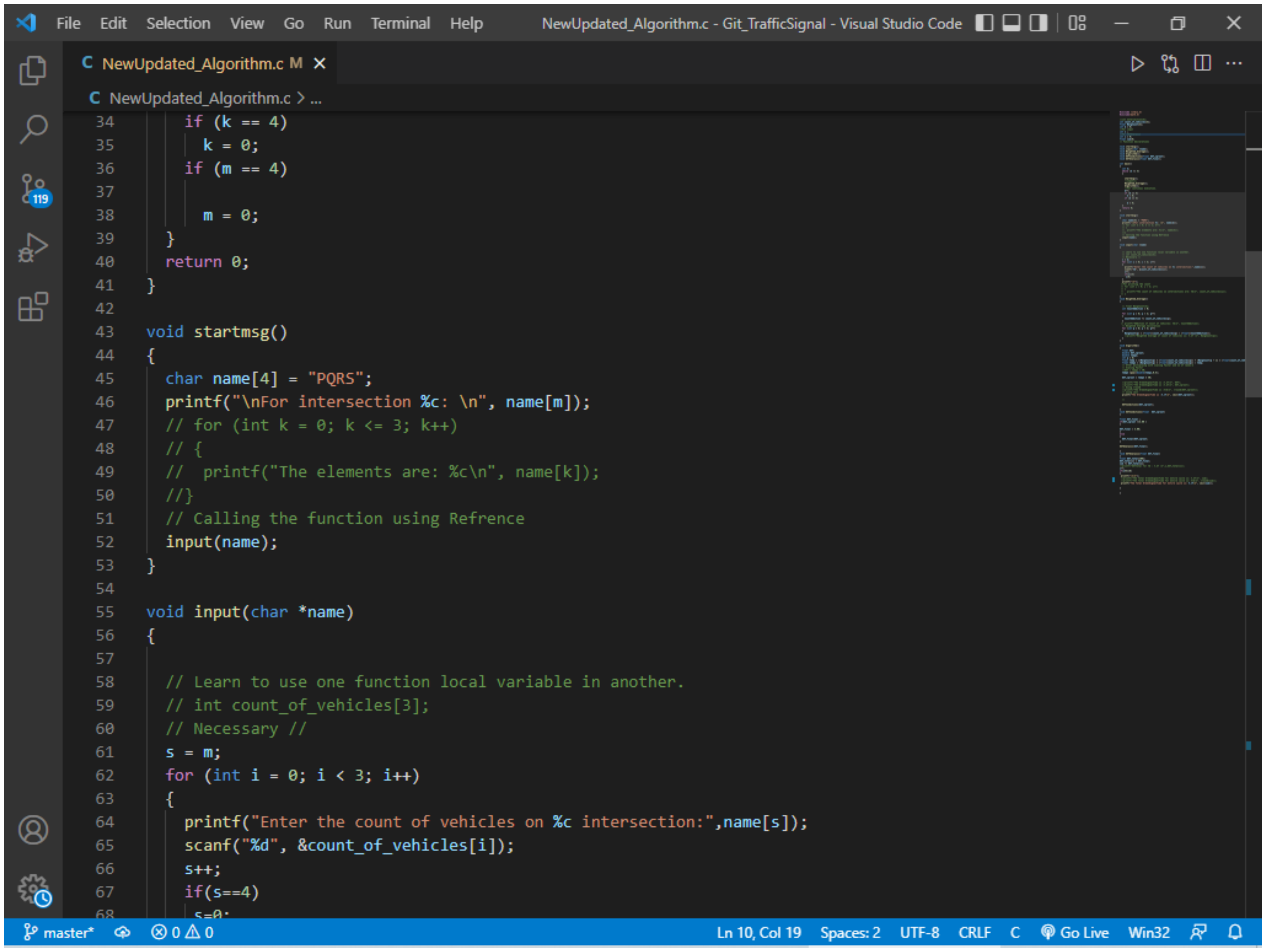
Figure 5 – When there is only 1 vehicle between 2 lines

4.2 Algorithm Code


File Edit Selection View Go Run Terminal Help
NewUpdated_Algorithm.c - Git_TrafficSignal - Visual Studio Code
NewUpdated_Algorithm.c M
NewUpdated_Algorithm.c > ...
master*
0 0
Ln 10, Col 19 Spaces: 2 UTF-8 CRLF C Go Live Win32

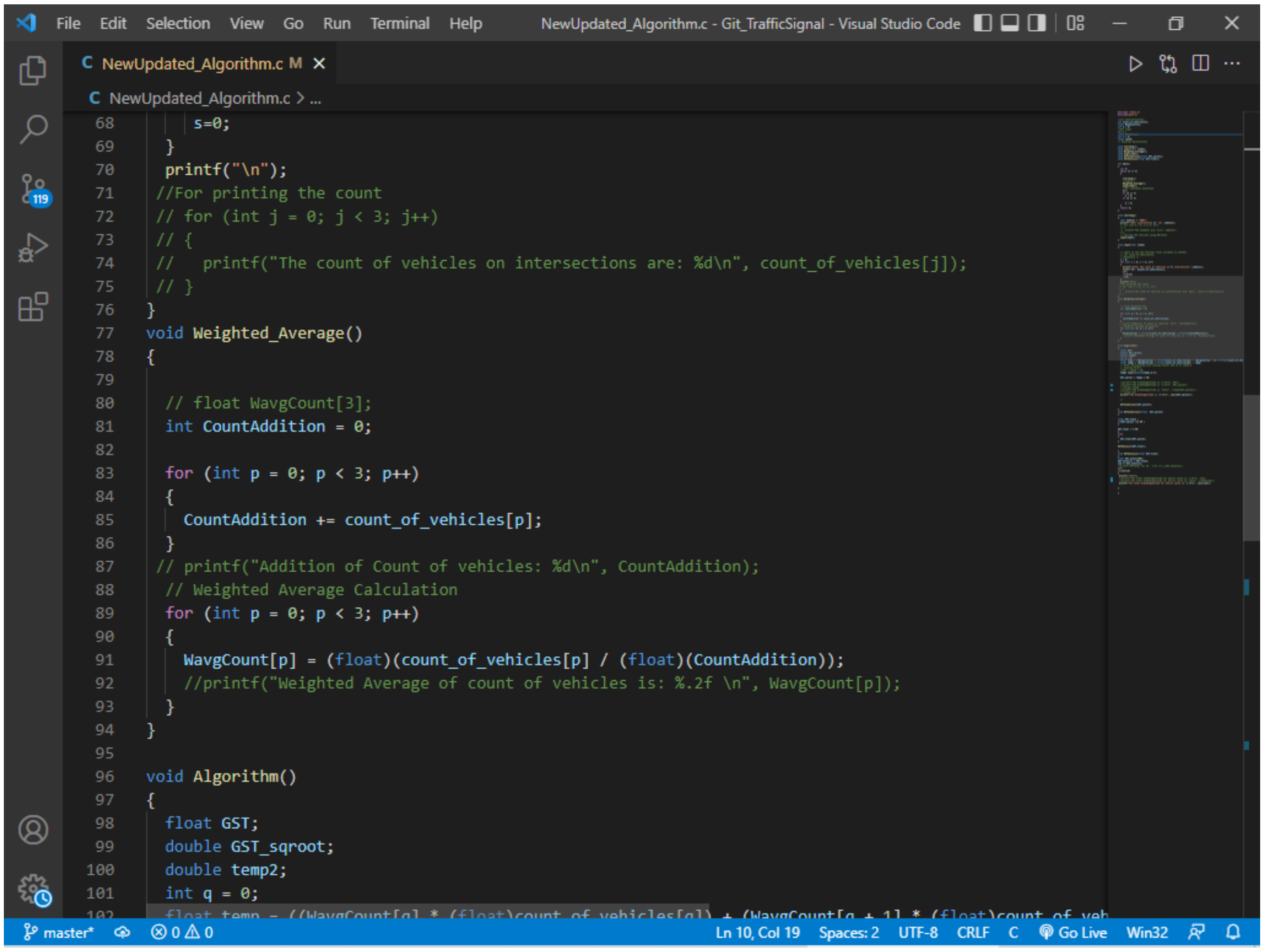
```c
#include <stdio.h>
#include<math.h>

//int Intersection[4];
int count_of_vehicles[3];
float WavgCount[3];
int m = 0;
//For input
int s ;
//For GSTAnalysis
int z = 0;
float sum=0;
// Function declarations

void startmsg();
void input(char *name);
void Weighted_Average();
void Algorithm();
void GSTConditions(float GST_sqroot);
void GSTAnalysis(float GST_final);

int main()
{
  int k;
  while (k <= 4)
  {

    startmsg();
    //input();
    Weighted_Average();
    Algorithm();
    //For continous execution.
    m++;
    if (k == 4)
      k = 0;
```

Figure 6 – Algorithm Code #1


File Edit Selection View Go Run Terminal Help
NewUpdated_Algorithm.c - Git_TrafficSignal - Visual Studio Code
NewUpdated_Algorithm.c M
NewUpdated_Algorithm.c > ...
master*
0 0
Ln 10, Col 19 Spaces: 2 UTF-8 CRLF C Go Live Win32

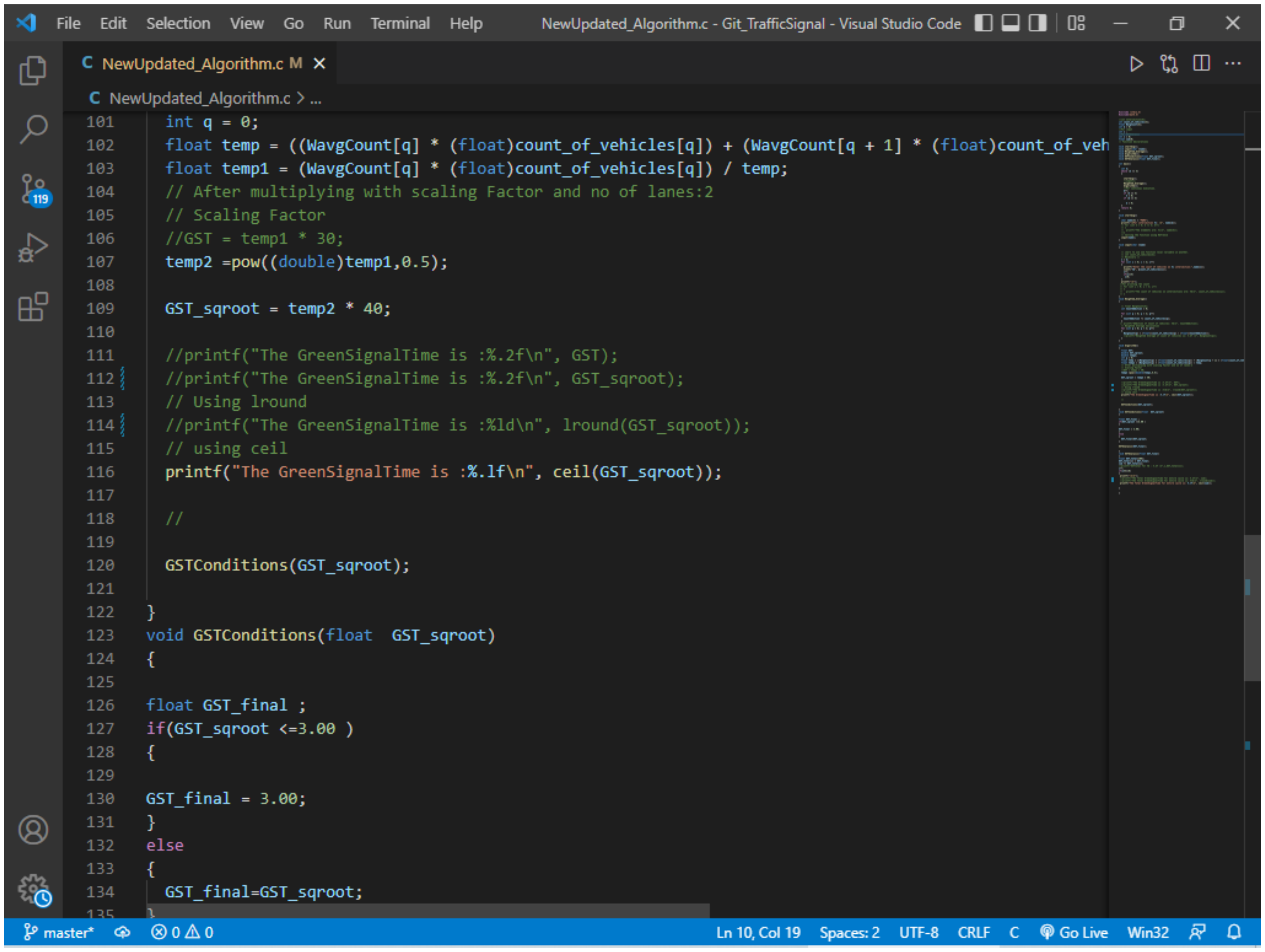
```c
    if (k == 4)
      k = 0;
    if (m == 4)

      m = 0;
  }
  return 0;
}

void startmsg()
{
  char name[4] = "PQRS";
  printf("\nFor intersection %c: \n", name[m]);
  // for (int k = 0; k <= 3; k++)
  // {
  //   printf("The elements are: %c\n", name[k]);
  //}
  // Calling the function using Refrence
  input(name);
}

void input(char *name)
{

  // Learn to use one function local variable in another.
  // int count_of_vehicles[3];
  // Necessary //
  s = m;
  for (int i = 0; i < 3; i++)
  {
    printf("Enter the count of vehicles on %c intersection:",name[s]);
    scanf("%d", &count_of_vehicles[i]);
    s++;
    if(s==4)
      s=0;
```

Figure 7 – Algorithm Code #2



```
      s=0;
    }
    printf("\n");
   //For printing the count
   // for (int j = 0; j < 3; j++)
   // {
   //   printf("The count of vehicles on intersections are: %d\n", count_of_vehicles[j]);
   // }
  }
  void Weighted_Average()
  {

    // float WavgCount[3];
    int CountAddition = 0;

    for (int p = 0; p < 3; p++)
    {
      CountAddition += count_of_vehicles[p];
    }
   // printf("Addition of Count of vehicles: %d\n", CountAddition);
    // Weighted Average Calculation
    for (int p = 0; p < 3; p++)
    {
      WavgCount[p] = (float)(count_of_vehicles[p] / (float)(CountAddition));
      //printf("Weighted Average of count of vehicles is: %.2f \n", WavgCount[p]);
    }
  }

  void Algorithm()
  {
    float GST;
    double GST_sqroot;
    double temp2;
    int q = 0;
    float temp = ((WavgCount[q] * (float)count_of_vehicles[q]) + (WavgCount[q + 1] * (float)count_of_veh
```

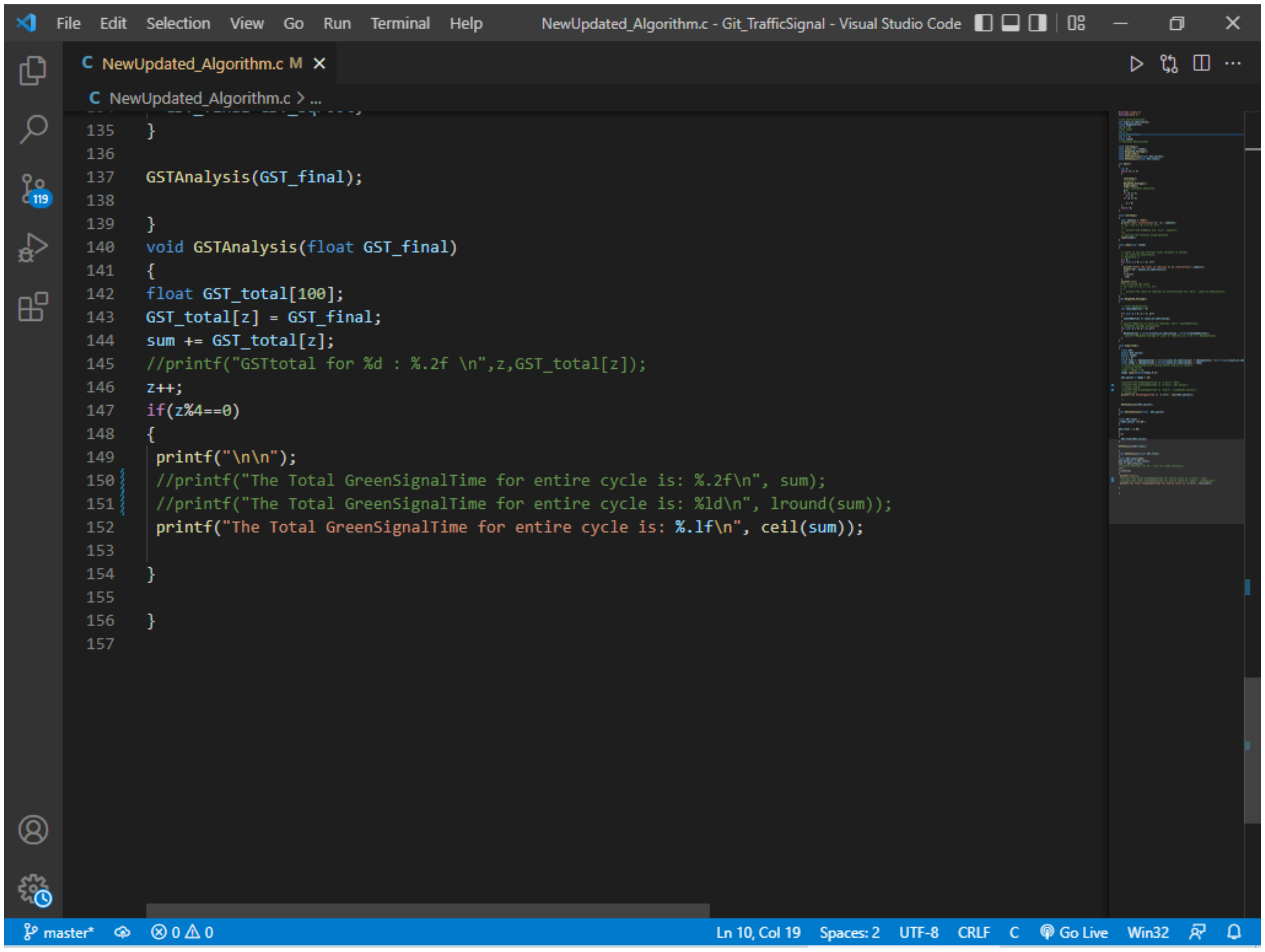
Figure 8 – Algorithm Code #3



File Edit Selection View Go Run Terminal Help NewUpdated_Algorithm.c - Git_TrafficSignal - Visual Studio Code
NewUpdated_Algorithm.c M
NewUpdated_Algorithm.c > ...
master* ⊗ 0 ⚠ 0 Ln 10, Col 19 Spaces: 2 UTF-8 CRLF C Go Live Win32

```c
  int q = 0;
  float temp = ((WavgCount[q] * (float)count_of_vehicles[q]) + (WavgCount[q + 1] * (float)count_of_veh
  float temp1 = (WavgCount[q] * (float)count_of_vehicles[q]) / temp;
  // After multiplying with scaling Factor and no of lanes:2
  // Scaling Factor
  //GST = temp1 * 30;
  temp2 =pow((double)temp1,0.5);

  GST_sqroot = temp2 * 40;

  //printf("The GreenSignalTime is :%.2f\n", GST);
  //printf("The GreenSignalTime is :%.2f\n", GST_sqroot);
  // Using lround
  //printf("The GreenSignalTime is :%ld\n", lround(GST_sqroot));
  // using ceil
  printf("The GreenSignalTime is :%.1f\n", ceil(GST_sqroot));

  //

  GSTConditions(GST_sqroot);

}
void GSTConditions(float  GST_sqroot)
{

float GST_final ;
if(GST_sqroot <=3.00 )
{

GST_final = 3.00;
}
else
{
  GST_final=GST_sqroot;
}
```

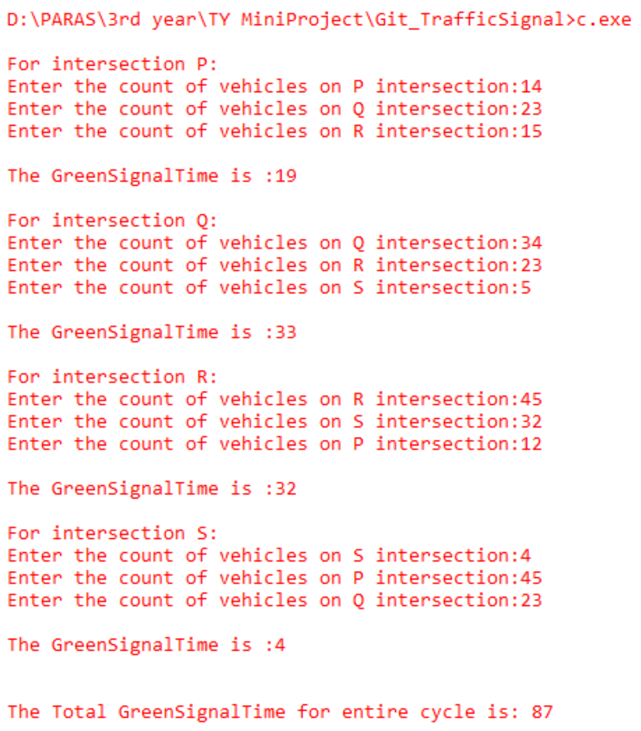
Figure 9 – Algorithm Code #4


File Edit Selection View Go Run Terminal Help
NewUpdated_Algorithm.c - Git_TrafficSignal - Visual Studio Code
NewUpdated_Algorithm.c M
NewUpdated_Algorithm.c > ...
master*
Ln 10, Col 19 Spaces: 2 UTF-8 CRLF C Go Live Win32

```c
}

GSTAnalysis(GST_final);

}
void GSTAnalysis(float GST_final)
{
float GST_total[100];
GST_total[z] = GST_final;
sum += GST_total[z];
//printf("GSTtotal for %d : %.2f \n",z,GST_total[z]);
z++;
if(z%4==0)
{
 printf("\n\n");
 //printf("The Total GreenSignalTime for entire cycle is: %.2f\n", sum);
 //printf("The Total GreenSignalTime for entire cycle is: %ld\n", lround(sum));
 printf("The Total GreenSignalTime for entire cycle is: %.lf\n", ceil(sum));

}

}
```

Figure 10 – Algorithm Code #5

4.3 Algorithm Output



```
D:\PARAS\3rd year\TY MiniProject\Git_TrafficSignal>c.exe

For intersection P:
Enter the count of vehicles on P intersection:14
Enter the count of vehicles on Q intersection:23
Enter the count of vehicles on R intersection:15

The GreenSignalTime is :19

For intersection Q:
Enter the count of vehicles on Q intersection:34
Enter the count of vehicles on R intersection:23
Enter the count of vehicles on S intersection:5

The GreenSignalTime is :33

For intersection R:
Enter the count of vehicles on R intersection:45
Enter the count of vehicles on S intersection:32
Enter the count of vehicles on P intersection:12

The GreenSignalTime is :32

For intersection S:
Enter the count of vehicles on S intersection:4
Enter the count of vehicles on P intersection:45
Enter the count of vehicles on Q intersection:23

The GreenSignalTime is :4


The Total GreenSignalTime for entire cycle is: 87
```

Figure 11 – Algorithm Ouput Data

4.4 Simualtion using Pygame

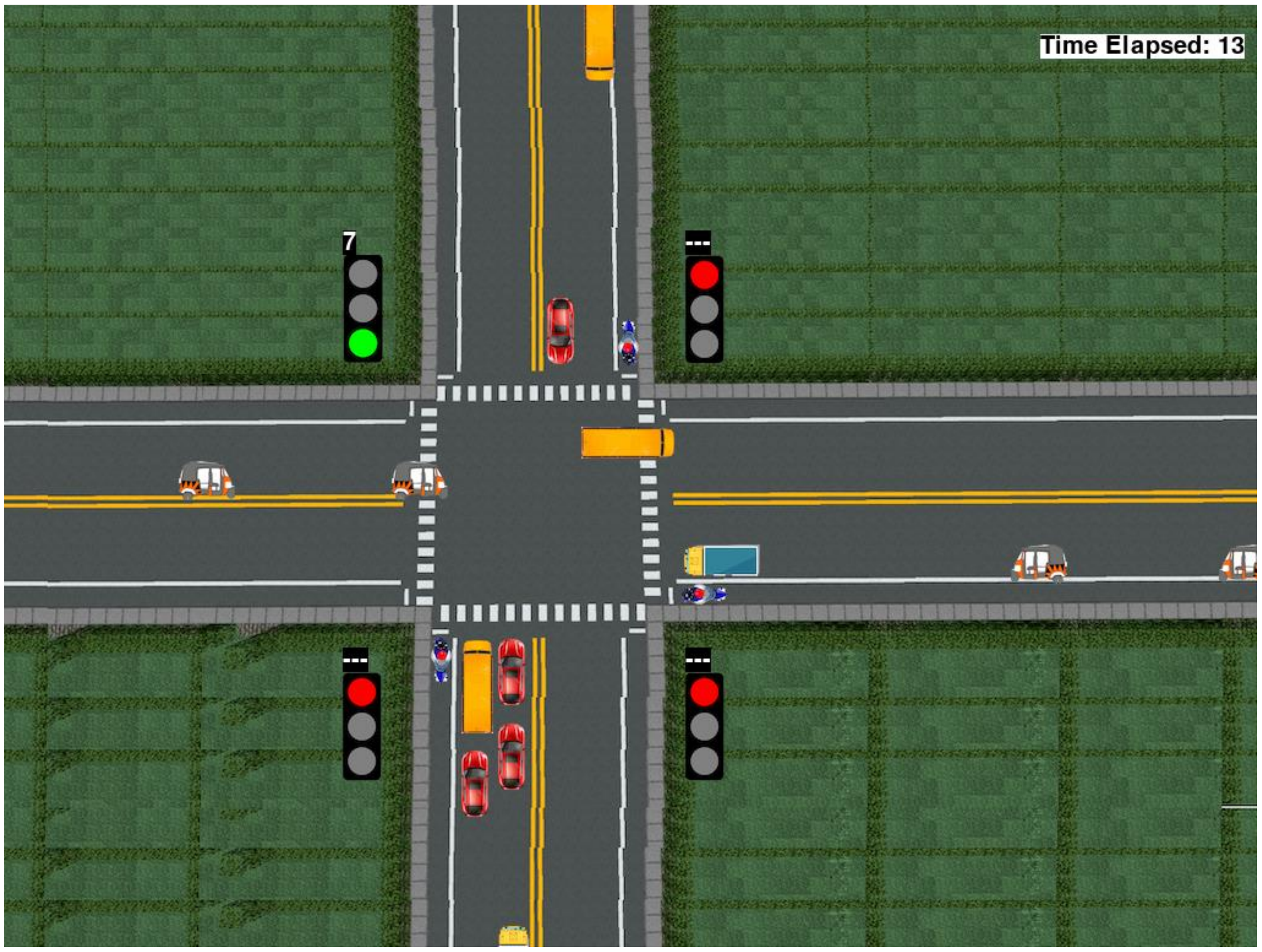


Figure 12 – Simulation #1

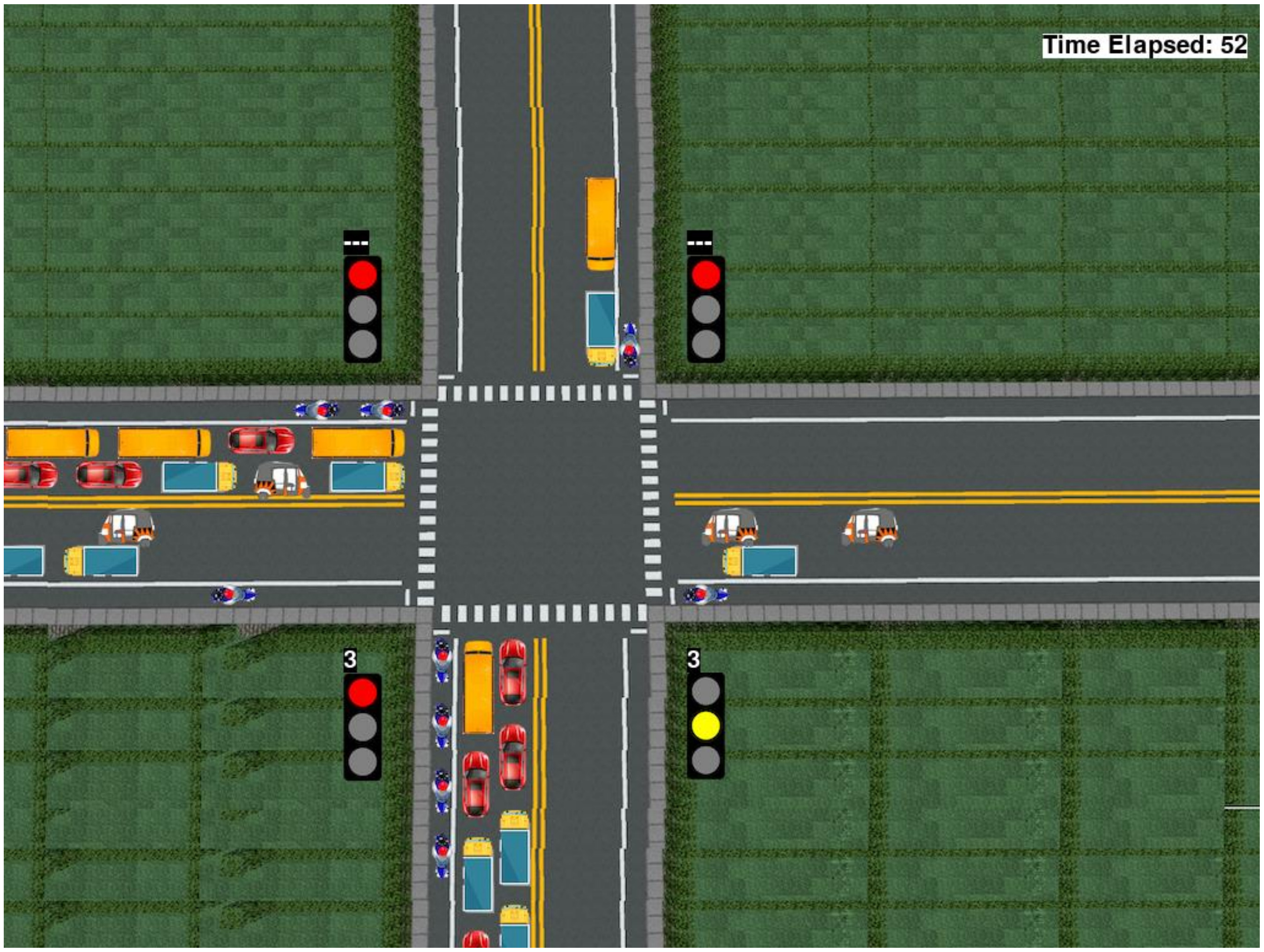
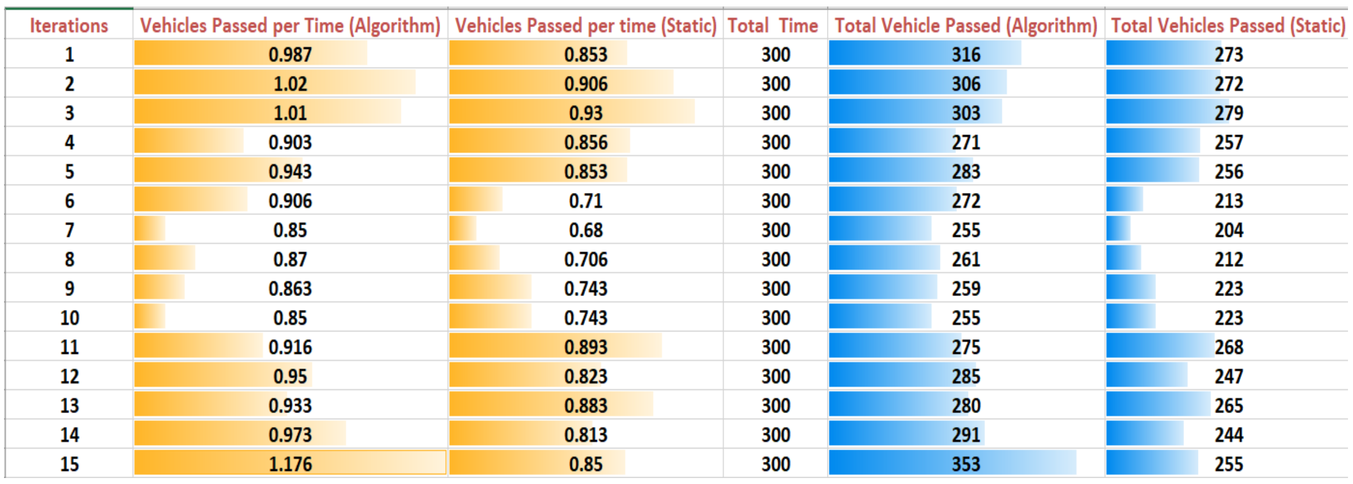


Figure 13 – Simulation #2

4.5 Simualtion Results and Data Analysis



Iterations	Vehicles Passed per Time (Algorithm)	Vehicles Passed per time (Static)	Total Time	Total Vehicle Passed (Algorithm)	Total Vehicles Passed (Static)
1	0.987	0.853	300	316	273
2	1.02	0.906	300	306	272
3	1.01	0.93	300	303	279
4	0.903	0.856	300	271	257
5	0.943	0.853	300	283	256
6	0.906	0.71	300	272	213
7	0.85	0.68	300	255	204
8	0.87	0.706	300	261	212
9	0.863	0.743	300	259	223
10	0.85	0.743	300	255	223
11	0.916	0.893	300	275	268
12	0.95	0.823	300	285	247
13	0.933	0.883	300	280	265
14	0.973	0.813	300	291	244
15	1.176	0.85	300	353	255

Figure 14 – Vehicular Data Analysis

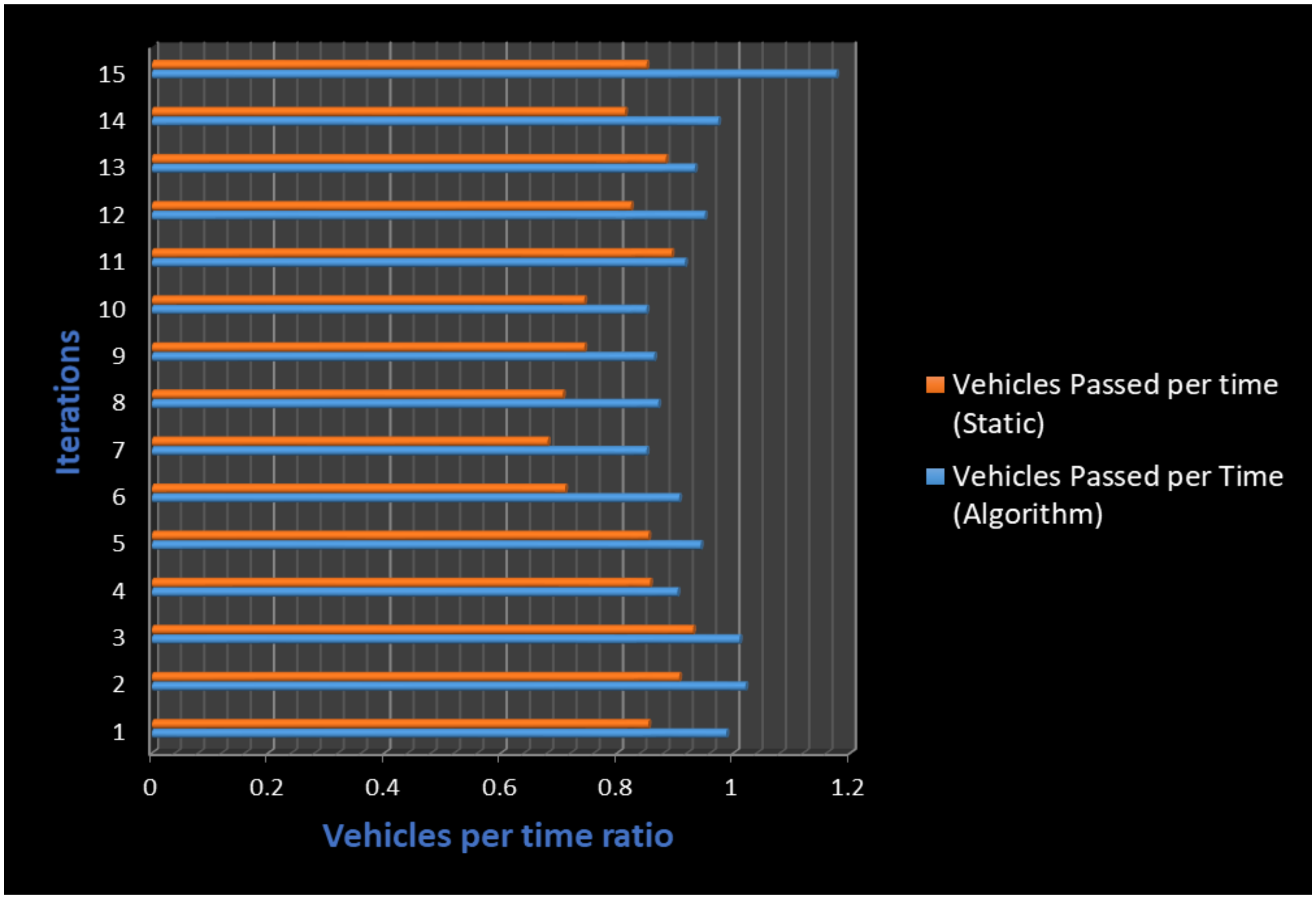


Figure 15 – No. of Iterations Vs. Vehicle per time ratio

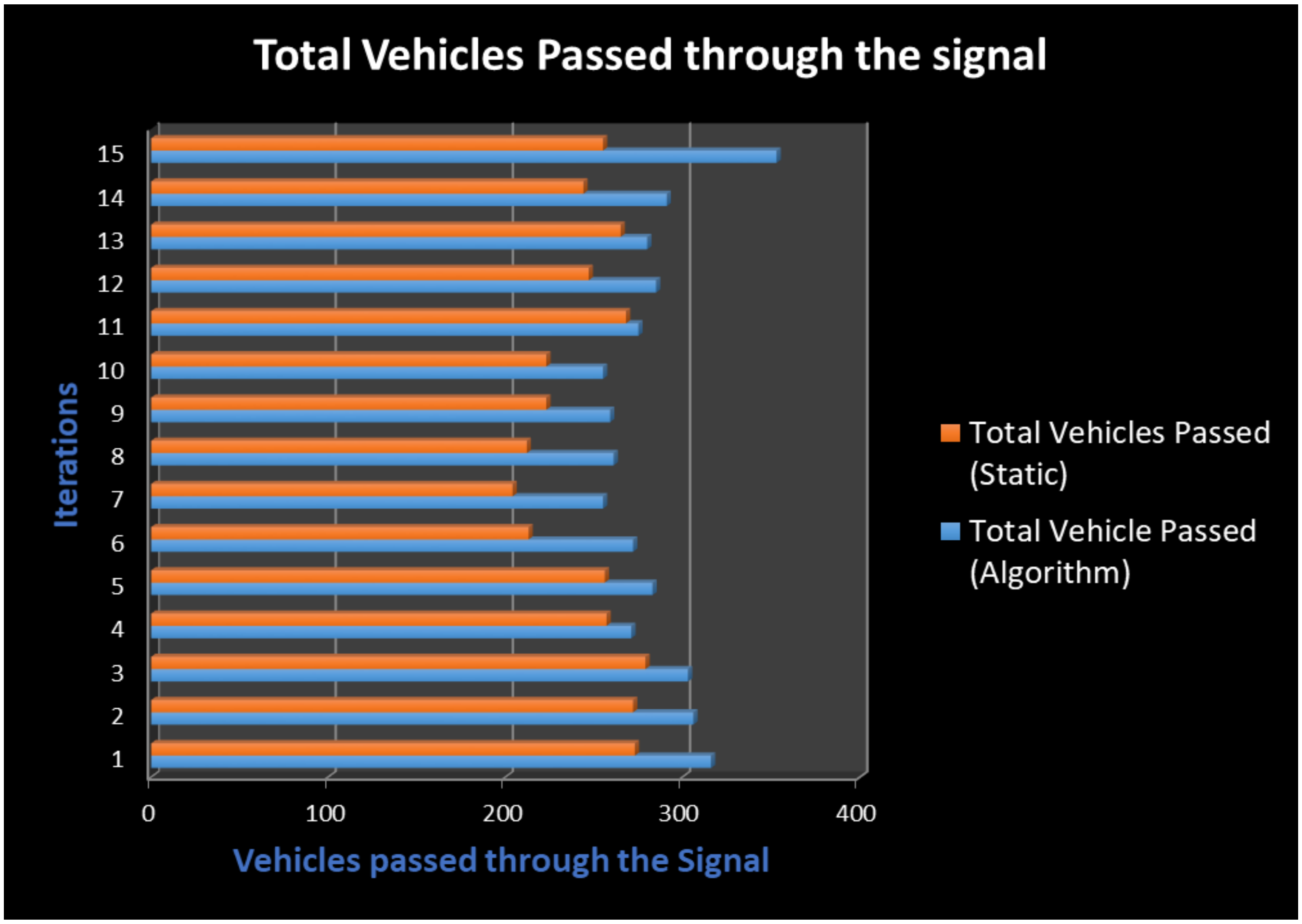


Figure 16 – Total Vehicles passed through the Traffic Signal

5 Conclusions and Scope of futher work

5.1 Conclusion

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

According to simulation results, the system shows about 20% improvement over the current system in terms of the number of vehicles crossing the intersection, which is a significant improvement. With further calibration and overall imporovements, this system can be optimised to perform even better.

The cost required to deploy the system is negligible as footage from CCTV cameras from traffic signals is used, which requires no additional hardware in most cases, as intersections with heavy traffic are already equipped with such cameras. Only minor alignment may need to be performed. Thus, the proposed system can thus be integrated with the CCTV cameras in major cities in order to facilitate better management of traffic.

4.2 Scope of further work

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

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

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

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

Referrences

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

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

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

4. A. A. Zaid, Y. Suhweil and M. A. Yaman, "Smart controlling for traffic light time," 2017 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT), Aqaba, 2017, pp. 1-5, doi: 10.1109/AEECT.2017.8257768.

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

6. A. Kanungo, A. Sharma and C. Singla, "Smart traffic lights switching and traffic density calculation using video processing," 2014 Recent Advances in Engineering and Computational Sciences (RAECS),Chandigarh, 2014, pp. 1-6, doi: 10.1109/RAECS.2014.6799542.

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

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