S^2 - Smart Signaling

A Major 1 Project Report submitted in partial fulfilment of the

requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

with IBM specialization in CYBER SECURITY AND FORENSICS

Submitted by

DHANISHTHA AWASTHI	R134215026	500046214	CSE-CSF	7th Sem
KUNAL MOHAN BANSAL	R134215039	500046496	CSE-CSF	7th Sem
SHUBHAM KUMAR GUPTA	R134215077	500045997	CSE-CSF	7th Sem

Under the guidance of

Dr. Ajay Prasad
Cyber Platform
SCS

UPES
UNIVERSITY WITH A PURPOSE

SCHOOL OF COMPUTER SCIENCE

UNIVERSITY OF PETROLEUM & ENERGY STUDIES

November - 2018.

Approved By

Dr. Ajay Prasad
Project Head

Dr. Neelu Jyoti Ahuja
Head of Department

DECLARATION

I hereby declare that this submission is my own and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other Degree or Diploma of the University or other Institute of Higher learning, except where due acknowledgement has been made in the text.

Dhanishtha Awasthi (Enroll No. R134215026)

Kunal Mohan Bansal (Enroll No. R134215039)

Shubham Kumar Gupta (Enroll No. R134215077)

CERTIFICATE

This is to certify that the project titled S^2- Smart Signaling submitted by Dhanishtha Awasthi (Enroll. No. R134215026), Kunal Mohan Bansal (Enroll. No. R134215039), Shubham Kumar Gupta (Enroll. No. R134215077) to the University of Petroleum & Energy Studies, for the award of the degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING is a Bona fide record of project work carried out by them under my supervision and guidance. The content of the project, in full or parts have not been submitted to any other Institute or University for the award of any other degree or diploma.

Date: **28th November, 2018.**

Dr. Ajay Prasad	Dr. Neelu Jyoti Ahuja
Mentor	Head of Department
Cyber Platform | SCS	Dept. of Systematics |SCS

ACKNOWLEDGEMENT

I would like to show my sincere gratitude to my supervisors Dr. Neelu Jyoti Ahuja & Dr. Ajay Prasad for providing me invaluable guidance, comments and suggestions throughout the course of the project. I would like to especially thanks Christalin Nelson sir for constantly motivating me throughout my work.

This report would not have been possible without their constant support. Also, I would like to thank University of Petroleum and Energy Studies for providing me the platform and facilities, along with the opportunity to work on this project.

Finally, I would like to thank my family and friends for their understanding and support while completing this project.

ABSTRACT

India is a developing country. Time is an essential resource for development of nation. Unfortunately, millions of people waste hours of their precious time in traffic. There are many reasons for poor traffic management. One such reason is operation of traffic lights being static instead of being adaptive difference of traffic between lanes it is responsible for. This causes green signal for empty lane and red signal for filled one, most of the times.

In this Project we will tackle this issue and will try to make a smart signal which will decide by itself on the basis of data set whether to show green or red signal to a particular lane.

The main purpose of our project is the traffic congestion control, so as to save the precious time of the people stuck in traffic.

The project deals with the Machine Learning algorithm for the decision of the amount of time for which the signal must be green or red for the particular traffic light post on the cross.

The camera on the post will be connected to the server and will capture the photograph of the road and do the background subtraction, so as to finally get the amount of congestion in the traffic. Then the calculation for the amount of time for which the signal must be green will be fed as data set of if-else conditions, resulting to the learning by self, using the machine learning approach.

Keywords: Traffic, smart signal, machine learning.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Background of the project

Delhi, the national capital region and the most important among all, is facing a vast issue related to traffic signaling system [2]. According to the traffic lightning protocol, the controller shows red light to lanes for a certain interval of time and thus makes it difficult for the people to communicate daily to their destinations if in the case the road is empty though the light is green for that lane and the one filled more than half has still red light. This issue is detected and reported to the government which is continuous procession of finding the solution.

Market Dynamics

The project is made in hue of finding solution to a governmental problem which may proof beneficial to not only Delhi but other states of India too. Thus may make huge market value not only in India but in foreign too.

So, serving as a foundation for becoming a fast and productive country.

Technology

Machine learning and Open CV are the two major approaches nowadays for the smart development. The above stated approaches when combined with JAVA will result into the maximum profitable approach. Our project comprises of the implementation of these techniques and so the major goal of our project to make the traffic signaling system smart becomes easier.

Users

The main users of our project will be not only the people of a particular state, but can be from anywhere within the country. If used by other nations too, the traffic control would become much easier. The smartness of the system will help the people appointed on duty on crosses by relieving them from pressure of unnecessary nuisance created by people around them to give them green signal being getting late for their work in jams.

Chapter 2

PROBLEM STATEMENT

Design a smart solution to meet one of the main issues for traffic jams i.e. static nature of traffic signals which are unable to differentiate between 0 and 100% traffic. This solution should also include capabilities for central server to hold control in case of emergency or a rare event. Central server must be able to change traffic light at any time.

Chapter 3

LITERATURE REVIEW

1. Smart traffic light control system. [1] is the white paper published in 2016, denoting an IR system and microcontroller-based solution to give basic. Currently no such system exists in India and is the need of the hour to build so.
2. Smart Traffic Light System (IoT) [3] is the GitHub project based on the IOT approach for the signal sensing systems.
3. Smart Traffic Signal [4] Traditional traffic signal system only gives instructions to stop and not to vehicle driver. But if someone is breaking the signal then this system is not able to catch them and there are chances of taking bribe. Thus, to increase the security of traffic signal and to reduce human efforts and to avoid the bribery we are introducing smart traffic signal system through this MINI PROJECT.

None of the above stated projects developed so far were able to generate the central server through which the traffic must be detected and congestion might be controlled, and so our project will deal with making of the traffic signals smart in themselves instead of using any extra external hardware system such as IOT devices and others.

Chapter 4

OBJECTIVES

1. The main objective of our research and development for this project is to build an intelligent and smart traffic light signaling software.
2. Also, we need to design a solution that would proof itself as not only a solution to requirement design but to requirement perceptions too.
3. In this project, designing an intelligent traffic control system by studying traffic light configurations and signaling protocols will be the main motive throughout the designing of the solution.
4. The machine learning approach will be inhibited and integrated while exploring the solutions to the problem statement and reaching to the conclusion with the effective one.
5. The outcome must be fully or partially able to resolve traffic congestion problem because of static lightning technologies.
6. The large datasets will be incorporated to reach the maximum precision level of the outcome and working of the project.
7. Background subtraction will help in getting the amount of traffic on the road whose image is captured.

Chapter 5

DESIGN

Algorithms used

i. Machine Learning.
ii. Image Analytics
iii. Data Set Collection.
iv. Server connection.

Machine learning Algorithm

Step 1: Get the amount of congestion being calculated for the traffic on a road.

Step 2: Using the if-else statement-based data set calculate the amount of time for which signal must be green.

Step 3: Self-learn for the next time i.e. if only 7 secs are required for two cars to pass then from next time the time allotted must be equal to 7 secs + All the red time. If the vehicles are able to be passed before these 7 secs then next time allotted time must be 7 secs.

Image Analytics Algorithm

Step 1: Capture an image in open CV with java using camera.

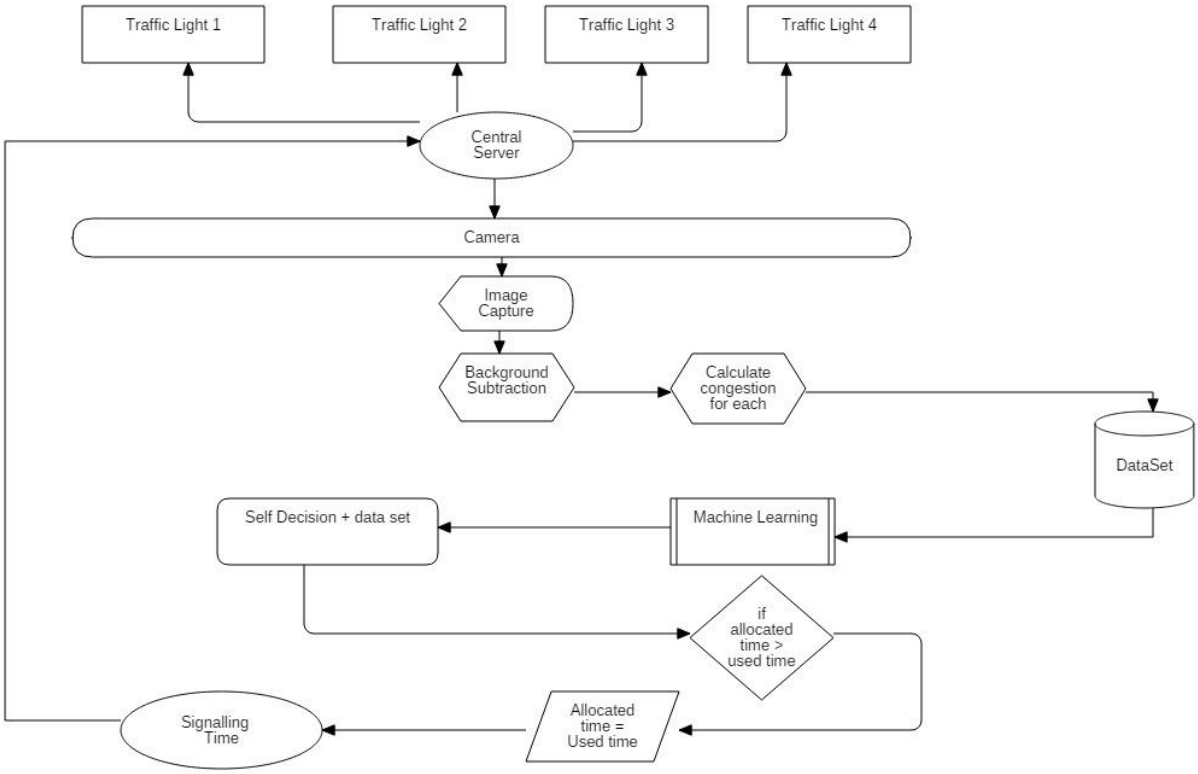
Step 2: Apply the background subtraction mechanism using contour detection technique.

Data Set Collection Algorithm

Step 1: Categorize images data using if-else statement i.e. if <20% then 10secs, if <30% then 15 secs, and so on be the time allocated.

Server Connection Algorithm

Step 1: Enable socket connection with various ports open for the traffic signaling system of the central server.



b

FIG 5.1 FLOW CHART

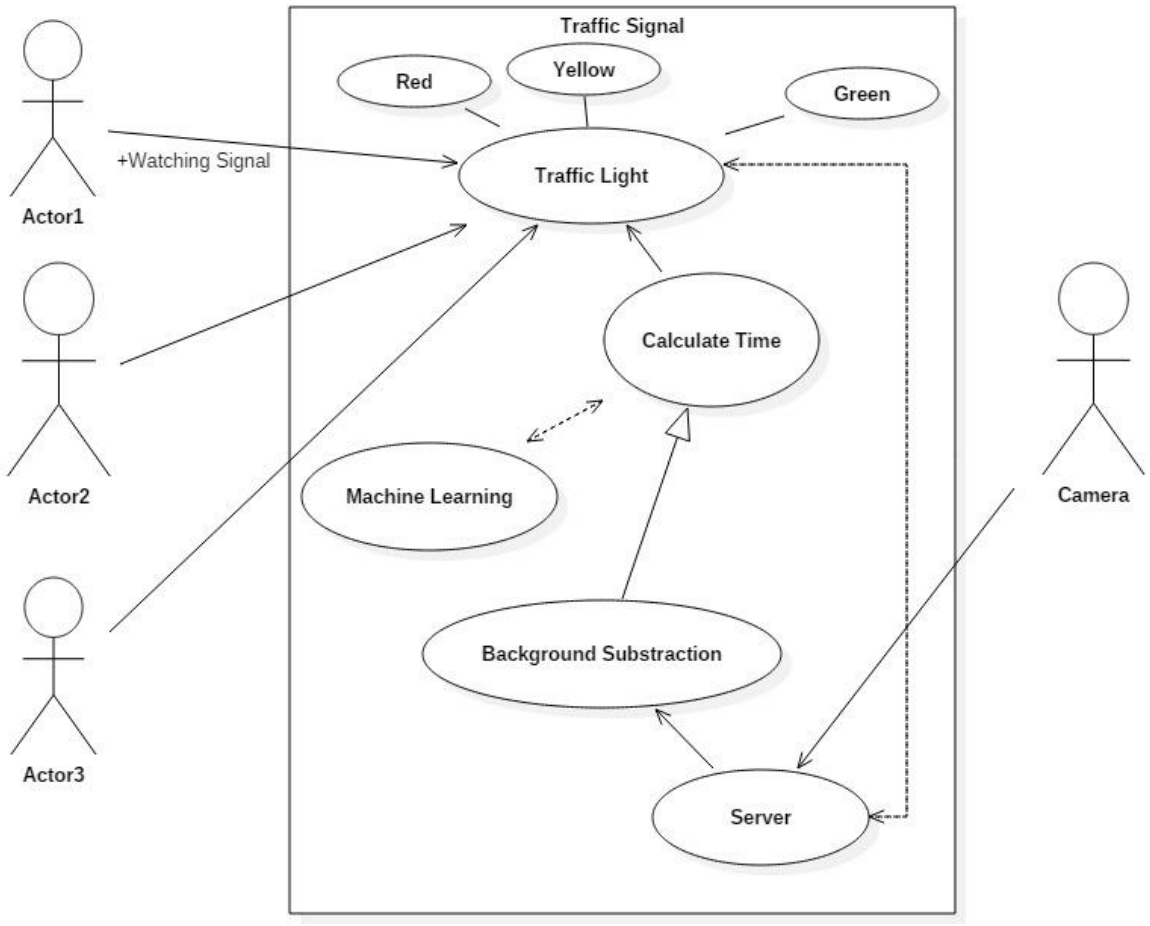


FIG 5.2 USE CASE DIAGRAM

Prerequisites:

1. The Camera captured image from continuous video.
2. Cross road signaling system data set.
3. Data collection of Signal phases and cycles.
4. The timings of yellow signal and all red time.

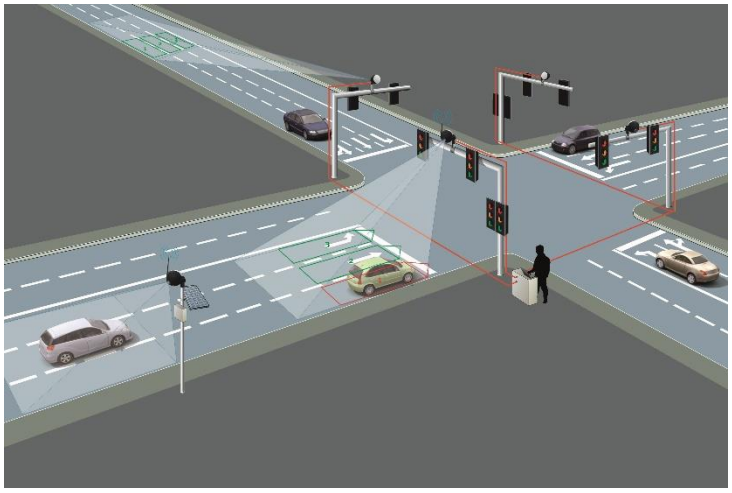


FIG 5.3 TRAFFIC LIGHT POST WITH CAMERA.

Chapter 6

IMPLEMENTATION

Our Project is basically divided into two main modules

Module 1

For the traffic lights to get the signal, the Server will be started. The start of Server will mark the opening of 4 ports namely –

- 6061
- 6062
- 6063
- 6064 and the ports will be used for four different Traffic lights for cross on roads.

Module 2

The second module is the Background Subtraction approach where using the camera, background subtraction will occur. The background subtraction algorithm will help us finding the appropriate amount of time for which the signal must be green.

FIG 6.1 BACKGROUND SUBTRACTION

Module 3

The third and last module is the Machine learning module, where the analytics will be provided for the time calculation and green light signaling sequence.

Post Module

Now the three rules need to be followed

1. No two traffic lights should be green simultaneously.
2. The Machine learning should understand by itself that if previously time for two vehicles to cross was allotted to be 10secs but they used only 7secs next time it should try for 7secs and learn by self in the process.
3. There should be yellow light and all red time kept in mind before next port to generate a green signal to prevent vehicle collision.

Future Enhancements

The future enhancement for this project will include, certain analytics, improved background subtraction, implementation through Proof of Concept using 3D camera or 4 different cameras, on real time systems.

Output Screens

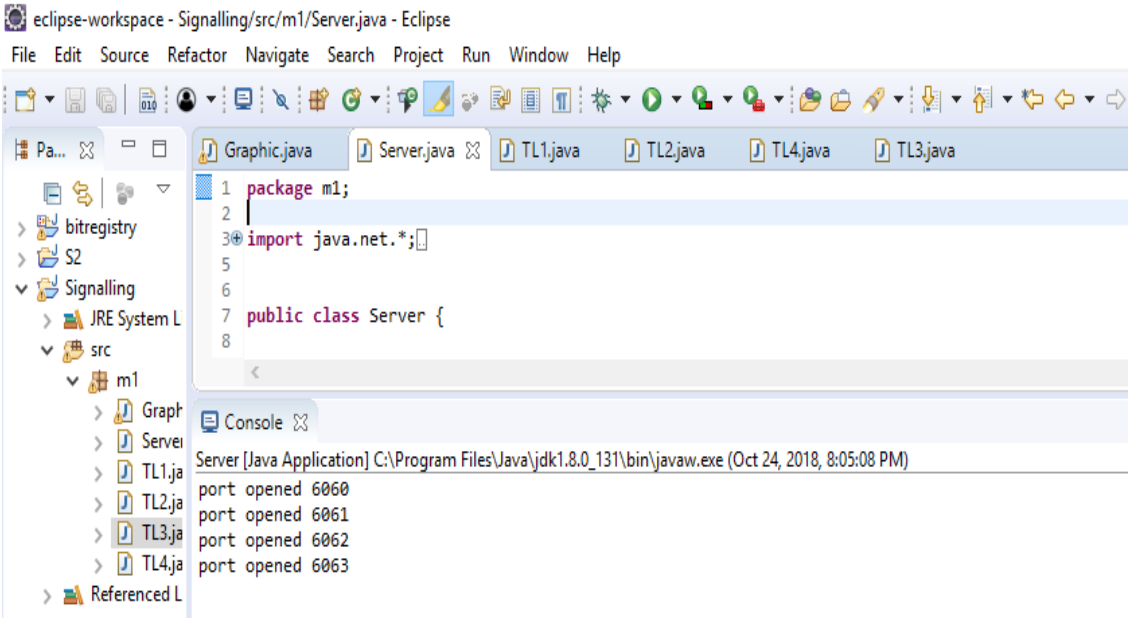


FIG 6.2 SERVER IS STARTED

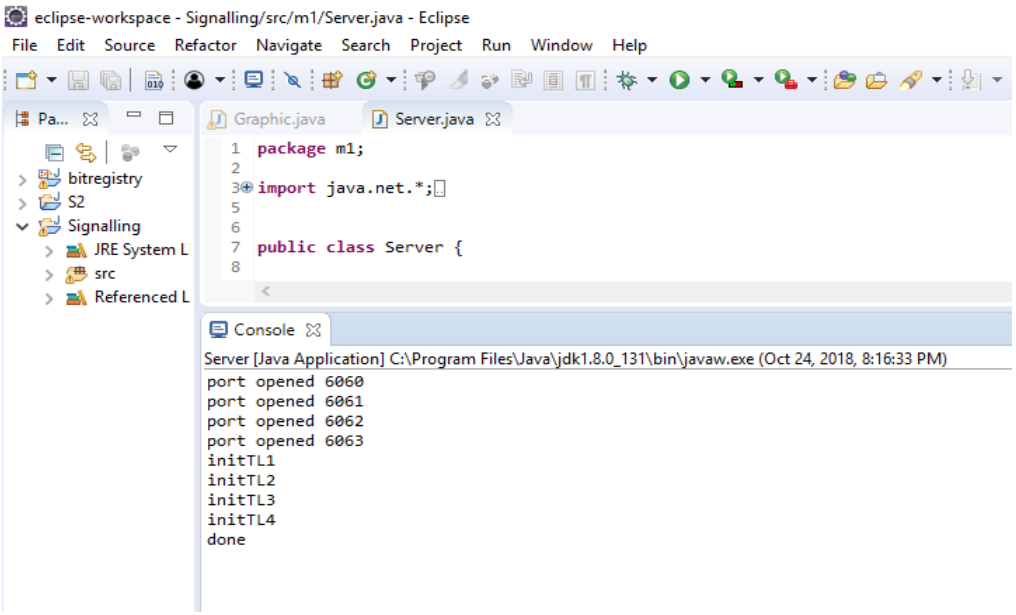


FIG 6.3 T1 / T2 / T3 / T4 ARE STARTED ONE BY ONE

```
TL4 [Java Application] C:\Program Files\Java\jre1.8.0_192\bin\javaw.exe (28-Nov-2018, 1:30:42 AM)
connecting to server on port6063
conneted tolocalhost/127.0.0.1:6063 me = 32TL4
RED
green 22
```

FIG 6.4 TL4 GREEN AT A TIME AT INTERVAL SEND BY SERVER (TL4 ,22 SECS)

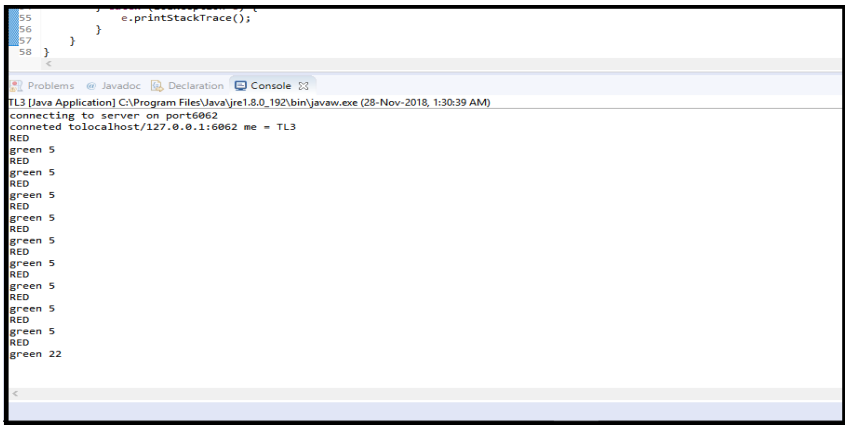


FIG 6.5 TL3 CHANGE IN DURATION FROM 5 SEC TO 22 ACCORDING TO FOREGROUND MASK

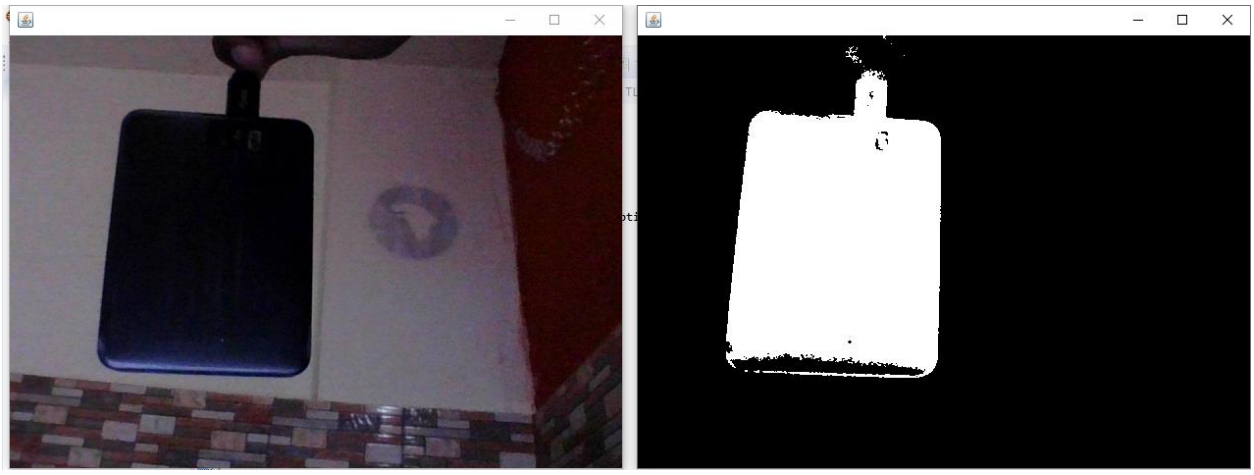


FIG 6.6 BACKGROUND SUBTRACTION DONE BY PROCESSFRAME METHOD OF TCOUNTER CLASS

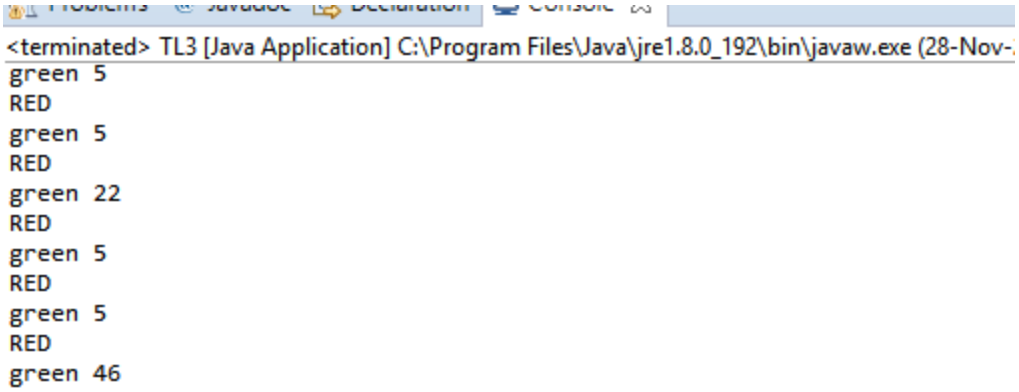


FIG 6.7 : TL CHANGE IN TIME (46 SECS, TL3)

Chapter 7

SYSTEM REQUIREMENTS

Software Requirements:

1. Eclipse
2. Open CV
3. Java

Hardware Requirements:

1. Minimum requirements and configuration support.

Chapter 8

SCHEDULE

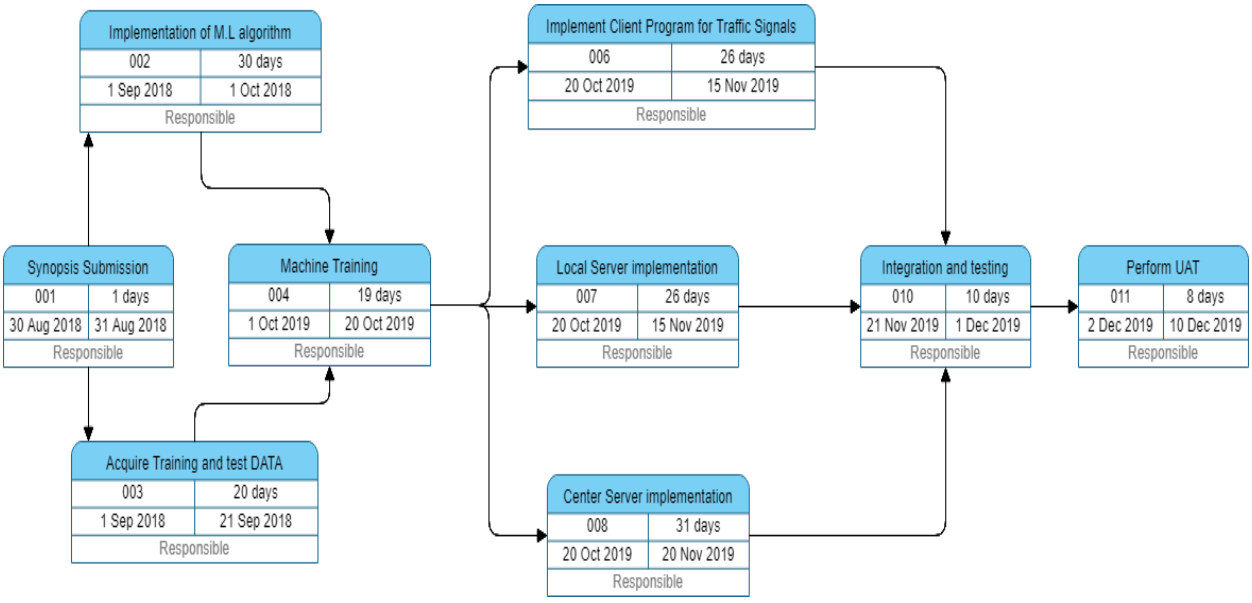


TABLE 1 PERT CHART

Chapter 9

CONCLUSIONS

At last, we would like to conclude our project by simply claiming that we were successful in delivering the intended results and achieving the objectives laid at the start of our project. This project will simply proof itself in decreasing the congestion in the traffic on the crosses. This project will proof itself valuable for not only Indian Government but also inter-nation-wide.

Chapter 10

REFERENCES

[1] **Conference Paper (PDF Available)** · April 2016 with 23,706 Reads

DOI: 10.1109/EECEA.2016.7470780
Conference: Conference: 2016 *Third International Conference on Electrical, Electronics, Computer Engineering and their Applications (EECEA)*

.

[2] https://timesofindia.indiatimes.com/city/delhi/10-years-on-plan-to-use-ai-to-manage-city-traffic-takes-off/articleshow/63495347.cms

[3] https://github.com/salamzantout/Smart-Traffic-Light

[4] https://www.researchgate.net/publication/262941886_Smart_Traffic_Signal

Appendix A

Code Snippets

Server.java

```
package sSquare;

import java.net.*;
import java.io.*;

public class Server extends Thread  {

    static int turn = 0;

    public static void main(String[] args) throws IOException, InterruptedException  {
        // TODO Auto-generated method stub

        TCounter tc = new TCounter();
        Thread.currentThread();



            /*int time = tc.calculate();
            System.out.println(time);
            Thread.sleep(5000);
            */



        ServerSocket[] TL = new ServerSocket[4];


        try {
            for (int i = 0; i < 4; i++) {
                TL[i] = new ServerSocket(6060 + i);
                System.out.println("port opened " + (6060 + i));
            }
        }   catch (IOException e) {
                e.printStackTrace();
        }


        DataInputStream[] in = new DataInputStream[4];
        DataOutputStream[] out = new DataOutputStream[4];
        try {
            Socket[] l = new Socket[4];
            for (int i = 0; i < 4; i++) {
                l[i] = TL[i].accept();
                in[i] = new DataInputStream(l[i].getInputStream());
                out[i] = new DataOutputStream(l[i].getOutputStream());
                String str = in[i].readUTF();
                System.out.println("init" + str);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }



         while (true) {
             try {
                 out[turn].writeUTF("green " + (new Integer(tc.calculate())).toString());
                 String str = in[turn].readUTF();
                 System.out.println(str);

             } catch (Exception e) {
                 e.printStackTrace();
             }
             turn = (turn + 1) % 4;
         }
    }
}
```

TCounter.java

```java
package sSquare;

import java.awt.image.BufferedImage;
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStream;

import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;

import org.opencv.core.Core;
import org.opencv.core.CvType;
import org.opencv.core.Mat;
import org.opencv.core.MatOfByte;
import org.opencv.core.Size;
import org.opencv.highgui.Highgui;
import org.opencv.highgui.VideoCapture;
import org.opencv.imgproc.Imgproc;
import org.opencv.video.BackgroundSubtractorMOG;

public class TCounter implements Runnable {
    static {
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
    }
    static int factor = 0;
    Mat imag;
    static Mat imag2;

    int signal;
    JFrame[] jframe = new JFrame[2];
    JLabel[] vidpanel = new JLabel[2];
      Thread runner;
      public TCounter() {
        this.runner = new Thread(this);
        this.runner.start();
      }
      static double pre = 0,f=1 , pre_t = 0,ff = 1.0;
      public  int calculate() // calculate area of white space in frame returned by getFrame method.
        {

            int time =0;
            int nwp =0;

            try {
            nwp = Core.countNonZero(imag2);//number of white pixel
            }
            catch(Exception e)
            {
                System.out.println("tumse na ho payega");
            }
            double tp =  imag2.size().width * imag2.size().height;      //total pixels
            double percentage = nwp/tp;
            percentage *= 100;
            int per = (int)percentage;
            int c = 0;
            System.out.println("nwp="+nwp+" tp="+tp+" percentage ="+ percentage + " per ="+per + " c="+c);
            if((per>=0)&&(per<25))
                c = 0;
            if((per>=25)&&(per<50))
                c= 1;
            if((per>=50)&&(per<75))
                c= 2;
            if((per>=75)&&(per<=100))
                c= 3;

            switch(c)
            {
            case(0):
                time = (int) ( 5 + (c*f));
                break;
            case(1):
                time = (int) ( 20 + (c*f) * 2);
                break;
            case(2):
                time = (int) ( 40 + (c*f) * 3);
                break;
            case(3):
                time =(int) (  60 + (c*f) * 4);
                break;
            default:
                break;
            }


            cal_f(time);

            return time;

        }
```

```
92
93      private void cal_f(int time) {
94        if(time<pre_t)
95        {
96            if(pre == -1 )  {
97
98                ff = ff/10;
99                ff = (ff<0.01) ? 0.01 : ff ;
100               f -= ff;
101           }
102           else if(pre == 1) {
103
104               ff = 0.1;
105               f -=  ff;
106           }
107           pre = -1;
108       }
109       if (time >pre_t)
110       {
111           if(pre == -1 )  {
112               ff  = 0.1;
113               f +=  ff;
114           }
115           else if(pre == 1) {
116
117               ff = ff/10;
118               ff = (ff<0.01) ? 0.01 : ff ;
119               f += ff;
120
121           }
122           pre = 1;
123       }
124   }
125   @Override
126     public void run() {
127         jframe[1] = new JFrame();
128           jframe[0] = new JFrame();
129         System.out.println("Running....");
130           for (int counter = 0; counter < 2; counter++) {
131
132               jframe[counter].setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
133               vidpanel[counter] = new JLabel();
134               jframe[counter].setContentPane(vidpanel[counter]);
135               jframe[counter].setSize(640, 480);
136               jframe[counter].setVisible(true);
137               jframe[counter].setLocation(640 * (counter % 2), 0);
138
139           }
140
141
142
143
144           Mat frame = new Mat();
145           Mat outerbox = new Mat();
146
147           Mat diffframe = null;
148           Mat temponframe = null;
149           // ArrayList<Rect> array = new ArrayList<Rect>();
150
151           BackgroundSubtractorMOG mBGsub = new BackgroundSubtractorMOG(3, 4, 0.8);
152           VideoCapture camera = new VideoCapture();
153           camera.open(0);
154           Size sz = new Size(640, 480);
155           int i = 0;
156
157           // KalmanFilter kf = new KalmanFilter(2,1);
158
159           if (!camera.isOpened()) {
160               System.out.println("Cannot open the camera, Please try again later");
161           }
162
163
164       synchronized(this) {
165           while (true) { // while loop start
166               if (camera.read(frame)) {
167                   Imgproc.resize(frame, frame, sz);
168                   imag = frame.clone();
169                   imag2 = frame.clone();
170                   processFrame(camera, frame, imag2, mBGsub);
171                   ////////////////////////////////////////
172                   outerbox = new Mat(frame.size(), CvType.CV_8UC1);
173                   Imgproc.cvtColor(frame, outerbox, Imgproc.COLOR_BGR2GRAY);
174                   //Imgproc.erode(outerbox, outerbox, Imgproc.getStructuringElement(Imgproc.MORPH_RECT, new Size(8, 8)));
175                   Imgproc.erode(frame, outerbox, Imgproc.getStructuringElement(Imgproc.MORPH_RECT, new Size(8, 8)));
176                   Imgproc.dilate(outerbox, outerbox, Imgproc.getStructuringElement(Imgproc.MORPH_RECT, new Size(8, 8)));
177                   Imgproc.GaussianBlur(outerbox, outerbox, new Size(3, 3), 0);
178
```

```
                    if (i == 0) {
                        jframe[1].setSize(frame.width(), frame.height());
                        diffframe = new Mat(outerbox.size(), CvType.CV_8UC3);
                        temponframe = new Mat(outerbox.size(), CvType.CV_8UC3);
                        diffframe = outerbox.clone();
                    }
                    if(i==1) {
                        processFrame(camera, temponframe, diffframe, mBGsub);
                        Imgproc.adaptiveThreshold(diffframe, diffframe, 255, Imgproc.ADAPTIVE_THRESH_MEAN_C,
                                Imgproc.THRESH_BINARY_INV, 5, 2);

                    }



                    ImageIcon image = new ImageIcon(Mat2bufferedImage(imag));
                    vidpanel[0].setIcon(image);
                    temponframe = outerbox.clone();
                    ImageIcon image2 = new ImageIcon(Mat2bufferedImage(imag2));
                    vidpanel[1].setIcon(image2);


                }
            }           }   // while loop ends

    }

    protected static void processFrame(VideoCapture capture, Mat mRgba, Mat mFGMask, BackgroundSubtractorMOG mBGSub) {
          capture.retrieve(mRgba, Highgui.CV_CAP_ANDROID_COLOR_FRAME);
          //capture.retrieve(mRgba ,Highgui.CV_CAP_ANDROID_COLOR_FRAME_GREY_FRAME);
          mBGSub.apply(mRgba, mFGMask, 0);
          //Imgproc.cvtColor(mFGMask, mRgba, Imgproc.COLOR_GRAY2BGRA, 4);

      }

      private static BufferedImage Mat2bufferedImage(Mat image) {
          MatOfByte bytemat = new MatOfByte();
          Highgui.imencode(".jpg", image, bytemat);
          byte[] bytes = bytemat.toArray();
          InputStream in = new ByteArrayInputStream(bytes);
          BufferedImage img = null;
          try {
              img = ImageIO.read(in);
          } catch (IOException e) {
              e.printStackTrace();
          }
          return img;
      }

}
```

T1.java

```java
package sSquare;

import java.net.*;
import java.io.*;

public class TL1 {

    public static void main(String[] args) {
        String serverName = "localhost"; // TO BE TAKE AS COMMAND LINE ARGUMENT
        int port = 6060; // TO BE TAKE AS COMMAND LINE ARGUMENT
        String me = "TL1";
        String[] ms = new String[2];
        int time = 0; // TO BE CALCLATED USING M.L
        try {
            System.out.println("connecting to server on port" + port);
            Socket client = new Socket(serverName, port);

            System.out.println("conneted to" + client.getRemoteSocketAddress() + " me = " + me);

            OutputStream outToServer = client.getOutputStream();
            DataOutputStream out = new DataOutputStream(outToServer);
            System.out.println("RED");
            out.writeUTF(me);
            InputStream inFromServer = client.getInputStream();
            DataInputStream in = new DataInputStream(inFromServer);
            while (true) {
                // for(int i=0;i<3;i++) {

                String mess = in.readUTF();
                String green = "green";
                String exit = "exit";
                ms = mess.split(" ");
                time = Integer.parseInt(ms[1]);
                if (green.equals(ms[0])) {
                    System.out.println(mess);

                    try {
                        Thread.sleep(1000 * time);
                    } catch (InterruptedException ex) {
                        Thread.currentThread().interrupt();
                    }
                    System.out.println("RED");

                    out.writeUTF("done");
                } else if (exit.equals(ms[0])) {
                    System.out.println("EXITING! PROGRAM EXECUTED SUCCESSFULLY!! ;) ");
                    out.writeUTF("exit-ack");
                    break;
                }
            }

            client.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

TL2.java

```java
package sSquare;

import java.net.*;
import java.io.*;

public class TL2 {

    public static void main(String[] args) {
        String serverName = "localhost"; // TO BE TAKE AS COMMAND LINE ARGUMENT
        int port = 6061; // TO BE TAKE AS COMMAND LINE ARGUMENT
        String me = "TL2";
        String[] ms = new String[2];
        int time = 10; // TO BE CALCLATED USING M.L
        try {
            System.out.println("connecting to server on port" + port);
            Socket client = new Socket(serverName, port);

            System.out.println("conneted to" + client.getRemoteSocketAddress() + " me = " + me);

            OutputStream outToServer = client.getOutputStream();
            DataOutputStream out = new DataOutputStream(outToServer);
            System.out.println("RED");
            out.writeUTF(me);
            InputStream inFromServer = client.getInputStream();
            DataInputStream in = new DataInputStream(inFromServer);
            while (true) {
                // for(int i=0;i<3;i++) {

                String mess = in.readUTF();
                String green = "green";
                String exit = "exit";
                ms = mess.split(" ");
                time = Integer.parseInt(ms[1]);
                if (green.equals(ms[0])) {
                    System.out.println(mess);

                    try {
                        Thread.sleep(1000 * time);
                    } catch (InterruptedException ex) {
                        Thread.currentThread().interrupt();
                    }
                    System.out.println("RED");

                    out.writeUTF("done");
                } else if (exit.equals(ms[0])) {
                    System.out.println("EXITING! PROGRAM EXECUTED SUCCESSFULLY!! ;) ");
                    out.writeUTF("exit-ack");
                    break;
                }
            }

            client.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

TL3.java

```java
package sSquare;

import java.net.*;

public class TL3 {
    ;
    public static void main(String[] args) {
        String serverName = "localhost"; // TO BE TAKE AS COMMAND LINE ARGUMENT
        int port = 6062; // TO BE TAKE AS COMMAND LINE ARGUMENT
        String me = "TL3";
        String[] ms = new String[2];
        int time = 10; // TO BE CALCLATED USING M.L
        try {
            System.out.println("connecting to server on port" + port);
            Socket client = new Socket(serverName, port);

            System.out.println("conneted to" + client.getRemoteSocketAddress() + " me = " + me);

            OutputStream outToServer = client.getOutputStream();
            DataOutputStream out = new DataOutputStream(outToServer);
            System.out.println("RED");
            out.writeUTF(me);
            InputStream inFromServer = client.getInputStream();
            DataInputStream in = new DataInputStream(inFromServer);
            while (true) {
                // for(int i=0;i<3;i++) {

                String mess = in.readUTF();
                String green = "green";
                String exit = "exit";
                ms = mess.split(" ");
                time = Integer.parseInt(ms[1]);
                if (green.equals(ms[0])) {
                    System.out.println(mess);

                    try {
                        Thread.sleep(1000 * time);
                    } catch (InterruptedException ex) {
                        Thread.currentThread().interrupt();
                    }
                    System.out.println("RED");

                    out.writeUTF("done");
                } else if (exit.equals(ms[0])) {
                    System.out.println("EXITING! PROGRAM EXECUTED SUCCESSFULLY!! ;) ");
                    out.writeUTF("exit-ack");
                    break;
                }
            }

            client.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

TL4.java

```java
package sSquare;

import java.net.*;

public class TL4 {

    public static void main(String[] args) {
        String serverName = "localhost"; // TO BE TAKE AS COMMAND LINE ARGUMENT
        int port = 6063; // TO BE TAKE AS COMMAND LINE ARGUMENT
        String me = "TL4";
        String[] ms = new String[2];
        int time = 10; // TO BE CALCLATED USING M.L
        try {
            System.out.println("connecting to server on port" + port);
            Socket client = new Socket(serverName, port);

            System.out.println("conneted to" + client.getRemoteSocketAddress() + " me = 32"
                    + "" + me);

            OutputStream outToServer = client.getOutputStream();
            DataOutputStream out = new DataOutputStream(outToServer);
            System.out.println("RED");
            out.writeUTF(me);
            InputStream inFromServer = client.getInputStream();
            DataInputStream in = new DataInputStream(inFromServer);
            while (true) {
                // for(int i=0;i<3;i++) {

                String mess = in.readUTF();
                String green = "green";
                String exit = "exit";
                ms = mess.split(" ");
                time = Integer.parseInt(ms[1]);
                if (green.equals(ms[0])) {
                    System.out.println(mess);

                    try {
                        Thread.sleep(1000 * time);
                    } catch (InterruptedException ex) {
                        Thread.currentThread().interrupt();
                    }
                    System.out.println("RED");

                    out.writeUTF("done");
                } else if (exit.equals(ms[0])) {
                    System.out.println("EXITING! PROGRAM EXECUTED SUCCESSFULLY!! ;) ");
                    out.writeUTF("exit-ack");
                    break;
                }
            }

            client.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```