A

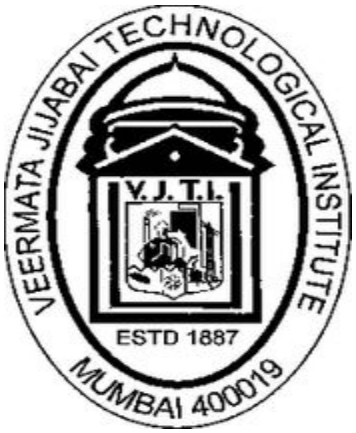
Project Report On

"AUTOMATIC VEHICULAR TRAFFIC CONTROLLER"

Submitted in partial fulfillment of the requirements of the degree of Bachelor of Technology in Computer Engineering by:

Priyanka Rajpal 131071023
Saiksha Shetty 131071026
Anisha Motwani 131071028
Arsha Khan 131071031
Chetana Patel 131071045

Under the guidance of
Prof. Varshapriya Jyotinagar



Department of Computer Engineering and Information Technology
Veermata Jijabai Technological Institute
(An Autonomous Institute affiliated to University of Mumbai)
Mumbai-400019

2016-17

STATEMENT OF CANDIDATES

We state that work embodied in this Project titled **"Automatic Vehicular Traffic Controller"** forms our group's contribution of work under the guidance of **Prof. Varshapriya Jyotinagar** at the Department of Computer Engineering and Information Technology, Veermata Jijabai Technological Institute. No part of this work has been used by us for the requirement of another degree except where explicitly stated in the body of the text and the attached statement.

_____	_____
Priyanka Rajpal	Saiksha Shetty
131071023	131071026
_____	_____
Anisha Motwani	Arsha Khan
131071028	131071031
	_____
	Chetana Patel
	131071045

Date: _____
Place: VJTI, MUMBAI

APPROVAL SHEET

The final year project entitled **"Automatic Vehicular Traffic Controller"** by

Name	**Id. No.**
Priyanka Rajpal	131071023
Saiksha Shetty	131071026
Anisha Motwani	131071028
Arsha Khan	131071031
Chetana Patel	131071045

is found to be satisfactory and is approved for the degree of Bachelor of Technology in Computer Engineering.

_____

_____

External Examiner

_____

Prof. Varshapriya J N

(Project Guide)

Date: _____

Place: VJTI, MUMBAI

CERTIFICATE

This is to certify that the following students have satisfactorily carried out work for the project entitled **"Automatic Vehicular Traffic Controller"** in partial fulfillment of B. Tech. in Computer Engineering at Veermata Jijabai Technological Institute (An Autonomous Institute affiliated to University of Mumbai) during the academic year 2016-2017 under the guidance of Prof. Varshapriya J N.

Name	**Id. No.**
Ms. Priyanka Rajpal	131071023
Ms. Saiksha Shetty	131071026
Ms. Anisha Motwani	131071028
Ms. Arsha Khan	131071031
Ms. Chetana Patel	131071045

_____ _____

Prof. Varshapriya J N
(Project Guide)

External Examiner

CERTIFICATE

This is to certify that this student of B-Tech as a member of team of students has completed the project entitled, **"Adaptive Vehicular Traffic Controller"** to our satisfaction.

Prof. Varshapriya J N	Dr. S. G. Bhirud
(Project Guide)	Head, CE & IT Department

ACKNOWLEDGEMENT

We would like to thank all each person whose support and cooperation has been an invaluable asset during the course of this Project. We would like to thank our guide Prof. Varshapriya J N for guiding us throughout this project and giving it the present shape. It would have been impossible to complete the project without her support, valuable suggestions, criticism, encouragement and guidance.

We convey our gratitude to Dr. S. G. Bhirud, Head of Department for his motivation and providing various facilities, which helped us greatly in the whole process of this project.

We are also grateful to all other teaching and non-teaching staff members of the Computer Technology for directly or indirectly helping us for the completion of this project and the resources provided.

Priyanka Rajpal

Saiksha Shetty

Anisha Motwani

Arsha Khan

Chetana Patel

Chapter One

Introduction

1.1 Background

The number of vehicles on the road increases day by day therefore for the best utilization of existing road capacity, it is important to manage the traffic flow efficiently. Traffic congestion has become a serious issue especially in the modern cities. The main reason is the increase in the population of the large cities that subsequently raise vehicular travel, which creates congestion problem. Due to traffic congestions there is also an increasing cost of transportation because of wastage of time and extra fuel consumption .

Traffic jams also create many other critical issues and problems which directly affect the human routine lives.Traffic system is at the heart of civilized world and development in many aspects of life relies on it. Excessive number of traffic on roads and improper controls of that traffic create traffic jam. It either hampers or stagnates schedule, business, and commerce. Automated traffic detection system is therefore required to run the civilization smooth and safe- which will eventually lead us towards proper analysis of traffic, proper adjustment of control management, and distribution of controlling signals.

1.2 Existing Traffic Controlling Systems

1.2.1 Manual Controlling

Manual controlling the name instance it require man power to control the traffic. Depending on the countries and states the traffic polices are allotted for a required area or city to control traffic. The traffic polices will carry sign board, sign light and whistle to control the traffic. They will be instructed to wear specific uniforms in order to control the traffic.

1.2.2 Automatic Controlling

Automatic traffic light is controlled by timers and electrical sensors.

1.2.2.1 Timers

In traffic light each phase a constant numerical value loaded in the timer. The lights are automatically getting ON and OFF depending on the timer value changes.

1.2.2.2 Electrical Sensors

Electrical sensors are used to capture the availability and generates signals on each phase, depending on the signal the lights automatically switch ON and OFF. The types of detectors used are

- loop detector
- radar detector
- infrared detector
- ultrasonic detector
- microwave detector

1.3 Drawbacks:

In the manual controlling system we need more manpower. As we have poor strength of traffic police we cannot control traffic manually in all area of a city or town. So we need a better solution to control the traffic. On the other side, automatic traffic controlling a traffic light uses timer for every phase. Using electronic sensors is another way in order to detect vehicles, and produce signal that to this method the time is being wasted by a green light on an empty road. Traffic congestion also occurred while using the electronic sensors for controlling the traffic. All these drawbacks are supposed to be eliminated by using image processing.

1.4 Automated Vehicular Traffic Controller:

We propose a system for controlling the traffic light by image processing. The vehicles are detected by the system through images instead of using electronic sensors embedded in the pavement. A camera will be placed alongside the traffic light. It will capture image sequences. Image processing offer a relatively low installation cost with little traffic disruption during maintenance. Image processing provides a better technique to control the state change of the traffic light. It shows that it can decrease the traffic congestion and avoids the time being wasted

by a green light on an empty road. It is also more reliable in estimating vehicle presence because it uses actual traffic images. It visualizes the practicality, so it functions much better than those systems that rely on the detection of the vehicle's metal content.

1.4.1 Problem Statement

Traffic congestion is a big problem for everyone throughout our country. It can lead to drivers becoming frustrated and engaging in road rage.In this project, we propose an Automated Vehicular Traffic Control Framework based on Image Processing, which would detect the current traffic in a particular lane in real time and determine the duration for which the vehicles in that lane must be allowed to move. Traffic system is at the heart of civilized world and development in many aspects of life relies on it.

1.4.2 Motivation

The existing traffic controlling systems are timer based, each lane is allotted the same time to let the traffic irrespective of the occupancy of vehicles on it. This leads to wastage of time when a particular lane has heavy traffic and the other is empty.Some adaptive traffic controllers used include the detectors such as loop, radar, infrared detectors which are expensive with limited capacity and disrupt traffic during installation and maintenance. Image processing based adaptive traffic controllers solve all these problems.

Chapter Two

Literature Review

2.1 Image Processing

Image processing is a method to convert an image into digital form and perform some operations on it, in order to get an enhanced image or to extract some useful information from it. It is a type of signal dispensation in which input is image, like video frame or photograph and output may be image or characteristics associated with that image. Usually Image Processing system includes treating images as two dimensional signals while applying already set signal processing methods to them.

It is among rapidly growing technologies today, with its applications in various aspects of a business. Image Processing forms core research area within engineering and computer science disciplines too.

Image processing basically includes the following three steps.

- Importing the image with optical scanner or by digital photography.
- Analyzing and manipulating the image which includes data compression and image enhancement and spotting patterns that are not to human eyes like satellite photographs.
- Output is the last stage in which result can be altered image or report that is based on image analysis.

2.1.1 Purpose of Image processing

The purpose of image processing is divided into 5 groups. They are:

1. Visualization - Observe the objects that are not visible.
2. Image sharpening and restoration - To create a better image.
3. Image retrieval - Seek for the image of interest.
4. Measurement of pattern – Measures various objects in an image.
5. Image Recognition – Distinguish the objects in an image.

2.1.2 Types

The two types of methods used for Image Processing are

- **Analog Image Processing**

Analog or visual techniques of image processing can be used for the hard copies like printouts and photographs. Image analysts use various fundamentals of interpretation while using these visual techniques. The image processing is not just confined to area that has to be studied but on knowledge of analyst. Association is another important tool in image processing through visual techniques. So analysts apply a combination of personal knowledge and collateral data to image processing.

- **Digital Image Processing**

Digital Processing techniques help in manipulation of the digital images by using computers. As raw data from imaging sensors from satellite platform contains deficiencies. To get over such flaws and to get originality of information, it has to undergo various phases of processing. The three general phases that all types of data have to undergo while using digital technique are Pre-processing, enhancement and display, information extraction.

2.1.3 Image Processing Techniques

Enhancement:

Enhancement programs make information more visible.

- Histogram equalization: Redistributes the intensities of the image of the entire range of possible intensities (usually 256 gray-scale levels).
- Unsharp masking: Subtracts smoothed image from the original image to emphasize intensity changes.

Convolution:

Convolution programs are 3-by-3 masks operating on pixel neighborhoods.

- Highpass filter: Emphasizes regions with rapid intensity changes.
- Lowpass filter-Smoothes images, blurs regions with rapid changes.

Math processes:

Math processes programs perform a variety of functions.

- Add images: Adds two images together, pixel-by-pixel.
- Subtract images: Subtracts second image from first image, pixel by pixel.
- Exponential or logarithm: Raises 'e' to power of pixel intensity or takes log of pixel intensity.
- Nonlinearly accentuates or diminishes intensity variation over the image.
- Scaler add, subtract, multiply, or divide: Applies the same constant values as specified by the user to all pixels, one at a time. Scales pixel intensities uniformly or non-uniformly
- Dilation-Morphological operation: Expanding bright regions of image.
- Erosion-Morphological operation: Shrinking bright regions of image.

Noise filters:

Noise filters decrease noise by diminishing statistical deviations.

Adaptive smoothing filter: Sets pixel intensity to a value somewhere between original value and mean value corrected by degree of noisiness. Good for decreasing statistical, especially single-dependent noise.

- Median filter: Sets pixel intensity equal to median intensity of pixels in neighborhood. An excellent filter for eliminating intensity spikes.
- Sigma filter: Sets pixel intensity equal to mean of intensities in neighborhood within two of the mean. Good filter for signal-independent noise.

Trend removal:

Trend removal programs remove intensity trends varying slowly over the image.

- Row-column fit: Fits image intensity along a row or column by a polynomial and subtract fit from data. Chooses row or column according to direction that has the least abrupt changes.

Edge detection:

Edge detection programs sharpen intensity-transition regions.

- First difference: Subtracts intensities of adjacent pixels. Emphasizes noise as well as desired changes.
- Sobel operator-3-by-3 mask: Weighs inner pixels twice as heavily as corner values. Calculates intensity differences.
- Morphological edge detection: Finds the difference between dilated (expanded) and eroded (shrunken) version of image.

Image analysis:

Image analysis programs extract information from an image.

- Gray-scale mapping: Alters mapping of intensity of pixels in file to intensity displayed on a computer screen.
- Slice-Plots intensity versus position for horizontal, vertical, or arbitrary direction. Lists intensity versus pixel location from any point along the slice.
- Image extraction: Extracts a portion or all of an image and creates a new image with the selected area.
- Images statistics: Calculates the maximum, minimum, average, standard deviation, variance, median, and mean-square intensities of the image data.

2.2 Real time Systems:

Real-Time systems span several domains of computer science. They are defense and space systems, networked multimedia systems, embedded automotive electronics etc. In a real-time system the correctness of the system behavior depends not only the logical results of the computations, but also on the physical instant at which these results are produced. A real-time system changes its state as a function of physical time, e.g., a chemical reaction continues to change its state even after its controlling computer system has stopped. Based on this a real-time system can be decomposed into a set of subsystems i.e., the controlled object, the real-time computer system and the human operator. A real-time computer system must react to stimuli from the controlled object (or the operator) within time intervals dictated by its environment. The instant at which a result is produced is called a deadline. If the result has utility even after the deadline has passed, the deadline is classified as soft, otherwise it is firm. If a catastrophe could result if a firm deadline is missed, the deadline is hard. Commands and Control systems, Air traffic control systems are examples for hard real-time systems. On-line transaction systems, airline reservation systems are soft real-time systems.

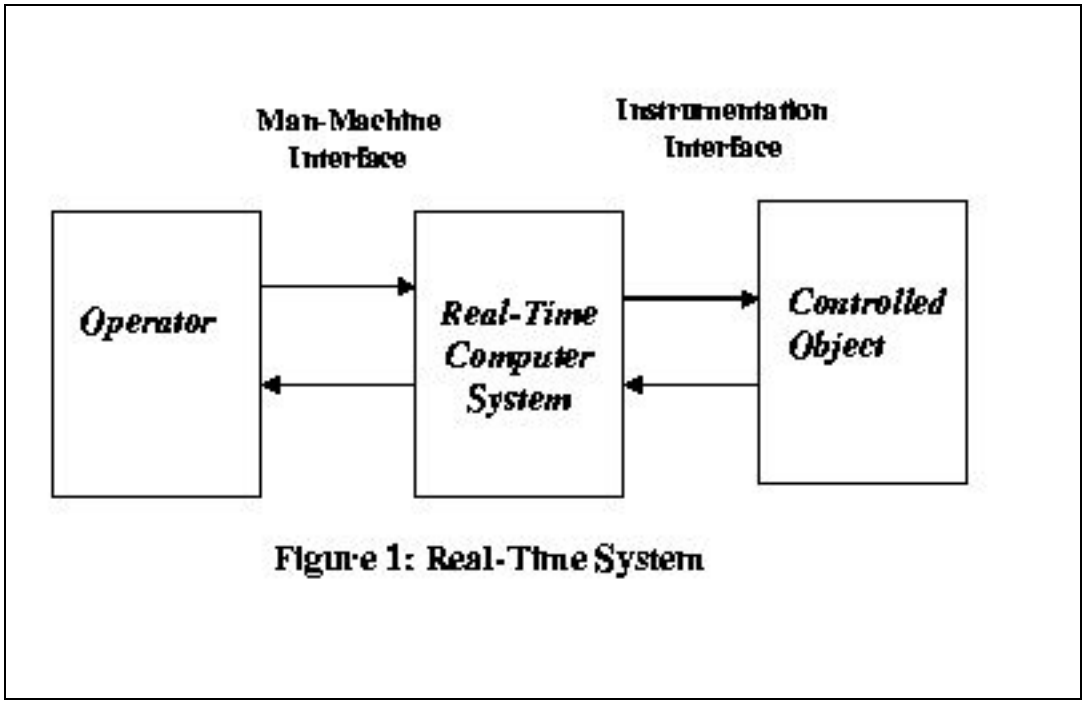


Figure 1: Real-Time System

2.2.1 Real Time System Scheduling:

Real-time task scheduling essentially refers to determining the order in which the various tasks are to be taken up for execution by the operating system. Every operating system relies on one or more task schedulers to prepare the schedule of execution of various tasks it needs to run. Each task scheduler is characterized by the scheduling algorithm it employs. A large number of algorithms for scheduling real-time tasks have so far been developed such as:

1. Clock Driven

- Table-driven
- Cyclic

2. Event Driven

- Simple priority-based
- Rate Monotonic Analysis (RMA)
- Earliest Deadline First (EDF)

3. Hybrid

Round-robin

2.2.2 Round Robin Scheduling :

In round robin algorithm no process is allocated CPU for more than one time slice in a row. If the CPU process exceeds one time slice, the concern process will be preempted and put into the ready queue. The process is preempted after the first time quantum and the CPU is given to the next process which is in the ready queue (process B), similarly schedules all the processes and completes the first cycle. In the second cycle same method is used to schedule the processes.

Traffic Load Categories:

Intersection Point with four sides has been assumed for the study.Three categories of traffic load have been proposed

1.SPARSE

Traffic load is light on all four sides of the intersection point.

2.DENSE

Traffic load is heavy on all four sides of the intersection point.

3. SPARSE - DENSE

Traffic Load is light on one or two sides of the intersection point while traffic load is heavy on other sides. In each category, four processes with same processing time are taken with dynamic quantum time taken. And after that their average waiting time is calculated for each category.

Chapter 3

Proposed Solution

The application of image processing and computer vision techniques to the analysis of video sequences of traffic flow offers considerable improvements over the existing methods of traffic data collection and road traffic monitoring. Other methods including the inductive loop, the sonar and microwave detectors suffer from serious drawbacks in that they are expensive to install and maintain and they are unable to detect slow or stationary vehicles. Video sensors offer a relatively low installation cost with little traffic disruption during maintenance. Furthermore, they provide wide area monitoring allowing analysis of traffic flows and turning movements (important to junction design), speed measurement, multiple point vehicle counts, vehicle classification and highway state assessment (e.g. congestion or incident detection).

We focus on video systems considering both areas of road traffic monitoring and automatic vehicle guidance. There two major subtasks involved in traffic applications, i.e. the automatic lane finding (estimation of lane and/or central line) and vehicle detection (stationary object/obstacle). With the progress of research in computer vision, it appears that these tasks should be trivial. The reality is not so simple; a vision-based system for such traffic applications must have the features of a short processing time, low processing cost and high reliability . Moreover, the techniques employed must be robust enough to tolerate inaccuracies in the 3D reconstruction of the scene, noise caused by vehicle movement and calibration drifts in the acquisition system.

The image acquisition process can be regarded as a perspective transform from the 3D world space to the 2D image space. The inverse transform, which represents a 3D reconstruction of the world from a 2D image, is usually indeterminate (ill-posed problem) because information is lost in the acquisition mapping. Thus, an important task of video systems is to remove the inherent perspective effect from acquired images . This task requires additional spatio-temporal information by the analysis of temporal information from a sequence of images. Optical flow method aid the regularization of the inversion process and help recover scene depth. Static camera observes a dynamic road scene for the purpose of traffic surveillance. In this case, the static camera generally has a good view of the road objects because of the high position of the camera. Therefore, 2D intensity images may contain enough information for the model-based

recognition of road objects. Both lane and object detection become quite different in the cases of stationary (traffic monitoring) and moving camera (automatic vehicle guidance), conceptually and algorithmically. In traffic monitoring, the lane and the objects (vehicles) have to be detected on the image plane, at the camera coordinates.

The proposed system will be able to capture video/images in real time. It will carry out various aspects of image processing on the captured video to detect lane area. The proposed system will then use this data to control the signal by assigning relative durations to each lane. The proposed system will work in real time and deal with situations in parallel which will help in efficient signal control.

Chapter 4

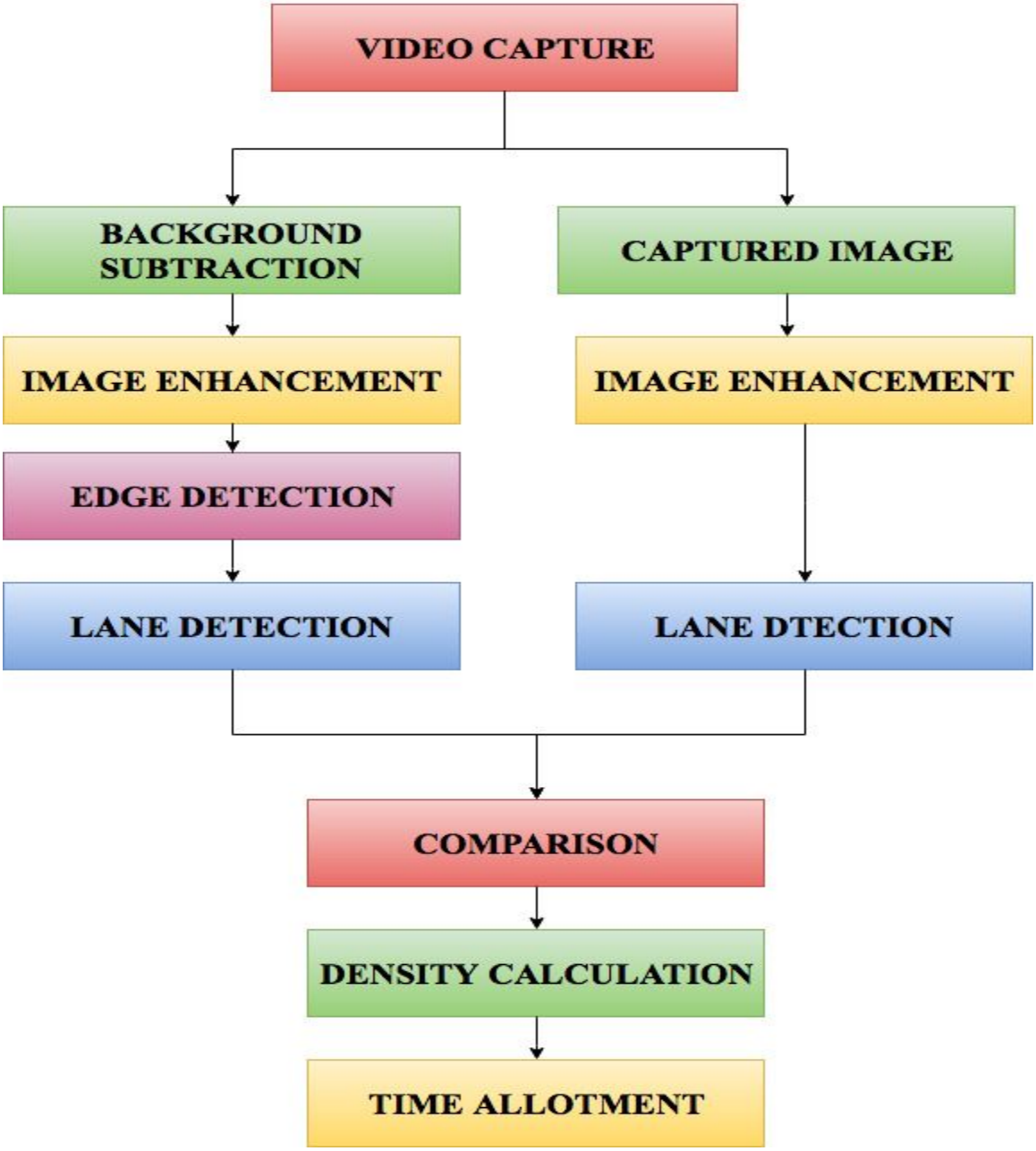
System Analysis

The application of image processing and computer vision techniques to the analysis of video sequences of traffic flow offers considerable improvements over the existing methods of traffic data collection and road traffic monitoring. Other methods including the inductive loop, the sonar and microwave detectors suffer from serious drawbacks in that they are expensive to install and maintain and they are unable to detect slow or stationary vehicles. Video sensors offer a relatively low installation cost with little traffic disruption during maintenance. Furthermore, they provide wide area monitoring allowing analysis of traffic flows and turning movements (important to junction design), speed measurement, multiple point vehicle counts, vehicle classification and highway state assessment (e.g. congestion or incident detection).

We focus on video systems considering both areas of road traffic monitoring and automatic vehicle guidance. There two major subtasks involved in traffic applications, i.e. the automatic lane finding (estimation of lane and/or central line) and vehicle detection (stationary object/obstacle). With the progress of research in computer vision, it appears that these tasks should be trivial. The reality is not so simple; a vision-based system for such traffic applications must have the features of a short processing time, low processing cost and high reliability . Moreover, the techniques employed must be robust enough to tolerate inaccuracies in the 3D reconstruction of the scene, noise caused by vehicle movement and calibration drifts in the acquisition system.

The image acquisition process can be regarded as a perspective transform from the 3D world space to the 2D image space. The inverse transform, which represents a 3D reconstruction of the world from a 2D image, is usually indeterminate (ill-posed problem) because information is lost in the acquisition mapping. Thus, an important task of video systems is to remove the inherent perspective effect from acquired images . This task requires additional spatio-temporal information by the analysis of temporal information from a sequence of images. Optical flow method aid the regularization of the inversion process and help recover scene depth. Static camera observes a dynamic road scene for the purpose of traffic surveillance. In this case, the static camera generally has a good view of the road objects because of the high position of the camera. Therefore, 2D intensity images may contain enough information for the model-based recognition of road objects. Both lane and object detection become quite different in the cases of stationary (traffic monitoring)

and moving camera (automatic vehicle guidance), conceptually and algorithmically. In traffic monitoring, the lane and the objects (vehicles) have to be detected on the image plane, at the camera coordinates.



BLOCK DIAGRAM

When converting an RGB image to grayscale, we have to consider the RGB values for each pixel and make as output a single value reflecting the brightness of that pixel. One of the approaches is to take the average of the contribution from each channel: (R+B+C)/3. However, since the perceived brightness is often dominated by the green component, a different, more "human-oriented", method is to consider a weighted average, e.g.: 0.3R + 0.59G + 0.11B.

4.1 Image Enhancement

The principal objective of image enhancement is to process a given image so that the result is more suitable than the original image for a specific application. It accentuates or sharpens image features such as edges, boundaries, or contrast to make a graphic display more helpful for display and analysis. The enhancement doesn't increase the inherent information content of the data, but it increases the dynamic range of the chosen features so that they can be detected easily. The greatest difficulty in image enhancement is quantifying the criterion for enhancement and, therefore, a large number of image enhancement techniques are empirical and require interactive procedures to obtain satisfactory results. Image enhancement methods can be based on either spatial or frequency domain techniques. According to the operations on the image pixels, spatial domain techniques can be further divided into 2 categories: Point operations and spatial operations (including linear and non-linear operations). Low-contrast images can result from poor illumination, lack of dynamic range in the image sensor, or even wrong setting of a lens aperture during image acquisition. The idea behind contrast stretching is to increase the dynamic range of the gray levels in the image being processed.

4.2 Background Subtraction

Background subtraction is a major preprocessing step in many vision based applications. For example, consider the cases like visitor counter where a static camera takes the number of visitors entering or leaving the room, or a traffic camera extracting information about the vehicles etc. In all these cases, first you need to extract the person or vehicles alone. Technically, you need to extract the moving foreground from static background. If you have an image of background alone, like image of the room without visitors, image of the road without vehicles etc, it is an easy job. Just subtract the new image from the background. You get the foreground objects alone. But in most of

the cases, you may not have such an image, so we need to extract the background from whatever images we have. It become more complicated when there is shadow of the vehicles. Since shadow is also moving, simple subtraction will mark that also as foreground. It complicates things. Several algorithms were introduced for this purpose. OpenCV has implemented three such algorithms which is very easy to use. We will see them one-by-one.

4.2.2 BackgroundSubtractorMOG

It is a Gaussian Mixture-based Background/Foreground Segmentation Algorithm. It uses a method to model each background pixel by a mixture of K Gaussian distributions (K = 3 to 5). The weights of the mixture represent the time proportions that those colours stay in the scene. The probable background colours are the ones which stay longer and more static. While coding, we need to create a background object using the function, cv2.createBackgroundSubtractorMOG(). It has some optional parameters like length of history, number of gaussian mixtures, threshold etc. It is all set to some default values. Then inside the video loop, use backgroundsubtractor.apply() method to get the foreground mask.

4.2.3 BackgroundSubtractorMOG2

It is also a Gaussian Mixture-based Background/Foreground Segmentation Algorithm. One important feature of this algorithm is that it selects the appropriate number of gaussian distribution for each pixel. (Remember, in last case, we took a K gaussian distributions throughout the algorithm). It provides better adaptibility to varying scenes due illumination changes etc. As in previous case, we have to create a background subtractor object. Here, you have an option of selecting whether shadow to be detected or not. If detectShadows = True (which is so by default), it detects and marks shadows, but decreases the speed. Shadows will be marked in gray color. OpenCV has few implementations of Background Segmentation. We will be looking at one of those. The class is called BackgroundSubtractorMOG2. It is a Gaussian Mixture-based Background Segmentation Algorithm. This algorithm takes the background pixels and assigns a Gaussian Distribution to each one. The weight of this distribution is the amount of time the colors stay in the scene. Basically, the algorithm tries to identify the background by the information from the Gaussian

mixture. The idea is that the longer the color stays the higher the probability that it is a part of the background. The gaussian distribution helps this method to adapt to variance in illumination. This class supports parallel computing.

4.2.4 BackgroundSubtractorGMG

This algorithm combines statistical background image estimation and per-pixel Bayesian segmentation.It uses first few (120 by default) frames for background modelling. It employs probabilistic foreground segmentation algorithm that identifies possible foreground objects using Bayesian inference. The estimates are adaptive; newer observations are more heavily weighted than old observations to accommodate variable illumination. Several morphological filtering operations like closing and opening are done to remove unwanted noise. You will get a black window during first few frames. It would be better to apply morphological opening to the result to remove the noises.

4.3 Edge Detection

Edge detection is one of the most commonly used operations in image analysis. An edge is defined by a discontinuity in gray level values. In other words, an edge is the boundary between an object and the background. The shape of edges in images depends on many parameters: The geometrical and optical properties of the object, the illumination conditions, and the noise level in the images. The importance of the classification is that it simplifies several problems in Artificial Vision and Image Processing, by associating specific processing rules to each type of edges.

Edge detectors may well be classified into 5 categories as follows:

1. **Gradient edge detectors:** Which contains classical operators and uses first directional derivative operation. It includes algorithms such as: Sobel, Prewitt, Kirsch, Robinson, Frei-Chen, Deatsch and Fram, Nevatia and Babu, Ikonomopoulos, Davies, Kitchen and Malin, Hancock and Kittler, Woodhall and Lindquist and Young-won and Udpa.
2. **Zero Crossing:** Which uses second derivative and includes Laplacian operator and second directional derivative.
3. **Laplacian of Gaussian (LoG):** Which was invented by Marr and Hildreth (1980) who

combined Gaussian filtering with the Laplacian.This algorithm is not used frequently in machine vision.

4. **Gaussian Edge Detectors:** Which is symmetric along the edge and reduces the noise by smoothing the image. The significant operators here are Canny and ISEF (Shen-Castan) which convolve the image with the derivative of Gaussian for Canny and ISEF for Shen-Castan.
5. **Colored Edge Detectors:** Which are divided into three categories output Fusion methods, Multi-dimensional gradient methods and Vector methods

As edge detection is a fundamental step in computer vision, it is necessary to point out the true edges to get the best results from the matching process. That is why it is important to choose edge detectors that fit best to the application. In this respect, we first present some advantages and disadvantages of algorithms within the context of our classification in Table 1.

Operator	Advantages	Disadvantages
Classical(Sobel,Prewitt, Kirsch)	Simplicity, Detection of edges and their Orientation.	Sensitivity to noise, Inaccurate.
Zero Crossing(Laplacian,Second Directional Derivative)	Detection of edges and their Orientations, Having fixed characteristics in all direction.	Reresponsing to some of the existing edges, Sensitivity to noise.
Laplacian of Gaussian (LoG) (Marr-Hildreth)	Finding the correct places of edges, Testing wider area around the pixel.	Malfunctioning at corners, curves and where the gray level intensity varies, Not finding the orientation of edge because of using the Laplacian filter.
Gaussain (Canny, ShenCastan)	Using Probability for finding error rate, Localization, and response , Improving signal to nise ratio, Better detection specially in noise conditions	Complex Computations, False zero crossing, Time consuming.
Colored Edge Detectors	Accurate, More efficient in object recognition	Complicated, Complex Computations.

Some fundamental issues of object detection are considered in this section. Different approaches have been categorized according to the method used to isolate the object from the background on a single frame or a sequence of frames. Some of these approaches are:

- Boolean edge detector
- Marr-Hildreth Edge Detector
- Sobel operator
- Canny Edge Detector
- Prewitt Edge Detector

4.3.1 Boolean Edge Detector

This is a simple technique. Boolean edge detector converts a window of pixels into a binary pattern based on a local threshold, and then applies masks to determine if an edge exists at a certain point or not.

4.3.2 Marr-Hildreth Edge Detector

Marr–Hildreth algorithm is a method of detecting edges in digital images, that is, continuous curves where there are strong and rapid variations in image brightness. The Marr–Hildreth edge detection method is simple and operates by convolving the image with the Laplacian of the Gaussian function, or, as a fast approximation by Difference of Gaussians. Then, zero crossings are detected in the filtered result to obtain the edges. The Laplacian-of-Gaussian image operator is sometimes also referred to as the Mexican hat wavelet due to its visual shape when turned upside-down. The Marr–Hildreth operator, however, suffers from two main limitations. It generates responses that do not correspond to edges, so-called "false edges", and the localization error may be severe at curved edges.

4.3.3 Sobel Operator

It computes a 2-D spatial gradient measurement on an image and emphasizes regions of high spatial frequency that correspond to edges. Usually it is used to find the approximate absolute gradient magnitude of an input grayscale image at each point. The discrete computation of the partial derivation is approximated in digital images by using the Sobel operator, which is shown in the masks below:

-1	0	+1
-2	0	+2
-1	0	+1

G_x

+1	+2	+1
0	0	0
-1	-2	-1

G_y

Figure: Sobel Convolution Masks

4.3.4 Canny Edge Detector

Canny Edge Detector technique is very important for detecting edges in an image. This operator isolates noise from an image before finding, edges of an image without affecting the features of the image, and then applying the tendency to find the edges and the critical value for threshold.

The Canny edge detector is considered to be one of the most widely used edge detection algorithms in the industry. It works by first smoothing the image and finds the image gradient to highlight regions with high spatial derivatives. It then tracks along these regions to suppress any pixel that is not at the maximum. Finally, through hysteresis, it uses two thresholds and if the magnitude is below the first threshold, it is set to zero. If the magnitude is above the high threshold, it is made an edge and if the magnitude is between the two thresholds, it is set to zero unless there is a path from this pixel to a pixel with a gradient above the second threshold. That is to say that the two thresholds are used to detect strong and weak edges, and include the weak edges in the output only if they are connected to strong edges. It is a multi-step process, which can be implemented on the GPU as a sequence of filters.

Canny edge detection technique is based on three basic objectives.

1. **Low error rate:-** All edges should be found, and there should be no spurious responses. That is, the edges must be as close as possible to the true edges.
2. **Edge point should be well localized:-** The edges located must be as close as possible to the true edges. That is , the distance between a point marked as an edge by the detector and the centre of the true edge should be minimum.
3. **Single edge point response:-** The detector should return only one point for each true edge point. That is, the number of local maxima around the true edge should be minimum. This means that the detector should not identify multiple edge pixels where only a single edge point exist.

The essence of Canny's work was in expressing the preceding three criteria mathematically and then attempting to find optimal solution to these formulations, in general, it is difficult to find a close form solution that satisfies all the preceding objectives. However, using numerical optimization with 1-D step edges corrupted by additive while Gaussian noise led to the conclusion that a good approximation to the optimal step edge detector is the first derivative of Gaussian:

$$\frac{d}{dx}e^{\frac{-x^2}{2\sigma^2}} = \frac{-x}{\sigma^2}e^{\frac{-x^2}{2\sigma^2}} \qquad (1)$$

,

Generalizing this result to 2-D involves recognizing that the 1-D approach still applies in the direction of the edge normal. Because the direction of the normal is unknown beforehand, this would require applying the 1-D edge detector in all possible directions. This task can be approximated by first smoothing the image with circular 2-D Gaussian function, computing the gradient of the result, and then using the gradient magnitude and direction to estimate edge strength and direction at every point. Let f(x,y) denote the input image and G(x,y) denote the Gaussian function:

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (2)$$

We form a smoothed image, fs(x, y), by convolving G and f:

$$f_s(x,y) = G(x,y) * f(x,y) \quad (3)$$

This operation is followed by computing the gradient and direction (angle)

$$M(x,y) = \sqrt{g_x^2 + g_y^2} \quad (4)$$

And

$$\alpha(x,y) = \tan^{-1}\frac{g_y}{g_x} \quad (5)$$

with gx = $\partial \mathbf{f}_s / \partial \mathbf{x}$ and gy = $\partial f_s / \partial y$.

Equation (2) is implemented using an n x n Gaussian mask. Keep in mind that M(x, y) and α(x, y) are arrays of the same size as the image from which they are computed. Because it is generated using the gradient M(x, y) typically contains wide ridges around local maxima.

The next step is to thin those ridges. One approach is to use non maxima suppression. This can be done in several ways, but the essence of the approach is to specify a no. of discrete orientations of edge normal (gradient vector). For example, in 3x3 region we can define four orientation for an edge passing through the centre point of the region: horizontal, vertical, +45° and -45°.

The final operation is to threshold $g_N(x, y)$ to reduce false edge point. We do it by using a single threshold, in which all value below the threshold were set to 0. If we set the threshold too low, there will still be some false edge (called false positives). If the threshold id set too high, then actual valid edge points will be eliminated (false negatives). Canny's algorithm attempts to improve on this situation by using hysteresis threshold. We use two threshold, A low threshold, T_L, and a high threshold, T_H. Canny suggested that the ratio of high to low threshold should be two or three to one.

We visualize the thresholding operation as creating two additional images.

$$\mathbf{g_{NH}(x,y) = g_N(x,y) \geq T_H} \qquad \mathbf{(6)}$$

And

$$\mathbf{g_{NL}(x,y) = g_N(x,y) \geq T_L} \qquad \mathbf{(7)}$$

Where, initially, both $g_{NH}(x, y)$ *and* $g_{NL}(x, y)$ are set to 0. After thresholding, $g_{NH}(x, y)$ will have fewer nonzero pixels than $g_{NL}(x, y)$ in general, but all the nonzero pixels in $g_{NH}(x, y)$ will be contained in $g_{NL}(x, y)$ because the latter image is formed with lower threshold. We eliminate from $g_{NL}(x, y)$ all the nonzero from $g_{NH}(x, y)$ by letting

$$g_{NL}(x, y) = g_{NL}(x, y) - g_{NH}(x, y) \qquad (8)$$

The nonzero pixels in $g_{NL}(x, y)$ and $g_{NH}(x, y)$ may be viewed as being "strong". And "weak" edge pixels, respectively. After the thresholding operations, all strong pixels in $g_{NH}(x, y)$ are assumed to be valid edge pixels and are so marked immediately. Depending on the value of T_H, the edges in $g_{NH}(x, y)$ typically have gaps. Longer edges are formed using the following procedure

a. Locate the next unvisited pixel p in $g_{NH}(x, y)$.

b. Mark as valid edge pixels all the weak pixels in $g_{NL}(x, y)$ that are connected to p using say 8 connectivity.

c. If all nonzero pixels in $g_{NH}(x, y)$ have been visited go to step d. else return to step a.

d. Set to zero all pixels in $g_{NL}(x, y)$ that were not marked as valid edge pixels.

At the end of this procedure, the final image output by the Canny is formed by appending to $g_{NH}(x, y)$ all the nonzero pixels from $g_{Nl}(x, y)$.

We use two additional images, $g_{NH}(x, y)$ and $g_{Nl}(x, y)$, to simplify the discussion. In practice, hysteresis threshold can be implemented directly during non-maxima suppression, and thresholding can be implemented directly on $g_{NL}(x, y)$ by forming a list of strong pixels and the weak pixels connected to them.

Summarizing, the Canny edge detection algorithm consist of the following basic steps:

1. Smooth the input image with Gaussian filter.
2. Compute the gradient magnitude and angle images.
3. Apply non-maxima suppression to the gradient magnitude image.
4. Use double thresholding and connectivity analysis to detect and link edges.

In our project we use "CANNY EDGE DETECTION TECHNIQUE" because of its various advantages over other edge detection techniques.

4.3.5 Prewitt Edge Detector

This operator is an appropriate way to estimate the magnitude and orientation of an edge in an image. Although differential gradient edge detection needs a rather time consuming calculation to estimate the orientation from the magnitudes in the x and y-directions, the compass edge detection obtains the orientation directly from the kernel with the maximum response. The Prewitt operator is limited to 8 possible orientations. This gradient based edge detector is estimated in the 3x3 neighborhoods for eight directions. One convolution mask which yielded maximum response is selected at each point.

-1	0	1
-1	0	1
-1	0	1

G_x

1	1	1
0	0	0
-1	-1	-1

G_y

Figure: Prewitt Convolution Masks

4.4 Lane Boundary Detection:

Another method in lane finding is Lane Boundary Detection. Here the lane is determined by finding its edges, boundaries. It is known as a feature driven approach.

This class of approach is based on the detection of edges in the image and the organization of edges into meaningful structures (lanes or lane markings). This class involves two levels of processing:

- Feature detection
- Feature aggregation.

4.4.1 Feature Detection:

This part aims at extracting intensity discontinuities. To make the detection more effective, image enhancement is performed followed by a gradient operator. The dominant edges are extracted based on thresholding of the gradient magnitude. At this stage, the direction of edges at each pixel can be computed based on the phase of the gradient and a curvature of line segments can be estimated based on neighborhood relations.

4.4.2 Feature Aggregation:

Feature aggregation organizes edge segments into meaningful structures (lane markings) based on short-range or long-range attributes of the lane. Short-range aggregation considers local lane fitting into the edge structure of the image. A realistic assumption that is often used requires that the lane (or the lane marking) width does not change drastically. Hence, meaningful edges of the video image are located at a certain distance apart, in order to fit the lane-width model. Long-range aggregation is based on a line intersection model, based on the assumption of smooth road curvature.

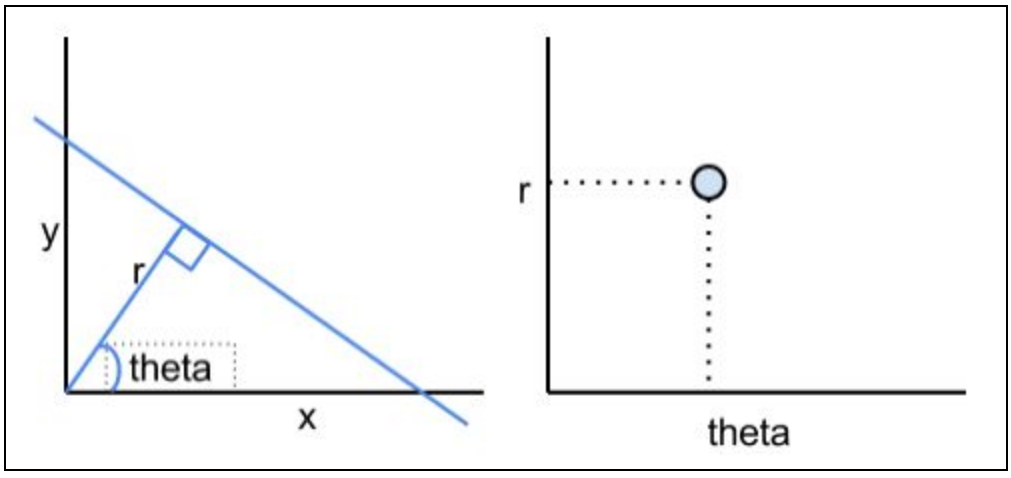
In this project, we want to detect the lines indicating the boundary of the lane, so we use the Hough Transform (HT), which is a robust method for finding lines in images

Hough Transform:

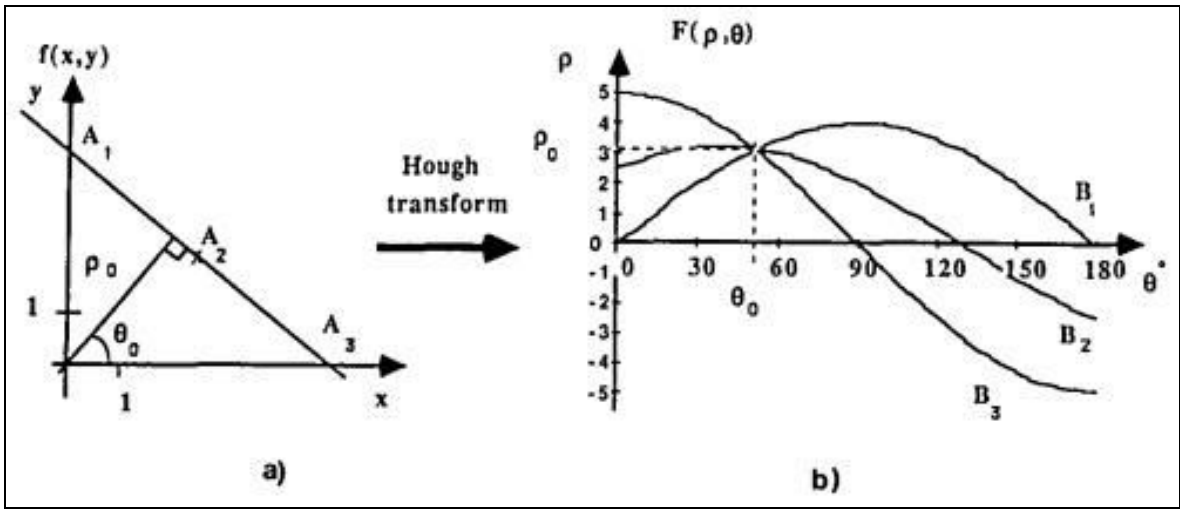
The Hough transform is a technique which can be used to isolate features of a particular shape within an image. Because it requires that the desired features be specified in some parametric form, the classical Hough transform is most commonly used for the detection of regular curves such as lines, circles, ellipses, etc. A generalized Hough transform can be employed in applications where a simple analytic description of a feature(s) is not possible. Due to the computational complexity of the generalized Hough algorithm, we restrict the main focus of this discussion to the classical Hough transform. Despite its domain restrictions, the classical Hough transform (hereafter referred to without the classical prefix) retains many applications; as most manufactured parts (and many anatomical parts investigated in medical imagery) contain feature boundaries which can be described by regular curves. The main advantage of the Hough transform technique is that it is tolerant of gaps in feature boundary descriptions and is relatively unaffected by image noise.

The main idea for the HT is as follows:

- For each line L, there is a unique line L perpendicular to L which passes through the origin.
- L has a unique distance (r) and angle (θ) from the horizontal axis of the image. This angle and distance define a point in the parameter space, sometimes known as Hough space.



- A point in image space has an infinite number of lines that could pass through it, each with a unique distance and angle.
- This set of lines corresponds to a sinusoidal function in parameter space. Two points on a line in image space correspond to two sinusoids which cross at a point in parameter space.



- That point in parameter space corresponds to that line in image space, and all sinusoids corresponding to points on that line will pass through that point.

The real solution to implement this algorithm is to quantize the parameter space by using a 2D array of counters, where the array coordinates represent the parameters of the line; this is commonly known as an accumulator array.

The HT method for finding lines in images generally consists of the following three stages:

- Perform accumulation on the accumulator array using the binary edge image.
- Find peak values in the accumulator array
- Verify that the peaks found correspond to legitimate lines, rather than noise.

Local Maxima Finder

The Local Maxima Finder object is a function that uses to find the peaks in the Hough transform. After constructing the object and optionally setting properties, researchers use the step method to find the coordinates of the local maxima in the input image frame.

4.5 Comparison

The comparison is done between the reference image and current image. Current image is formed by taking frames from the running traffic video. Both the reference image and current image undergo edge detection and lane detection. For comparing the two images, we consider the current area of interest of the two images. Each pixel value in this area is compared by calculating the absolute difference between the two values.

4.6 Density Calculation

After comparing the two images in previous step, we will assign values to the pixels. The pixel value of zero is assigned to those pixels that completely match with each other, while those that differ even slightly are changed to white color. For calculating density we will make use of these white pixels.

Density is calculated using the formula:

$$\text{DENSITY} = \frac{\text{NO. OF WHITE PIXELS}}{\text{TOTAL NO. OF PIXELS}}$$

4.7 Time Allotment

Based on our results, the time for which the traffic light must be green is determined. The time allotment is based on the Round Robin Scheduling, where the time allotted for a particular lane would be directly proportional to the traffic on it. A fixed time quantum ‘t’ would be set, for which the green light for each lane must be turned on. If time taken to compute slot duration crosses the time by which green signal has to be activated, then the fixed quantum t would be assigned to that signal.

The algorithm for time allotment is as follows:

- Assign a fixed time quantum ’t’ by taking input from the user.
- Each lane will be given a time slot that will be a multiple of t i.e. n*t.
- For each lane, calculate the density value.

The value of n is decided using the table:□□

DENSITY RANGE	ALLOTMENT FACTOR(n)
0-15	0.5
16-30	0.8
31-40	1
41-50	1.2
51-75	1.5
76-100	1.75

Chapter 5

System Design

5.1 Hardware Requirements

- Processor: The memory specifications used in this project consists of Intel 2 GHz DualCore processor, 2 GB RAM and a 250 GB hard drive.
- Camera : Camera Should be placed at a height between 30ft to 40 ft with resolution 4K and it should not be moveable. Camera Settings: 8 mega(16:9),30fps,3840 x 2160, 3x Zoom, VBR, Fine Quality.
- Traffic Signal Controllers

5.2 Software Tools

- OpenCV
- Visual Studio
- QT C++

5.2.1 OpenCV

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision.The library is cross-platform and free for use under the open-source BSD license. Downloaded Latest Version 3.1

OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. There are bindings in Python, Java and MATLAB/OCTAVE. The API for these interfaces can be found in the online documentation.Wrappers in other languages such as C#, Perl, Ch, Haskell and Ruby have been developed to encourage adoption by a wider audience.

All of the new developments and algorithms in OpenCV are now developed in the C++ interface.

OpenCV runs on a variety of platforms. Desktop: Windows, Linux, OS X, FreeBSD, NetBSD, OpenBSD; Mobile: Android, iOS, Maemo, BlackBerry 10. The user can get official releases from SourceForge or take the latest sources from GitHub.

5.2.2 Visual Studio

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs for Microsoft Windows, as well as web sites, web applications and web services. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native code and managed code.

Visual Studio supports different programming languages and allows the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists. Built-in languages include C, C++ and C++/CLI (via Visual C++), VB.NET (via Visual Basic .NET), C# (via Visual C#). Support for other languages such as Python, Ruby, Node.js, and M among others is available via language services installed separately. It also supports XML/XSLT, HTML/XHTML, JavaScript and CSS. Java (and J#) were supported in the past.

5.2.3 Qt C++

Qt is a cross-platform framework, that is usually used as a graphical toolkit, although it is also very helpful in creating CLI applications. It runs on the three major desktop OSes, as well as on mobile OSes, such as Symbian, Nokia Belle, and on embedded devices. Ports for Android (Necessitas) and iOS are also in development.

Purposes and abilities

Qt is used for developing multi-platform applications and graphical user interfaces (GUIs); however, programs without a GUI can be developed, such as command-line tools and consoles for servers. An example of a non-GUI program using Qt is the Cutelyst web framework. GUI

programs created with Qt can have a native-looking interface, in which case Qt is classified as a widget toolkit.

Qt uses standard C++ with extensions including signals and slots that simplify handling of events, and this helps in development of both GUI and server applications which receive their own set of event information and should process them accordingly. Qt supports many compilers, including the GCC C++ compiler and the Visual Studio suite. Qt also provides Qt Quick, that includes a declarative scripting language called QML that allows using JavaScript to provide the logic. With Qt Quick, rapid application development for mobile devices became possible, although logic can be written with native code as well to achieve the best possible performance. Qt can be used in several other programming languages via language bindings. It runs on the major desktop platforms and some of the mobile platforms. It has extensive internationalization support. Non-GUI features include SQL database access, XML parsing, JSON parsing, thread management and network support.

5.2. Implementation

5.2.1 Code:

AVTC2.pro

```
QT       += core gui
greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
TARGET = AVTC2
TEMPLATE = app
DEFINES += QT_DEPRECATED_WARNINGS
SOURCES += main.cpp\
      mainwindow.cpp \
  output.cpp \
  myprocess.cpp
HEADERS  += mainwindow.h \
  output.h \
```

```
  myprocess.h
FORMS    += mainwindow.ui \
  output.ui
INCLUDEPATH += C:\opencv-build\install\include
LIBS += -LC:\opencv-build\install\x86\mingw\lib \
  -lopencv_core320.dll \
  -lopencv_highgui320.dll \
  -lopencv_imgcodecs320.dll \
  -lopencv_imgproc320.dll \
  -lopencv_features2d320.dll \
  -lopencv_calib3d320.dll \
  -lopencv_video320.dll \
  -lopencv_videoio320.dll \
  -lopencv_videostab320.dll \
  -lopencv_stitching320.dll
```

Hearders Files:

1. **mainwindow.h**

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QMainWindow>
#include <output.h>
#include <myprocess.h>
#include <qnamespace.h>
#include <QLineEdit>

namespace Ui {
class MainWindow;
}
```

```
class MainWindow : public QMainWindow
{
  Q_OBJECT
public:
  QLineEdit *n,*gt,*rt;
  MyProcess *p;
  explicit MainWindow(QWidget *parent = 0);
  ~MainWindow();

signals:
  void send(int lanes);
private slots:
  void on_browse_clicked();
  void on_submitButton_clicked();
  void on_lanesEdit_returnPressed();
private:
  Ui::MainWindow *ui;
};
#endif // MAINWINDOW_H
```

2. myprocess.h

```
#ifndef PROCESS_H
#define PROCESS_H
#include <sys/types.h>
#include <qnamespace.h>
#include "mainwindow.h"
#include <qobject.h>
#include "opencv2/imgcodecs.hpp"
#include <opencv2/highgui.hpp>
```

```
#include "opencv2/imgproc.hpp"
#include <opencv2/video.hpp>
#include "opencv2/videoio.hpp"
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/video/background_segm.hpp>
#include <iostream>
#include <math.h>
#include <string.h>
#include <thread>
using namespace std;

class MyProcess: public QObject
{
  Q_OBJECT
public:
  // Global variables
  cv::Mat frame, segm; //current frame
  cv::Mat fgMaskMOG2; //fg mask fg mask generated by MOG2 method
  cv::Ptr<cv::BackgroundSubtractor> pMOG2; //MOG2 Background subtractor
  int keyboard; //input from keyboard
  cv::Mat refs[10];
  int n,gt,rt;
  string paths[10];
  thread t[10];

  MyProcess(QObject* parent=0);
  void setPath(int i,string value);
  void start();
```

```
  void processVideo(string videoFilename, int pos, int gt, int rt);
  void display(int pos,int gt, time_t n,time_t a);
  cv::Mat imageEnhance(cv::Mat src, int r1, int s1, int r2, int s2);
  int computeOutput(int x, int r1, int s1, int r2, int s2);
  void hough(cv::Mat src, int p[], int pos);
   void diffLane(cv::Mat backgroundImage, cv::Mat currentImage, int p[], int pos,int gt,time_t act);
  int timecalculate(float p,int gt );
  cv::Mat maskLane(cv::Mat fg, int new_h, int new_w, int a[], int p[]);

signals:
  void displaySignal(int pos,int gt, time_t n,time_t a);
};
#endif // PROCESS_H
```

3. output.h

```
#ifndef OUTPUT_H
#define OUTPUT_H
#include <QWidget>
#include <QLabel>
#include <qnamespace.h>
namespace Ui {
class Output;
}
class Output : public QWidget
{
  Q_OBJECT
public:
  explicit Output(QWidget *parent = 0);
```

```
  void display();
  QLabel *label;
  ~Output();

public slots:
  void receiveMsg(int);
  void show(int pos,int gt, time_t n,time_t a);
private:
  Ui::Output *ui;
};
#endif // OUTPUT_H
```

Source Files:

1. **main.cpp**

```
#include "mainwindow.h"
#include <QApplication>
#include "opencv2/imgcodecs.hpp"
#include <opencv2/highgui.hpp>
#include "opencv2/imgproc.hpp"
#include <opencv2/video.hpp>
#include "opencv2/videoio.hpp"
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/video/background_segm.hpp>
#include <iostream>
#include <math.h>
using namespace cv;
using namespace std;
```

```
int main(int argc, char *argv[])
{
  QApplication a(argc, argv);
  MainWindow w;
  w.show();
  return a.exec();
}
```

2. mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include<QPixmap>
#include<QtWidgets>
#include "output.h"
#include "myprocess.h"
#include "ui_output.h"
#include <iostream>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
using namespace std;

QLineEdit *vEdit;

MainWindow::MainWindow(QWidget *parent) :
  QMainWindow(parent),
  ui(new Ui::MainWindow)
{
  ui->setupUi(this);
  p=new MyProcess();
```

```
  //set a background image for main window
  QPixmap bkgnd("C:/Users/admin/Desktop/img.png");
  bkgnd = bkgnd.scaled(this->size(), Qt::IgnoreAspectRatio);
  QPalette palette;
  palette.setBrush(QPalette::Background, bkgnd);
  this->setPalette(palette);
  this->setWindowTitle("AVTC Input");

  ui->lanesEdit->setValidator(new QIntValidator(2,10,this));
  ui->verticalLayout_2->setAlignment(ui->submitButton,Qt::AlignHCenter);
  layout()->setAlignment(ui->verticalLayout_3,Qt::AlignHCenter);
  ui->heading->setAlignment(Qt::AlignHCenter);
  ui->demo->setVisible(0);
  n=ui->lanesEdit;
  gt=ui->greenTime;
  rt=ui->redTime;
}

MainWindow::~MainWindow()
{
  delete ui;
}

void MainWindow::on_lanesEdit_returnPressed()
{
  ui->demo->setText("NOTE: Select the videos for the lanes in order the green signal goes");
  ui->demo->setVisible(1);
  QLayout *vLayout=ui->videoLayout;
```

```
  QLayoutItem *child;
  while((child=vLayout->takeAt(0))!=0){
     vLayout->removeItem(child);
     vLayout->removeWidget(child->widget());
     delete child;
  }
  int i,lanes=ui->lanesEdit->text().toInt();

  for(i=1;i<=lanes;i++){
     QHBoxLayout *vLayout=new QHBoxLayout;
     vEdit=new QLineEdit;
     vEdit->setObjectName("vEdit"+QString::number(i));

     QPushButton *btn=new QPushButton("Browse");
     btn->setObjectName("btn"+QString::number(i));

     QObject::connect(btn,SIGNAL(clicked()),SLOT(on_browse_clicked()));

     vEdit->setFixedWidth(250);
     vLayout->addWidget(vEdit);
     vLayout->addWidget(btn);
     ui->videoLayout->addLayout(vLayout);
  }
}

void MainWindow::on_browse_clicked()
{
  //open dialog box
  QPushButton *buttonSender = qobject_cast<QPushButton*>(sender());
```

```
  QString btnName=buttonSender->objectName();

  QString vName="vEdit"+btnName[3];
  QString pn=btnName[3]+"";
  int pathNumber=pn.toInt();
  QLineEdit *vE=this->findChild<QLineEdit *>(vName);
      QString s = QFileDialog::getOpenFileName(this,tr("Choose Video"),
"C:/Users/admin/Desktop/videos", tr("Video Files (*.mp4)"));
  vE->setText(s);
  p->setPath((pathNumber-1),s.toStdString());
}

void MainWindow::on_submitButton_clicked()
{
  if(ui->lanesEdit->text()!=NULL){
     //call start function pf process class
     p->n=ui->lanesEdit->text().toInt();
     p->gt=ui->greenTime->text().toInt();
     p->rt=ui->redTime->text().toInt();
     p->start();
     Output *ansWindow=new Output();
     connect(this,send,ansWindow,&Output::receiveMsg);
     connect(p,&MyProcess::displaySignal,ansWindow,&Output::show);
     int lanes=ui->lanesEdit->text().toInt();
     emit send(lanes);
  }
  else{
     ui->demo->setText("Enter all the above parameters");
```

```
    ui->demo->setVisible(1);
  }
}
```

3. myprocess.cpp

```
#include "myprocess.h"
#include "opencv2/imgcodecs.hpp"
#include <opencv2/highgui.hpp>
#include "opencv2/imgproc.hpp"
#include <opencv2/video.hpp>
#include "opencv2/videoio.hpp"
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/video/background_segm.hpp>
#include <iostream>
#include <math.h>
#include <thread>
#include <string.h>
#include <string>
#include <qnamespace.h>
using namespace cv;
using namespace std;

MyProcess::MyProcess(QObject* parent):QObject(parent)
{
  n=0,rt=0,gt=0;
  pMOG2 = cv::createBackgroundSubtractorMOG2(); //MOG2 approach
}
```

```
void MyProcess::setPath(int i,string value){
  paths[i]=value;
}

void MyProcess::start(){
  for (int i = 0; i < n; i++)
     t[i] =thread(processVideo, paths[i], (i + 1), gt, rt);
  for (int i = 0; i < n; i++)
     t[i].join();
}

void MyProcess::processVideo(string videoFilename, int pos, int gt, int rt) {
  //create the capture object
  VideoCapture capture(videoFilename);
  if (!capture.isOpened()) {
     //error in opening the video input
     cerr << "Unable to open video file: " << videoFilename << endl;
     exit(3);
  }
  stringstream fps;
  fps << capture.get(CV_CAP_PROP_FPS); // no. of frames per second
  string q = fps.str();
  int nf = stoi(q);
  stringstream fcs;
  fcs << capture.get(CV_CAP_PROP_FRAME_COUNT); // no. of frames per second
  string          q1 = fcs.str();
  int fc = stoi(q1)/nf;
  //read input data. ESC or 'q' for quitting
  Mat f;
```

```
int cf = 0;
string sname;
int p[10];
time_t now = time(0);
int d = 0;
time_t maxTime = now + fc;
this_thread::sleep_for(chrono::seconds(gt*(pos - 1))); //as expected 1st will be green and others red sleep till green
now = time(0);
time_t end_time = now + gt + rt - 5;
time_t actualTime = end_time + 5;
int br = 0;
while ((char)keyboard != 'q' && (char)keyboard != 27&&br==0)
{
   if (cf % 1800 == 0)
   {   cf = cf + gt*(pos - 1) ; // for synced data as current frame would also move ahead
      int slpt = 0;
      slpt = gt-10;
         //this_thread::sleep_for(chrono::seconds(slpt));// not needed in static videos bt for real time feeds
      cout << "started refernce calculation for " << pos << endl;
      cf = cf + slpt;
      capture.set(CV_CAP_PROP_POS_FRAMES, nf * cf);
      cf = cf + 10;
      while ((char)keyboard != 'q' && (char)keyboard != 27) {
         time_t n = time(0);
         if (n >=actualTime)
         {
            d = 1;// boolean for default
```

```
    display(pos,gt, n,actualTime);
    emit displaySignal(pos,gt, n,actualTime);
    end_time = end_time + gt + rt;
    if (end_time > maxTime)
    {
        end_time = end_time - gt - rt;
        br = 1;
    }
    actualTime = end_time + 5;
}
//read the current frame
if (!capture.read(frame)) {
    cerr << "Unable to read next frame." << endl;
    cerr << "Exiting..." << endl;
    exit(4);
}
//update the background model
pMOG2->operator()(frame, fgMaskMOG2, 0.01);
//get the frame number and write it on the current frame
pMOG2->getBackgroundImage(segm);
stringstream ss;
ss << capture.get(CV_CAP_PROP_POS_FRAMES);
string frameNumberString = ss.str();
int a = stoi(frameNumberString);
if (a == (nf * (cf)))
{
    break;
}
keyboard = waitKey(30);
```

```
    }
    sname = "BG" + to_string(pos);
    namedWindow(sname, WINDOW_NORMAL);
    imshow(sname, segm);
    refs[pos] = segm;
    hough(segm, p, pos);
}
else
{
    int slpt = gt; //sleep during green signal
    this_thread::sleep_for(chrono::seconds(slpt));
}
if (d == 0)
{
    time_t n = time(0);
    if (n >= actualTime)
    {
        d = 1;// boolean for default
        display(pos, gt, n, actualTime);
        emit displaySignal(pos, gt, n, actualTime);
        end_time = end_time + gt + rt;
        if (end_time > maxTime)
        {
            end_time = end_time - gt - rt;
            br = 1;
        }
        actualTime = end_time + 5;
    }
    else
```

```
        {
            time_t c = time(0);
            int remtime = end_time - c;
            this_thread::sleep_for(chrono::seconds(remtime));
            cout << "started traffic calculation for " << pos << endl;
            cf = rt + cf - 5;
            capture.set(CV_CAP_PROP_POS_FRAMES, nf * cf);
            //capture.set(CV_CAP_PROP_POS_FRAMES, 1100);
            capture.read(f);
            //cvtColor(f, f, CV_RGB2GRAY);
            sname = "Traffic" + to_string(pos);
            namedWindow(sname, WINDOW_NORMAL);
            imshow(sname, f);
            diffLane(refs[pos], f, p, pos, gt, actualTime);
            end_time = end_time + gt + rt;
            if (end_time > maxTime)
            {
                end_time = end_time - gt - rt;
                br = 1;
            }
            actualTime = end_time + 5;
        }
    }
    d = 0;
  }
  capture.release();
  waitKey(0);
}
```

```
void MyProcess::display(int pos,int gt, time_t n,time_t a)
{
  cout << "Lane No: " << pos << "  ";
  cout <<"Green signal time: "<< gt << "  ";
  tm *ltm = localtime(&n);
        string ctime= to_string(ltm->tm_hour)+ ":"+ to_string(ltm->tm_min) + ":"+
to_string(ltm->tm_sec) ;
  cout << "Calculation Time: " << ctime<<"  ";

  ltm = localtime(&a);
      string atime = to_string(ltm->tm_hour) + ":" + to_string(ltm->tm_min) + ":" +
to_string(ltm->tm_sec);
  cout << "Green signal Start Time: " << atime << endl;
}

cv::Mat MyProcess::imageEnhance(cv::Mat src, int r1, int s1, int r2, int s2)
{
  Mat new_image = src.clone();
  for (int y = 0; y < src.rows; y++) {
    for (int x = 0; x < src.cols; x++) {
      for (int c = 0; c < 3; c++) {
        int output = computeOutput(src.at<Vec3b>(y, x)[c], r1, s1, r2, s2);
        new_image.at<Vec3b>(y, x)[c] = saturate_cast<uchar>(output);
      }
    }
  }
  return new_image;
}
```

```
int MyProcess::computeOutput(int x, int r1, int s1, int r2, int s2)
{
  float result;
  if (0 <= x && x <= r1) {
     result = s1 / r1 * x;
  }
  else if (r1 < x && x <= r2) {
     result = ((s2 - s1) / (r2 - r1)) * (x - r1) + s1;
  }
  else if (r2 < x && x <= 255) {
     result = ((255 - s2) / (255 - r2)) * (x - r2) + s2;
  }
  return (int)result;
}

void MyProcess::hough(cv::Mat src, int p[], int pos)
{
  Mat cdst, dst;
  int ht = src.rows;
  int wt = src.cols;
  src = imageEnhance(src, 50, 0, 150, 255);
  cv::cvtColor(src, src, CV_BGR2GRAY);
  Canny(src, dst, 50, 200, 3);
  cv::cvtColor(dst, cdst, CV_GRAY2BGR);
  int rmost = 0, rmostd = 0;
  int div[4];
  int last[4];
  vector<Vec4i> lines;
  cv::HoughLinesP(dst, lines, 1, CV_PI / 180, 100, 100, 200);
```

```
int thr = wt / 10;
for (size_t i = 0; i < lines.size(); i++)
{
   Vec4i l = lines[i];
   int x1 = l[0];
   int y1 = l[1];
   int x2 = l[2];
   int y2 = l[3];
   float dist = sqrt((x2 - x1)*(x2 - x1) + (y2 - y1)*(y2 - y1));
   float m = 0;
   if (x2 != x1)
      m = ((y2 - y1)*1.0) / (x2 - x1);

    if (dist > ((1.0*ht) / 3))// length more than 1/3ht n angle between 30 to 150 for other lanes and 80 to 100 for divider
    {
      if (m <= -5 || m >= 5)// divider
      {
         int xmaxd = x1>x2 ? x1 : x2;
         if (rmostd <= xmaxd && (i == 1 || (xmaxd - last[0]<thr) && (xmaxd - last[2]<thr)))
         {
            rmostd = xmaxd;
            div[0] = x1;
            div[1] = y1;
            div[2] = x2;
            div[3] = y2;
         }
      }
      else if (m <= -0.6 || m >= 0.6)// baaki
```

```
        {
            int xmax = x1>x2 ? x1 : x2;
            if (rmost <= xmax)
            {
                rmost = xmax;
                last[0] = x1;
                last[1] = y1;
                last[2] = x2;
                last[3] = y2;
            }
        }
    }
}
line(cdst, Point(last[0], last[1]), Point(last[2], last[3]), Scalar(0, 0, 255), 3, CV_AA);
line(cdst, Point(div[0], div[1]), Point(div[2], div[3]), Scalar(0, 255, 0), 3, CV_AA);
if (last[1] > last[3])
{
    int e = last[0];
    last[0] = last[2];
    last[2] = e;
    e = last[1];
    last[1] = last[3];
}
if (div[1] > div[3])
{
    int e = div[0];
    div[0] = div[2];
    div[2] = e;
    e = div[1];
```

```
    div[1] = div[3];
  }

  for (int i = 0; i < 4; i++)
    p[i] = div[(i + 2) % 4];
  for (int i = 4; i < 8; i++)
    p[i] = last[i - 4];
  p[8] = last[2];
  p[9] = (div[3] > last[3] ? div[3] : last[3]);

  string sname = "lanes detected" + to_string(pos);
  //cout << sname;
  namedWindow(sname, WINDOW_NORMAL);
  cv::imshow(sname, cdst);

}

void MyProcess::diffLane(Mat backgroundImage, Mat currentImage, int p[], int pos,int gt,time_t
act)
{
  int a[1];
  int r = currentImage.rows, c = currentImage.cols;
  backgroundImage = imageEnhance(backgroundImage, 50, 0, 150, 255);
  currentImage = imageEnhance(currentImage, 50, 0, 150, 255);
  Mat bgm = maskLane(backgroundImage, r, c, a, p);
  string sname = "cbg" + to_string(pos);
  namedWindow(sname, CV_WINDOW_NORMAL);
  imshow(sname, bgm);
  Mat cgm = maskLane(currentImage, r, c, a, p);
```

```
sname = "ccg" + to_string(pos);
namedWindow(sname, CV_WINDOW_NORMAL);
imshow(sname, cgm);
cv::Mat diffImage;
cv::absdiff(bgm, cgm, diffImage);
cv::Mat foregroundMask = cv::Mat::zeros(diffImage.rows, diffImage.cols, CV_8UC1);
float threshold = 30.0f;
float dist;
for (int j = 0; j<diffImage.rows; ++j)
   for (int i = 0; i<diffImage.cols; ++i)
   {
      cv::Vec3b pix = diffImage.at<cv::Vec3b>(j, i);
      dist = (pix[0] * pix[0] + pix[1] * pix[1] + pix[2] * pix[2]);
      dist = sqrt(dist);
      if (dist>threshold)
      {
         foregroundMask.at<unsigned char>(j, i) = 255;
      }
   }
sname = "fg" + to_string(pos);
namedWindow(sname, CV_WINDOW_NORMAL);
imshow(sname, foregroundMask);
//imshow("fg1", diffImage);
int cf = countNonZero(foregroundMask);
float percentage = ((1.0*cf) / a[0]) * 100;
cout <<percentage << endl;
int t = timecalculate(percentage, gt);
time_t now = time(0);
display(pos, t, now, act);
```

```
  emit displaySignal(pos,t,now,act);
  waitKey(0);
}

int MyProcess::timecalculate(float p,int gt )
{
  if (p <= 15)
     return gt*0.5;
  else if (p <= 30)
     return gt*0.8;
  else if (p <= 40)
     return gt;
  else if (p <= 50)
     return gt*1.2;
  else if (p <= 75)
     return gt*1.5;
  else return gt*1.75;
}

Mat MyProcess::maskLane(Mat fg, int new_h, int new_w, int a[], int p[])
{
  Mat mask(new_h, new_w, CV_8UC1, cv::Scalar(0));
  vector< vector<Point> >  co_ordinates;
  co_ordinates.push_back(vector<Point>());
  for (int i = 0; i<10; i += 2)
     co_ordinates[0].push_back(Point(p[i], p[i + 1]));
  drawContours(mask, co_ordinates, 0, Scalar(255), CV_FILLED, 8);
  Mat fgm;
  fg.copyTo(fgm, mask);
```

```
  a[0] = contourArea(co_ordinates[0]);
  return fgm;
}
```

4. output.cpp

```
#include "output.h"
#include "ui_output.h"
#include <qnamespace.h>
#include <iostream>
#include <thread>
using namespace std;

Output::Output(QWidget *parent) :
  QWidget(parent),
  ui(new Ui::Output)
{
  ui->setupUi(this);
  ui->heading->setAlignment(Qt::AlignHCenter);
  this->setWindowTitle("Result");
}

int nol;
void Output::receiveMsg(int lanes){
  nol=lanes;
  display();
}


void Output::display(){
```

```
  QString si,sj;
  for(int i=1;i<=nol;i++){//for rows
     for(int j=0;j<4;j++){//for 4 columns
        label=new QLabel;
        si=QString::number(i);
        sj=QString::number(j);
        label->setObjectName("label"+si+sj);
        label->setText(si+sj);
        label->setAutoFillBackground(1);//1=true, 0=false
        label->setStyleSheet("background-color: rgb(255, 255, 255);");
        label->setAlignment(Qt::AlignCenter);
        ui->gridLayout->addWidget(label);
     }
  }
}

void Output::show(int pos,int gt, time_t n,time_t a){
  QString ln="label"+pos;
  QLabel *l;
  ln=ln+"0";
  l=this->findChild<QLabel *>(ln);
  l->setText(pos+"");

  ln=ln+"1";
  l=this->findChild<QLabel *>(ln);
  l->setText(gt+"");

  tm *ltm = localtime(&n);
```

```
        string ctime= to_string(ltm->tm_hour)+ ":"+ to_string(ltm->tm_min) + ":"+
to_string(ltm->tm_sec) ;
  cout << "Calculation Time: " << ctime<<"  ";
  ln=ln+"2";
  l=this->findChild<QLabel *>(ln);
  l->setText(QString::fromStdString(ctime+""));

  ltm = localtime(&a);
      string atime = to_string(ltm->tm_hour) + ":" + to_string(ltm->tm_min) + ":" +
to_string(ltm->tm_sec);
  cout << "Green signal Start Time: " << atime << endl;
  ln=ln+"3";
  l=this->findChild<QLabel *>(ln);
  l->setText(QString::fromStdString(atime+""));

}

Output::~Output()
{
  delete ui;
}
```

User Interface:

1. **MainWindow.ui**

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
<class>MainWindow</class>
<widget class="QMainWindow" name="MainWindow">
 <property name="geometry">
```

```
 <rect>
  <x>0</x>
  <y>0</y>
  <width>960</width>
  <height>540</height>
 </rect>
</property>
<property name="windowTitle">
 <string>MainWindow</string>
</property>
<widget class="QWidget" name="centralWidget">
 <widget class="QWidget" name="verticalLayoutWidget_3">
  <property name="geometry">
   <rect>
    <x>280</x>
    <y>240</y>
    <width>151</width>
    <height>201</height>
   </rect>
  </property>
  <layout class="QVBoxLayout" name="verticalLayout_2">
   <property name="spacing">
    <number>10</number>
   </property>
   <property name="sizeConstraint">
    <enum>QLayout::SetFixedSize</enum>
   </property>
   <item>
    <layout class="QVBoxLayout" name="videoLayout">
```

```
   <property name="sizeConstraint">
    <enum>QLayout::SetFixedSize</enum>
   </property>
  </layout>
 </item>
 <item>
  <widget class="QPushButton" name="submitButton">
   <property name="styleSheet">
    <string notr="true">background-color: rgb(255, 255, 255);</string>
   </property>
   <property name="text">
    <string>SUBMIT</string>
   </property>
  </widget>
 </item>
</layout>
</widget>
<widget class="QWidget" name="verticalLayoutWidget">
 <property name="geometry">
  <rect>
   <x>270</x>
   <y>90</y>
   <width>471</width>
   <height>131</height>
  </rect>
 </property>
 <layout class="QVBoxLayout" name="verticalLayout_3">
  <property name="spacing">
   <number>3</number>
```

```
  </property>
  <property name="sizeConstraint">
   <enum>QLayout::SetDefaultConstraint</enum>
  </property>
  <item>
   <layout class="QHBoxLayout" name="horizontalLayout_2" stretch="3,3">
    <property name="spacing">
     <number>0</number>
    </property>
    <property name="sizeConstraint">
     <enum>QLayout::SetFixedSize</enum>
    </property>
    <item>
     <widget class="QLabel" name="label_2">
      <property name="font">
       <font>
        <family>Arial</family>
        <pointsize>10</pointsize>
        <weight>75</weight>
        <bold>true</bold>
       </font>
      </property>
      <property name="autoFillBackground">
       <bool>false</bool>
      </property>
      <property name="styleSheet">
       <string notr="true">background-color: rgb(255, 255, 255);
border-color: rgb(0, 0, 0);</string>
      </property>
```

```
   <property name="text">
    <string>Time of green light</string>
   </property>
   <property name="alignment">
    <set>Qt::AlignCenter</set>
   </property>
  </widget>
 </item>
 <item>
  <widget class="QLineEdit" name="greenTime">
   <property name="autoFillBackground">
    <bool>false</bool>
   </property>
   <property name="styleSheet">
    <string notr="true">background-color: rgb(255, 255, 255);</string>
   </property>
  </widget>
 </item>
</layout>
</item>
<item>
 <layout class="QHBoxLayout" name="horizontalLayout_4" stretch="3,3">
  <property name="spacing">
   <number>0</number>
  </property>
  <property name="sizeConstraint">
   <enum>QLayout::SetFixedSize</enum>
  </property>
  <item>
```

```
      <widget class="QLabel" name="label_4">
       <property name="font">
        <font>
         <family>Arial</family>
         <pointsize>10</pointsize>
         <weight>75</weight>
         <bold>true</bold>
        </font>
       </property>
       <property name="autoFillBackground">
        <bool>false</bool>
       </property>
       <property name="styleSheet">
        <string notr="true">background-color: rgb(255, 255, 255);
border-color: rgb(0, 0, 0);</string>
       </property>
       <property name="text">
        <string>Time of red light</string>
       </property>
       <property name="alignment">
        <set>Qt::AlignCenter</set>
       </property>
      </widget>
     </item>
     <item>
      <widget class="QLineEdit" name="redTime">
       <property name="sizePolicy">
        <sizepolicy hsizetype="Expanding" vsizetype="Maximum">
         <horstretch>0</horstretch>
```

```
      <verstretch>0</verstretch>
     </sizepolicy>
    </property>
    <property name="autoFillBackground">
     <bool>false</bool>
    </property>
    <property name="styleSheet">
     <string notr="true">background-color: rgb(255, 255, 255);</string>
    </property>
   </widget>
  </item>
 </layout>
</item>
<item>
 <layout class="QHBoxLayout" name="horizontalLayout" stretch="3,3">
  <property name="spacing">
   <number>0</number>
  </property>
  <property name="sizeConstraint">
   <enum>QLayout::SetDefaultConstraint</enum>
  </property>
  <item>
   <widget class="QLabel" name="label">
    <property name="font">
     <font>
      <family>Arial</family>
      <pointsize>10</pointsize>
      <weight>75</weight>
      <bold>true</bold>
```

```
        </font>
       </property>
       <property name="autoFillBackground">
        <bool>false</bool>
       </property>
       <property name="styleSheet">
        <string notr="true">background-color: rgb(255, 255, 255);
border-color: rgb(0, 0, 0);</string>
       </property>
       <property name="text">
        <string>No. of lanes</string>
       </property>
       <property name="alignment">
        <set>Qt::AlignCenter</set>
       </property>
      </widget>
     </item>
     <item>
      <widget class="QLineEdit" name="lanesEdit">
       <property name="autoFillBackground">
        <bool>true</bool>
       </property>
       <property name="styleSheet">
        <string notr="true">background-color: rgb(255, 255, 255);</string>
       </property>
      </widget>
     </item>
    </layout>
   </item>
```

```
   <item>
    <widget class="QLabel" name="demo">
     <property name="font">
      <font>
       <family>Arial</family>
       <pointsize>10</pointsize>
       <weight>75</weight>
       <bold>true</bold>
      </font>
     </property>
     <property name="autoFillBackground">
      <bool>false</bool>
     </property>
     <property name="styleSheet">
      <string notr="true">background-color: rgb(255, 255, 255);
border-color: rgb(0, 0, 0);</string>
     </property>
     <property name="text">
      <string>NOTE: Select the videos for the lanes in order the green signal goes</string>
     </property>
     <property name="alignment">
      <set>Qt::AlignCenter</set>
     </property>
    </widget>
   </item>
  </layout>
 </widget>
 <widget class="QLabel" name="heading">
  <property name="geometry">
```

```
  <rect>
   <x>430</x>
   <y>20</y>
   <width>151</width>
   <height>51</height>
  </rect>
 </property>
 <property name="font">
  <font>
   <family>Algerian</family>
   <pointsize>35</pointsize>
  </font>
 </property>
 <property name="text">
  <string>AVTC</string>
 </property>
 <property name="alignment">
  <set>Qt::AlignCenter</set>
 </property>
</widget>
</widget>
</widget>
<layoutdefault spacing="6" margin="11"/>
<resources/>
<connections/>
</ui>
```

2. Output.ui

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<ui version="4.0">
<class>Output</class>
<widget class="QWidget" name="Output">
 <property name="geometry">
  <rect>
   <x>0</x>
   <y>0</y>
   <width>960</width>
   <height>540</height>
  </rect>
 </property>
 <property name="windowTitle">
  <string>Form</string>
 </property>
 <widget class="QWidget" name="gridLayoutWidget">
  <property name="geometry">
   <rect>
    <x>260</x>
    <y>120</y>
    <width>489</width>
    <height>291</height>
   </rect>
  </property>
  <layout class="QGridLayout" name="gridLayout">
   <property name="sizeConstraint">
    <enum>QLayout::SetDefaultConstraint</enum>
   </property>
   <item row="0" column="2">
    <widget class="QLabel" name="label_3">
```

```
  <property name="font">
   <font>
    <family>Arial</family>
    <pointsize>10</pointsize>
    <weight>75</weight>
    <bold>true</bold>
   </font>
  </property>
  <property name="styleSheet">
   <string notr="true">background-color: rgb(255, 255, 255);</string>
  </property>
  <property name="text">
   <string>Time needed to calculate</string>
  </property>
  <property name="alignment">
   <set>Qt::AlignCenter</set>
  </property>
  <property name="wordWrap">
   <bool>true</bool>
  </property>
 </widget>
</item>
<item row="0" column="0">
 <widget class="QLabel" name="label">
  <property name="font">
   <font>
    <family>Arial</family>
    <pointsize>10</pointsize>
    <weight>75</weight>
```

```
   <bold>true</bold>
  </font>
 </property>
 <property name="styleSheet">
  <string notr="true">background-color: rgb(255, 255, 255);</string>
 </property>
 <property name="text">
  <string>Lane Number</string>
 </property>
 <property name="alignment">
  <set>Qt::AlignCenter</set>
 </property>
 <property name="wordWrap">
  <bool>true</bool>
 </property>
</widget>
</item>
<item row="0" column="1">
 <widget class="QLabel" name="label_2">
  <property name="font">
   <font>
    <family>Arial</family>
    <pointsize>10</pointsize>
    <weight>75</weight>
    <bold>true</bold>
   </font>
  </property>
  <property name="styleSheet">
   <string notr="true">background-color: rgb(255, 255, 255);</string>
```

```
  </property>
  <property name="text">
   <string>Green Time</string>
  </property>
  <property name="alignment">
   <set>Qt::AlignCenter</set>
  </property>
  <property name="wordWrap">
   <bool>true</bool>
  </property>
 </widget>
</item>
<item row="0" column="3">
 <widget class="QLabel" name="label_4">
  <property name="font">
   <font>
    <family>Arial</family>
    <pointsize>10</pointsize>
    <weight>75</weight>
    <bold>true</bold>
   </font>
  </property>
  <property name="styleSheet">
   <string notr="true">background-color: rgb(255, 255, 255);</string>
  </property>
  <property name="text">
   <string>Actually when green signal should start</string>
  </property>
  <property name="alignment">
```

```
    <set>Qt::AlignCenter</set>
   </property>
   <property name="wordWrap">
    <bool>true</bool>
   </property>
  </widget>
 </item>
</layout>
</widget>
<widget class="QLabel" name="heading">
 <property name="geometry">
  <rect>
   <x>430</x>
   <y>20</y>
   <width>151</width>
   <height>51</height>
  </rect>
 </property>
 <property name="font">
  <font>
   <family>Algerian</family>
   <pointsize>35</pointsize>
  </font>
 </property>
 <property name="text">
  <string>AVTC</string>
 </property>
 <property name="alignment">
  <set>Qt::AlignCenter</set>
```

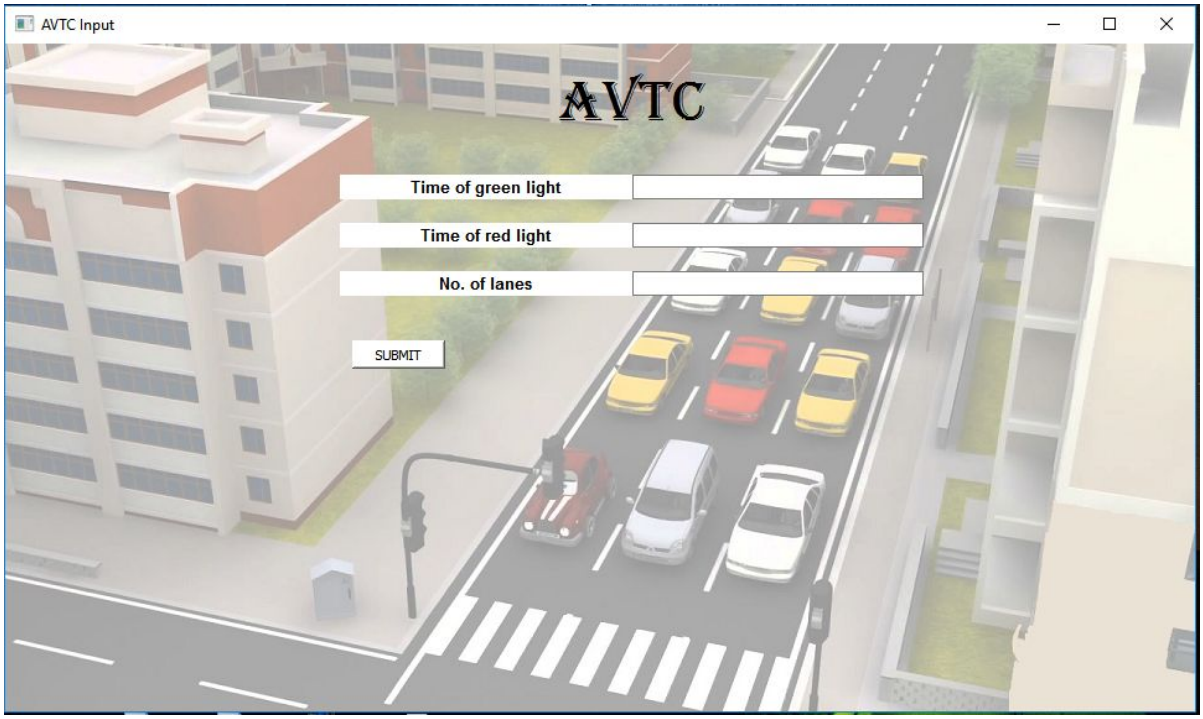
```
 </property>
 </widget>
</widget>
<resources/>
<connections/>
</ui>
```

Chapter 6

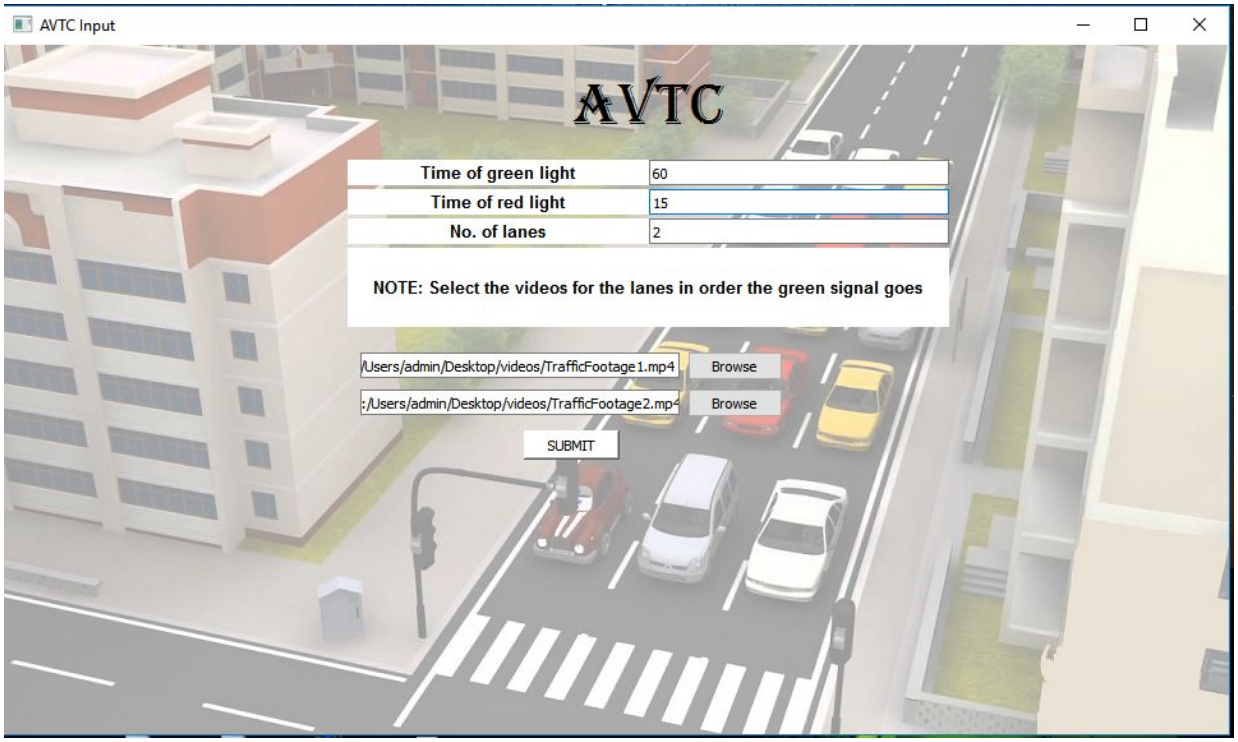
System Implementation

6.1 Snapshots

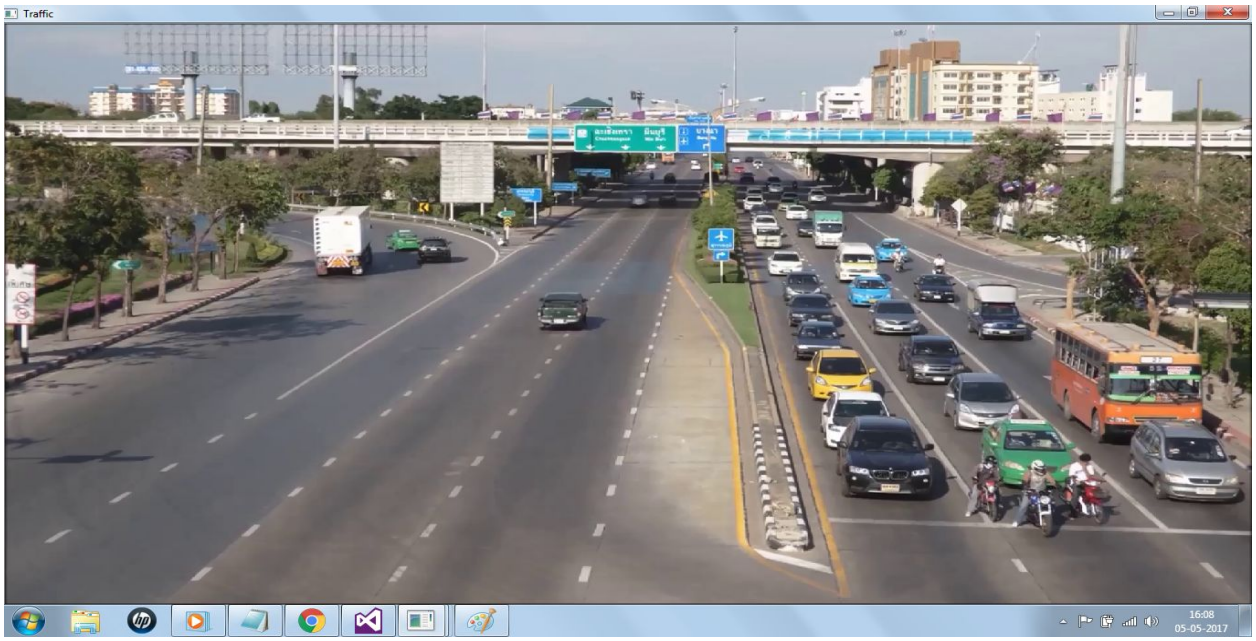
USER INTERFACE



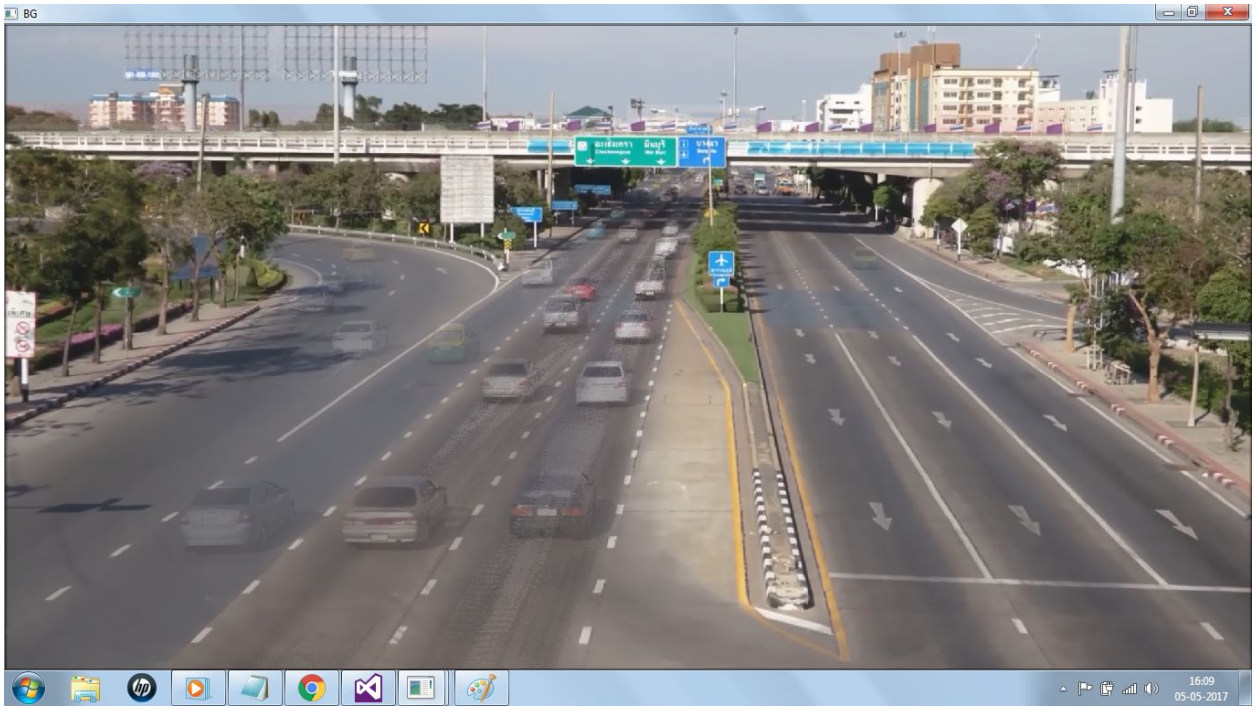
AFTER ENTERING VALUES



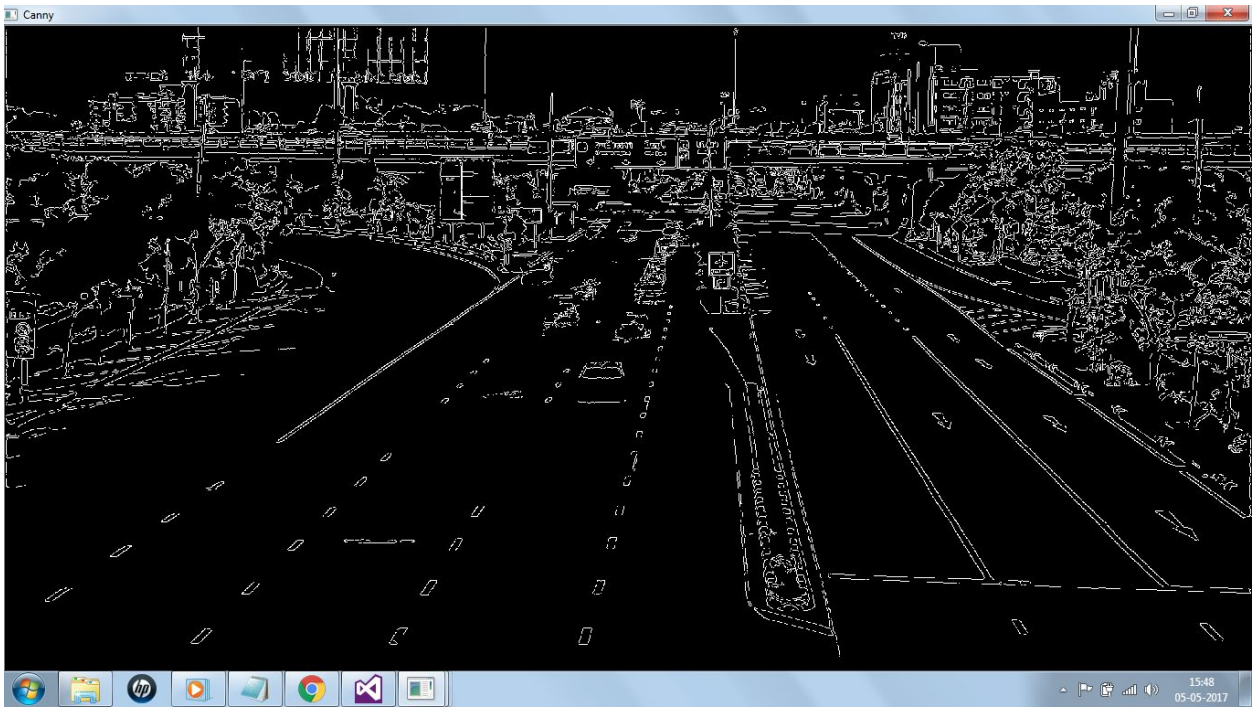
VIDEO CAPTURE



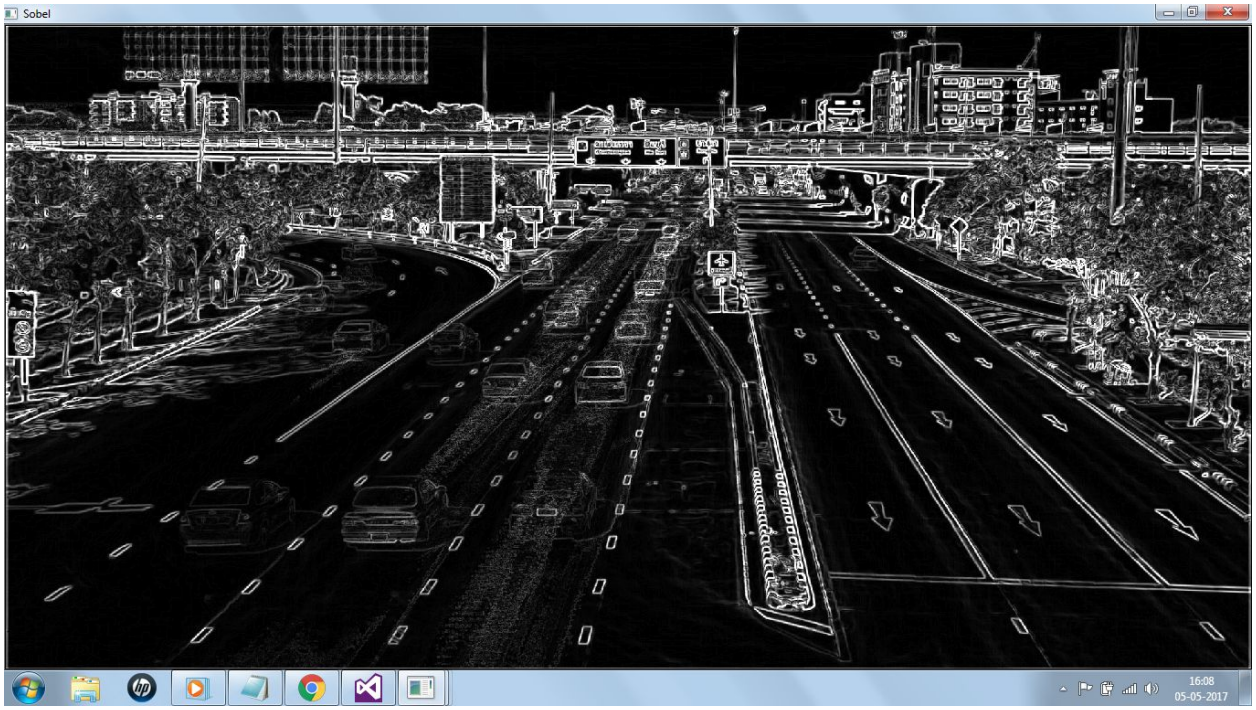
BACKGROUND SUBTRACTION



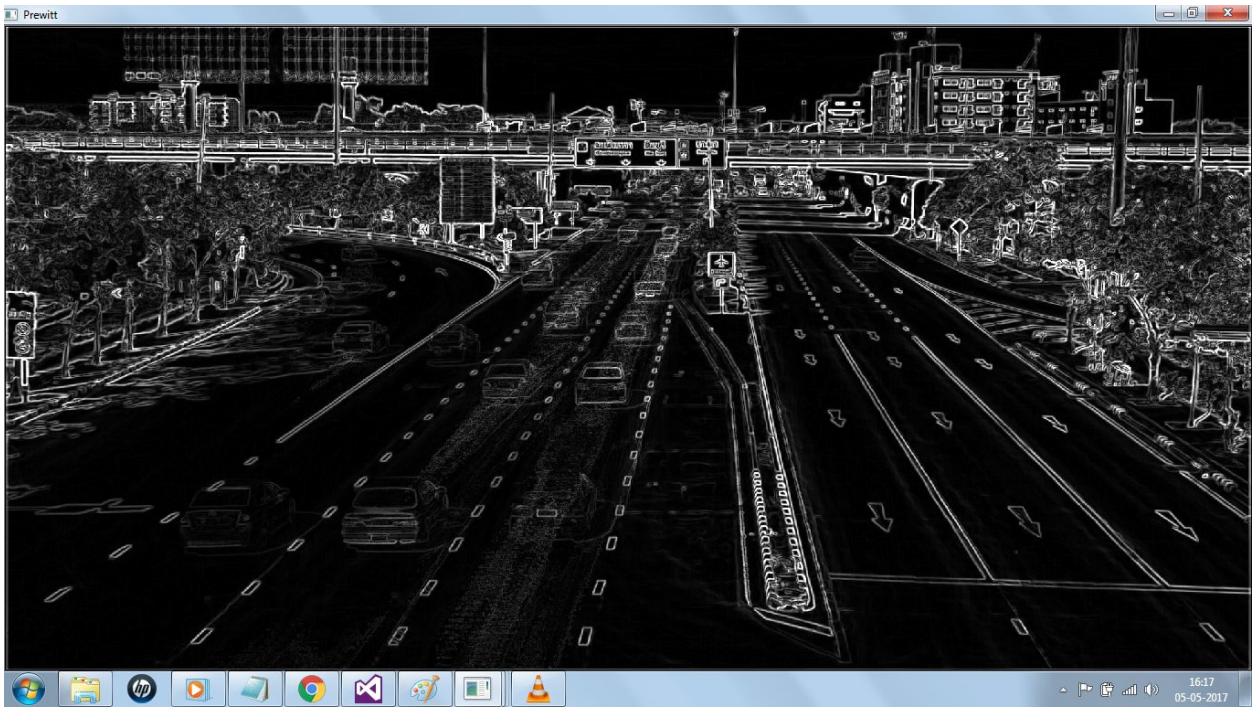
CANNY



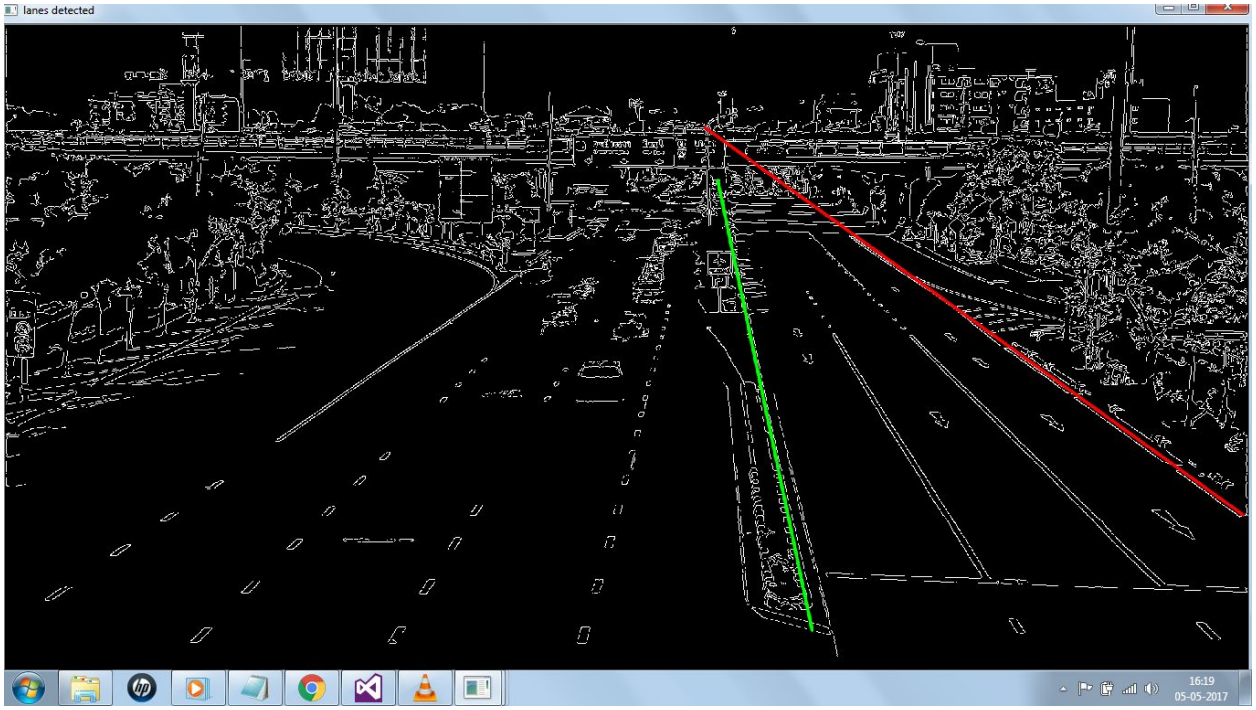
SOBEL



PREWITT

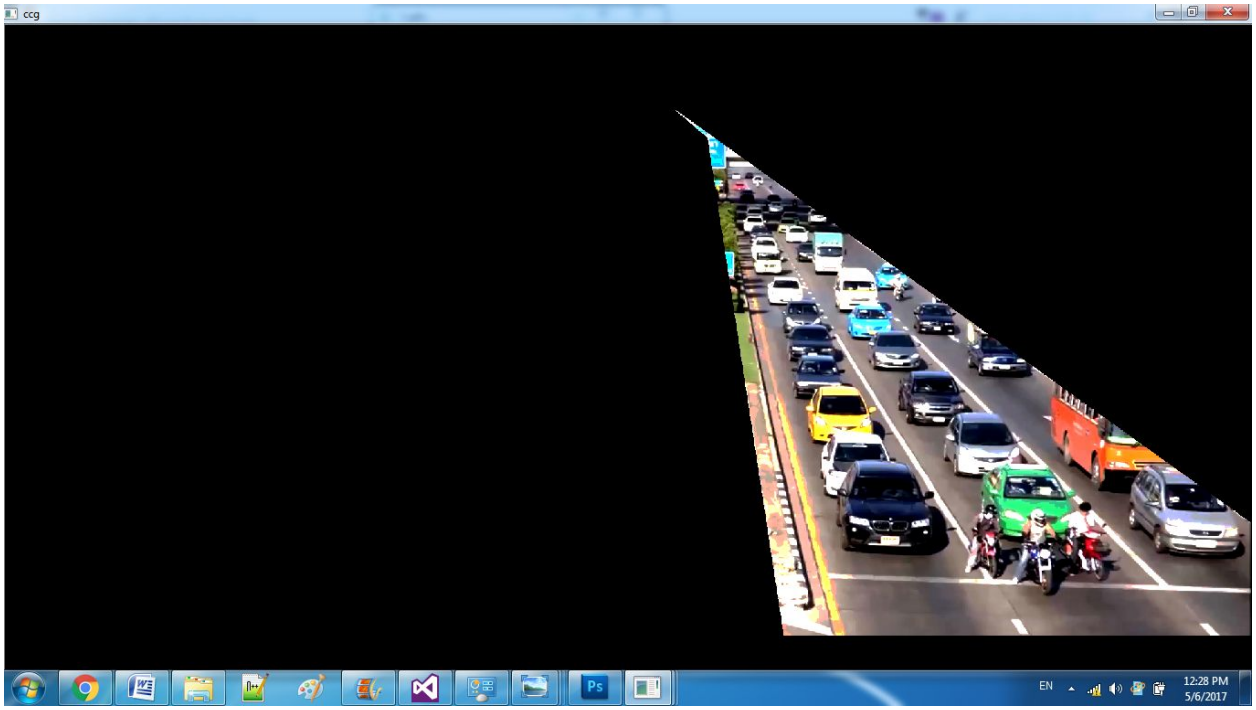


HOUGH TRANSFORM ON CANNY

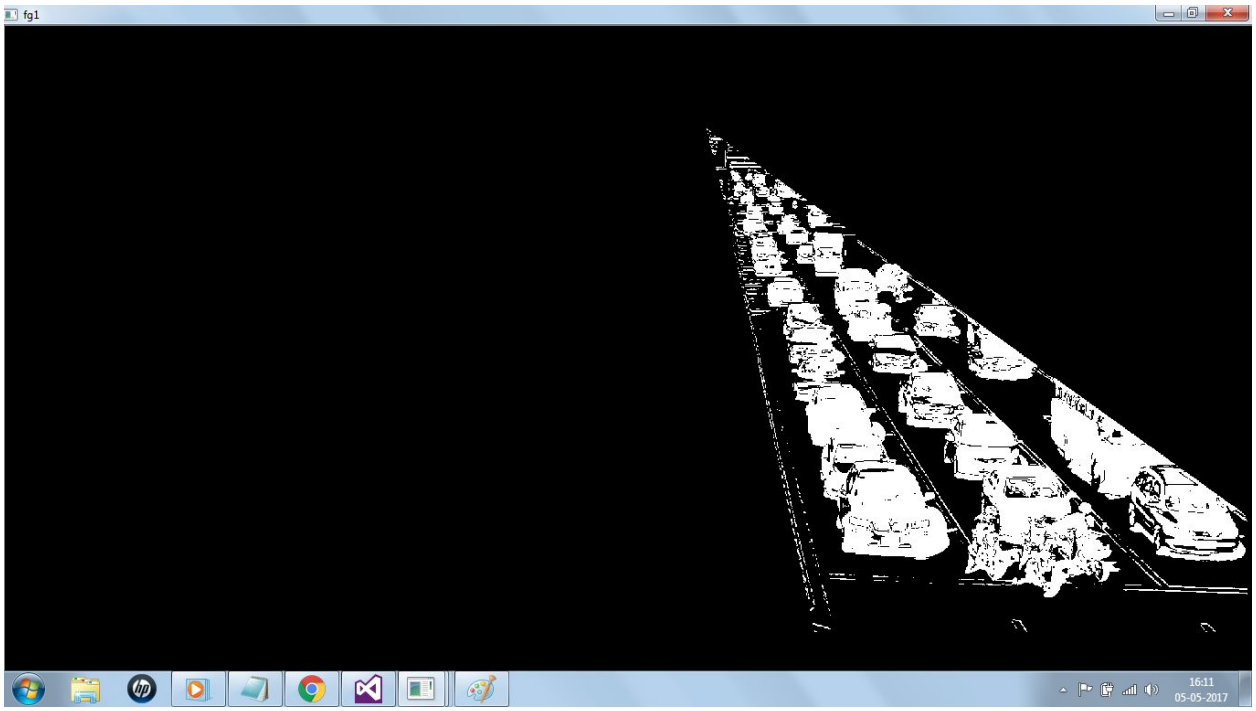


REFERENCE CROPPED IMAGE

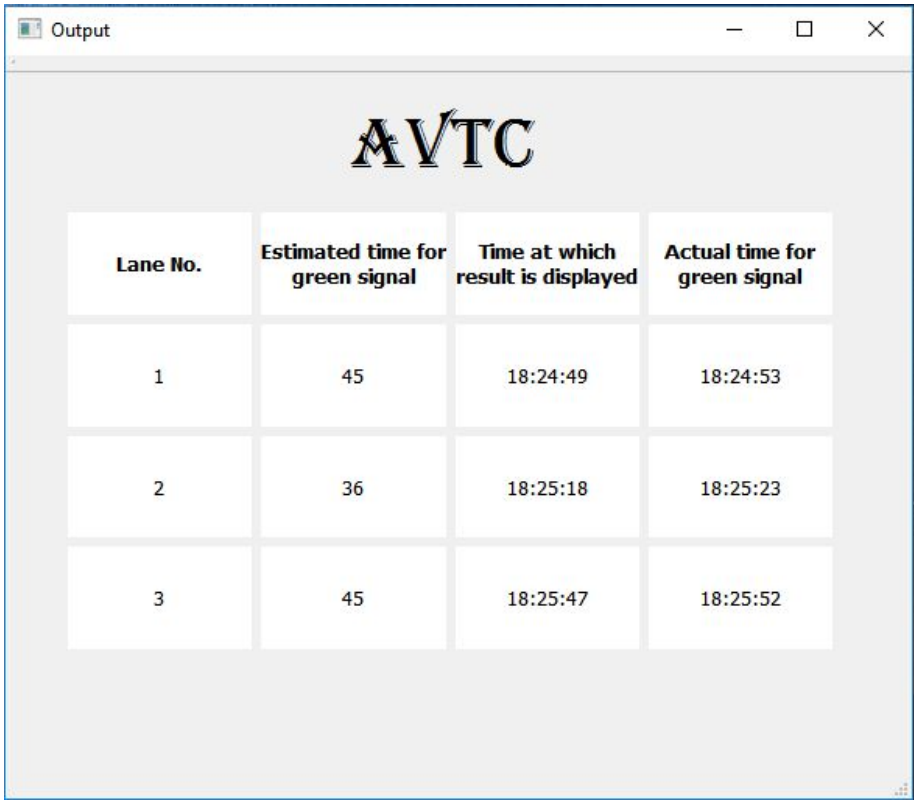
TRAFFIC CROPPED IMAGE



DENSITY



RESULT



Chapter 7

Conclusion and Future Work

7.1 Conclusion

“Adaptive Vehicular Traffic Controller ” technique that we propose overcomes all the limitations of the earlier (in use) techniques used for controlling the traffic. Earlier in automatic traffic control use of timer had a drawback that the time is being wasted by green light on the empty. This technique avoids this problem. Upon comparison of various edge detection algorithms, it was inferred that Canny Edge Detector technique is the most efficient one. The project demonstrates that image processing is a far more efficient method of traffic control as compared to traditional techniques. The use of our technique removes the need for extra hardware such as sound sensors. The increased response time for these vehicles is crucial for the prevention of loss of life. Major advantage is the variation in signal time which control appropriate traffic density using Image matching. The accuracy in calculation of time due to single moving camera depends on the registration position while facing road every time.

7.2 Future Work

- **Digital Signal Processor:** It can avoid heavy investment in industrial control computer while obtaining improved computational power and optimised system structure.The hardware implementations would enable the project to be used in real time applications.
- **On-spot Integration:** The proposed system currently works at traffic control room where video will be relayed in real time which may cause delay due to network traffic. We can extend the implementation to be directly hard coded on an Integrated Chip with specific CCTV sites.
- **Emergency Detection:** An algorithm can be made to detect emergency situation such as an ambulance or police van with beacon on. As soon as any such vehicle is detected that lane will get first priority to move ahead.

Chapter 8

References

1. Nikita Sankhe, Poonam Sonar , Deven Patel , “An Overview of Image processing for traffic applications”, IOSR Journal of Engineering (IOSRJEN).
2. V. Kastrinaki, M. Zervakis*, K. Kalaitzakis, “A survey of video processing techniques for traffic applications”, Image and Vision Computing 21 (2003) 359–381.
3. Vikramaditya dangi by, Amol Parab, Kshitij Pawar & S.S Rathod “Image processing based intelligent traffic controller”, Undergraduate Academic Research Journal (UARJ), ISSN :2278 – 1129, Volume-1, Issue-2, 2012.
4. A. K. Jain, Fundamentals of Digital Image Processing, Englewood Cliffs, NJ: Prentice Hall, 1989.
5. *Image Analysis and Mathematical Morphology* by Jean Serra, ISBN 0-12-637240-3 (1982).
6. Tarun Kumar and Karun Verma, A Theory Based on Conversion of RGB image to Gray image, International Journal of Computer Applications (0975 – 8887) Volume 7– No.2, September 2010.
7. Y.Ramadevi, T.Sridevi, B.Poornima, B.Kalyani, SEGMENTATION AND OBJECT RECOGNITION USING EDGE DETECTION TECHNIQUES, International Journal of Computer Science & Information Technology (IJCSIT), Vol 2, No 6, December 2010.
8. Ajit Danti, Jyoti .Y. Kulkarni and P.S.Hiremath, A Realtime Road Boundary Detection and Vehicle Detection for Indian Roads, International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868, Foundation of Computer Science FCS, New York, USA, Volume 5– No.4, March 2013.
9. Theo Gevers*, Arnold W.M. Smeulders, Color-based object recognition, Pattern Recognition 32 (1999) 453—464.
10. Ammu M Kumar and Philomina Simon, REVIEW OF LANE DETECTION AND TRACKING ALGORITHMS IN ADVANCED DRIVER ASSISTANCE SYSTEM, International Journal of Computer Science & Information Technology (IJCSIT) Vol 7, No 4, August 2015.

11. Nasim Arshad, Kwang-Seok Moon, Seung-Seob Park, and Jong-Nam Kim, Lane Detection with Moving Vehicles Using Color Information, Proceedings of the World Congress on Engineering and Computer Science 2011 Vol I WCECS 2011, October 19-21, 2011.
12. Mr. D. W. Chinchkhede & Mr. N. J. Uke, IMAGE SEGMENTATION IN VIDEO SEQUENCES USING MODIFIED BACKGROUND SUBTRACTION, International Journal of Computer Science & Information Technology (IJCSIT) Vol 4, No 1, Feb 2012.
13. Parichita Basak, Ramandeep Kaur, Intelligent Traffic Control System using Image Processing, International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064.
14. Deepjoy Das and Dr. Sarat Saharia, Implementation And Performance Evaluation Of Background Subtraction Algorithms, International Journal on Computational Sciences & Applications (IJCSA) Vol.4, No.2, April 2014.
15. Sen-Ching S. Cheung and Chandrika Kamath, Robust techniques for background subtraction in urban traffic video.
16. Pallavi Choudekar, Sayanti Banerjee and M.M.Muju, REAL TIME TRAFFIC LIGHT CONTROL USING IMAGE PROCESSING, Indian Journal of Computer Science and Engineering (IJCSE).
17. Vismay Pandit, Jinesh Doshi, Dhruv Mehta, Ashay Mhatre and Abhilash Janardhan, Smart Traffic Control System Using Image Processing, International Journal of Emerging Trends & Technology in Computer Science (IJETTCS).
18. Mahesh C. Pawaskar, Mr. N. S.Narkhede and Mr. Saurabh S. Athalye, Detection Of Moving Object Based On Background Subtraction, International Journal of Emerging Trends & Technology in Computer Science (IJETTCS).