



Addis Ababa University

Addis Ababa Institute of Technology

School of Electrical and Computer Engineering

Real Time Traffic Light Control with Image Processing

By: Dawit Samuel ATR/1378/05

Biruk Tesfaw ATR/4514/05

Advisors' Name: Belayneh Melka

Co advisor: Nebyou Yonas

Date of submission: Jun, 23, 2017

Abstract

The sheer volume of vehicles in countries around the world is growing at a staggering rate, Ethiopia is no exception. Inevitably with the increase of traffic density comes an increase in amount of traffic related accidents, the Addis Ababa Transport Authority reports that more than 12,000,000 birr is being lost every year because of traffic accidents.

This paper documents the construction of an intelligent traffic system based on computer vision which will optimize traffic flow throughout a city. The system makes use of cameras on various intersections to analyze current traffic data and make traffic control decisions. Furthermore the system collects and stores data from each location to be used for further processing and statistical analysis.

Acknowledgement

The scope of the project is a bit wide involving construction of the video analysis algorithm, the control logic, the server storage and display of long term data along with the construction of a model to showcase the project. We would like to extend our gratitude to our advisor Belayneh melka for guiding us in the process of making this application and personally providing us with the microcontroller kit for the model.

Table of contents

Abstract	i
Acknowledgement	ii
Table of contents	iii
Table of Figures	v
Objective	1
General Objectives	1
Specific Objectives	1
Introduction.....	2
Background	4
Vehicle detection with Image processing	4
Background subtraction	4
Frame Differencing	4
Edge detection.....	5
Blob analysis.....	6
Haar Feature based detection	6
Spring	8
Spring Boot.....	8
Microcontroller	9
Methodology	10
The Traffic System	10
The Traffic Client	11
The C++ video analyzer.....	12
The count cars class	13
The intersection classes.....	13
The main_window class.....	14
Server	15
Data model	15
Structure of the rest API.....	16
Traffic control panel	17
Location Tab	18
Server Report Tab	18
Graph Tab	19

Real time traffic control using image processing

Results and conclusion.....	20
Challenges	22
Future Work.....	23
Appendix.....	24
Extras	24
Masker.....	24
Trainer.....	25
References.....	26

Table of Figures

Figure 1: Traffic congestion in addis ababa.....	2
Figure 2: Image differencing, $c = a - b$	5
Figure 3: Edge detection	5
Figure 4: Haar-Like Features	7
Figure 5: Tiva C Series TM4C123G LaunchPad Evaluation Board	9
Figure 6: System Diagram	10
Figure 7: The three different kinds of intersections	11
Figure 8: binary mask.....	12
Figure 9: video analyzer in action.....	12
Figure 10: Simulation of a single four way intersection.....	13
Figure 11: The main window interface	14
Figure 12: Data model.....	15
Figure 13: The location tab	18
Figure 14: pie chart representation of traffic volume at a certain intersection	19
Figure 15: Graph representation of traffic volume at a certain intersection	19
Figure 16: Multiple intersections mid-operation.....	20
Figure 17: model	21
Figure 18: Masker UI.....	24
Figure 19: cascade classifier in training	25

Objective

General Objectives

The main objective of the project was to make a computer vision based large scale traffic control system that makes real-time decision based on the number of cars.

Specific Objectives

The specific objectives of these projects are

- 1) Design and develop a traffic control system
- 2) Study various image processing techniques
- 3) select/use a suitable image processing method for the traffic control system
- 4) Design and develop a system that keeps record about car count information
- 4) Study the architecture of the selected microcontroller
- 5) Test the designed traffic controller

Introduction

Whatever infrastructure we put into place doesn't seem to be satisfying the ever growing traffic density. One reason for this is that both schools and economy sectors operate at the same hours leading to traffic congestion on peak hours.



Figure 1: Traffic congestion in addis ababa

There are systems in place to help traffic flow in rush hours like putting traffic officers on intersections to direct traffic, increasing available public transport, increasing the size of roads... however these methods are exhaustible and the amount of vehicles crowding the streets keeps growing at too much of a rapid rate for these remedies to keep up. A better Way to manage this traffic congestion would be to create a system that allocates the available resources in an effective and efficient manner.

Currently traffic lights are on a timer that is set on previous knowledge of the roads that get more crowded, Problem with this system is that it's not able to account for current circumstances. It simply operates on the presets that were set during its installation meaning a lane could be open with no cars currently in queue while another full of awaiting vehicles waits on the timer. Previously sensors have been used to mend this situation but these sensors are hard to install and

once installed they are difficult to maintain or upgrade. A visual based system is effective in responding to real time situations also it is easy on maintenance and upgrade.

The goals of an Intelligent Transportation System (ITS) are to enhance public safety, reduce congestion, improved travel and transit information, generate cost savings to motor carriers and emergencies operators, reduce detrimental environmental impacts, etc. ITS technologies assist states, cities, and towns nationwide to meet the increasing demands on surface transportation system.

Background

Vehicle detection with Image processing

Vision based approach is replacing the use of conventional sensors in many industrial applications, due to its flexibility. There has been a great deal of works pertaining to image processing and vehicle detection, listed below are the current standards for vehicle detection based on image processing.

Background subtraction

The process of extracting moving foreground objects (input image) from stored background image (static image) or generated background frame from image series (video) is called background subtraction, after that, the extracted information (moving objects) is resulted as the threshold of image differencing. This method is one of widely used change detection methods in vehicle detection. The fact that this method is non-adaptive is a drawback which is raised due to the changing in the lighting and the climate situations. Still good results have been achieved by dynamically extracting the background image from the video sequence in a given time interval.

Frame Differencing

The frame differencing is the process of subtracting two subsequent frames in image series to segment the foreground object (moving object) from the background frame image. Also, the motion segmentation process is another fundamental step in detecting vehicle in image series which is done by isolating the moving objects (blobs) through analyzed and assignment sets of pixels to different classes of objects which based on orientations and speed of their movements from the background of the motion scene image sequence.

Experimental tests show an effective performance and sufficient accuracy for general vehicle type classification within the approach. It works by setting an upper and lower limit and counting the number of vehicles currently In view.

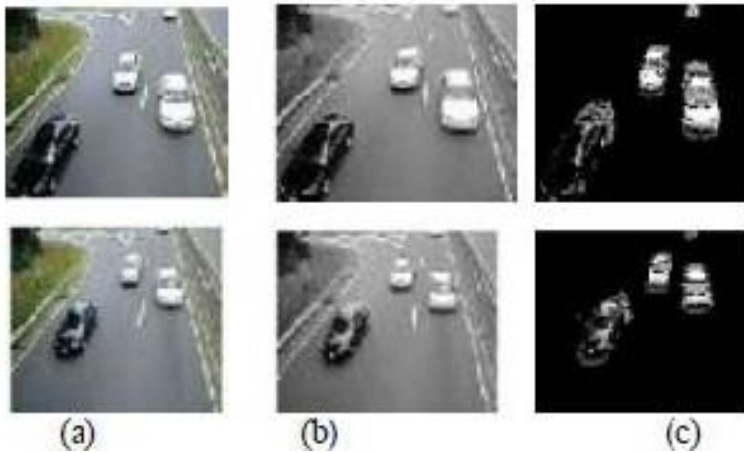


Figure 2: Image differencing, $c = a - b$

Edge detection

Edge detection is a process of locating an edge of an image. Detection of edges in an image is a very important step towards understanding image features. Edges consist of meaningful features and contain significant information. It significantly reduces the image size and filters out information that may be regarded as less relevant, thus preserving the important structural properties of an image.

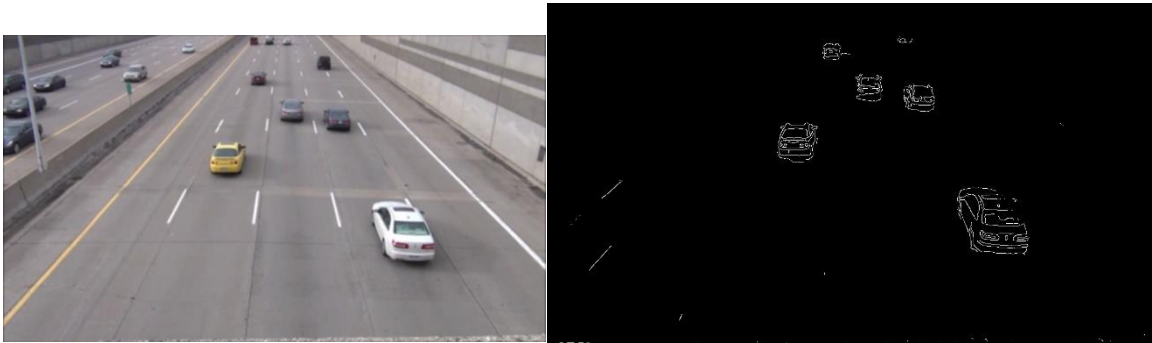


Figure 3: Edge detection

Steps in edge detection

Filtering: is commonly used to improve the performance of an edge detector with respect to noise.

Enhancement: In order to facilitate the detection of edges, it is essential to determine changes in intensity in the neighborhood of a point. Enhancement emphasizes pixels where there is a significant change in local intensity values and is usually performed by computing the gradient magnitude.

Detection: In order to isolate points with strong edge content some method should be used to

determine which criterion is used for detection. Multiple edge detection algorithms have been developed, off all the detection algorithms canny edge detection is the better choice in terms of producing the most usable result.

Blob analysis

After running frame differencing, background subtraction or edge detection the customary next step is Blob analysis. Blob stands for Binary Large Object and refers to a group of connected pixels in a binary image. It is a fundamental technique of machine vision based on analysis of consistent image regions. As such it is a tool of choice for applications in which the object being inspected are clearly discernible from the background.

There are several motivations for studying and developing blob detectors. Blob detection can be used to obtain region of interest for further processing. These regions could signal the presence of objects or parts of objects in the image domain with application to object recognition or object tracking. In other domains such as histogram analysis, blob descriptors can also be used for peak detection with application to segmentation. Another common use of blob descriptors is a main primitive for texture analysis.

Haar Feature based detection

First, a classifier (namely a *cascade of boosted classifiers working with haar-like features*) is trained with a few hundred sample views of a particular object (in this case cars), called positive examples, that are scaled to the same size (say, 48x48), and negative examples - arbitrary images of the same size.

After a classifier is trained, it can be applied to a region of interest (of the same size as used during the training) in an input image. The classifier outputs a “1” if the region is likely to show the object (i.e., face/car), and “0” otherwise. To search for the object in the whole image one can move the search window across the image and check every location using the classifier. The classifier is designed so that it can be easily “resized” in order to be able to find the objects of interest at different sizes, which is more efficient than resizing the image itself. So, to find an object of an unknown size in the image the scan procedure should be done several times at different scales.

The word “cascade” in the classifier name means that the resultant classifier consists of several simpler classifiers (*stages*) that are applied subsequently to a region of interest until at some stage the candidate is rejected or all the stages are passed. The word “boosted” means that the classifiers at every stage of the cascade are complex themselves and they are built out of basic classifiers using one of four different boosting techniques (weighted voting). The basic classifiers are decision-tree classifiers with at least 2 leaves. Haar-like features are the input to the basic classifiers, and are calculated as described below. The current algorithm uses the following Haar-like features:

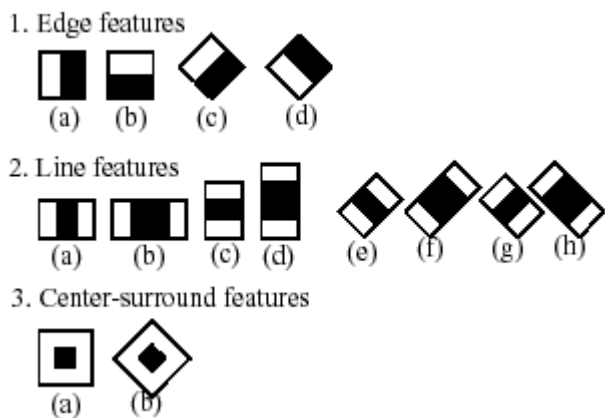


Figure 4: Haar-Like Features

The feature used in a particular classifier is specified by its shape (1a, 2b etc.), position within the region of interest and the scale (this scale is not the same as the scale used at the detection stage, though these two scales are multiplied). For example, in the case of the third line feature (2c) the response is calculated as the difference between the sum of image pixels under the rectangle covering the whole feature (including the two white stripes and the black stripe in the middle) and the sum of the image pixels under the black stripe multiplied by 3 in order to compensate for the differences in the size of areas.

Spring

It is a huge framework for the java platform initially it was mostly known as a dependency injection framework which later it transcended into becoming the most popular web framework for the java ecosystem. Spring framework includes several modules that provide a wide range of use cases. Some of them are

- **Spring Security** Authentication & authorization
- **Spring Data** used for Data access/storage for relational database

Spring security: is a powerful and highly customizable authentication and access control framework. it is the standard way for securing spring-based applications. We use spring security for securing our rest api end points.

Spring data jpa: it makes it easy for creating data-access across our data model the advantage of spring data jpa is it makes it easy to create query by having declarative approach which will reduce a lot of boilerplate query code. we use spring data jpa for building queries.

Spring Boot

Spring Boot makes it easy to create stand-alone, production-grade web application. spring boot takes an opinionated view of the spring platform and third party libraries or in other words it takes little configuration. spring boot Provide a wide range of non-functional features that are common to large classes of projects (e.g. embedded servers, security, metrics, health checks, externalized configuration).

Microcontroller

Texas Instruments is an American technology company that designs and manufactures semiconductors which it sells to electronics designers and manufacturers globally. The products have increased power efficiency, higher precision, more mobility and better quality.

The Microcontroller used in this project is the TM4C123GH6PM. This microcontroller is a 32-bit ARM Cortex-M4-based microcontroller with 256-kB Flash memory, 32-kB SRAM, and 80-MHz operation USB 2.0 host device. A Hibernation module and PWM and a wide range of other peripherals.

The Ide used for programming and debugging the microcontroller is the keil arm microcontroller development kit.

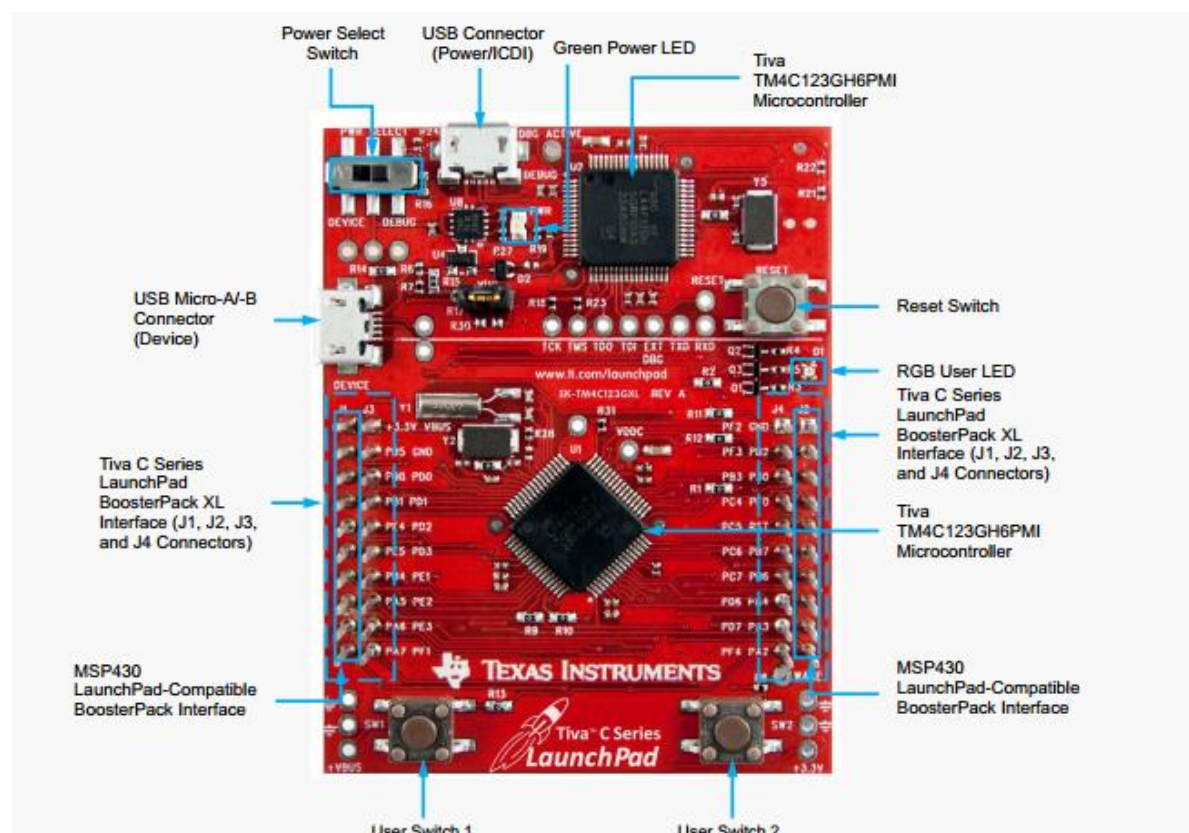


Figure 5: Tiva C Series TM4C123G LaunchPad Evaluation Board

Methodology

The system mainly consists of two levels of control, the Traffic control panel and the traffic client interface. The traffic client interface is responsible for the decision making and communication with the microcontroller. It is also responsible for updating the server at a regular interval. The traffic control panel communicates with the server to allow the administrator to view the car count for each location and retrieves the data in a manner that makes for easy analysis of data using various types of graphs and charts.

The Traffic System

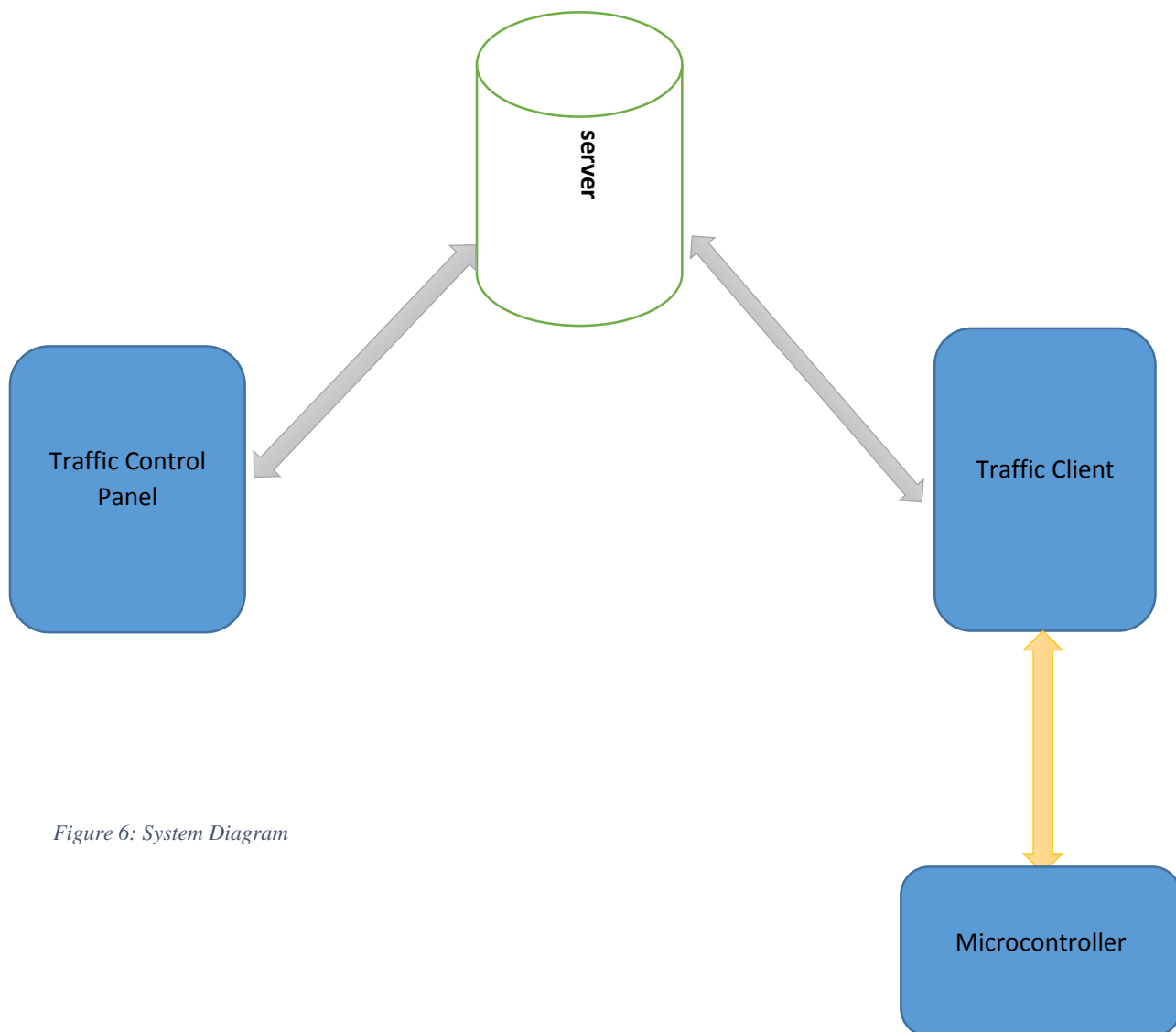


Figure 6: System Diagram

The Traffic Client

The traffic client interface responsible for video analysis and decision making is a multiplatform system in its own right built using C++ and java. C++ was used for processing of the video to extract usable data on account of its superior computational capabilities. However, we found that java was better suited when it came to building the user interface, multithreading and communicating with the server. Below is a concise description of how the system works followed by a brief description of the main classes involved.

The system considers a traffic intersection as a collection of video inputs from the various directions accompanied by information about the direction of these video inputs. It then analyzes this data to extract usable information (current number of cars in view). Once it has the necessary data it moves on to deciding which lane to open next and for how long. It is designed to handle multiple intersections at the same time.

One of the major design considerations throughout the project was scalability. The system is designed in such a way that the video analysis is separate from the user interface and control logic. Making it easily swappable if better algorithms for image processing are encountered and it also allowed us to implement different types of intersections regardless of their processing needs.

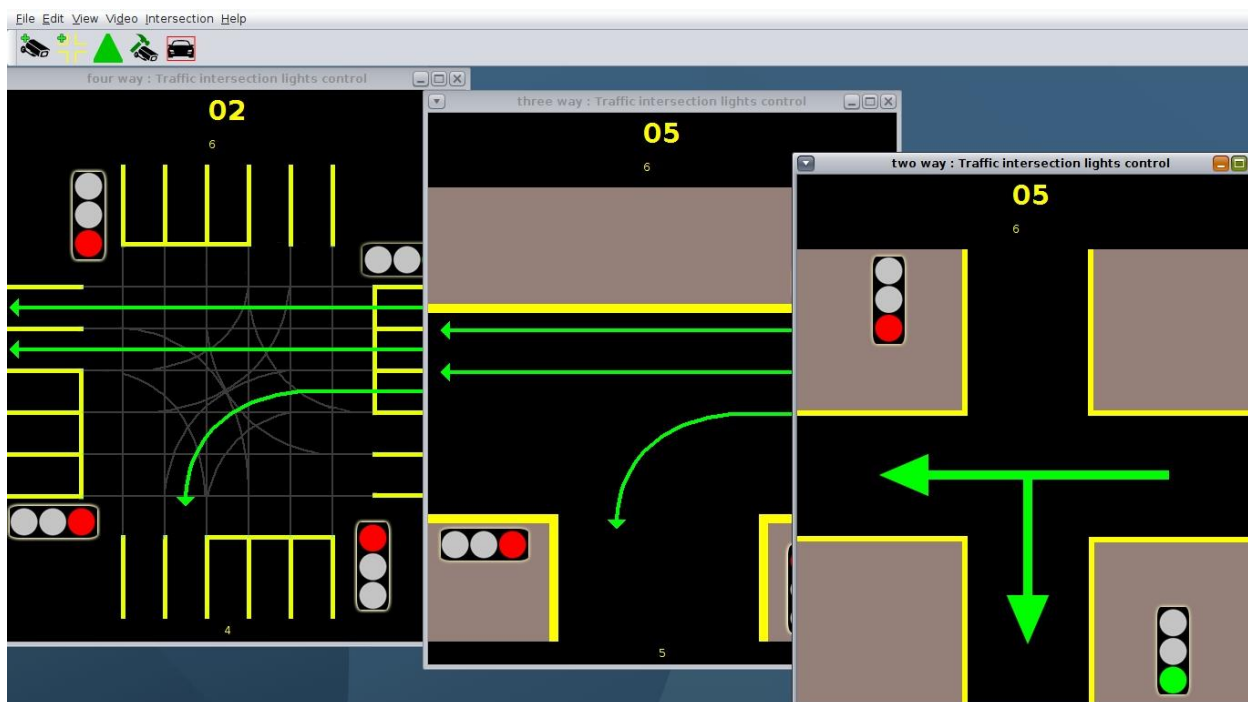


Figure 7: The three different kinds of intersections

The C++ video analyzer

The video analyzer code is written using C++ and openCV it takes as an argument the video title (location), the xml file to use and the location of a mask description file, if one exists. It opens the video at the provided location and runs a search for cars on every frame using a haar cascade xml file (which is an xml file with a list of features to scan for). The process of filtering out the cars from the background is explained in the background.

Once it processes a frame it outputs the number of cars in that frame. If a mask is present it will first create a binary mask which it will later overlay over every frame before processing in order to extract the region of interest. Displaying the video while also processing it can be an expensive process, therefore the program will display the video drawing rectangles over the detected cars only if debug mode is enabled.

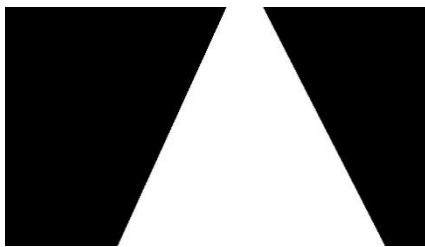


Figure 8: binary mask

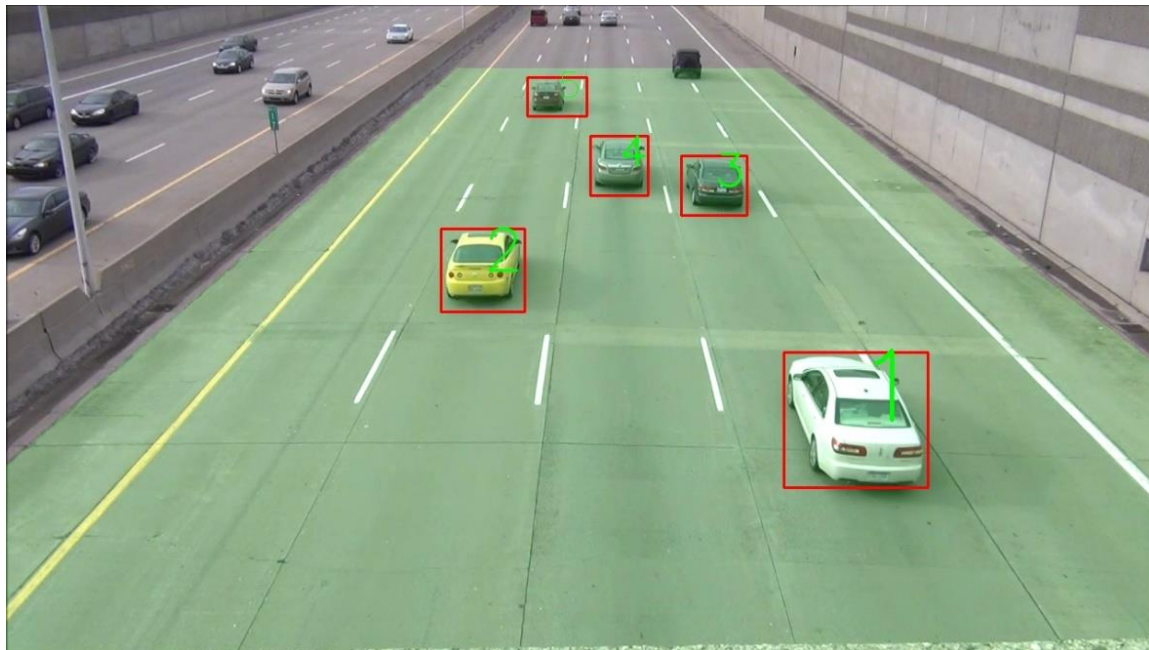


Figure 9: video analyzer in action

The count cars class

The count cars class is a sort of java envelop of the C++ video analyzer. Upon execution it starts a new thread to run the C++ process. Once it starts the process it instantiates a scanner variable to listen for numeric output from the video being scanned. it saves this output of the process to an accessible local field.

The intersection classes

An abstract 'Intersection' class is implemented from which multiple intersection classes extend the intersection could be a four way intersection, a three way intersection or two one way intersections. All intersections have common attributes like a list of videos, the currently open lane and a pool of threads running *count cars* processes.

When an intersection is instantiated it creates a pool of threads running a *count cars* operation for each lane. When this intersection object is instructed to be online it starts these threads and starts queuing of the lanes based on input from these threads.

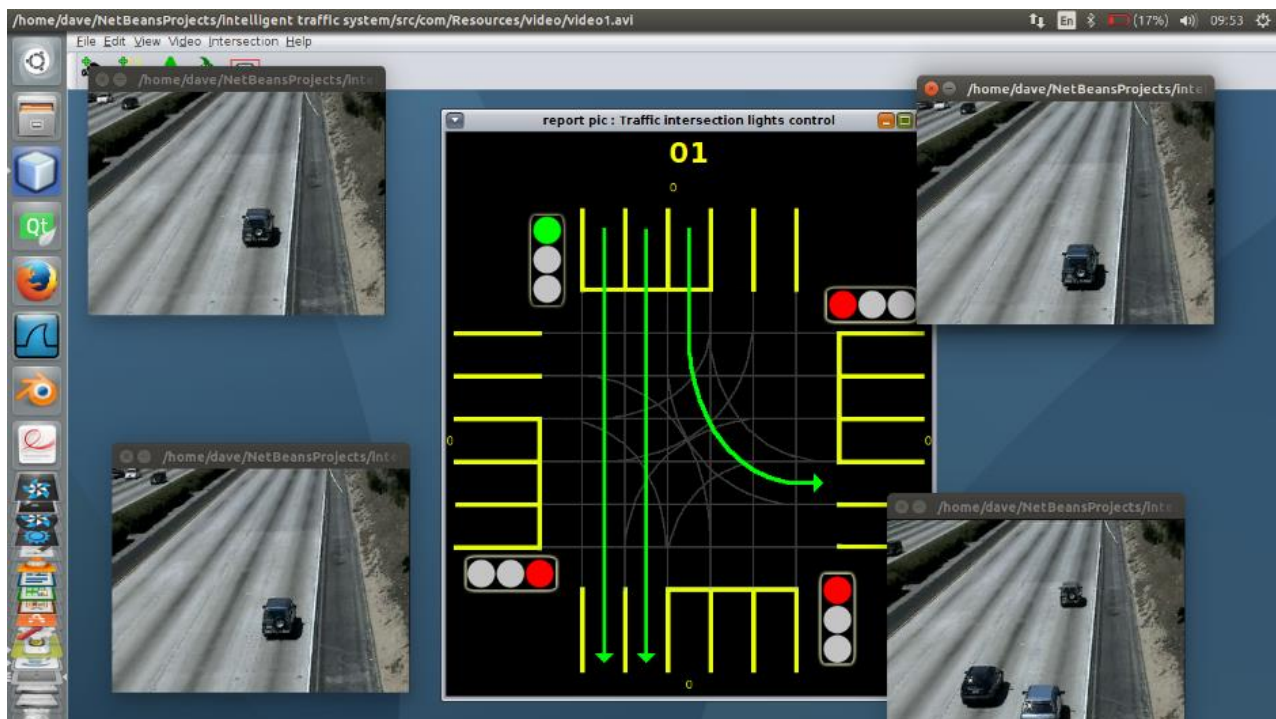


Figure 10: Simulation of a single four way intersection

The main_window class

This is the class that ties all the operations into a neat user friendly interface. It makes use of javas' Multiple Document Interface API to create an interface where the user can add cameras and intersections. The main window class provides an interface to select region of interest. This masker interface takes as input and provides a UI for the user to specify the region of interest in the image.

The main window also houses the cascade trainer which is a user friendly interface that presents various unsorted images to be processed for training a cascade classifier. It presents these images for selection and can be used to train a cascade to detect virtually anything (given the right input dataset). It also has options to allow viewing and debugging of videos and intersection operations.

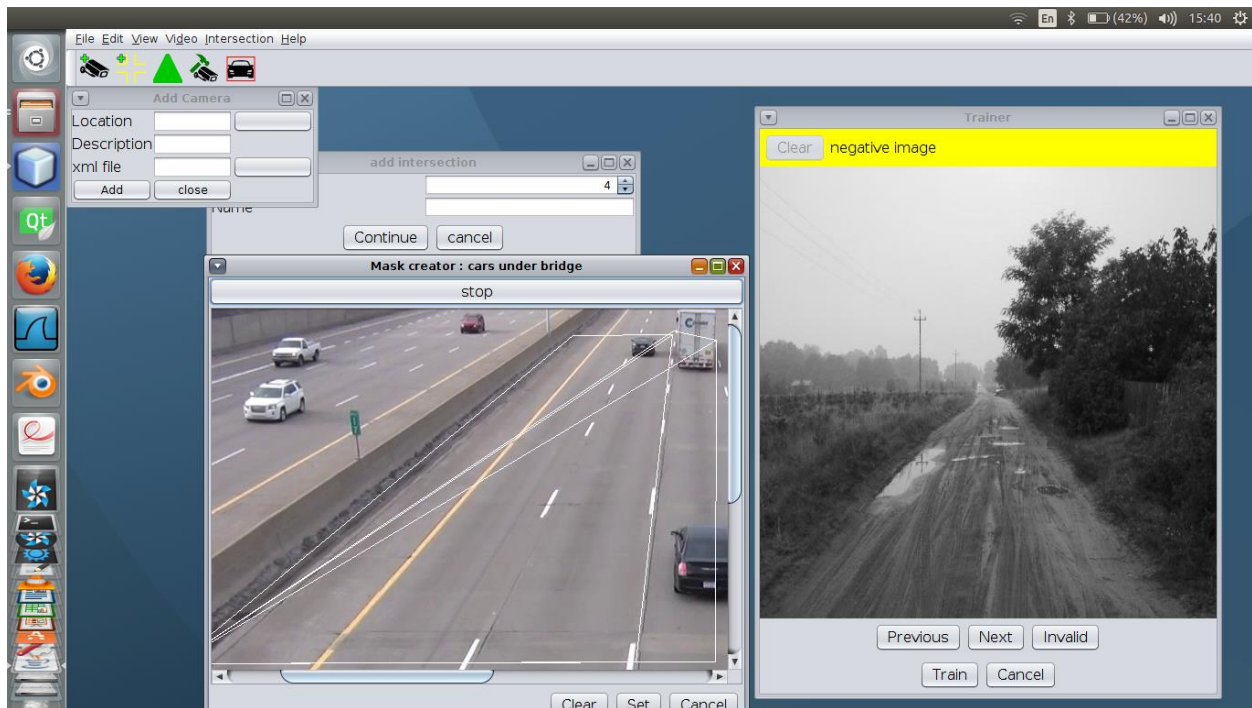


Figure 11: The main window interface

Server

The server or the backend system is written using spring boot.it consists of a rest api that uses basic auth for authorization, The backend system is used for storing traffic information. Once the information is collected it is gathered and stored in a persistence storage system (in our case a relational database). The collected information is car count from each lane of every intersection location the system is deployed on.

Data model

The data model represents the data that will be persisted on the database. The structure of the data is shown below.

1. **Location** represents a street or a main intersection road that consists of four lanes
2. **Lane** subcomponent of a location each location can have up to four lanes
3. **Log** consists of the log(record) information for each lane at a particular time
4. **Admin** holds information that will be used for authentication

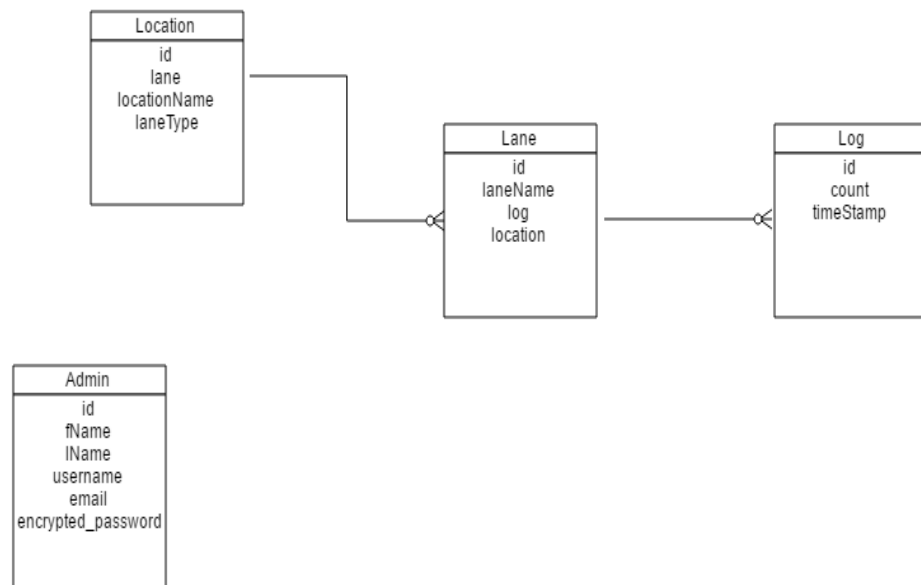


Figure 12: Data model

Real time traffic control using image processing

Structure of the rest API

As mentioned above the rest API uses basic auth for authorization, also there are various security methods for secure API basic auth is one of the easiest to implement for securing a rest API.

url	Http Method	Function
api/traffik/location/all	GET	Returns list of location
api/traffic/location/{location_id}	GET	Returns a single location which have the parameter matching the location id
api/traffic/location/{location_id}	PUT	Changes/edits the location
api/traffic/location/add	POST	Creates a new location
api/traffic/location/{location_id}	DELETE	Deletes a new location
api/traffic/user/create	POST	Creates a new user(Admin)
api/traffic/user/all	GET	Returns a list of users/admin
api/traffic/user/{user_id}	PUT	Changes/edit the user(Admin) info
api/traffic/user/{user_id}	DELETE	Removes the user(Admin) info
api/traffic/user/{user_id}	GET	Returns the user(Admin) whose id is equal to the user_id
api/traffic/log/{location_id}/carcount/{a}-{b}-{c}-{d}	POST	Creates a log/record of car count for particular location.where parameters (a,b,c,d) are the car count for lane a,...d respectively

api/traffic/log/{location_id}/carcount/{timefilter}	GET	Returns the list of car count a specific period of time
api/traffic/log/{location_id}/carcount/from/{start}-{end}	GET	Returns the list of car count from the specified period of time(i.e start and end)

Traffic control panel

The traffic control panel is a desktop application that is used to

- View the car count for each location using various types of graphs
- View the server health metrics/reports
- Filter the car count for each location based on certain criteria's
- Perform create/read/edit operation for each location

The application consists of four main tabs

1. Location Tab
2. Server Report Tab
3. Graph Tab
4. Export Tab

Location Tab

The location tab is used for performing crud operations i.e. (Create, Read, Edit, Delete) on each location and lane.

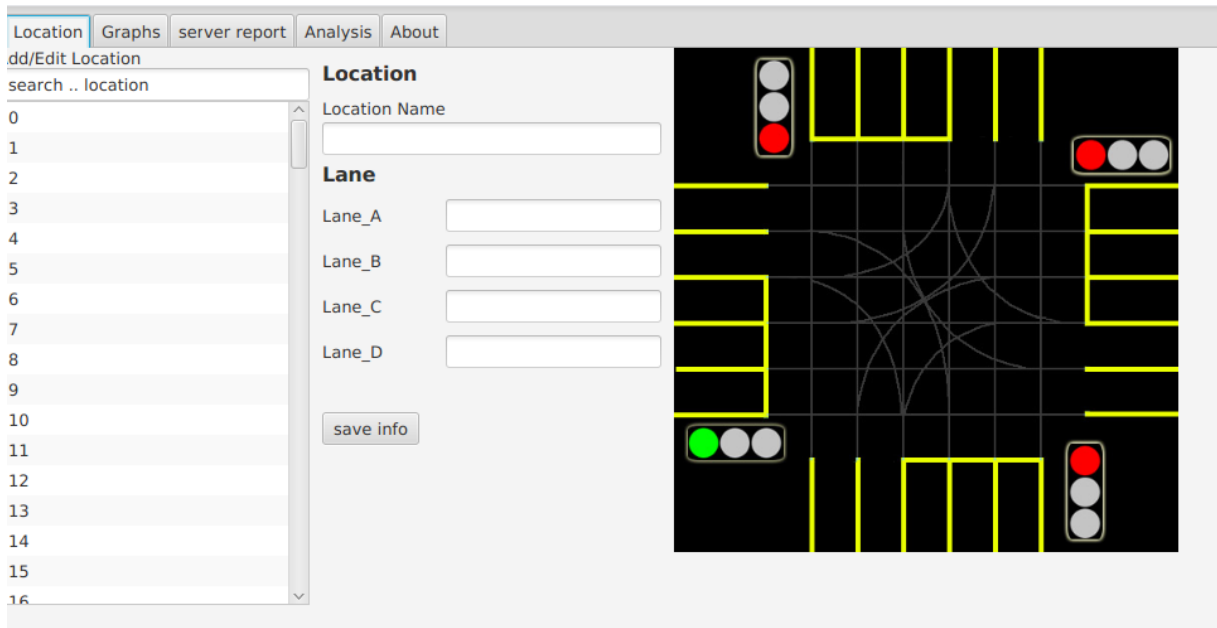


Figure 13: The location tab

Server Report Tab

The server tab is used to show application health information like whether the application is secure, a simple 'status' when accessed over an unauthenticated connection or full message details when authenticated.

Graph Tab

The graph tab is used to show the data's in different graph visualizations like pie chart/line chart. The figures below show different types of graphs for a certain location.

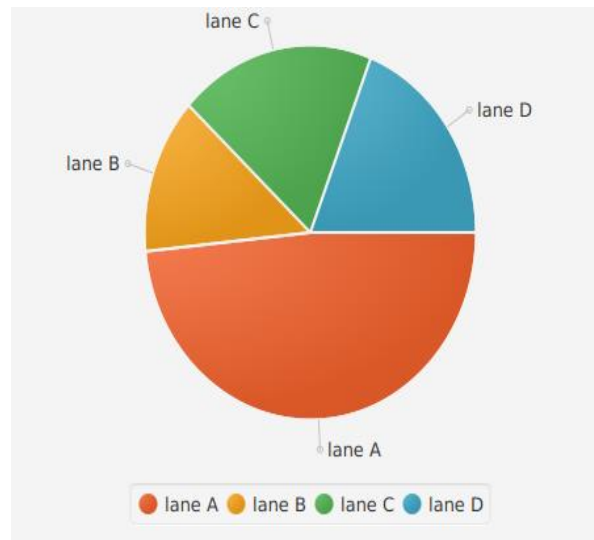


Figure 14: pie chart representation of traffic volume at a certain intersection

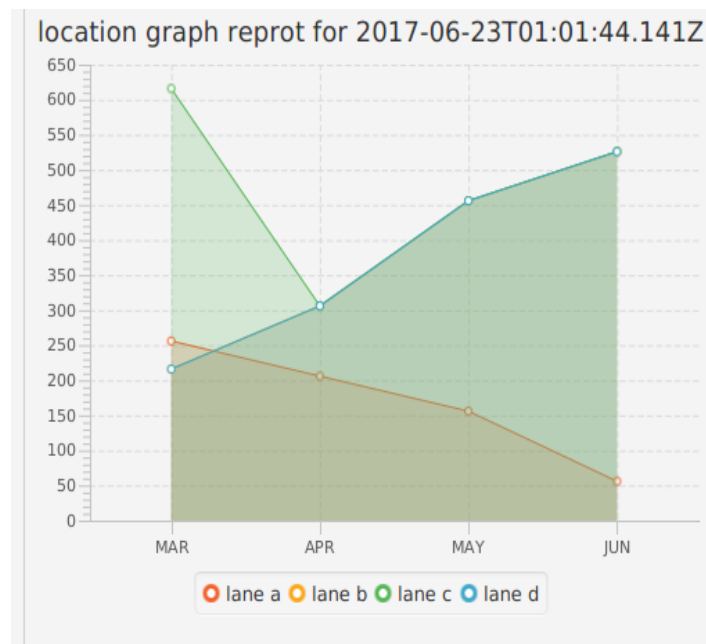


Figure 15: Graph representation of traffic volume at a certain intersection

Results and conclusion

The software simulations were run on an hp core i5 laptop with 4GB of RAM

In Linux operating environment (Ubuntu 14.04 Trusty Tahr) we have found the system to be at optimal performance handling up to 12 intersections simultaneously. Every intersection has the decision making thread and threads for analysis of video inputs from the various lanes. The tests showed that the system operates optimally, running a total of 60 taxing threads concurrently on an average laptop computer. On windows OS processes start slowing down at 8 intersections.

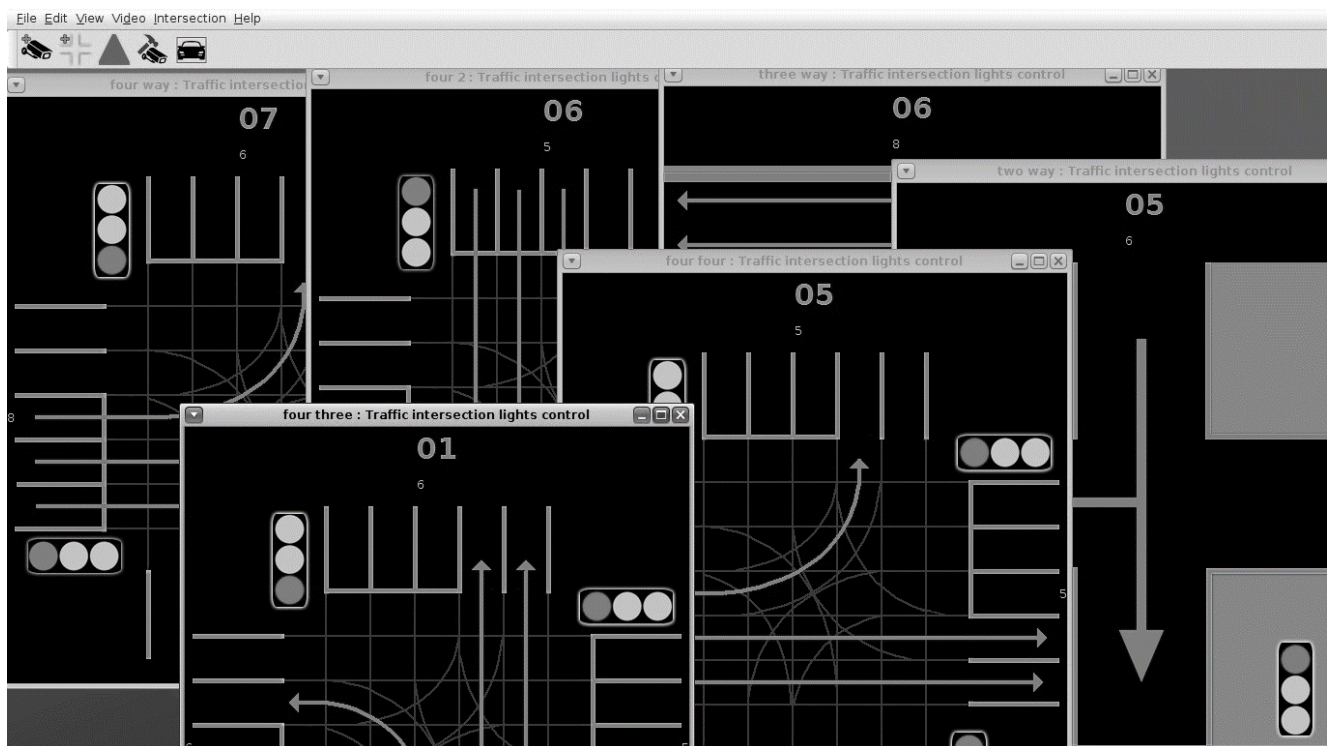


Figure 16: Multiple intersections mid-operation

However in debugging mode (displaying all the videos after analysis and editing) the number of intersections that can run maintaining good overall performance decreases by half. Also it is worth noting that the above results were with the video detection algorithm at >85% accuracy. Unfortunately increasing the accuracy also means increasing the work load and the drain on the systems resources but we have found 90% accuracy works well for the detection with acceptable drain on the system.

Real time traffic control using image processing

We built a small street model using cardboard and a bunch of LEDs to test the functionality of the system. We implemented the code using TM4C123gx1 evaluation kit described in the background to construct a model for the two one way street type intersection. We communicated with the microcontroller using its UART interface.

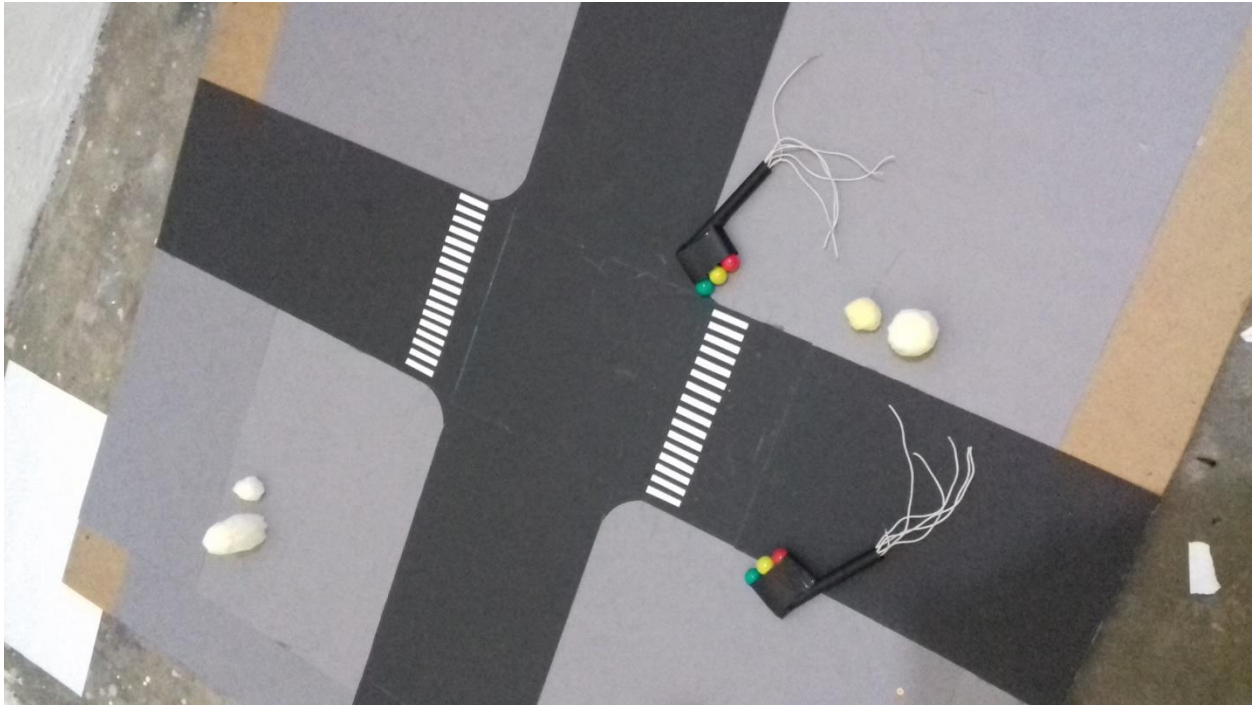


Figure 17: model

We have found that the system functions as expected.

Challenges

Initially the traffic client desktop application was developed for the Linux operating system. In order to integrate the desktop application to the microcontroller and debug it, the application had to be ported to windows OS this seemingly simple process was made complicated by the fact that the application involved code written both in java and C++.

Making the haar cascade trained classifiers was also challenging. This forced us to make a user interface to handle the preparation of unsorted training images.

Future Work

Currently the traffic system uses a greedy algorithm designed to achieve optimum solution by finding localized optimum solutions which may eventually land in globally optimized solutions. However machine learning could be applied to analyze consecutive intersections and further improve the systems decision making.

Appendix

Extras

Listed below are parts of the system not extensively described in the Methodology.

Masker

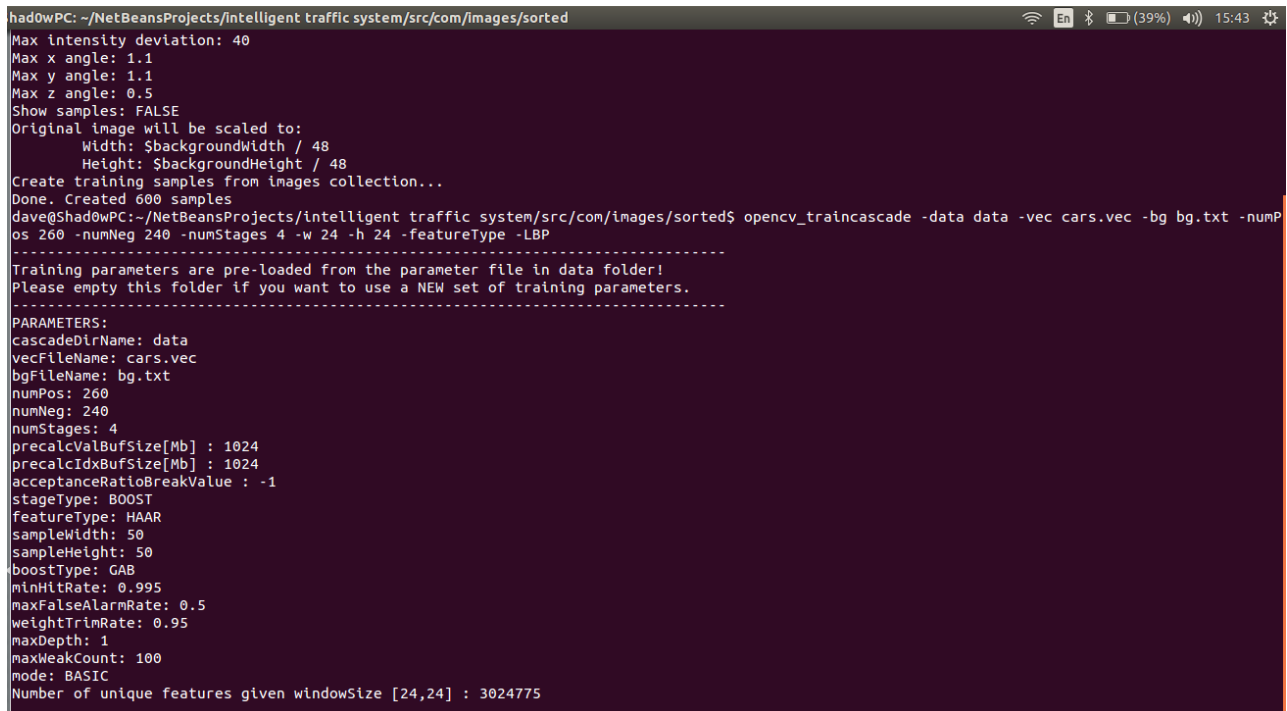
The masker class is a class that provides an interface to select region of interest. It takes as input the video to be masked after that it takes a snap shot from a given video source and presents a UI for the user to specify the region of interest in the image this feature seems especially important to places like Ethiopia where the roads are more often than not, two way streets



Figure 18: Masker UI

Trainer

The trainer class is a custom made haar cascade classifier trainer user interface which can be used to train a haar cascade to practically detect anything it provides a user friendly interface where one can train a cascade by pointing out the region of interest in a set of training images.



```
had0wPC: ~/NetBeansProjects/intelligent traffic system/src/com/Images/sorted
Max intensity deviation: 40
Max x angle: 1.1
Max y angle: 1.1
Max z angle: 0.5
Show samples: FALSE
Original image will be scaled to:
    Width: $backgroundWidth / 48
    Height: $backgroundHeight / 48
Create training samples from images collection...
Done. Created 600 samples
dave@ShadowPC:~/NetBeansProjects/intelligent traffic system/src/com/Images/sorted$ opencv_traincascade -data data -vec cars.vec -bg bg.txt -numP
os 260 -numNeg 240 -numStages 4 -w 24 -h 24 -featureType -LBP
-----
Training parameters are pre-loaded from the parameter file in data folder!
Please empty this folder if you want to use a NEW set of training parameters.
-----
PARAMETERS:
cascadeDirName: data
vecFileName: cars.vec
bgFileName: bg.txt
numPos: 260
numNeg: 240
numStages: 4
precalcValBufSize[Mb] : 1024
precalcIdxBufSize[Mb] : 1024
acceptanceRatioBreakValue : -1
stageType: BOOST
featureType: HAAR
sampleWidth: 50
sampleHeight: 50
boostType: GAB
minHitRate: 0.995
maxFalseAlarmRate: 0.5
weightTrimRate: 0.95
maxDepth: 1
maxWeakCount: 100
mode: BASIC
Number of unique features given windowSize [24,24] : 3024775
```

Figure 19: cascade classifier in training

Microcontroller

The code for the model is set up in such a way that the microcontroller constantly scans for input which will be supplied by our application once it gets this input. It will control the 6 LEDs attached accordingly. The logic for the model is set up such that if there are no cars then the green light will alternate every minute if there are more cars in the closed direction than the open direction it will give the cars in the open a small time window to pass along and open the other side accordingly.

References

- [1] Video Based Vehicle Detection and Its Application in Intelligent Transportation Systems
Naveen Chintalacheruvu, Venkatesan Muthukumar
- [2] Embedded image processing with DSP examples in matlab
Shehrzad Qureshi
- [3] OpenCV official documentation
- [4] Moving Vehicle Detection for Measuring Traffic Count Using OpenCV
Nilesh J. Uke
- [5] M. Piccardi, "Background subtraction techniques: a review", IEEE
International Conference on Systems, Man and Cybernetics 2004.
- [6] Learning Opencv
by **Gary** Bradski and Adrian Kaehler
- [7] R. Cucchiara, M. Piccardi, and P. Mello, "Image analysis and rule based reasoning for a traffic monitoring system," IEEE Trans. Intell. Transport. Syst., vol. 1, no. 2, pp. 119-130, June 2002.
- [8] A. Ilyas, M. Scuturici, and S. Miguët, "Real time foreground background segmentation using a modified Codebook model," in Proc. 6th IEEE Int. Conf. AVSS, 2009, pp. 454-459.
- [9] VEHICLE DETECTION AND TRACKING TECHNIQUES: A CONCISE REVIEW by
Raad Ahmed Hadi, Ghazali Sulong and Loay Edwar
- [10] ASSESSING VEHICLE DETECTION UTILIZING VIDEO IMAGE PROCESSING
TECHNOLOGY By Duane Hartmann, Dan Middleton and Dwayne Morris, Texas Transportation
Institute
- [11] Sumitomo Electric Intelligent Traffic System
- [12] **Spring in action 4th edition by craig walls**
- [13] **Spring Boot Reference Guide 1.4.2.RELEASE** by Phillip Webb , Dave Syer , Josh Long ,
Stéphane Nicoll , Rob Winch , Andy Wilkinson , Marcel Overdijk , Christian Dupuis , Sébastien
Deleuze