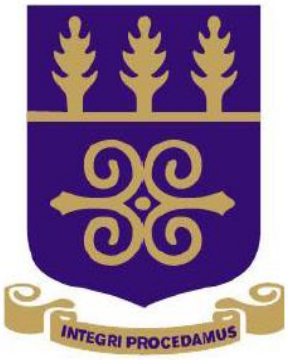

UNIVERSITY OF GHANA

DEPARTMENT OF COMPUTER ENGINEERING

SCHOOL OF ENGINEERING SCIENCES

COLLEGE OF BASIC AND APPLIED SCIENCES

FINAL YEAR PROJECT REPORT

ON

AUTOMATIC TRAFFIC SCHEDULING AND ALERTING SYSTEM USING COMPUTER VISION

PROJECT REPORT SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE BACHELOR OF SCIENCE DEGREE IN ENGINEERING

NAMES OF STUDENTS

BINEY AMOAH KINGSLEY	10515844
AGUDOGO A. JERRY	10535255
ADZORGENU WISDOM	10528837
APPIAH DEREK PRINCE A.	10515761

NAME OF SUPERVISOR:	Dr. ROBERT SOWAH
DATE OF SUBMISSION:	MM DD, YYYY

AUTOMATIC TRAFFIC SCHEDULING AND ALERTING SYSTEM USING COMPUTER VISION

BY

BINEY AMOAH KINGSLEY	10515844
AGUDOGO A. JERRY	10535255
ADZORGENU WISDOM	10515XXX
APPIAH A. DEREK PRINCE	10515XXX

Submitted to the Department of Computer Engineering in Partial Agreement of the fulfillment of the Requirements for the Degree of Bachelor of Science in Computer Engineering.

University of Ghana

MM DD, YYYY

Name of Student	**Signature**
BINEY AMOAH KINGSLEY	_____
AGUDOGO A. JERRY	_____
ADZORGENU WISDOM	_____
APPIAH A. DEREK	_____

Name of Supervisor: _____

Signature of Supervisor: _____

Name of Head of Department: _____

Signature of Head of Department: _____

Declaration of Originality Form

Department of Computer Engineering

We certify that this thesis is our own work except where indicated by referencing. We confirm that we have read and understood the guidelines on plagiarism in the Computer Engineering Undergraduate Thesis Handbook including the University of Ghana's policy on plagiarism. We have acknowledged in the text of this document all sources used. We have also referenced all relevant texts, figures, data and tables quoted from books, journals, articles, reports, Internet websites, and works of other people. We certify that we have not used the services of any professional person or agency to produce this thesis document. We have also not presented the work of any student present or past from other academic institutions. We understand that any false claim in respect of this thesis document will result in disciplinary action in accordance with regulations of the University of Ghana.

NAME OF STUDENT	INDEX NUMBER	SIGNATURE
DATE:		

THESIS TITLE	---------- ---------- ----------
CERTIFIED BY SUPERVISOR:	
SUPERVISOR SIGNATURE:	DATE:

AUTOMATIC TRAFFIC SCHEDULING AND ALERTING SYSTEM USING COMPUTER VISION

ABSTRACT

Effective and efficient traffic control is one system every major city in the world craves for and in Ghana, there is no exception. Most people find themselves stuck in heavy traffics during their day-to-day activities due to poor traffic management and as such efforts are being made to develop smart systems which can autonomously control the flow of vehicle and human traffic in cities. Several mechanisms have been implemented—an example being the use of sensors amongst others, but these systems lack the capability to offer full traffic control and surveillance functionalities. Thus, Tertia Oculus, which is the name of this project seeks to offer complete traffic management and surveillance system ranging from traffic management to anomaly detection using Tensorflow's open source Object Detection Application Programming Interface (API) capabilities, backed by mobile surveillance using the services of a drone. Quick accident detection will reduce the time it takes for authorities to be alerted whenever roads accidents occur, and this will in turn reduce response time. This project also features an android mobile application which keeps users updated on the traffic situation and any emergency occurrences (such as accidents) that might happen on the path they are commuting on. The object detection aspect of this project performs two basic tasks: vehicle and pedestrian detection and counting as well as anomaly (roads in poor conditions and road accident occurrences) detection. These information is then passed on to the traffic modelling and simulation system of the project. Anomalies detected in the video streams being sent to the object detected system from various camera (attached to raspberry pi) points via HTTP streams are made available to the android application via APIs. In addition to these functionalities, the drone provides mobile surveillance by autonomously patrolling around a designated geographical area and providing video streams to the object detection system for anomaly checks. Any anomaly (for instance a road accident) detected in the video stream is captured as an image and sent to the android application together with the timestamp and Global Positioning System (GPS) coordinates to be laid to the appropriate authorities. In this project, the object detection and camera subsystem were successfully integrated with video streaming, start and stop, and GPS coordinates sent to the detection subsystem. A working quadrotor was built and flown as well as integrated with streaming vehicle state to the ground control app. Autonomous actions and commands from the ground control app and on-board computer were successfully executed in a simulated environment. The object detection API was able to classify vehicles with accuracies mostly between 90-98% and pedestrians with an accuracy of 95-98%. Results of detections were streamed in real time to both the simulation subsystem and incidence reporting subsystem. Results sent from the object detection API were successfully imported into the simulator with dynamics of the simulated vehicles observed in a GUI interface. The incidence reporting subsystem successfully send out notifications of incidents detected by the object detection API on twitter and via email.

ACKNOWLEDGEMENT

We will first of all like to give thanks to the Almighty God for seeing us through the entire project. We will also like to thank **Dr. Robert A. Sowah**, under whose supervision we were able to achieve the aims of the project and to develop the accompanying thesis report. His technical as well as moral support proved invaluable in the conception of this idea and its actualization. We will also like to duly acknowledge the inspiring and technical lessons from our lecturer from whom we gained the skill to achieve our success in the project. **Dr. Wiafe Owusu-Banahene** and **Mr. Prosper Azaglo** whose constructive advice proved helpful in the course of development of the various systems of this project. We will also like to thank our teaching assistants for all the training and experience they passed on and for all the support given throughout the year. We will finally like to thank our respective parents for their immeasurable support, without which this project will not have seen its beginning or end.

Table of Contents

LIST OF FIGURES

CHAPTER 1—INTRODUCTION

1.1 BACKGROUND

Most Traffic Signal Lights we see within this country and in other African countries operates based on regular timers. [1] With this, what it means is that each particular portion or lane of the is allocated a specific amount of time during which vehicles are made to move and after the time elapse, the Traffic Signal Lights changes to red and the vehicles come to halt, allowing other lanes to take turns. This has been the normal system in most countries (even in some advanced countries) for a very long time.

The down side of this system however is that, in reality, there is always an imbalanced movement of vehicles especially in cities at various times of the day. For instance, the number of vehicles which will move from residential suburbs to industrial areas in the morning is naturally expected to be higher as compared to the number of vehicles which will be expected to move from industrial and office areas to residential parts of the city. Similarly, in the evening, the number of vehicles which will move from industrial and office areas back to residential parts of the city will be higher than the number of vehicles moving from residential areas to industrial areas.

This quick statistic shows that, given an intersection with road networks to industrial areas and residential areas and the Traffic Light Signals are given the same amount of time schedule, the control of the vehicular traffic will be inefficient since with time, one part of the road will be choked while the other get barely empty eventually leading to Police personnel being called upon to manually help the traffic.

Most of the traffic jams caused on the Ghanaian roads, especially in the city of Accra is due to the Traffic Signal Light (TSL) selection with improper and less detailed investigation of vehicle flow. Most of the traffic lights signals are controlled based on timers thereby accounting for the disproportionate amount of time that is allocated to each traffic without taking any other factor(s) such as the number of cars on the road at each point in time as well as the density of cars at each intersection. This therefore tends to create congestion at certain junctures at certain times of the day while other junctures become very free. This imbalance of vehicle traffic flow is what tends to cause a lot of traffic.

The current mechanism used for the scheduling operations of most if not all Traffic Signal Light (TSL) in the country is not the very best and lags behind in the ranks of optimized TSL scheduling mechanisms in the sense that most of the traffic light available operates based on timers. This means that the schedule the movement and stoppage of cars using a certain fixed amount of time and this methodology creates a lot of inconveniences in our cities most of the time. Human interventions are therefore sought after all the time in order for sanity to get restored. The fact that traffic lights rely on timers to operate makes them basically inappropriate when the traffic becomes very biased towards one side or direction at a particular time in the day because the timers become inaccurate. The causes and effects of traffic congestion on the activities of private commercial drivers (trotro) were it reduces the number of trips they could make in a day, increase fuel use and reduce daily sales. To passengers, it delays their time, leading to low productivity, and is associated with other health effects, among many others.[1]

Another major problem on our roads yet to be curbed is road accidents and sadly enough, the time it takes to report one to authorized personnel should it even occur and finally the state of

some major roads right in the city. There are certain major roads linking towns together but due to their deplorable state, they have become a no-go zone making motorists divert course causing pressure and another traffic to mount on other roads.

With this critical problem gradually becoming a burden to the society and the country as a whole, a solution was proposed—To apply the powerful abilities of computer vision, coupled with machine learning to be able automatically detect when where there are or there are no cars and with the aid of a traffic scheduling algorithm, control the Traffic Signal Light accordingly.

The bigger picture of this project seeks to develop the capability to schedule road traffic using computer vision by reporting the number of vehicles detected on the road at any particular time to a central database where the scheduling system of the project will use this information to direct traffic by manipulating the traffic light signal. Another aspect of this project seeks to help update motorists ahead of time should the computer vision system of the project detect road anomalies like accidents or bad roads and also help redirect their path to a more convenient route. The fourth section of this project is geared at using the power of drones to perform mobile road surveillance and also rely on the computer vision system to perform critical operations during its surveillance.

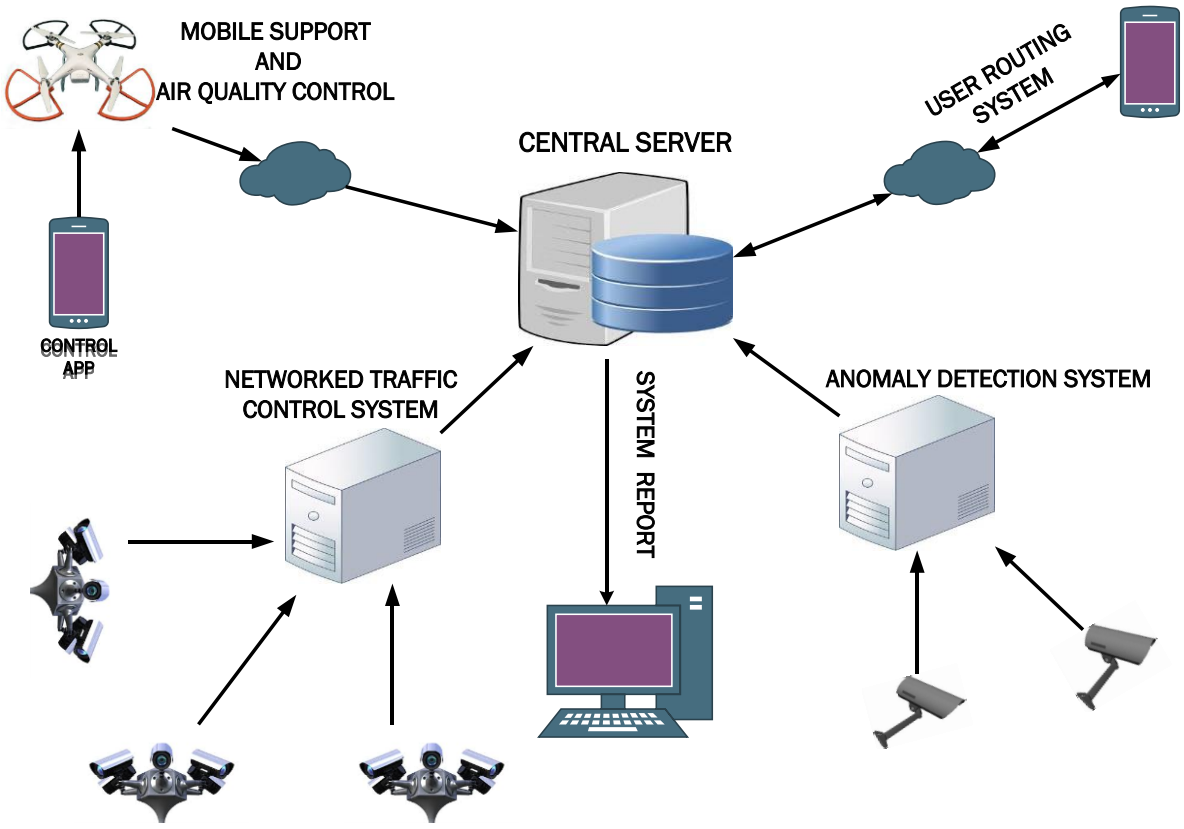

Figure 1: Top Level Architecture

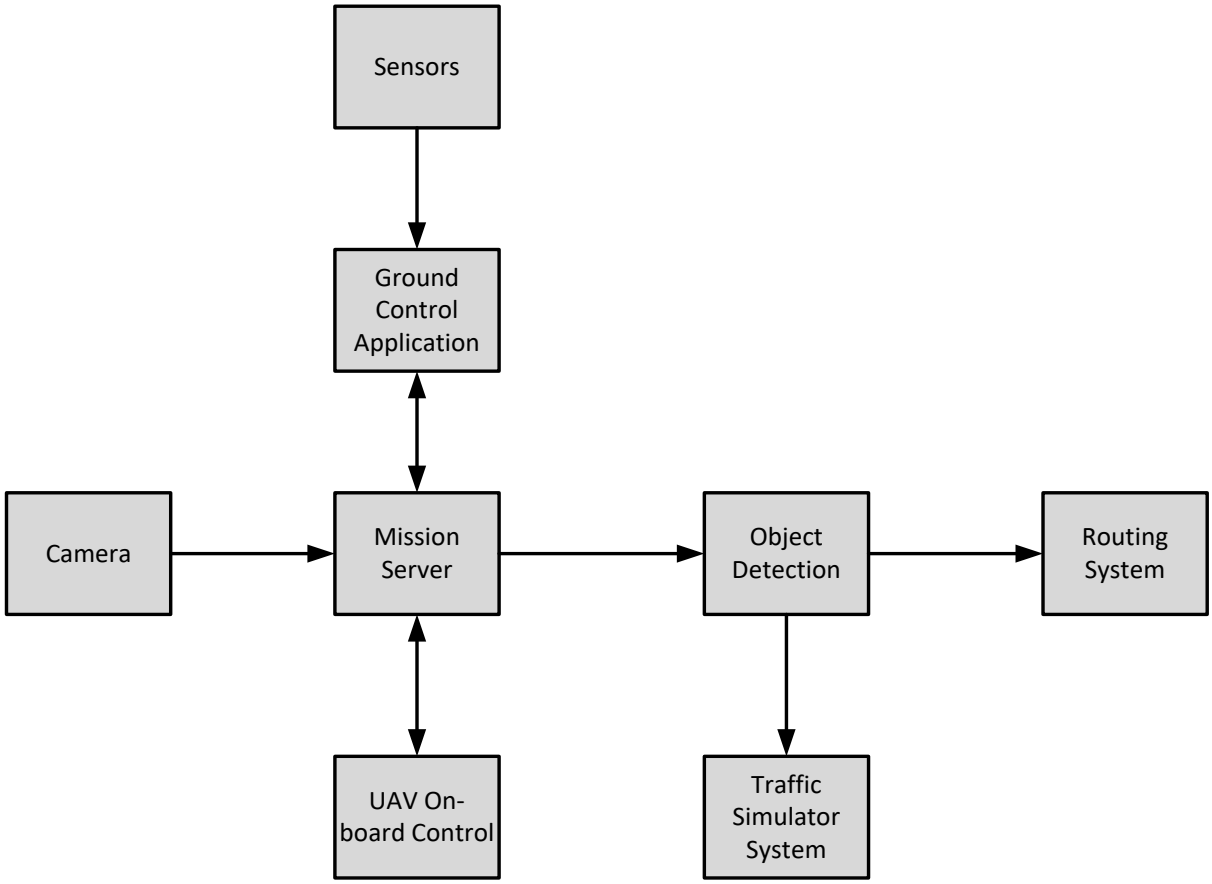

Figure 2: General Flowchart of the System

The computer vision system of this project was developed using Python programming language and Tensorflow Object Detection Application Programming Interface. The Python programming language was designed by Guido van Rossum and first appeared on the 20th of February, 1991. One core feature of the programming language that makes it very much accepted in the programming community is the strong emphasis it lays on code readability and the ability to express complex concepts in very few lines of code. Its paradigms are Object-oriented, functional, procedural and imperative.

The Tensorflow Object Detection Application Programming Interface is also an open source framework developed by Google Inc. which makes it very feasible for one to design models to perform object detection and classifications or even apply some pre-trained models available to make object detections.

Python was made as the programming language of choice for this project due to its simplicity, straightforward implementation of concept and code readability. Also, in the field of machine learning and computer vision, Python programming language is the most preferred programming language due to huge libraries it comes with and the large community it has. All these benefits it has makes it an ideal programming language to use.

After critically analyzing the situation, the main aspects noted were:

- Inefficient use the timer for traffic scheduling.

- Inability to report road accidents to those in charge as quickly as possible in order for the possible actions to be taken.
- Deteriorated state of some roads leading to unnecessary road traffic.

This project proposes an effective way to detect the density/number of vehicles and pedestrians on the road as well as the quality of the road and road accidents, and based on this data, enable traffic lights perform appropriate scheduling operations on the road.

There are many ways in which the current state of Traffic Signal Light (TSL) together with the quality of the various roads can be optimized as well as road anomalies such as road accidents, minimized using vehicle flow statistics obtained by this proposed model for vehicle detection and Traffic Signal Control. A lot of scientific research has been made over the past years by researchers in the quest to enhance computer vision and intelligence. The end results have been marvelous and considering the outcome of such researches in the fields of Computer Vision and Artificial Intelligence, a lot of progress has come out and ascertain the fact that it is possible for machines to actually see and make decisions.

These scientific advances made can therefore be applied to traffic lights to enable them operate based on the current traffic intensity of the road and also detect the state/quality of the road or accidents that have occurred on the road, etc. This approach will help ease a lot of vehicular and human traffic due to the fact that accurate or near-accurate scheduling decisions will be made by traffic lights at all times to ensure that much traffic is not accumulated at one side of the road. This will also help ease a lot of vehicular and human traffic in the city thereby making road commuting easier, safer, faster and more efficient.

Transportation has been a vital aspect of human civilization helping in transporting goods to and from various places. There are several modes of transportation: by air, water and land. There was barely any traffic congestion on our roads in the distant past. Traffic congestion happens when too many vehicles attempt to use a common road infrastructure within a specific time. Most of these roads usually have limited capacities. Traffic congestion began proving to be a challenge since the dawn of industrialization and technology in the late 19th century. Traffic congestion leads to delays which eventually result in higher wastage of fuel by vehicles. Road traffic congestion also reduces productivity. With the advent of technology, problems of traffic congestion are being reduced in countries worldwide. The introduction of traffic lights systems has significantly reduced delays on our roads but with the increasing demand of roads, better techniques would have to be adopted. The goal of this project is to analyze road traffic conditions on a citywide level, compute traffic densities at various selected road intersections and implement control algorithms to dynamically control traffic at these intersections to help minimize congestions and delays on our roads.

1.2 OBJECTIVES

To develop a computer vision backed system to:

- Examine the quantity of vehicles on the road.
- Probe into road usage by pedestrians and their safety and involvement in road traffic control.

- Observe the quality of roads.
- Find road accidents which may occur on the road.
- Report all incidences or findings to a central database

For traffic simulation:

- Retrieve vehicular traffic information from central server for traffic analysis.
- Design a simple traffic model for traffic simulation
- Implement traffic control algorithms
- Simulate traffic flow with traffic information from the central server.
- Simulate traffic flow with randomly generated traffic flow data.

For Mobile Surveillance:

- Build an autonomous drone vehicle to patrol a designated area while streaming live videos to the Object Detection System

For Routing System:

- Create an android application with capabilities of communicating with the Object Detection System in order to report anomalies to authorized personnel as soon as they occur.
- Re-route commuters whenever there is much traffic congestion on a particular road.

1.3 PROBLEM STATEMENT

With the advancement in technology, there need to be a more advanced way of controlling traffic since the use of timers in our Traffic Signal Lights no longer seem efficient due to the fact it always leads up to traffic congestion and the need for human intervention. Also, our roads deteriorate very fast always ends up in poor state due to poor surveillance.

1.4 RELEVANCE OF PROJECT

With the computer vision system being able to detect the number of vehicles on the road, an effective algorithm will then be used to control the Traffic Signal Lights based on the number of cars detected. With this measure in place, there will be less time wastage in traffic and an efficient vehicular traffic control. Also, with the ability to detect road accidents, authorized personnel will be immediately notified once there is an occurrence of a road accident. This will go a long way to save lives since there will be much faster responses to such emergencies. Also, with our road networks, the ability of the system to recognize a bad road can be used to track the state of our roads since whenever a road network is identified by the system to be in poor state, the authority in charge will be notified and appropriate actions taken.

1.5 SIGNIFICANCE OF PROJECT

- Reduction of time wasted road traffic daily is one aspect it would be of very great importance.
- Lives would be saved due to the early detection and reporting of road accidents.
- Finally, our road management and monitoring would be efficiently done from time to time.

1.6 SCOPE OF THE PROJECT

This project aims to examine the various bodies involved in road traffic control and scheduling. It therefore seeks to detect vehicles, pedestrians, the occurrence of road accidents and the also monitor the quality of the road. Any other situation that does not have direct involvement with road traffic is therefore out of the scope of this project. The road network information used within the scope of the project is Legon district and vehicles plying target roads these roads.

1.7 OUTLINE OF THESIS

The structure of the thesis is as follows. Chapter 2 presents a brief overview of past projects related to this thesis. Chapter 3 presents the system architectures and design models of the integrated system and various subsystems used. Chapter 4 extends the proposed methodologies to discuss progress made on the project, its implementation and testing. This thesis concludes with discussions and recommendations for the project in chapter 5.

CHAPTER 2—LITERATURE REVIEW

2.1 REVIEW OF RELATED WORKS

In this section, there is a brief and concise description of related works on this project and their strength and weaknesses.

2.2 OVERWIEW

Technology has gradually become a way of life and with its immense benefits it offers, all systems are getting digitized. The application of Computer Vision to the aspect of traffic scheduling has been widely embraced due to its huge advantages such as much accuracy therefore less time wastage in traffic. This has therefore led to a lot of researchers trying out various means in order to come with a better result. Also, in recent years, a lot of systems and applications amongst them, video surveillance systems, human detection, human face detection and recognition, object detection and tracking, video surveillance systems etc. have been developed.[2] Video processing techniques have also attracted a lot of attention in the various scientific research fields in recent years for the implementation of various processes such as vehicle detection.[2] The discovery of Neural Networks has also seen it being widely used in traffic control system. Hybrid computational intelligent techniques and fuzzy neural networks have been applied in multi-agents in order to control traffic signals and this have seen a massive cut in the average waiting time in traffic. Coordinate based intelligent traffic signal control systems have been proposed to reduce congestion in road networks.[3]

2.3 AUTOMATIC GENERATION OF TRAFFIC SIGNAL BASED ON TRAFFIC VOLUME

T. Sridevi et al., worked on an automatic generation of traffic based on traffic volume. In their work, they used traffic surveillance videos of vehicles as input from the MIT Traffic dataset. The videos were then further processed frame by frame while performing background subtraction using Gaussian Mixture Model (GMM). From the background subtracted result, some amount of noise was then removed with the help of Morphological opening operation and Blob analysis was finally done to detect the vehicles. These vehicles were then counted by incrementing a counter whenever a bounding box appeared for a detected vehicle. Based on this, a Traffic Signal Light was then generated. In comparing the research done by T. Srivedi et al., to my work, it is clearly seen that the approaches used are totally different. Whiles Blob analysis and Gaussian Mixture Model was applied in their process of vehicle detection, The Tensorflow Object Detection Application Programming Interface was applied to build a machine learning model to detect the vehicles.

The approach used in this research resulted in a faster implementation of the project and also, no intense computation required since the approach did not involve the application of neural networks. These are some of the advantages of this project, however, due to the application of Blob analysis and background subtraction, the angle of the camera greatly affected the detection of vehicles and this was one major downside of the research. [4]

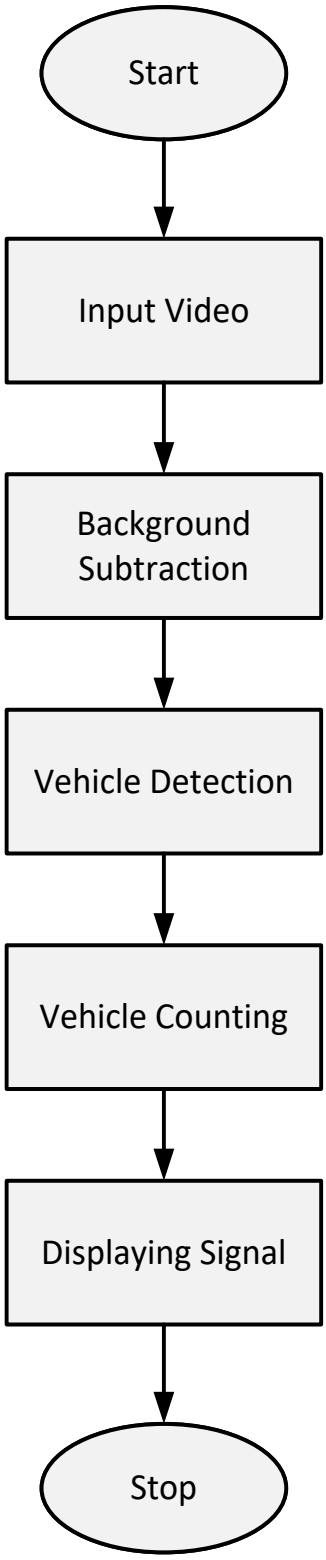

Figure 3: Proposed Flowchart of Research

2.4 AUTONOMOUS ROAD SURVEILLANCE SYSTEM: A PROPOSED MODEL FOR VEHICLE DETECTION AND TRAFFIC SIGNAL CONTROL

Another research, performed by MD. Hazrat et al. [2], was also aimed at detecting both vehicles and pedestrians. The number of vehicles were calculated per road lane and similarly a count on the total number of vehicles in all directions was made in order to decide which direction the vehicles can move. They went about their approach by first of all detecting the object, saving the images, classifying the images and then finally recording and saving the video on a hard drive. Once the system receives the first image from the camera, the image processing algorithm is applied on the acquired image and after several steps, the final video is then saved by the system on a selected hard disk. The algorithms that were used in their research for the vehicle and pedestrian detection could be described using the components: separation of the images into reference image and background image, gray scaling of the image, application of a mean filter on the image, image sharpening, background subtraction, applying motion threshold to the image and then finally, performing opening and blob segmentation.

Though this research was able to achieve its purpose of detecting pedestrians and vehicles in an image, there were some major setbacks that were faced. For instance, the results obtained as a result of the blob ratio for different objects were highly dependent on the camera view angle (degree). There had to be a lot of camera angle adjustments and each of them affected the outcome of the prediction. The detection accuracies mainly depended on the angle of the camera and not system designed. This meant that such systems could be easily flawed by occlusions and other environmental factors. Also, the parameters were rigidly set in order to obtain the needed results.

This system designed differs also differs in comparison to my system because unlike theirs, my system is mainly built based on the principles of neural networks, specifically Convolutional Neural Networks (ConvNets) and an advanced level of ConvNets in the form of Tensorflow Object Detection Application Programming Interface where based on the object to detect, a corresponding model is designed and built with gathered sample data of the object to the detected and after intense training, the model is used to make predictions. This prevents the system from being dependent on any environmental factor in one way or the other.

The positive side of this research is that, its process is quite faster to implement and it does not demand much resources to run. Nevertheless, the detection accuracy of the system mainly depends on the angle of the camera. Also, the system could be easily flawed by occlusions and the parameters used were also rigidly set in order to obtain the needed results. [2]

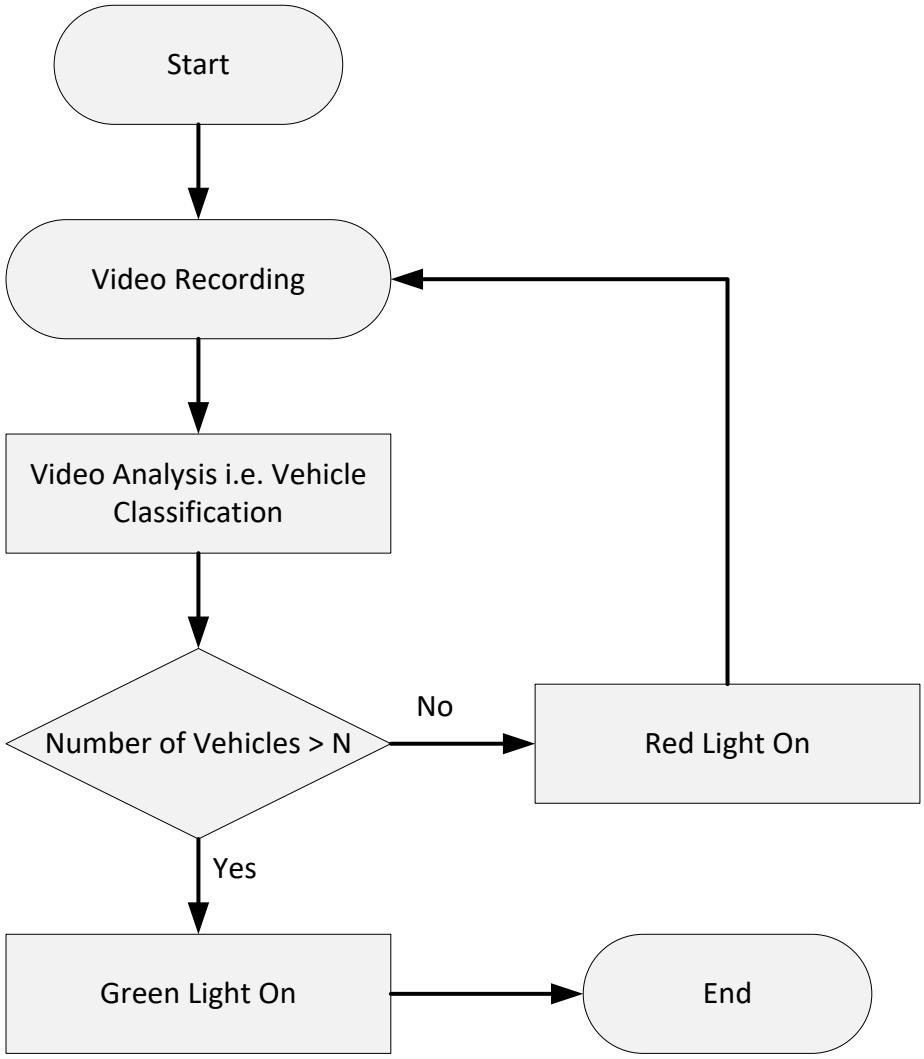

Figure 4: Proposed Flow chart of the Project

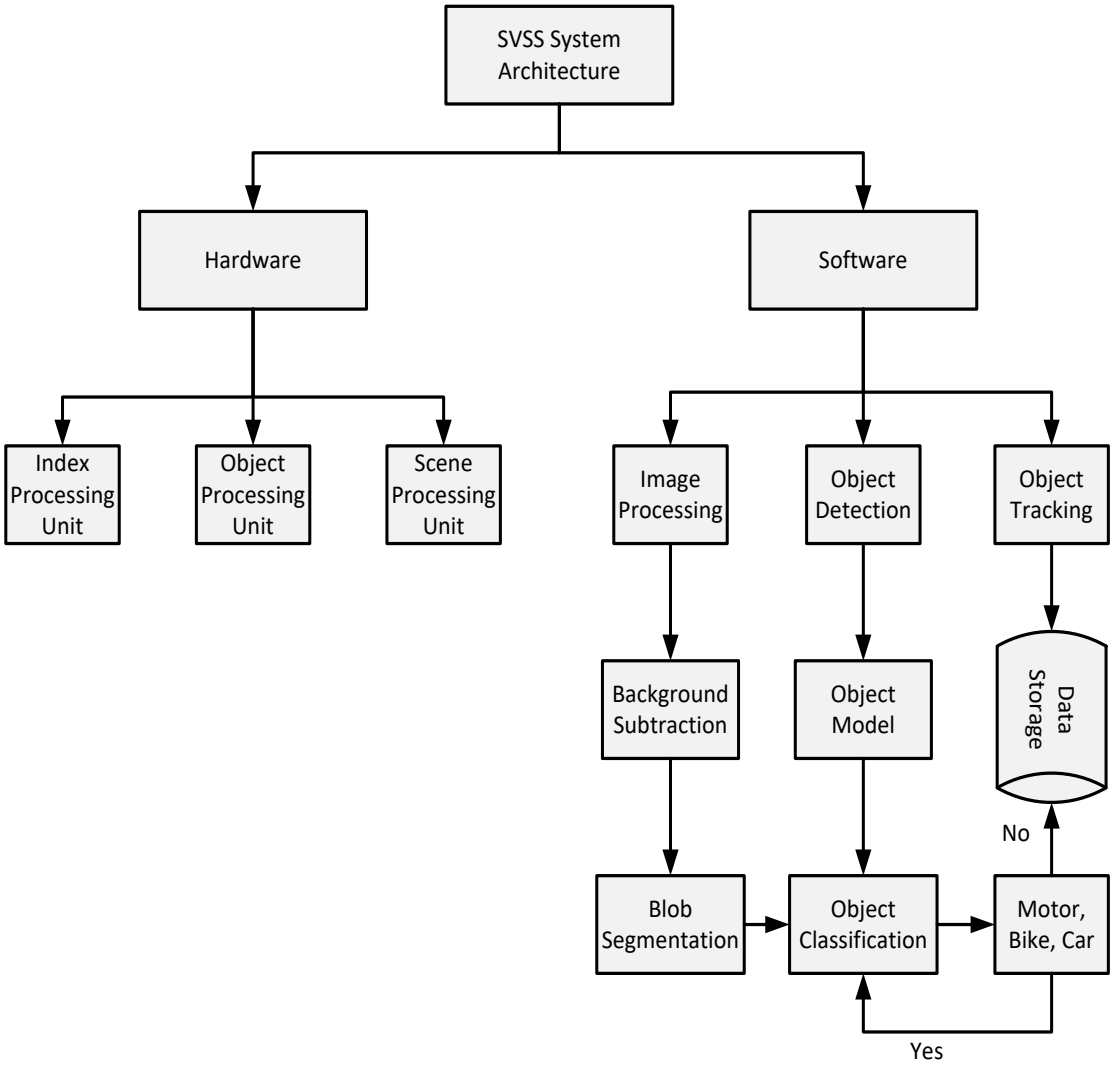

Figure 5: Proposed System Architecture for the Autonomous video surveillance

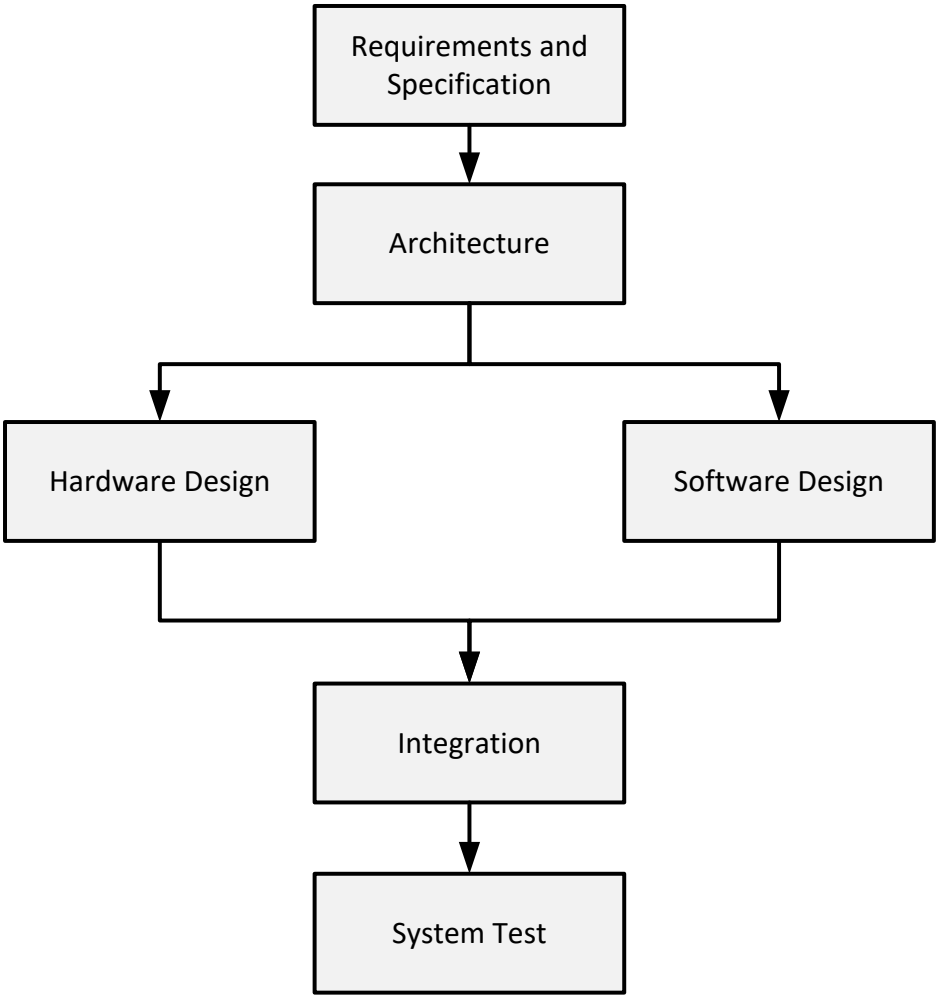

Figure 6: Proposed System Design Architecture

2.5 IMAGENET CLASSIFICATION WITH DEEP CONVOLUTIONAL NEURAL NETWORKS

In this research paper published by Geoffrey Hinton et al., there was a demonstration of how a new method that had been discovered was able to clock state-of-the-art accuracy in object detection when tested on ImageNet dataset. A large deep convolutional neural network was used to classify 1.2 million high resolution images in the ImageNet LSVRC-2010 contest into 1000 different classes with a Top-1 error rate of 37.5% and Top-5 error rate of 17.0% being achieved. The neural network architecture used consisted of about 60 million parameters, 650, 000 neurons and 5 convolutional layers with some followed by max-pooling layers. All these were followed by 3 fully connected layers and a 1000-way SoftMax output layer.

As at the time this paper was released, it was deemed as state-of-the-art and its classification accuracy was very much impressive. The negative side to this achievement was that it was very computationally expensive and very high GPUs were needed in order to achieve similar results. Also, only one category could be classified per image frame. [5]

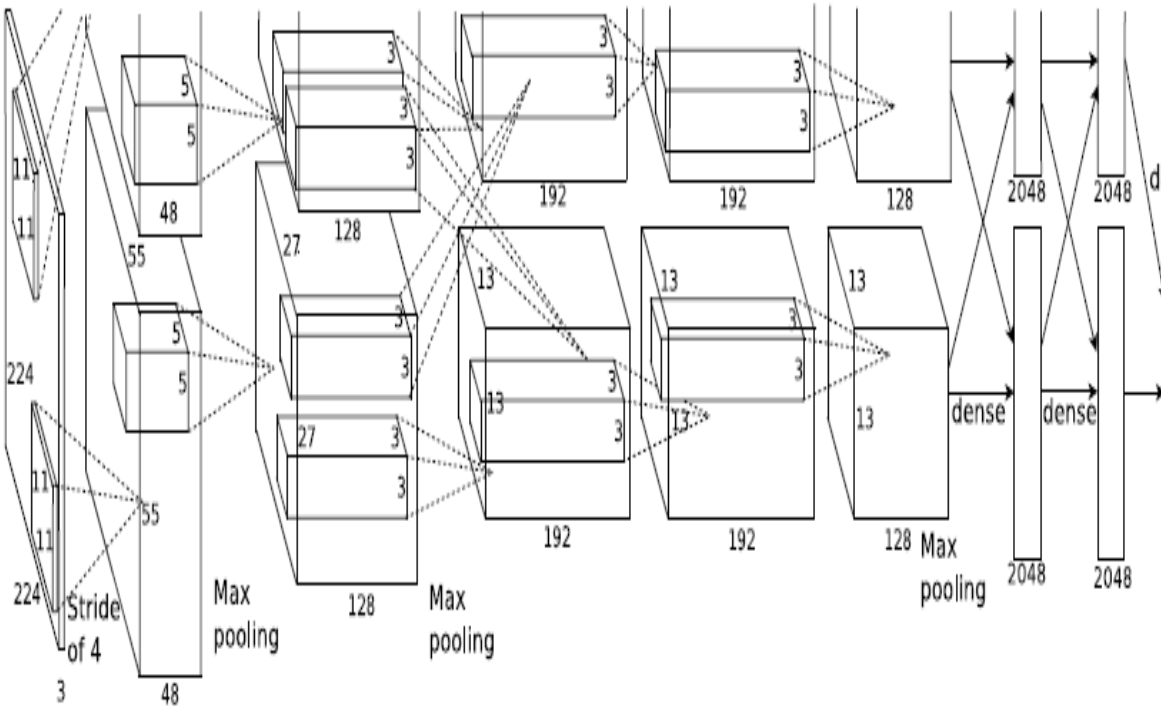

Figure 7: Illustration of the CNN Architecture used [5]

2.6 REAL-TIME MULTI-VEHICLE DETECTION AND SUB-FEATURE BASED TRACKING FOR TRAFFIC SURVEILLANCE SYSTEMS

This paper also proposed a real time multi-vehicle detection and tracking approach. Lane marker was carried out for vehicle counting on each lane. This helped to remove foreground noises and shadows. Also, instead of blob tracking, vehicle sub-feature based Kalman filter was used. This made the system more robust to partial occlusions which happened a lot in congestions. This approach is scalable to most freeway surveillance video. This approach, when compared with standard blob tracking was found to outperform it in terms of correct tracking rate. This paper embeds lane detection which provides an accurate estimate of the vehicle position on each lane. After each tracking step, the features are grouped and combined with the contours to refine and update the vehicle position until it leaves the detection and tracking zone.

The use of Kalman filter instead of blob tracking made the system more robust to partial occlusions and also it was also able to outperform systems applying blob tracking in terms of correct tracking rate however, the position of the camera hugely affected the detection accuracy. This is an example of the weakness of this research. [6]

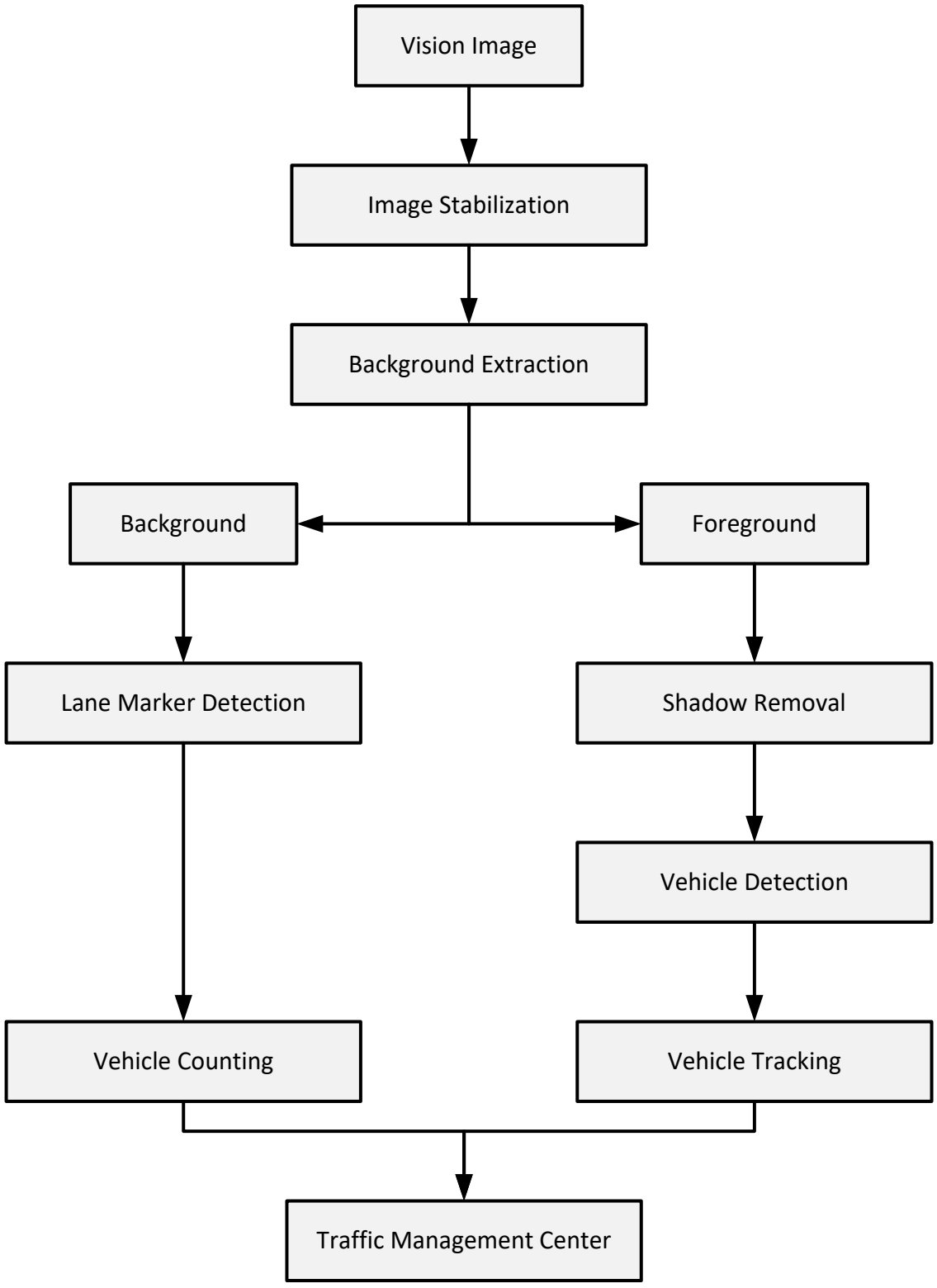

Figure 8: Proposed Block diagram of the vehicle tracking and counting system

2.7 AN INTEGRATED CONTROL APPROACH FOR TRAFFIC CORRIDORS

2.7.1 INTRODUCTION

Markos Papageorgiou[7] presented a unified approach to the design of integrated control strategies for traffic corridors of arbitrary topology including both motorways and signal controlled urban roads. The main objective of his work was to minimize the total delay or the total time spent in a network. His approach involved the formulation and solution of linear optimal control problem based on store-and-forward type of modelling. The basic methodology employed was to formulate the integrated corridor traffic control problem in the format of an optimal control problem based on a store-and-forward type of mathematical modelling. Control problem was formulated as a linear programming problem considering traffic signal controls, ramp metering, motorway control, route guidance and motorway control. An overall objective function was incorporated into the system taking into consideration all constraints in the system and solutions were found using numerical algorithms.

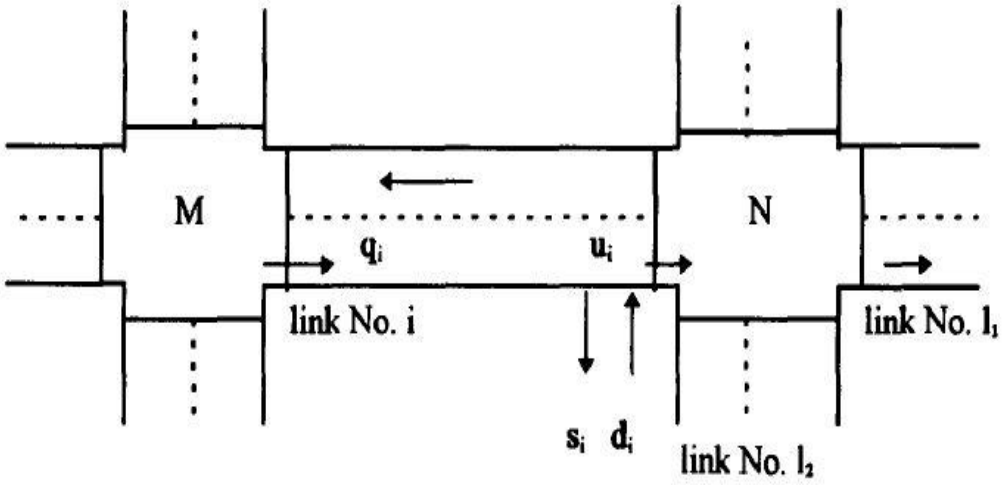

Figure 9: Flow theory for a simple two intersection network

2.7.2 REVIEW OF WORK

The paper provided an optimal solution to very complex control problem and includes a powerful algorithm that guarantees the calculation a global minimum subject to all constraints. However, the approach was purely mathematical and theoretical. Also, there was no graphical user interface for visualization. Computation becomes very expensive and impractical as the network size grows.

2.8.0 REVIEW OF ROAD TRAFFIC CONTROL STRATEGIES

2.8.1 INTRODUCTION

This paper[8] provided an overview of advanced traffic control strategies for three particular areas: urban road networks, freeway networks, and route guidance and information systems. The paper classified control strategies for road traffic control based on some characteristics. Fixed-time strategies are applicable for a given time of the day, isolated strategies are applicable to single intersections while coordinated strategies consider urban zone or even whole network comprising of many intersections.

2.8.2 REVIEW OF WORK

This paper provides a better understand of road traffic problems. It also provides in-depth analysis of different approaches to solving road traffic problems.

2.9.0 AUTOMATIC GENERATION OF TRAFFIC SIGNAL BASED ON TRAFFIC VOLUME

2.9.1 INTRODUCTION

This paper[4] proposed a system to control traffic signals based the traffic volume detected on a selected link. The main features of the system were facing detection, pedestrian detection, and vehicle detection. A single road link was used to test the system.

2.9.2 REVIEW OF WORK

Pedestrians and vehicles are well detected. The traffic volume of a network is well calculated too. However, traffic control is only limited to a single link not a network of them.

2.10 TRAFFIC FORECASTING AND NAVIGATION ASSISTANCE SYSTEM VIA WEB APPLICATION[9]

This project presents a model on Vehicular Traffic forecasting and navigation assistance. This was designed in a web application to provide guidance information to drivers for efficient route planning. The system uses the technology of Global Positioning Satellites to acquire traffic data in the form of traffic densities and makes decision on the most effective route to ply.

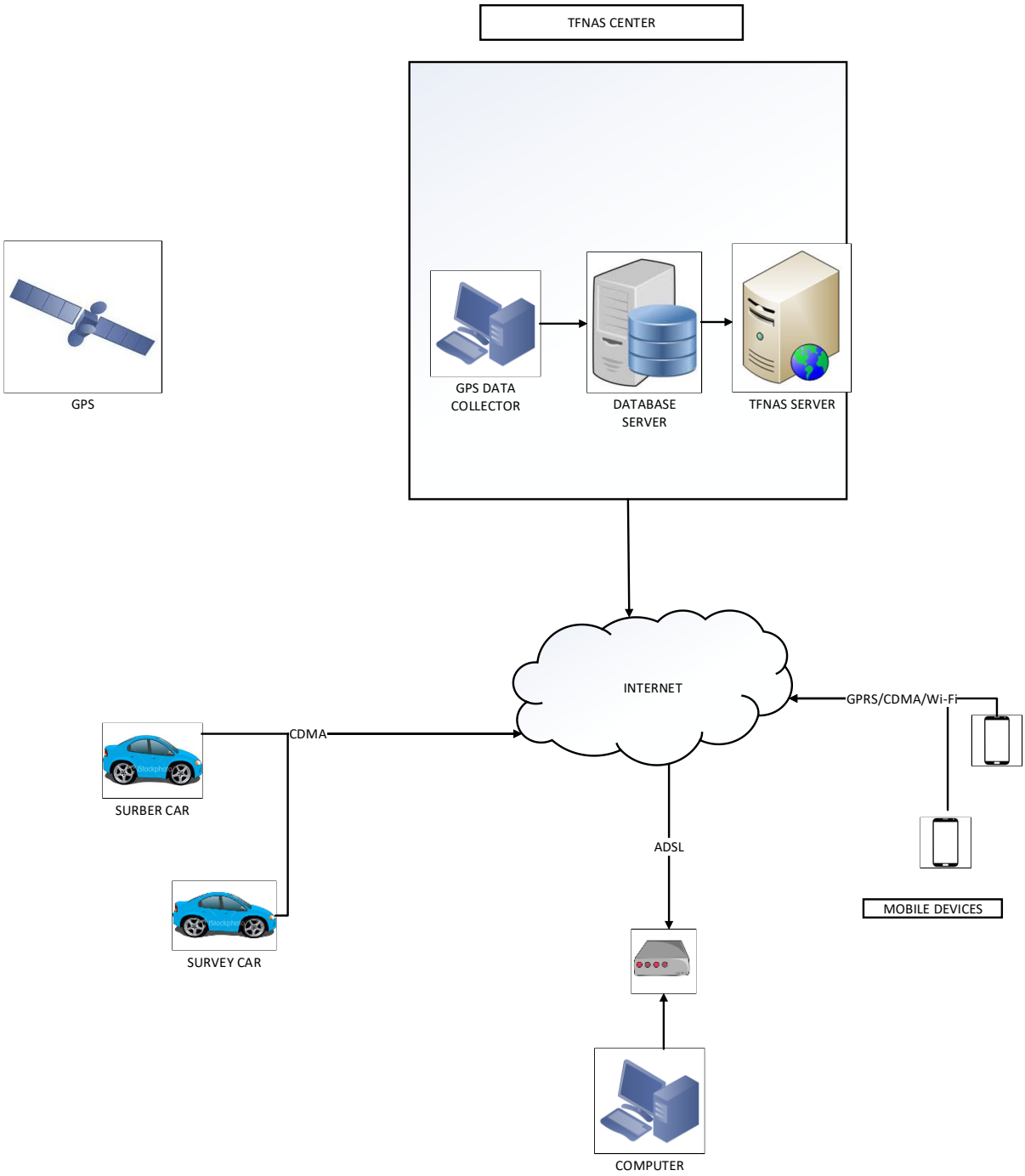

Figure 10: Project Architecture [9]

The above diagram displays the architecture of the system. The system has 5 parts namely: GPS module, Survey cars, TFNAS centre, Internet Connections and front-end devices like phones.

GPS is a satellite-based navigation network system consisting of 26 satellites in orbit.[9] The receiver of a GPS calculates the exact location of a user.]9] By this technology, the speed of the vehicle can be calculated by finding the location displacement over a period. The survey car is deployed and the speed of the car is calculated and reported to the centre using a Wi-Fi

or GPRS connection. At the TFNAS centre, the data from the survey cars are received in real-time and saved in a database.

The speed of the survey car together with the route distance is therefore used to rate the various routes. The formula: $T = DAVD$ where T is the Driver time, D is the driver distance and AVD is the average vehicle distance. The front-end is used to display routes calculated for by the above equation.

The obvious disadvantage of this project is the use of survey cars. This approach will prove to be more expensive. Acquiring multiple cars, having to fuel them for trips and general maintenance alone will prove costly. In the paper, the system was tested with a location with 5 possible routes from the current location. This could mean that there needed to be 5 survey cars. However, having still cameras with Internet of Things capability will prove to be cheaper and more cost efficient.

2.11 SMARTROPOLIS: AN AUTOMATED MONITORING SYSTEM FOR AIR QUALITY, TRAFFIC AND VIOLENT SCENE FOR SMART CITY

This project sought to utilize the concept of Smart-City and Internet of Things to build systems for various functionality. The main objectives were to:

- Acquire data from devices like cameras, to be processed and analyzed in real-time.
- Build an intelligent Traffic Control Monitoring and Prediction System.
- Model and build an intelligent Air Quality Monitoring System.

These were just a number of their objectives related to 0ur system. The project was a combination of Computer Vision and Machine Learning concepts. For a part of the project called the SmartTraffic system, still-cameras connected to single-board computers (raspberry pi) were mounted at points on streets of the city at specific heights and angles for capturing accurate data. The camera provides the raspberry pi with the video feed it needs to give the average speed of the vehicles moving in that video as the output.

The process of calculating the average speed involved the use of various Computer Vision techniques. The techniques involved background subtraction and object detection using vehicle contours and tracking over a video sequence. Therefore, if an object (car) is detected in the video stream, its displacement in the video stream is found by comparing discrete frames at various times interval.

The in-frame displacement is converted to real life displacement and the speed of the object is calculated. Taking the speeds of multiple vehicles over some period of time, they were able to calculate the average speed of vehicles along a particular route. This output is then sent to a database on a server. They developed an android application to display the average speed of various parts of a street along various routes selected by the end-user.

2.12 A SMART TRANSPORTATION SYSTEM FACILITATING ON-DEMAND BUS AND ROUTE ALLOCATION[10]

This project sought to investigate the feasibility of Internet of Things based hardware modules and how they can be used to optimise transportation. They modified an algorithm called Dijkstra's algorithm for shortest paths to incorporate traffic conditions and other features. The new algorithm was then evaluated for scenarios and the results were discussed. The system was to be implemented for busses using GPS devices on microcontroller system for route selection and optimization.

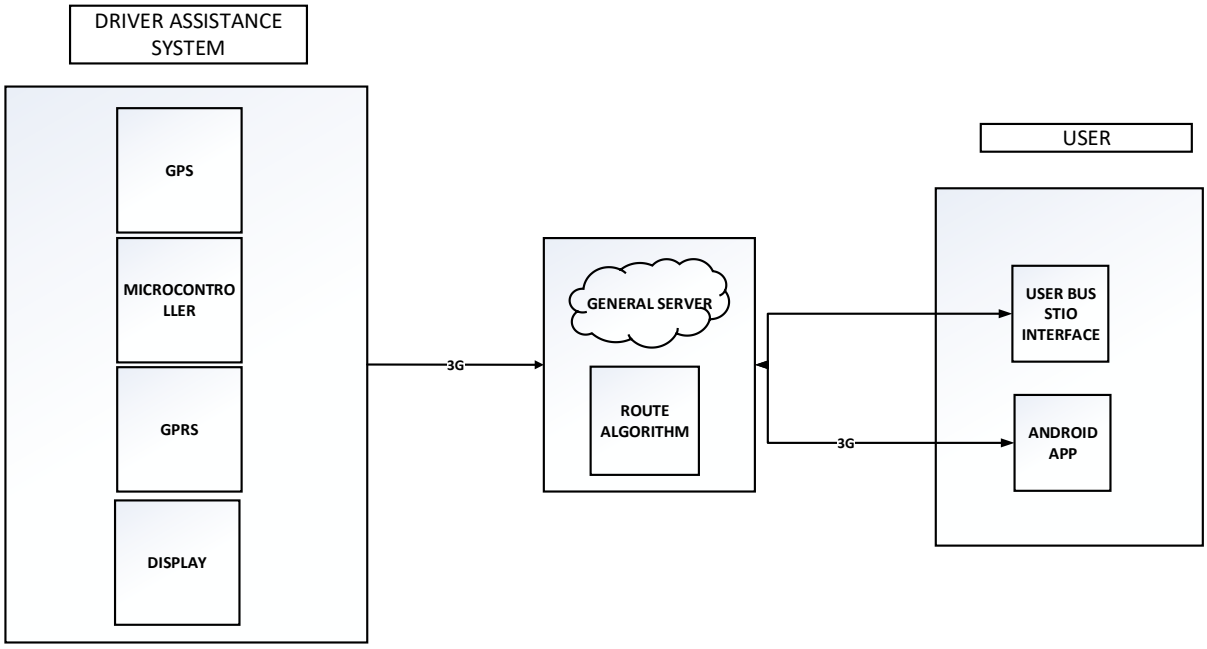

Figure 11: Project Architecture [10]

The system consisted of an on-board Driver Assistance System (DAS), User Interface, Central server and android application as in the figure above. The DAS connects wirelessly to the central server for data exchange. The GPS unit gives accurate update of bus position of the bus in real-time. The DAS display unit display and shows driver route selection as they drive the vehicle. The GPS unit used 3G connectivity for quick updates. The android application was built to help display the optimal route after a user selects the destination.

For the route optimization and selection, the Dijkstra algorithm was only able to find the shortest path to a destination which was not always the optimal one. In this module, the algorithm was modified to consider factors like city speed limits, current traffic conditions etc. The weights of various routes are calculated with an optimization objective as:

$$W = \min(d, c, \max(s), \max(r)) \text{ [10]}$$

Where:

$$d = distance\ between\ source\ and\ destination\ of\ all\ possible\ routes\ in\ km.$$

$$c = Traffic\ congestion\ factor\ between\ 0\ and\ 1\ with\ 1\ being\ highly$$

$$s = Speed\ limit\ for\ a\ route$$

$$r = the\ number\ of\ similar\ requests\ on\ a\ route\ to\ a\ location$$

The number of requests was necessary in this case because the project was supposed to be for a public transport system. The bus needed to prioritise places where customers were making request to join the bus.

Our project sought to use real-time traffic data together with road quality status, distance and other factors for the optimization. The project also sought to use Fuzzy Logic algorithms for the optimal route selection.

2.13 AUTOMATIC SPEECH RECOGNITION [11]

In the subsequent sub-chapters, we will be looking into literature that has to deal with Automatic Speech Recognition (ASR) systems. These systems detect and convert speech from recorded audio signal to text format.[12] These systems try to infer words from the recorded audio. There exist numerous approaches for Speech Recognition but the commonest is the probabilistic approach.[13] By this method, a test speech signal corresponds to a word or sequence of words depending on some probability factors.

By this method, to associate word X with a speech signal we need to calculate some scores. We calculate these scores by analysing the phonemes of the various words in the knowledge base.[13] Also, we need to know how probable it is for a word to follow another. By these determining factors, we select the word with highest probability score as the recognized word.

The process of Speech Recognition can be divided into steps:

- Pre-processing – processing the speech signal and preparing it for the recognition process. This includes filtering the signal, sampling etc.[13]
- Feature extraction – extracting word signals from the raw audio signal to generate feature vector. [13]
- Decoding – Using the language model, acoustic model and dictionary to recognize speech.
- Results output[13]

2.14 STUDY OF DEEP LEARNING AND CMU SPHINX IN AUTOMATIC SPEECH RECOGNITION[12]

This study was on the various ways in which Deep Learning Neural networks are applied in the field of Automatic Speech Recognition. It also looked at the application of the CMU Sphinx software which is an open source Speech Recognition software.

The paper enlisted 3 approaches to speech recognition, namely:

2.14.1 THE ACOUSTIC PHONETIC APPROACH[12]

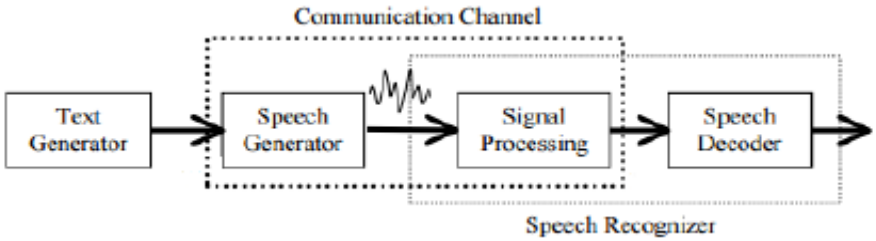

Figure 12: Speech Recognition and Generation Process[12]

From the diagram above, the text generator is our human brain. The speech generator is the vocal tract. The book described speech as a composition of various frequency components. The paper researched into the various parts of the body that are related to speech and also made some observations concerning how the tongue and teeth. It observed the various positions of the tongue and how the closeness of the tongue to the teeth determine the frequency of the sounds generated.

These observations are the facts that help in the Acoustic-Phonetic approach. The first step in the approach is pre-processing. By this approach, the features of every sound frame were calculated using Mel Frequency Cepstral Coefficients (MFCC) and Linear Predictive Coding (LPC). These two measures provide the spectral description of the speech signal over time.[12]

The next step is the detection stage where the spectral measurements are identified with each phonetic unit.

In the third step, the system finds where the spectral or other features remain approximately constant. The stable regions help with segmentation. The segmented regions are labelled according to the phonetic units whose features are most close. The word is then outputted.

2.14.2 THE PATTERN RECOGNITION APPROACH[12]

For this approach, there exists two methods which will both be explained. The pattern recognition system tries to make speech patterns out of features like the frequency component arrangement to build a classification set based on the similarity between the test pattern (test speech signal) and the classified speech signals.

For the first method, the features of every sound file are measured using Discrete Fourier Transform analyser or Filter Bank analyser. These analysers produce features like MFCC and LPC. After this the next step is training of test patterns. The multiple speech patterns of a corresponding class (text) are used to create a new pattern that houses all the invariant features of the class. This pattern can now be used to classify the speech signal under a particular class.

The next step will now be to classify the patterns. In this step the test pattern (test speech signal) is compared with the various classes. The similarity between the test pattern and the reference pattern (classes) are computed by using methods like the Mahalannobis distance. The highest similarity is chosen as the class of the test pattern.

The second method is the method used by the CMU Sphinx software. CMU Sphinx uses a language model, acoustic model and a dictionary. The dictionary specifies the phonetic units of the words to be recognized. The language model consists of a large set of grammar sentences. The complexity of the model depends on the number of sentences to be recognised.

The pattern recognition is done using. Mathematically:

Given a set containing acoustic data: $A = a_1, a_2, a_3, \dots, a_k$

Given a sequence of words: $W = w_1, w_2, w_3, \dots, w_k$

The goal is to maximize the probability of W given $A * P(\frac{W}{A})$

$$P\left(\frac{W}{A}\right) = \frac{P\left(\frac{A}{W}\right) * P(W)}{P(A)}$$

$$P\left(\frac{A}{W}\right) = Acoustic\ Model\ (HRMMs)$$

$$P(W) = Language\ Model$$

$$P(A) = Constant\ for\ a\ complete\ sentence$$ [12]

The paper finally stated on this topic that conventional speech recognition systems use the Gaussian Mixture Model based Hidden Markov models to represent the sequential structure of speech signals.

2.14.3 THE DEEP LEARNING APPROACH [12]

The paper viewed the Deep Learning Approach as a successful replacement of the Gaussian Mixture Models with Hidden Markov Models because they are more efficient as compared to GMM-HMM systems. The paper also spoke of the fact that the Pattern Recognition Approach was disadvantaged because to achieve pattern recognition, there will need to be done before the invariant features in the signal can be determine and recognized. Therefore, a great effort is put into the analysis including computational power and storage.

Deep Learning employs the technique of Convolutional Neural Networks which automatically learns the invariant features for recognizing and labelling the test speech signals.[14] Since these are Machine Learning algorithms, they learn from the invariant features used in the training process to distinguish between the various words.

The neural network model is fed with raw speech signal divided into a sequence of sound frames. These raw speech frames are classified using the convolution layer. The paper also mentioned that the best tool for creating Neural Network model is the Google TensorFlow.

The paper also went ahead to view the differences between CMU Sphinx and the Deep Learning approach. The paper stated first, that the Deep Learning approach requires more data for in terms of the acoustic functionality as compared to CMU Sphinx.[12] The acoustic model can be adapted for different accents in the CMU Sphinx after about an hour of recording by the

speaker but the Deep learning approach with have a more cumbersome approach to solving this problem.

The accuracy and performance of Deep Learning also depends on the amount of data fed to the Network. As more data is used in training both the CMU Sphinx and Deep Learning approaches, the gap between their performances widen. The Deep Learning achieves greater performance as compared to the CMU Sphinx.

In the Deep Learning Approach where the Convolutional Neural Networks multi-layered approach is used, the training data should be a map of features. Examples of these features were mentioned in previous sub-chapters. These features go through layers of convolution and pooling layers for to finish the training process.

The paper finalised that the best Deep Learning system will be better than the best CMU Sphinx system by a large margin.

2.16 FUZZY LOGIC AND THE IMPLEMENTATION IN ROUTE PLANNING

Fuzzy logic is an extension of Boolean logic developed by Lot Zadeh in 1965 based on the mathematical theory of fuzzy sets.[15] Fuzzy logic introduces the notion that not all logical decisions can be viewed as True or False decisions.[16] Rather, it works on the principle of degree of Truth of a case. Degree of Truth meaning that a case might not always be true or false. An example is the multi-class classification problem. [15]
Fuzzy logic is based on Fuzzy set which is a generalization of the classical set theory.[17] A membership function that classifies data under a subset or a degree of truth.[17] Fuzzy logic is called an expert algorithm because the rules of the membership function are defined by the person developing the logic algorithm.[18] Either this or based on statistical data, the membership function can be defined.[15] This project seeks to use the concept of Fuzzy Logic for optimal routing.

2.17 REAL-TIME VIDEO RELAY FOR UAV TRAFFIC SURVEILLANCE SYSTEMS THROUGH AVAILABLE COMMUNICATION NETWORKS[19]

In this paper, the authors described a data link to connect an on-board camera of an unmanned aerial vehicle to monitoring terminals at the Michigan Department of transportation. The video stream was viewable in real time at the terminals or stored for later analysis. The video signal was relayed over mobile networks.
In their method, the authors developed a system where the UAV sends its IP address to the end user computers which open a media player to retrieve the video stream from that address. This was done in order to be able to deal with dynamic IP addresses on certain networks. An alternate scheme involved using a central server that receives a video from the UAV and then makes it accessible for end-users via the servers IP address as well as storage for later analysis. Video streaming was done via the HTTP protocol.
The authors utilized Spring and Novatel Wireless Networks which operated on CDMA 1x Evolution Data Optimized (EvDO) and 1xRTT wireless technologies. This allowed for a maximum forward link capacity ranging from 500 Kbps up to 2.4 Mbps, and a maximum reverse link capacity of 153 Kbps.

For recording the real-time video stream, a Sony DV video deck device was used with the file compressed using the Windows Media Format.
While the authors noted the security concerns of using public networks, it was noted that this method led to a remarkably easy setup to implement as opposed to building a private network infrastructure.
Their experimental results showed that this method could deliver quality videos in real time by successfully relaying a video signal with image size of 320x280 resolution at 15 frames per second.

2.18 APPLICATIONS OF DEEP LEARNING AND UNMANNED AERIAL VEHICLE TECHNOLOGY IN TRAFFIC FLOW MONITORING [20]

In this paper, the authors seek to study the real time capturing of traffic video data through UAV to understand road traffic density conditions. They used Faster-RCNN to train a model to identify vehicles in the video. From this, the traffic flow was calculated to derive the road density condition.
The authors utilized a video with frame rate of 24 frames per second with their implementation sampling the video at 3 frames per second and 6 frames per second. The authors did not specify the resolution of the video stream or the method by which this stream was fed into their recognition model.
However, in their paper the authors proved that UAVs can be a flexible method to monitor road conditions. And that the video streamed from such source could adequately be used to accurately identify vehicles on the road and hence draw accurate statistical inferences on the road density conditions from the provided data of continuous video frames.

SUMMARY OF LITERATURE REVIEW

Papers	**Approach/Test System**	**Mathematics Formulations**	**Strengths**	**Weaknesses**
M. H. Ali S. Kurokawa A. A. Shafie Autonomous road surveillance system: A proposed model for vehicle detection and traffic signal contro	This system was tested on a motor bike, a mini truck, a big bus, a van and a human since each had a different blob ratio. The camera was also positioned at specific angles during image capturing on the road.	$h'_f(z) = \frac{1}{2K+1} + \sum_{j=-K}^{K} h_f(z+j)$	1. Faster implementation 2. Less resources required to run the system.	1.The detection accuracy of the system mainly depends on the angle of the camera. 2.The system could be easily flawed by occlusions. 3.The parameters used were also rigidly set in order to obtain the needed results
T. Sridevi A. Proffesor Automatic Generation of Traffic Signal Based on Traffic Volume	Traffic surveillance videos were of vehicles were taken from the MIT Traffic dataset, processed and background subtraction applied to it. Blob analysis was used to detect the vehicles on the road and a signal		1. Less intense computations. 2. Faster implementation	Due to the application of Blob analysis and background subtraction, the angle of the camera greatly affected the detection of vehicles

	was generated depending on count.			
A. Krizhevsky I. Sutskever G. E. Hinton ImageNet Classification with Deep Convolutional Neural Networks	A large deep convolutional neural network with about 60 million parameters and 650,000 neurons was used to classify 1.2 million high resolution images in the ImageNet LSVRC-2010 contest into 1000 different classes.	$b_{x,y}^{i} = \frac{a_{x,y}^{i}}{\left(k + a\sum_{j=\max\left(0,i-\frac{n}{2}\right)}^{\min\left(N-1,i+\frac{n}{2}\right)}\left(a_{x,y}^{j}\right)^{2}\right)^{\beta}}$	1. Impressive classification accuracy	1. Very high GPUs required to achieve state-of-the-art results. 2. Only one category could be detected per frame.
L. Huang Real-time multi-vehicle detection and sub-feature based tracking for traffic surveillance systems	Lane marker was carried out for vehicle counting on each lane. Vehicle sub-feature based Kalman filter was used in the vehicle detection. This made the system more robust to partial occlusions which happened a lot in congestions	$p(X) = \sum_{i=1:K} w_{i,t}.N(X;\mu_{i,t},\sum i,t)$ $B = argminK_b(\sum_{k=1}^{K_b} w_k > T)$ $X_k = f(X_{k-1},u_k;\ v_k),$ $Z_K = h(X_k,w_k)$	1. The use of Kalman filter instead of blob tracking made the system more robust to partial occlusions. 2. It was also able to outperform systems applying blob tracking in terms of correct tracking rate	1. The position of the camera hugely affected the detection accuracy

CHAPTER 3—METHODOLOGY

3.1 VEHICLE AND ANOMALY DETECTION SYSTEM

3.1.1 INTRODUCTION

This chapter lays a major emphasis on the procedure and systematic approach which was used to achieve this project. In this chapter, details of the various technologies implemented (both hardware and software) are explained while information about the process of development followed is noted.

3.1.2 PROJECT OVERVIEW

Since this project consisted of a software system which will be executed and made to function on a hardware device, this project was divided into two sections — 1. Software module and 2. Hardware module. The software module was developed using the following software technologies:

- Python Programming Language.
- TensorFlow Object Detection Application Programming Interface

The TensorFlow Object Detection Application Programming Interface is an open source library which makes it very feasible for one to perform object detection by either training his/her model ground up or by utilizing any of the available pre-trained models. This open source library is the tool which was used to accomplish the vehicle, pedestrian, bad road and road accident detection and it was implemented using the Python programming language.

The hardware module of the project comprises of:

- A Raspberry pi 3 that enables the video frames captured to be sent to the server for processing
- A Raspberry pi camera which was used for the recording of the video frames
- A Wi-Fi dongle which serves as a source of internet connection to enable the captured video frames to be transmitted to the server via internet.

3.1.3 SYSTEM ARCHITECTURE

The system architecture consists of the general architecture and the various proposed flow charts of the system.

3.1.3.1 PROJECT OVERVIEW

The general architecture is made up of a display of the various component (mostly hardware) that were used to develop the hardware module.

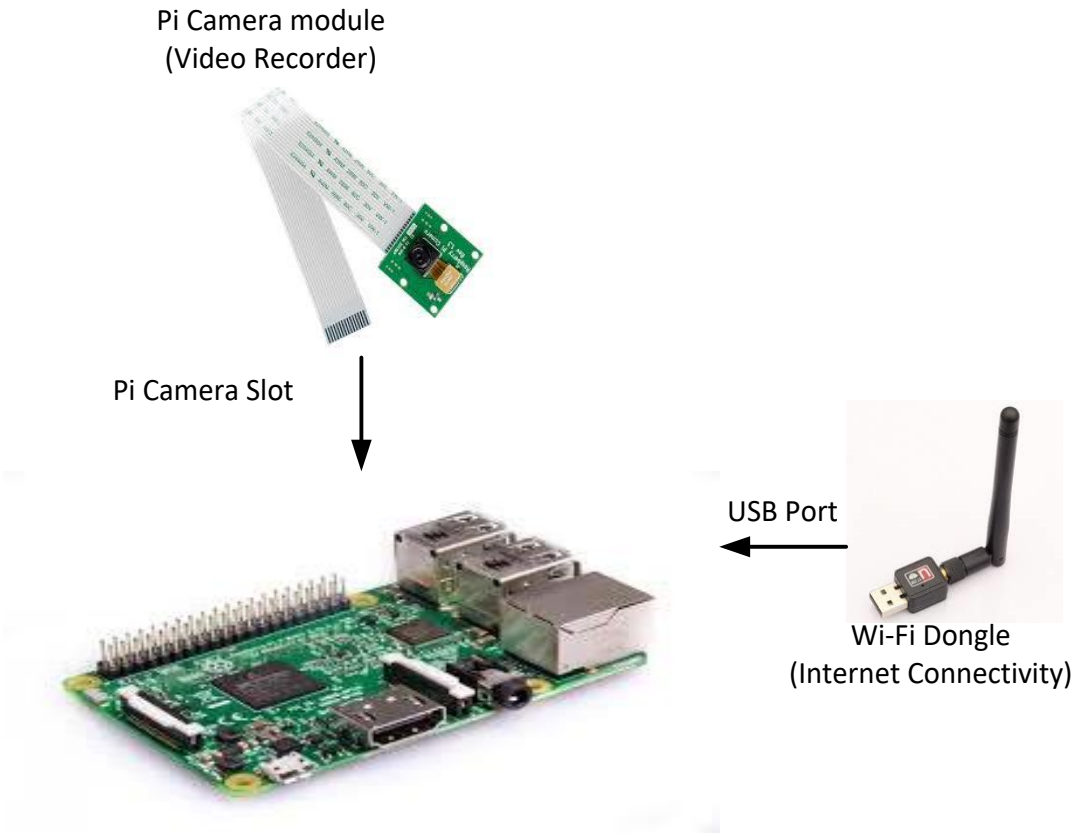

Figure 13: General Architecture displaying the various components utilized

3.1.3.2 FLOW CHARTS

This section contains the flow chart of how the object detection model used to detect vehicles, pedestrians, bad roads and road accidents was designed and built. It also lay emphasis on the steps followed by the system in detecting vehicles, pedestrians, bad roads and road accidents by showing the flow chart of how the system functions.

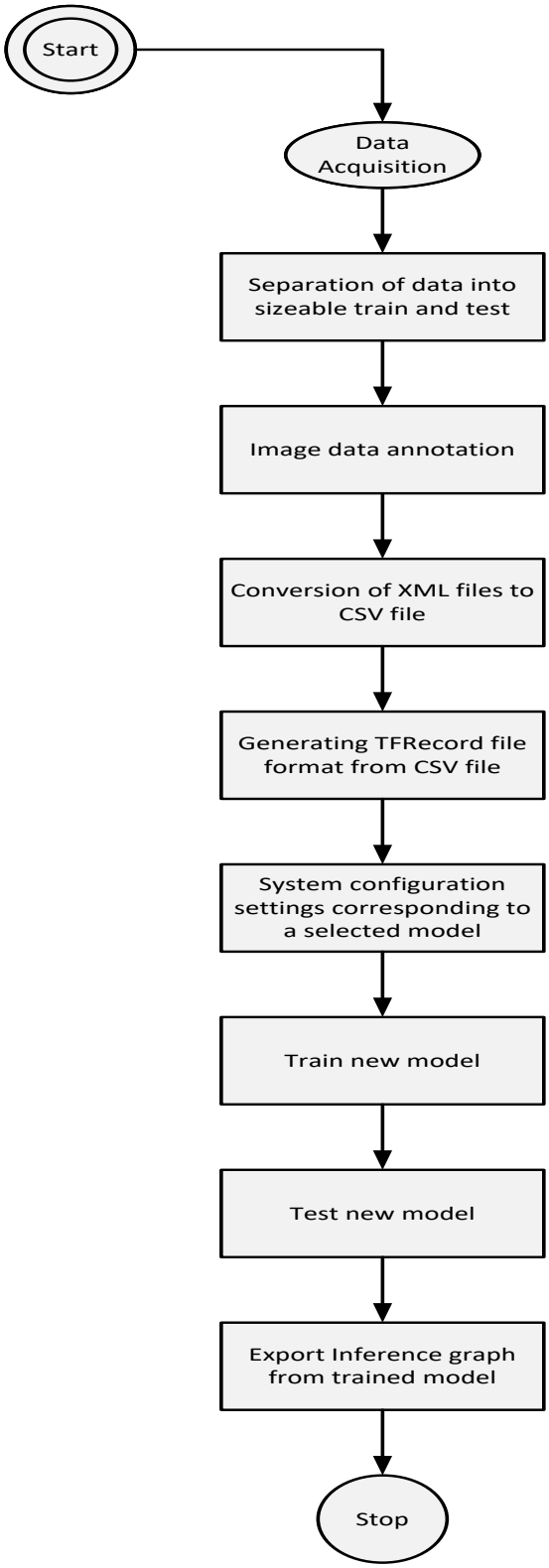

Figure 14:Flowchart of how the object detection model was built

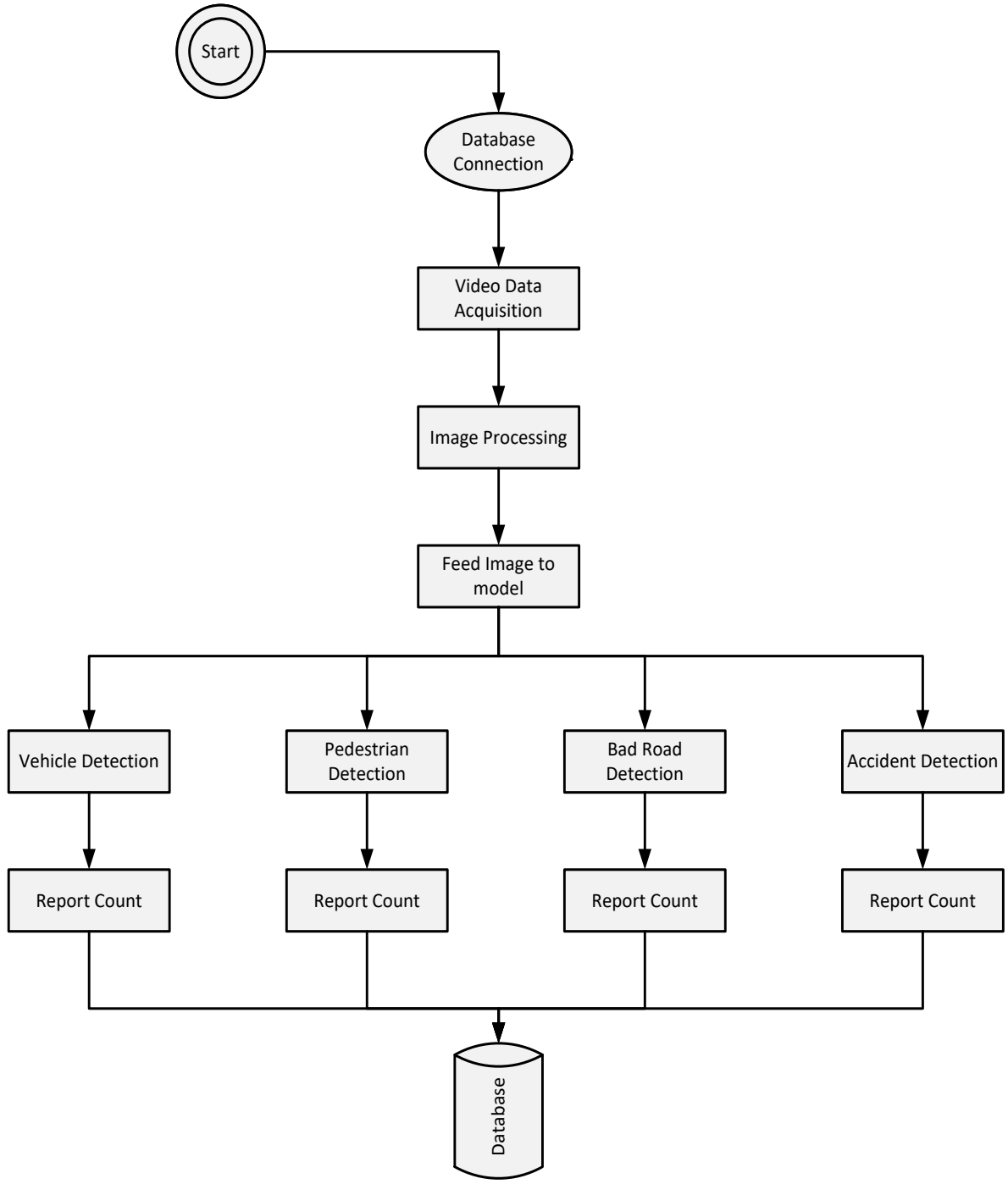

Figure 15:Flow chart of how the object detection system functions

3.1.4 FUNCTIONAL REQUIREMENTS

3.1.4.1 FUNCTIONAL REQUIREMENTS

The system should be able to:

1. Detect vehicles.
2. Probe into the number of pedestrians on the road and how they relate with road traffic control.
3. Examine road accident should one occur.
4. Discover roads that are out of shape.
5. Count the number of vehicles, pedestrians, road accidents and out of shape roads detected.
6. Report all the information gathered to the central database.

3.1.4.2 NON-FUNCTIONAL REQUIREMENTS

1. Vehicles and pedestrian detection should be done at a very fast pace.
2. The object detection should not be affected by weather conditions.

3.1.5 HARDWARE SPECIFICATIONS

3.1.5.1 RASPBERRY PI 3

The Raspberry Pi is a series of small single-board computers. The Raspberry Pi 3 has its processor speed ranging from 700 MHz to 1.4 GHz with on-board memory ranging from 256 MB to 1 GB RAM. The board has one to four USB ports and for video output, HDMI and Composite video are supported. In terms of processor, the Raspberry Pi 3+ uses a Broadcom BCM2837B0 SoC with a 1.4 GHz 64-bit quad-core ARM Cortex-A53 processor, with 512 KB shared L2 cache. In terms of performance, the Raspberry Pi 3 is described as 10 times faster than the Raspberry Pi 1 due to its quad-core ARM Cortex-A53 processor. Also, the Raspberry Pi 3 comes with a 2.4 GHz Wi-Fi 802.11n (150Mbit/s) and Bluetooth 4.1 (24Mbit/s). The operating system recommended for use on a Raspberry Pi is a Raspbian, Debian-based Linux operating system though other operating systems can also run on it. [21]

Figure 16:Image of a Raspberry Pi 3

3.1.5.2 RASPBERRY PI CAMERA

The Raspberry Pi camera is a camera module compatible with the Raspberry Pi. The camera module v2, which was released in 2016 has a still resolution of eight (8) megapixels, with video modes of 1080p30, 720p60 and 640 x 480p60/90. It uses the V4L2 driver available for Linux integration and uses the Sony IMX219 sensors with sensor resolution of 3280 x 2464 pixels. [22]

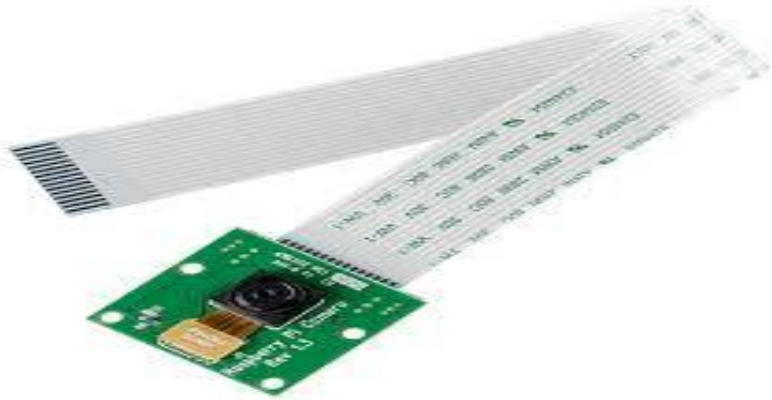

Figure 17:: Image of Raspberry Pi Camera

3.1.5.3 WI-FI DONGLE

The Wi-Fi dongle is a device compatible with the Raspberry Pi that enables internet connectivity to the Raspberry Pi. It has the IEEE 802.11n wireless standards with a high-speed USB 2.0/1.1 host interface. It has up to 150Mbps (downlink) and also up to 150Mbps (uplink). It has a frequency band of 2.4GHz Industrial Scientific Medical (ISM) Band and a chipset produced by Realtek. Its antenna is integrated. It has a 64/128-bit WEP encryption and supports Ad-Hoc, Infrastructure WLAN network and wireless roaming.[23]

Figure 18: Image of a Wi-Fi Dongle

3.1.6 SOFTWARE SPECIFICATIONS

3.1.6.1 INTRODUCTION

In designing the system to detect vehicles, pedestrians, road accidents and bad roads, TensorFlow Object Detection Application Programming Interface was used. Due to the critical and hard real-time nature of the system to be developed, a lot of factors were taken into consideration — for instance:

- The time taken to perform successful detections given an image frame.
- The level or percentage of accuracy of the system
- The overall performance of the system and its up-time.

These factors were necessary because the system to be built had to be very fast in its object detection (especially with vehicle and pedestrian detection) and it should be able to accurately detect as many vehicles as possible in a given image frame. Also, the confidence level of the system was taken into serious consideration since it was necessary to ensure that all objects detected were actually true positives. Finally, the overall performance of the system and how long it could be up and running was also a major concern that was taken into consideration. My decision to opt for the TensorFlow Object Detection Application Programming Interface was informed after carefully analysing all the factors stated above. This is because the TensorFlow Object Detection Application Programming Interface provided state-of-the-art models, some which were specifically built for speed and others specially designed for accuracy.

3.1.6.2 OBJECT DETECTION

Ever since it became a possibility to detect objects and also classify objects in images, there has been countless use of object detection in various applications worldwide. The discovery of object detection has enabled us to be able to build applications which could detect intruders in our homes in our absence, detect number plates of vehicles which do not obey traffic rules on the road and even applications which are able to detect specific types of homes in home ads without human intervention. All these numerous applications are built on mathematical foundations which make their operations possible. The most popular technique which has been used in object detection for a very long time is the Convolutional Neural Network, which is also the underlying technique of the TensorFlow Object Detection Application Programming Interface.

3.1.6.3 CONVOLUTIONAL NEURAL NETWORK

Over the recent years, a lot of researchers have developed or propounded various theories that have tremendously improved the area of image classifications in computer visions. Many theories have been proposed over the years but the most effective way so far have been the usage of Convolutional Neural Network (ConvNet). Convolutional Neural Networks are a category of Neural Networks that are made up of neurons that have learnable weights and biases where each neuron receives some inputs, perform dot operation and optionally follow it with a non-linearity.

3.1.6.3.1 ARCHITECTURE OF THE CONVOLUTIONAL NEURAL NETWORK

There are various architectures of ConvNet. Some of these are AlexNet (2012), ZF Net (2013), GoogLeNet (2014), VGGNet (2014), ResNets (2015), DenseNet (2016) and LeNet architectures but most of these architectures have their foundation similar to the LeNet architecture. The LeNet architecture is briefly explained below.

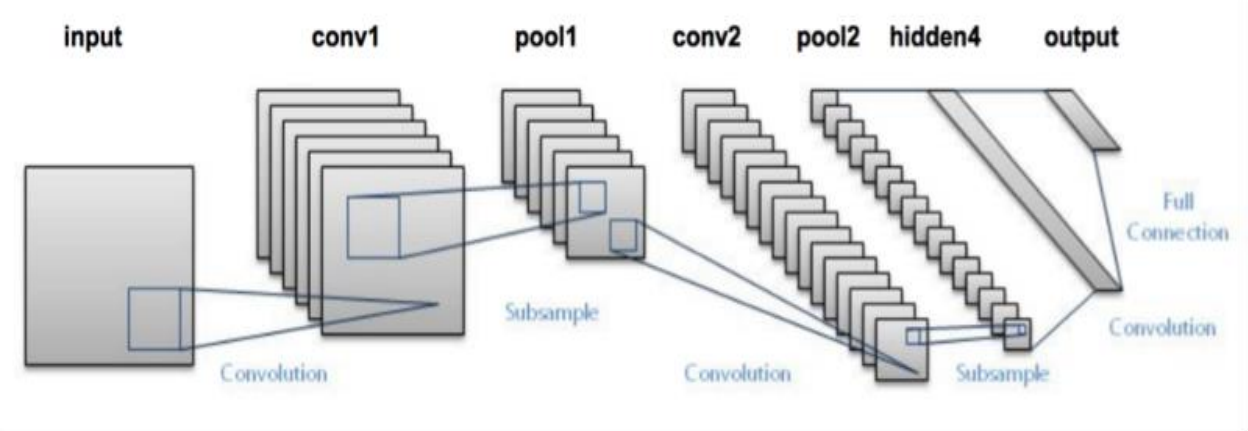

Figure 19: Image of ConvNet Architecture. [12]

3.1.6.3.2 LeNet ARCHITECTURE

Being one of the very first CNN architectures to propel the field of Deep Learning, the LeNet Architecture is made up of four main operations which also found themselves resounding in almost all the other architectures available. The LeNet architecture was pioneered by Yann LeCunn was named LeNet5 after many previous successful iterations since the year 1998.[24] At that time, the LeNet architecture was used mainly for character recognition tasks such as reading zip codes, digits, etc. The four main operations performed by this architecture include:

- Convolution
- Non-Linearity (ReLU)
- Pooling or Sub sampling
- Classification (Fully connected layers)

3.1.6.3.3 CONVOLUTION STEP

The purpose of convolution in CNN is to extract features from the input image.[24] Every neuron takes input from a rectangular n x n section of the previous layer. The rectangular section is called the local receptive field. Convolution preserves the spatial relationship between the pixels by learning image features using the local receptive field. An activation map or Feature map is obtained when the local receptive field, also known as the filter or kernel is convoluted with the input image.

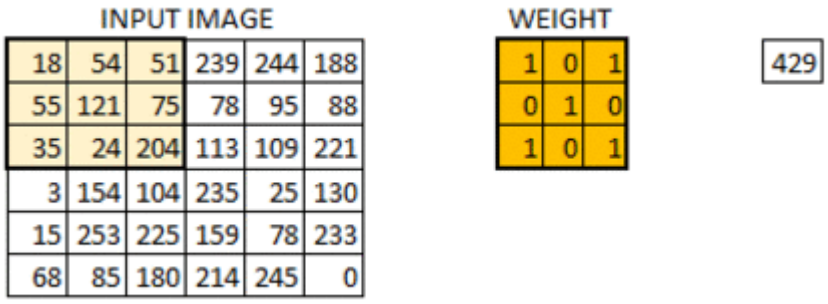

Figure 20: Convolution of an Image matrix and a weight matrix

3.1.6.3.4 RECTIFIED LINEAR UNIT STEP

This is a non-linear operation which is performed during CNN in order to replace all negative pixel values in the feature map by zero [referencing here]. The actual purpose of this step in CNN is to introduce a reasonable degree of non-linearity. This is important because most of the real-world data we would want our CNN to learn would be in a non-linear form. Also, Deep Convolutional Neural Networks with ReLUs train several times faster than their equivalent tanh units.

3.1.6.3.5 POOLING

Down Sampling (another term for pooling) is another very important step in CNN. This is because it is used to reduce the dimensionality of each feature map while retaining the most important information. Pooling can be of different types such as Max, Average, Sum, etc. Max Pooling is achieved by defining a spatial neighbourhood and taking the largest element from the rectified feature map within that window. In Pooling, Max pooling is the technique that has been proved to be the most effective so far by researchers.

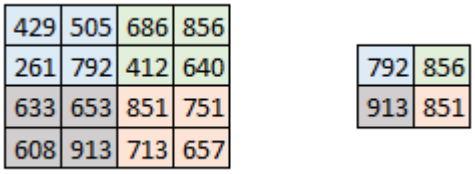

Figure 21: Image of Max Pooling

3.1.6.3.6 CLASSIFICATION

This layer uses the SoftMax activation function in the output layer. It is a traditional Multilayer Perceptron. The output from the convolutional and pooling layers represent high-level features of the input image [24] and the whole purpose of this function is to classify the input image

into various classes based on the training dataset. Also, in ConvNets, just like in every neural network has each neuron in a layer is connected to all the numbers in the previous volume.

The TensorFlow Object Detection Application Programming Interface is an open source software library, made available to the programming community by Google and is used to perform object detection and classification by either using a pre-trained model or deciding to train a model from scratch to perform the same task. In the TensorFlow Object Detection Application Programming Interface, the decision was made to select a pre-trained model and further train it to be able to perform the vehicle, pedestrian, bad road and road accident detection. This decision was made because all the pre-trained model have been specially trained with state-of-the-art computers and for over a long period of time with very huge dataset therefore it would be literally impossible to achieve the same feat using a personal CPU computer. Also, a lot of time will be saved by selecting a pre-trained model and training it to perform further tasks. In choosing a pre-trained model to be used in building the system, the SSD Inception v2 model was selected.

3.1.6.4 SINGLE SHOT MULTIBOX DETECTOR

The Single Shot Multibox Detector Inception v2 model is a pre-trained model which was trained by validating its dataset against ImageNet, a highly-recognized standard for computer vision models. Single Shot Detector (SSD) simple refers to architectures that use a single feedforward convolutional network to directly predict classes and anchor offsets without requiring a second stage per proposal classification operation.[25]. The approach used by the SSD is that, it discretizes the output space bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. During prediction in SSD, scores for the presence of every single object category in each default box are generated by the network and adjustments are made to the box to better match the object shape.[26]

The SSD framework functions in a very simple manner. Only the input image and the ground truth boxes for each object are needed during training. Then, by performing convolution, a small set of default boxes of different aspect ratios at each location in several feature maps with different scales are evaluated (example is the 8 x 8 and 4 x 4 in Fig 15). Afterwards, for each default box, prediction is made for both the shape offsets and the confidences for all object categories. Most importantly, during training, the default boxes are matched to the ground truth boxes and the model loss is the weighted sum between the localization loss and confidence loss.[26]

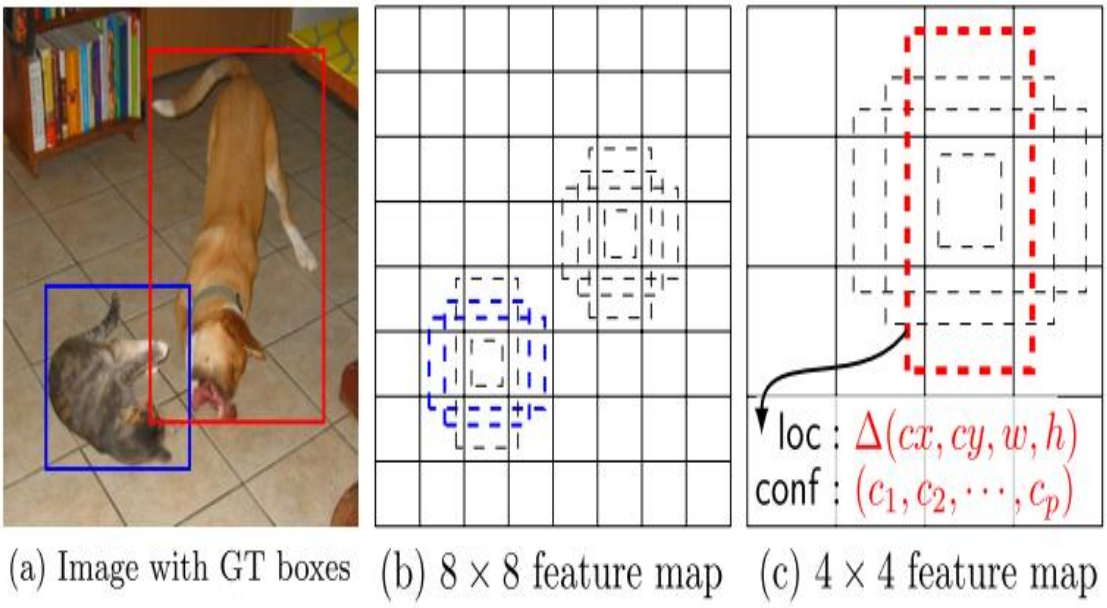

Figure 22: SSD Framework [15]

The SSD model is based on the feed-forward convolutional neural network. This approach produces a fixed-size collection of bounding boxes and scores for the presence of object class instance in those boxes. This is followed by a non-maximum suppression step to produce final detections. [13] SSD's Multi-scale feature maps for detection makes use of the addition of convolutional feature layers to the end of the truncated base network. These layers reduce in size as more convolutional feature layers are added to the end and this enable predictions of detections at multiple scales.

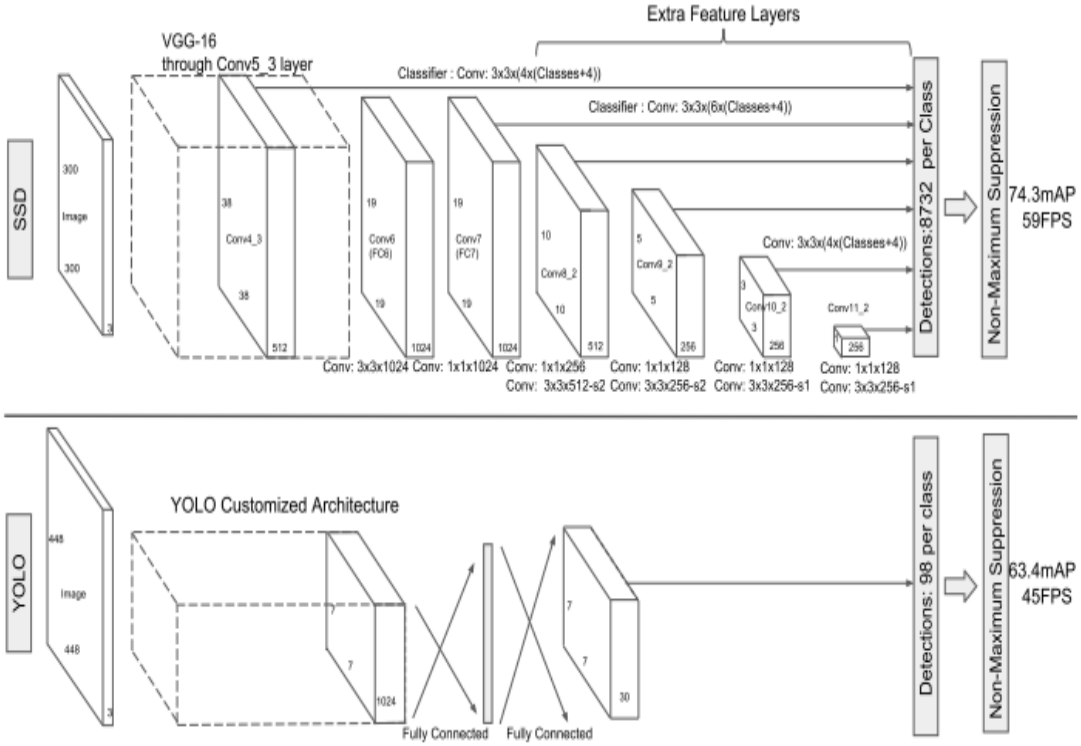

Figure 23: Comparison between SSD and You Only Look Once (YOLO) Architecture [15]

In the comparison in Figure 16, it is clearly noted that SSD models adds several feature layers to the end of the base network which predict the offset to default boxes. SSD with a 300 x 300 input size outperforms YOLO with an input size of 448 x 448 in accuracy on VOC2007 test while also improving its speed. [13]

In the Inception deep learning architecture, the linear convolutions which are the core of ConvNets are gotten rid of and instead, connects the convolutional layers through multi-layer perceptrons that can learn non-linear functions. These perceptrons are mathematically equivalent to 1x1 convolutions and hence fit neatly within the ConvNet framework.

Advantages of the Inception deep learning architecture

The increased abstraction power of the convolutional layers which can lessen the need for fully connected layers at the top of the network is one major achievement of the Inception architecture. Also, the absence of the fully connected layers cuts the number of parameters, and thus reduces the risk of overfitting and computational load. This also improves robustness to spatial translation. [27]

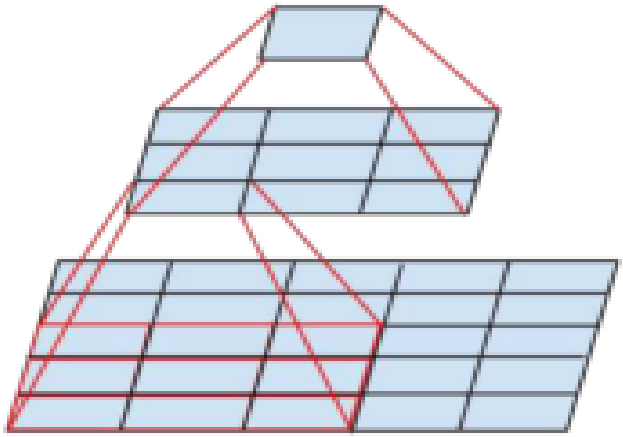

Figure 24: Mini-network replacing the 5x5 convolution [16]

Any convolution whose kernel is larger than 3x3 can be expressed more sufficiently with a series of smaller convolutions. Convolutions with larger spatial filters (e.g. 5x5 or 7x7) tends to be disproportionally expensive in terms of computation. Supporting the statement by an example, a 5x5 convolution with n filters over a grid with m filters is $\frac{25}{9}$ = 2.78 times more computationally expensive than a 3x3 convolution with the same number of filters. This is known as factorization into smaller convolutions. [28]

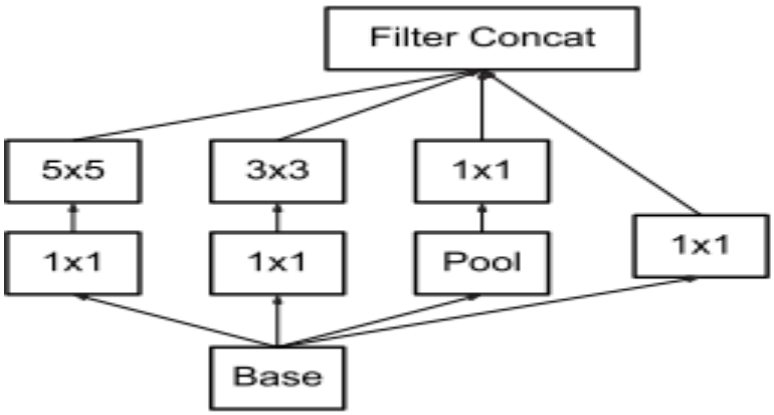

Figure 25:Original Inception module [17]

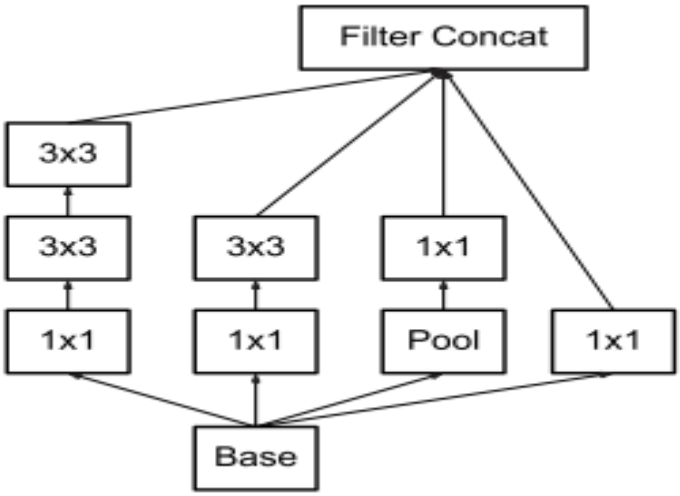

Figure 26: Inception module where each 5x5 convolution is replaced with two 3x3 convolution [17]

3.1.7 SYSTEM DESIGN

In order to build the model to detect vehicles, pedestrians, road accidents and bad roads, a lot of data (images) of different types of vehicles, pedestrians crossing the road, road accidents and bad roads were collected from google.

Figure 27: Sample Image of road accident collected from Google

Figure 28:Sample Image of bad road collected from Google

Figure 29:Sample Image of pedestrians collected from Google

Figure 30: Sample Image of vehicle collected from Google

The downloaded images were then pre-processed and separated into training and testing directories. All inappropriate images were deleted. Information about these images were needed in xml format and therefore in order to obtain these data, a tool known as Labelling was used. All the images were manually opened one after the other in the tool and the area of interest was cropped out using a feature in the tool. This generated the corresponding xml version of the image. This procedure had to be repeated for all other images (approximately 1500). Since the required input format of the TensorFlow Object Detection Application Programming Interface is tfrecord, the images in both the training and testing directories were needed to be converted to csv formats. To achieve this, a python script was written to convert all the images

into csv files. Another script was then written in python to convert all the csv files into the required tfrecord format.

Needed system configuration settings were done for the TensorFlow Object Detection Application Programming Interface and the new model was trained. This took about two weeks to achieve a loss of about 2. The cause for the long delay in training is due to the fact that the model was trained using a CPU computer since there was no GPU computer available to be used. After a successful training of the model, it was tested against the test images in the test directory and inference graph was generated out of the trained model and exported to be used for detection in the system.

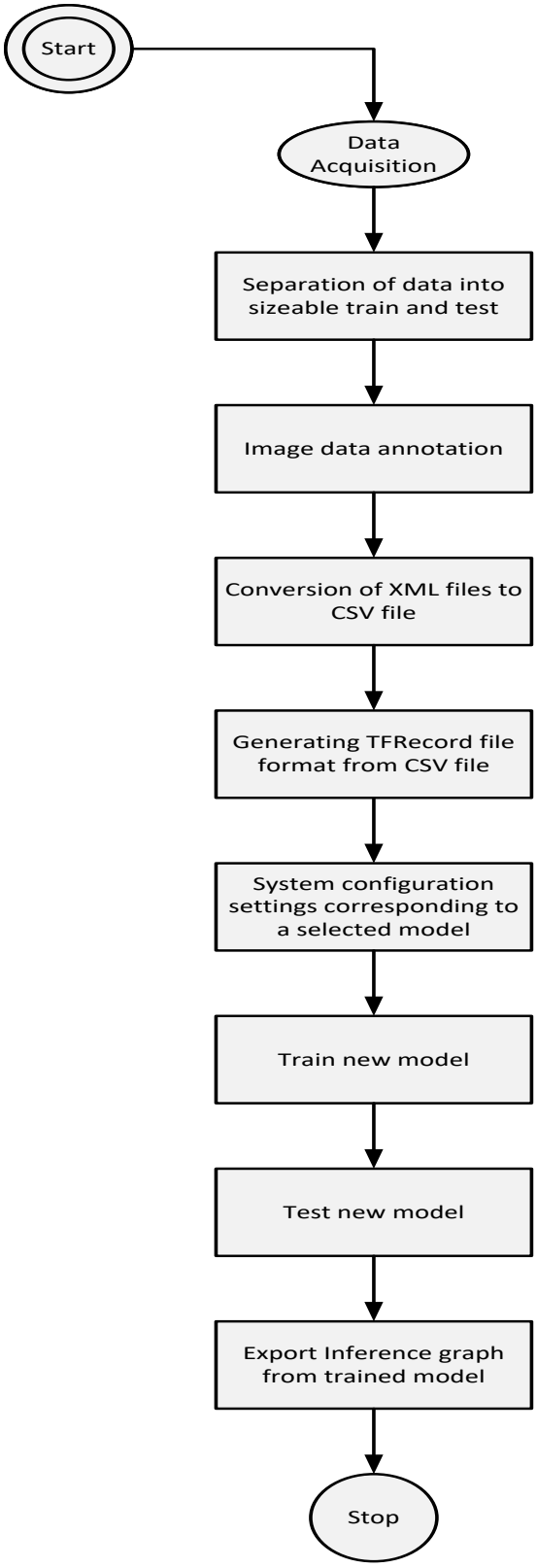

Figure 31: Flow diagram of the model implementation

With the model ready for use, the program to utilize the model was built. The language used for implementation was the Python programming language. The operations of the system are as follows:

- The system receives an event emitted by Media Server to enable video stream from cameras.
- Video data is then acquired either by live video stream via web sockets or by utilizing pre-recorded video streams.
- The image is the processed and prepared to be fed to the trained model.
- Based on the type of model requested by the Media Server in its message during the event emission, objects detections are made correspondingly. Therefore, if the message requires that only anomaly model is used, then only anomalies in the video stream are detected similarly, if the message requires that the traffic model is used, then only vehicles and pedestrians in the video stream are detected.
- During this process, all objects being detected are counted and the data obtained is communicated to the Traffic Simulator and Scheduling System via web API.
- Also, whenever an anomaly is detected, the particular frame containing this anomaly is captured as an image, converted to a base64 encoded string and together with the GPS coordinates of the location where this video feed was received from and the timestamp, transmitted to the Android Application System via web API as well.
- Due to the fact that multiple video streams from various cameras will be needed to be processed and visually analysed simultaneously, multithreading was implemented in the system to effectively handle concurrency and display feeds from various cameras at the same time.

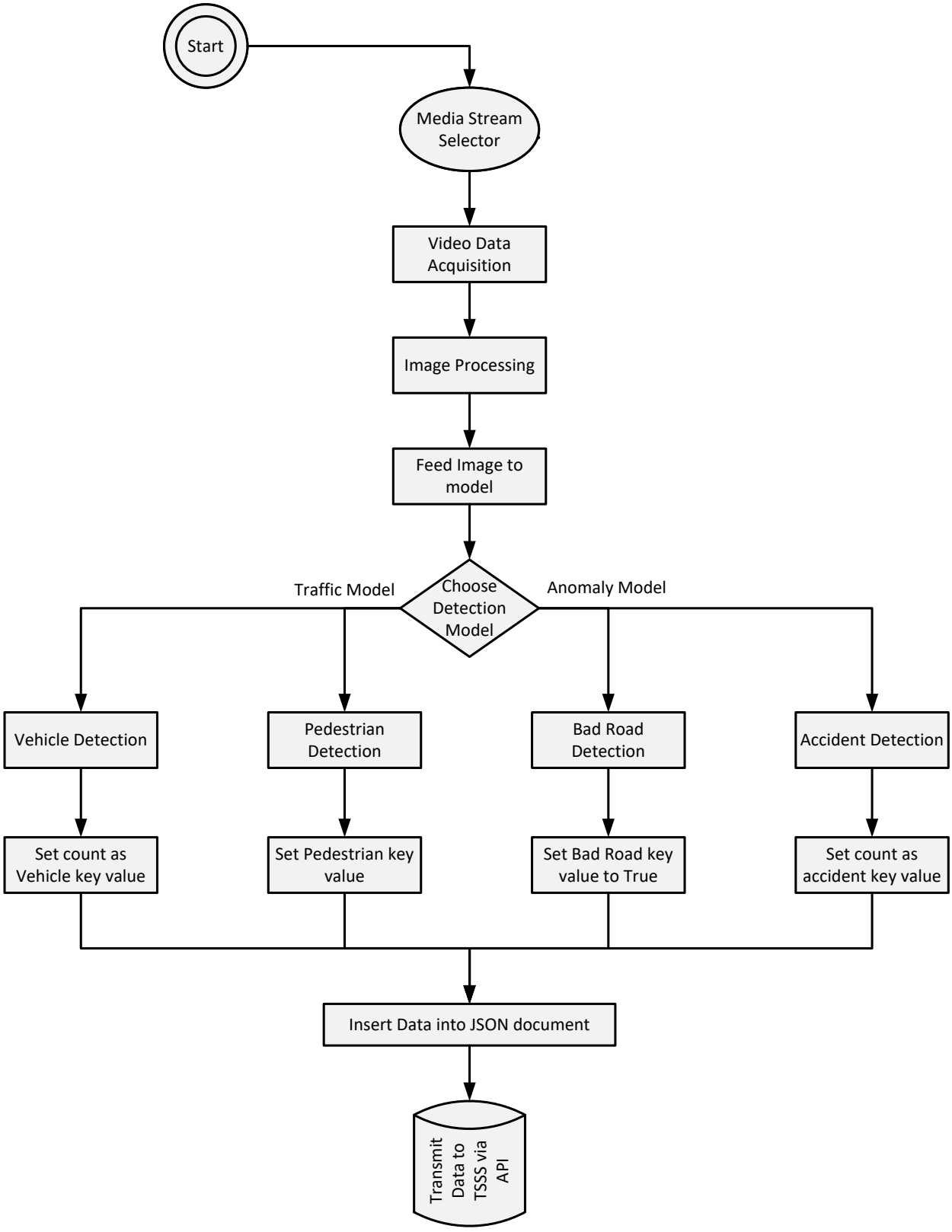

Figure 32:Flowchart of the operation of the Vehicle, Pedestrian, bad road and road accident detection system

3.2.0 NETWORKED TRAFFIC CONTROL SYSTEM

3.2.1 INTRODUCTION

The networked traffic control system provides road traffic information for road links within the scope of the project. The system has a road network map stored in its database for providing the traffic information. A typical information the system provides is traffic densities of various stretch of roads lying within the scope of the project. The system adopts a microscopic approach where individual vehicles and the behaviour of drivers are taken into consideration during the modelling stage. Also, the system implements control algorithms to dynamically direct traffic with the aim of minimizing the overall delay spent in a network.

3.2.2 SYSTEM REQUIREMENTS

The system should be able to:

- Retrieve vehicular information from the central server periodically.
- Implement a simple traffic model to analyse traffic.
- Compute traffic densities of links and intersections.
- Dynamically control traffic lights at various intersections.
- Report analysed traffic information to the central server.
- Visualize traffic flow.

3.2.3 SYSTEM DESIGN

- Feed road network information to the traffic visualization library.
- Retrieve vehicular traffic data from the central server periodically for traffic analysis.
- Model a simple traffic system using a microscopic approach. The microscopic approach deals with individual vehicles and the behaviour of individual drivers on the road.
- Feed information to traffic visualization and analysis software for visualization.
- Compute traffic densities for various links and intersections.
- Send analysed traffic information to the central server through a public API.
- Implement control algorithms to dynamically control traffic at different intersections based on traffic volume.
- Simulate traffic flow with traffic information from the central server and randomly generated vehicular data.
- Control the states of different traffic signal groups in the network concurrently.
- Feed results to the visualization system to be visualized.

3.2.4 TRAFFIC CONTROL SYSTEM FLOW DIAGRAM

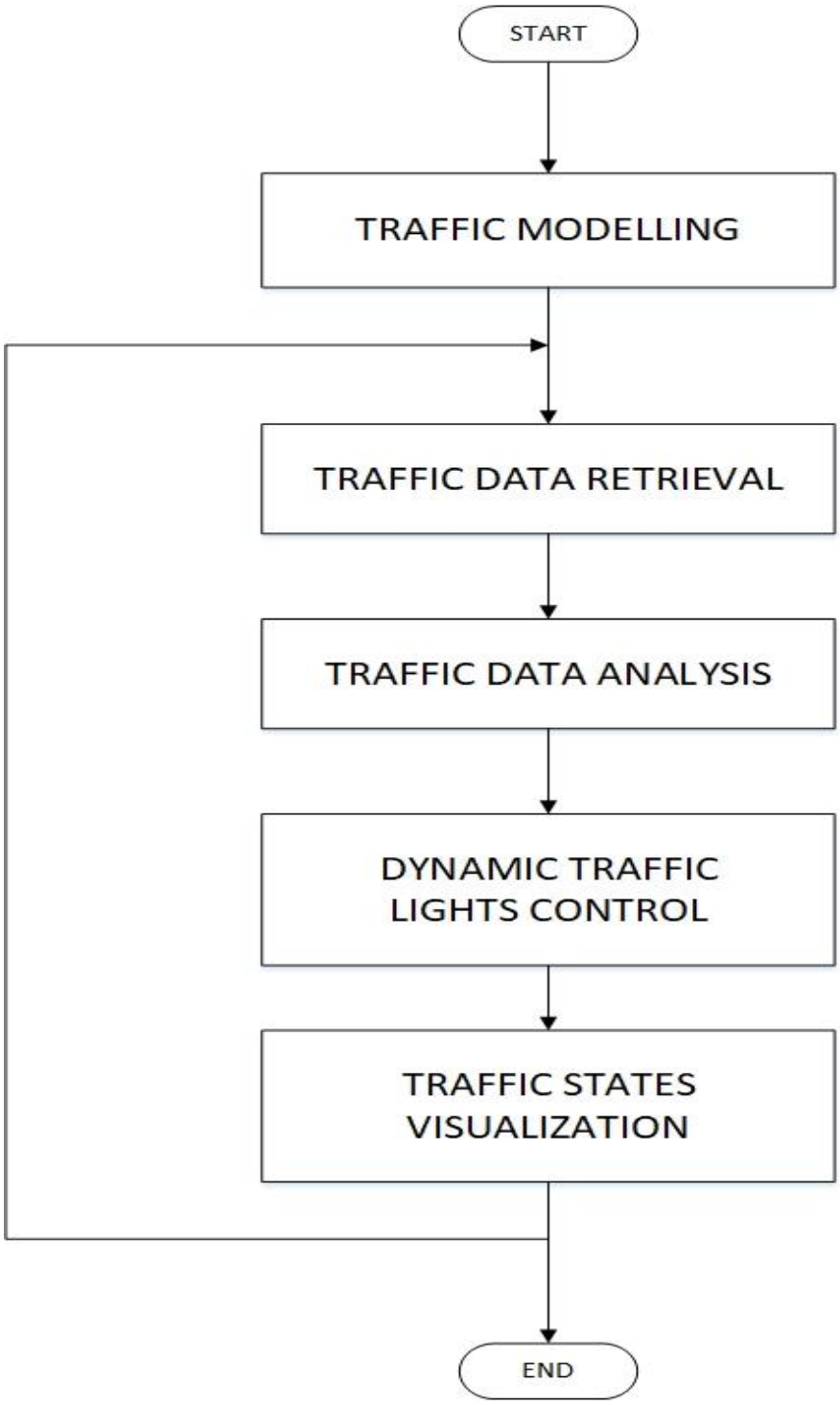

Figure 33: Flowchart for traffic control system

3.2.5 SERVER COMMUNICATION

The central server houses information to be shared across all subsystems. Vehicles, pedestrians and detected anomalies are submitted to the central server by the detection system. A request

is sent to central server to retrieve the vehicular data of a particular link or several links. The central server responds with a JSON formatted response which contains the information. All communications with the central server is through an application programming interface(API) either on the same network or a different network. Other subsystems that communicate with the central server are vehicle detection system and routing system. Communication between the traffic control system and the simulation client should be established to enable updating traffic simulation as when data comes through. Communication between client traffic control program and the central traffic server is through a socket implementation. A socket server is designed and built on top of the central traffic server to enable real-time communication. The central traffic server sends information like traffic density and road network information to the routing and navigation system via an API endpoint.

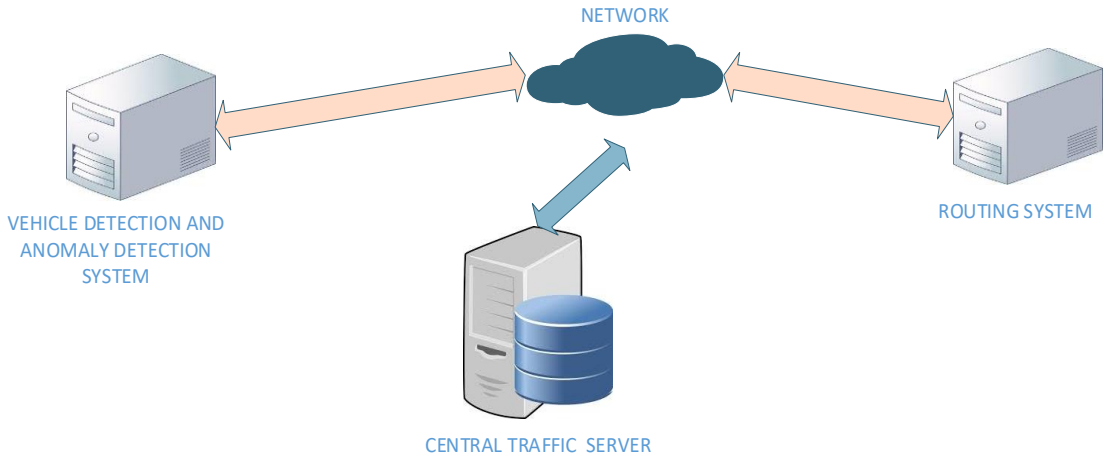

Figure 34: Server communication architecture

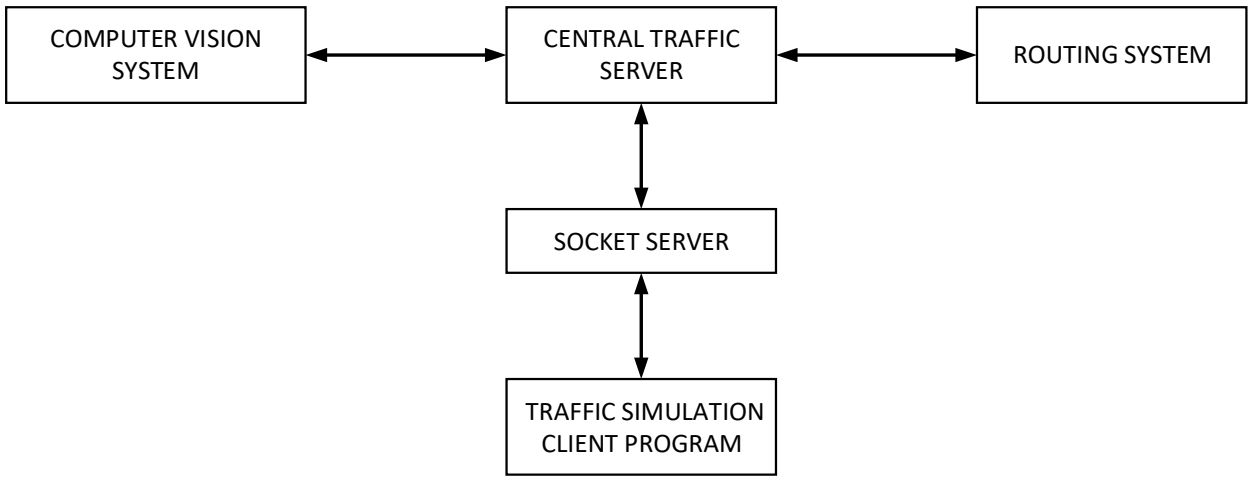

Figure 35: Block diagram representing server communication

3.2.6.0 ROAD NETWORK

3.2.6.1 INTRODUCTION

The system contains road network map used for determining the locations of entities on the road. Some of these entities include vehicles, intersections and traffic lights. The road uses the coordinate system where entities have latitudes and longitudes. This will help pinpoint the exact location of entities in the network. For simplicity sake, the road network comprises of a multitude of nodes also referred to as points or junctions, several links or edges or roads linking these nodes, intersections and traffic lights at intersections. A link or a road joins at least two nodes. Roads can be unidirectional or bidirectional. Also, roads can have a single lane or multiple lanes, speed ramps etc. Roads join other roads leading to a network of roads on the map. The road network information will be stored in a MYSQL database on both the traffic control subsystem and the central server. The road network to be used is extracted from Openstreet map. The road network data is prepared using Java OpenStreetMap Editor and exported to a format suitable for the traffic simulation software SUMO [18].

3.2.6.2 ROAD NETWORK ARCHITECTURE

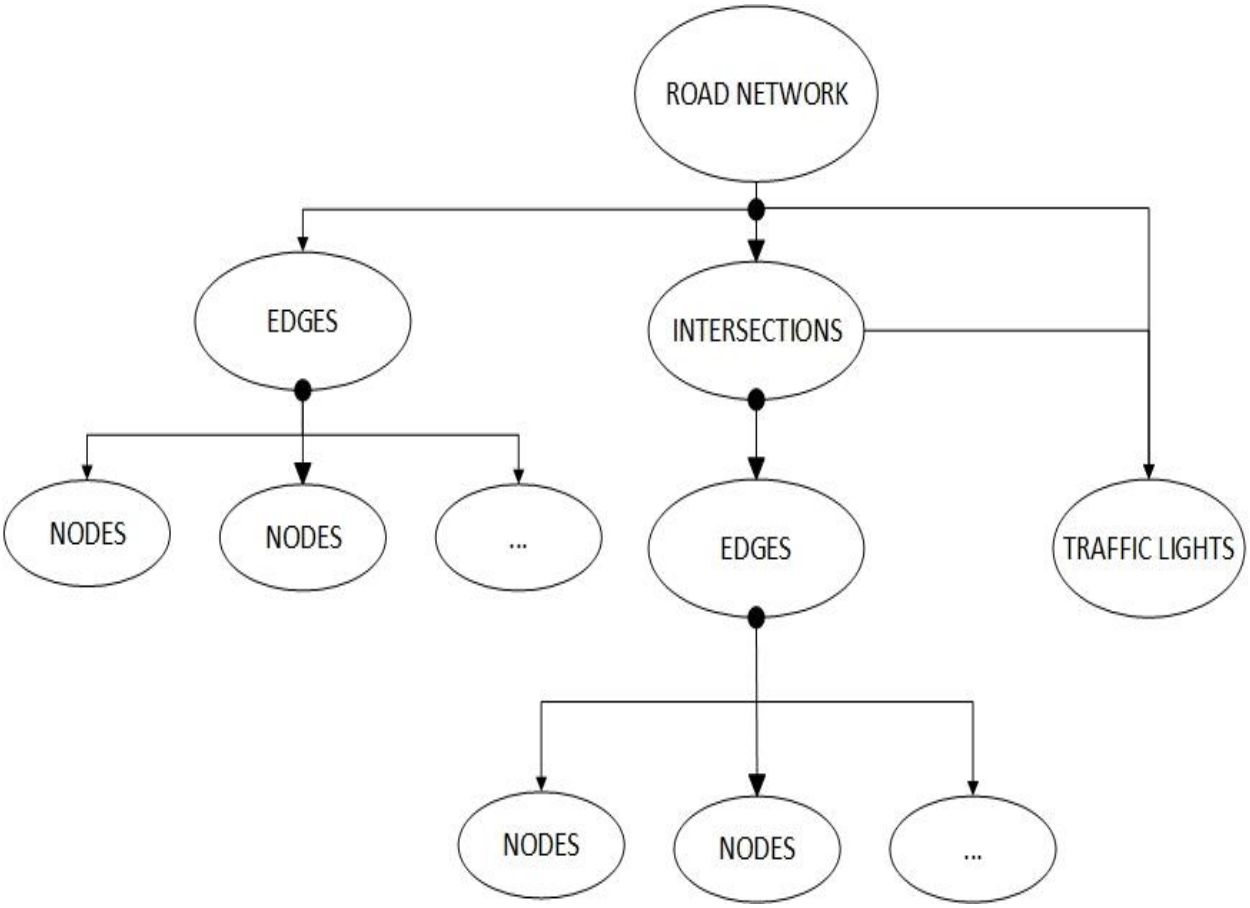

Figure 36: Architecture for road network

3.2.6.3 ROAD NETWORK DATABASE SCHEMA

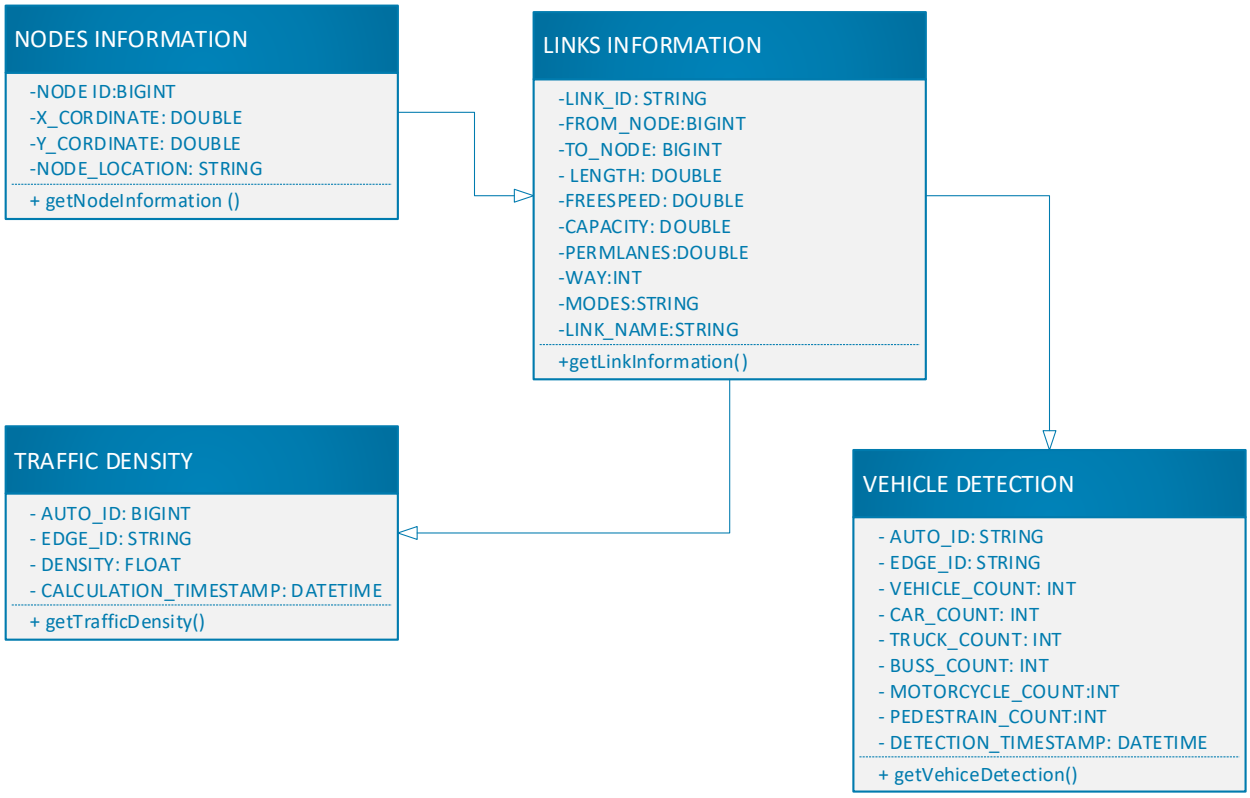

Figure 37: Database schema for road network

3.2.7 TRAFFIC NODES

Traffic network encompasses the Legon vicinity including the borders of Hatso-Atomic road, GIMPA road, Dworwulu round-about road, Shiashie, East Legon, UPSA and Madina as shown on the map.

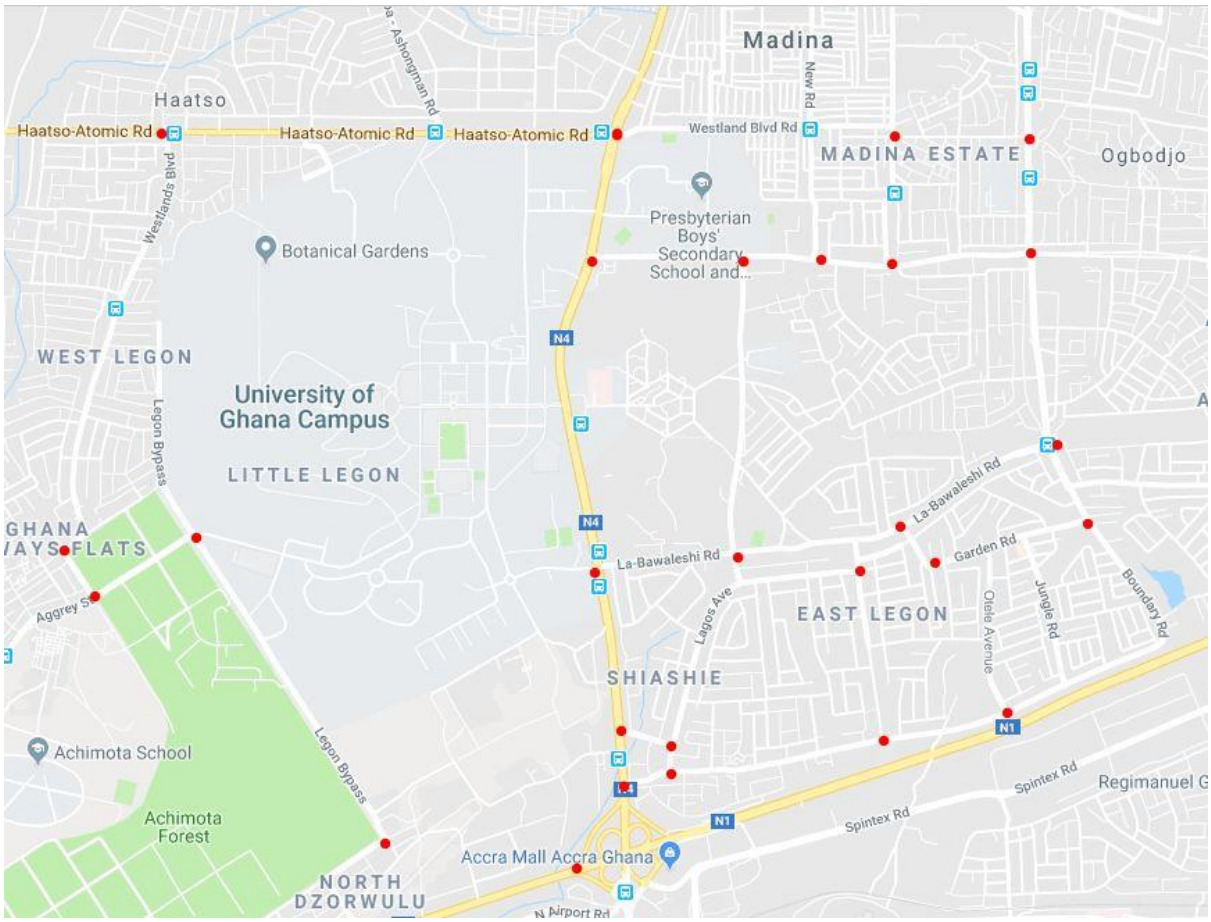

Figure 38: Google Map view of Legon district

3.2.8 TRAFFIC VISUALIZATION LIBRARY

SUMO (Simulator of Urban Mobility), is a well-known traffic simulator that provides an open source, highly portable, and microscopic road traffic simulation tool designed to handle large road scenarios. SUMO requires several input files that contain information about the traffic and the streets to be simulated. A network (*.net.xml* file) holds the information about the structure of the map: nodes, edges, and connections between them. The network can be imported from popular digital maps such as OpenStreetMap [19] and converted to a valid SUMO network by means of a series of scripts provided in the SUMO package. We have chosen OpenStreetMap (OSM) because it provides both, geographic data and traffic light information.

3.2.8.1 FLOWCHART FOR SUMO

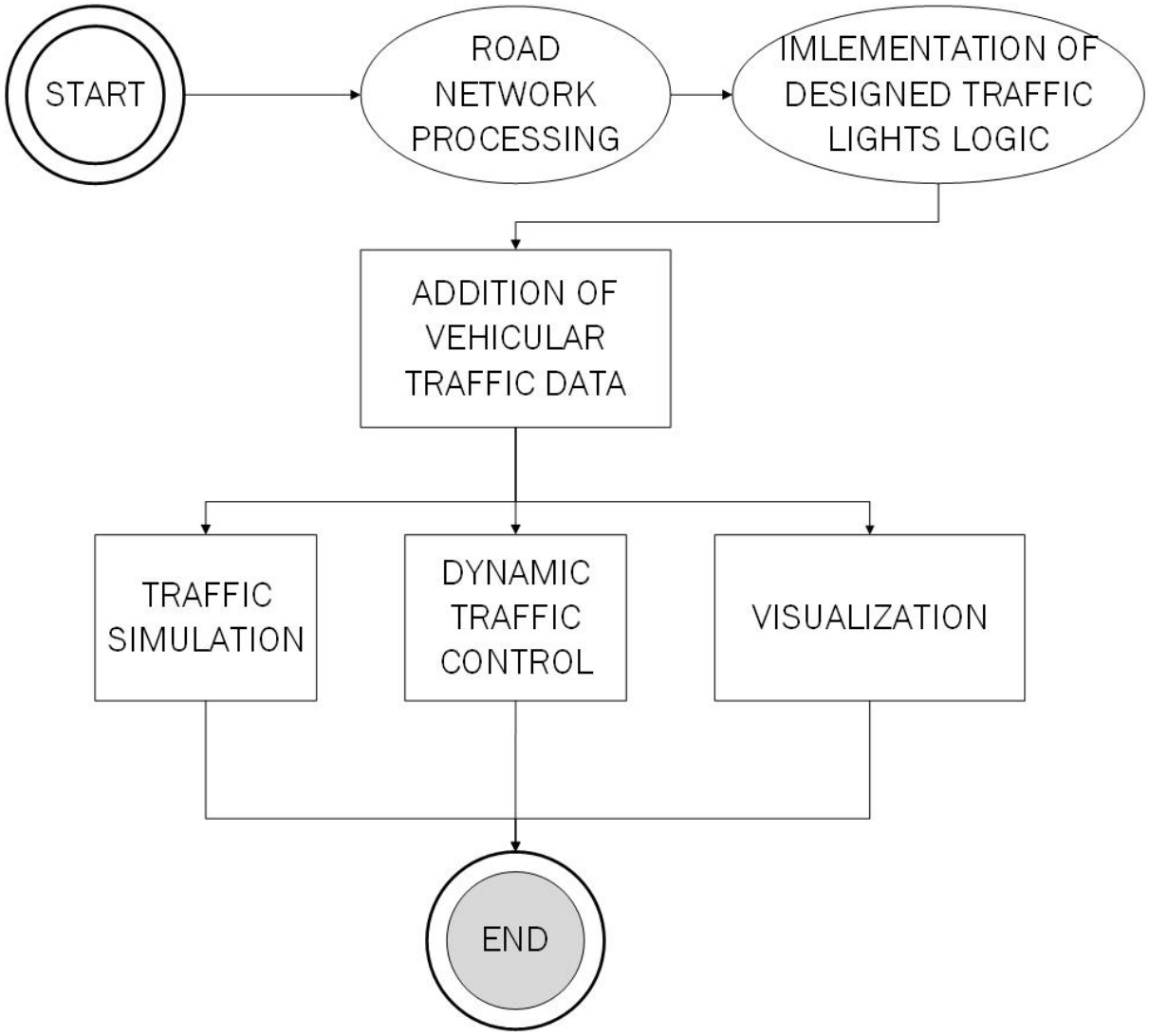

Figure 39: Flowchart for SUMO [18]

3.2.8.2 ROAD NETWORK FOR SUMO

Figure 40: SUMO capture view of Legon district

3.2.8.2 ROAD NETWORK WITH TRAFFIC LIGHTS

Traffic Lights are mapped to junctions in the network: Okponglo, UPSA, Shiashie, Dworwulu, American House, Spintex Road and some parts of east Legon.

Figure 41: SUMO capture view of Legon district with traffic lights

3.3 ROUTE NAVIGATION AND ALERTING SYSTEM

3.3.1 SUBSYSTEM DIAGRAM

The system architecture for this project is displayed below in Figure 5. The whole system is implemented in software only.

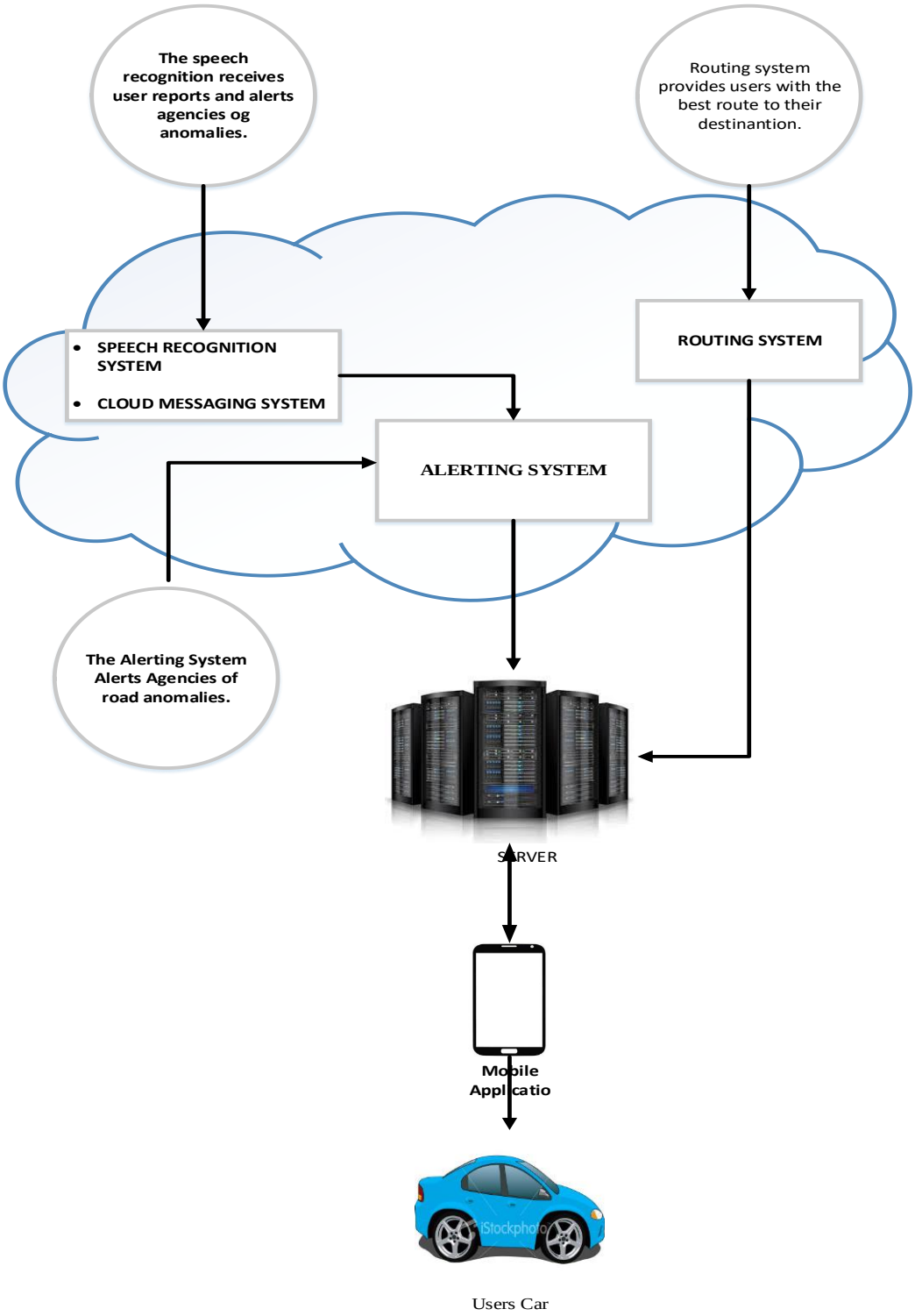

Figure 42: Route Navigation and Alerting Subsystem Architecture

From the diagram above it can be observed that the project consists of these individual systems:

i. A speech recognition system designed to read in user speech alerts from the mobile application.
ii. A Routing System for selecting optimal routes for road commuters.
iii. An Alerting system for notifying authorities of any road anomalies detected by either a road commuter or by the Computer Vision System through social media and electronic mail.
iv. A Cloud Messaging platform that uses Text-to-Speech to read out user updates on anomalies.

All systems are mainly implemented on a server that uses the HTTP protocol, however these systems can be triggered from the mobile application by the users. The mobile application communicates with the server through various APIs developed on the server. The various subsystems will be explained below.

3.3.2 ALERTING SYSTEM

The main use of the Alerting System is to send alerts to agencies in real-time. These alerts are either generated by the Computer Vision system through images or by road commuters through the application. In the mobile application, the road commuter has numerous means by which the Alerting System can be triggered. When the system is triggered, it performs various pre-processing tasks and then reports the anomalies to a group of registered agencies. Since this is to be done in real-time the Alerting System implements a host of APIs for real-time, asynchronous access and computation. As part of the Alert System, the user is given the ability to call the nearest police station to the user's current location.

To be able to notify agencies through the various channels (Twitter and E-Mail), the system needed to access these channels via third party applications and therefore access tokens needed to be generated for access.

Below is an image of the system flow chart of the Alerting System. This shows the sequence of events that take place in the process of alerting. It shows the process that are undergone for each type of alert request. An alert request is simply a call by either user or Vision System to make an alert.

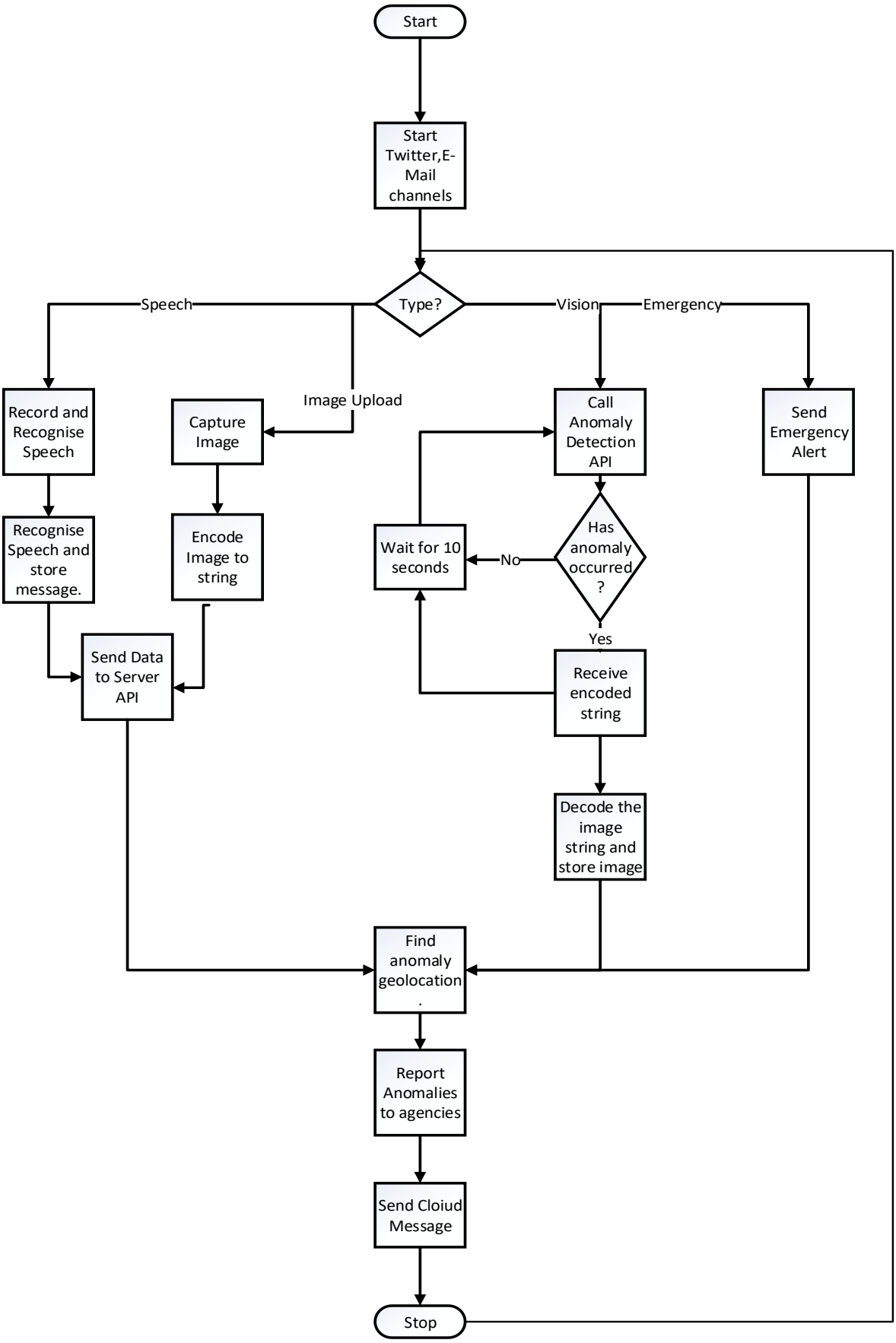

Figure 43: Flow Chart of Alerting System

3.3.3 ROUTING SYSTEM

The main purpose of the routing system is to aid users in making correct decision during route planning. The system is mainly built on a server and interfaced with the mobile application through various APIs communicating using the HTTP protocol. The application is used to display routes and guide the road commutes as to what way to go while driving.

Below is a flow chart showing the sequence of processes undergone in the process of finding optimal routes for users.

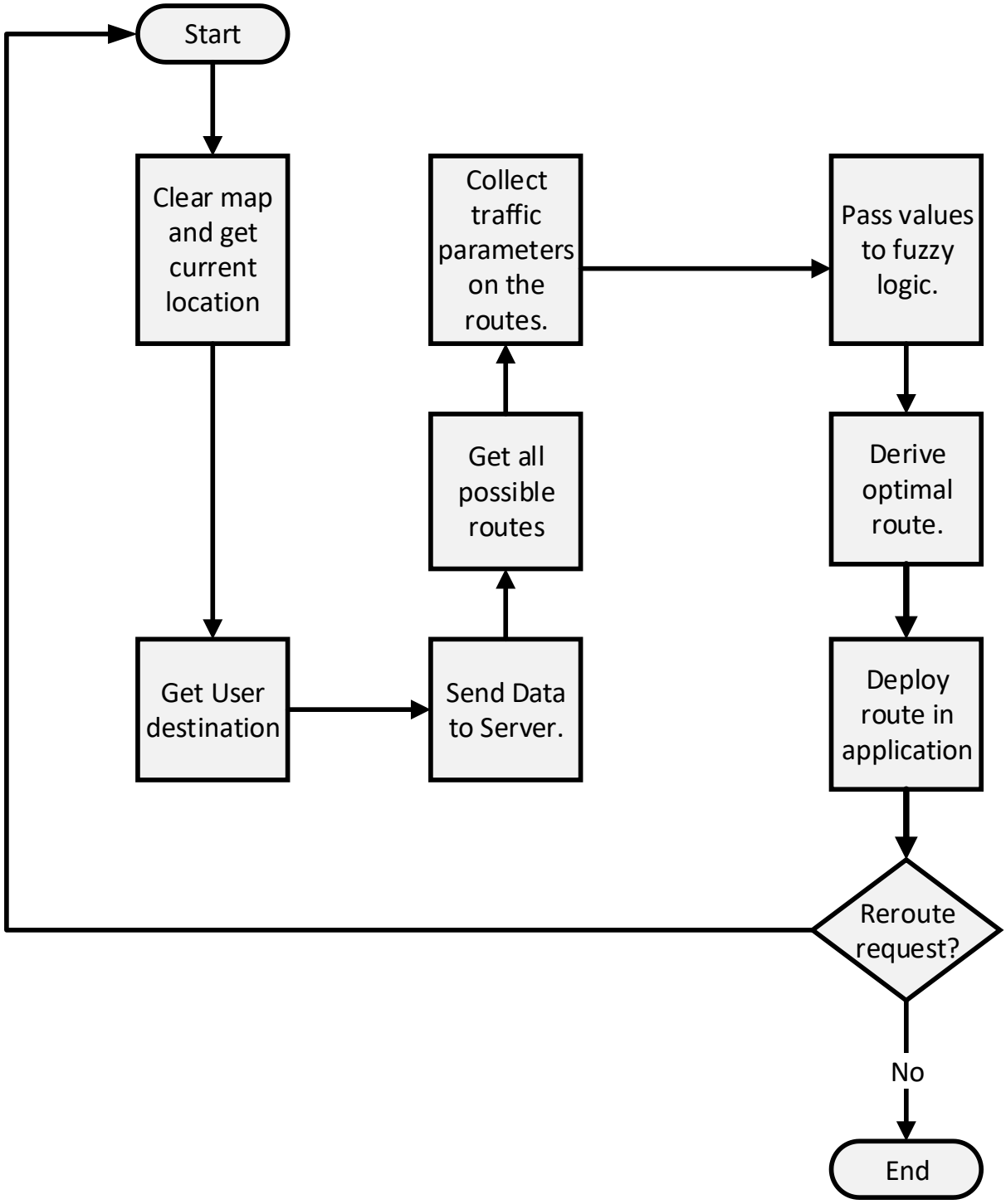

Figure 44 Flow Chart of Routing System

All other subsystems are systems to aid the activities of these two major systems. The speech recognition system is one way of feeding the Alerting System with information as shown in

the Alerting System flow diagram. The Cloud Messaging platform is used by the Alerting System to notify road commuters of reported anomalies.

3.3.4 SYSTEM REQUIREMENTS

3.3.4.11 FUNCTIONAL REQUIREMENTS

7. The system should be able to serve the user with optimized routes
8. The system should be able to select a new route for user upon re-route request.
9. The system should be able to notify users of road anomalies in real-time using Text-To-Speech and the Cloud Messaging Platform.
10. The system should be able to record audios and recognize the speech in them for reporting.

3.3.4.2 NON-FUNCTIONAL REQUIREMENTS

3. The alerting of authorities should be done in real-time
4. All API calls to the server should be done with a verification key in the header.
5. All alerts should be done asynchronously to reduce latency and increase throughput.

3.4 STATIC & MOBILE CAMERA SUBSYSTEM

3.4.1 FUNCTIONAL REQUIREMENTS:

- A camera system to capture videos and stream back to server and application.
- Control a quadcopter via a ground control application to patrol between certain locations and its home location.
- Stream video data, vehicle state and sensor readings back to ground control app from the quadcopter on board camera and sensor systems.
- Trigger detection server to start and stop detection when mission starts and stops respectively.

3.4.2 NON-FUNCTIONAL REQUIREMENTS:

- There should be low latency between communication of subsystems.
- Ground control application should have easy usability.

3.4.3 EQUIPMENT

Figure 45: APM 2.6 Flight Controller

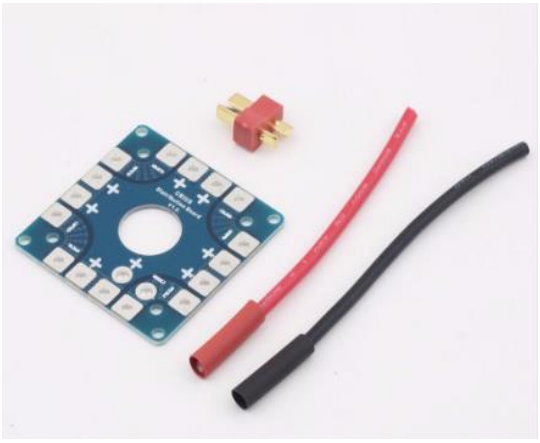

Figure 46: Power Distribution Board

Figure 47: 4 30A ESC, 4 1000KV Motors & 4 Propellers (2 Pusher Propellers)

Figure 48: 2200mAh 11.1V Battery

Figure 49: 5V 2.1A Power Bank

Figure 50: 3DR Power Module

Figure 51: FlySky FS-i6X Digital Proportional Radio Control System and a FlySky FS-iA6B 2.4GHz 6 Channel Receiver.

Figure 52: Arduino UNO

Figure 53: Raspberry Pi 3 Model B +

Figure 54: DHT-11 Temperature & Humidity Sensor

Figure 55: Ublox 7M GPS Module

Figure 56: 450mm Quadcopter Frame

Figure 57: Zip Ties

Figure 58: Bullet Connectors

Figure 59: Electrical Tape

3.4.4 RASPBERRYPI SETUP:

A Raspberry Pi was setup and configured by the following steps:

- Raspbian Jessie image was uploaded to a flash drive and installed on the system.
- SSH was enabled via the Raspbian system configuration menu.
- VNC Server was setup on the freshly installed operating system to enable access via remote desktop.
- RaspberryPi was configured to connect to local LAN via Wi-Fi on start-up.

3.4.5 RASPBERRYPI CAMERA:

The Pi Camera was setup as the device to capture video data to be streamed over the network.

- Connect Pi Camera to Raspberry Pi via serial interface.
- Pi Camera was activated in the RaspberryPi System configuration menu.
- Setup MJPEGStreamer to stream video on system start-up at 30 fps and 640 x 480 resolution.

3.4.6 ARDUINO & SENSORS:

In order to enable readings from additional sensors, an Arduino was setup to interface with such sensors and send those readings to the RaspberryPi over a serial link.

- An Arduino was connected to the RaspberryPi via serial link.
- The Arduino was connected to a DHT-11 Temperature & Humidity sensor.
- A C Program was written using the Arduino IDE to read the voltage on the sensor and continuously write a formatted string of temperature and humidity readings to the RaspberryPi over the Serial link.

3.4.7 QUADCOPTER:

A 450mm quadcopter was built to carry the camera and sensor setup with the RaspberryPi as well as to receive and to respond autonomously to commands from the ground control application.

- The power connectors of all ESCs were soldered to the power distribution board.
- The wires that connect the motors and ESCs were all soldered to bullet connectors to allow easy attachment and detachment when configuring motor spin directions.
- Wires were soldered to a Deans Connector on one end and the other end soldered to the battery inputs of the power distribution board. A Deans to XT-60 connector was used to connect this setup to the LiPo battery.
- The quadrotor frame with the body plates and arms was assembled.
- The motors, ESCs and Power distribution board were attached to the quadrotor frame.
- The APM2.6 Flight Controller was flashed with the Arducopter 3.2 Firmware using the official Mission Planner software.

- The on-board IMU of the Flight Controller, the input ranges of the 6 channels of the radio controller, as well as the compass were calibrated using the Mission Planner Software.
- The radio receiver was connected to the flight controller utilizing only input 1, with a jumper placed over the signal pins of inputs 2 & 3 in order to trigger the flight controller to accept input from the receiver in PPM mode.
- The ESCs were connected to the appropriate output pins of the flight controller based on their location on the frame.
- All-at-once ESC calibration method was used to calibrate the ESCs to the throttle range of the radio controller.
- The flight controller was attached to the quadrotor frame with on shock absorbing plates to reduce the effect of vibrations during flight on the internal electronics of the flight controller.
- The spin directions of the motors were adjusted by arming the quadcopter without propellers and switching any two wires connecting the motors to the ESCs to reverse spin directions where necessary.

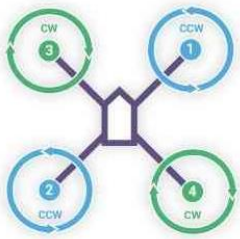

Figure 60: Correct Motor Spin Configuration from Ardupilot Website

- A 3DR Power Module was connected to the Flight Controller and Battery in order to enable the APM to read battery states such as current and voltage level.
- The Raspberry Pi & Arduino along with sensors were mounted on the frame with a power bank.

3.4.8 SENSOR API:

As part of the software system, the need for streaming sensor data over the network was solved by a RESTful web server written in Python.

- A simple REST API server was written using Flask & Python and deployed on the RaspberryPi to read in the sensor readings over the serial port and to serve these readings on request as JSON.

3.4.9 STREAMING API:

A streaming server was setup on the Raspberry Pi in order to serve the images captured by the on-board camera over the network as a video stream.

- MJPEG Streamer was used for video streaming.

- The video stream was configured to serve an MJPEG video stream at a URL matching:
- <ip-address-of-raspberry-pi>:8080/?action=stream

3.4.10 RECORDING & SNAPSHOT UTILITIES:

In order to enable the storing of media from the incoming video, command line scripts were written in order to take snapshots as well as record video in web compatible formats.

- Command line utilities were written using OpenCV to capture still images from the video stream in '.jpeg' format as well as to record the video stream and encode the resulting video file in the '.webm' format so as to be easily served to the browser via the HTML5 Video API.

3.4.11 ON-BOARD CONTROL PROGRAM:

- A Python Program was written and deployed on the RaspberryPi utilizing the open-source DroneKit library along with socket IO to receive instructions from the mission server and relay such instructions to the UAV via the serial connection using the MAVLink protocol.
- This program was also setup to receive system status data from the UAV and relay that information back to the mission server via web sockets.

3.4.12 MISSION SERVER:

- Another Python program was written and deployed to act as the central mission server between the on-board control-program and the ground control application.
- The purpose of this program was to relay messages between the ground control application and the on-board control program via socket-IO.

3.4.13 GROUND CONTROL APPLICATION:

- A Ground Control Application was built utilizing the Ionic Framework to create a cross-platform application that could run on Android, iOS as well as the Web.
- The application was built to accept inputs on the target height of patrol, number of patrol loops and amount of hover time at patrol destination from the ground control operator.
- A map was displayed, for the operator to pick a target location by touching or clicking on the map in the appropriate location.
- On the start of a mission, the ground control application will signal the UAV on its intent to start a mission, to which the UAV will accept and send back the exact UAV home location to the app to appropriately display on the map.
- On receiving the home location of the UAV, the application plots a path to the target location from the UAVs home location using Google Maps API. These paths are sent as an array of polylines, which are decoded into an array of GPS coordinates by the on-board control application.

- On start of a mission, the detection server is also sent a message to indicate the start of a mission along with parameters to identify the particular UAV triggering a detection process.
- Once the initial interaction is complete, the mission is started and a continuous stream of vehicle status data i.e. height, GPS location, sensor readings and battery state that are displayed in the app in real time.
- A continuous stream of vehicle GPS location is also sent to the detection server during the active mission, in order to inform it of the GPS location of detected objects.
- The location of the vehicle is shown in the application in real time and a continuous video stream on the on-board camera is shown for viewing while the mission is active.
- At the end of a mission, both the ground control app and detection server are signalled to stop the data streaming and detection.

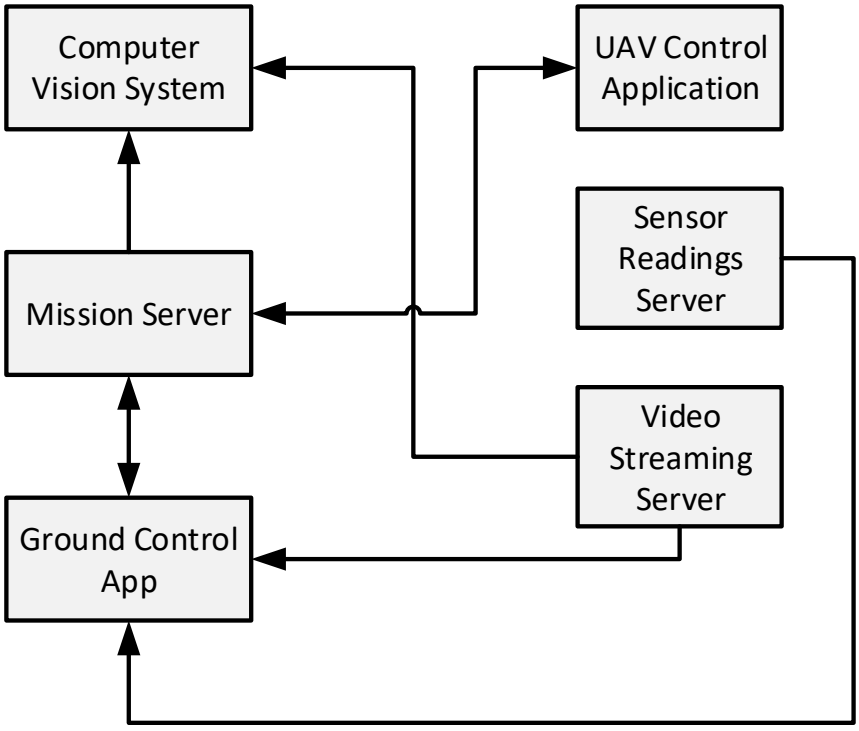

Figure 61: Static & Mobile Camera Subsystem - Software System Architecture

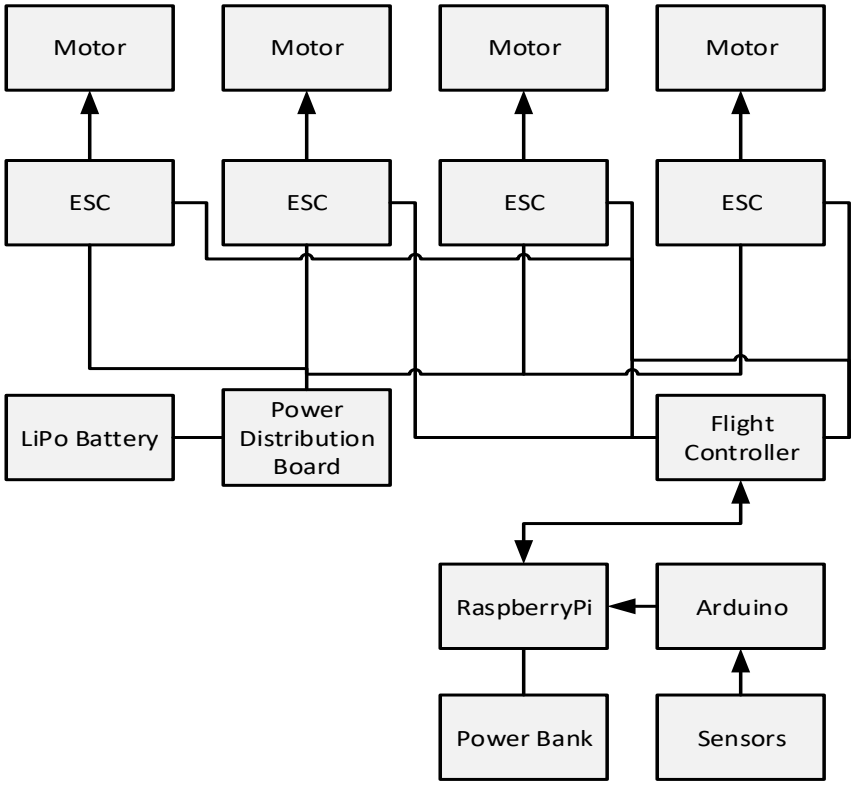

Figure 62: Static & Mobile Camera Subsystem: Hardware System Architecture

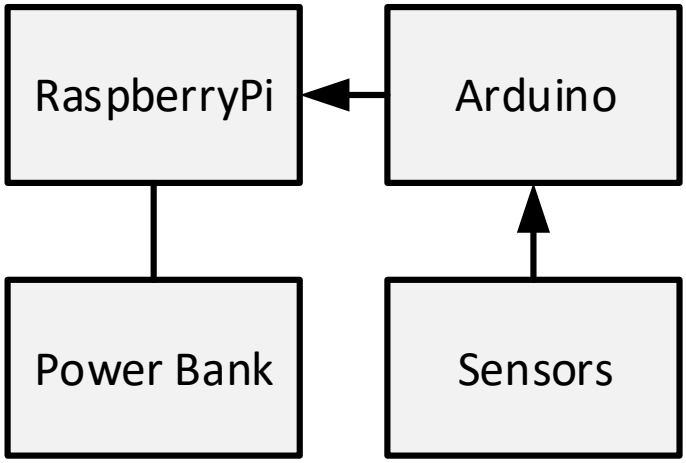

Figure 63: RaspberryPi with Pi Camera & Arduino subset form a static camera subsystem.

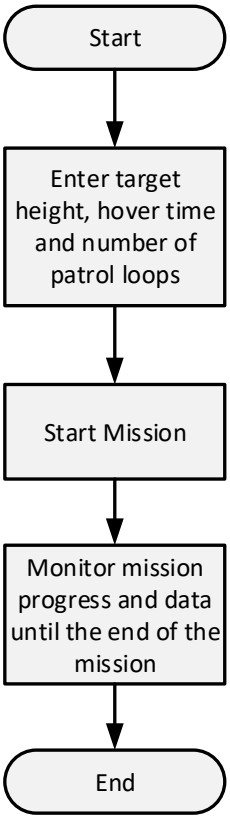

Figure 64: Ground Control App User Flowchart

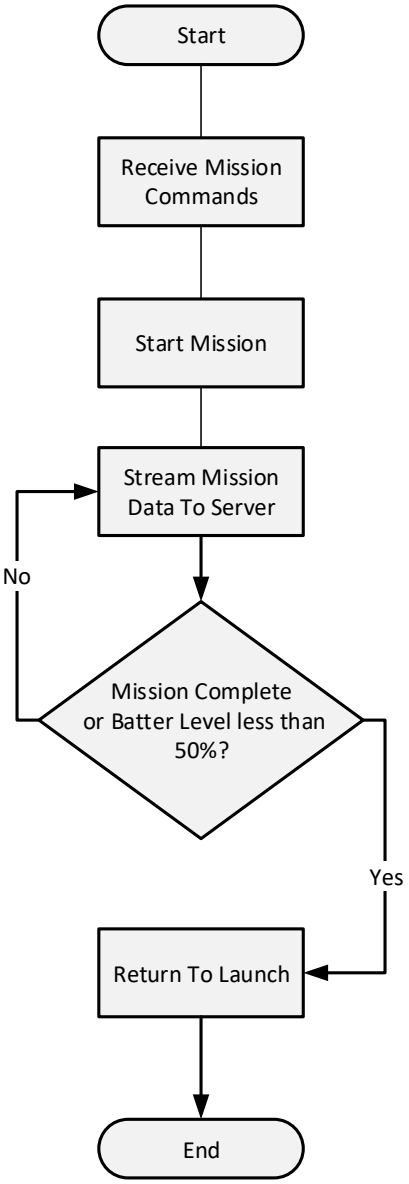

Figure 65: Ground Control App Server Flowchart

CHAPTER 4—RESULTS AND DISCUSSION

4.1 INTRODUCTION

The outcome of this project was tested and verified in order to ensure that its functions validate the objectives and that all requirements are met. Discussions about the results of the project are discussed further in this chapter.

4.2 OBJECT DETECTION AND CLASSIFICATION

This project was tested with video streams fed to it either via HTTP stream or pre-recorded videos. The video stream is mainly that of vehicles on the road. This is due to its purpose in the larger picture. With its two trained models, one for detecting vehicles and pedestrians and the other for detecting anomalies such as road accidents and bad roads, the project was expected to detect objects from video streams corresponding to what it has been trained on and it performed pretty well in this sector. However, due to the high computational power needed by these models to fully function and the low-spec computer used for testing, huge lagging problems were experienced in the processing of the video frames thereby leading to the model missing some of the objects it was supposed to detect—a drop in accuracy. The specifications of the computer that was used to run this project are:

1. **Processor**: Intel® Core(TM) i5 CPU of 2.67GHz
2. **RAM**: 4.00 GB
3. **Operating System**: Ubuntu 16.04 (with a swap file of size 5GB)

The model trained for detecting vehicles was programmed to further classify the vehicles detected into categories such as buses, trucks, saloon cars, motorcycles, etc. and it performed very well in this aspect too.

With the accident and bad road detection model, a decent loss of approximately 1.2 was recorded during its training. There was still some level of inaccuracies it its performance however since it was trained with a low specification computer instead of the regular GPU powered computer and was therefore not able to adequately train.

Figure 66: A Snap of the System Performing Vehicle and Pedestrian Detection

Figure 67: Vehicles detected in a video stream

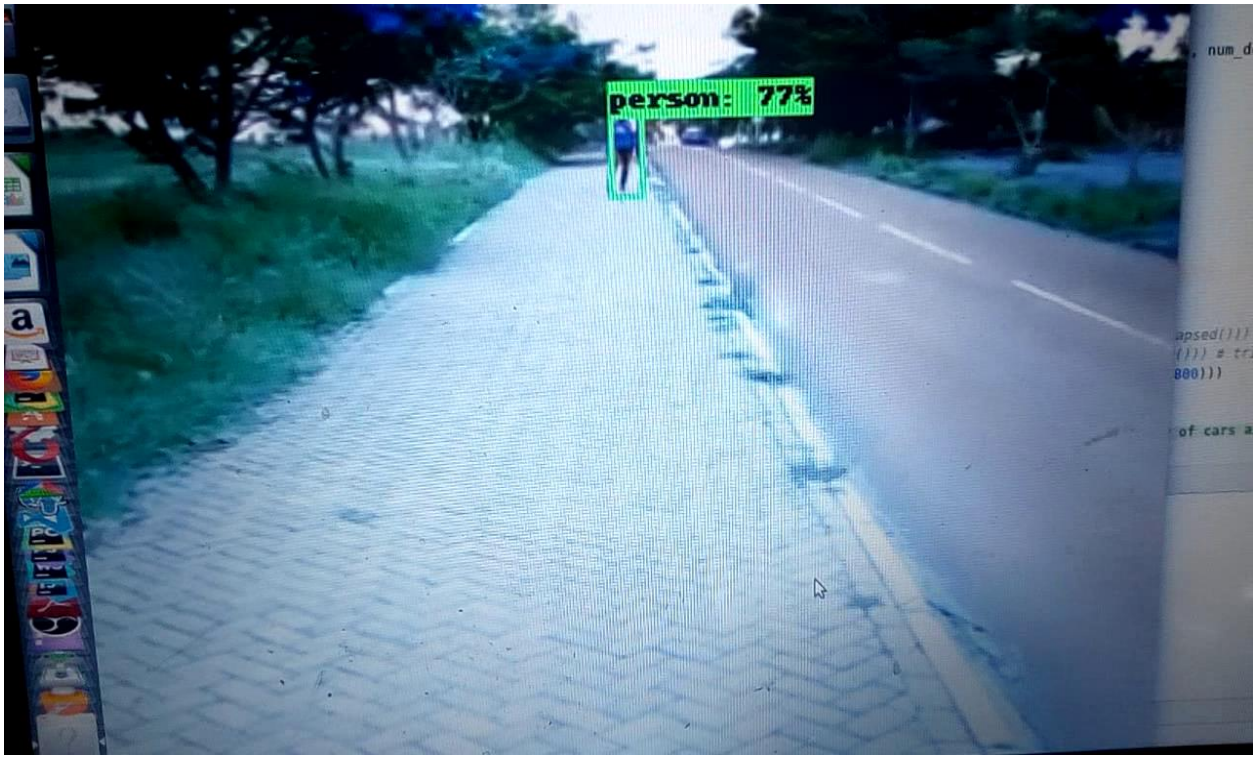

Figure 68: Pedestrian Detected in the Video Stream

4.3 TRAFFIC MONITORING AND CONTROL SYSTEM

Road traffic was simulated using vehicle demand data from two sources: vehicular data from our vehicle detection and anomaly system and randomly generated data.

Our vehicle detection system sends vehicular information to our central traffic server through an API endpoint. The traffic server resends all vehicular information to the client program via a socket implementation to enable real-time communication between them. Vehicle demand data is also generated using a python script and fed to the simulation software.

Vehicle trips for some selected roads were generated at different times of the day making up a large demand or vehicular information to traffic simulation. Three key timeframes identified and used during artificial demand data generation for SUMO [18] were: morning peak hours between 6:30am and 8:30am, normal afternoon traffic between 11:30am and 1:30pm and evening peak traffic between 5:00pm and 6:30pm.

Traffic scenarios at some key vantage points were simulated. Okponglo junction, Shiashie junction, UPSA junction and Tetteh Quashie roundabout are just a few of the many intersections simulated using SUMO [18].

4.3.1 SIMULATION RESULTS

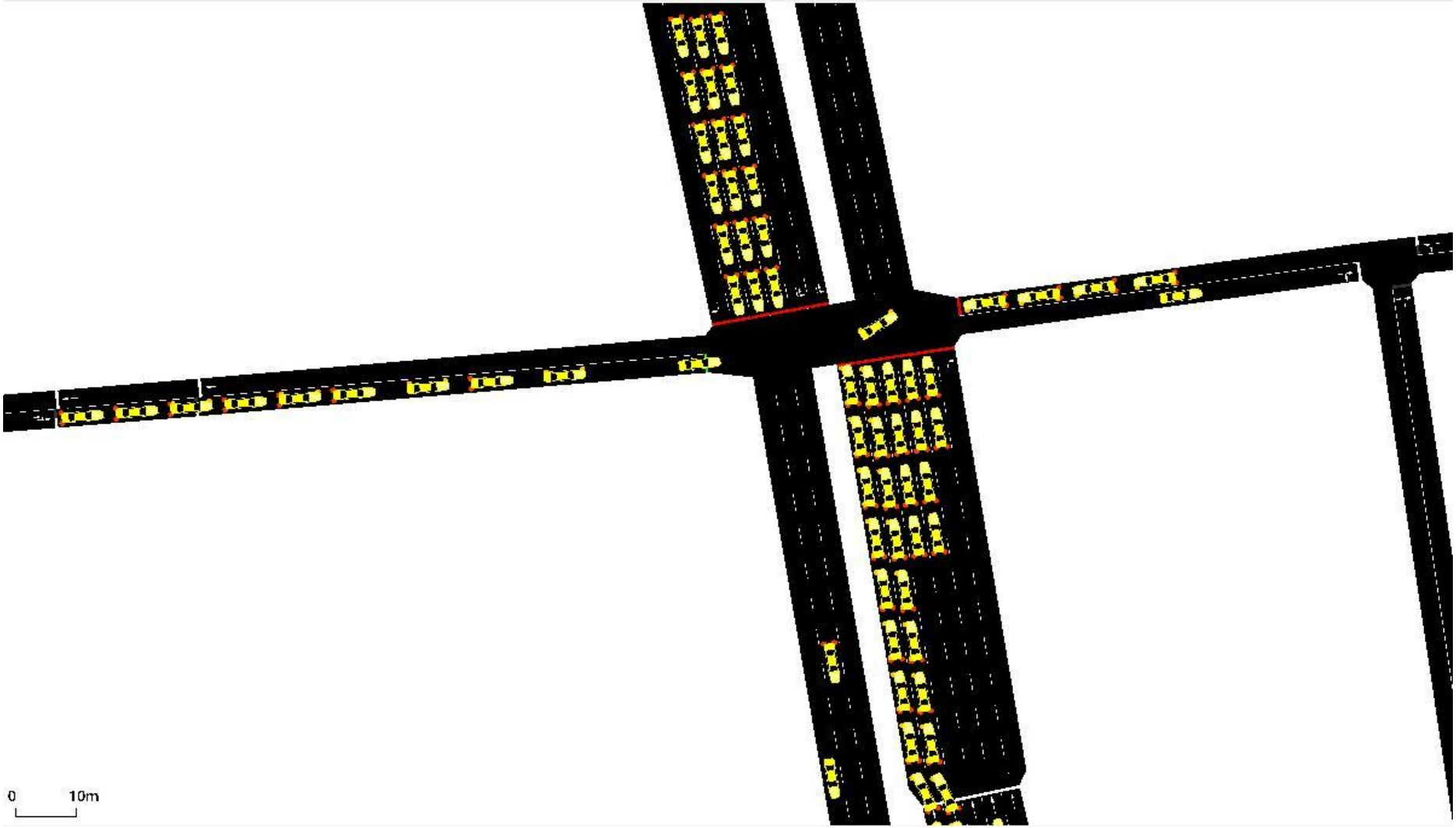

Figure 69: Traffic light simulation at an Okponglo junction in SUMO [18]

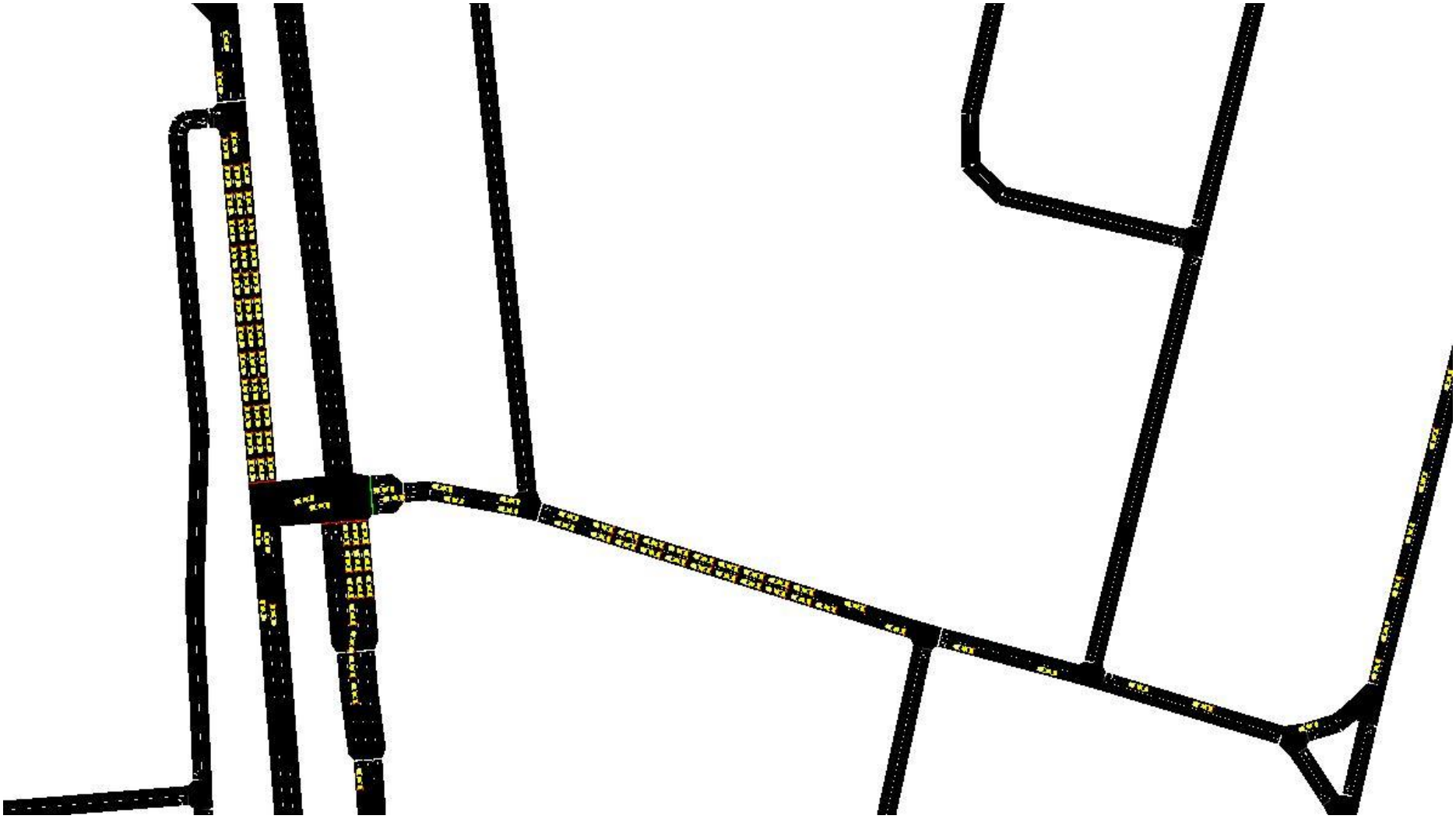

Figure 70: Simulation of traffic at Shiashie with SUMO [18]

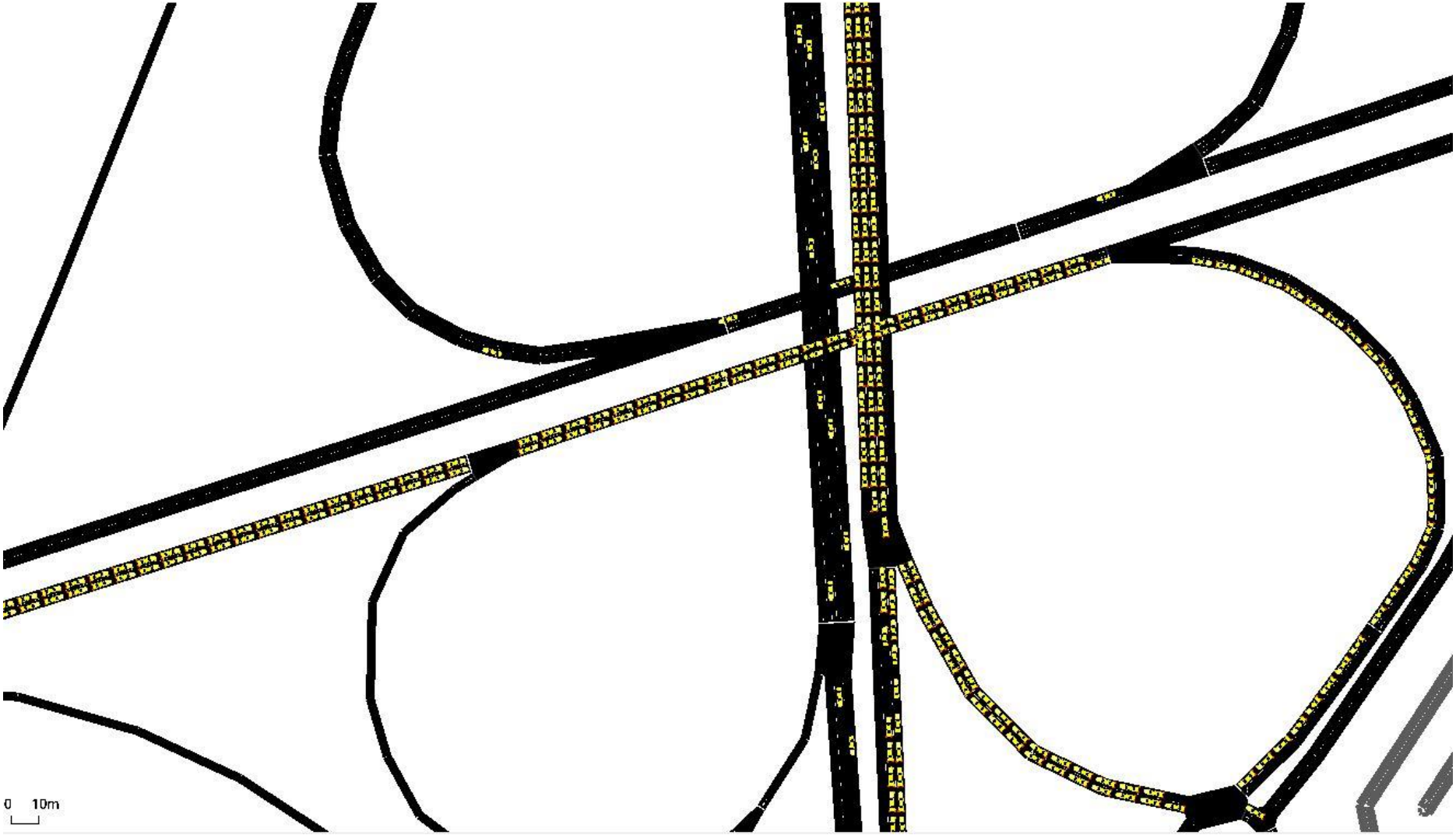

Figure 71: Simulation of a jammed traffic scenario at Tetteh Quarshite Roundabout with SUMO [1]

4.4 ROUTE NAVIGATION AND ALERTING SYSTEM

4.4.1 DESIGN OF MOBILE APPLICATION

The mobile application was built in Android since Android has the largest use base in our part of the world.[29] The application utilised various elements from the Google services for the sake of efficiency and accuracy of data like the Google Maps which was used for displaying the user routes as soon as the optimization process is completed. The application also used Google Places APIs to help road commuters find their destination spots a little easier than having to look for the destinations on the map itself. As part of the design phase, we had to make sure that the application was very responsive and fast in delivering its result. Since most of the processing was done on a server, various API calls to the server were implemented in the application for transfer of data. A Firebase Cloud messaging platform is also deployed in the application for receiving notifications.

4.4.2 IMPLEMENTATION OF ALERTING SYSTEM

The Alerting System was implemented as a software server running with HTTP protocol. It was developed in Java Play framework because the framework allowed a wide range of libraries to make code implementation simpler. Also, the framework was known to allow good E-Mail service rendering.

The mobile application also provided alerting functionality for road commuters to be able to report road anomalies like accidents first hand and in real-time. The application offered 4 modes of alerting depending on the user's conditions and resources available. During the implementation, 3 main resources were considered for selecting the forms of alert to implement.

i. The luxury of time for the user
ii. The availability of call and message data for the user.
iii. The urgency of response required for the anomaly

From these 3 considerations, we came out with 4 alerting modes for road commuters, namely:

i. A feature for calling the nearest police station to a user's current – We use the current location of the road commuter to select the closest station to it. A database of all the numbers from the Ghana Police Website was created for this purpose. This gives the user the ability to speak directly to the police from an accident scene. The Google Places platform helps us with Geo-referencing to find the closest stations near you. In total, we recorder about 60 contacts across the city of Accra.
ii. A feature was added to allow the user to send a general alert message to the server for processing. This is necessary when the road commuter cannot afford the luxury of time to actually send out a vivid description of the anomaly. In this case an API call is handled on the server and the location of the user is found out. A general alert message is sent out to users to warn them of an anomaly reported at a given location.
iii. An image uploading feature – the user can also send a captured image. The image will first be encoded in base64 for a smaller upload time because alerts are to be done in real-time. Sending a file through HTTP tends to be time inefficient and also

depends on the quality of internet on the handset. The encoded image is then decoded at the server end and then sent to the various agencies through the various channels created. These images are also stored on the server in a database for reference. APIs are created so that these images can be queried for in the future for review and referencing.

iv. The speech recognition feature – The speech recognition feature was put in place purposely to be able to understand exactly what the user intends to put across. It gives the user a chance of specificity of the alert. A Google Speech API was implemented for this course. Initially a speech recognition system was built with CMU Sphinx. The system was seen to have quite low accuracy. This was mainly due to the fact that a corpus on road traffic and road traffic anomalies is inexistent. However, when a general text corpus was used, it seemed to return wrong recognitions from the language model since there was a lot of unrelated words which altered the performance of the language model training. Unfortunately, after this, we resorted to the Google Speech API which gave a great accuracy level. After the speech recogniser recognizes the words in the speech, it is transmitted to the server for general alerting.

For all four features, the data is transmitted over Hypertext Transfer Protocol. The information sent are mostly strings to allow quick transmission and high throughput on the server side. Also, the data transmitted always contains the current location of the user alert, so that the geolocation can be obtained on the server side for more specific alerting.

Aside the aforementioned modes of alert, the Computer Vision system was also integrated with the Alerting System to be able to serve detected anomalies. An API on the Vision System is called every 10 seconds to check if an anomaly has been detected by our cameras. A cron job is deployed on the alerting system to seek road anomalies detected by the Vision System. The Vision System responds with a base64 encoding of the list of images it detects in a space of ten seconds. It also sends the accompanying location of the anomalies in terms of latitude and longitude. The geolocation is then derived on the server side and the alert is now sent to the agencies.

For all these alerting methods, road commuters are also alerted of these things with the help of a cloud messaging platform deployed on the server. This was built to give road commuters a heads-up in case of an anomaly on roads which they might decide to ply.

The agencies are alerted either by calls, through Twitter, or through the E-Mail channels. Twitter was selected because it proves to be one of the most active social platforms with public APIs for third party applications.[30] Also, the fact that most news agencies and the Ghana Police are very active on the platform. To be able to integrate the Alerting system with Twitter, we first needed to design a Twitter Developer Application for a created Twitter Account. After this, code was written to access Twitter APIs with account security credentials. Therefore, when an alert comes either from the mobile application or from the Vision System, it is sent straight to all the agencies that are signed up onto the alerting system. Since we did not want to give out wrong information to agencies, Due to this, dummy accounts were created and used for testing. For testing, we used these two accident images below:

Figure 72 Sample Car Accident Image

Figure 73 Sample Car Accident 2

These two images were from accidents detected at different places. Due to the fact that it was very difficult to get accident scenes on the University premises we decided to use these pre-

determined accident images. Therefore, the images were sent to the agencies through the Twitter APIs.

For the E-mail, it was chosen because of its large range of use by various agencies. For this purpose, a test E-Mail account was also used for the transmission. This test account was used to also alert test agency accounts of various anomalies as reported by the user or the V the Vision system.

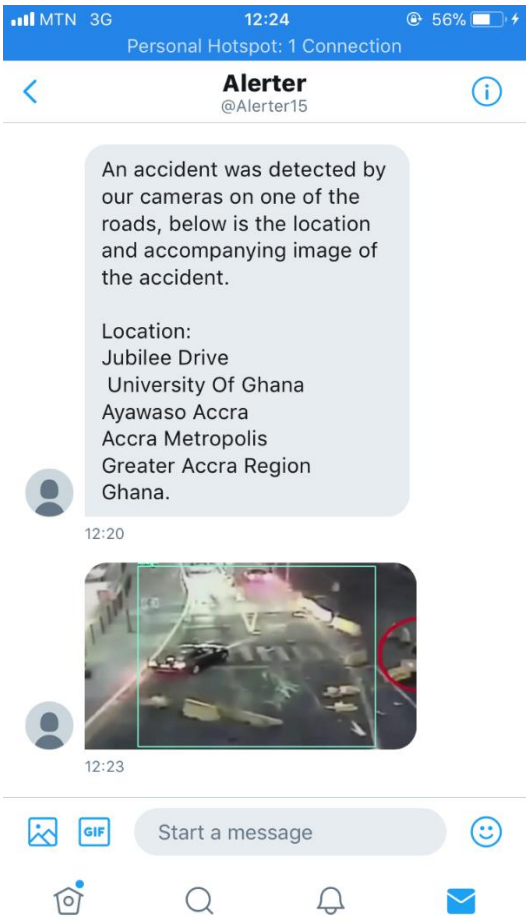

Figure 74 - Sample Anomaly Reported from Vision System

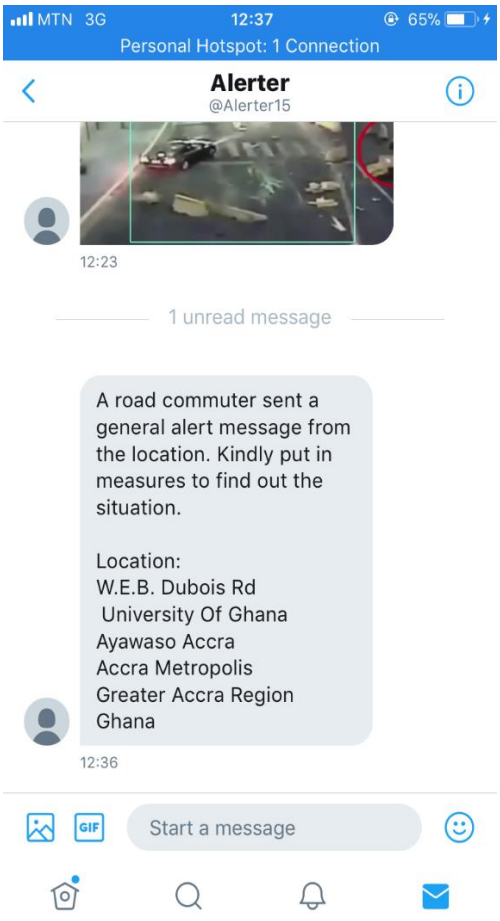

Figure 75- Sample User Report of Anomaly from App

These sample images were obtained from the test Twitter Account. In both of them, messages were sent from the server to alert the agency of an anomaly. One of these came from the Vision System and hence had an image attached. The other came from the Mobile Application as an emergency alert and hence had only the location of the anomaly. All these anomalies were generated from Legon campus. Both of these requests were sent to the agencies in a space of averagely 10 seconds. This was very good performance output. Initially the response came in 30 seconds. But upon code optimization and various forms of pre-processing, the time taken reduced.

For the E-Mail channel, the same method of testing was used to generate the intended outputs. Below are a sample of two test cases.

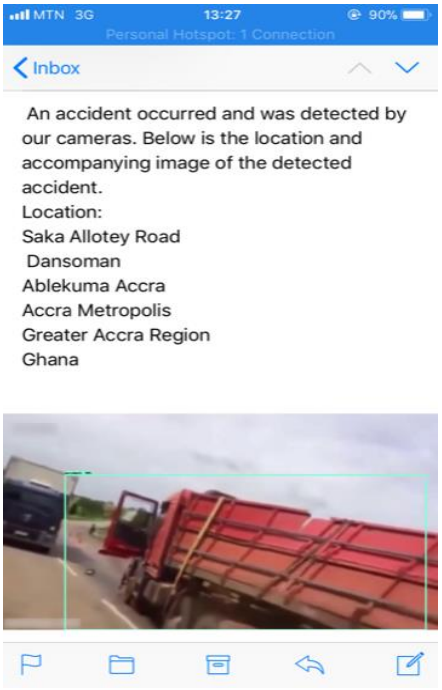

Figure 76- Sample E-Mail Alert from Vision System

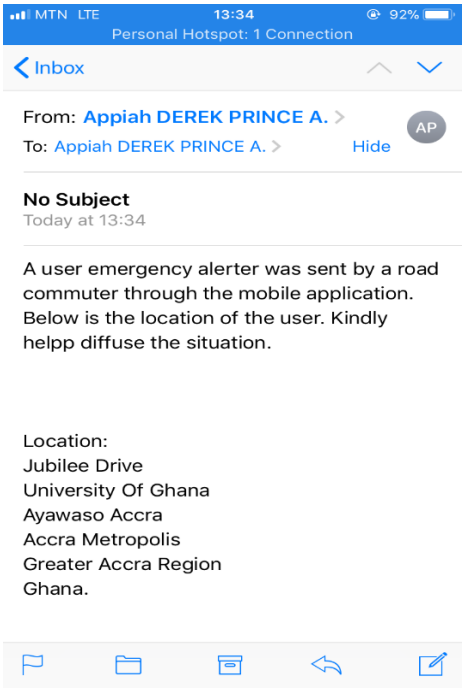

Figure 77- Sample Emergency Alert From User

4.4.3 ROUTING SYSTEM

The idea of the routing system was to help road commuters get to their place of destination with prior knowledge and intelligence. The user, based on his/her current location would select an end location and the best route to get thee is returned.

For this course, a map was deployed in the mobile application to help users view navigation paths. The user decides to either click on the destination or to search for it. The routing system itself is deployed on the server. The mobile application accesses it through an HTTP API call. This was done so that applications could access the routing system asynchronously and independent of each other.

The API receives the starting and end location of the road commuter. The system then generates all possible routes using Google directions. Now to select the optimal route for the user, we

needed obtain live traffic data. After some time of research and observation, we came out with four parameters that normally determine traffic flow, namely:

i. Traffic density – this has to deal with the number of cars detected on a particular route by our Vision System. For each of the routes, there could be numerous connecting roads, the traffic density for each of these routes is calculated taking the number of lanes of the connecting roads into consideration.
ii. Route distance – this is the total distance covered by the route. This is necessary because the distance of a route might in a way determine how fast you get to your destination.
iii. Road Quality – This is a binary value determined by our Vision System. For each connecting road, the road quality is calculated and then an average is found for the total route. Sometimes how good a road is might determine how much time you spend on it.
iv. Road anomaly occurrence- This is the measure of the number of road anomalies that have been reported on the connecting roads of the route. If an accident occurs on a road, it hinders the flow of traffic and hence it needs to be considered when finding the route.

Using this set of parameters, a fuzzy logic algorithm was built to determine route optimality value. This fuzzy logic system was used because of how efficient and how scalable it is.[31] Therefore, for a given set of routes with their various parameters, the logic system returns a value measure of the optimality of the route between 0 and 1. To know the optimal route we simply select the one with the highest value for display.

Before the optimality is determined, the various parameters use membership functions to assign classes to the raw data that is fed in.[15] Two forms of member functions were used to classify the various values for the parameters. This process is called fuzzification.[32] The functions that were used in ts project are the Trapezium (four-point member function) and the Triangular (three point) member function. All parameters used the triangular member function except the Traffic Density and Route Distance parameters which used the Trapezium member function.

The optimality is determined by a set of rules. Therefore, a set of rules was derived. Most of these rules were defined by us using basic common sense and some mathematical predications. These are the rules that determine which route is optimal. If the rules are bias, the output of the system is in-turn bias, therefore various series of tests were run to ensure that the rules were fair and gave correct result. After the fuzzy logic system was tested, the system was observed to give 97% accurate result when tested against decisions with obvious outcomes. This was seen to be fairly accurate.

Below are some sample results obtained from the Routing System in the mobile application.

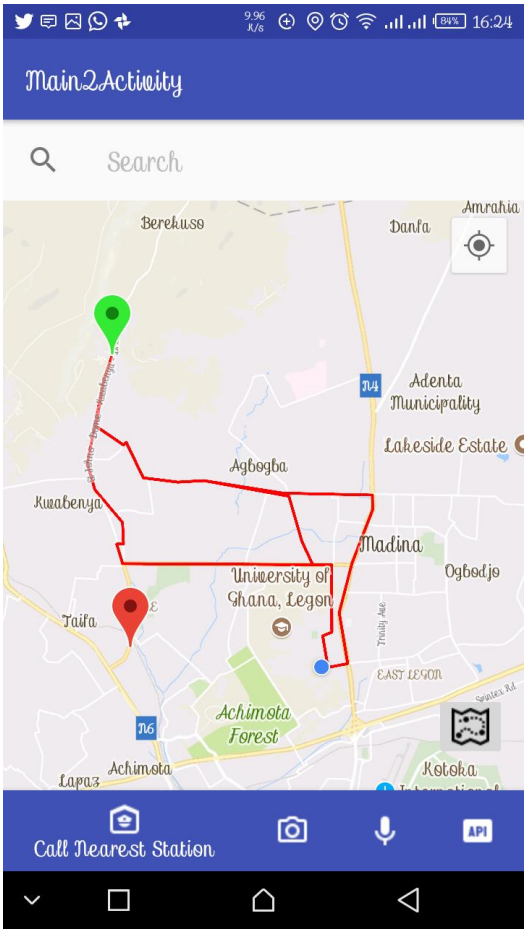

Figure 78- Sample of ALL Possible Routes to a Location

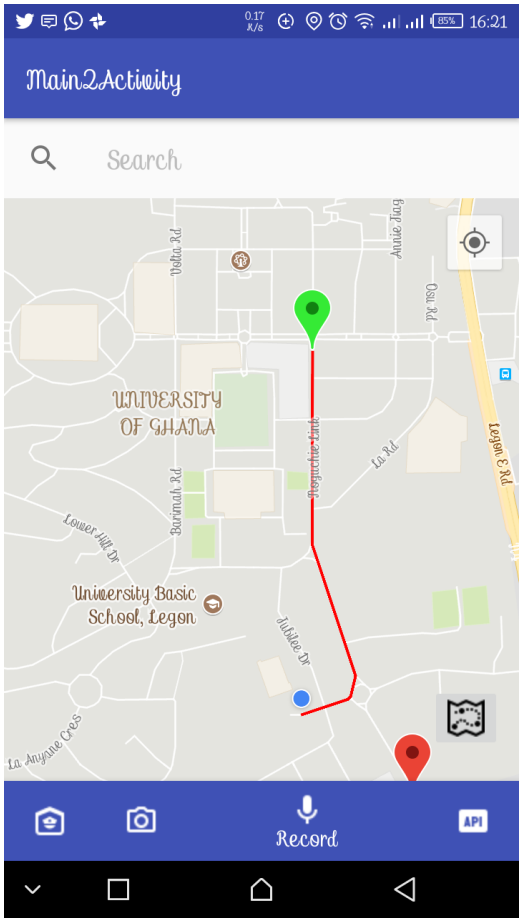

Figure 79: Sample Image of optimal route prediction

4.5 STATIC & MOBILE CAMERA SUBSYSTEM

4.5.1 MANUAL TEST FLIGHT:

- The quadrotor was successfully flown using the FlySky Controller to ensure systems worked as expected before implementing autonomous flight.

4.5.2 SIMULATION OF AUTONOMOUS FLIGHT:

In order to enable the testing of autonomous flight capabilities in a low risk environment as they were being developed:

- A simulated vehicle was setup using the DroneKit-SITL program.
- The MAVProxy program was used in order to split the MAVLink connection to the SITL vehicle so as to be accessible from both the control application the mission control program simultaneously.

This was done to ensure that the actions of the simulated vehicle could be monitored for accuracy by a third-party program

The UAV in the simulation was able to perform the patrol task as initiated in the ground control app with the appropriate parameters.

Video & Sensor data were successfully streamed back to the ground control application as well as the detection server.

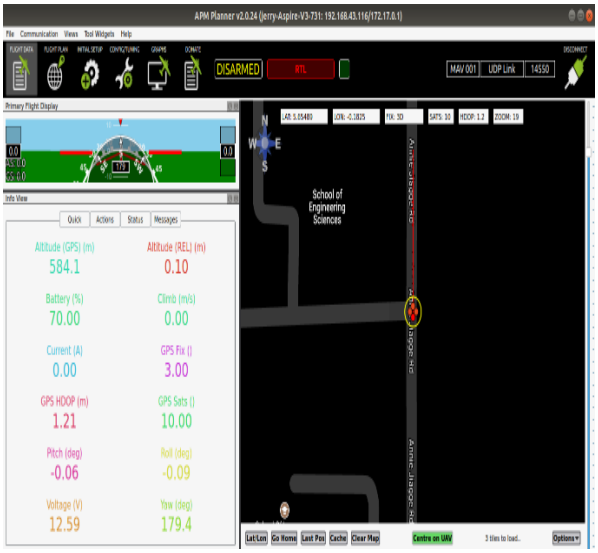

Figure 80: View of the UAVs state during mission using mission planner

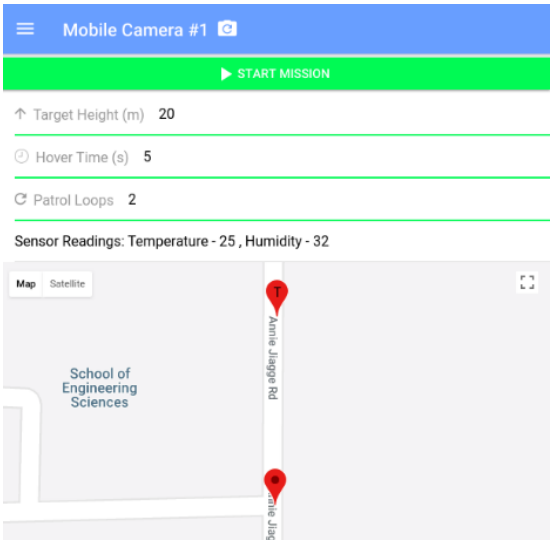

Figure 81: Ground Control Application - Configure and Start mission

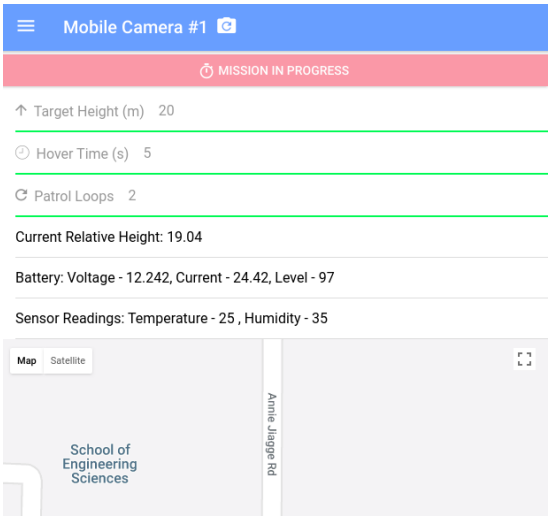

Figure 82:Ground Control Application - On-going mission stream of vehicle state

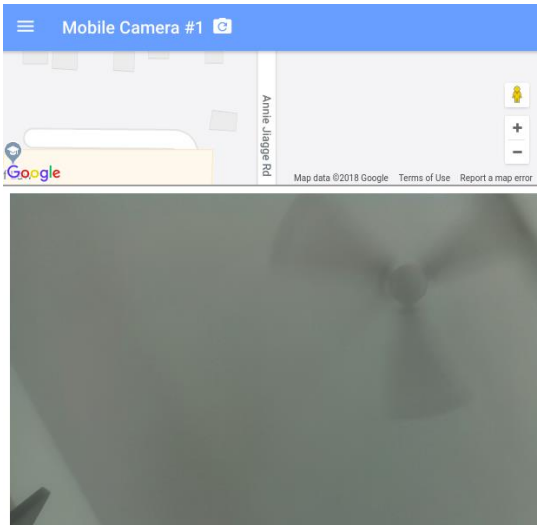

Figure 83: Ground Control Application - Live video streaming

4.5.3 FLIGHT CONTROLLER PARAMETER ADJUSTMENTS:

On initial flight of the quadrotor, some ESCs failed to respond in sync for the minimum throttle range that was set when the copter was armed.

The flight controller parameters were adjusted to using Mission Planner to write the values to the controller:

$MIN\_THROT(\frac{42}{1000})$

$MOT\_SPIN\_MIN(\frac{450}{1000})$

After which uniform spin start and end times as well as speeds were observed on all motors.

4.5.4 CRASH INCIDENCE AND REPAIR:

Successful Manual flight was achieved using the radio controller, due to inexperience of the pilot, sustained multiple crashes, autonomous flight program connected to drone displayed erratic positioning and state data esp. in altitude and GPS HDOP of 14+.

Q-Bond Adhesive was used to fix a broken arm that was damaged during the first test flight.

CHAPTER 5—CONCLUSION

5.1 INTRODUCTION

In this chapter, conclusions are drawn on the project automatic traffic scheduling and alerting system using computer vison. However, several recommendations for further work are stated to enhance this project.

5.2 CONCLUSION

In short, this project combines together a computer vision powered system capable of detecting vehicles and pedestrians along the road (referred together as the traffic data) together with anomalies such as road accidents that might happen on the road and in some instances bad road conditions. The traffic data is then seamlessly transmitted to the Traffic Scheduling System (TSS) via HTTP Post request in order to be used in the optimization of road traffic scheduling to ease unnecessary traffic congestions that occur at certain times of the day especially in the morning and evening. The road anomalies detected are also transmitted to the Android Application System (AAS) to be forwarded to appropriate authorities for immediate actions. Live video streaming is also transmitted from the Mobile Surveillance System using drone capabilities to the Computer Vision System (CVS) to be processed for anomaly detection.

The development environments include Python, Node.js and Java programming. Multithreading was also used in the CVS to enable video streams being transmitted from cameras mounted on Raspberry Pis at various locations where traffic needs to be controlled to be processed concurrently. Sockets were used to ensure that connections between systems that need to communicate continuously as a result of operations such as video streaming were kept open all the time. Web APIs were also used to transmit JSON data from one system to another. Smart and traffic scheduling to complement the regular timer-based traffic signal lights, road anomaly detection, mobile surveillance and reporting as well as commuter rerouting based on road optimization, the main objectives of this project has been successfully implemented.

All the individual systems of this project have been successfully integrated and checked to ensure that all requirements stated are met and also, a prototype of this project has been built to meet the objectives brought forward during our proposal.

5.3 FUTURE WORKS

With all objectives that were proposed during our proposal having been met, there are still some petty issues which could be eliminated to even make this system more robust and efficient than it even is. For instance, a GPU enabled computer can be used to run this system in order to increase the frames per second of the video frames and reduce the lagging effects experienced during its operation. Also, more models can be designed, developed and trained in order to boost the detection powers of this system to an even greater extent. A more efficient algorithm: Particle Swarm Optimization(PSO) can be used improve traffic optimization.

REFERENCES

[1] R. AmoahNyarko, “The Causes and Effects of Traffic Jam on Commercial Transport Operations. A Case Study of Trotro Drivers at Awutu Senya East Municipality,” no. 10130638, 2014.

[2] M. H. Ali, S. Kurokawa, and A. A. Shafie, “Autonomous road surveillance system: A proposed model for vehicle detection and traffic signal control,” *Procedia Comput. Sci.*, vol. 19, pp. 963–970, 2013.

[3] California, “Short Term Freeway Traffic Flow Prediction Using Genetically-Optimized Time-Delay-Based Neural Networks,” no. iv, 2005.

[4] T. Sridevi and A. Proffesor, “Automatic Generation of Traffic Signal Based on Traffic Volume,” 2017.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks.”

[6] L. Huang, “Real-time multi-vehicle detection and sub-feature based tracking for traffic surveillance systems,” in *CAR 2010 - 2010 2nd International Asia Conference on Informatics in Control, Automation and Robotics*, 2010.

[7] M. Papageorgiou, “AN INTEGRATED CONTROL APPROACH,” 1995.

[8] M. Papageorgiou, C. Diakaki, V. Dinopoulou, A. Kotsialos, and Y. Wang, “Review of Road Traffic Control Strategies,” vol. 91, no. 12, 2003.

[9] A. Meemongkol, A. Numsomran, V. Tipsuwannaporn, and P. Boonsrimuang, “Traffic forecasting and navigation assistance system via web application,” *ICCAS 2007 - Int. Conf. Control. Autom. Syst.*, pp. 857–860, 2007.

[10] L. Sadanandan and S. Nithin, “A smart transportation system facilitating on-demand bus and route allocation,” *2017 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI 2017*, vol. 2017–Janua, pp. 1000–1003, 2017.

[11] P. Khilari and P. B. V P, “Implementation of Speech to Text Conversion,” pp. 6441–6450, 2015.

[12] A. Dhankar, “Study of deep learning and CMU sphinx in automatic speech recognition,” *2017 Int. Conf. Adv. Comput. Commun. Informatics*, pp. 2296–2301, 2017.

[13] R. E. Gruhn, W. Minker, and S. Nakamura, “Statistical Pronunciation Modeling for Non-Native Speech Processing,” 2011.

[14] A. Bhandare, M. Bhide, P. Gokhale, and R. Chandavarkar, “Applications of Convolutional Neural Networks,” *Int. J. Comput. Sci. Inf. Technol.*, vol. 7, no. 5, pp. 2206–2215, 2016.

[15] R. Rojas, “Fuzzy Logic,” *Neural Networks*, pp. 289–310, 1996.

[16] G. Feng, “Introduction to Fuzzy Logic Control,” vol. 20102151, no. January, pp. 1–12, 2010.

[17] H. J. Zimmermann, “Fuzzy set theory,” *Wiley Interdiscip. Rev. Comput. Stat.*, vol. 2,

no. 3, pp. 317–332, 2010.

[18] “Knowledge Based Systems using Fuzzy Logic Final Report,” pp. 1–18.

[19] Y. M. Chen and L. Dong, “Real-Time Video Relay for UAV Traffic Surveillance Systems Through Available Communication Networks,” pp. 2610–2614, 2007.

[20] J. Zhang, J. I. E. Cao, and B. O. Mao, “APPLICATION OF DEEP LEARNING AND UNMANNED AERIAL VEHICLE TECHNOLOGY IN TRAFFIC FLOW MONITORING,” 2012.

[21] “Raspberry Pi.” .

[22] “Raspberry Pi camera module.” .

[23] “Wi-Fi Dongle.” .

[24] “Neural Network Architectures.” .

[25] J. Huang *et al.*, “Speed/accuracy trade-offs for modern convolutional object detectors.”

[26] D. Impiombato *et al.*, “Characterization and performance of the ASIC (CITIROC) front-end of the ASTRI camera,” *Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*. 2015.

[27] “Inception v2.” .

[28] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” 2015.

[29] J. Poushter *et al.*, “Cell Phones in Africa : Communication Lifeline,” *Pew Res. Cent.*, pp. 1–16, 2015.

[30] S. Kemp, “Digital in 2017: Global Overview,” *we are Soc. Hootsuite*, p. 107, 2017.

[31] X. Daura, “Fuzzy Logic,” vol. 1, 2005.

[32] A. Nasrinahar, E. E. Faculty, and K. Lumpur, “Effective Route Planning of a Mobile Robot for Static and Dynamic Obstacles with Fuzzy Logic,” no. November, pp. 25–27, 2016.

APPENDICES

A. VEHICLES, PEDESTRIAN, ROAD ACCIDENT AND BAD ROAD DETECTION

```
from __future__ import print_function

'''
 BRIEF DOCUMENTATION ABOUT CLASS
 -------------------------------
    The purpose of this script is to detect the number of vehicles
(trucks, saloon cars, buses and
 motorcycles) and pedestrians at various intersections at a chosen
location (test point) with the purpose of
 using these data to efficiently and effectively control the flow of
traffic in such areas hence gradually
 wiping out the timer traffic light signals and replacing them with
much smarter systems. The number of
 vehicles (trucks, saloon cars, buses and motorcycles) and
pedestrians detected is then transmitted via an
 API request to the Traffic Simulator System (TSS).
    Also, whenever an anomaly such as a road accident is detected,
the system automatically captures that scene
 in the video as an image and then save it. This data is later
transmitted to the Android Application System (AAS)
 of the project via an API from the AAS which pings a specially
created API made for it every 10 seconds.
 This API (named cus_anomalies_server) checks a directory to see it
any accident has been detected when pinged and
 if yes, transmit the image as a base64 encoded string over to the
AAS.

 CLASS NAME:
    TertiaOculus


 METHODS:
    * __init__

            This method takes as argument -

            # A model_selector (int) to chose between the two models
available for detection(
                1 - for selecting the model for detecting vehicles
and pedestrians
                2 - for selecting the model for detecting bad roads
and accidents)
```

```
            # path to the video source on which detection will be
made


        :param model_selector: for selecting road accident and bad
roads model (1) or
         vehicle and pedestrian detection (2)
        :param source_path: Path  to the stream source
        :param source_name: Name of the video stream source
        :param node_id: Location of the Pi Camer sending video
stream
        :param socket_source: Boolean controlling the type of video
stream (locally saved files/http stream)
        :param gps_location: the gps coordinates of the camera
location

        :return - void

    * pre_processing
            This class presets the detection graph to be used (
which by default and fixed is tensorflow's graph).
            It also maps the labels to each class and correctly
identifies the categories and category indices
            involved

        :param - void

        :returns: label_maps category and category_index

    * run

            The core operations of this class is in this method.
Detections and database operations are done in
            this class.

        :param - void

        :return - void

'''


try:
    # database imports for data backup
    from pymongo import MongoClient
```

```
    # multithreading for better camera performance
    from imutils.video import WebcamVideoStream
    from imutils.video import FPS

    # Computer Vision system Imports
    import tensorflow as tf
    import numpy as np
    import cv2, urllib, os, sys, datetime, scipy.misc, requests,
json

    # API Builds import
    from flask import Flask
    from flask_cors import CORS
    from random import randint
    from bson.objectid import ObjectId

    # custom object detection utils import
    from utils import label_map_util
    from utils import visualization_utils as vis_util
except ImportError as e:
    print(str(e))


app = Flask(__name__)
app.config['SECRET_KEY'] = str(randint(1000, 10000))
CORS(app)

sys.path.append("..")


class TertiaOculus():

    # model_selector:
        # 1 - for model detecting road accident and bad roads
        # 2 - for model detecting vehicles and pedestrians

    def __init__(self, model_selector=2,
source_path='cus_test_dir/cus_road_accident.mp4',
                 source_name='Default Stream', node_id='nx3433',
gps_location={'lat':'43', 'long':'-102'},
                 socket_source=False):

        '''
        :param model_selector: for selecting road accident and bad
roads model (1) or
        vehicle and pedestrian detection (2)
```

```
        :param source_path: Path  to the stream source
        :param source_name: Name of the video stream source
        :param node_id: Location of the Pi Camer sending video
stream
        :param socket_source: Boolean controlling the type of video
stream (locally saved files/http stream)
        :param gps_location: the gps coordinates of the camera
location
        '''

        self.socket_source = socket_source
        self.source_name = source_name
        self.source_location = node_id
        self.anomaly_location = gps_location
        self.counter_sender = 0

        # camera initialization (using an mp4 video file or http
video stream in this instance)
        if self.socket_source is False:
            self.cap = WebcamVideoStream(src=source_path).start()  #
trial
        elif self.socket_source is True:
            self.cap = urllib.urlopen(source_path)

        # start measuring the frame per seconds to see performance
        self.fps = FPS().start()  # trial

        # create the collection name for the database
        # connect to the db

        # create a db client
        self.client = MongoClient()

        # create a db
        self.db = self.client['traffic_storage']

        # create a collection for cars and pedestrians observation
        self.general_data = self.db['traffic_data']

        # create a collection for bad roads and accidents
        self.br_acc = self.db["anomaly_data"]


        ''''
        ------------------------- NOTE -----------------------
```

```
        1. model_1 has been trained to detect road accidents, cars,
pedestrians and bad roads but
        performs best in detecting road accidents and bad roads

        2. model_2 is specifically used to detect pedestrians,
trucks, cars, buses and trucks
        '''

        # available models which can be used
        self.model_1 = 'final_year_project_2018'
        self.model_2 = 'ssd_inception_v2_coco_2017_11_17'

        # controls which particular model to use
        if model_selector == 1:
            self.model_choice = self.model_1
        elif model_selector == 2:
            self.model_choice = self.model_2
        else:
            self.model_choice = self.model_2

        # chooses the model to use :-> default set to model 2 @
class init
        self.MODEL_NAME = self.model_choice

        # Path to frozen detection graph. This is the actual model
that is used for the object detection.
        self.PATH_TO_CKPT = self.MODEL_NAME +
'/frozen_inference_graph.pb'

        # label to use.
        # Note: label map chosen should match the model number
chosen above
        self.label_map_1 = 'object_detection.pbtxt'
        self.label_map_2 = 'mscoco_label_map.pbtxt'

        # directory where label map chosen is located
        # Note: Once again, there should be consistency in the
numbers chosen
        self.label_map_dir_1 = 'training'
        self.label_map_dir_2 = 'data'

        # List of the strings that is used to add correct label for
each box.
        # first argument is the directory of the label map and the
second argument is the label map file
```

```
        # ensures consistency in model-label relationship mapping
        if model_selector == 1:
            self.PATH_TO_LABELS = os.path.join(self.label_map_dir_1,
self.label_map_1)
        elif model_selector == 2:
            self.PATH_TO_LABELS = os.path.join(self.label_map_dir_2,
self.label_map_2)
        else:
            self.PATH_TO_LABELS = os.path.join(self.label_map_dir_2,
self.label_map_2)

        # number of possible classes that can be detected.
        # Note model_1 has been trained to detect only 4 different
object classes while model_2 has been
        #  trained to detect 90 different object objects.
        self.classes_1 = 4
        self.classes_2 = 90

        # number of classes to detect

        # ensures consistency in label-class relationship mapping
        if model_selector == 1:
            self.NUM_CLASSES = self.classes_1
        elif model_selector == 2:
            self.NUM_CLASSES = self.classes_2
        else:
            self.NUM_CLASSES = self.classes_2

        # set the detection graph to tensorflow graph
        self.detection_graph = tf.Graph()

    # begin preprocessing
    def pre_processing(self):
        with self.detection_graph.as_default():
          od_graph_def = tf.GraphDef()
          with tf.gfile.GFile(self.PATH_TO_CKPT, 'rb') as fid:
            serialized_graph = fid.read()
            od_graph_def.ParseFromString(serialized_graph)
            tf.import_graph_def(od_graph_def, name='')

        label_map =
label_map_util.load_labelmap(self.PATH_TO_LABELS)
        categories =
label_map_util.convert_label_map_to_categories(label_map,

max_num_classes=self.NUM_CLASSES,
```

```
use_display_name=True)
        category_index =
label_map_util.create_category_index(categories)

        return label_map, categories, category_index

    def gen_filename(self, filename, folder_path):
        i = 1
        while os.path.exists(os.path.join(folder_path, filename)):
            name, extension = os.path.splitext(filename)
            filename = '%s_%s%s' % (name,
str(datetime.datetime.utcnow()), extension)
            i += 1
        return filename

    # begin core job
    def run(self):
        # Size, in inches, of the output images.
        # IMAGE_SIZE = (12, 8)
        # get the label_map, categories and their indices from the
pre-processing method
        label_map, categories, category_index =
self.pre_processing()

        with self.detection_graph.as_default():
            with tf.Session(graph=self.detection_graph) as sess:
                image_tensor =
self.detection_graph.get_tensor_by_name('image_tensor:0')
                detection_boxes =
self.detection_graph.get_tensor_by_name('detection_boxes:0')
                detection_scores =
self.detection_graph.get_tensor_by_name('detection_scores:0')
                detection_classes =
self.detection_graph.get_tensor_by_name('detection_classes:0')
                num_detections =
self.detection_graph.get_tensor_by_name('num_detections:0')
                bytes = ''
                while True:
                    # Handles live video streams
                    if self.socket_source is True:
                        bytes+=self.cap.read(1024)
                        a = bytes.find('\xff\xd8')
                        b = bytes.find('\xff\xd9')
                        if a != -1 and b != -1:
                            jpg = bytes[a:b + 2]
```

```
                            bytes = bytes[b + 2:]
                            image_np =
cv2.imdecode(np.fromstring(jpg, dtype=np.uint8),
cv2.CV_LOAD_IMAGE_COLOR)
                            image_np = cv2.cvtColor(image_np,
cv2.COLOR_RGB2BGR)
                            self.fps.update()  # trial

                            image_np_expanded =
np.expand_dims(image_np, axis=0)
                            (boxes, scores, classes, num) =
sess.run(
                              [detection_boxes, detection_scores,
detection_classes, num_detections],
                              feed_dict={image_tensor:
image_np_expanded})

                            # Visualization of the results of a
detection.

vis_util.visualize_boxes_and_labels_on_image_array(
                              image_np,
                              np.squeeze(boxes),
                              np.squeeze(classes).astype(np.int32),
                              np.squeeze(scores),
                              category_index,
                              use_normalized_coordinates=True,
                              line_thickness=5)

                            self.fps.stop() # trial
                            # print("[INFO] elasped time:
{:.2f}s".format(self.fps.elapsed())) # trial
                            # print("[INFO] approx. FPS:
{:.2f}s".format(self.fps.fps())) # trial
                            cv2.imshow(self.source_name,
cv2.resize(image_np, (800, 600)))

                            '''
                            To get the statistics needed, we need to
count the number of cars and breaking
                            them down further, we need to obtain the
number of trucks, buses, motorcycles
                            and saloon cars(miscellaneous) that use
a particular road together with the
                            pedestrians and other aspects of
interest
```

```
                            '''

                            # count the number of pedestrians and
cars in the image
                            person_counter = 0  # count the
pedestrians
                            vehicle_counter = 0  # count all
vehicles
                            truck_counter = 0  # count only trucks
                            bus_counter = 0  # count only buses
                            car_counter = 0  # count other
miscellaneous cars
                            motorcycle_counter = 0  # count
motorcycles

                            # count the number of accidents and
bad_roads observed/detected in the image
                            accident_counter = 0 # count only
accidents occured
                            bad_road_counter = 0 # count only bad
roads detected

                            # report detected objects only if
confidence is greater than 50%
                            detected = [category_index.get(value)
for index, value in enumerate(classes[0])
                                        if scores[0, index] > 0.5]
                            # print('detected: {}'.format(detected))

                            # get the list of all detected objects
                            for ind_list in detected:

                                # get the values of the detected
objects in the list
                                ids = ind_list.values()

                                # loop through the values obtained
                                for i in ids:

                                    # check for activated model and
act accordingly
                                    if self.model_choice ==
self.model_2:

                                        # acquire data for analytics
```

```
                                        # for a car, bus or a truck
detected
                                        if i == 'car' or i == 'bus'
or i == 'truck':
                                            vehicle_counter += 1

                                        # for a pedestrian
                                        if i == 'person':
                                            person_counter += 1

                                        # for a truck
                                        if i == 'truck':
                                            truck_counter += 1

                                        # for a bus
                                        if i == 'bus':
                                            bus_counter += 1

                                        # for a car
                                        if i == 'car':
                                            car_counter += 1

                                        # for a motorcycle
                                        if i == 'motorcycle':
                                            motorcycle_counter += 1

                                        # log results on terminal
for debugging
                                        if person_counter != 0:
                                            print('(LIVE)
Pedestrians count: {}'.format(person_counter))
                                        if vehicle_counter != 0:
                                            print('(LIVE) Vehicles
count: {}'.format(vehicle_counter))
                                        if car_counter != 0:
                                            print('(LIVE) Cars
count: {}'.format(car_counter))
                                        if truck_counter != 0:
                                            print('(LIVE) Trucks
count: {}'.format(truck_counter))
                                        if bus_counter != 0:
                                            print('(LIVE) Buses
count: {}'.format(bus_counter))
                                        if motorcycle_counter != 0:
                                            print('(LIVE)
Motorcycles count: {}'.format(motorcycle_counter))
```

```
                                        # prepare and format
information to be inserted into the database
                                        traffic_info = {
                                            "edgeID":
str(self.source_location),
                                            "noOfVehicles":
str(vehicle_counter),
                                            "noOfPedestrians":
str(person_counter),
                                            "noOfCars":
str(car_counter),
                                            "noOfTrucks":
str(truck_counter),
                                            "noOfBuses":
str(bus_counter),
                                            "noOfMotorcycles":
str(motorcycle_counter),
                                            "timestamp":
str(datetime.datetime.utcnow())
                                        }

                                        # insert into database and
get the traffic information id
                                        # general_traffic_info_id =
self.general_data.insert(traffic_info)
                                        url_path =
'http://192.168.43.39:6030/vehicleDetect/add'
                                        # transmit information to
TSS
                                        requests.post(url=url_path,
json={"detection": traffic_info})


                                    # checks for activated model and
works accordingly
                                    if self.model_choice ==
self.model_1:

                                        # for bad roads detected
                                        if i == 'bad_road':
                                            # DONE: SAVE THE BAD
ROAD IMAGE FRAME
                                            path =
'cus_anomalies/bad_roads'
```

```
                                            concat = "%s_%s" %
(self.anomaly_location.get("lat"),

self.anomaly_location.get("long"))
                                            name =
"bad_road_%s_%s.jpg" % (str(datetime.datetime.now()), str(concat))

scipy.misc.imsave(os.path.join(path, name), image_np)
                                            bad_road_counter += 1

                                        # for accidents detected
                                        if i == 'accident':
                                            # DONE: SAVE THE
ACCIDENT IMAGE FRAME
                                            path =
'cus_anomalies/accidents'
                                            concat = "%s_%s" %
(self.anomaly_location.get("lat"),

self.anomaly_location.get("long"))
                                            name =
"accident_%s_%s.jpg" % (str(datetime.datetime.now()), str(concat))

scipy.misc.imsave(os.path.join(path, name), image_np)
                                            accident_counter += 1

                                        if accident_counter != 0:
                                            print('(LIVE) Number of
Accidents Detected: {}'.format(accident_counter))
                                        if bad_road_counter != 0:
                                            self.counter_sender = 1
                                            print('(LIVE) Number of
Bad Roads Detected: {}'.format(bad_road_counter))

                                        # prepare and format
information to be inserted into the database
                                        br_acc_info = {
                                            "bad_road":
str(self.counter_sender),
                                            "accident":
str(accident_counter),
                                            "source_location":
str(self.source_location),
                                            "timestamp":
str(datetime.datetime.utcnow())
                                        }
```

```
                                        # insert into bad roads and
accidents db
                                        # br_acc_info_id =
self.br_acc.insert(br_acc_info)
                                        url_path =
'http://192.168.43.39:6030/anomalies/add'
                                        # ensure data being sent is
relevant
                                        # transmit information to
TSS
                                        requests.post(url=url_path,
json={"anomalies": br_acc_info})

                            # cv2.waitKey(0)
                            if cv2.waitKey(25) & 0XFF == ord('q'):
                                cv2.destroyAllWindows()
                                self.cap.stop()  # trial
                                break
                    # Handles pre-recorded video streams
                    elif self.socket_source is False:

                            image_np = self.cap.read()
                            image_np = cv2.cvtColor(image_np,
cv2.COLOR_RGB2BGR)
                            self.fps.update()  # trial

                            image_np_expanded =
np.expand_dims(image_np, axis=0)
                            (boxes, scores, classes, num) =
sess.run(
                                [detection_boxes, detection_scores,
detection_classes, num_detections],
                                feed_dict={image_tensor:
image_np_expanded})

                            # Visualization of the results of a
detection.

vis_util.visualize_boxes_and_labels_on_image_array(
                                image_np,
                                np.squeeze(boxes),

np.squeeze(classes).astype(np.int32),
                                np.squeeze(scores),
                                category_index,
```

```
                                use_normalized_coordinates=True,
                                line_thickness=5)

                            self.fps.stop()  # trial
                            # print("[INFO] elasped time:
{:.2f}s".format(self.fps.elapsed())) # trial
                            # print("[INFO] approx. FPS:
{:.2f}s".format(self.fps.fps())) # trial
                            cv2.imshow(self.source_name,
cv2.resize(image_np, (400, 300)))

                            '''
                            To get the statistics needed, we need to
count the number of cars and breaking
                            them down further, we need to obtain the
number of trucks, buses, motorcycles
                            and saloon cars(miscellaneous) that use
a particular road together with the
                            pedestrians and other aspects of
interest
                            '''


                            # count the number of pedestrians and
cars in the image
                            person_counter = 0  # count the
pedestrians
                            vehicle_counter = 0  # count all
vehicles
                            truck_counter = 0  # count only trucks
                            bus_counter = 0  # count only buses
                            car_counter = 0  # count other
miscellaneous cars
                            motorcycle_counter = 0  # count
motorcycles

                            # count the number of accidents and
bad_roads observed/detected in the image
                            accident_counter = 0  # count only
accidents occured
                            bad_road_counter = 0  # count only bad
roads detected

                            # report detected objects only if
confidence is greater than 50%
                            detected = [category_index.get(value)
for index, value in enumerate(classes[0])
```

```
                                    if scores[0, index] > 0.5]
                    # print('detected: {}'.format(detected))

                    # get the list of all detected objects
                    for ind_list in detected:

                        # get the values of the detected
objects in the list
                        ids = ind_list.values()

                        # loop through the values obtained
                        for i in ids:

                            # check for activated model and
act accordingly
                            if self.model_choice ==
self.model_2:


                                # acquire data for analytics
                                # for a car, bus or a truck
detected
                                if i == 'car' or i == 'bus'
or i == 'truck':
                                    vehicle_counter += 1

                                # for a pedestrian
                                if i == 'person':
                                    person_counter += 1

                                # for a truck
                                if i == 'truck':
                                    truck_counter += 1

                                # for a bus
                                if i == 'bus':
                                    bus_counter += 1

                                # for a car
                                if i == 'car':
                                    car_counter += 1

                                # for a motorcycle
                                if i == 'motorcycle':
                                    motorcycle_counter += 1
```

```
                                        # log results on terminal
for debugging
                                        if person_counter != 0:
                                            print('(PR) Pedestrians
count: {}'.format(person_counter))
                                        if vehicle_counter != 0:
                                            print('(PR) Vehicles
count: {}'.format(vehicle_counter))
                                        if car_counter != 0:
                                            print('(PR) Cars count:
{}'.format(car_counter))
                                        if truck_counter != 0:
                                            print('(PR) Trucks
count: {}'.format(truck_counter))
                                        if bus_counter != 0:
                                            print('(PR) Buses count:
{}'.format(bus_counter))
                                        if motorcycle_counter != 0:
                                            print('(PR) Motorcycles
count: {}'.format(motorcycle_counter))

                                        # prepare and format
information to be inserted into the database
                                        traffic_info = {
                                            "edgeID":
str(self.source_location),
                                            "noOfVehicles":
str(vehicle_counter),
                                            "noOfPedestrians":
str(person_counter),
                                            "noOfCars":
str(car_counter),
                                            "noOfTrucks":
str(truck_counter),
                                            "noOfBuses":
str(bus_counter),
                                            "noOfMotorcycles":
str(motorcycle_counter),
                                            "timestamp":
str(datetime.datetime.utcnow())
                                        }

                                        # insert into database and
get the traffic information id
                                        # general_traffic_info_id =
self.general_data.insert(traffic_info)
```

```
                                        url_path =
'http://192.168.43.39:6030/vehicleDetect/add'
                                        # ensure data being sent is
relevant
                                        # transmit information to
TSS
                                        #
requests.post(url=url_path, json={"detection": traffic_info})

                                    # checks for activated model and
works accordingly
                                    if self.model_choice ==
self.model_1:

                                        # for bad roads detected
                                        if i == 'bad_road':
                                            # DONE: SAVE THE BAD
ROAD IMAGE FRAME
                                            path =
'cus_anomalies/bad_roads'
                                            concat = "%s_%s" %
(self.anomaly_location.get("lat"),

self.anomaly_location.get("long"))
                                            name =
"badRoad_%s_%s.jpg" % (str(datetime.datetime.now()), str(concat))

scipy.misc.imsave(os.path.join(path, name), image_np)
                                            bad_road_counter += 1

                                        # for accidents detected
                                        if i == 'accident':
                                            # DONE: SAVE THE
ACCIDENT IMAGE FRAME
                                            path =
'cus_anomalies/accidents'
                                            concat = "%s_%s" %
(self.anomaly_location.get("lat"),

self.anomaly_location.get("long"))
                                            name =
"accident_%s_%s.jpg" % (str(datetime.datetime.now()), str(concat))

scipy.misc.imsave(os.path.join(path, name), image_np)
                                            accident_counter += 1
```

```
                                        if accident_counter != 0:
                                            print('(PR) Number of
Accidents Detected: {}'.format(accident_counter))
                                        if bad_road_counter != 0:
                                            self.counter_sender = 1
                                            print('(PR) Number of
Bad Roads Detected: {}'.format(bad_road_counter))

                                        # prepare and format
information to be inserted into the database
                                        br_acc_info = {
                                            "bad_road":
str(self.counter_sender),
                                            "accident":
str(accident_counter),
                                            "source_location":
str(self.source_location),
                                            "timestamp":
str(datetime.datetime.utcnow())
                                        }

                                        # insert into bad roads and
accidents db
                                        # br_acc_info_id =
self.br_acc.insert(br_acc_info)
                                        url_path =
'http://192.168.43.39:6030/anomalies/add'
                                        # transmit information to
TSS
                                        #
requests.post(url=url_path, json={"anomalies": br_acc_info})

                                # cv2.waitKey(0)
                                if cv2.waitKey(25) & 0XFF == ord('q'):
                                    cv2.destroyAllWindows()
                                    self.cap.stop()  # trial
                                    break

                    # end session
                    print('ended session')

# to handle  MongoDB's ObjectID data type and make it JSON
Serializable
class JSONEncoder(json.JSONEncoder):
    def default(self, o):
        if isinstance(o, ObjectId):
```

```
            return str(o)
        return json.JSONEncoder.default(self, o)

# Uncomment the line below to directly test this class
TertiaOculus(socket_source=False,node_id='gneE40').run()
```