September 2016

A Data Fusion Approach to Automated Decision Making in Intelligent Vehicles

Besat Zardosht
The University of Western Ontario

Supervisor
Prof. Michael Bauer
The University of Western Ontario

Graduate Program in Computer Science

A thesis submitted in partial fulfillment of the requirements for the degree in Doctor of Philosophy

Abstract

The goal of an intelligent transportation system is to increase safety, convenience and efficiency in driving. Besides these obvious advantages, the integration of intelligent features and autonomous functionalities on vehicles will lead to major economic benefits from reduced fuel consumption to efficient exploitation of the road network.

While giving this information to the driver can be useful, there is also the possibility of overloading the driver with too much information. Existing vehicles already have some mechanisms to take certain actions if the driver fails to act. Future vehicles will need more complex decision making modules which receive the raw data from all available sources, process this data and inform the driver about the existing or impending situations and suggest, or even take actions.

Intelligent vehicles can take advantage of using different sources of data to provide more reliable and more accurate information about driving situations and build a safer driving environment. I have identified five general sources of data which is available for intelligent vehicles: the vehicle itself, cameras on the vehicle, communication between the vehicle and other vehicles, communications between vehicles and roadside units and the driver information. But facing this huge amount of data requires a decision making module to collect this data and provide the best reaction based on the situation.

In this thesis, I present a data fusion approach for decision making in vehicles in which a decision making module collects data from the available sources of information and analyses this data and provides the driver with helpful information such as traffic congestion, emergency messages, etc.

The proposed approach uses agents to collect the data and the agents cooperate using a black board method to provide the necessary data for the decision making system. The Decision making system benefits from this data and provides the intelligent vehicle applications with the best action(s) to be taken.

Overall, the results show that using this data fusion approach for making decision in vehicles shows great potential for improving performance of vehicular systems by reducing travel

time and wait time and providing more accurate information about the surrounding environment for vehicles. In addition, the safety of vehicles will increase since the vehicles will be informed about the hazard situations.

Keywords

Intelligent Transportation System, Vehicle-to-Vehicle Communication, Vehicle-to-Infrastructure Communication, Cooperative Collision Warning and Rerouting System, Vehicle Tracking System, Road Side Unit

Acknowledgments

I would like to extend my gratitude to the many people who helped to bring this research project to fruition. First, I would like to express my special appreciation and thanks to my supervisor Professor Michael Bauer, you have been a tremendous mentor for me. Your support and advice on my research was priceless. It was an honor for me to work under your supervision and I really believe I was lucky for that and I do not have enough words to express my deep and sincere appreciation.

I would like to express the deepest appreciation to my co-supervisor Professor Steven Beauchemin, who without his guidance and persistent help this project would not have been possible. I am so deeply grateful for his help, professionalism and valuable guidance throughout this project.

I would like to thank the University of Western Ontario, the Faculty of Science and the Computer Science Department for providing me financial support as a Teaching Assistant and a Research Assistant.

Finally, I must express my very profound gratitude to my beloved husband for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without him. Thank you.

Table of Contents

List of Tables

List of Figures

List of Appendices

1 Introduction

The growing number of vehicles over the last few decades has affected our lives, particularly in urban areas. Increasing need for traveling more and more between different places has resulted in more traffic congestion, accidents, traffic delays and larger vehicle pollution emission. Since the conception of Intelligent Transportation Systems (ITS) in the 1980s, there have been many studies in this field. Several solutions have been introduced to overcome these driving problems and to make driving a much better experience. The research presented in this thesis aims at increasing safety and convenience in driving by introducing a decision making system which assists the driver in certain situations and provides necessary information about the surrounding environment, such as other vehicles on the road.

In recent years, many intelligent vehicle (IV) applications have been introduced to enhance driving safety. The range of applications for ITS is quite broad and applies to all types of vehicles. The usage of these applications varies from safety in driving to passenger entertainment. Generally, the most significant goals of IV applications are safety, productivity and traffic assistance.

There are different kinds of IV applications which are designed to assist the driver and increase safety in driving. Some important and well-known IV applications are:

- Adaptive Cruise Control
- Forward Collision Warning/Mitigation/Avoidance
- Vehicle Tracking Systems
- Lane/Road Departure Warning/Avoidance
- Parking Assist
- Stability Control
- Blind Spot Monitoring
- Pedestrian/Animal Detection/Warning.

However, these systems perform individually in the vehicle and do not benefit from integration of the information. That led us to design and implement an in-vehicle decision

making system which is an integrated system that can be installed in vehicles and collects data from all available sources, such as other vehicles, driver, vehicle, infrastructure, sensors, etc. and decides whether any action should be taken and if so what the best action to be taken should be. While there is large amount of existing research on different aspects of intelligent vehicle applications, little attention is devoted to making a central system for the vehicle to benefit from all information that is available.

1.1 Research Overview

The primary goal of this research is to study usage of an in-vehicle decision making system and its effects on the IV applications. In this thesis I describe this novel decision making system and the simulation which I have partially designed to implement the intended decision making system and four Intelligent Vehicle applications which use this decision making system to perform. The decision making system is designed in such a way to provide sufficient information to the driver either by warning the driver about an accident and suggesting an alternative route in order to avoid traffic congestion. I have used my decision making module to design some intelligent vehicle applications: a cooperative collision warning and rerouting system, a vehicle tracking system and an emergency message propagation system. Though I have designed and tested each intelligent system individually, all these systems can use the same decision making module at the same time. The performance of the system is evaluated in various circumstances such as heavy traffic versus light traffic, urban areas versus highways, different portion of vehicles equipped with our system, etc. In our context, I have examined the effects of using the decision making system in certain Intelligent Vehicles applications (cooperative collision warning and rerouting system, extended cooperative rerouting system using an accident model, vehicle tracking system and emergency message propagation system) understanding that all mentioned circumstances can have major effect on the way our system would perform.

1.2 Thesis Organization and Contributions

This thesis is a part of the ongoing research across the world concerned with Intelligent Vehicles. Our focus is on the idea of an intelligent decision making system for vehicles and how data from multiple sensors and input sources can be effectively utilized. In Chapter 2 I

introduce the decision making system which is the main contribution on this thesis, and its design and implementation using a simulator. The structure of this simulator is also explained in Chapter 2. In Chapters 3, 4, 5 and 6 I present published or submitted work addressing problems around intelligent decision making in different scenarios. My contributions with regards to each publication within the thesis are as follows:

Chapter 2: In this chapter the architecture and structure of my decision making system and its specification is explained. In addition, the simulation which I have used to test my work is described in this Chapter. My work on the simulator extended a previous simulator to incorporate the decision making module and components.

Chapter 3: Besat Zardosht, Steven Beauchemin and Michael Bauer, "A Decision Making Module for Cooperative Collision Warning System Using Vehicular Ad-Hoc Network". *The 16th International IEEE Annual Conference on Intelligent Transportation Systems, 2013*

The focus of this study is the design of a cooperative collision warning and rerouting system which benefits from our decision making system to advise the driver based on the driving situation. I designed and implemented this cooperative collision warning and rerouting system which uses a wireless agent's data to avoid accidents and to suggest alternative routes in case of an accident.

Chapter 4: Besat Zardosht, Steven Beauchemin and Michael Bauer, "A Cooperative Traffic Management System Using Accident Duration Prediction in Highway and Urban Areas". *Elsevier Vehicular Communication Journal (to be submitted)*

To make our cooperative collision warning and rerouting system more realistic I used an accident duration model to predict the duration of the accident based on accident conditions, such as the number of lanes blocked by the accident, the number of lanes of the road, the number of vehicles involved in the accident, etc.

Chapter 5: Besat Zardosht, Steven Beauchemin and Michael Bauer, "An In-Vehicle Tracking Method Using Vehicular Ad-Hoc Networks with a Vision-Based System". *IEEE International Conference On Systems, Man, And Cybernetics(SMC'14), 2014*

I proposed a new method of vehicle tracking as an application of the decision making module which uses both wireless-based and vision-based technologies together to track the vehicles. In my vehicle tracking method, each vehicle sends a map request via wireless to other vehicles in range and based on their response updates its own information.

Chapter 6: Besat Zardosht, Steven Beauchemin and Michael Bauer, “An Emergency Message Propagation System Using Roadside Units and Vehicle-To-Vehicle Communication”. *IEEE Smart Vehicles, 2016 (to be submitted)*

In this paper I have designed a novel Intelligent Vehicle system for emergency message propagation using both vehicle-to-vehicle communications and vehicle-to-infrastructure communications using our decision making system.

Finally, Chapter 7 offers a conclusion and outline path for future research.

2 Decision Making System for Intelligent Vehicles

The goal of an intelligent transportation system (ITS) is to increase safety, convenience and efficiency in driving. Besides these obvious advantages, the integration of intelligent features and autonomous functionalities on vehicles will lead to economic benefits, from reduced fuel consumption to efficient exploitation of the road network. Central to ITS, are decision making modules. These modules, as part of intelligent vehicles, can assess information about the infrastructure, environment and neighboring vehicles, sense the driver status and vehicle status and provide information to the driver so that they can make more reliable decisions in emergency situations or for the vehicle's own control systems, such as automatic braking, to take action.

While giving this information to the driver can be useful, there is also the possibility of overloading the driver with too much information. Existing vehicles already have some mechanisms to take certain actions if the driver fails to act. Future vehicles will need more complex decision making modules which receive the raw data from a range of available sources, process this data and inform the driver about the existing or impending situations and suggest, or even take actions.

In previous work, the sources of information are used separately and there is no central decision making module which considers all the information and makes decisions based on the overall situation. Vehicles may use both camera and wireless communication systems, but they do not cooperate. Intelligent vehicles can benefit from a central decision making module which collect the information from all sources and advises or warns the driver or even takes necessary actions.

I have focused on the design and implementation of a decision making module for vehicles which collects data from five available sources of information, can carry out

analyses of this data and provides the best action to be taken for certain applications. The proposed module uses four agents to collect data: vehicle agent, wireless agent (to collect Vehicle-2-Vehicle (V2V) and Vehicle-2-Infrustructure (V2I) information), driver status agent and camera agent (to provide images of the environment). Vehicle data contains all information about the vehicle itself namely speed, acceleration, engine status, etc. V2V data is the information about all other vehicles around our subject vehicle, V2I data is the information collected from roadway units such as emergency warnings propagated by central emergency system trough road side units (RSU) and finally driver data consist of all the information that can be collected about the driver namely, driver hearth beat, driver gaze, etc. Since both V2V and V2I data are collected using wireless technologies, I have used one agent for both.

I have designed and implemented four Intelligent Vehicle applications which use our Decision making system with data from its information agents to provide warning and suggestions for the driver. These Intelligent Vehicle systems are discussed in detail in Chapter 3 (cooperative collision warning and rerouting system), Chapter 4 (extended version of our CCW and rerouting system), Chapter 5 (vehicle tracking system) and Chapter 6 (Emergency Message Propagation system).

2.1 Related Work

In the following, I consider some related work that has been done on decision support modules and in the area of simulation for intelligent vehicle systems.

2.1.1 Collision Avoidance Systems

Decision making methods have been implemented and used in some sensor-based or vision-based collision avoidance systems. A method for decision making in collision avoidance applications was presented by Jansson et al. [14]. This method uses modern tracking theory along with a decision making module to avoid or mitigate a possible

accident. The prototype system they developed and evaluated significantly reduces the impact speed in frontal collisions [1]. The decision making model has to predict how the position of the tracked object evolves in time. This model is based on the coordinated turn model, where the object is supposed to follow straight line segment and circle segment.

A framework for a collision avoidance system is provided by Jansson using statistical decision making and stochastic numerical integration [2]. This system uses radar sensors to detect and track other vehicles. Since inaccurate sensor information can lead to uncertain state information and can influence the performance of collision avoidance system, a statistical decision making algorithm was developed to deal with estimation uncertainties by calculating the probability for each action.

The potential benefit of using sensor-based collision mitigation systems and the prediction uncertainties of these kinds of systems are two significant tradeoff issues which Hillenbrand [3] has tried to deal with. Hillenbrand has proposed a decision making approach to allow an intuitive tradeoff between potential benefit on one hand and readiness to take risk with respect to product liability and driver acceptability on the other hand. The performance of this system is investigated on three dangerous traffic situations: rear-end collision due to an unexpected braking; cutting-in vehicles; and crossing traffic at intersections.

Karlsson [4] has implemented a decision rule in a collision mitigation by braking (CMbB) system for late braking using a hypothesis test based on estimates of the relative longitudinal dynamics. The brake decision is based on estimates from tracking sensors. The required acceleration to obtain a zero velocity at a possible impact has been calculated for this statistical decision making system.

2.1.2 Simulation of Transportation Systems

To develop an intelligent transportation system, a reliable simulation environment is a key element. There have been some simulation environments developed in this area.

Gruyer has presented a cooperative system simulation architecture developed within the interconnection of the sensors simulation platform SiVIC (“Simulateur Véhicule-Infrastructure-Capteurs”, Vehicle-Infrastructure-Sensors Simulator) and the prototyping platform RTMaps (Real Time Multisensor Advanced Prototyping Software) [5]. The SiVIC simulator is interfaced in real-time with the RTMaps software which allows prototyping and testing ADAS (advanced driver assistance systems) and behavioral analysis applications in a simulated environment.

Eichler has presented a simulation environment which can be used to analyze the effect of real-time vehicle-to-vehicle warning message distribution applications on road traffic [6]. Three major components of this simulation are: the traffic simulator CARISMA, developed by BMW to simulate the traffic network; the network simulator NS2 to simulate mobile Vehicle-to-vehicle network; and a comprehensive ad-hoc agent for vehicle-to-vehicle warning message propagation.

Sommer [7] has developed a simulation framework that provides coupled network and road traffic simulation called Veins (VEhicles In Network Simulation). For network simulation, OMNeT++, a simulation environment free for academic use, was used to model realistic communication patterns of VANET nodes and traffic simulation is performed by the microscopic road traffic package, SUMO. Veins supports the active exchange of control and statistics data and also Veins provides a framework for the real-time interaction between the network simulation and the road traffic microsimulation. Both road traffic simulation and network simulation are bi-directionally coupled and simulations are performed on-line. This way, not only the influence of road traffic on network traffic can be modeled, but also vice versa. In particular, the influences of inter-

vehicle communication (IVC) on road traffic can be modeled and complex interactions between both domains examined. I use Veins as our starting point end extend it to accommodate our sensor and vehicle environment as well as incorporating our decision making module.

2.2 Decision Making System Structure

Intelligent vehicles can access different sources of data, such as vehicle data, data from other vehicles through wireless communication, data from cameras or other sensors, data about the driver and infrastructure data. To process this data a decision making module is presented which performs data collection and provides the best action to be taken in different situations.

I have designed and implemented an initial decision making module for vehicles which collects information from different sources using a blackboard method. The decision making module works in conjunction with one or more intelligent vehicle applications. The decision making module warns or suggest actions on some specific situation, for example our decision making module could suggest an alternative route as soon as its wireless agent warns about an upcoming accident (See Figure 2.1). In this system each source of data represents an agent in the system.

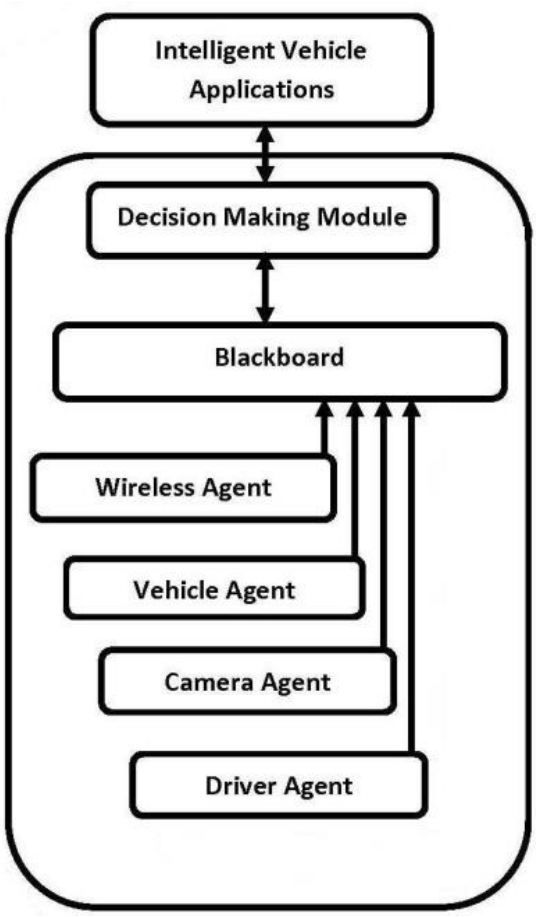

Figure 2-1: Decision Making Module Integration

A blackboard system is an application based on the blackboard architectural model, where a common knowledge base, the "blackboard", is iteratively updated by a diverse group of knowledge sources known as agents. Each knowledge agent updates the blackboard with partial information [8] .

A blackboard-system application consists of three major components:

- The knowledge sources (KSs). Each knowledge source provides specific expertise needed by the application. In our system each agent represents a knowledge source which provides part of the information.

- The blackboard is a shared repository of problems, partial solutions, suggestions, and contributed information. The blackboard can be thought of as a dynamic "library" of contributions to the current problems that have been recently "published" by other knowledge sources. A simple text file is used in our system as the blackboard to which all the agents write and read data from it.
- The control shell controls the flow of problem-solving activity in the system. KSs need a mechanism to organize their use in the most effective and coherent fashion which is provided by the control shell. While in our system the problem-solving activity is detecting emergency or critical situation in driving, the decision making module role is as the control shell which collects the blackboard information and decides whether an action is required.

Intelligent applications use the information provided by the decision making module to assist the driver in certain circumstances. Based on the data provided, the decision making module decides which agent should do what action(s). For instance, in case of receiving an accident message the wireless agent writes the location of the accident, the time of the accident, etc. on the blackboard. On the other hand, the vehicle agent provides GPS information and trajectory and writes this information on the blackboard. Subsequently, the decision making module reads the blackboard, extracts and processes the information and if the accident has happened on its road, provides an alternative route and asks the vehicle agent to change the vehicle trajectory. Also the decision making module asks the wireless agent to resend the wireless message if the number of hops is more than one. All information written to the blackboard contains a timestamp and the name of the agent which has provided that piece of information. In other words, all agents provide information to the decision making modules using the blackboard and the decision making module processes this information and sends an appropriate request to the corresponding agent(s). Without using the decision making module there would be no

integration of information and each agent could be taking action(s), if any, based on their own limited information.

I have designed and implemented the Wireless Agent, Vehicle Agent, Camera Agent but the driver agent is not designed and implemented. Four different intelligent applications which use the decision making module to operate; a cooperative collision warning and rerouting system, a cooperative collision warning and rerouting system using an accident model, a vehicle tracking system and an emergency propagation system using V2I and V2V. Using the decision making module in intelligent vehicle applications makes it possible to collect data and make decisions based on all of the data in order to have more accurate decision making. With the decision making module these applications can integrate together and benefit from machine learning methods to adapt to various circumstances. The details of the applications are explained in Chapter 3, 4, 5 and 6 and the implementation of the decision making module as well as camera agent, wireless agent and vehicle agent is explained in the next section.

2.3 Simulation

I have developed a simulation environment for inter-vehicle communication. To model the communication between VANET nodes, I have used OMNeT++ using the MiXiM framework. Road network simulations are performed using the Simulation of Urban Mobility (SUMO) package. The Vehicle in Network Simulation (Veins) simulator has been used to link OMNeT++ with SUMO [7]. Figure 2.2 shows the simulator components and flow of information between these components as well as how our decision making module is related to other modules in the simulation.

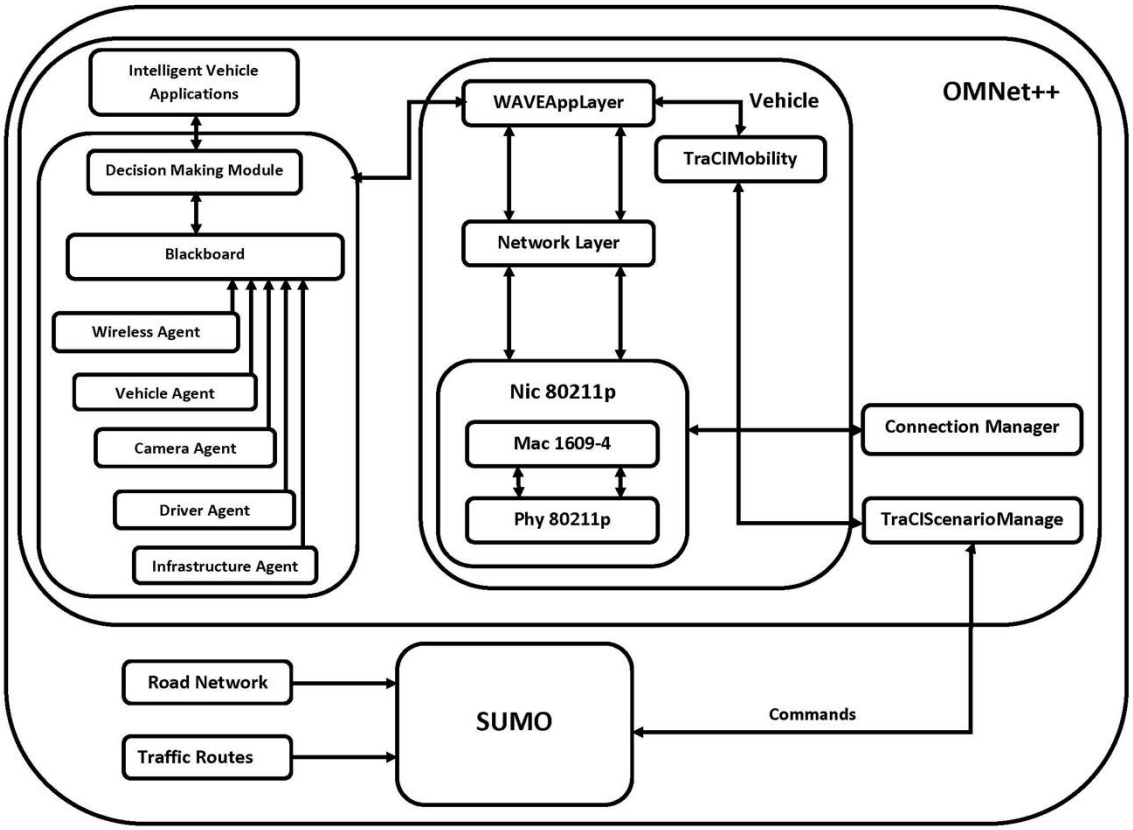

Figure 2-2 Simulator Components Integration

What follows briefly describes our road network simulator (Vehicle Agent), wireless module (Wireless Agent) and computer vision simulator (Camera Agent).

2.3.1 Road and Traffic Network Simulator (Vehicle Agent)

SUMO (Simulation of Urban MObility) is an open source, highly portable, microscopic and continuous road traffic simulation package designed to handle large road networks. In our work SUMO has been used to simulate road and traffic networks. Any selected part of a map can convert into a XML file representing the network of streets, roads, traffic lights, etc. Another XML file defines all the vehicles traveling in this network, their routes, speed, etc. Sumo obtains these XML files as inputs and simulates road and

traffic networks. To communicate with SUMO there are API calls (known as commands) available in the *TraCIScenarioManager* and *TraCIMobility* modules of Veins which each module can use to directly interact with the running traffic simulation (SUMO). In order to design our decision making module, I have used some of these commands and also have implemented additional commands which were not available in the original *TraCIScenarioManager* module or in *TraCIMobility* module. For this simulation I have added eight commands to the existing ones:

- commandReroutingByTravelTime: this command computes a new route using the vehicle's global edge travel time (the time to travel one specific street when there is no traffic jam) information and replaces the current route by the new route found.
- commandGetCurrentTravelTime: this command returns the travel time for the edge (graphical representation of the street) which the vehicle is currently on.
- commandGetEdgeTravelTime: this command returns the travel time for a specific edge.
- commandChangeEdgeTravelTime: this command changes the travel time for a specific edge.
- commandGetVehiclePosition: this command returns the position of any vehicle from SUMO.
- commandGetVehicleAngle: this command returns the direction of the vehicle based on its last step in the simulator.
- commandGetVehicleLength: this command returns the length of a given vehicle.
- commandGetVehicleWidth: this command returns the width of given vehicle.

Using these commands, the vehicles can react based on the messages they receive in the simulation and take action(s).

2.3.2 Wireless Module (Wireless Agent)

A multi-Channel IEEE 1609.4 and IEEE 802.11p Enhanced Distributed Channel Access (EDCA) model is implemented in Veins. This model encompasses the 80211.p Dedicated Short Range Communication[1] (DSRC) PHY and MAC layers, including Access Categories for QoS, the Wave Short Message (WSM) handling, and beaconing WAVE service announcements, as well as multi-channel operations, such as the periodic switching between the Control Channel (CCH) and Service Channels (SCHs) [9], [10].

The messages are transmitted with a bitrate of 8Mbps and a transmission power[2] of 20mW on the Control Channel (CCH). I model path loss[3] with path loss coefficient of 2.0 and shadowing with a mean signal attenuation of -89dB with a standard deviation of 4dB.

The wireless module could be installed in a vehicle as V2V communication enhanced system or can be installed as road way communication unit to propagate wireless messages on the road. The road side units are simulated using the same specification.

2.3.3 Vision Simulator (Camera Agent)

In order to simulate the use of cameras on the vehicles, I needed to have a vision module in our simulator which could act in a manner similar to a real camera installed in the vehicle.

---

[1] Dedicated short-range communications are one-way or two-way short-range to medium-range wireless communication channels specifically designed for automotive use and a corresponding set of protocols and standards

[2] Power transmission is the movement of energy from its place of generation to a location where it is applied to perform useful work

[3] Path loss (or path attenuation) is the reduction in power density (attenuation) of an electromagnetic wave as it propagates through space

Computer vision algorithms provide mathematical models of the world based on series of images captured by cameras. Since I am using a VANET and a traffic simulator, with access to a 2D mathematical model of the world which can be used to determine which vehicle or obstacle in this model would be visible to each camera. All vehicles are modeled as rectangles and therefore can be presented by four corner points. All buildings are modeled as a set of points which are the corners of the shape of the building. All cameras are specified by a) the angle specifying the area covered by each camera and b) the associated maximum distance over which they can accurately detect and position the objects.

Having shape and position of all buildings and vehicles from SUMO [11] and also the specification of each camera from the camera simulator, I present the following algorithm to determine all visible objects for each vehicle equipped with a camera:

1. All vehicles or buildings in which at least one of their points is in the observable area of the vehicle's camera will be determined.
2. Based on the camera position for each object, the two outermost points of any vehicle or building will be determined and that object will be defined by the line joining these two points.
3. For each object identified (represented as a line), all other objects which are placed completely or partially beyond it will be determined.
4. For all identified objects (step 2) the visibility percentage of each object is determined and any object for which its visibility percentage is less than 50% will be removed from the list.

Figure 2.3 illustrates how camera simulator works. The vehicles which are beyond other vehicles or buildings will be considered as invisible to the camera. In Figure 2.3, *vehicle1* is completely visible, *vehicle 2* in completely invisible, *vehicle 3* is partially visible but

less than 50% visible and would be considered as invisible and more than 50% of *vehicle4* is visible and it is considered as visible.

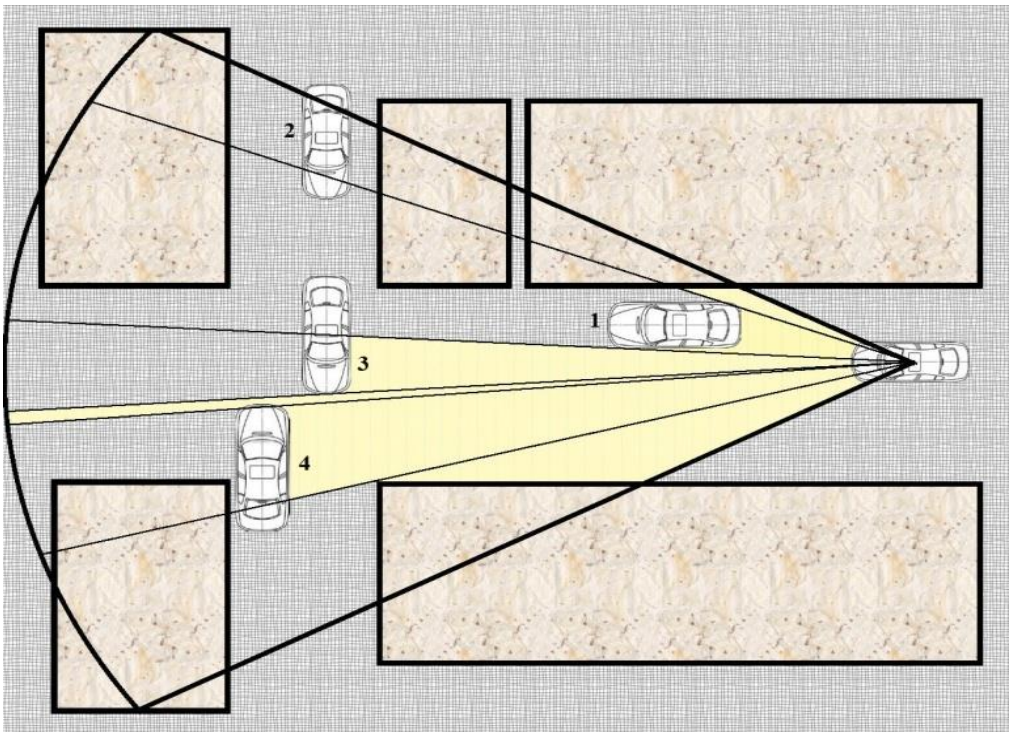

Figure 2-3: Camera Simulator

This approach is, of course, somewhat simplified as it does not take into account the vertical axis and creates the model in 2D. However, since most roads are relatively flat and all buildings are higher than most vehicles, identifying vehicles in 3D would be approximately the same as this approach. In addition, SUMO only operates in 2D so doing a 3D analysis is unfortunately not possible without changes to SUMO.

The code for vision simulator is in Appendix A.

3 Cooperative Collision Warning and Rerouting System

This Chapter is a reformatted version of the following article:

Besat Zardosht, Steven Beauchemin and Michael Bauer, “A Decision Making Module for Cooperative Collision Warning System Using Vehicular Ad-Hoc Network”. *The 16th International IEEE Annual Conference on Intelligent Transportation Systems, 2013*

In this paper I present a rerouting system that is an event based algorithm which informs other vehicles about an accident and can provide an alternative route to the driver in order to avoid traffic congestion. I assume that each car is equipped with GPS and wireless communication hardware that can implement our decision making algorithm and benefit from its rerouting algorithm. Our decision making system is an event based system so it just triggers when an event (*accident* message or *release* message from the accident) happens, and, therefore, does not consume much channel bandwidth.

Our intelligent application is a cooperative collision warning and rerouting system which uses our decision making module to inform other vehicles about an accident and provide an alternative route to avoid traffic congestion. Each car that is equipped with GPS and wireless communication hardware can implement our system and benefit from its rerouting algorithm. Our decision making system is an event based system so it just triggers when an event (accident message or release message from the accident) happens, therefore it does not consume much channel bandwidth. Overall, the results show that using our intelligent application shows great potential for improving performance of vehicular systems by reducing travel time and wait time for vehicles. In addition, the safety of vehicles will increase since the vehicles will be informed about the accident by wireless communication.

3.1 Introduction

Intelligent vehicles can benefit from exchanging data about driving situations, other vehicles, the surrounding environment and even the driver; and process this data to inform the driver about the existing or impending situations and suggest, or even take, actions.

Other than exchanging information about the overall state of the vehicles, vehicles can send wireless messages to each other in emergency situations, such as an accident, to warn other vehicles about the accident and decrease the possibility of danger from them. In addition to the obvious advantage of increasing safety, warning the driver in the accident situation can be helpful in decreasing the traffic in the accident area.

I have implemented a decision making module for vehicles which collaborates with data collection agents (vehicle agent for information about vehicle such as vehicle direction and trajectory, wireless agent which collects the information about other vehicles, etc.) to collect the information about other vehicles, informs the driver about it and suggests an alternative route in order to avoid the traffic caused by the accident. This decision making module has been implemented and tested using the Vehicles in Network Simulation (Veins) which uses OMNet++ [12], (wireless network simulation tool) linked to SUMO [11] (a road network simulation tool). Our decision making module has been tested in a city network based on Erlangen [7].

A cooperative collision warning (CCW)Intelligent vehicle system is one which makes use of data, including communication between vehicles, to enhance vehicle safety and warn drivers of potentially dangerous conditions. Our decision making approach contributes to research in the following ways:

CCW systems have mostly been used to provide warning information for the driver and do not suggest possible actions. These systems inform the following vehicle about the

potential collision, but do not provide rerouting choices for the driver which can help avoid traffic congestion. I have used my decision making algorithm in CCW and rerouting system to provide an alternative route for vehicles approaching the accident location in order to decrease waiting and travel time and avoid traffic.

It is the first event based CCW and rerouting system which has used a decision making system for a rerouting system based on wireless communication. Our decision making system triggers when an accident happens and the car which has been in the accident sends accident message(s). In this work, I assume that all vehicles are equipped with the communication hardware for vehicle to vehicle communication and wireless protocols, a GPS, maps of the roadways and street information, namely, the length of each street, maximum legal speed of each street, and our decision making algorithm. In other proposed rerouting algorithms, vehicles send request messages to other vehicles in order to find out about traffic congestion based on the responses. In our system there is no need to continue sending redundant messages and this reduces channel bandwidth by not sending unnecessary messages.

Our system uses a specific “resending” accident message alongside the propagating messages for one hop by receivers in order to make sure that all needed vehicles are aware of the accident and can take action to reroute to avoid the traffic jam caused by the accident. The wireless messages are tagged with hop number when a vehicle receives a wireless message which is tagged with a hop number bigger than zero, it reduces the number by one and propagates the message again. The vehicles which propagate the message again are known as hops.

3.2 Related Work

Using wireless communication among vehicles is a potentially useful way to make driving more intelligent. There have been several studies which have shown the beneficial use of wireless communication among vehicles in cooperative collision

warning (CCW) systems and in driving assistance systems. Some of this previous work is described in the following.

The technical feasibility of CCW systems was shown by Sengupta et al. [13]. In their paper, they introduced a CCW prototype that provides the driver with both warnings and situation awareness through displays provided in the vehicle. Their prototype has been tested in low speeds in an urban office campus with poor GPS coverage, and at high speed on an unused airfield. This prototype is the first prototype able to provide 360-degree awareness by using GPS and wireless communication. However, the warning system used in this prototype simply informs the driver about ongoing situation and does not suggest any alternative actions to take. In other words, the analysis of the information provided by the system is left to the driver. Also, this approach does not make use of a map and results in shortage of information about road geometry.

A DGPS (Differential Global Positioning System)-based vehicle-to-vehicle collision warning system is introduced by Tan [14] which requires a simple GPS unit and basic motion sensors to detect a possible collision situation. This system predicts the hazard situation using the information of nearby vehicles to provide safety but it covers a very small area around the vehicle so it cannot support traffic leading applications.

Dashtinezhad et al. have proposed the “Traffic View” system which gathers information about other vehicles and the environment through wireless ad-hoc communication among vehicles and provides traffic information that helps driving in situations such as foggy weather, or finding an optimal route in a trip several miles long [15], [16]. This system provides a map of the vehicles nearby. However, it does not have any prediction of their actions or any information about hazard situations.

Yung has proposed another vehicle to vehicle communication protocol for meeting delay constraints in cooperative collision warning systems [17]. In this protocol, if a vehicle faces a mechanical failure or unexpected road hazard, the warning system repeatedly

transmit the emergency wireless message to other equipped vehicle in range of 300m and by defined congestion control policies for emergency warning messages, a low emergency warning message delivery has been achieved.

Biswas has presented an overview of a highway cooperative collision avoidance (CCA) system, which is an emerging vehicular safety application using the IEEE- and ASTM-adopted Dedicated Short Range Communication (DSRC) standard [18]. In this paper it is assumed that all the equipped vehicles are aware of each other and communicate via wireless to warn each other about a collision.

Huang has proposed a joint rate-power control algorithm for broadcast of a self-information message that enables neighbor tracking in VANETs. This algorithm decides how frequently a vehicle should broadcast its own state information and how far the state information should be broadcast to obtain the best performance. This algorithm is evaluated through realistic network and microscopic traffic simulations [19]. However, sending the state information to other vehicles frequently can consume channel bandwidth.

Elbatt has studied the suitability of the standard DSRC protocol for inter-vehicle communication applications and, in particular, cooperative collision warning systems [20]. In this paper two novel latency metrics are introduced to calculate the performance of CCW system using the DSRC protocol: Packet Inter-Reception Time (IRT) at the vehicle for packets sent by a given transmitter and Cumulative Number of Packet Receptions at the vehicle from a given transmitter.

Lakas has proposed a traffic jam detection system which uses wireless communication for information exchange. In this system each vehicle periodically sends a request message to other vehicles and by their responses a vehicle can detect road congestions. Then a modified version of the Dijkstra algorithm can be used to find a better route for the requested vehicle [21]. The main problem with this system is that every vehicle has to

frequently send a request message to other vehicles in order to detect and avoid road traffic congestion.

Dogan has designed an intersection collision warning system using digital GPS location data and then broadcasts this information at a certain distance from the intersection using an ad-hoc wireless network [22]. This intersection collision warning system has been evaluated by a MATLAB-based simulator which consists of vehicle traffic simulator and wireless simulator.

3.3 Decision Making System for Cooperative Collision Warning and Rerouting

In this work, I assume that all vehicles are equipped with the communication hardware for vehicle to vehicle communication and wireless protocols, a GPS, maps of the roadways and street information, namely, the length of each street, maximum legal speed of each street, and the decision making algorithm. I have also assumed that the traveling route has been determined by the driver and that the decision making algorithm has access to the basic information of the travelling route.

Travel time is the approximate time that one vehicle needs to travel through that specific street and at a point in time it is calculated by the length of the street divided by the maximum legal speed of the street; a delay constant is added for each accident, if any, in progress on that street (Equation (3.1)). Since each car is provided with a map of the road it has access to travel time information of each street, and when it is informed about an accident on a specific street, it can change its local travel time information for that street. In this work any situation which causes the vehicle to stop unusually for a while would be considered as an accident. The Delay Constant is the mean delay (s) which an accident would cause for a vehicle.

$$TravelTime\ (S_i) = \left(\frac{length(S_i)}{Max\ Speed\ (S_i)}\right) + (DelayConstant * NumAccident(S_i)) \qquad (3.1)$$

When an accident happens, the vehicle which had the accident broadcasts an accident message containing the identifier for the type of message (*accident* or *release*) and location of itself. Then the travel time for the street on which the accident happened will increase by a specific amount (see Equation (3.1)).

The vehicles which receive the *accident* signal are divided into three different categories based on the location of the accident and their current locations: those not affected by the accident; those affected by the accident but can do nothing and those affected by the accident and can change route. The decision making module in a vehicle can determine the category of its vehicle by comparing the street on which the accident has occurred to the route provided by the driver for each vehicle.

The first category contains the vehicles where the street on which the accident happened is not in their trajectories. Therefore, they simply ignore the accident message and continue their journey. The second category contains the vehicles which are currently on the same street that the accident has happened. They may or may not be able to change their routes; however, they can reduce their speed to avoid the accident. These vehicles are often those that become stuck in the traffic. The last category of vehicles is those that are not currently on the same street that the accident has happened but that street is on their route. These vehicles can change their route to avoid the traffic jam created by the accident. The decision module will try to find a new route to avoid the accident and where the travel time is minimal.

Figure 3.1 shows a vehicle which is sending an *accident* message to other vehicles. The red circles indicate the vehicles which are stopped because of the accident, the green circles show the vehicles which are not affected by the accident (first category), the blue ones shows the vehicles which are on the same street as the one where the accident happened (second category) and the yellow circles indicate the vehicles which can reroute to avoid the accident (third category).

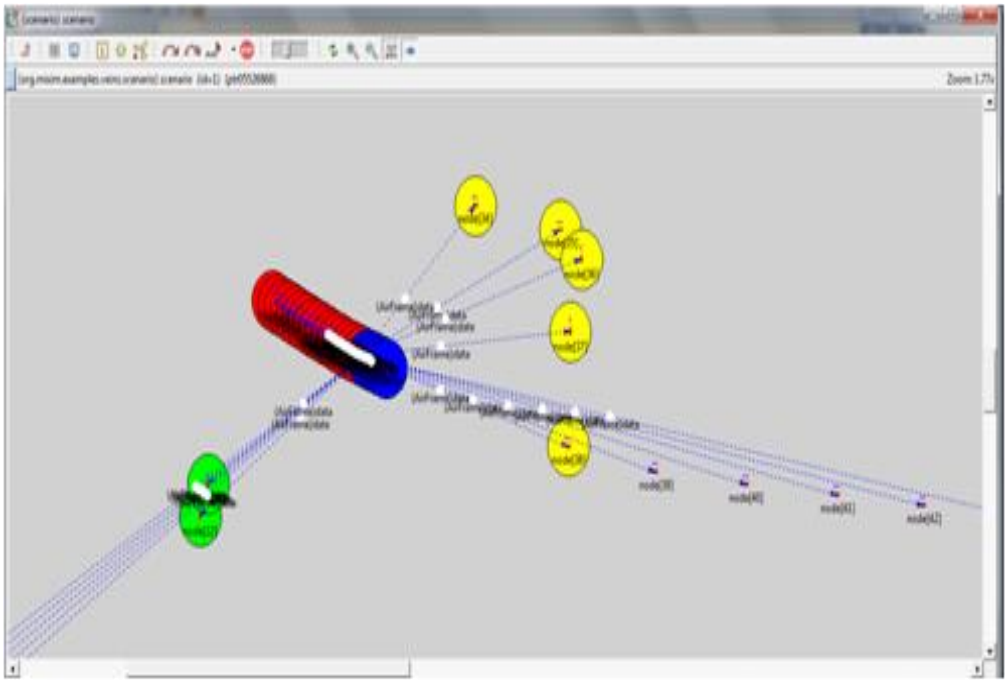

Figure 3-1. A vehicle sending an accident message to other vehicles

I also assume that when a vehicle is "removed" from the accident situation (by driving away or being taken away), it broadcasts a *release* message to other vehicles in the range and each vehicle which receives release message reduces the street travel time by the specific amount of delay constant.

The vehicles which receive the *release* signal are categorized as follows. Not affected; affected but can do nothing and finally, affected and can reroute. The first category contains the vehicles where the accident was not in their way and therefore they ignore the message. The second category contains the vehicles which are currently affected by the accident street and are stuck in the traffic. They ignore the release message as well. The last category contains the vehicles which are not currently on the street where the accident has happened, but this street is on their way. As they receive the release signal they calculate the best travel route based on new information and reroute if necessary. Again, these categories are based on the information about the route for each vehicle.

The vehicle which had the accident can send the *accident* message periodically rather than send it just once to inform upcoming traffic about the accident. In this case, more vehicles receive the message and take the proper action. The other method to inform more vehicles is message propagation. Each vehicle that receives the *accident* message or *release* message can propagate it to other vehicles in range.

By using the combination of these two methods more vehicles will be informed about the accident and therefore the decision making system can be more efficient and more reliable. Therefore, I can compare four versions of the decision making module based on how they propagate the accident message.

In the first and very simple version the car that has had the accident sends the accident message once and each other car which is stuck in the accident broadcasts this message once. In the second scenario, the car which had the accident sends the *accident* message once and each car which receives this message propagates it once. The next scenario is the case in which the car that had the accident sends the *accident* message periodically and no other vehicle propagates this message. Finally, in the fourth scenario the car which had the accident sends the *accident* message periodically and other vehicles which receive the *accident* message propagate it once.

The other question that should be addressed is how often should a message be resent and for how long should a car resend a message. The very first seconds of the accident are the most critical and making sure that all vehicles around are informed about the accident soon enough is very important. However, the more time that passes from the occurrence of the accident the less critical it would be to resend the message and after a while it is not necessary to resend the message again.

In our decision making module the car that had the accident sends a message every 2 seconds for the first minute, every 10 seconds for next minute, every 30 seconds for third

minute, every 60 second for forth minute and it stops resending the message after five minutes. The simulation environment and its details are discussed in next section.

3.4 Evaluation

In order to evaluate our decision making module I execute our simulation without using our decision making module and with this module and in four different scenarios. In the first and very simple scenario (Scenario1), the decision making module in the car which had the accident sends an *accident* message and each car which is stuck in the accident send this message again. In next scenario (Scenario2), after the car which had the accident sends the *accident* message, each vehicle which receives it will propagate it once. The third scenario (Scenario3) is the case in which the car that had the accident sends an *accident* message periodically but no other vehicle propagates it. And in last configuration (Scenario4) the car which had the accident sends the *accident* message periodically and any other vehicle which receives this message will propagate it once.

I calculated the waiting time for each vehicle, the overall travel time of each vehicle and the number of messages transferred between all vehicles in each case. Waiting time is the time the vehicles have been stopped due to an accident, a traffic light or even heavy traffic jam. One result of our approach is that the overall waiting time of the vehicles is reduced by 47% using Scenario1, it is reduced by 38% using Scenario2, 54% using Scenario3 and the best result was by using Scenario4 which reduced waiting time by 86%.

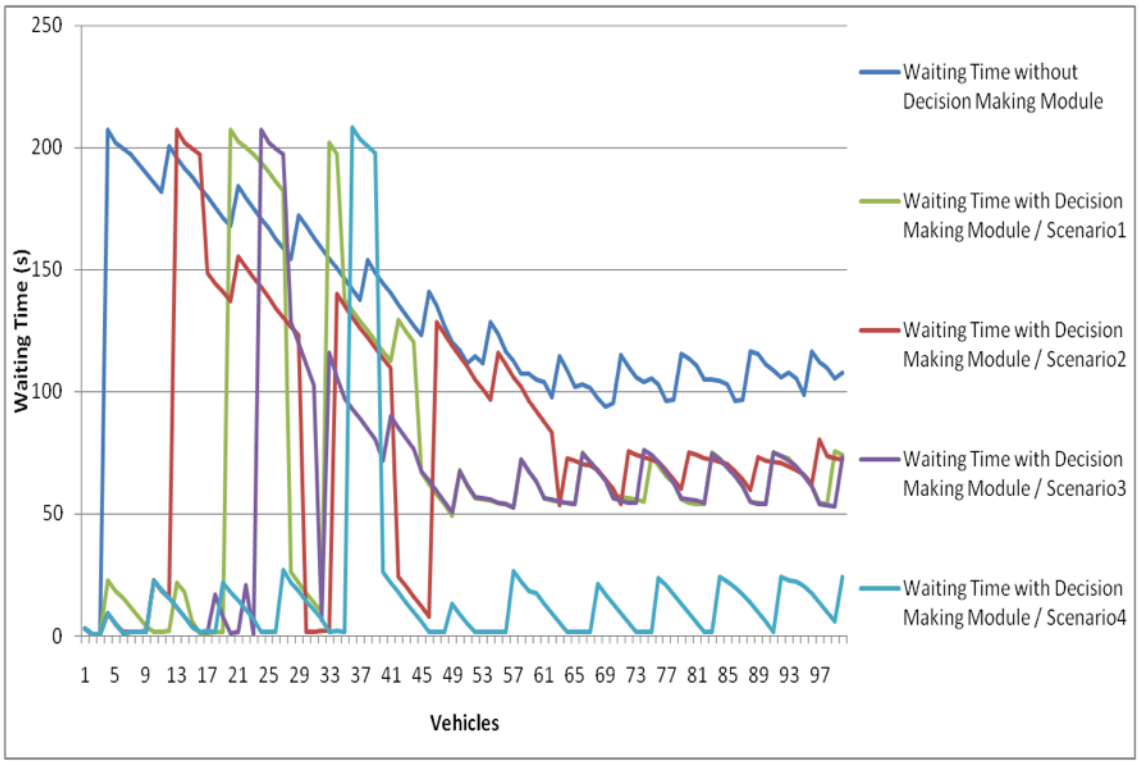

Figure 3-2. Waiting Time for each vehicle in the road network.

Figure 3.2 shows the waiting time(s) for each vehicle on the road when they use the decision making module in different scenarios and when they do not use decision making module.

Travel time is calculated by considering the time that the vehicle enters the network and starts its journey and the time it reaches its destination. From Figure 3.3, I can see that the average travel time for vehicles has been reduced significantly when they use our decision making module. The overall travel time has been reduced by 31% in Scenario1, it is reduced by 25% in Scenario2, 34% in Scenario3 and, more significantly, 52% in Scenario4 of the decision making module.

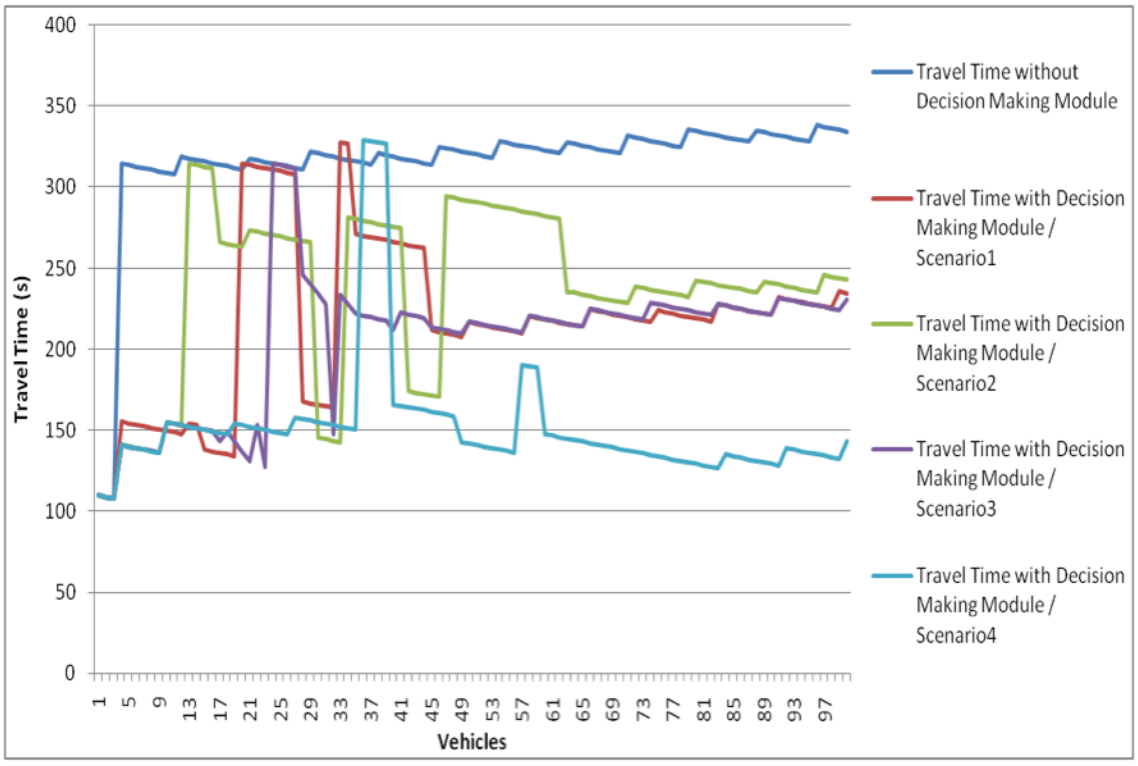

Figure 3-3. Travel Time for each vehicle in the road network

Table 3.1 shows the sums of waiting times and sums of travel times for all vehicles. It can be seen from this table that the least waiting time and travel time is obtained when Scenario4 of the decision making module has been used. In other words, when the car which had the accident sends an accident message periodically and each vehicle which receives the message propagates it, the overall waiting time and travel time of the vehicles reduced the most.

Table 3-1. Total waiting time and total travel time in different cases.

	Sum of Waiting Time (s)	**Sum of Travel Time (s)**
Without Decision Making module	13076.5	31618.6
Using Decision Making module-Scenario1	6886.191	21666.1
Using Decision Making module-Scenario2	8102.642	23638.8
Using Decision Making module-Scenario3	5957.396	20600.5
Using Decision Making module-Scenario4	1785.692	15133.4

The other factor to be considered is the number of messages which have been passed between the vehicles. In the simple case, when the vehicles do not use the decision making module, there is no message passing between the vehicles. But when they use the decision making module, they propagate wireless messages to inform other vehicles about the accident. The total number of messages that have been sent while using Scenario1 was 40 messages and the total number of received messages by all 100 vehicles was 995 messages. By using Scenario2 of decision making module, 996 messages were sent and 17414 messages were received in total. Scenario3 involved 58 sent messages and 910 received messages and finally by using Scenario4 372 messages were sent and 3453 messages have been received. In other words, the average of 0.4 message has been send by each vehicle and each vehicle has received approximately 10 messages during its journey in Scenario1, an average of 10 messages per vehicle were sent and average of 17.4 messages were received in Scenario2, approximately 6 messages were sent and 9 messages were received by each vehicle in Scenario3 and there are about 37 messages sent and 345 messages received per vehicle in Scenario4.

Table 3-2. Number of transferred messages in different cases

	Number of Messages Sent	**Number of Messages Received Messages**
Without using Decision Making module	0	0
Using Decision Making module-Scenario1	40	995
Using Decision Making module-Scenario2	996	17414
Using Decision Making module-Scenario3	58	910
Using Decision Making module-Scenario4	372	3453

The total number of transferred messaged among vehicles are shown in Table 3.2. Overall, the best result regarding travel time and waiting time is observed in using Scenario4 of the decision making module. The number of transferred messages between vehicles is small compared to the significant reduced amount of travel time and waiting time.

4 Cooperative Collision Warning and Rerouting System using an Accident Model

This Chapter is a reformatted version of the following article:

Besat Zardosht, Steven Beauchemin and Michael Bauer, “A Decision Making Module for Cooperative Collision Warning and Rerouting in Highway and Urban Area Using VANET”. *IEEE Transaction on ITS (ITSC'13), 2014* (to be Submitted)

Our cooperative collision management system is an event-based algorithm that informs other vehicles about accidents and provides alternative routes to avoid traffic congestion. Each vehicle equipped with GPS and wireless communication hardware can implement our decision-making algorithm and benefit from its rerouting capabilities. In addition, our system does not consume much channel bandwidth, due to its event-based implementation.

Since traffic situations vary between highway driving with high speeds and city areas with much lower speeds, I evaluated both cases and compared results. The decision-making module reduces waiting times in both driving environments. The results suggest that travel time can also be reduced in a city environment. However, this may not be the case in a highway environment and further investigation is needed.

4.1 Introduction

Providing drivers with relevant information about the environment surrounding their vehicle can assist them in making driving safer and easier. Wireless communication is one of the ways to obtain information on the status of surrounding vehicles such as their position and speed, for instance. With relevant information at hand, a driver can make

more reliable decisions and stands a better chance of reacting properly in emergency situations.

While providing this information to the driver can be useful, there is also the possibility of giving the driver unnecessary or extraneous information. Many existing vehicles already have mechanisms, which may take specific actions if the driver fails to act, such as dynamic cruise control and emergency braking. Future vehicles will undoubtedly feature more advanced decision-making modules capable of taking complex maneuvers in the event that a driver becomes incapacitated, or simply to drive the vehicle in autonomous mode.

Vehicles, other than simply exchanging information about their respective state with each other, can also send wireless messages in emergency situations, such as when an accident occurs, to warn other vehicles and decrease the possibility of danger for them. In addition to the obvious advantage of increasing safety, warning the driver of an accident situation can be helpful in decreasing the traffic congestion in the area of the accident.

I have implemented a Vehicular Ad-Hoc Network (VANET)-based decision-making module for vehicles that receives accident information from other vehicles, informs the driver about it and suggests an alternative route in order to avoid the traffic caused by the accident. This decision-making module has been implemented and tested using the Vehicles in Network Simulation (Veins) which uses OMNet++ [12], (wireless network simulation tool) linked to SUMO [11] (a road network simulation tool).

Our decision-making module has been tested in a road network of the city of Erlangen in Germany and in a road network of the 401 highway in Ontario, Canada. I also considered the impact of the decision-making module in situations where various proportions of vehicles are equipped with the system. Our proposed decision-making system uses an accident model to predict the duration of the accident based on factors such as its location, the number of lanes in the road, the number of lanes blocked by the accident,

the type of vehicles, and so on [23]. Using this model, the system estimates an accident duration based on the time passed from the accident and provides an alternative route for vehicles approaching the accident zone in order to decrease waiting and travel times and avoid traffic using VANET. This module uses an event-based decision making approach for vehicle rerouting and triggers only when accident messages are received. Hence, there is no need to continually send redundant messages. In addition, our system does not require a central management control mechanism; any so-equipped vehicle can provide the necessary warning messages to other surrounding vehicles. In case of an accident and in order to cover an adequately wide area, the module resends the relevant accident message periodically and propagates them for two hops by receivers in order to make other vehicles aware of the accident, such that they can reroute accordingly.

Our system is capable of detecting whether the driver is in an urban or highway environment, since driving on highways is a different activity from driving in urban areas. Our system also considers traffic volume. High traffic is detected based on the difference between the speed of the vehicle and the maximum legal speed of the road it is traveling on. Both of these capabilities do not use Vehicle to Infrastructure (V2I) communication.

Our contribution is structured as follows: Section 4.2 provides a survey of related work. In Section 4.3, the decision-making module is described in detail. In Section IV, the simulation environment is explained and in Section V, the results of the simulation of our decision-making module are examined. Lastly, Section VI provides some concluding remarks and future directions for this research.

4.2 Related Work

Wireless communication among vehicles is a potentially useful way to make driving more intelligent. There have been several studies that have shown the beneficial use of wireless communication among vehicles in cooperative collision warning (CCW)

systems and in driving assistance (DAS) systems. Another use of these cooperative systems is dynamic routing to avoid traffic congestion.

VANETs have been used widely in transportation systems to provide safety and convenience in driving. In particular VANETs have been used to develop Traffic Information Systems (ITS) in order to monitor traffic situations and detect congestion [24]. Traffic monitoring has been traditionally addressed with algorithms using the outputs of infrastructure cameras, or other similar sensors [25], [26]. Nowadays, loosely decentralized approaches to this problem (such as V2I and V2V) have become popular. For instance, a V2I traffic management system introduced by Milanes *et al.* [27] requires an intelligent traffic control station which manages all the incoming information from vehicles and, when the environment requires it, returns warning signals with the state of the traffic to all vehicles.

It has been recognized that V2V cooperative applications could provide better opportunities to monitor traffic congestion without depending on infrastructure. There have been many algorithms and methods developed to provide a view of the road, detect traffic congestion, and enhance driver decision-making regarding traffic without the need to communicate with infrastructure. For instance, a traffic management protocol presented by Santamaria *et al.* [28] uses V2V to inform nearby vehicles about accidents. In this approach the road network is considered as a graph and the weight of the arc in which the accident occurred is increased, therefore allowing vehicles to use a simple rerouting graph-based algorithm. The system proposed by Leontiadis *et al.* [29] detects traffic conditions through information gathered from other vehicles using V2V communication. The effectiveness of this decentralized traffic management system is evaluated using a realistic test case scenario. They have designed a method that allows the vehicles to dynamically reroute based on individually collected traffic information. Another method to reduce the impact of traffic jams via VANET is presented by Knorr using beacon messages which periodically broadcast status messages containing vehicle

speed, location and acceleration [30]. This system does require vehicles to be connected at all times and share their speed and location in order to evaluate the traffic situation.

The key problem of these systems is the limited bandwidth they must work with. Another factor to be considered in these systems is information age or how long the exchanged information remains valid. The problem of information age in vehicular network where vehicles periodically broadcast information is addressed in [31].

The effects of traffic density on rerouting is considered in [32]. In this approach, centralized and distributed routing methods are introduced and evaluated. In terms of time efficiency, it is found that in low traffic volume the distributed method is more suitable, while in heavy traffic the centralized method appears superior.

All of these methods have been developed to detect traffic congestion based on information gathered from other vehicles (V2V), infrastructure (V2I), or both. However, none of them predicts possible traffic congestion when a specific event such as an accident occurs. Other than predicting possible traffic congestion, intelligent vehicles would benefit from a method to estimate the delay caused by an accident.

Given the occurrence of an accident in the vicinity of a vehicle, if estimates of its location and duration could be obtained, then an optimized rerouting decision could be made. There are studies that attempt to predict the severity and duration of accidents using various models [23], [33], [34]. Our approach uses such a prediction model to estimate the duration of the accident and provide smarter rerouting for upcoming traffic.

4.3 Decision Making Module for Rerouting System Using an Accident Duration Model

In this work, I assume that some vehicles are equipped with the hardware for V2V communication and wireless protocols, a GPS, maps of the roadways, and street information, such as the length and maximum legal speed of each street and the number

of lanes in each direction. I also assume that the driver has determined the traveling route and that the decision-making algorithm has access to it via the on-board maps. Lastly, I assumed that the vehicle(s) involved in the accident simulations are equipped with our decision-making module.

I define travel time as the approximate time a vehicle needs to travel through a specific street and calculate it by dividing the length of the street by the maximum legal speed of the street. A delay $D(t)$ is added for accidents, if any, in progress on that street, as per:

$$T(s_i, t) = \left(\frac{L(s_i)}{M(s_i)}\right) + D(t) \quad (1)$$

where $T(s_i, t)$ is the travel time on street s_i, $L(s_i)$ is the length of s_i, and $M(s_i)$ is the maximum posted speed on s_i. Each vehicle is provided with a map of the roads it has access to and the travel time information for each street. When a vehicle is informed of an accident on a specific street, it can change its local travel time information for that particular street. I consider any situation that causes a vehicle to stop unusually for an extended period of time to be an accident, causing an increase of travel time for the street. The accident-induced delay $D(t)$ is determined with the help of Zong's accident duration model [23]. This model is based on an Accelerated Failure Time (AFT) metric employing a Weibull distribution to model the relationship between the time an accident has lasted already and the likelihood of it ending soon. Figure 4.1 indicates the baseline survival function (S_0) for accident duration.

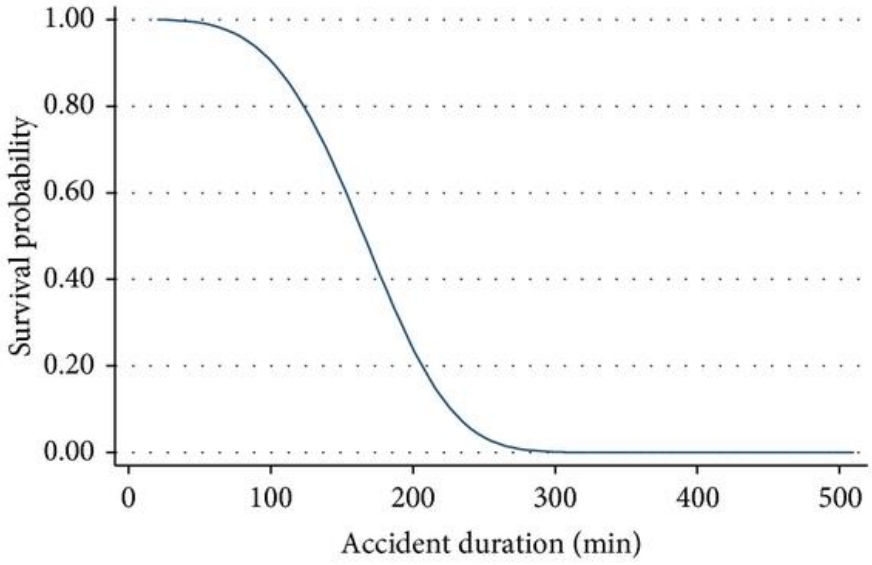

Figure 4-1. Estimated probability of survival versus accident duration [23].

With the baseline survival probability, the AFT model uses identified variables to provide an estimate of the duration of the accident. The survival function given the accident duration model is obtained with:

$$S(t) = s_0\left[te^{(-\beta^T X)}\right] \qquad (2)$$

where s_0 is the baseline survival function, X is a vector of accident attributes, and β is a vector of estimable coefficients for each accident attribute. These are given in Table 4.1.

I assume our decision-making module has access only to some of these parameters, namely: number of lanes blocked, accident location, and number of lanes in each direction. For other variables our system uses the provided default values (Table 4.1) to build the survival model.

Table 4-1. Accident severity variables with default values. These parameters are selected from previous research in [23].

Variable	Coefficient
Constant	5.12
Number of fatalities	0.51
Number of Injuries	0.33
Rear-end type collision	-0.37
Vehicle rollover	0.28
Number of lanes blocked	0.24
Bus involved	0.60
Truck involved	0.58
Debris involved	0.55
Hazard material	0.88
Weekend or festival	-0.14
Accident location	-0.57
Number of lanes in each direction	-0.18
Tow services	0.38

When an accident occurs the decision-making module creates the survival model then, with the survival function, it calculates the probability of the accident being cleared after a time t has passed from the time of the accident. To calculate the time to accident clearance (accident duration), I multiply the probability $S(t)$ by M_d, the mean accident duration time, which is the average of accident durations derived from real traffic accident data used to create the survival model [23]:

$$D(t) = M_d S(t) \tag{3}$$

When a vehicle experiences an accident, it broadcasts an *accident message* containing the identifier for the type of message (*accident* or *release*), its location, and estimated accident duration $D(t)$. Then it increases the local travel time for the street on which the accident happened by a delay equal to the predicted accident duration.

In this current study I am interested in understanding the potential impact of the decision-making module on traffic given different adoption rates for V2V. Hence, I assume that the vehicle involved in the accident is equipped with V2V and the decision-making module.

The vehicles that receive the accident signal are divided into three categories, based on the location of the accident and their current situation relative to it: those which are not affected by the accident (the street on which the accident occurred is not on their route); those which are already stuck in the congestion caused by the accident, and finally, those which are affected by the accident and yet have the opportunity to change routes and avoid the traffic congestion.

When a vehicle receives an *accident message*, the decision making module in the vehicle determines its category by comparing the street location on which the accident has occurred to the route provided by the driver and verifies if the route contains the specific street or not.

The first category contains the vehicles where the street on which the accident happened is not in their future routes. Therefore, they simply ignore the *accident message* and continue their journey. The second category contains the vehicles which are currently on the same street that the accident has happened. They may or may not be able to change their routes; however, they can reduce their speed to increase safety and avoid further accidents. These vehicles are often those that become stuck in the ensuing traffic. The last category of vehicles comprises those that are not currently on the same street that the accident has happened but that street is on their route. These vehicles can change their route to avoid the traffic created by the accident. The decision module tries to find an alternate route while minimizing travel time. I assume that when the accident situation is clear and the vehicle is removed from the accident area (by driving away or being taken away), it broadcasts a *release message* to other vehicles in the range and each vehicle that

receives the *release message* reduces their local street travel time to the default street travel time and reroutes if necessary. The vehicles that receive the *release* signal are categorized in the same way as when they received the *accident message*: not affected; affected but cannot reroute and finally, affected and can reroute.

In our simulations, the vehicle which experiences an accident sends the *accident message* periodically rather than just once to inform upcoming traffic about it. In this case, more vehicles receive the message and take proper action. I used an *adaptive resending method* that updates the *time to clear the accident* before resending the message. The system calculates a new accident duration time with respect to the time passed from the accident and creates new messages based on the updated accident duration. Therefore, the *time to clear accident* value decreases each time the accident message is resent by the system. The very first seconds of the accident are the most critical and making sure that nearby vehicles are informed about the accident soon enough is very important. However, the more time that passes from the occurrence of the accident the less critical it would be to resend the message. In our decision making module the vehicle involved in the accident sends a message every 2 seconds for the first minute, every 10 seconds for the next minute, every 30 seconds for the third minute, and every minute afterwards until it is released from the accident.

A message propagation technique is also in place in our simulations. Each vehicle that receives the *accident message* or *release message* propagates it to other vehicles in range. The other factor to be considered is to decide how far a message should go or on how many hops is enough when the message is propagated. This factor could differ based on traffic congestion, the average speed of the vehicle, and road type, (whether it is highway or urban). In our system, I considered two hops for highways and one hop for urban areas. Our earlier work [35] showed that this message resending and propagating approach was effective in reducing waiting time and in keeping the amount of communication relatively low.

I assessed the performance of our decision-making module by providing different simulation scenarios. The factors considered in the simulations are:

- Accident Location (highway, urban)
- Traffic (high, normal)
- Number of equipped vehicles or adoption rate of V2V in percent (0, 10, … ,100)
- Number of lanes on the road in the direction which the accident happened (1, 2, more than 2)
- Number of blocked lanes caused by the accident (1, 2, more than 2)

The results are summarized and analyzed in the following Section.

4.4 Evaluation

I simulated three accidents within the city environment: one involving 1 blocked lane in a street with only one lane, one accident involving 1 blocked lane in a street with 2 lanes and one with 2 blocked lanes in a street with 2 lanes. Similarly, I simulated four accidents within the highway environment, again involving different number of lanes (3 or 4) and different number of blocked lanes (1 or 2). This yields four lane-related accidents for each driving environment, and allows us to compare the impact of accidents across simulations. If the location of the accidents is left vary, then the predicted accident duration, which is based on accident information (highway or city, number of lanes, and number of blocked lanes), would be different in each case and render it difficult to study the impact of the adoption rate on the decision-making strategy.

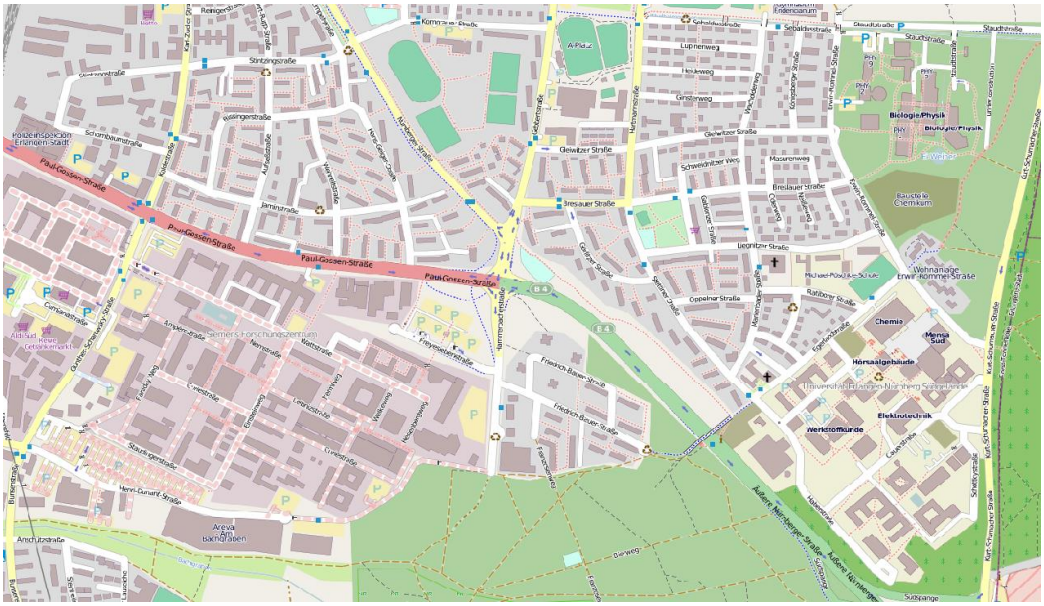

Figure 4-2. Map of Erlangen, Germany, is available from the OpenStreetMap project [7], [36].

I have executed the simulations with different proportions of vehicle equipped with the decision-making module, starting with no vehicle equipped with my system, then proceeding by increments of 10%, up to a 100% rate of adoption.

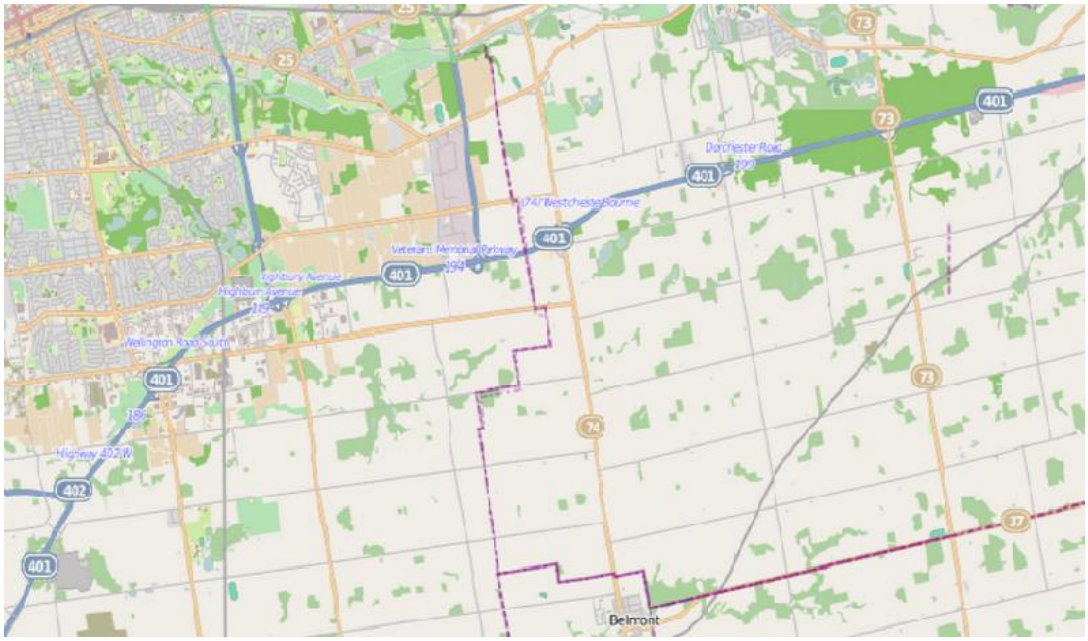

Figure 4-3. Map of highway 401, in Ontario, Canada, as available from the OpenStreetMap project [36].

I tested these different scenarios both in city and highway environments. In both driving environments, I considered normal and high traffic situations. I consider normal traffic as approximately 150 vehicles per square kilometer and 300 vehicles per square kilometer as high traffic. These numbers are based on my own empirical evaluation of the simulator.

Additionally, the accident scenarios provided additional parameters, such as the number of lanes on the road, the number of blocked lanes caused by the accident, and the accident duration. There are two choices of areas, two choices of traffic, eleven choices of adoption rates, two choices for the number of lanes, and two choices for the number of blocked lanes resulting in a set of 176 experiments (see Table 2). However, in the cases where there is only one lane on the road, the number of blocked lanes cannot be more than one and hence 22 cases are not applicable, yielding a set of 154 experiments.

Table 4-2. Different simulation parameters resulting in 154 experiments.

Experiment Factors	**Choices**
Area	City or highway
Traffic	Normal (approximately 150 vehicles per square kilometer) or High (approximately 300 vehicles per square kilometer)
Adoption Rate	By increments of 10%
Number of Lanes	1 or 2 in city and 3 or 4 on highway
Number of Blocked Lanes	1 or 2 (for roads with 2 or more lanes)

I calculated the waiting time (delay) and the total travel time for each vehicle and the number of messages transferred between all vehicles in each case. Waiting time is the time the vehicles have been stopped due to an accident, a traffic light or by heavy traffic.

Figure 4.4 and Figure 4.5 illustrate the waiting times for the different adoption rates on highways and in city environments. I observe that the average waiting time for the vehicles in both normal and high traffic conditions are reduced by higher adoption rates of my system.

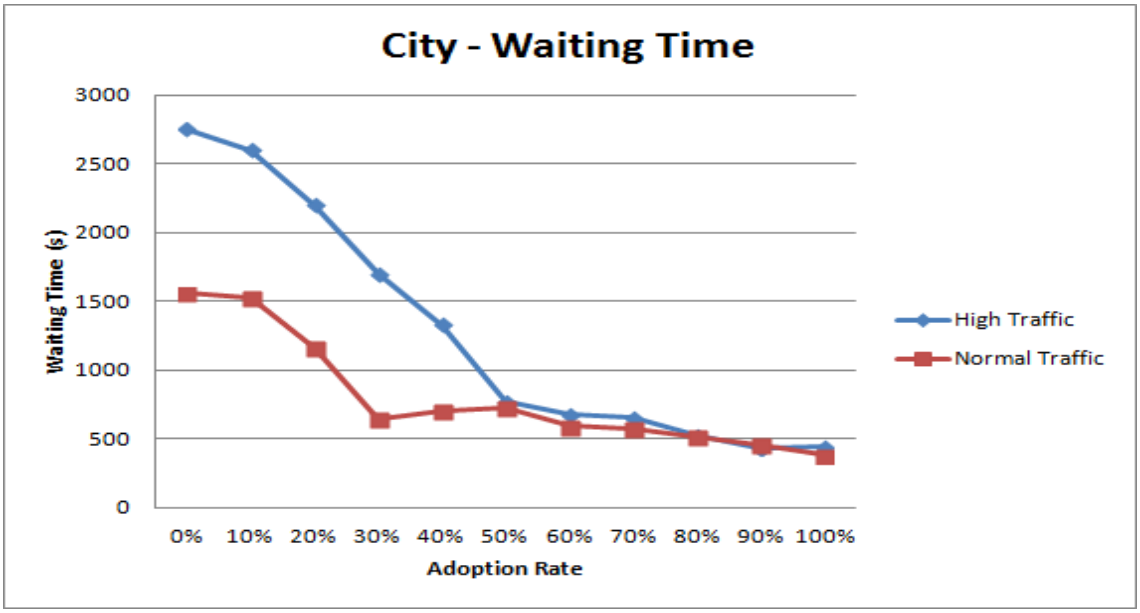

Figure 4-4. Average waiting time in urban area with different proportion of vehicles equipped with my system.

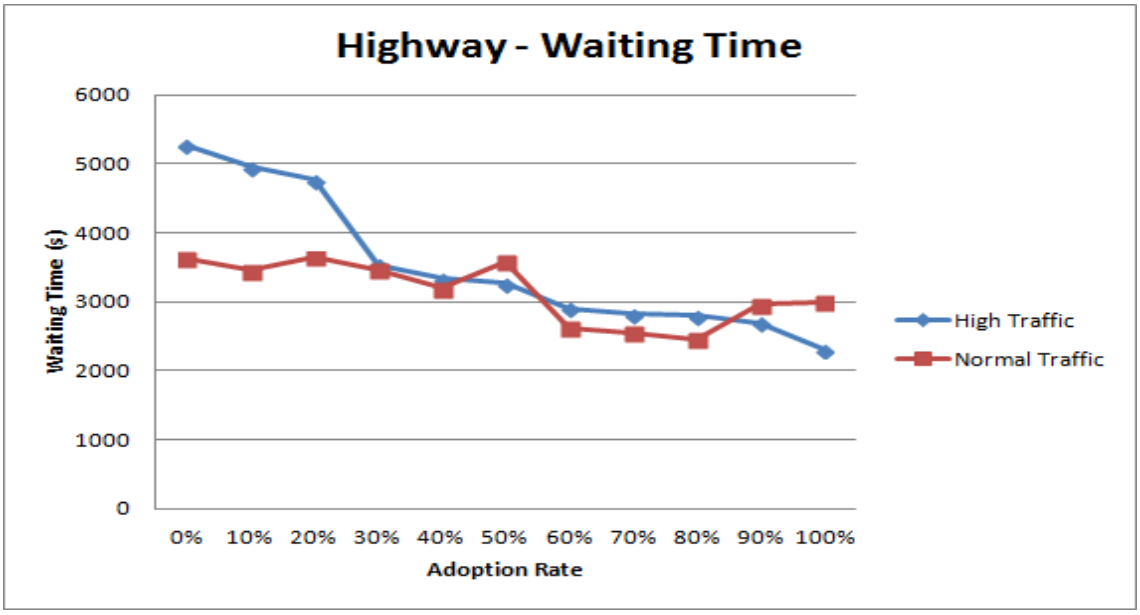

Figure 4-5. Average waiting time in highway with different proportion of vehicles equipped with my system.

A similar result is observed for travel time[4] in urban environments as adoption rates increase (see Figure 4.6). As shown in Figure 4.7, the travel time in a highway environment under both normal and high traffic is increased slightly. Since alternative routes for highways could take longer to travel onto (non-highway roads with possibly lower maximum legal speed limits), this result is not surprising.

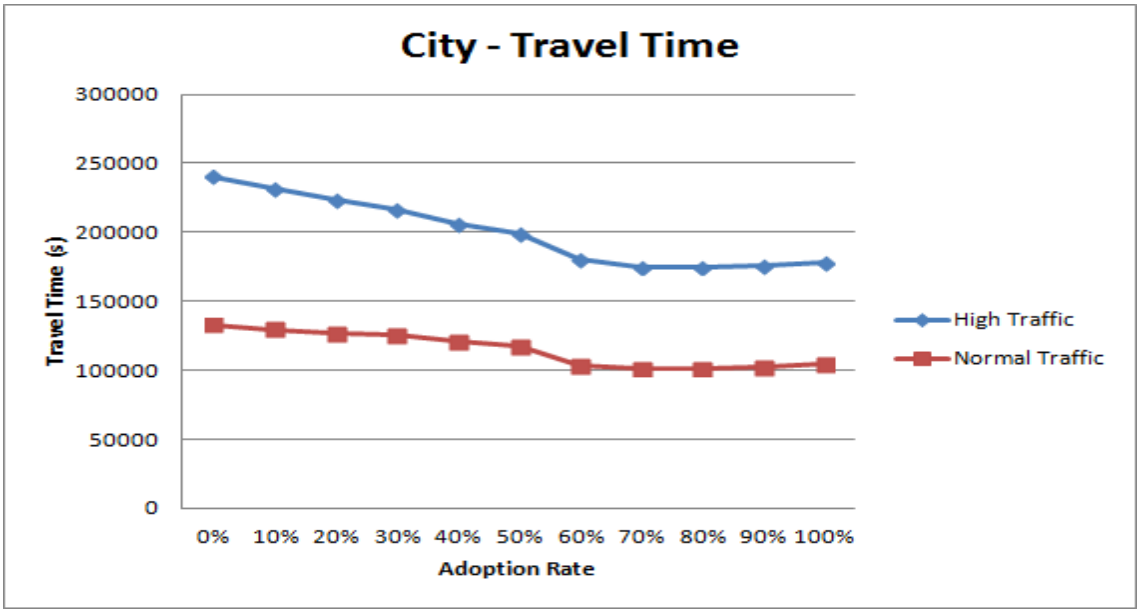

Figure 4-6. Average Travel Time in urban area with different proportion of vehicles equipped with my system

[4] Travel time is computed by considering the time a vehicle enters the network and starts its journey until the time it reaches its destination.

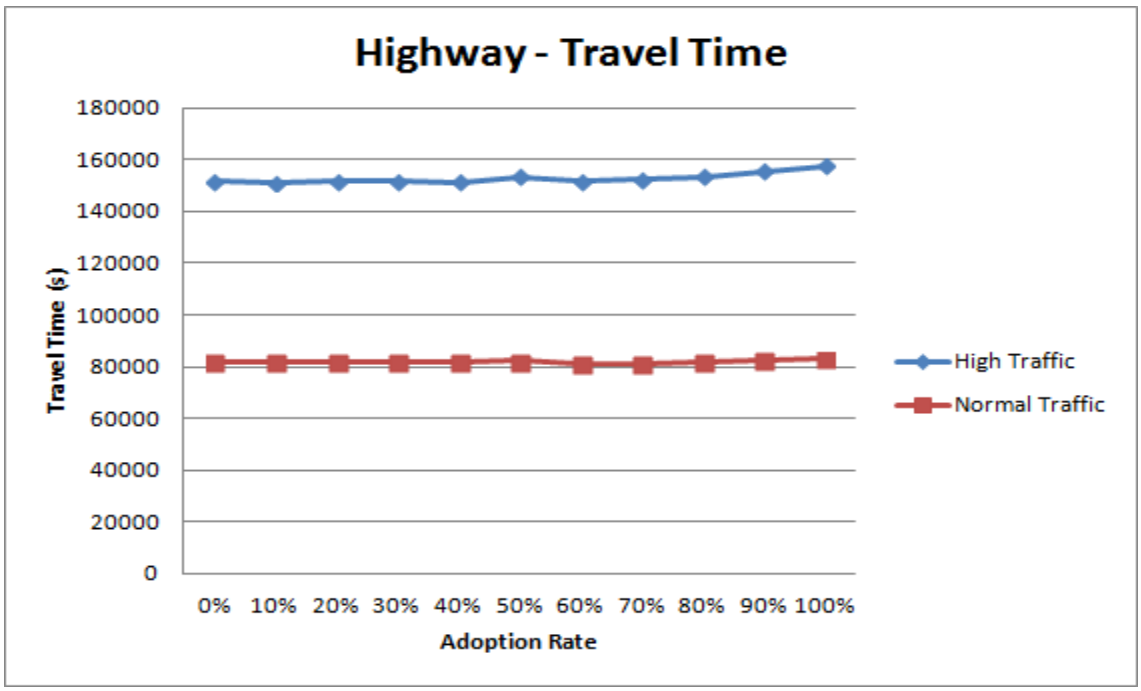

Figure 4-7. Average Travel Time in highway with different proportion of vehicles equipped with my system

The other factor of importance in the different scenarios is the number of messages exchanged by the vehicles. In the simplest case, when vehicles are not equipped with the decision-making module, there is no message passing between them. However, when vehicles use the decision-making module, they propagate messages to inform others about accidents.

The average number of sent messages when the vehicles were traveling within a city area in normal traffic is shown in Table 3.2. When 100% of vehicles are equipped with the decision-making module, an average of 0.9 messages were sent per vehicle and each vehicle received an average of 16.5 messages during its journey.

Table 4-3. Average number of transferred messages in normal traffic within city areas

Adoption Rate	Number of Sent Messages	Number of Received Messages
0%	0	0
10%	30	62.33
10%	45.33	168
30%	63.33	366.66
40%	81	605.33
50%	90	825.66
60%	100.66	1068.66
70%	112.33	1376
80%	123.33	1779.66
90%	135.33	2213.33
100%	142.33	2482.33

Table 4-4. Average number of transferred messages in high traffic of city area

Adoption Rate	Number of Sent Messages	Number of Received Messages
0%	0	0
10%	55.66	296.66
10%	93.33	780.66
30%	127.66	1287.66
40%	161	1782.66
50%	191.66	2131.66
60%	189	2372.33
70%	198	2619.66
80%	166.33	2497
90%	179.33	2833.33
100%	190.33	3228

Table 4.4 shows the average number of sent and received messages in a high-traffic urban area per adoption rate. An average of 0.6 messages were sent by each vehicle and

each vehicle received an average of 10.7 messages during its journey at the 100% adoption rate.

Table 4-5. Average number of transferred messages in normal traffic on highway

Adoption Rate	Number of Sent Messages	Number of Received Messages
0%	0	0
10%	80.25	459
10%	137.5	1157.75
30%	177	1764.75
40%	232	2528.5
50%	255.25	3194.5
60%	324	4095.25
70%	358	5236.75
80%	381	6407.25
90%	451.5	7225.25
100%	455	7896.75

Table 4-6. Average number of transferred messages in high traffic on highway

Adoption Rate	Number of Sent Messages	Number of Received Messages
0%	0	0
10%	84.5	527
10%	138.75	1173.75
30%	206.75	2381.5
40%	274.25	3586
50%	346	4604.75
60%	369	5401.5
70%	412	6601.25
80%	449.75	7533.75
90%	501.25	8234.5
100%	554	9469.75

Table 4.5 and Table 4.6 show the total number of transferred messages in the highway environment for normal and high traffic volumes. In this environment, the average number of sent and received messages turns out to be less than in urban environments.

The results are summarized in Table 4.7 for the 100% adoption rate in order to compare different scenarios. In general, I observe that there are fewer messages transferred within the urban environment than on the highway. This is because in the city messages propagate for one hop as opposed to two hops on the highway. In addition, in higher traffic density areas of the highway, more messages are transferred between vehicles due to their relative proximity.

Table 4-7. Number of sent and received messages when all vehicles are equipped with the decision-making module

	Number of Sent Messages	**Number of Received Messages**
City – Normal Traffic	142.33	2482.33
City – High Traffic	190.33	3228.00
Highway – Normal Traffic	455.00	7896.75
Highway – High Traffic	554.00	9469.75

In general, the results show that in urban environments, both travel times and waiting times are reduced with increasing adoption rates. On highways, using my system results in less waiting time, at the price of a slightly increased travel time. The number of transferred messages per vehicle remains small and manageable across the set of simulations.

5 Vehicle Tracking System

This Chapter is a reformatted version of the following article:

Besat Zardosht, Steven Beauchemin and Michael Bauer, “An In-Vehicle Tracking Method Using Vehicular Ad-Hoc Networks with a Vision-Based System”. *IEEE International Conference On Systems, Man, And Cybernetics(SMC'14), 2014*

My vehicle tracking system integrates a vision based tracking system with wireless based tracking system. The approach seems to have the benefits of both technologies while avoiding their disadvantages. I evaluated the system via simulation and it shows potential for improving performance of intelligent driving assistance systems making use of information about the surrounding vehicles’ locations. The results show that the system can perform well even if a small percentage of the vehicles are equipped.

5.1 Introduction

Having information about the environment surrounding a vehicle can assist a driver in driving safer and making driving more convenient. Different sources of information can be available for use by vehicles. Having knowledge of neighboring vehicles can provide useful information for intelligent vehicle (IV) applications, such as collision warning systems or alternative route planning systems. With such information, the driver can make more reliable decisions and has a better chance of reacting properly in emergency situations.

Using wireless communication is one of the ways to obtain information about the vehicle’s environment, and, in particular, the status of other vehicles, such as their location, speed and other data. Vehicles can exchange information with other vehicles and inform them about their location, speed, acceleration, etc. Having this information

gathered from other vehicles a vehicle can locate neighboring vehicles. Such exchanges of information about neighboring vehicles are constrained to vehicles within the range and messages can be interfered with, as in an urban environment.

On the other hand, vehicles can also benefit by using cameras as another source of information to monitor the road and nearby traffic. Vehicle tracking via image processing systems is done by mounting cameras on the vehicles which provide images to a processing system to recognize other vehicles. Depending on the cameras and image processing, vehicles at some distance can be detected and even properties of those vehicles, such as their speed, can be determined. This information could augment the information being exchanged among nearby vehicles and even propagated to other vehicles. However, if a vehicle is occluded or partially occluded, the cameras may not be able to detect it and it would be out of the view. Wireless communications between vehicles could augment such vehicle identification.

Using cameras to capture elements of the surrounding environment and tracking the neighboring vehicles provides the technology with valuable information which can be used in many different situations and for many different applications. As noted, the main shortcoming in using cameras for tracking vehicles is that they can provide the information about the vehicles only in their sight and in lots of situations. On the other hand, vehicle-to-vehicle communication can provide driving assistance systems with more information about position, speed and directions of nearby vehicles regardless of their visibility. However wireless based methods also have some limitations. Not all the vehicles may be equipped with wireless communication facilities, messages may experience interference and there could be other objects, like pedestrians, animals, etc., which cannot report their status using wireless systems, although information about their location could be critical.

To overcome some of these shortcomings, I propose a new method of vehicle tracking which uses both technologies together to track the vehicles. In my vehicle tracking method, each vehicle sends a map request via wireless to other vehicles in range and based on their responses it updates its own information. I also assume that not all the vehicles are fully equipped and consider the implications.

I have implemented a Vehicular Ad-Hoc Network (VANET)-based vehicle tracking method for vehicles which receives the camera information from other vehicles and uses its own camera information to match the received information and to update its own information. This vehicle tracking method has been implemented and tested using the Vehicles in Network Simulation (Veins) which uses OMNet++ [37], (wireless network simulation tool) linked to SUMO [38] (a road network simulation tool) and a camera simulator which mimics camera operation which is mounted on a vehicle. My tracking method has been tested in a city network based on Erlangen [39] and in highway network based on 401 highway in Ontario [36]. My vehicle tracking method contributes to research in the following ways:

- Previous tracking methods have used either wireless communication or vision based systems to detect neighboring objects and to provide a view of the surrounding environment. In the proposed tracking system, both of these technologies have been used to overcome their respective limitations and provide more reliable and more accurate information about the objects around the vehicle.
- My vehicle tracking method does not rely on other vehicles and it can work in the situation in which no vehicle around is equipped with wireless technology. In this case the system just uses its own information obtained from its cameras and a vision based tracking system. Generally, my system can work if all the vehicles are equipped with both camera and wireless communication, with just wireless or neither.

- My system works well in specific traffic situations, such as an intersection or in low light situations, where other tracking methods cannot operate well.

This paper is structured as follows. I present related work on which this paper is based in Section II. In Section III, the vehicle tracking method is described. In Section IV the simulation environment is explained and in Section V, the results of a simulation of my vehicle tracking method are examined. Finally, Section VI provides some concluding remarks and future directions for this research.

5.2 Related Work

In this section, some of the previous works in the field of vehicle tracking are discussed. I also review simulation environments for vehicles on roads.

Using one or more cameras to detect and track neighboring vehicles is a common way to provide necessary information for many different intelligent transportation applications such as forward collision warning systems, travel management systems, etc. A vision based vehicle detection and tracking system was presented by Coifman [40], [41]. This tracking system was designed to operate under challenging conditions, such as various lighting conditions. In this vision based tracking system, instead of tracking an entire vehicle, vehicle features are tracked which makes the system less sensitive to the problem of partial visibility.

Another vision based vehicle tracking system has been presented by Bertozzi which detects and tracks vehicles based on a monocular image sequence [42]. Betke has also introduced a vision based tracking system which recognizes and tracks multiple cars in hard real time from sequences of images [43]. Alin [44] has presented a vision based tracking system which uses the street information and attractor-based adjustment of the probabilistic forward prediction in a Bayesian grid filter to track other vehicles.

A real time object tracking approach for the design of a video based freeway traffic monitoring system was proposed by Gloyer [45]. The tracking algorithm operates based on mapping the detected vehicles onto the real 3D scene. The proposed tracking algorithm makes an estimate of expected position of the vehicles as well as tracking all the vehicles on the road [45].

Other than using a camera to capture surrounding environment, wireless communication among vehicles can also provide the information for vehicle tracking systems.

Rezaei et al. introduced four different schemes for tracking neighboring vehicles with the use of wireless communications. Based on these schemes, each vehicle broadcasts its GPS position, speed and heading to other vehicles via wireless communication. In their first scheme, the sender broadcasts its information every 100ms and the receiver assumes that the sender remains constant until reception of the next message. The second scheme provides the receiver with a model estimator which estimates the position of the sender based on the model and the received information. In the third scheme, the sender uses a model estimator as well as the receiver. Finally, in the fourth scheme the sender repeats its message a few times within a short time window [46].

Shafiee introduced a routing protocol for vehicular ad hoc networks which uses a vehicle tracking method to position neighboring vehicles [47]. In this vehicle tracking method, vehicles send beacons reporting their position to other vehicles. Based on the information obtained from neighboring vehicles, each vehicle can calculate the density of vehicles in the network and select the adequate route to communicate via VANET.

A joint rate-power control algorithm for broadcast of self-information that provides vehicle tracking is presented by C. Huang [48]. This algorithm performs based on two modules, a rate control module which decide how frequently a vehicle should broadcast its information, and a power control module which determine how far the information should be broadcast.

Fallh has introduced a cooperative tracking method for vehicles, which uses the state information of neighboring vehicles broadcast by themselves and provide an estimation of their locations on the road. The effect of different choices of rate and range of the transmission on such kinds of tracking system is analyzed [49].

The most significant problem of vision based vehicle tracking systems is that these systems do not have any information about the vehicles or other objects which are not in camera's field of vision, especially near intersections. Also, object detection with use of a camera depends on the lighting in each situation. In contrast, communication among vehicles can provide position information of the vehicles within communication range or even propagate that information. But vehicles out of range or without communications capability are not trackable. In contrast, cooperative vehicle tracking systems can be considered to overcome these problems. I have combined both vision based systems and wireless systems and introduced a new vehicle tracking method and tested it in a simulation environment.

Having a reliable simulation environment is a significant element in the development and evaluation of an intelligent transportation application. There has been a number of different simulation environments developed in this area.

Gruyer has presented a cooperative system simulation architecture developed within the interconnection of the sensors simulation platform SiVIC ("Simulateur Véhicule - Infrastructure - Capteurs", Vehicle – Infrastructures - Sensors Simulator) and the prototyping platform RTMaps (Real Time Multisensor Advanced Prototyping Software) [5]. The SiVIC simulator is interfaced in real-time with the RTMaps software which allows prototyping and testing of ADAS (advanced driver assistance systems) and behavioral analysis applications in a simulated environment.

Eichler has presented a simulation environment which can be used to analyze the effect of real-time vehicle-to-vehicle warning message distribution applications on road traffic

[6]. Three major components of this simulation are: the traffic simulator CARISMA, developed by BMW to simulate the traffic network; the network simulator NS2 to simulate mobile vehicle-to-vehicle network; and a comprehensive ad-hoc agent for vehicle-to-vehicle warning message propagation.

C. Sommer has developed a simulation framework that provides coupled network and road traffic simulation called Veins (vehicles in network simulation) [7]. For network simulation, OMNeT++, a simulation environment free for academic use, is implemented to model realistic communication patterns of VANET nodes and the traffic simulation is performed by the microscopic road traffic package, SUMO. Veins supports the active exchange of control and statistics data and also Veins provides a framework for the interaction between the network simulation and the road traffic micro-simulation. Both road traffic simulation and network simulation are bi-directionally coupled and simulations are performed on-line. This way, not only the influence of road traffic on network traffic can be modeled, but also vice versa. In particular, the influences of inter-vehicle communication (IVC) on road traffic can be modeled and complex interactions between both domains examined. I have used Veins as the basis of the current research.

5.3 Vehicle Tracking Method

As noted, both vehicle-to-vehicle communications and using cameras for tracking vehicles both have plusses and limitations. Cooperative tracking methods can provide driving assistance systems with more information about vehicles. I present a new vehicle tracking method which integrates both camera based methods and wireless based methods to take the advantages of both kinds of systems. Figure 5.1 shows how sharing camera information can provide more accurate view of road for each vehicle.

My tracking method uses camera technology integrated with wireless technology to provide information about neighboring vehicles. The camera captures the surrounding environment and the associated vision system provides the position, direction and speed

of all the vehicles which are visible to the camera. In some situations, like reaching an intersection, it would be helpful if the system had information about other vehicles which are not in camera's sight, e.g. the example in Figure 5.1. To do so, wireless technology can be used.

Each subject vehicle (SV) will send a wireless message to neighboring vehicles (NV) in the surrounding area and request their position, speed and direction as well as their camera's information about other vehicles' position, speed and direction. Each NV sends the requested information along with a timestamp.

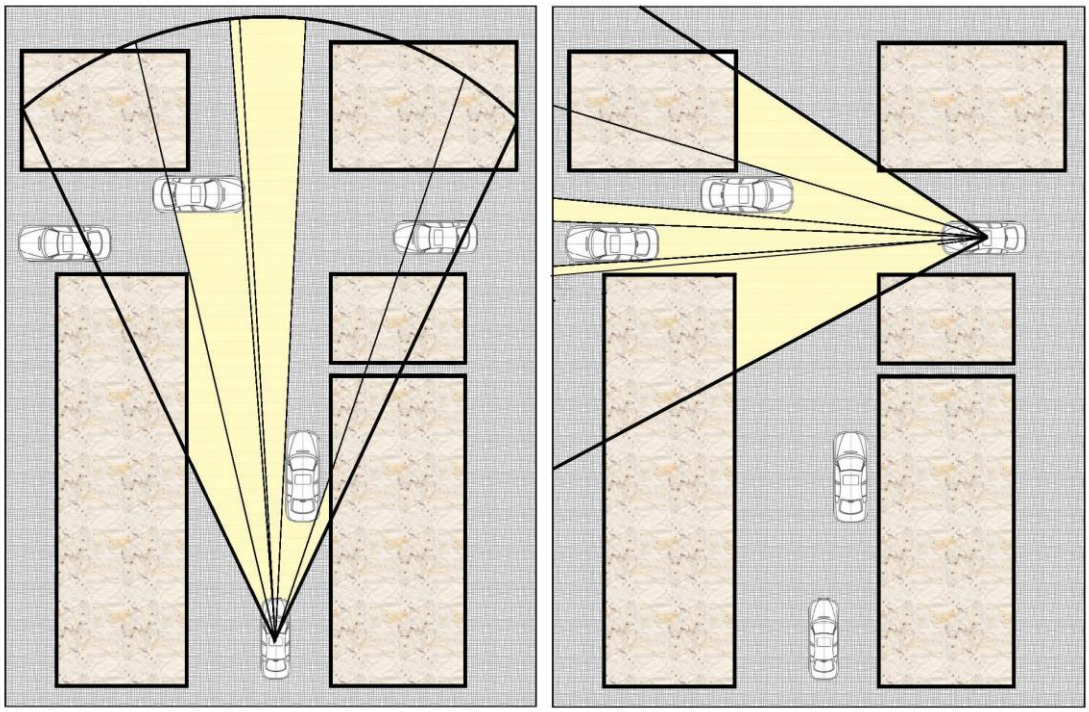

Figure 5-1. In the right picture the vehicle can detect two other vehicles using its camera and in left picture another vehicle can detect two other vehicles by its camera. If these two vehicles share their camera views they can have a more complete view of road

Each wireless message received by an SV contains the positions of detected vehicles by the sender and the position of the sender vehicle itself. Other than a list of detected vehicle positions, the message contains a timestamp which shows the time this list was created. The SV has its own list of the positions of detected vehicles and when it receives a wireless message, the SV processes the message and adds all the vehicles' positions in that list to its own list. Each response message is of the following format:

Response Message = (TimeStamp , ListOfVehicles)

$ListOfVehicles = \{Vehicle_{sender}, Vehicle_1, Vehicle_2, \dots , Vehicle_i, \dots , Vehicle_n\}$

$Vehicle_i = (Position_i , Speed_i , Direction_i)$

The first triple in the list represents the sender information and the next ones are the information about the vehicles detected by the sender's camera.

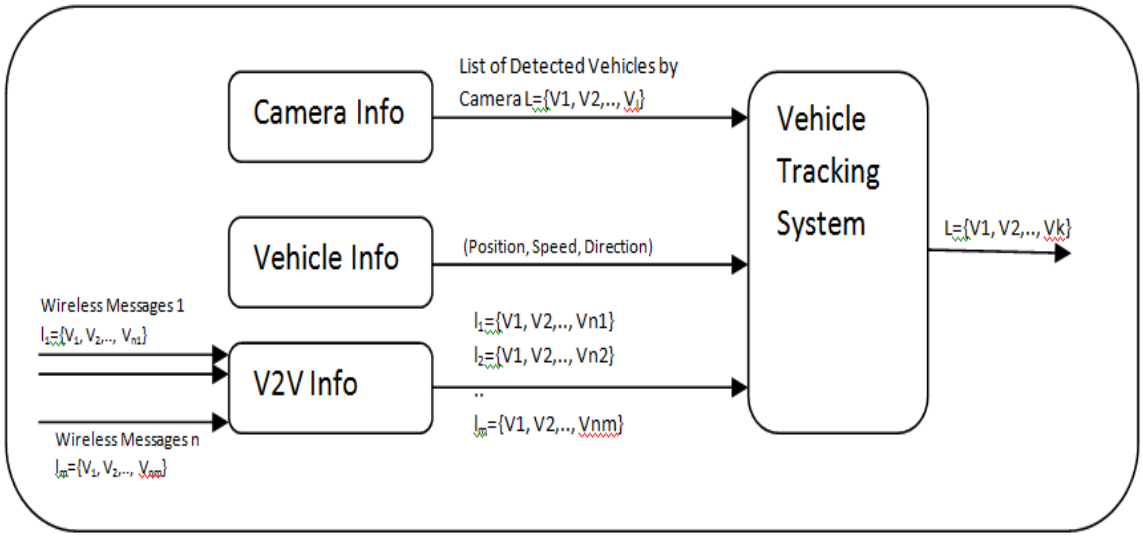

Figure 5-2. Tracking System Structure

SV collects the responses within 2 seconds and ignores the messages received after that. When the SV receives the NVs' information, it matches the information to its own list of detected vehicles and creates a bigger view of surrounding environment. To do so, based on the estimation of vehicles' position, the system either finds the match for each vehicle in its own list provided by its camera, considering an acceptable error, or adds its "view" of vehicles as a new vehicle (See Figure 5.2).

The error is based on the time at which the request was sent, the timestamp of the received message, the vehicle speed and direction; it is calculated as follows:

For Each $ListOfVehicle_{si}.Vehicle_j$ from V2V $Message_{si}$
$Error_{i,j} = (CurrentTime - MessageTimeStamp_i) * Speed_{i.j}$
If There is no $Vehicle_k$ in $ListOfVehicles_{Sv}$ where
$(Vehicle_{i,j}.Position - Error <= Vehicle_k.Position <= Vehicle_{i,j}.Position + Error)$
Then Add $Vehicle_{i,j}$ to $ListOfVehicle_{Sv}$

If the system can find each vehicle with same position in its list of vehicles or if it can find one with an error less than or equal to the Error calculated above, consider both the same vehicle and ignore it. But if it cannot find such a vehicle in its list, it adds the vehicle information to the list (See Figure 5.3).

The more vehicles that can be detected by the system, the more accurate and reliable the tracking system can be.

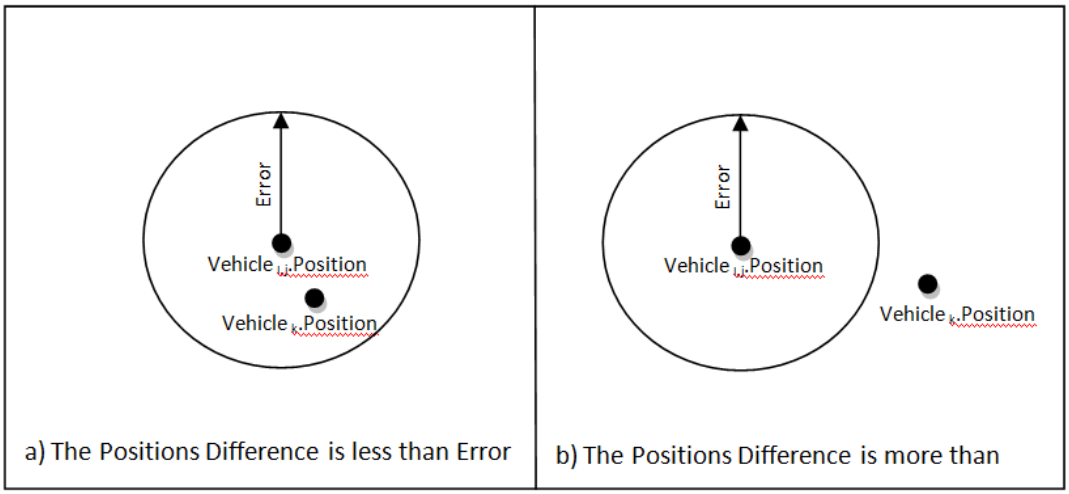

Figure 5-3. Vehicle Tracking System adds a new vehicle to the list if it is not already in the list within an error; Vehiclei,j in 3.a will not be added to the list while Vehiclei,j in 3.b will be added to the list

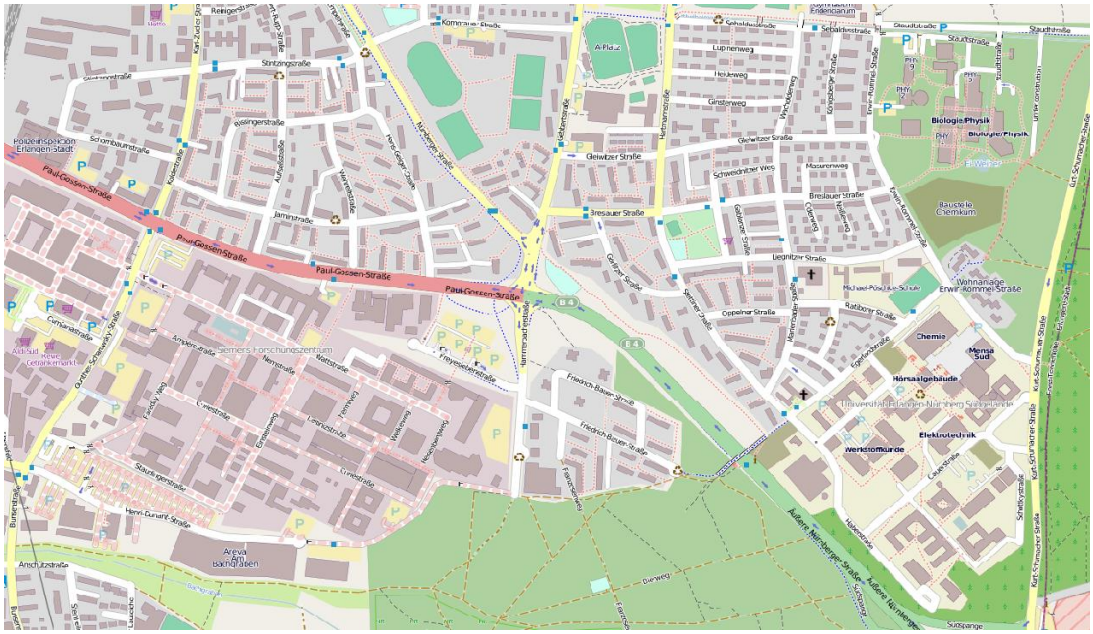

Figure 5-4. Map of Erlangen, Germany, as available from the OpenStreetMap project [7], [36]

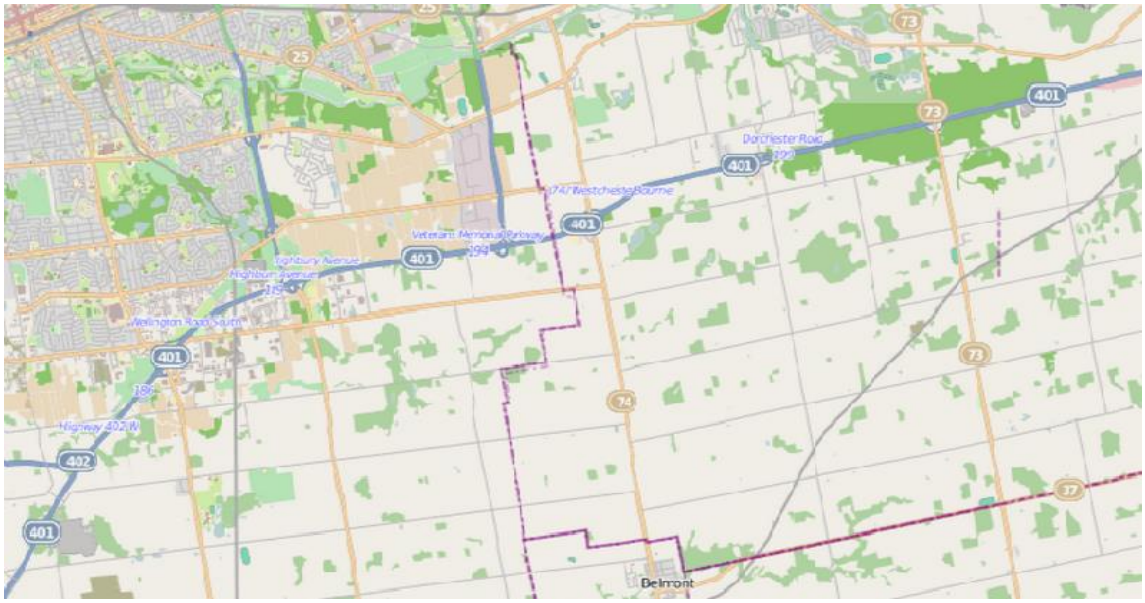

Figure 5-5. Map of 401 Highway, Canada, as available from the OpenStreetMap project [36]

I have tested the tracking method with an Erlangen city map (Figure 5.4) and a 401 Highway map (Figure 5.5) in Ontario in light traffic congestion and heavy traffic congestion and also specifically at intersections.

In order to evaluate the vehicle tracking method, I have used a simulator which consists of three main components; a vision simulator, a wireless communication simulator and a traffic simulator. The specification of these components and their connections is explained in next section.

5.4 Evaluation

I evaluated the vehicle tracking system under different situations when different percentages of vehicles are equipped with wireless communication or both camera and wireless (our presented vehicle tracking system). Each subject vehicle (SV) sends a map request every 100 seconds through a wireless message to neighboring vehicles (NV) in

range and asks for their information about other vehicles in their cameras' sight. When NVs which have wireless technologies receive the map request message they send the information about their positions along with the positions of the vehicles identified through their cameras. In the case that they do not have a camera, they just send their own positions. The SV collects all the information within two seconds and ignores the messages received after two seconds. Then the SV combines the collected information with its own information provided by its camera to form a better view of the road and vehicles on it.

I calculated the number of messages transferred between vehicles and the number of tracked vehicles assuming different adoption rates (number of equipped vehicles) in six different traffic road simulation scenarios; light traffic on highway (150 vehicles traveling on the roads), heavy traffic on highway(300 vehicles), light traffic in urban area (90 vehicles), heavy traffic in urban area (180 vehicles, light traffic at intersections (76 vehicles) and heavy traffic at intersections (160 vehicles).

The adoption rate could be different based on the proportion of the vehicles which are: a) not equipped with tracking technologies; b) are only equipped with wireless communication technologies and no camera; c) are equipped with the presented tracking system and d) use both camera and wireless technologies to track neighboring vehicles. When 100% of the vehicles are not equipped, 0% are equipped with wireless technology and 0% with wireless and camera technologies, the adoption rate is denoted as 100%-0%-0%. In other words, the first number shows the proportion of the vehicles which are not equipped, the second number shows the proportion of the vehicles which only use wireless technology to track other vehicles and the third number shows the proportion of the vehicles that use the integrated tracking system. Therefore, a 33%-33%-33% adoption rate means that 33% of all vehicles traveling on the road are not equipped with any technology, 33% of the vehicles are equipped with wireless technology, and 33% of the vehicles are equipped with the tracking system (wireless and camera).

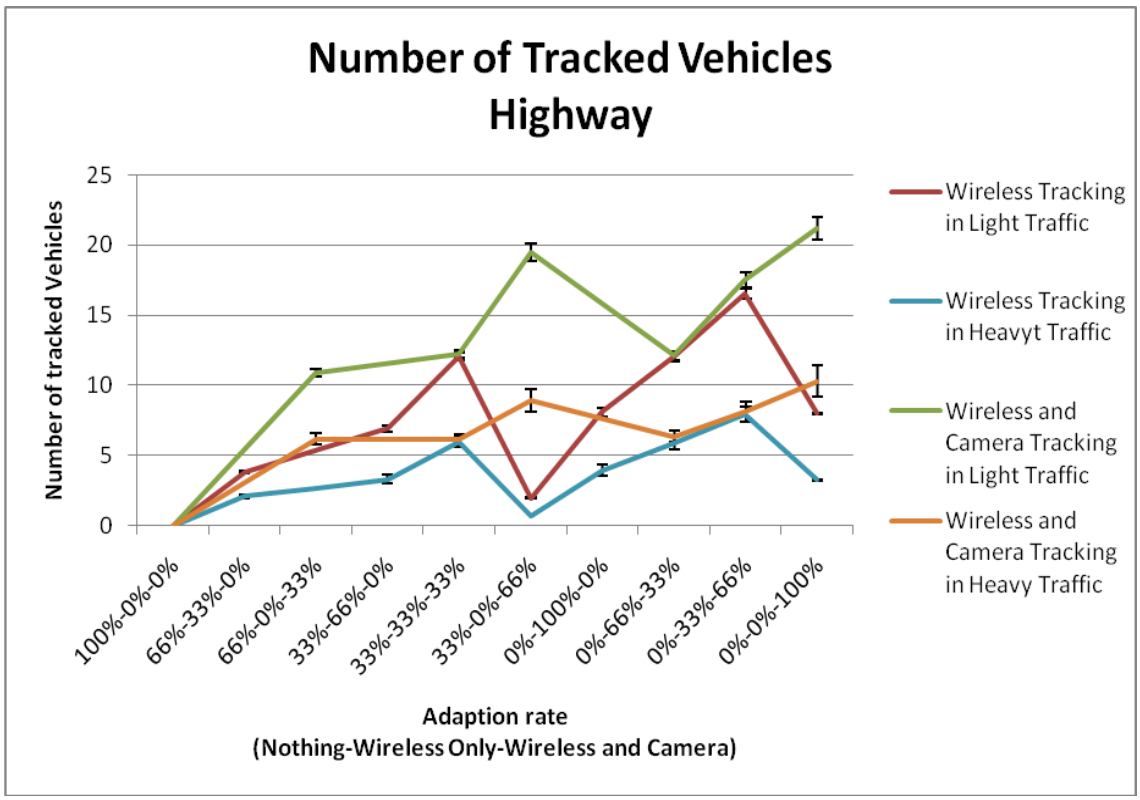

Figure 5-6. Normalized number of tracked vehicles in different tracking methods with various adoption rates in a highway heavy and light traffic; error bars show the range of one standard deviation

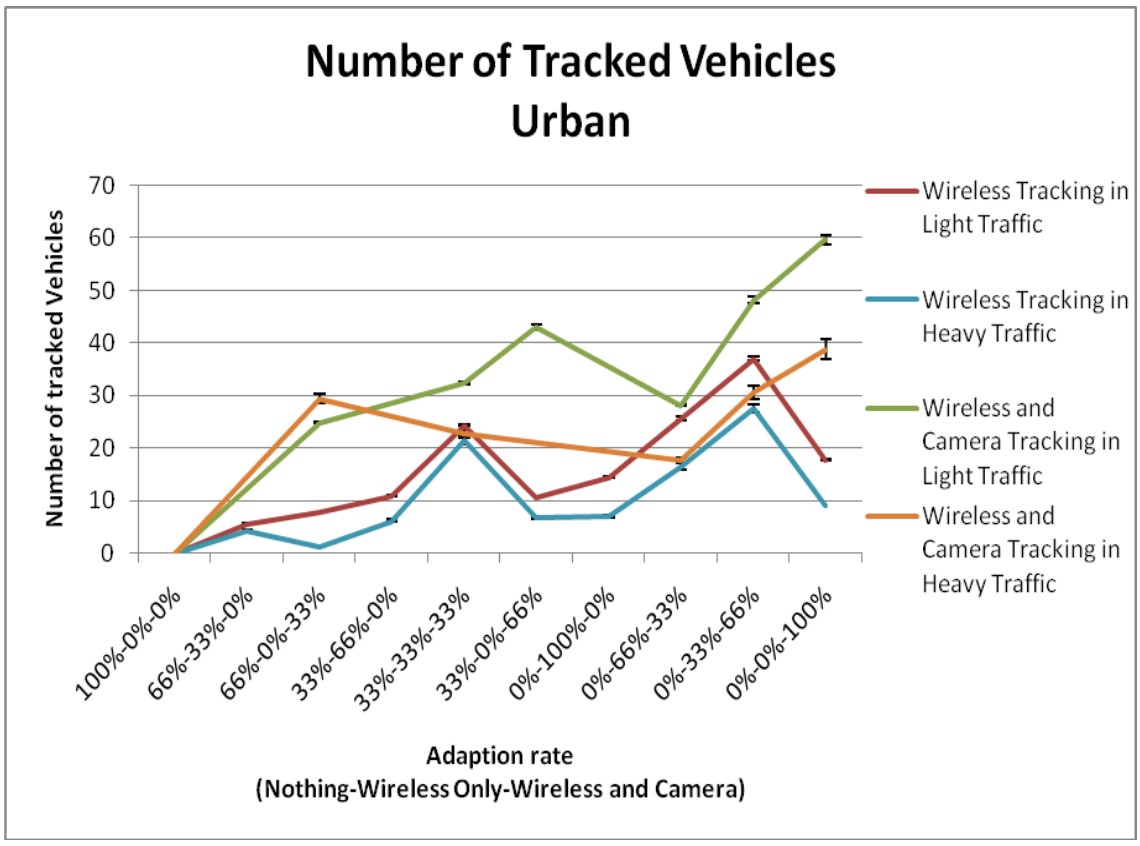

Figure 5-7. Normalized number of tracked vehicles in different tracking methods with various adoption rates in the city of Erlangen with heavy and light traffic; error bars show the range of one standard deviation

Figure 5.6 illustrates the normalized[5] average number of tracked vehicles for different adoption rates in the different scenarios for highway with heavy and light traffic. The results show, as expected, that by increasing the adoption rate, the number of tracked vehicles increases and overall number of tracked vehicles in the tracking method is much more than just the wireless based method.

[5] The normalized average of tracked vehicles was computed by dividing the actual average of tracked vehicles by the total number of vehicles and multiplied by 100.

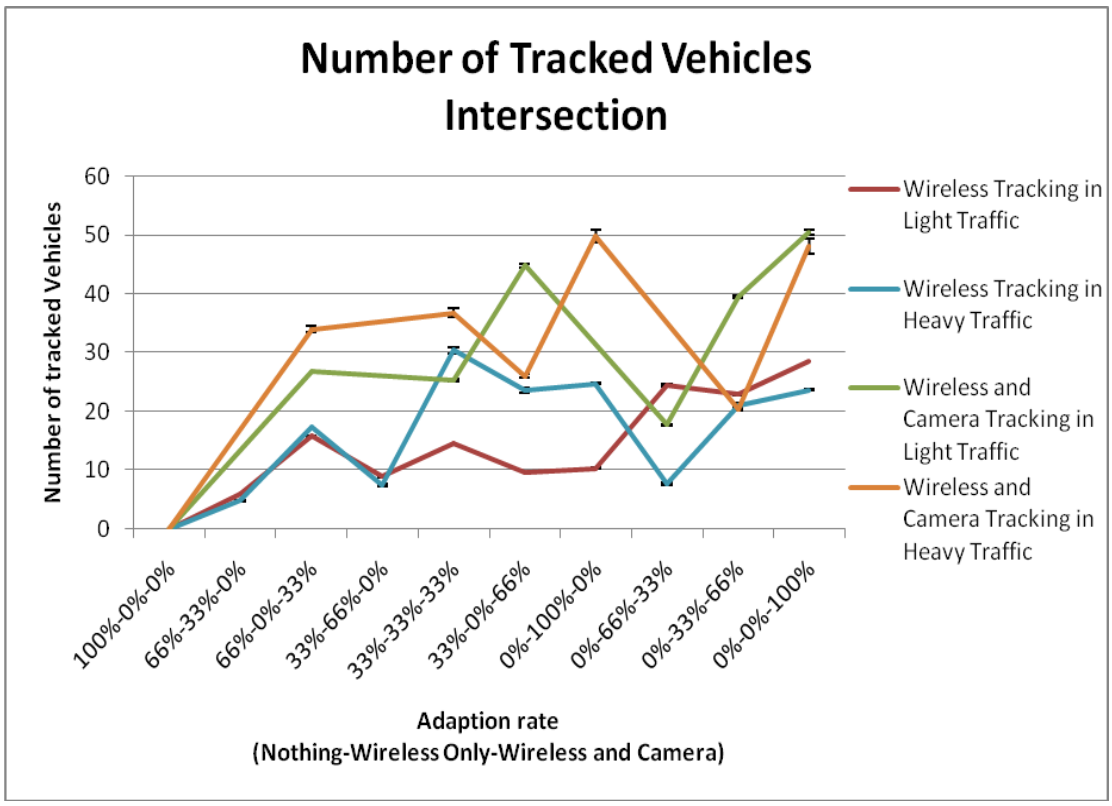

Figure 5-8. Normalized number of tracked vehicles in different tracking methods with various adoption rates at intersections with heavy and light traffic; error bars show the range of one standard deviation

The normalized number of tracked vehicles in both tracking methods using various adoption rates in an urban area is shown in Figure 5.7. The number of tracked vehicles in the wireless based tracking method and the integrated tracking method using different adoption rates at intersections with heavy traffic and light traffic is shown in Figure 5.8.

The number of vehicles which could be recognized and tracked only with camera only depends on the number of the vehicles in camera's sight of view. The average numbers of the vehicles tracked only with cameras in different scenarios are shown in Table 5.1.

Table 5-1. Average number of tracked vehicles with camera only

	Number of Tracked vehicles
Highway Heavy Traffic	1.68
Highway Light Traffic	1.77
Urban Heavy Traffic	3.87
Urban Light Traffic	2.80
Intersection Heavy Traffic	2.68
Intersection Light Traffic	1.08

The results show that using the proposed vehicle tracking method can have significant impact on number of tracked vehicles, especially at intersections where cameras' sights are limited. In these scenarios, even with low adoption rates, a vehicle can recognize a large number of neighboring vehicles. Integrating a vision based system and wireless technologies is an effective approach to track a larger number of the vehicles on the road and provide a better view of the surrounding environment. This, in turn, can provide safer and more reliable intelligent transportation applications.

The other factor which should be considered is the number of messages transferred between vehicles in order to provide requested information. Figure 5.9 shows the average number of transferred wireless messages between vehicles for tracking requests with considering various adoption rates in highway scenarios; the average number of transferred messages per request for an urban area is shown in Figure 5.10 and for intersection scenarios in Figure 5.11.

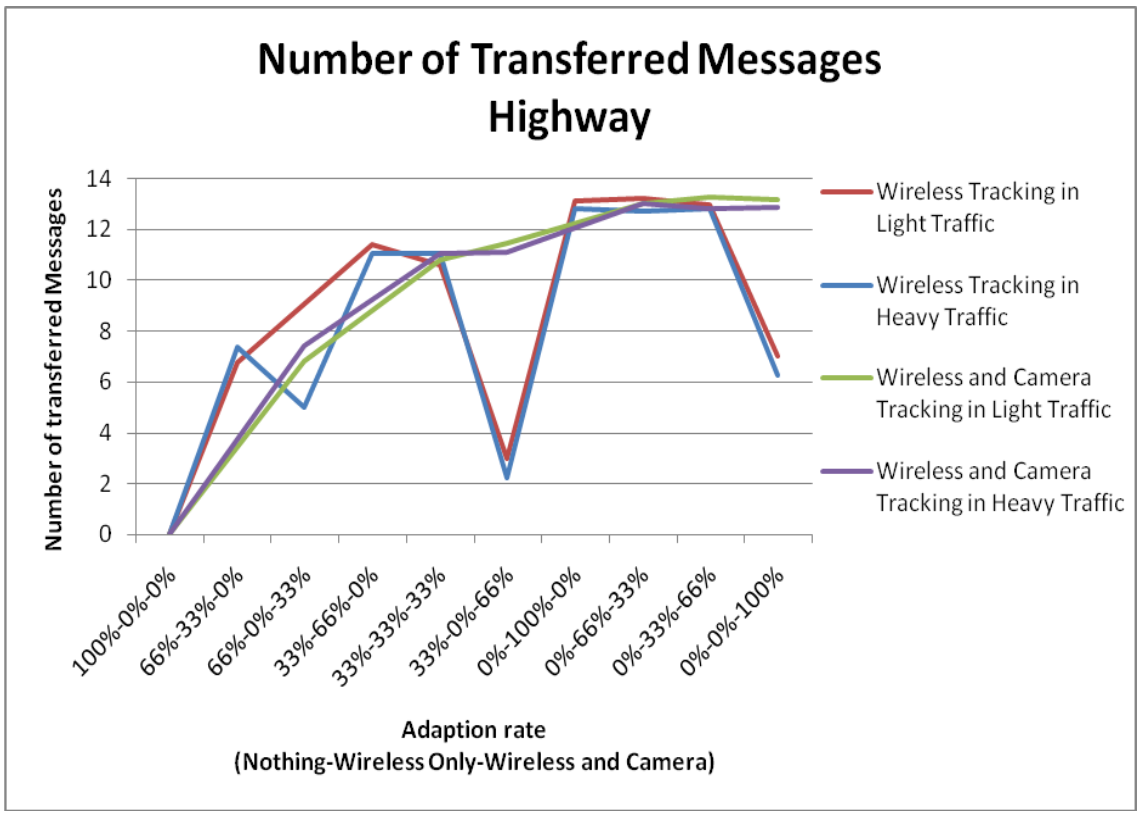

Figure 5-9. Number of transferred messages for tracking requests in highway

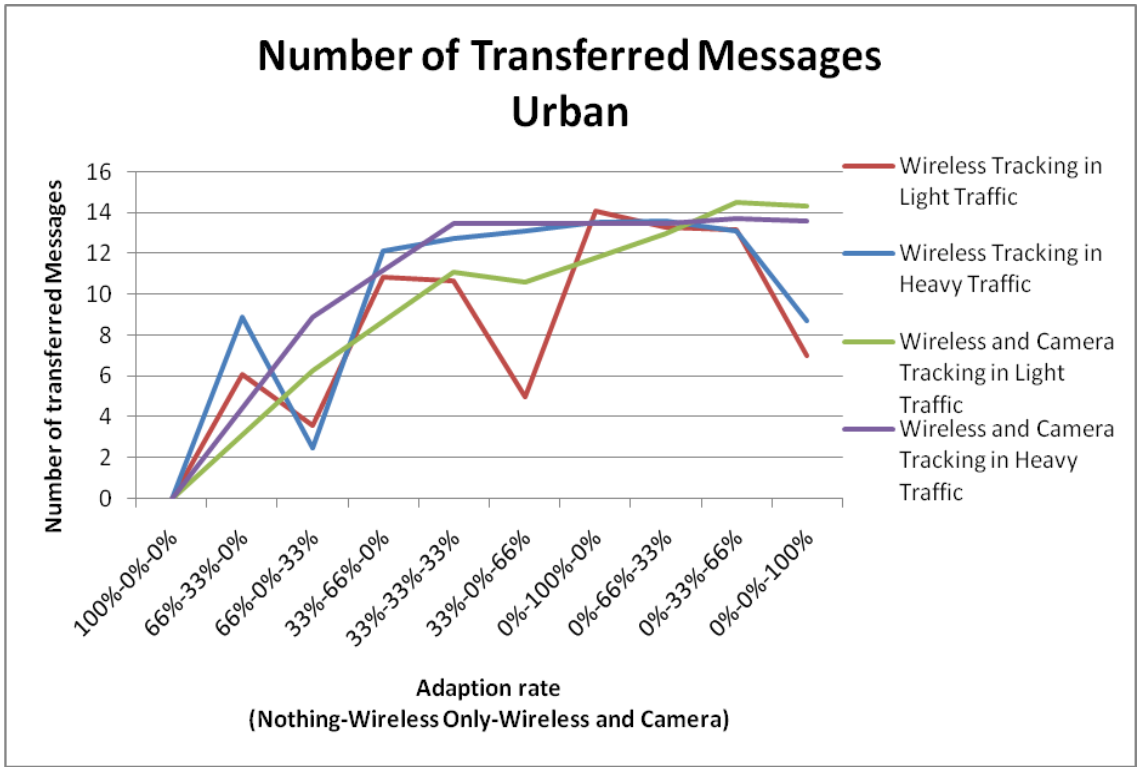

Figure 5-10. Number of transferred messages for tracking requests in urban area

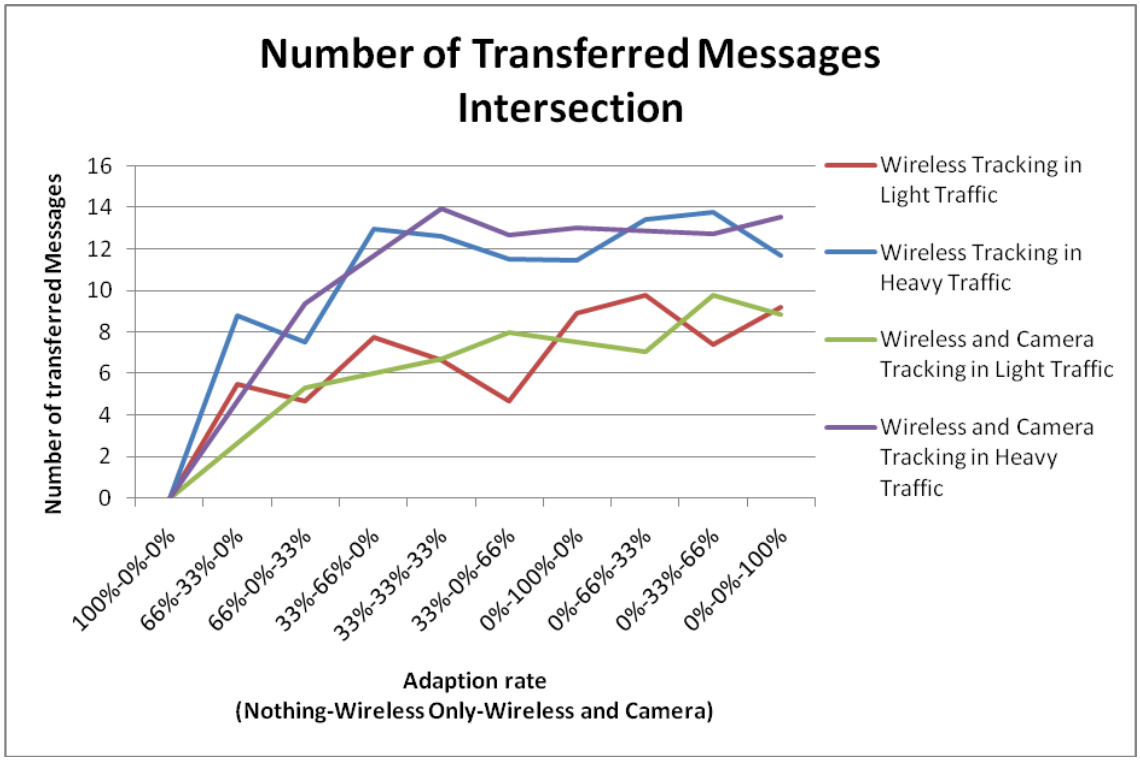

Figure 5-11. Number of transferred messages for a tracking request at intersection

Though the numbers of transferred messages in both systems are almost the same, the sizes of messages are different since the amount of transferred information is different. Each tracked vehicle location data contains latitude and longitude which is represented as a float variable with 4 bytes. Therefore, for each tracked vehicle 8 bytes is added to the size of the wireless message. So in the wireless tracking system, the size of each message is approximately 8 bytes because it just includes just one vehicle's location information. The average size of each message in the integrated wireless-camera tracking system can be calculated based on the average number of tracked vehicles with cameras plus its own location information (See Table 5.2).

Table 5-2. Average size of each message in integrated wireless-camera tracking system

	Average Size of Each Message (Bytes)
Highway Heavy Traffic	21.47
Highway Light Traffic	22.13
Urban Heavy Traffic	38.93
Urban Light Traffic	30.36
Intersection Heavy Traffic	29.48
Intersection Light Traffic	16.61

Overall, the size of the messages in an integrated wireless-camera tracking system is larger than the size of the messages in wireless-only tracking system. The communication system's bitrate is 11Mbps, so the overall impact is not so big as to influence the overall performance of the system. Generally, the integrated camera-wireless vehicle tracking system has shown great potential in increasing efficiency and accuracy in vehicle tracking applications.

6 An Emergency Message Propagation System

This Chapter is a reformatted version of the following article:

Besat Zardosht, Steven Beauchemin and Michael Bauer, “An Emergency Message Propagation System Using Roadside Units and Vehicle-To-Vehicle Communication”. *IEEE Smart Vehicles, 2016 (to be submitted)*

A message propagation system which uses Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication is presented in this chapter. In emergency situations, wireless technology enables vehicles to share warning messages with other vehicles using V2V and with emergency services by using V2I. The time that passes before an emergency service is notified about an emergency situation, such as an accident, is critical. In this paper I evaluate the effects of traffic density, V2V adoption rates, the number of hops for messages in V2V communication, location and number of roadside units on the performance of message propagation for emergency response and network coverage.

6.1 Introduction

The number of vehicles in service has grown dramatically in the past decades and has led to increases in the number of accidents. The time between an accident and the arrival of medical assistance is critical and is often referred to as the golden hour [1]. One of the largest time fractions of this hour is the time between the occurrence of an accident and when emergency services are notified of it. If the occupants of a vehicle are injured and cannot call for assistance, this time may increase and the result can be detrimental to those involved in the accident. With V2V and V2I capabilities, other vehicles passing by the accident can inform emergency services about it using wireless communication

between vehicles and roadside units. One could rely on V2V or V2I only, but a hybrid approach using both V2V and V2I communications makes the most sense. With a hybrid approach, however, there are a number of tradeoffs, including the number of times an emergency message is propagated and the geographical distribution of roadside units. The former impacts the volume of V2V communication and the latter impacts the deployment of roadside units.

In this work, I use a simulation environment to study the effect of vehicle location (whether it is city or highway), traffic density, V2V adoption rates (percentage of vehicles with V2V capability), the number of hops emergency messages can be propagated, and the number of roadside units on the notification time[6] for emergency message propagation. For Simplicity, I have used equidistant square grid for RSU (Road Side Units) placement. I have used a modified version of Veins [2] as my simulator to evaluate the emergency message propagation system.

This paper is organized as follows: Section II reviews the related work regarding roadside unit placements and emergency warning systems; Section III describes my proposed emergency message propagation system; Section IV introduces the simulation environment of the system; Section V shows the evaluation results, and Section VI offers a conclusion and avenues for future work.

6.2 Previous Work

In this Section I discuss pervious work on emergency message propagation systems and the integration of V2V and V2I communication modes into these systems.

Intelligent Vehicle (IV) systems typically use one or more forms of communication that can be categorized into three types: Inter-Vehicle Communication (IVC) which uses V2V

[6] Notification time is the time between the occurrence of an accident and the time when emergency services are notified about it.

communication, Roadside-to-Vehicle Communication (RVC) which uses V2I communication and Hybrid Vehicular Communication (HVC), employing both V2V and V2I communication [3].

There are several cooperative collision warning systems which use V2V communication to inform other vehicles about impeding collisions [4]–[7]. In addition, Hybrid Vehicular communication methods have been introduced for similar purposes [8], [9]. These systems integrate V2V with V2I in order to improve the performance of various IV applications.

Martinez *et al.* present a futuristic architecture of an accident notification system that combines V2V and V2I communication in order to reduce the notification time after an accident occurs [10]. In this system, wireless messages are delivered to a control unit which in turn estimates the severity of the accident and determines the appropriate rescue resources to be deployed. However, there is yet to be an implementation of these ideas.

A message propagation system protocol for both V2V and V2I communication is described by Vegni and Little [11]. In this model the lower and upper bound for the message propagation rate is characterized by factors such as the direction and speed of vehicles. To prioritize emergency messages over other messages, a dual frequency channel approach is presented by Maeshima *et al.* [12].

V2V and V2I communication can be used for purposes other than accident or driving information. In the case of a natural catastrophe, such as earthquakes or floods, V2V and V2I communication may be used to spread warning messages about the specific threat [13]. Alternatively, a model to address security and efficiency issues in Vehicular Ad-Hoc Networks (VANET) is presented by Zhu *et al.* [14].

Roadside Units (RSUs) constitute additional hardware that emergency message propagation systems may use. However, they are expensive to install and maintain in

vehicular environments and thus reducing their number is an important aspect of their deployment [15]. There are many approaches to RSU placement in vehicular networks. Lochert *et al.* present an RSU placement model which uses a genetic algorithm aimed at overcoming problems inherent to limited bandwidth while ensuring minimal deployment. [16].

Another approach to minimize the number of RSUs in vehicular networks presented by Abdrabou and Zhuang considers vehicle density, vehicle speed, and warning message lifespan in the placement of RSUs [17]. Wu *et al*. present a similar method also based on vehicle density and speed [19]. Alternatively, Barrachina *et al.* observe that in areas of high vehicle density, V2V communication can be used to propagate messages over large distances and present a placement method that considers fewer RSUs for high vehicle density areas and vice-versa [18]. The problem of bounded-delay RSU placement is studied by Li *et al*. and provides a model in which all the vehicles are able to receive the messages within a given time window [20].

In this work I use a decision-making module to detect accidents and propagate relevant messages to other vehicles. The number of times (hops) that an accident message can be propagated from vehicle to vehicle, the traffic density, the number of RSUs, and the V2V adoption rate are all important factors in the process of informing emergency services regarding situations such as accidents. Other than the statistical study of the effect of these factors on the overall performance of the system, the decision-making module provides useful information for emergency services such as the location of an accident and an estimate of its duration.

6.3 Emergency Message Propagation System

The problem of RSU placement is difficult but important in vehicular networks. Too many RSUs on the road network may result in high installation and maintenance costs while fewer RSUs may cause low performance of V2I systems. Hence, finding the

optimal number of RSUs is crucial. The number of RSUs needed depends on many factors including traffic location, traffic density, vehicle speed and number of vehicles equipped with wireless communication systems.

Once the decision-making system of the vehicle detects that it is in an accident, it collects location information from the sensor agent, gets the number of lanes in the current road, and speed and acceleration of the vehicle prior to the accident from the vehicle agent. Then the decision-making module uses an accident duration model to predict its duration [22] and, with the use of a wireless agent, sends an accident message to other nearby vehicles and RSUs.

I have designed an emergency message propagation system which uses both V2V and V2I communication to inform emergency services about the occurrence of an accident. With this system I evaluate the effect of traffic location, traffic density, V2V adoption rates, and the number of hops that an emergency message is propagated. I measure both the notification time, the time for an emergency message to reach an emergency center, and the percentage of number of vehicles which are in the range of the communication network. I have used a uniform, equally spaced RSU placement grid scheme and considered two different scenarios: one with nine RSUs, and another with four RSUs only (see Figure 6-1 and Figure 6-2).

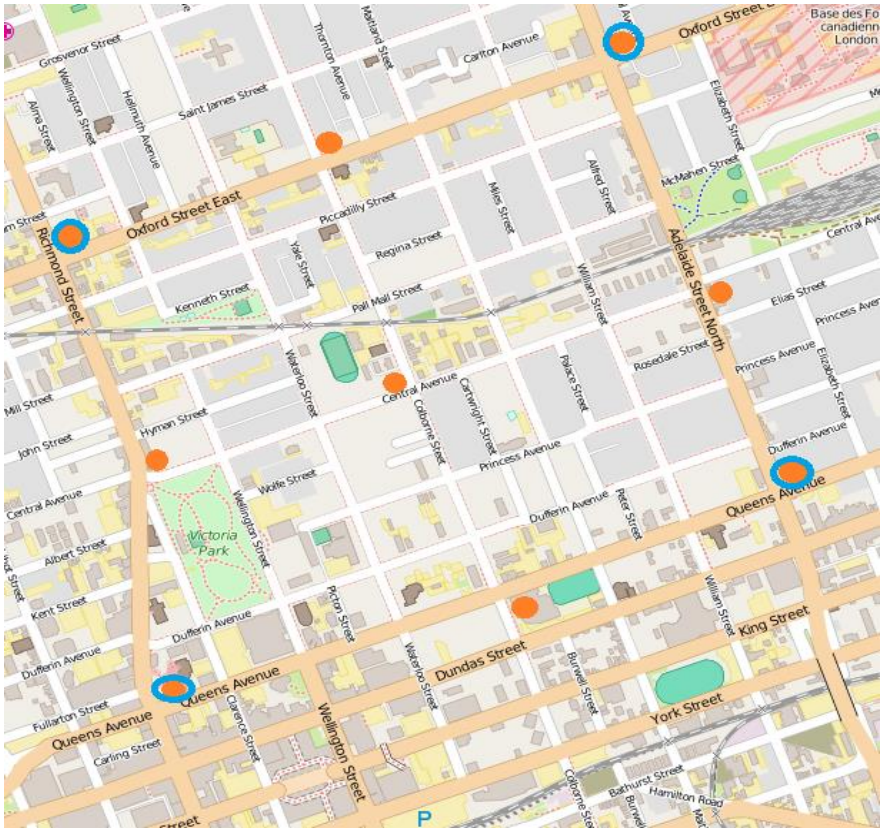

Figure 6-1 Location of RSUs in City: Red circles are the location of RSUs in a nine RSU configuration and the circles with blue outlines are the locations of RSUs in a four RSU configuration

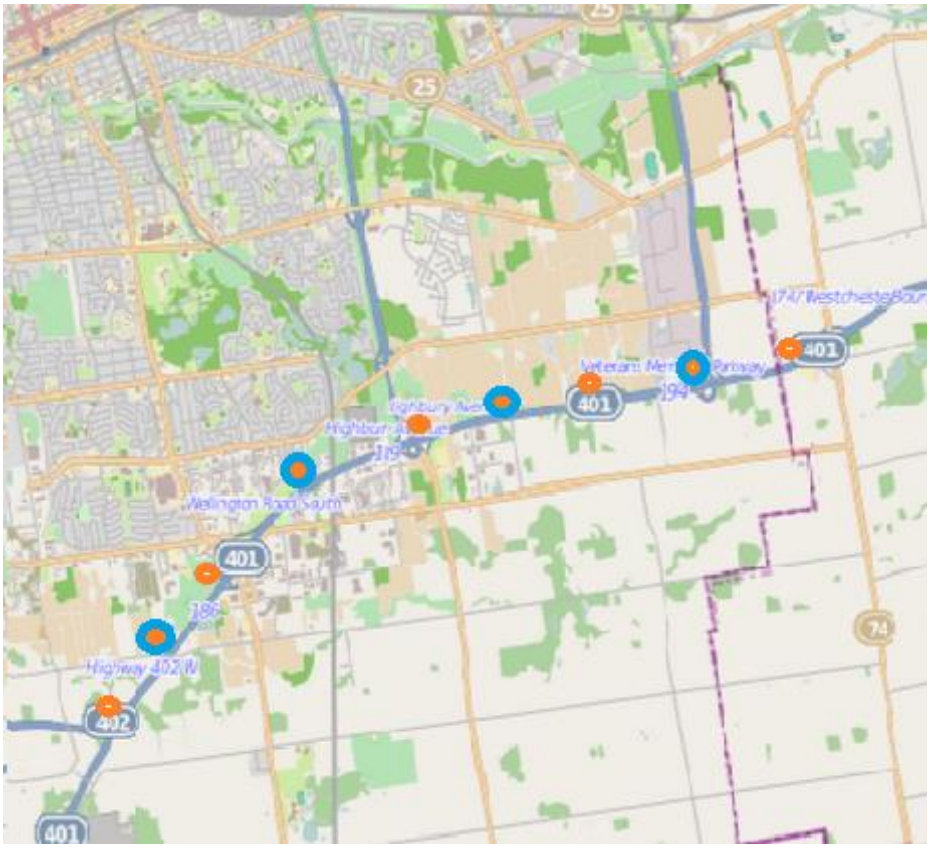

Figure 6-2 Location of RSUs in Highway: Red circles are the location of RSUs in a nine RSU configuration and the circles with blue outlines are the locations of RSUs in a four RSU configuration

In the configuration with nine RSUs, the distance between them is about one kilometer and in the other configuration with four RSUs, the distance is approximately two kilometers. The RSUs are assumed to be connected to each other by a network and to an emergency centre. When a RSU receives an emergency message, it forwards it to an emergency response center and the assumption is that the center will then notify the emergency response units that should proceed to the accident. When a vehicle is involved in an accident, it propagates a wireless message; for the current study I assume that at least one vehicle per accident is equipped with V2V. Any vehicle or RSU can receive an accident message if they are in communication range. If a vehicle receives an accident message, it will resend it if the hop number is more than one. The first RSU that receives

this message will forward it to the emergency response center. The RSU also broadcasts the accident message to warn other nearby vehicles. When a vehicle receives the message from the RSU, it stops resending it since the emergency services have already been notified. The emergency message contains important information about the accident, such as the time of the accident, its location (GPS), and its predicted duration. The predicted duration of the accident is calculated with a survival model [22]. Accident duration predictions can be helpful for emergency systems to estimate the severity of an accident and to perform necessary actions. In addition, vehicles in proximity to the accident can provide an estimation of when the road might be cleared, warn other drivers about a possible traffic jams, and perhaps look for alternative routes.

I assessed the performance of my system by providing different simulation scenarios. The factors considered in the simulations are shown in Table 6-1. The details of my simulation are discussed in the next Section.

Table 6-1. Different simulation parameters

Experiment Factors	Choices
Vehicle Location	Highway or City
Traffic	Low (approximately 150 vehicles in four square kilometers) or High (approximately 750 vehicles in four square kilometers)
Adoption Rate	10, by increments of 10%
Number of Hops	1 or 2
Number of RSUs	9 or 4

6.4 Evaluation

I measured the notification time and average number of vehicles which are within communication range of the emergency network under different circumstances using my

simulation: two different locations (City and Highway), two sets of RSUs (nine and four), two traffic densities (low traffic: 150 vehicles in 4 square kilometer and high traffic: 750 vehicles in 4 square kilometers), two sets of hops (one and two), and ten adoption rates (from 10% to 100%). These choices gave us 160 distinct simulations. Every 10 seconds I compute the notification time of an emergency message propagated from each vehicle within range of the communication network *as if* each was sending a message regarding an accident. This allows us to measure the average notification without running many more simulations for specific accidents and vehicle travel patterns. It gives us a means of comparing the impact on notification time of the different criteria.

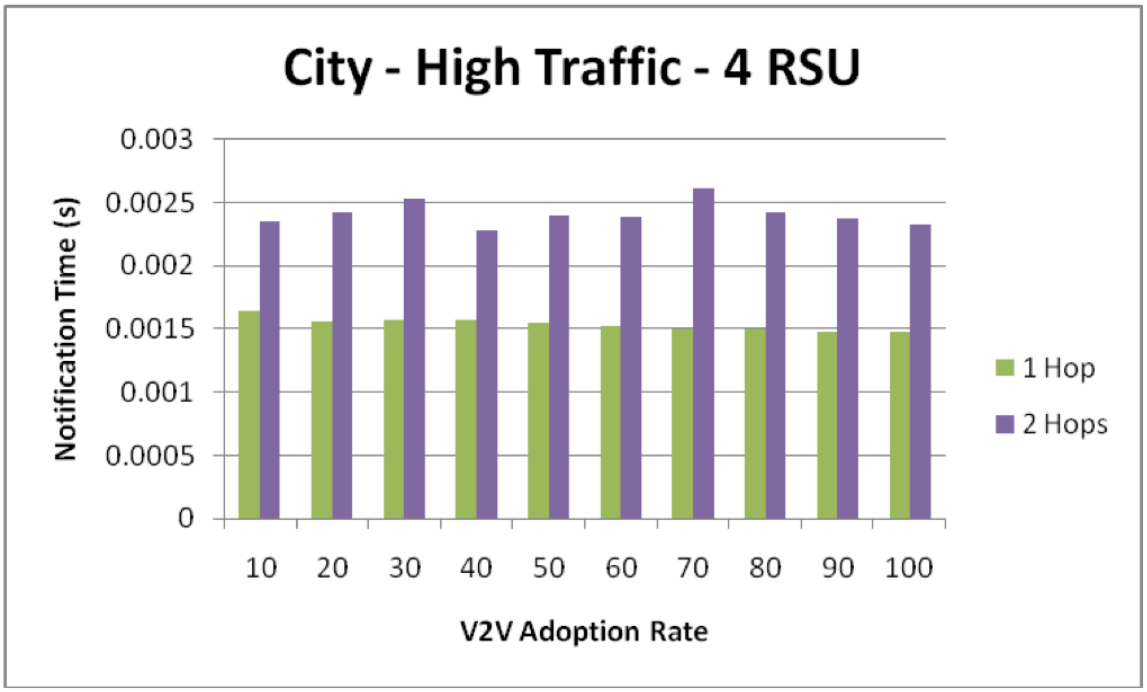

Figure 6-3. Notification time in city of London ON in high traffic with four roadside units

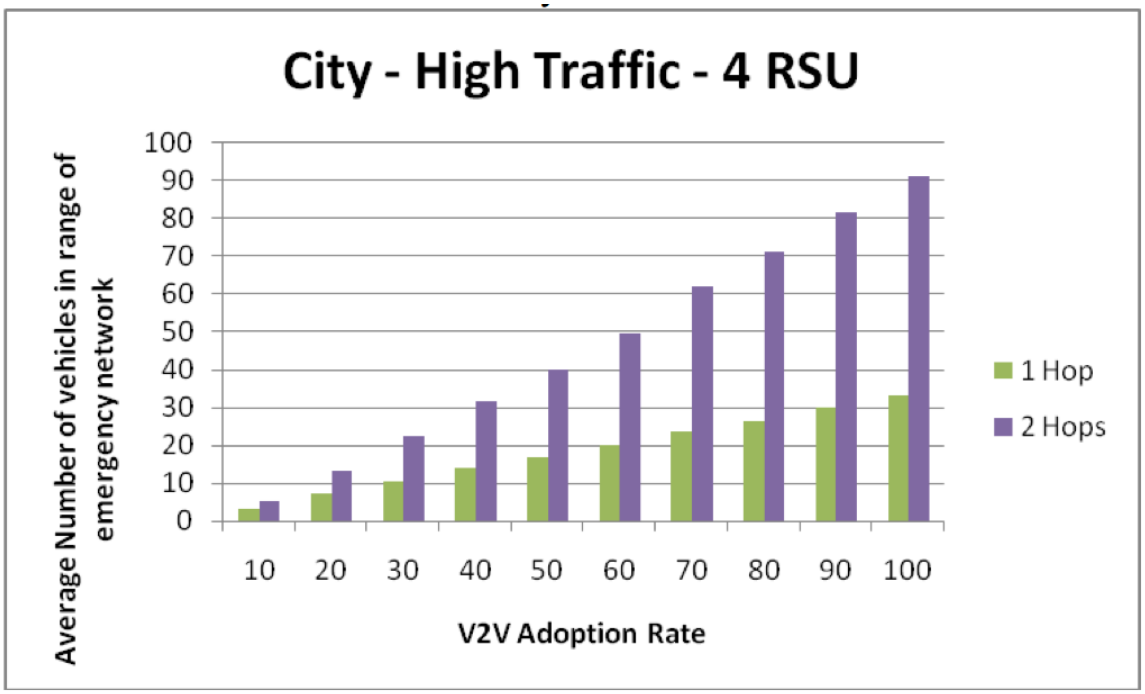

Figure 6-4. Network coverage for high traffic density with four roadside units in city area

Figure 6-3 and Figure 6-4 show notification times and average number of vehicles covered by the network in high density traffic with four RSUs installed on the road network in a city area. As shown, when the emergency message is propagated for two hops the number of vehicles covered increases dramatically. Another observation is that the average response time increases with two hops. The reason is that when the number of hops increases, more vehicles which had not been covered will be in network range but for those vehicles the notification time is higher since the message will pass through more hops to reach a central point. This results in higher average notification time. As expected, the adoption rate has great impact on the number of covered vehicles since the number of potential covered vehicles increases when more vehicles are equipped with wireless technology.

Figure 6-5 and Figure 6-6 show the results for the situation with high traffic density and 9 RSUs on the road in the city. The trend for network coverage is almost identical to the results obtained with fewer RSUs but in this case the average number of covered vehicles are generally higher. In this set of experiments the notification time is higher with more hops but it decreases when a higher percentage of vehicles are equipped with wireless technologies.

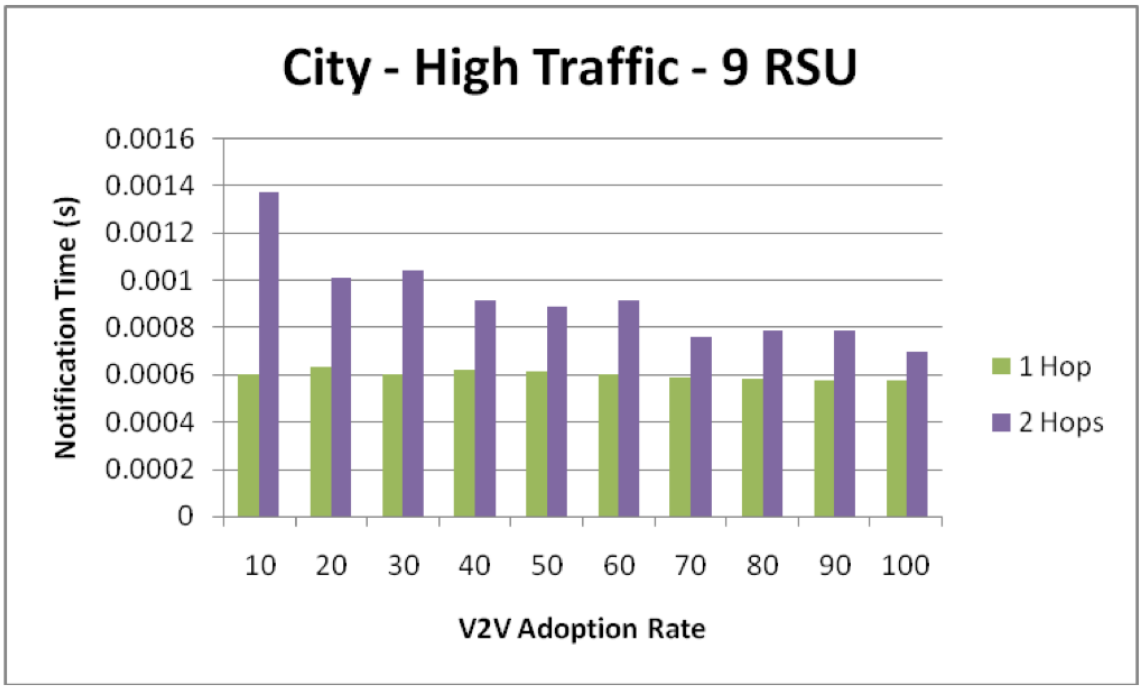

Figure 6-5. Notification time in high traffic with nine roadside units in city environment

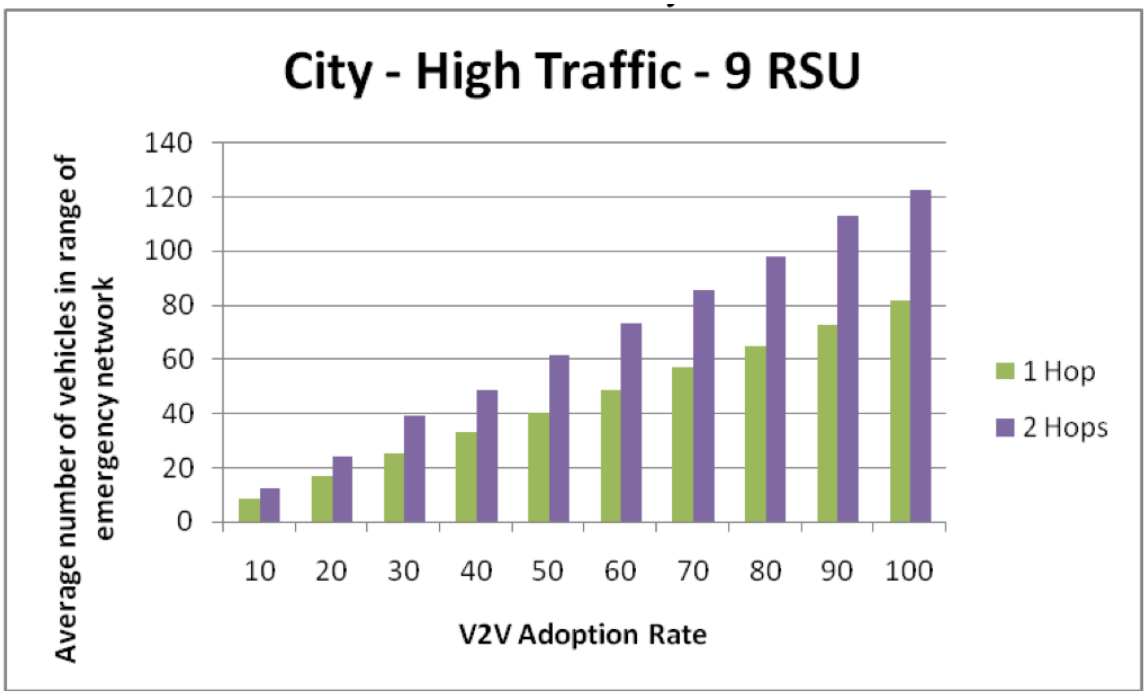

Figure 6-6 . Average number of covered vehicles in high density traffic with nine roadside units in city area

For low traffic density in a city environment, Figure 6-7, Figure 6-8, Figure 6-9 and Figure 6-10 summarize the results. In low traffic density with 4 RSUs, the results for the notification time are similar to the high density situation, but the number of covered vehicles is different. When there is low traffic density with 4 RSUs there are many fewer vehicles covered by the network if with the adoption rate is below 60%; this is the case for 9 RSUs as well. Interestingly, the lower coverage rate has little impact on the notification time when there are 4 RSUs (steady at aroun 0.002 seconds) but in the case of the 9 RSUs the increased coverage when over 60% of the vehicles have V2V, the notification rate drops.

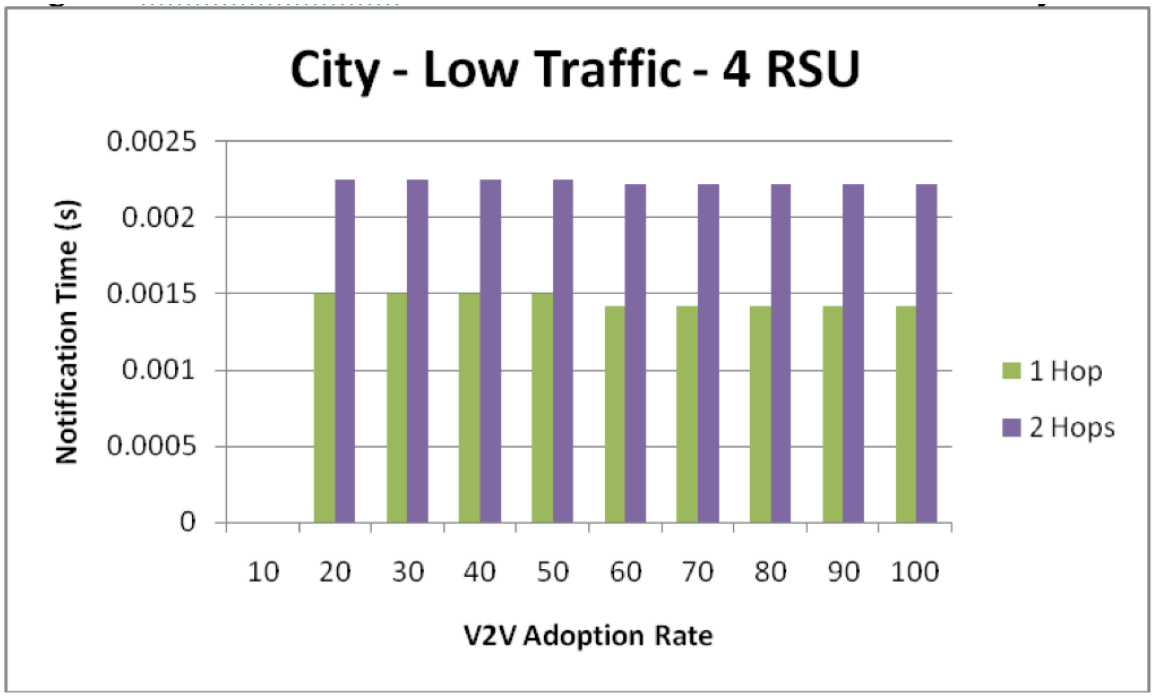

Figure 6-7. Notification time in low traffic with four roadside units in city area

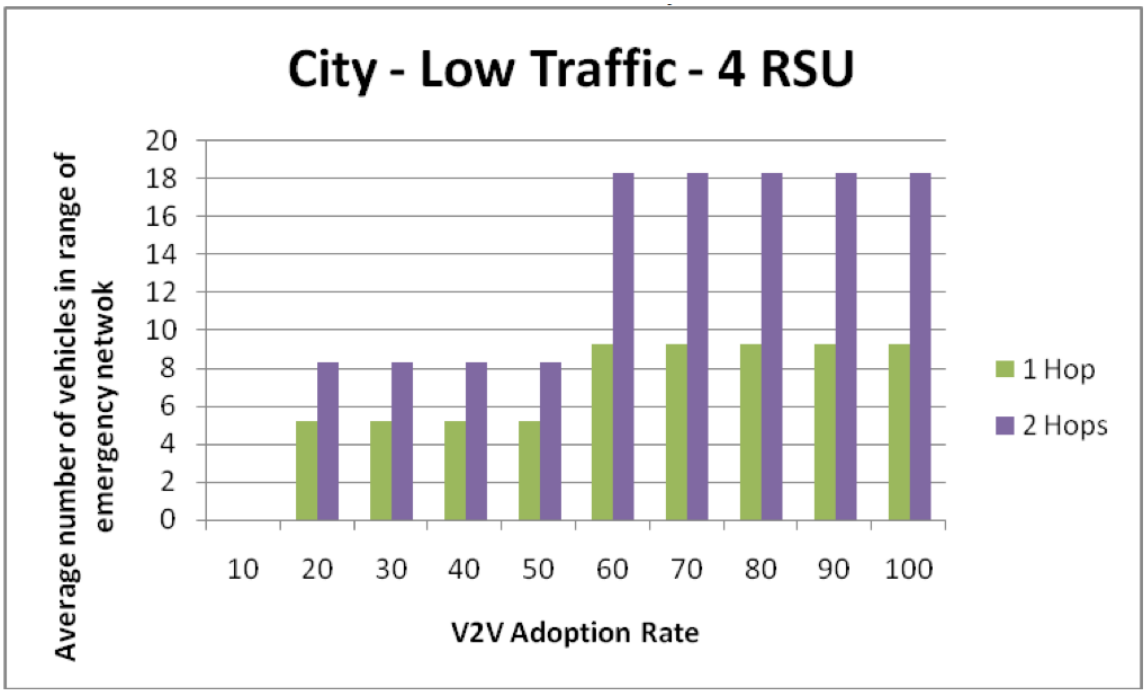

Figure 6-8. Average number of covered vehicles in low density traffic with nine roadside units in city area

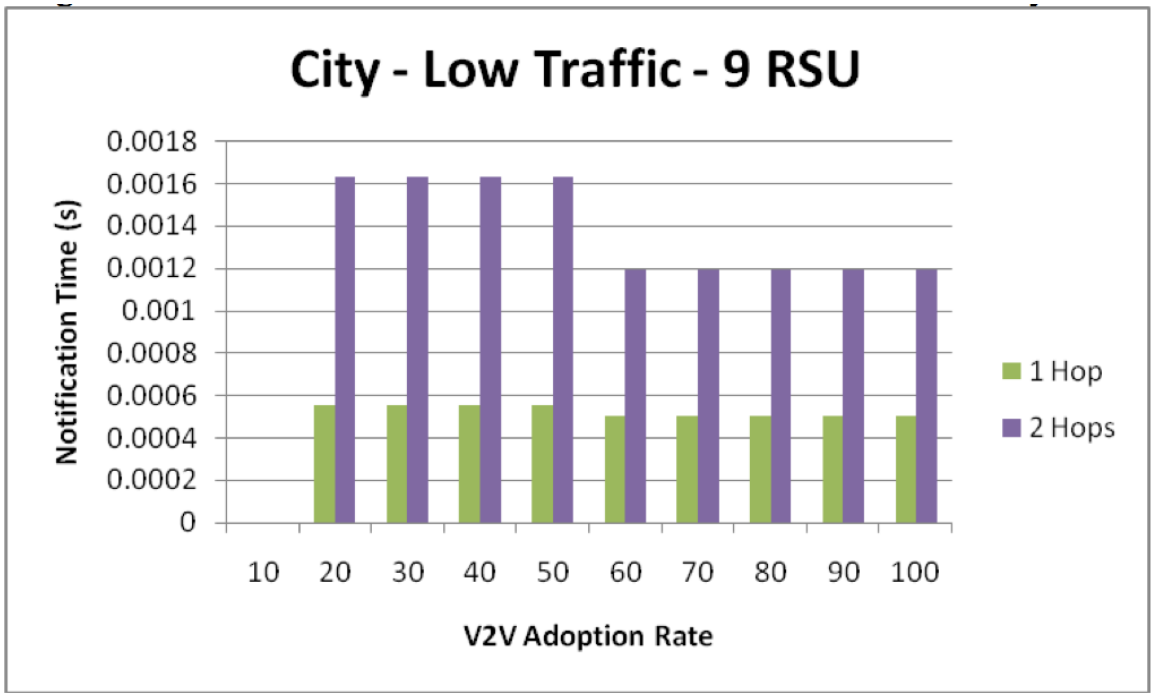

Figure 6-9. Notification time in low traffic with nine roadside units in city area

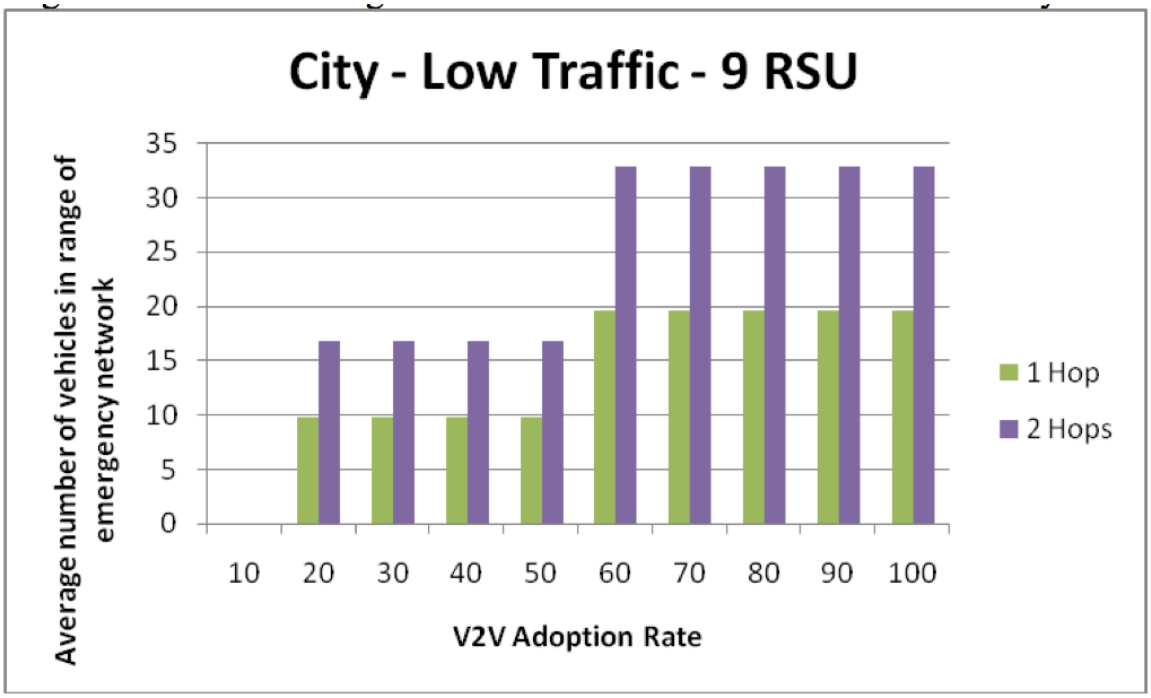

Figure 6-10. Network coverage in low traffic with nine roadside units in city area

Driving in a highway environment is different than driving in a city area. One might expect that the notification time in highway environment might be higher than in city because of less dense traffic or network coverage. However, the notification time and network coverage trends are almost the same. The results for notification time and network coverage in highway environment with high traffic and with four RSUs are shown in Figure 6-11 and Figure 6-12.

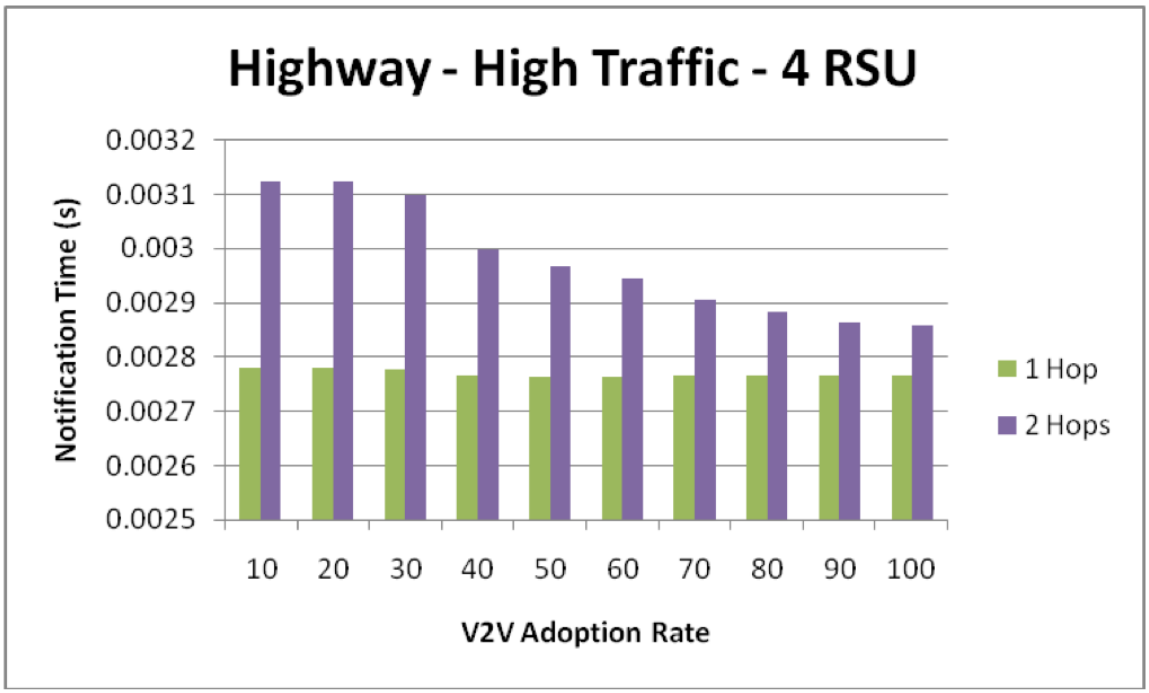

Figure 6-11. Notification time in high traffic with four roadside units in highway area

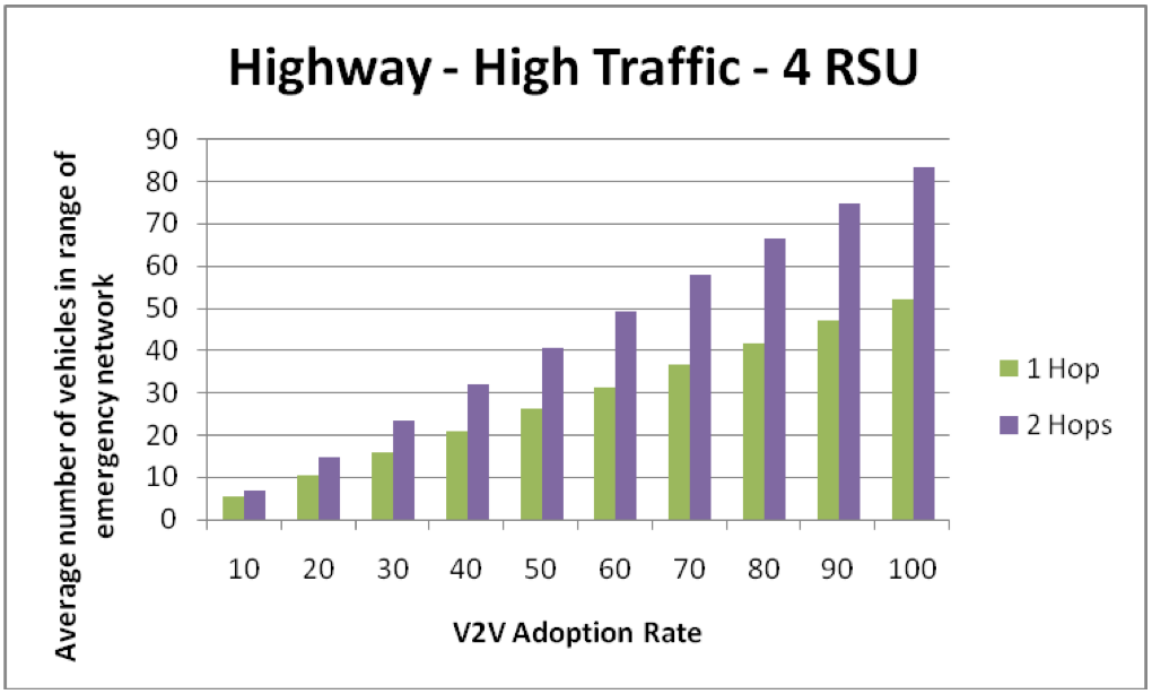

Figure 6-12. Network coverage in high traffic with four roadside units in highway area

The notification time and network coverage for high traffic density with nine roadside units in a highway environment is shown in Figure 6-13 and Figure 6-14. The notification time is over half the time with with only four RSUs. However, the number of covered vehicles is almost the same. These results suggest that adding the number of RSUs in highway with high traffic density does not have considerable impact on the network coverage nor has an impact on the notification time. In contrast, in a low traffic density environment with fewer RSUs (4), since there are fewer vehicles to propagate the wireless messages, higher notification times and lower network coverage occurs (see Figure 6-15 and Figure 6-16). Finally, the results for the notification time and network coverage for highway in low traffic density with nine RSUs, the notification time is lower when compared the are shown in Figure 6-17 and Figure 6-18. Response time is much lower comparing to the low traffic density 4 RSU environment; this is not surprising.

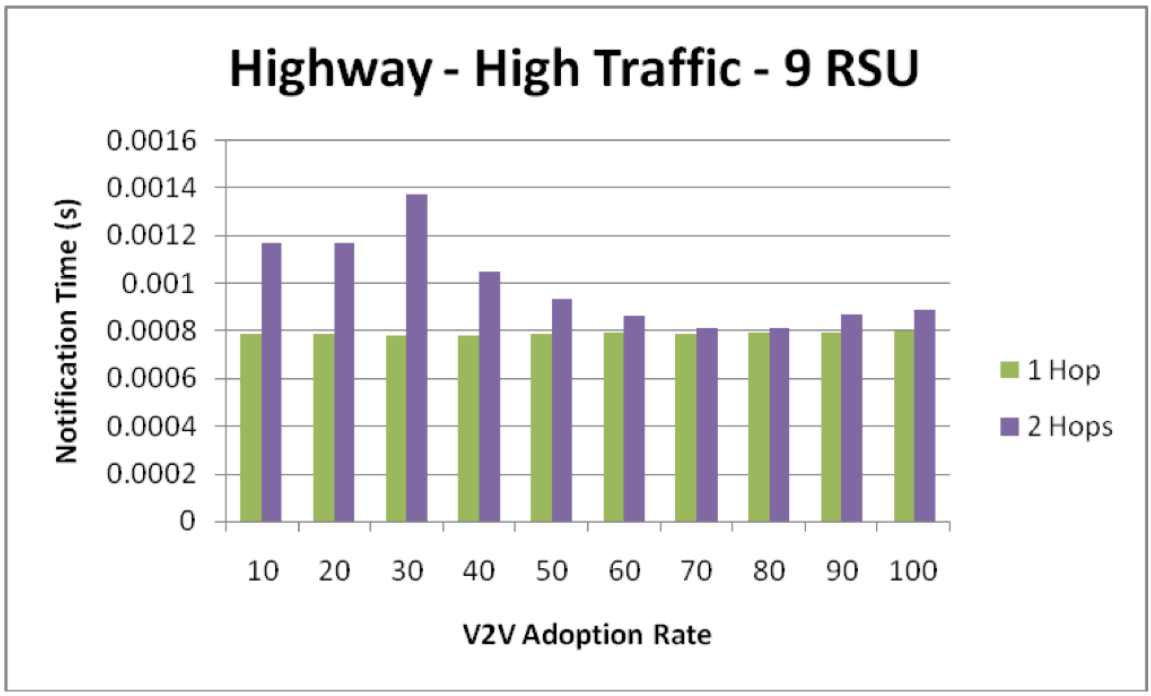

Figure 6-13. Notification time for highway in high traffic density with nine roadside units

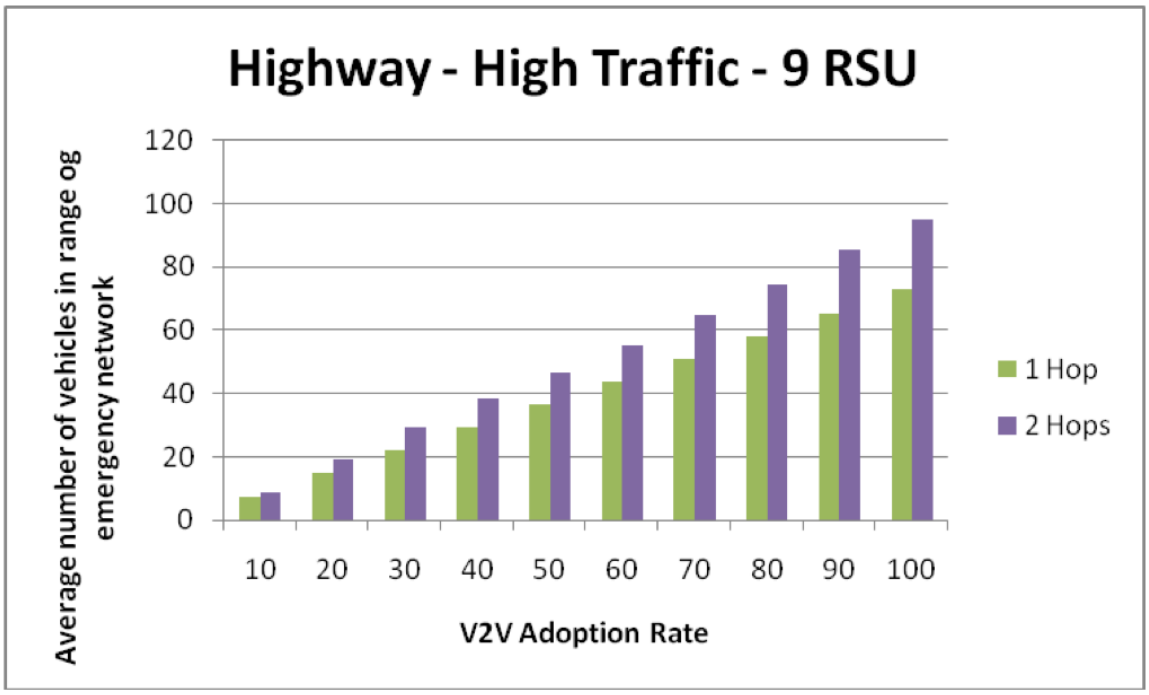

Figure 6-14. Number of covered vehicles for highway in high traffic density with nine roadside units

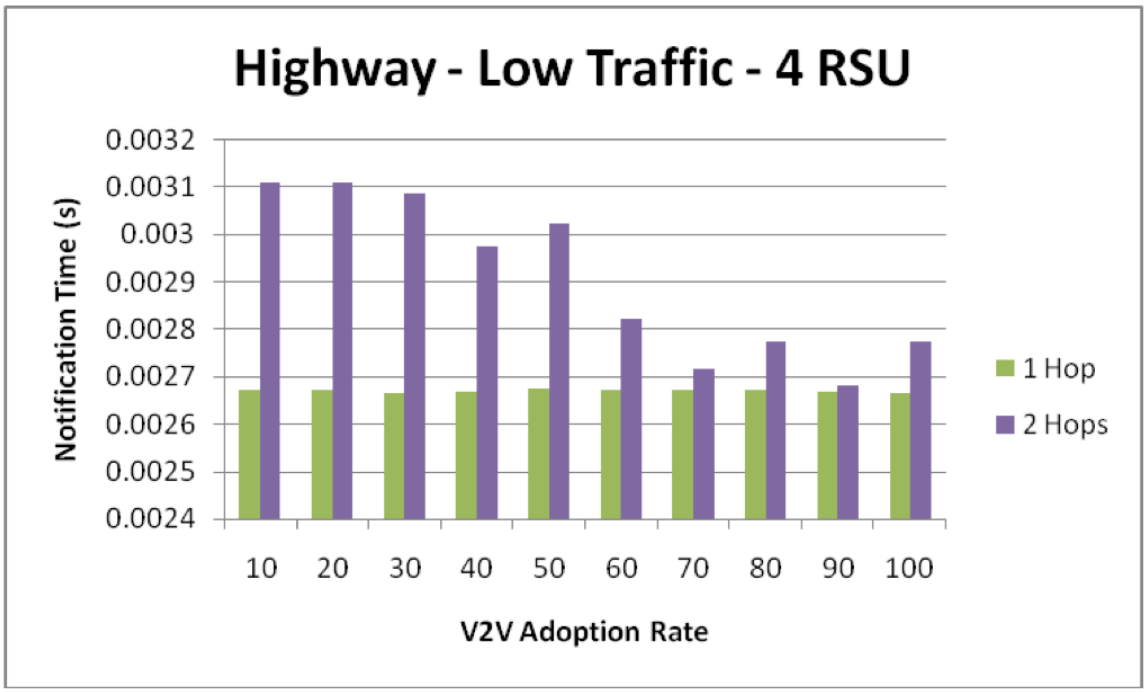

Figure 6-15. Notification time for highway in low traffic density with four roadside units

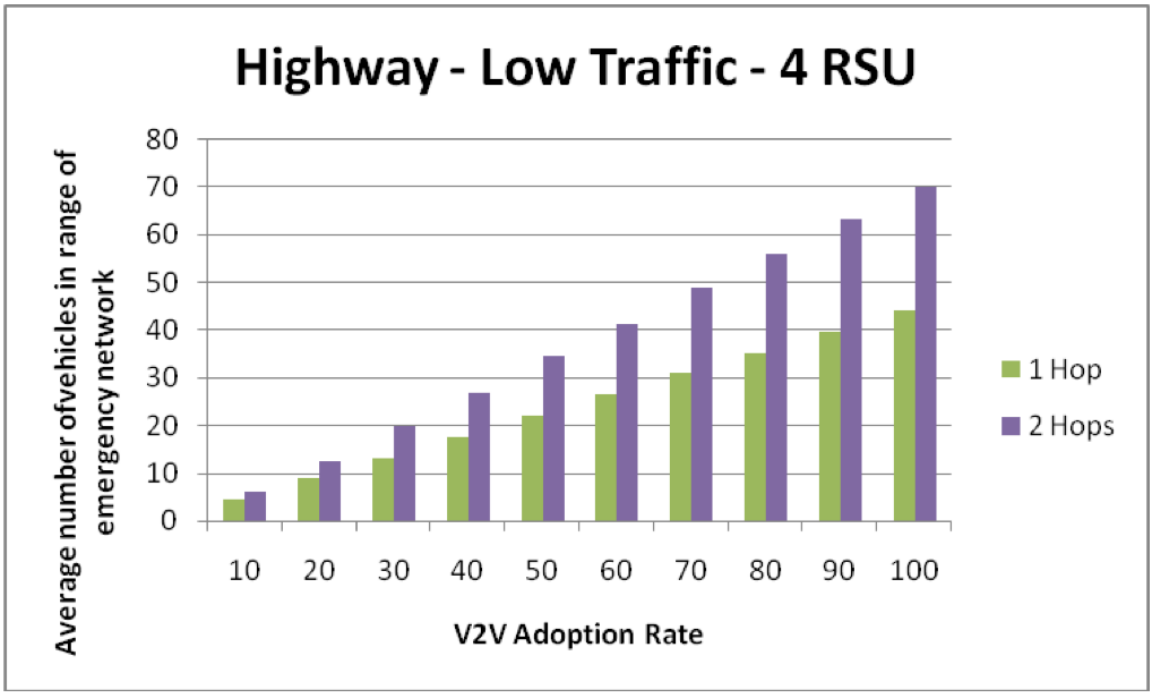

Figure 6-16. Number of covered vehicles for highway in low traffic density with four roadside units

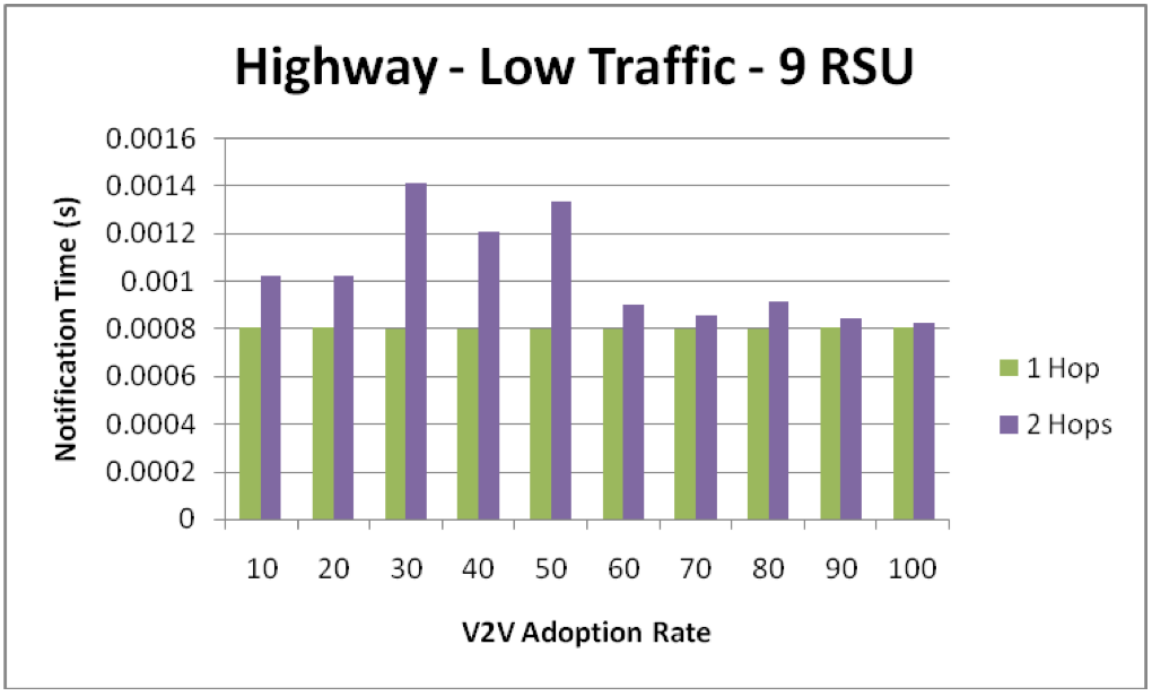

Figure 6-17. Notification time for highway in low traffic density with nine roadside units

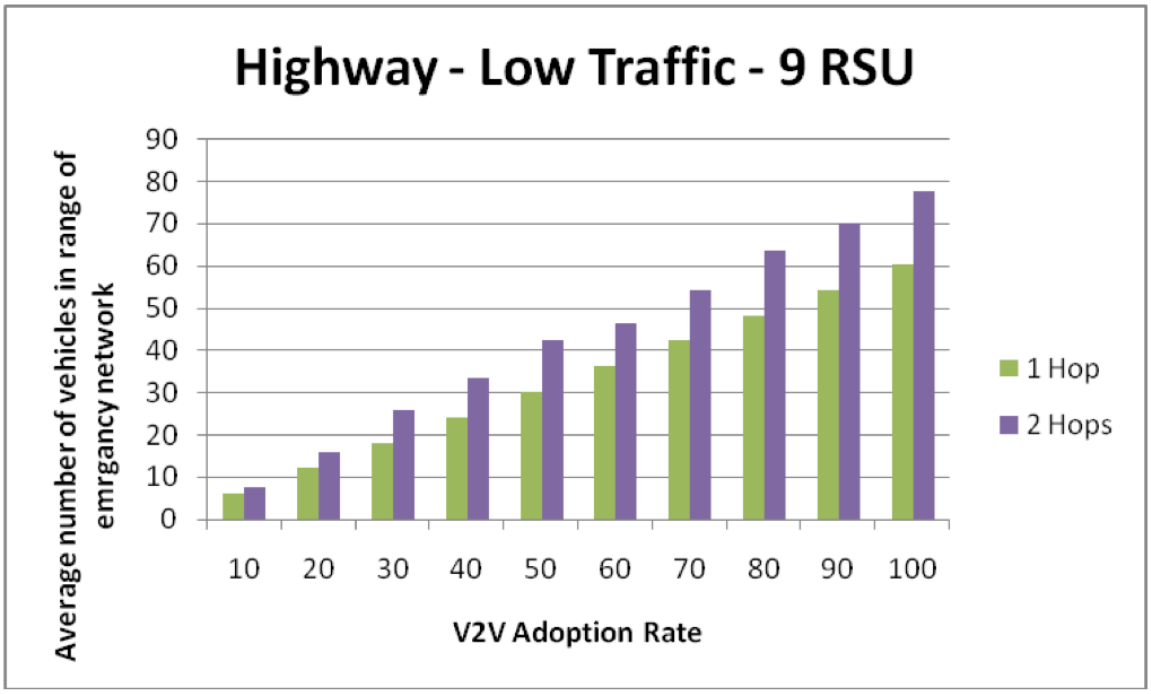

Figure 6-18. Number of covered vehicles for highway in low traffic density with nine roadside units

6.5 Conclusion

I have used a decision-making system to design an emergency message notification system simulation and examined the effects of traffic location, traffic density, V2V adoption rate, number of hops in V2V, and number of RSUs installed on vehicular networks on the time to notify emergency services. Generally, the results show that in higher traffic density the notification time is lower. The results also show that traffic density, number of RSUs (and likely placement), and V2V/I adoption rate impacts notification time. The interplay among these factors is not clear. As the adoption rate increases, the potential value of more RSUs seems to diminish. With two hops, there is increased coverage. The reason for increasing notification time with use of more hops may not be clear at first but more precise analysis of result specifies that the reason is

higher network coverage. With use of two hops all of the vehicles that have been covered in the same situation but with use of one hop are covered with same notification time. In addition, there are some more vehicles that have not been covered with use of one hop but with use of two hops they are covered and they can communicate with at least one RSU. The second group of covered vehicles have higher notification time since their messages should go through more hops to reach a RSU. Having first group of covered vehicles with same notification time and second group of covered vehicles with higher notification time results in higher average notification time for the situation with two hops comparing to one hop. Higher adoption rates have a positive impact on the notification time. This is not surprising given the increased coverage. The higher number of RSUs is also associated with decreased notification times. This impact may be more pronounced if the RSUs were less frequent. One could easily imagine the same level of performance if the RSUs were further apart and there were more hops. These trade-offs require further study.

7 Conclusion and Future Work

I have implemented a decision making module for vehicles which collects data from available sources of information and analyses this data to provide the best action to be taken. The proposed module uses three agents to collect data: vehicle agent, wireless agent (V2V and V2I) and camera agent. Using this module, I have evaluated the approach with a cooperative collision warning and rerouting application and a vehicle tracking system.

The main contributions of this work are:

- A novel in-vehicular decision making system based on blackboard architectural model which collects raw data from available sources of information using its agents and process these data and provides the best action to be taken. The decision making system works in conjunction with one or more IV applications simultaneously.
- Four intelligent vehicle application which operate using decision making system
- Extend Veins Simulator to accommodate camera simulator and the decision making system which uses 2D mathematical model of road network obtained from SUMO and determines if an object is detectable by a camera installed on the vehicle or not.

The presented cooperative collision warning and rerouting system uses wireless agent's data to avoid accidents and to suggest alternative routes in case of an accident. My application contributes to research in the following ways:

- CCW systems have mostly been used to provide warning information for the driver and do not suggest possible actions to be taken. These systems inform the following

vehicle about the potential collision, but do not provide rerouting choices for the driver which can help avoid traffic congestion and reduce waiting time. My decision making algorithm provides an alternative route for vehicles approaching the accident location in order to decrease waiting and travel time and avoid traffic using VANET communication.

- My decision making module is the first event based decision making approach for collision warning and rerouting system based on wireless communication. My decision making system triggers when an accident happens and the car which has been in the accident sends accident message(s). In other proposed rerouting algorithms, vehicles send request messages to other vehicles in order to find out about traffic congestion based on the responses. In the presented system there is no need to continue sending redundant messages and this reduces channel bandwidth by not sending unnecessary messages.
- My decision making module does not need a central management control to collect the information from vehicles and manage the following traffic. Any equipped vehicle can provide the necessary warning messages to other vehicles.
- My system uses a specific “resending” accident messages alongside the propagating messages for one or two hop(s) in order to make sure that all needed vehicles are aware of the accident and can take action to reroute to avoid the traffic jam caused by the accident.
- Since driving in highways is different than driving in urban areas, I evaluated my system in different areas to compare the differences based on driving areas.
- The other factor which can be important is traffic congestion. The decision making module chooses different scenarios in high traffic and normal traffic. High traffic is detected based on the speed of the vehicle and the maximum legal speed of the road it is traveling on.

Knowledge of neighboring vehicles can provide useful information for safety intelligent vehicle (IV) applications, such as collision warning systems. One approach to vehicle tracking via image processing systems is by mounting cameras on the vehicles. However, if a vehicle is behind other objects the camera is not able to detect as it would be out of view. Wireless communications between vehicles could be an alternative to this problem. Vehicles could exchange information about their location, speed, acceleration, etc. Having this information gathered from other vehicles, I can locate neighboring vehicles. However, what if not all the vehicles are equipped with a Dedicated Short Range Communication (DSRC) transceiver. In that case, using wireless communication alone among the vehicles for vehicle tracking cannot be a completely effective solution.

To overcome these shortcomings, I have proposed a new method of vehicle tracking as an application of the decision making module which uses both technologies together to track the vehicles. In my vehicle tracking method, each vehicle sends a map request via wireless to other vehicles in range and based on their response updates its own information. This system does not assume that all the vehicles are equipped. My vehicle tracking method contributes to research in the following ways:

- All previous tracking methods have used either wireless communication or vision based systems to detect neighboring objects and provide a view of surrounding environment. In the proposed tracking system, both of these technologies have been used to overcome their respective limitations and provide more reliable and more accurate information about the objects around the vehicle.
- My vehicle tracking method does not rely on other vehicles and it can work in the situation in which no nearby vehicle is equipped with cameras or wireless technology. In this case the system just uses its own information obtained from its cameras. Generally, my system can work if all the vehicles are equipped with both camera and wireless communication or one of the technologies.

- My system works well in specific traffic situations, such as reaching an intersection or in low light situations, where other tracking methods cannot operate well.

I have proposed a novel emergency message propagation system which uses both V2V and V2I communications in order to notify emergency services about the the emergency situations such as an accident. This emergency message propagation system contributes to research in the following ways:

- This VI application is a novel emergency message propagation system using decision making system and both V2V and V2I to decrease notification time for emergency systems.
- My system provides more information about the accident to emergency services such as the estimated duration of the accident to determine accident severity.
- I have considered traffic location, traffic density, Number of hops, etc. on system operation.

Research on using a decision making system for vehicles is relatively recent with the potential for significant results and applications in near future. Here are a few possible research areas that may be undertaken immediately:

- Driver information such as driver gaze, heartbeat etc. could have major influence on driving situation. Adding driver behavior data as a new data agent to the proposed decision making system could improve the overall result with making the whole system personalized by the driver.
- The first step of each technology could be testing the design on simulation. However, designing and developing a system on real world could provide more realistic overview of the system and show possible issues and limitations.

Overall, the results show that using a decision making module shows great potential for improving performance of vehicular systems by reducing travel time and wait time and

providing more accurate information about the surrounding environment for vehicles. In addition, the safety of vehicles will increase since the vehicles will be informed about the accident by wireless communication.

References

[1] J. Jansson, J. Johansson, and F. Gustafsson, “Decision making for collision avoidance systems,” *Soc. Automot. Eng. SAE*, no. 2002–01, p. 0403, 2002.

[2] J. Jansson and F. Gustafsson, “A framework and automotive application of collision avoidance decision making,” *Automatica*, vol. 44, no. 9, pp. 2347–2351, 2008.

[3] J. Hillenbrand, A. M. Spieker, and K. Kroschel, “A multilevel collision mitigation approach—Its situation assessment, decision making, and performance tradeoffs,” *Intell. Transp. Syst. IEEE Trans. On*, vol. 7, no. 4, pp. 528–540, 2006.

[4] R. Karlsson, J. Jansson, and F. Gustafsson, “Model-based statistical tracking and decision making for collision avoidance application,” in *American Control Conference, 2004. Proceedings of the 2004*, 2004, vol. 4, pp. 3435–3440.

[5] D. Gruyer, S. Demmel, B. d’Andrea-Novel, A. Lambert, and A. Rakotonirainy, “Simulation architecture for the design of Cooperative Collision Warning systems,” in *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, 2012, pp. 697–703.

[6] S. Eichler, B. Ostermaier, C. Schroth, and T. Kosch, “Simulation of car-to-car messaging: Analyzing the impact on road traffic,” in *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2005. 13th IEEE International Symposium on*, 2005, pp. 507–510.

[7] C. Sommer, R. German, and F. Dressler, “Bidirectionally coupled network and road traffic simulation for improved IVC analysis,” *Mob. Comput. IEEE Trans. On*, vol. 10, no. 1, pp. 3–15, 2011.

[8] D. D. Corkill, “Blackboard systems,” *AI Expert*, vol. 6, no. 9, pp. 40–47, 1991.

[9] D. Eckhoff, C. Sommer, and F. Dressler, “On the Necessity of Accurate IEEE 802.11 p Models for IVC Protocol Simulation,” in *Vehicular Technology Conference (VTC Spring), 2012 IEEE 75th*, 2012, pp. 1–5.

[10] D. Eckhoff and C. Sommer, “A Multi-Channel IEEE 1609.4 and 802.11 p EDCA Model for the Veins Framework,” in *Proceedings of 5th ACM/ICST International Conference on Simulation Tools and Techniques for Communications, Networks and Systems: 5th ACM/ICST International Workshop on OMNeT++.(Desenzano, Italy, 19-23 March, 2012). OMNeT+*, 2012.

[11] D. Krajzewicz, G. Hertkorn, C. Rössel, and P. Wagner, “Sumo (simulation of urban mobility),” in *Proc. of the 4th Middle East Symposium on Simulation and Modelling*, 2002, pp. 183–187.

[12] A. Varga, “The OMNeT++ discrete event simulation system,” in *Proceedings of the European Simulation Multiconference (ESM’2001)*, 2001, vol. 9.

[13] R. Sengupta, S. Rezaei, S. E. Shladover, D. Cody, S. Dickey, and H. Krishnan, “Cooperative collision warning systems: Concept definition and experimental implementation,” *J. Intell. Transp. Syst.*, vol. 11, no. 3, pp. 143–155, 2007.

[14] H. S. Tan and J. Huang, “DGPS-based vehicle-to-vehicle cooperative collision warning: Engineering feasibility viewpoints,” *Intell. Transp. Syst. IEEE Trans. On*, vol. 7, no. 4, pp. 415–428, 2006.

[15] S. Dashtinezhad, T. Nadeem, B. Dorohonceanu, C. Borcea, P. Kang, and L. Iftode, “TrafficView: a driver assistant device for traffic monitoring based on car-to-car communication,” in *Vehicular Technology Conference, 2004. VTC 2004-Spring. 2004 IEEE 59th*, 2004, vol. 5, pp. 2946–2950.

[16] T. Nadeem, S. Dashtinezhad, C. Liao, and L. Iftode, “TrafficView: traffic data dissemination using car-to-car communication,” *ACM SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 8, no. 3, pp. 6–19, 2004.

[17] X. Yang, L. Liu, N. H. Vaidya, and F. Zhao, “A vehicle-to-vehicle communication protocol for cooperative collision warning,” in *Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on*, 2004, pp. 114–123.

[18] S. Biswas, R. Tatchikou, and F. Dion, “Vehicle-to-vehicle wireless communication protocols for enhancing highway traffic safety,” *Commun. Mag. IEEE*, vol. 44, no. 1, pp. 74–82, 2006.

[19] C. L. Huang, Y. P. Fallah, R. Sengupta, and H. Krishnan, “Adaptive intervehicle communication control for cooperative safety systems,” *Netw. IEEE*, vol. 24, no. 1, pp. 6–13, 2010.

[20] T. ElBatt, S. K. Goel, G. Holland, H. Krishnan, and J. Parikh, “Cooperative collision warning using dedicated short range wireless communications,” in *Proceedings of the 3rd international workshop on Vehicular ad hoc networks*, 2006, pp. 1–9.

[21] A. Lakas and M. Cheqfah, “Detection and dissipation of road traffic congestion using vehicular communication,” in *Microwave Symposium (MMS), 2009 Mediterrannean*, 2009, pp. 1–6.

[22] A. Dogan, G. Korkmaz, Y. Liu, F. Ozguner, U. Ozguner, K. Redmill, O. Takeshita, and K. Tokuda, “Evaluation of intersection collision warning system using an inter-vehicle communication simulator,” in *Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference on*, 2004, pp. 1103–1108.

[23] F. Zong, H. Zhang, H. Xu, X. Zhu, and L. Wang, “Predicting Severity and Duration of Road Traffic Accident,” *Math. Probl. Eng.*, vol. 2013, 2013.

[24] M. P. Gardner, “Highway traffic monitoring,” *Transp. Res. Board Transp. New Millenn.*, p. 5, 2000.

[25] S. Coleri, S. Y. Cheung, and P. Varaiya, “Sensor networks for monitoring traffic,” in *Allerton conference on communication, control and computing*, 2004, pp. 32–40.

[26] C. Li, K. Ikeuchi, and M. Sakauchi, “Acquisition of traffic information using a video camera with 2D spatio-temporal image transformation technique,” in *Intelligent Transportation Systems, 1999. Proceedings. 1999 IEEE/IEEJ/JSAI International Conference on*, 1999, pp. 634–638.

[27] V. Milanés, J. Villagra, J. Godoy, J. Simó, J. Pérez, and E. Onieva, “An intelligent V2I-based traffic management system,” *Intell. Transp. Syst. IEEE Trans. On*, vol. 13, no. 1, pp. 49–58, 2012.

[28] A. F. Santamaria, C. Sottile, A. Lupia, and P. Raimondo, “An efficient traffic management protocol based on IEEE802. 11p standard,” in *Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2014), International Symposium on*, 2014, pp. 634–641.

[29] I. Leontiadis, G. Marfia, D. Mack, G. Pau, C. Mascolo, and M. Gerla, “On the effectiveness of an opportunistic traffic management system for vehicular networks,” *Intell. Transp. Syst. IEEE Trans. On*, vol. 12, no. 4, pp. 1537–1548, 2011.

[30] F. Knorr, D. Baselt, M. Schreckenberg, and M. Mauve, “Reducing traffic jams via VANETs,” *Veh. Technol. IEEE Trans. On*, vol. 61, no. 8, pp. 3490–3498, 2012.

[31] S. Kaul, M. Gruteser, V. Rai, and J. Kenney, “Minimizing age of information in vehicular networks,” in *Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2011 8th Annual IEEE Communications Society Conference on*, 2011, pp. 350–358.

[32] Y. S. Sun, L. Xie, Q. A. Chen, S. Lu, and D. Chen, “Efficient Route Guidance in Vehicular Wireless Networks.”

[33] Y. Chung, “Development of an accident duration prediction model on the Korean Freeway Systems,” *Accid. Anal. Prev.*, vol. 42, no. 1, pp. 282–289, 2010.

[34] Y. Chung and W. W. Recker, “A methodological approach for estimating temporal and spatial extent of delays caused by freeway accidents,” *Intell. Transp. Syst. IEEE Trans. On*, vol. 13, no. 3, pp. 1454–1461, 2012.

[35] B. Zardosht, S. Beauchemin, and M. A. Bauer, “A decision making module for cooperative collision warning systems using Vehicular Ad-Hoc Networks,” in *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*, 2013, pp. 1743–1749.

[36] M. Haklay and P. Weber, “Openstreetmap: User-generated street maps,” *Pervasive Comput. IEEE*, vol. 7, no. 4, pp. 12–18, 2008.

[37] A. Varga, “The OMNeT++ discrete event simulation system,” in *Proceedings of the European Simulation Multiconference (ESM’2001)*, 2001, vol. 9, p. 185.

[38] D. Krajzewicz, G. Hertkorn, C. Rössel, and P. Wagner, “Sumo (simulation of urban mobility),” in *Proc. of the 4th Middle East Symposium on Simulation and Modelling*, 2002, pp. 183–187.

[39] C. Sommer, R. German, and F. Dressler, “Bidirectionally coupled network and road traffic simulation for improved IVC analysis,” *Mob. Comput. IEEE Trans. On*, vol. 10, no. 1, pp. 3–15, 2011.

[40] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik, “A real-time computer vision system for measuring traffic parameters,” in *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, 1997, pp. 495–501.

[41] B. Coifman, D. Beymer, P. McLauchlan, and J. Malik, “A real-time computer vision system for vehicle tracking and traffic surveillance,” *Transp. Res. Part C Emerg. Technol.*, vol. 6, no. 4, pp. 271–288, 1998.

[42] M. Bertozzi, A. Broggi, A. Fascioli, and S. Nichele, “Stereo vision-based vehicle detection,” in *Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE*, 2000, pp. 39–44.

[43] M. Betke, E. Haritaoglu, and L. S. Davis, “Multiple vehicle detection and tracking in hard real-time,” in *Intelligent Vehicles Symposium, 1996., Proceedings of the 1996 IEEE*, 1996, pp. 351–356.

[44] A. Andreas, F. Jannik, and B. Martin, “Improved Tracking and Behavior Anticipation by Combining Map Information with Bayesian-Filtering,” in *Proceedings of the 16th International IEEE Annual Conference on Intelligent Transportation Systems*, The Hague, The Netherlands, 2013.

[45] B. Gloyer, H. K. Aghajan, K.-Y. Siu, and T. Kailath, “Video-based freeway-monitoring system using recursive vehicle tracking,” in *IS&T/SPIE’s Symposium on Electronic Imaging: Science & Technology*, 1995, pp. 173–180.

[46] S. Rezaei, R. Sengupta, H. Krishnan, X. Guan, and R. Bhatia, “Tracking the position of neighboring vehicles using wireless communications,” *Transp. Res. Part C Emerg. Technol.*, vol. 18, no. 3, pp. 335–350, 2010.

[47] K. Shafiee and V. Leung, “Connectivity-aware minimum-delay geographic routing with vehicle tracking in VANETs,” *Ad Hoc Netw.*, vol. 9, no. 2, pp. 131–141, 2011.

[48] C.-L. Huang, Y. P. Fallah, R. Sengupta, and H. Krishnan, “Adaptive intervehicle communication control for cooperative safety systems,” *Netw. IEEE*, vol. 24, no. 1, pp. 6–13, 2010.

[49] Y. P. Fallah, C.-L. Huang, R. Sengupta, and H. Krishnan, “Analysis of information dissemination in vehicular ad-hoc networks with application to cooperative vehicle safety systems,” *Veh. Technol. IEEE Trans. On*, vol. 60, no. 1, pp. 233–247, 2011.

[50] J. Barrachina, P. Garrido, M. Fogue, F. J. Martinez, J.-C. Cano, C. T. Calafate, and P. Manzoni, “Road side unit deployment: A density-based approach,” *Intell. Transp. Syst. Mag. IEEE*, vol. 5, no. 3, pp. 30–39, 2013.

[51] M. L. Sichitiu and M. Kihl, “Inter-vehicle communication systems: a survey,” *Commun. Surv. Tutor. IEEE*, vol. 10, no. 2, pp. 88–105, 2008.

[52] X. Yang, J. Liu, N. H. Vaidya, and F. Zhao, “A vehicle-to-vehicle communication protocol for cooperative collision warning,” in *Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on*, 2004, pp. 114–123.

[53] T. ElBatt, S. K. Goel, G. Holland, H. Krishnan, and J. Parikh, “Cooperative collision warning using dedicated short range wireless communications,” in *Proceedings of the 3rd international workshop on Vehicular ad hoc networks*, 2006, pp. 1–9.

[54] H.-S. Tan and J. Huang, “DGPS-based vehicle-to-vehicle cooperative collision warning: Engineering feasibility viewpoints,” *Intell. Transp. Syst. IEEE Trans. On*, vol. 7, no. 4, pp. 415–428, 2006.

[55] R. Sengupta, S. Rezaei, S. E. Shladover, D. Cody, S. Dickey, and H. Krishnan, “Cooperative collision warning systems: concept definition and experimental implementation,” *J. Intell. Transp. Syst.*, vol. 11, no. 3, pp. 143–155, 2007.

[56] J. Santa, A. F. Gómez-Skarmeta, and M. Sánchez-Artigas, “Architecture and evaluation of a unified V2V and V2I communication system based on cellular networks,” *Comput. Commun.*, vol. 31, no. 12, pp. 2850–2861, 2008.

[57] A. M. Vegni and T. D. Little, “Hybrid vehicular communications based on V2V-V2I protocol switching,” *Int. J. Veh. Inf. Commun. Syst.*, vol. 2, no. 3–4, pp. 213–231, 2011.

[58] F. J. Martinez, C.-K. Toh, J.-C. Cano, C. T. Calafate, and P. Manzoni, “Emergency services in future intelligent transportation systems based on vehicular communication networks,” *Intell. Transp. Syst. Mag. IEEE*, vol. 2, no. 2, pp. 6–20, 2010.

[59] A. M. Vegni and T. D. Little, “A message propagation model for hybrid vehicular communication protocols,” in *Communication Systems Networks and Digital Signal Processing (CSNDSP), 2010 7th International Symposium on*, 2010, pp. 382–386.

[60] O. Maeshima, S. Cai, T. Honda, and H. Urayama, “A roadside-to-vehicle communication system for vehicle safety using dual frequency channels,” in *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, 2007, pp. 349–354.

[61] D. Camara, C. Bonnet, and F. Filali, “Propagation of public safety warning messages: a delay tolerant network approach,” in *Wireless Communications and Networking Conference (WCNC), 2010 IEEE*, 2010, pp. 1–6.

[62] H. Zhu, X. Lin, R. Lu, P.-H. Ho, and X. S. Shen, “AEMA: An aggregated emergency message authentication scheme for enhancing the security of vehicular ad hoc networks,” in *Communications, 2008. ICC’08. IEEE International Conference on*, 2008, pp. 1436–1440.

[63] J. Barrachina, P. Garrido, M. Fogue, F. J. Martinez, J.-C. Cano, C. T. Calafate, and P. Manzoni, “Road side unit deployment: A density-based approach,” *Intell. Transp. Syst. Mag. IEEE*, vol. 5, no. 3, pp. 30–39, 2013.

[64] C. Lochert, B. Scheuermann, C. Wewetzer, A. Luebke, and M. Mauve, “Data aggregation and roadside unit placement for a VANET traffic information system,” in *Proceedings of the fifth ACM international workshop on VehiculAr Inter-NETworking*, 2008, pp. 58–65.

[65] A. Abdrabou and W. Zhuang, “Probabilistic delay control and road side unit placement for vehicular ad hoc networks with disrupted connectivity,” *Sel. Areas Commun. IEEE J. On*, vol. 29, no. 1, pp. 129–139, 2011.

[66] T.-J. Wu, W. Liao, and C.-J. Chang, “A cost-effective strategy for road-side unit placement in vehicular networks,” *Commun. IEEE Trans. On*, vol. 60, no. 8, pp. 2295–2303, 2012.

[67] J. Barrachina, P. Garrido, M. Fogue, F. J. Martinez, J.-C. Cano, C. T. Calafate, and P. Manzoni, “D-RSU: a density-based approach for road side unit deployment in urban scenarios,” in *International workshop on ipv6-based vehicular networks (Vehi6), collocated with the 2012 IEEE intelligent vehicles symposium*, 2012, pp. 1–6.

[68] P. Li, C. Huang, and Q. Liu, “Delay Bounded Roadside Unit Placement in Vehicular Ad Hoc Networks,” *Int. J. Distrib. Sens. Netw.*, 2015.

[69] F. Zong, H. Zhang, H. Xu, X. Zhu, and L. Wang, “Predicting Severity and Duration of Road Traffic Accident,” *Math. Probl. Eng.*, vol. 2013, 2013.

Appendices

To simulate camera operations, I have implemented a vision simulator which is defined for each vehicle. Each camera is defined by three factors; range, angle and direction. Having these factors and the mathematical model of road network it is possible to figure out if an obstacle is detectable by a camera or not. Followings are the codes which provide these calculations.

This code is written in c++ and as an Omnet++ component.

Appendix A: Vision simulator code

```
Shape TraCIDemo11p::Vehicle_to_Shape(Point Position, double Direction, double
Lenght, double Width)
{
   Shape VehicleShape;
   Point VehiclePoint;
   double r;
   double Teta,Gamma,Alpha;

   r=pow((pow(Width/2,2)+pow(Lenght/2,2)),0.5);
   Gamma=atan(Width/Lenght);
   Teta=(Direction*PI)/180;// degree to radian

   Alpha=Teta-Gamma;
   VehiclePoint.x=cos(Alpha)*r+Position.x;
   VehiclePoint.y=sin(Alpha)*r+Position.y;
   VehiclePoint.Visible=true;
   VehicleShape.PointList.push_front(VehiclePoint);

   Alpha=Teta+Gamma;
   VehiclePoint.x=cos(Alpha)*r+Position.x;
   VehiclePoint.y=sin(Alpha)*r+Position.y;
   VehiclePoint.Visible=true;
   VehicleShape.PointList.push_front(VehiclePoint);

   Alpha=((180*PI)/180)+Teta-Gamma;
```

```
   VehiclePoint.x=cos(Alpha)*r+Position.x;
   VehiclePoint.y=sin(Alpha)*r+Position.y;
   VehiclePoint.Visible=true;
   VehicleShape.PointList.push_front(VehiclePoint);

   Alpha=((180*PI)/180)+Teta+Gamma;
   VehiclePoint.x=cos(Alpha)*r+Position.x;
   VehiclePoint.y=sin(Alpha)*r+Position.y;
   VehiclePoint.Visible=true;
   VehicleShape.PointList.push_front(VehiclePoint);


   return VehicleShape;
}

bool TraCIDemo11p::Check_Point_Inside(Camera Camera, Point Point)
{
   double Alpha1,Alpha2;
   Alpha1=((Camera.Angle1+Camera.Direction)*PI)/180;
   Alpha2=((Camera.Angle2+Camera.Direction)*PI)/180;

   if (
         (
            (pow((Point.x-Camera.Position.x),2)+pow((Point.y-
Camera.Position.y),2))<=pow(Camera.Distance,2)
         )
       and (
             ((cos(Alpha1)==0) and (sin(Alpha1)>0) and (Point.x >= Camera.Position.x))
          or ((cos(Alpha1)==0) and (sin(Alpha1)<0) and (Point.x <= Camera.Position.x))
          or ((cos(Alpha1)>0) and ((Point.y-tan(Alpha1)*Point.x)<=(Camera.Position.y-
tan(Alpha1)*Camera.Position.x)))
          or ((cos(Alpha1)<0) and ((Point.y-tan(Alpha1)*Point.x)>=(Camera.Position.y-
tan(Alpha1)*Camera.Position.x)))
         )
       and (
             ((cos(Alpha2)==0) and (sin(Alpha2)>0) and (Point.x <= Camera.Position.x))
          or ((cos(Alpha2)==0) and (sin(Alpha2)<0) and (Point.x >= Camera.Position.x))
          or ((cos(Alpha2)>0) and ((Point.y-tan(Alpha2)*Point.x)>=(Camera.Position.y-
tan(Alpha2)*Camera.Position.x)))
          or ((cos(Alpha2)<0) and ((Point.y-tan(Alpha2)*Point.x)<=(Camera.Position.y-
tan(Alpha2)*Camera.Position.x)))
         )
```

```
      )
   {
      return true;
   }
   else
   {
      return false;
   }

}

std::list<Point> TraCIDemo11p::Determine_Outermost_Points(Camera Camera, Shape
Shape)
{
   std::list<Point> TempPointList,Outermosts;
   int ListSize=Shape.PointList.size();
   double MaxAngle;
   Point Point1,Point2,Outermost1,Outermost2;
   double Angle,Angle1,Angle2;

   MaxAngle= -1;
   TempPointList=Shape.PointList;

   Point1=TempPointList.front();
   if (Camera.Position.x==Point1.x)
   {
      if (Camera.Position.y>Point1.y) Angle1=270;
      if (Camera.Position.y<Point1.y) Angle1=90;
   }
   else Angle1=(atan((Camera.Position.y-Point1.y)/(Camera.Position.x-
Point1.x))*180)/PI;
   if (Camera.Position.x>Point1.x) Angle1+=180;
   if ((Angle1<0)and(Camera.Position.x<Point1.x)) Angle1+=360;
   for (int i=0;i<ListSize;i++)
   {
      Point2=TempPointList.front();
      TempPointList.pop_front();
      if (Camera.Position.x==Point2.x)
      {
         if (Camera.Position.y>Point2.y) Angle2=270;
         if (Camera.Position.y<Point2.y) Angle2=90;
      }
```

```
      else Angle2=(atan((Camera.Position.y-Point2.y)/(Camera.Position.x-
Point2.x))*180)/PI;
      if (Camera.Position.x>Point2.x) Angle2+=180;
      if ((Angle2<0)and(Camera.Position.x<Point2.x)) Angle2+=360;
      Angle=fabs(Angle1-Angle2);
      if (Angle>=180) Angle-=180;
      if (Angle>MaxAngle)
      {
         MaxAngle=Angle;
         Outermost1=Point2;
      }
   }

   TempPointList=Shape.PointList;
   MaxAngle= -1;
   Point1=Outermost1;
   if (Camera.Position.x==Point1.x)
   {
      if (Camera.Position.y>Point1.y) Angle1=270;
      if (Camera.Position.y<Point1.y) Angle1=90;
   }
   else Angle1=(atan((Camera.Position.y-Point1.y)/(Camera.Position.x-
Point1.x))*180)/PI;
   if (Camera.Position.x>Point1.x) Angle1+=180;
   if ((Angle1<0)and(Camera.Position.x<Point1.x)) Angle1+=360;
   for (int i=0;i<ListSize;i++)
   {
      Point2=TempPointList.front();
      TempPointList.pop_front();
      if (Camera.Position.x==Point2.x)
      {
         if (Camera.Position.y>Point2.y) Angle2=270;
         if (Camera.Position.y<Point2.y) Angle2=90;
      }
      else Angle2=(atan((Camera.Position.y-Point2.y)/(Camera.Position.x-
Point2.x))*180)/PI;
      if (Camera.Position.x>Point2.x) Angle2+=180;
      if ((Angle2<0)and(Camera.Position.x<Point2.x)) Angle2+=360;
      Angle=fabs(Angle1-Angle2);
      if (Angle>=180) Angle-=180;
      if (Angle>MaxAngle)
      {
```

```
            MaxAngle=Angle;
            Outermost2=Point2;
        }
    }
    Outermosts.push_front(Outermost1);
    Outermosts.push_front(Outermost2);
    return Outermosts;
}

std::list<Shape> TraCIDemo11p::Determine_Shape_Visibility(Camera
Camera,std::list<Shape> Shapes)
{

    Shape Shape1, Shape2, TempSahpe;
    int SSize = Shapes.size();
    double y1, y2, y3, y4, x1, x2, x3, x4;

    for (int s = 0; s < SSize; s++) {
        Shape1 = Shapes.front();
        Shapes.pop_front();
        Shape1.OutermostVisiblePoint1 = Shape1.OutermostPoint1;
        Shape1.OutermostVisiblePoint2 = Shape1.OutermostPoint2;
        Shape1.Visibility = 100;
        Shapes.push_back(Shape1);
    }

    for (int i = 0; i < SSize; i++) {
        Shape1 = Shapes.front();
        Shapes.pop_front();
        for (int j = 0; j < SSize - 1; j++) {
            int area1 = 0;
            int area2 = 0;
            double VisibleLenght = 0;
            Shape2 = Shapes.front();
            Shapes.pop_front();
            if (((((Shape2.OutermostVisiblePoint1.y
                    - Shape1.OutermostVisiblePoint1.y)
                    * (Shape1.OutermostVisiblePoint2.x
                            - Shape1.OutermostVisiblePoint1.x))
                    - ((Shape1.OutermostVisiblePoint2.y
                            - Shape1.OutermostVisiblePoint1.y)
                            * (Shape2.OutermostVisiblePoint1.x
```

```
                - Shape1.OutermostVisiblePoint1.x)))
    * (((Camera.Position.y - Shape1.OutermostVisiblePoint1.y)
            * (Shape1.OutermostVisiblePoint2.x
                - Shape1.OutermostVisiblePoint1.x))
            - ((Shape1.OutermostVisiblePoint2.y
                - Shape1.OutermostVisiblePoint1.y)
                * (Camera.Position.x
                    - Shape1.OutermostVisiblePoint1.x))))
    < 0) {
if (((((Shape2.OutermostVisiblePoint1.y
        - Shape1.OutermostVisiblePoint1.y)
        * (Camera.Position.x - Shape1.OutermostVisiblePoint1.x))
        - ((Camera.Position.y - Shape1.OutermostVisiblePoint1.y)
                * (Shape2.OutermostVisiblePoint1.x
                    - Shape1.OutermostVisiblePoint1.x)))
        * (((Shape1.OutermostVisiblePoint2.y
                - Shape1.OutermostVisiblePoint1.y)
                * (Camera.Position.x
                    - Shape1.OutermostVisiblePoint1.x))
                - ((Camera.Position.y
                    - Shape1.OutermostVisiblePoint1.y)
                    * (Shape1.OutermostVisiblePoint2.x
                        - Shape1.OutermostVisiblePoint1.x))))
        > 0) {
    if (((((Shape2.OutermostVisiblePoint1.y
            - Shape1.OutermostVisiblePoint2.y)
            * (Camera.Position.x
                - Shape1.OutermostVisiblePoint2.x))
            - ((Camera.Position.y
                - Shape1.OutermostVisiblePoint2.y)
                * (Shape2.OutermostVisiblePoint1.x
                    - Shape1.OutermostVisiblePoint2.x)))
            * (((Shape1.OutermostVisiblePoint1.y
                - Shape1.OutermostVisiblePoint2.y)
                * (Camera.Position.x
                    - Shape1.OutermostVisiblePoint2.x))
                - ((Camera.Position.y
                    - Shape1.OutermostVisiblePoint2.y)
                    * (Shape1.OutermostVisiblePoint1.x
                        - Shape1.OutermostVisiblePoint2.x))))
            > 0) {
        Shape2.OutermostVisiblePoint1.Visible = false;
```

```
            Shape2.OutermostPoint1.Visible = false;
            area1 = 0;
        } else
            area1 = 2;
    } else
        area1 = 1;
} else if (((((Shape2.OutermostVisiblePoint1.y
        - Shape1.OutermostVisiblePoint1.y)
        * (Camera.Position.x - Shape1.OutermostVisiblePoint1.x))
        - ((Camera.Position.y - Shape1.OutermostVisiblePoint1.y)
                * (Shape2.OutermostVisiblePoint1.x
                        - Shape1.OutermostVisiblePoint1.x)))
        * (((Shape1.OutermostVisiblePoint2.y
                - Shape1.OutermostVisiblePoint1.y)
                * (Camera.Position.x
                        - Shape1.OutermostVisiblePoint1.x))
                - ((Camera.Position.y
                        - Shape1.OutermostVisiblePoint1.y)
                        * (Shape1.OutermostVisiblePoint2.x
                                - Shape1.OutermostVisiblePoint1.x))))
        > 0) {
    if (((((Shape2.OutermostVisiblePoint1.y
            - Shape1.OutermostVisiblePoint2.y)
            * (Camera.Position.x - Shape1.OutermostVisiblePoint2.x))
            - ((Camera.Position.y - Shape1.OutermostVisiblePoint2.y)
                    * (Shape2.OutermostVisiblePoint1.x
                            - Shape1.OutermostVisiblePoint2.x)))
            * (((Shape1.OutermostVisiblePoint1.y
                    - Shape1.OutermostVisiblePoint2.y)
                    * (Camera.Position.x
                            - Shape1.OutermostVisiblePoint2.x))
                    - ((Camera.Position.y
                            - Shape1.OutermostVisiblePoint2.y)
                            * (Shape1.OutermostVisiblePoint1.x
                                    - Shape1.OutermostVisiblePoint2.x))))
            > 0)
        area1 = 5;
    else
        area1 = 4;
} else
    area1 = 3;
```

```
if (((((Shape2.OutermostVisiblePoint2.y
      - Shape1.OutermostVisiblePoint1.y)
      * (Shape1.OutermostVisiblePoint2.x
          - Shape1.OutermostVisiblePoint1.x))
      - ((Shape1.OutermostVisiblePoint2.y
          - Shape1.OutermostVisiblePoint1.y)
          * (Shape2.OutermostVisiblePoint2.x
              - Shape1.OutermostVisiblePoint1.x)))
      * (((Camera.Position.y - Shape1.OutermostVisiblePoint1.y)
          * (Shape1.OutermostVisiblePoint2.x
              - Shape1.OutermostVisiblePoint1.x))
          - ((Shape1.OutermostVisiblePoint2.y
              - Shape1.OutermostVisiblePoint1.y)
              * (Camera.Position.x
                  - Shape1.OutermostVisiblePoint1.x))))
      < 0) {
   if (((((Shape2.OutermostVisiblePoint2.y
        - Shape1.OutermostVisiblePoint1.y)
        * (Camera.Position.x - Shape1.OutermostVisiblePoint1.x))
        - ((Camera.Position.y - Shape1.OutermostVisiblePoint1.y)
            * (Shape2.OutermostVisiblePoint2.x
                - Shape1.OutermostVisiblePoint1.x)))
        * (((Shape1.OutermostVisiblePoint2.y
            - Shape1.OutermostVisiblePoint1.y)
            * (Camera.Position.x
                - Shape1.OutermostVisiblePoint1.x))
            - ((Camera.Position.y
                - Shape1.OutermostVisiblePoint1.y)
                * (Shape1.OutermostVisiblePoint2.x
                    - Shape1.OutermostVisiblePoint1.x))))
        > 0) {
      if (((((Shape2.OutermostVisiblePoint2.y
          - Shape1.OutermostVisiblePoint2.y)
          * (Camera.Position.x
              - Shape1.OutermostVisiblePoint2.x))
          - ((Camera.Position.y
              - Shape1.OutermostVisiblePoint2.y)
              * (Shape2.OutermostVisiblePoint2.x
                  - Shape1.OutermostVisiblePoint2.x)))
          * (((Shape1.OutermostVisiblePoint1.y
              - Shape1.OutermostVisiblePoint2.y)
              * (Camera.Position.x
```

```
                        - Shape1.OutermostVisiblePoint2.x))
                    - ((Camera.Position.y
                        - Shape1.OutermostVisiblePoint2.y)
                        * (Shape1.OutermostVisiblePoint1.x
                            - Shape1.OutermostVisiblePoint2.x))))
            > 0) {
          Shape2.OutermostVisiblePoint2.Visible = false;
          Shape2.OutermostPoint2.Visible = false;
          area2 = 0;
        } else
          area2 = 2;
      } else
        area2 = 1;
  } else if (((((Shape2.OutermostVisiblePoint1.y
        - Shape1.OutermostVisiblePoint1.y)
        * (Camera.Position.x - Shape1.OutermostVisiblePoint1.x))
        - ((Camera.Position.y - Shape1.OutermostVisiblePoint1.y)
            * (Shape2.OutermostVisiblePoint1.x
                - Shape1.OutermostVisiblePoint1.x)))
        * (((Shape1.OutermostVisiblePoint2.y
            - Shape1.OutermostVisiblePoint1.y)
            * (Camera.Position.x
                - Shape1.OutermostVisiblePoint1.x))
            - ((Camera.Position.y
                - Shape1.OutermostVisiblePoint1.y)
                * (Shape1.OutermostVisiblePoint2.x
                    - Shape1.OutermostVisiblePoint1.x))))
        > 0) {
    if (((((Shape2.OutermostVisiblePoint1.y
          - Shape1.OutermostVisiblePoint2.y)
          * (Camera.Position.x - Shape1.OutermostVisiblePoint2.x))
          - ((Camera.Position.y - Shape1.OutermostVisiblePoint2.y)
              * (Shape2.OutermostVisiblePoint1.x
                  - Shape1.OutermostVisiblePoint2.x)))
          * (((Shape1.OutermostVisiblePoint1.y
              - Shape1.OutermostVisiblePoint2.y)
              * (Camera.Position.x
                  - Shape1.OutermostVisiblePoint2.x))
              - ((Camera.Position.y
                  - Shape1.OutermostVisiblePoint2.y)
                  * (Shape1.OutermostVisiblePoint1.x
                      - Shape1.OutermostVisiblePoint2.x))))
```

```
            > 0)
         area2 = 5;
      else
         area2 = 4;
   } else
      area2 = 3;

   if ((area1 == 0) and (area2 == 0))
      VisibleLenght = 0;
   else if ((area1 == 0) and (area2 == 1)) {
      y1 = Shape2.OutermostPoint1.y;
      x1 = Shape2.OutermostPoint1.x;
      y2 = Shape2.OutermostPoint2.y;
      x2 = Shape2.OutermostPoint2.x;
      y3 = Camera.Position.y;
      x3 = Camera.Position.x;
      y4 = Shape1.OutermostVisiblePoint1.y;
      x4 = Shape1.OutermostVisiblePoint1.x;
      Shape2.OutermostVisiblePoint1.x =
            (((y2 - y1) * (x4 - x3) * (x1))
                  + ((y3 - y1) * (x2 - x1) * (x4 - x3))
                  + ((y3 - y4) * (x2 - x1) * (x3)))
                  / (((y2 - y1) * (x4 - x3))
                        - ((y4 - y3) * (x2 - x1)));
      Shape2.OutermostVisiblePoint1.y = (((y2 - y1)
            * (Shape2.OutermostVisiblePoint1.x - x1)) / (x2 - x1))
            + y1;
      Shape2.OutermostVisiblePoint1.Visible = true;
      y1 = Shape2.OutermostVisiblePoint1.y;
      x1 = Shape2.OutermostVisiblePoint1.x;
      y2 = Shape2.OutermostVisiblePoint2.y;
      x2 = Shape2.OutermostVisiblePoint2.x;
      VisibleLenght = pow(pow((y2 - y1), 2) + pow((x2 - x1), 2), 0.5);
   } else if ((area1 == 0) and (area2 == 2)) {
      y1 = Shape2.OutermostPoint1.y;
      x1 = Shape2.OutermostPoint1.x;
      y2 = Shape2.OutermostPoint2.y;
      x2 = Shape2.OutermostPoint2.x;
      y3 = Camera.Position.y;
      x3 = Camera.Position.x;
      y4 = Shape1.OutermostVisiblePoint2.y;
      x4 = Shape1.OutermostVisiblePoint2.x;
```

```
    Shape2.OutermostVisiblePoint1.x =
          (((y2 - y1) * (x4 - x3) * (x1))
                + ((y3 - y1) * (x2 - x1) * (x4 - x3))
                + ((y3 - y4) * (x2 - x1) * (x3)))
                / (((y2 - y1) * (x4 - x3))
                      - ((y4 - y3) * (x2 - x1)));
    Shape2.OutermostVisiblePoint1.y = (((y2 - y1)
          * (Shape2.OutermostVisiblePoint1.x - x1)) / (x2 - x1))
          + y1;
    Shape2.OutermostVisiblePoint1.Visible = true;
    y1 = Shape2.OutermostVisiblePoint1.y;
    x1 = Shape2.OutermostVisiblePoint1.x;
    y2 = Shape2.OutermostVisiblePoint2.y;
    x2 = Shape2.OutermostVisiblePoint2.x;
    VisibleLenght = pow(pow((y2 - y1), 2) + pow((x2 - x1), 2), 0.5);
} else if ((area1 == 1) and (area2 == 0)) {
    y1 = Shape2.OutermostPoint1.y;
    x1 = Shape2.OutermostPoint1.x;
    y2 = Shape2.OutermostPoint2.y;
    x2 = Shape2.OutermostPoint2.x;
    y3 = Camera.Position.y;
    x3 = Camera.Position.x;
    y4 = Shape1.OutermostVisiblePoint1.y;
    x4 = Shape1.OutermostVisiblePoint1.x;
    Shape2.OutermostVisiblePoint2.x =
          (((y2 - y1) * (x4 - x3) * (x1))
                + ((y3 - y1) * (x2 - x1) * (x4 - x3))
                + ((y3 - y4) * (x2 - x1) * (x3)))
                / (((y2 - y1) * (x4 - x3))
                      - ((y4 - y3) * (x2 - x1)));
    Shape2.OutermostVisiblePoint2.y = (((y2 - y1)
          * (Shape2.OutermostVisiblePoint1.x - x1)) / (x2 - x1))
          + y1;
    Shape2.OutermostVisiblePoint2.Visible = true;
    y1 = Shape2.OutermostVisiblePoint1.y;
    x1 = Shape2.OutermostVisiblePoint1.x;
    y2 = Shape2.OutermostVisiblePoint2.y;
    x2 = Shape2.OutermostVisiblePoint2.x;
    VisibleLenght = pow(pow((y2 - y1), 2) + pow((x2 - x1), 2), 0.5);
} else if ((area1 == 2) and (area2 == 0)) {
    y1 = Shape2.OutermostPoint1.y;
    x1 = Shape2.OutermostPoint1.x;
```

```
    y2 = Shape2.OutermostPoint2.y;
    x2 = Shape2.OutermostPoint2.x;
    y3 = Camera.Position.y;
    x3 = Camera.Position.x;
    y4 = Shape1.OutermostVisiblePoint2.y;
    x4 = Shape1.OutermostVisiblePoint2.x;
    Shape2.OutermostVisiblePoint2.x =
          (((y1 - y1) * (x4 - x3) * (x1))
                + ((y3 - y1) * (x2 - x1) * (x4 - x3))
                + ((y3 - y4) * (x2 - x1) * (x3)))
                / (((y2 - y1) * (x4 - x3))
                      - ((y4 - y3) * (x2 - x1)));
    Shape2.OutermostVisiblePoint2.y = (((y2 - y1)
          * (Shape2.OutermostVisiblePoint1.x - x1)) / (x2 - x1))
          + y1;
    Shape2.OutermostVisiblePoint2.Visible = true;
    y1 = Shape2.OutermostVisiblePoint1.y;
    x1 = Shape2.OutermostVisiblePoint1.x;
    y2 = Shape2.OutermostVisiblePoint2.y;
    x2 = Shape2.OutermostVisiblePoint2.x;
    VisibleLenght = pow(pow((y2 - y1), 2) + pow((x2 - x1), 2), 0.5);
} else if ((area1 == 1) and (area2 == 2)) {
    y1 = Shape2.OutermostPoint1.y;
    x1 = Shape2.OutermostPoint1.x;
    y2 = Shape2.OutermostPoint2.y;
    x2 = Shape2.OutermostPoint2.x;
    y3 = Camera.Position.y;
    x3 = Camera.Position.x;
    y4 = Shape1.OutermostVisiblePoint1.y;
    x4 = Shape1.OutermostVisiblePoint1.x;
    Shape2.OutermostVisiblePoint1.x =
          (((y2 - y1) * (x4 - x3) * (x1))
                + ((y3 - y1) * (x2 - x1) * (x4 - x3))
                + ((y3 - y4) * (x2 - x1) * (x3)))
                / (((y2 - y1) * (x4 - x3))
                      - ((y4 - y3) * (x2 - x1)));
    Shape2.OutermostVisiblePoint1.y = (((y2 - y1)
          * (Shape2.OutermostVisiblePoint1.x - x1)) / (x2 - x1))
          + y1;
    Shape2.OutermostVisiblePoint1.Visible = true;
    y1 = Shape2.OutermostPoint1.y;
    x1 = Shape2.OutermostPoint1.x;
```

```
    y2 = Shape2.OutermostPoint2.y;
    x2 = Shape2.OutermostPoint2.x;
    y3 = Camera.Position.y;
    x3 = Camera.Position.x;
    y4 = Shape1.OutermostVisiblePoint2.y;
    x4 = Shape1.OutermostVisiblePoint2.x;
    Shape2.OutermostVisiblePoint2.x =
          (((y2 - y1) * (x4 - x3) * (x1))
                + ((y3 - y1) * (x2 - x1) * (x4 - x3))
                + ((y3 - y4) * (x2 - x1) * (x3)))
                / (((y2 - y1) * (x4 - x3))
                      - ((y4 - y3) * (x2 - x1)));
    Shape2.OutermostVisiblePoint2.y = (((y2 - y1)
          * (Shape2.OutermostVisiblePoint1.x - x1)) / (x2 - x1))
          + y1;
    Shape2.OutermostVisiblePoint2.Visible = true;
    y1 = Shape2.OutermostPoint1.y;
    x1 = Shape2.OutermostPoint1.x;
    y2 = Shape2.OutermostPoint2.y;
    x2 = Shape2.OutermostPoint2.x;
    y3 = Shape2.OutermostVisiblePoint1.y;
    x3 = Shape2.OutermostVisiblePoint1.x;
    y4 = Shape2.OutermostVisiblePoint2.y;
    x4 = Shape2.OutermostVisiblePoint2.x;
    VisibleLenght = pow(((pow((y1 - y3), 2)) + (pow((x1 - x3), 2))),
          0.5)
          + pow(((pow((y2 - y4), 2)) + (pow((x2 - x4), 2))), 0.5);
} else if ((area1 == 2) and (area2 == 1)) {
    y1 = Shape2.OutermostPoint1.y;
    x1 = Shape2.OutermostPoint1.x;
    y2 = Shape2.OutermostPoint2.y;
    x2 = Shape2.OutermostPoint2.x;
    y3 = Camera.Position.y;
    x3 = Camera.Position.x;
    y4 = Shape1.OutermostVisiblePoint1.y;
    x4 = Shape1.OutermostVisiblePoint1.x;
    Shape2.OutermostVisiblePoint2.x =
          (((y2 - y1) * (x4 - x3) * (x1))
                + ((y3 - y1) * (x2 - x1) * (x4 - x3))
                + ((y3 - y4) * (x2 - x1) * (x3)))
                / (((y2 - y1) * (x4 - x3))
                      - ((y4 - y3) * (x2 - x1)));
```

```
    Shape2.OutermostVisiblePoint2.y = (((y2 - y1)
          * (Shape2.OutermostVisiblePoint1.x - x1)) / (x2 - x1))
          + y1;
    Shape2.OutermostVisiblePoint2.Visible = true;
    y1 = Shape2.OutermostPoint1.y;
    x1 = Shape2.OutermostPoint1.x;
    y2 = Shape2.OutermostPoint2.y;
    x2 = Shape2.OutermostPoint2.x;
    y3 = Camera.Position.y;
    x3 = Camera.Position.x;
    y4 = Shape1.OutermostVisiblePoint2.y;
    x4 = Shape1.OutermostVisiblePoint2.x;
    Shape2.OutermostVisiblePoint1.x =
          (((y2 - y1) * (x4 - x3) * (x1))
                + ((y3 - y1) * (x2 - x1) * (x4 - x3))
                + ((y3 - y4) * (x2 - x1) * (x3)))
                / (((y2 - y1) * (x4 - x3))
                      - ((y4 - y3) * (x2 - x1)));
    Shape2.OutermostVisiblePoint1.y = (((y2 - y1)
          * (Shape2.OutermostVisiblePoint1.x - x1)) / (x2 - x1))
          + y1;
    Shape2.OutermostVisiblePoint1.Visible = true;
    y1 = Shape2.OutermostPoint1.y;
    x1 = Shape2.OutermostPoint1.x;
    y2 = Shape2.OutermostPoint2.y;
    x2 = Shape2.OutermostPoint2.x;
    y3 = Shape2.OutermostVisiblePoint1.y;
    x3 = Shape2.OutermostVisiblePoint1.x;
    y4 = Shape2.OutermostVisiblePoint2.y;
    x4 = Shape2.OutermostVisiblePoint2.x;
    VisibleLenght = pow(((pow((y1 - y3), 2)) + (pow((x1 - x3), 2))),
          0.5)
          + pow(((pow((y2 - y4), 2)) + (pow((x2 - x4), 2))), 0.5);
} else if ((area1 == 3) and (area2 == 0)) {
    y1 = Shape2.OutermostPoint1.y;
    x1 = Shape2.OutermostPoint1.x;
    y2 = Shape2.OutermostPoint2.y;
    x2 = Shape2.OutermostPoint2.x;
    y3 = Camera.Position.y;
    x3 = Camera.Position.x;
    y4 = Shape1.OutermostVisiblePoint1.y;
    x4 = Shape1.OutermostVisiblePoint1.x;
```

```
    Shape2.OutermostVisiblePoint2.x =
          (((y2 - y1) * (x4 - x3) * (x1))
                + ((y3 - y1) * (x2 - x1) * (x4 - x3))
                + ((y3 - y4) * (x2 - x1) * (x3)))
                / (((y2 - y1) * (x4 - x3))
                      - ((y4 - y3) * (x2 - x1)));
    Shape2.OutermostVisiblePoint2.y = (((y2 - y1)
          * (Shape2.OutermostVisiblePoint1.x - x1)) / (x2 - x1))
          + y1;
    Shape2.OutermostVisiblePoint2.Visible = true;
    y1 = Shape2.OutermostVisiblePoint1.y;
    x1 = Shape2.OutermostVisiblePoint1.x;
    y2 = Shape2.OutermostVisiblePoint2.y;
    x2 = Shape2.OutermostVisiblePoint2.x;
    VisibleLenght = pow(pow((y2 - y1), 2) + pow((x2 - x1), 2), 0.5);
} else if ((area1 == 0) and (area2 == 3)) {
    y1 = Shape2.OutermostPoint1.y;
    x1 = Shape2.OutermostPoint1.x;
    y2 = Shape2.OutermostPoint2.y;
    x2 = Shape2.OutermostPoint2.x;
    y3 = Camera.Position.y;
    x3 = Camera.Position.x;
    y4 = Shape1.OutermostVisiblePoint1.y;
    x4 = Shape1.OutermostVisiblePoint1.x;
    Shape2.OutermostVisiblePoint1.x =
          (((y2 - y1) * (x4 - x3) * (x1))
                + ((y3 - y1) * (x2 - x1) * (x4 - x3))
                + ((y3 - y4) * (x2 - x1) * (x3)))
                / (((y2 - y1) * (x4 - x3))
                      - ((y4 - y3) * (x2 - x1)));
    Shape2.OutermostVisiblePoint1.y = (((y2 - y1)
          * (Shape2.OutermostVisiblePoint1.x - x1)) / (x2 - x1))
          + y1;
    Shape2.OutermostVisiblePoint1.Visible = true;
    y1 = Shape2.OutermostVisiblePoint1.y;
    x1 = Shape2.OutermostVisiblePoint1.x;
    y2 = Shape2.OutermostVisiblePoint2.y;
    x2 = Shape2.OutermostVisiblePoint2.x;
    VisibleLenght = pow(pow((y2 - y1), 2) + pow((x2 - x1), 2), 0.5);
} else if ((area1 == 0) and (area2 == 4)) {
    y1 = Shape2.OutermostPoint1.y;
    x1 = Shape2.OutermostPoint1.x;
```

```
    y2 = Shape2.OutermostPoint2.y;
    x2 = Shape2.OutermostPoint2.x;
    y3 = Camera.Position.y;
    x3 = Camera.Position.x;
    y4 = Shape1.OutermostVisiblePoint2.y;
    x4 = Shape1.OutermostVisiblePoint2.x;
    Shape2.OutermostVisiblePoint1.x =
          (((y2 - y1) * (x4 - x3) * (x1))
                + ((y3 - y1) * (x2 - x1) * (x4 - x3))
                + ((y3 - y4) * (x2 - x1) * (x3)))
                / (((y2 - y1) * (x4 - x3))
                      - ((y4 - y3) * (x2 - x1)));
    Shape2.OutermostVisiblePoint1.y = (((y2 - y1)
          * (Shape2.OutermostVisiblePoint1.x - x1)) / (x2 - x1))
          + y1;
    Shape2.OutermostVisiblePoint1.Visible = true;
    y1 = Shape2.OutermostVisiblePoint1.y;
    x1 = Shape2.OutermostVisiblePoint1.x;
    y2 = Shape2.OutermostVisiblePoint2.y;
    x2 = Shape2.OutermostVisiblePoint2.x;
    VisibleLenght = pow(pow((y2 - y1), 2) + pow((x2 - x1), 2), 0.5);
} else if ((area1 == 4) and (area2 == 0)) {
    y1 = Shape2.OutermostPoint1.y;
    x1 = Shape2.OutermostPoint1.x;
    y2 = Shape2.OutermostPoint2.y;
    x2 = Shape2.OutermostPoint2.x;
    y3 = Camera.Position.y;
    x3 = Camera.Position.x;
    y4 = Shape1.OutermostVisiblePoint2.y;
    x4 = Shape1.OutermostVisiblePoint2.x;
    Shape2.OutermostVisiblePoint2.x =
          (((y2 - y1) * (x4 - x3) * (x1))
                + ((y3 - y1) * (x2 - x1) * (x4 - x3))
                + ((y3 - y4) * (x2 - x1) * (x3)))
                / (((y2 - y1) * (x4 - x3))
                      - ((y4 - y3) * (x2 - x1)));
    Shape2.OutermostVisiblePoint2.y = (((y2 - y1)
          * (Shape2.OutermostVisiblePoint1.x - x1)) / (x2 - x1))
          + y1;
    Shape2.OutermostVisiblePoint2.Visible = true;
    y1 = Shape2.OutermostVisiblePoint1.y;
    x1 = Shape2.OutermostVisiblePoint1.x;
```

```
            y2 = Shape2.OutermostVisiblePoint2.y;
            x2 = Shape2.OutermostVisiblePoint2.x;
            VisibleLenght = pow(pow((y2 - y1), 2) + pow((x2 - x1), 2), 0.5);
         } else {
            Shape2.OutermostVisiblePoint1.Visible = true;
            Shape2.OutermostVisiblePoint2.Visible = true;
            y1 = Shape2.OutermostVisiblePoint1.y;
            x1 = Shape2.OutermostVisiblePoint1.x;
            y2 = Shape2.OutermostVisiblePoint2.y;
            x2 = Shape2.OutermostVisiblePoint2.x;
            VisibleLenght = pow(pow((y2 - y1), 2) + pow((x2 - x1), 2), 0.5);
         }

         Shape2.Visibility = (VisibleLenght
               / (pow(
                     (pow(
                           (Shape2.OutermostPoint2.y
                                 - Shape2.OutermostPoint1.y), 2)
                           + pow(
                                 (Shape2.OutermostPoint2.x
                                       - Shape2.OutermostPoint1.x),
                                 2)), 0.5))) * 100;
         Shapes.push_back(Shape2);
      }
      Shapes.push_back(Shape1);
   }

   SSize = Shapes.size();
   for (int s = 0; s < SSize; s++) {
      TempSahpe = Shapes.front();
      Shapes.pop_front();
      if (TempSahpe.Visibility > 0)
        Shapes.push_back(TempSahpe);
   }

   return Shapes;

}

std::list<Point> TraCIDemo11p::Determine_Points_Visibility(Camera Camera,
std::list<Shape> Shapes, std::list<Point> Points)
{
```

```
    Shape SS;
    Point PP;
    int PSize=Points.size();
    int SSize=Shapes.size();

    for (int s=0;s<SSize;s++)
    {
       SS=Shapes.front();
       Shapes.pop_front();
       for (int p=0;p<PSize;p++)
       {
          PP=Points.front();
          Points.pop_front();
          if (
                (PP.Visible!=false)
             and (((((PP.y-SS.OutermostPoint1.y)*(Camera.Position.x-
SS.OutermostPoint1.x))-((Camera.Position.y-SS.OutermostPoint1.y)*(PP.x-
SS.OutermostPoint1.x)))*(((SS.OutermostPoint2.y-
SS.OutermostPoint1.y)*(Camera.Position.x-SS.OutermostPoint1.x))-
((Camera.Position.y-SS.OutermostPoint1.y)*(SS.OutermostPoint2.x-
SS.OutermostPoint1.x))))>0)
             and (((((PP.y-SS.OutermostPoint2.y)*(Camera.Position.x-
SS.OutermostPoint2.x))-((Camera.Position.y-SS.OutermostPoint2.y)*(PP.x-
SS.OutermostPoint2.x)))*(((SS.OutermostPoint1.y-
SS.OutermostPoint2.y)*(Camera.Position.x-SS.OutermostPoint2.x))-
((Camera.Position.y-SS.OutermostPoint2.y)*(SS.OutermostPoint1.x-
SS.OutermostPoint2.x))))>0)
             and (((((PP.y-SS.OutermostPoint1.y)*(SS.OutermostPoint2.x-
SS.OutermostPoint1.x))-((SS.OutermostPoint2.y-SS.OutermostPoint1.y)*(PP.x-
SS.OutermostPoint1.x)))*(((Camera.Position.y-
SS.OutermostPoint1.y)*(SS.OutermostPoint2.x-SS.OutermostPoint1.x))-
((SS.OutermostPoint2.y-SS.OutermostPoint1.y)*(Camera.Position.x-
SS.OutermostPoint1.x))))<0)
             )
          {
             PP.Visible=false;
          }
          Points.push_back(PP);

       }
    }
    return Points;
```

```
}

std::list<Shape> TraCIDemo11p::FindShapeList()
{

   std::list<Shape> TempShapeList;
   TempShapeList.clear();

   std::list<std::string> VehicleIds;
   VehicleIds=traci->commandGetCurrentVehicleIds(CurrentVehicleID);
   int VSize=VehicleIds.size();
   Shape VShape;
   double direction,lenght,width;
   std::string VType,VId;
   std::list<Point> OutermostPointsV;

   for (int i=0;i<VSize;i++)
   {
      VId=VehicleIds.front();
      VehicleIds.pop_front();

      direction=traci->commandGetVehicleAngle(VId);
      if (direction<0) direction+=360;
      VType=traci->commandGetVehicleType(VId);
      lenght=traci->commandGetVehicleLenght(VType);
      width=traci->commandGetVehicleWidth(VType);
      VShape=Vehicle_to_Shape(VShape.Position,direction,lenght,width);
      VShape.ID=VId;
      VShape.Position.Visible=true;
      VShape.Position.x=traci->commandGetVehiclePosition(VId).x;
      VShape.Position.y=traci->commandGetVehiclePosition(VId).y;
      OutermostPointsV=Determine_Outermost_Points(Camera1,VShape);
      VShape.OutermostPoint1=OutermostPointsV.front();
      OutermostPointsV.pop_front();
      VShape.OutermostPoint2=OutermostPointsV.front();
      OutermostPointsV.pop_front();
      VShape.OutermostVisiblePoint1=VShape.OutermostPoint1;
      VShape.OutermostVisiblePoint2=VShape.OutermostPoint2;
      VShape.Type=VType;
      VShape.Visibility=100;
```

```
   TempShapeList.push_front(VShape);

}


std::list<std::string> PolygonIds;
PolygonIds=traci->commandGetPolygonIds();
int PSize=PolygonIds.size();
Shape PShape;
std::string PType,PId;
std::list<Point> OutermostPointsP;
std::list<Veins::TraCICoord> Coords;
Point TempPoint;
Veins::TraCICoord TempCoord;

for (int j=0;j<PSize;j++)
{
   PId=PolygonIds.front();
   PolygonIds.pop_front();

   PType=traci->commandGetPolygonTypeId(PId);
   PShape.ID=PId;
   PShape.PointList.clear();
   Coords=traci->commandGetPolygonShape(PId);
   int CSize=Coords.size();
   for (int c=0;c<CSize;c++)
   {
      TempCoord=Coords.front();
      Coords.pop_front();
      TempPoint.x=TempCoord.x;
      TempPoint.y=TempCoord.y;
      TempPoint.Visible=true;
      PShape.PointList.push_front(TempPoint);
   }
   OutermostPointsP=Determine_Outermost_Points(Camera1,PShape);
   PShape.OutermostPoint1=OutermostPointsP.front();
   OutermostPointsP.pop_front();
   PShape.OutermostPoint2=OutermostPointsP.front();
   OutermostPointsP.pop_front();
   PShape.OutermostVisiblePoint1=PShape.OutermostPoint1;
   PShape.OutermostVisiblePoint2=PShape.OutermostPoint2;
   PShape.Type=PType;
```

```
      PShape.Visibility=100;

      TempShapeList.push_front(PShape);

   }
   return TempShapeList;
}

std::list<Shape> TraCIDemo11p::Check_Sahpe_Inside (Camera Camera,
std::list<Shape> Shapes)
{
   int SSize=Shapes.size();
   Shape TempShape;
   for (int s=0;s<SSize;s++)
   {
      TempShape=Shapes.front();
      Shapes.pop_front();

      int PSize=TempShape.PointList.size();
      Point TempPoint;
      bool inside=false;
      for (int p=0;p<PSize;p++)
      {
         TempPoint=TempShape.PointList.front();
         TempShape.PointList.pop_front();
         TempPoint.Visible=Check_Point_Inside(Camera,TempPoint);
         if (TempPoint.Visible) inside=true;
         TempShape.PointList.push_back(TempPoint);
      }
      if (inside)
         {
            TempShape.Visibility=100;
            Shapes.push_back(TempShape);

         }
      else TempShape.Visibility=0;

   }
   return Shapes;
}

std::list<Shape> TraCIDemo11p::FindVisibleShapeList()
```

```
{
   Veins::TraCICoord TempCoord= traci-
>commandGetVehiclePosition(CurrentVehicleID);
   Camera1.Position.x=TempCoord.x;
   Camera1.Position.y=TempCoord.y;

   ShapeList=FindShapeList();
   ShapeList=Check_Sahpe_Inside(Camera1,ShapeList);
   ShapeList=Determine_Shape_Visibility(Camera1,ShapeList);

   return ShapeList;
}

std::list<Shape> TraCIDemo11p::FindVisibleVehicleList()
{

   ShapeList.clear();
   if (CameraEquipment)
   {
      ShapeList=FindVisibleShapeList();
      int SSL=ShapeList.size();
      Shape TempShape;
      std::list<Shape> TempShapeList;
      for (int i=0;i<SSL;i++)
      {
         TempShape=ShapeList.front();
         ShapeList.pop_front();
         if (TempShape.Type=="vtype0")
            TempShapeList.push_back(TempShape);
      }
      ShapeList=TempShapeList;
   }
   return ShapeList;
}

std::list<TrackingShape> TraCIDemo11p::FindTrackingVehicleList()
{
   std::list<TrackingShape> TempTrackingVehicleList;
   TempTrackingVehicleList.clear();
   int s1=VisibleShapes1.size();
   int s2=VisibleShapes2.size();
   std::list<Shape> TempVisibleShapes2=VisibleShapes2;
```

```
    double Y,X;
    Shape Shape1,Shape2;
    TrackingShape Shape3;

    for (int i=0; i<s1;i++)
    {
       Shape1=VisibleShapes1.front();
       VisibleShapes1.pop_front();
       for (int j=0;j<s2;j++)
       {
          Shape2=VisibleShapes2.front();
          VisibleShapes2.pop_front();
          if (Shape1.ID==Shape2.ID)
          {
             Y=Shape2.Position.y-Shape1.Position.y;
             X=Shape2.Position.x-Shape1.Position.x;
             Shape3.ID=Shape1.ID;
             if ((Y>0)and(X>0))  Shape3.Direction=((atan(Y/X)*180)/PI);
             else if ((Y>0)and(X<0))  Shape3.Direction=((atan(Y/X)*180)/PI)+180;
             else if ((Y>0)and(X==0))  Shape3.Direction=90;
             else if ((Y<0)and(X>0))  Shape3.Direction=((atan(Y/X)*180)/PI)+360;
             else if ((Y<0)and(X<0))  Shape3.Direction=((atan(Y/X)*180)/PI)+180;
             else if ((Y<0)and(X==0))  Shape3.Direction=270;
             else if ((Y==0)and(X<0))  Shape3.Direction=180;
             else if ((Y==0)and(X>0))  Shape3.Direction=0;
             else if ((Y==0)and(X==0))  Shape3.Direction=0;
             Shape3.Position=Shape2.Position;
             Shape3.Speed=(pow((pow(Y,2)+pow(X,2)),0.5))/TrackingTime;
             TempTrackingVehicleList.push_front(Shape3);
          }
       }
       VisibleShapes2=TempVisibleShapes2;
    }
    return TempTrackingVehicleList;

}
```

Curriculum Vitae

Name: Besat Zardosht

Post-secondary Education and Degrees:

Shahid Beheshti University
Tehran, Iran
2003-2007 B.A.

Amirkabir University of Technology
Tehran, Iran
2007-2009 M.A.

The University of Western Ontario
London, Ontario, Canada
2010-2016 Ph.D.

Related Work Experience

Software Developer
Binnj/Breeze Inc
2015-2016

Graduate Teaching Assistant
University of Western Ontario
2010-2015

Graduate Research Assistant
University of Western Ontario
2010-2015

Software Developer
London Employment Help Centre
2012-2015

Software Developer
Granir Inc
2007-2009

Publications:

Besat Zardosht, Steven Beauchemin and Michael Bauer, “An In-Vehicle Tracking Method Using Vehicular Ad-Hoc Networks with a Vision-Based System”. *IEEE International Conference On Systems, Man, And Cybernetics(SMC'14), 2014*

Besat Zardosht, Steven Beauchemin and Michael Bauer, “An Emergency Message Propagation System Using Roadside Units and Vehicle-To-Vehicle Communication”. *29th International Conference on Computer Applications in Industry and Engineering (CAINE 2016)*

Besat Zardosht, Steven Beauchemin and Michael Bauer, “A Decision Making Module for Cooperative Collision Warning System Using Vehicular Ad-Hoc Network”. *The 16th International IEEE Annual Conference on Intelligent Transportation Systems, 2013*

Besat Zardosht and Ahmad Abdollahzadeh Barforoush, “AUT-BLEU: Extending BLUE with use of Pars Tree”. *The 14th Machine Translation Summit, 2013*

Besat Zardosht and Ahmad Abdollahzadeh Barforoush, “A New Machine Translation Evaluation System with new approach in weighting the N-grams, Based on the Words Part of Speech”. *The 15th National CSI Computer Conference, 2010 (This paper is written in Persian)*