12-2016

Connected Vehicles at Signalized Intersections: Traffic Signal Timing Estimation and Optimization

S. AliReza Fayazi

Connected Vehicles at Signalized Intersections: Traffic Signal Timing Estimation and Optimization

A Doctoral Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Mechanical Engineering

by
S. AliReza Fayazi
December 2016

Accepted by:
Dr. Ardalan Vahidi, Committee Chair
Dr. Joshua Summers
Dr. Mohammed Daqaq
Dr. Paul Venhovens
Dr. Andre Luckow

Abstract

Summary: While traffic signals ensure safety of conflicting movements at intersections, they also cause much delay, wasted fuel, and tailpipe emissions. Frequent stops and goes induced by a series of traffic lights often frustrates passengers. However, the connectivity provided by connected vehicles applications can improve this situation. A uni-directional traffic signal to vehicle communication can be used to guide the connected vehicles to arrive at green which increases their energy efficiency; and in the first part of the dissertation, we propose a traffic signal phase and timing estimator as a complementary solution in situations where timing information is not available directly from traffic signals or a city's Traffic Management Center. Another approach for improving the intersection flow is optimizing the timing of traditional traffic signals informed by uni-directional communication from connected vehicles. Nevertheless, one can expect further increase in energy efficiency and intersection flow with bi-directional vehicle-signal communication where signals adjust their timings and vehicles their speeds. Autonomous vehicles can further benefit from traffic signal information because they not only process the incoming information rather effortlessly but also can precisely control their speed and arrival time at a green light. The situation can get even better with 100% penetration of autonomous vehicles since a physical traffic light is not needed anymore. However, the optimal scheduling of the autonomous vehicle arrivals at such intersections remains an open problem. The second part of the dissertation attempts to address the scheduling problem formulation and to show its benefits in microsimulation as well as experiments.

Intellectual Merit: In the first part of this research, we study the statistical patterns hidden in the connected vehicle historical data stream in order to estimate a signal's phase and timing (SPaT). The estimated SPaT data communicated in real-time to connected vehicles can help drivers plan over time the best vehicle velocity profile and route of travel. We use low-frequency probe data streams to show what the minimum achievable is in estimating SPaT. We use a public feed of bus location and velocity data in the city of San Francisco as an example data source. We show it is possible to estimate, fairly accurately, cycle times and duration of reds for pre-timed traffic lights traversed by buses using a few days worth of aggregated bus data. Furthermore, we also estimate the start of greens in real-time by monitoring movement of buses across intersections. The results are encouraging, given that each bus sends an update only sporadically (≈ every 200 meters) and that bus passages are infrequent (every 5-10 minutes). The accuracy of the SPaT estimations are ensured even in presence of queues; this is achieved by extending our algorithms to include the influence of queue delay. A connected vehicle test bed is implemented in collaboration with industry. Our estimated SPaT information is communicated uni-directionally to a connected test vehicle for those traffic signals which are not connected.

In the second part of the dissertation, another test bed, but with bi-directional communication capability, is implemented to transfer the connected vehicle data to an intelligent intersection controller through cellular network. We propose a novel intersection control scheme at the cyber layer to encourage platoon formation and facilitate uninterrupted intersection passage. The proposed algorithm is presented for an all autonomous vehicle environment at an intersection with no traffic lights. Our three key contributions are in communication, control, and experimental evaluation: i) a scalable mechanism allowing a large number of vehicles to subscribe to the intersection controller, ii) reducing the vehicle-intersection coordination problem to a Mixed Integer Linear Program (MILP), and iii) a Vehicle-in-the-Loop (VIL) test bed with a real vehicle interacting with the intersection control cyber-layer and with our customized microsimulations in a virtual road network environment. The proposed MILP-based controller receives information such as location and speed from each subscribing vehicle and advises vehicles of the optimal time to access the intersection. The access times are computed by periodically solving a MILP with the objective of minimizing intersection delay, while ensuring intersection safety and considering each vehicle's desired velocity. In order to estimate the fuel consumption reduction potential of the implemented system, a new method is proposed for estimating fuel consumption using the basic engine diagnostic information of the vehicle-in-the-loop car.

Broader Impacts: This research can transform not only the way we drive our vehicles at signalized intersections but also the way intersections are managed. As we evaluated in a connected test vehicle in the first part of the dissertation, our SPaT estimations in conjunction with the SPaT information available directly from Traffic Management Centers, enables the drivers to plan over time the best vehicle velocity profile to reduce idling at red lights. Other fuel efficiency and safety functionalities in connected vehicles can also benefit from such information about traffic signals' phase and timing. For example, advanced engine management strategies can shut down the engine in anticipation of a long idling interval at red, and intersection collision avoidance and active safety systems could foresee potential signal violations at signalized intersections. In addition, as shown in the second part of the dissertation, when a connected traffic signal or intersection controller is available, intelligent control methods can plan in real-time the best timings and the lengths of signal phases in response to prevailing traffic conditions with the use of connected vehicle data. Our MILP-based intersection control is proposed for an all autonomous driving environment; and right now, it can be utilized in smart city projects where only autonomous vehicles are allowed to travel. This is expected to transform driving experience in the sense that our linear formulations minimizes the intersection delay and number of stops significantly compared to pre-timed intersections.

Acknowledgments

The first part of this dissertation was supported by BMW Group Technology Office USA, Mountain View, CA. It was supported in part by the National Science Foundation under Grant CMMI-0928533. The second part was sponsored by a research award from BMW Information Technology Research Center (ITRC) in Greenville, SC, USA. I am thankful for the support provided by Mr. Andre Luckow and Mr. Hans-Peter Fischer from BMW ITRC, and Mr. Andreas Winckler from BMW Group Technology Office, USA in Mountain View, California.

I own an enormous debt of gratitude to my advisor, Prof. Ardalan Vahidi. His creative insight and experience was invaluable. He spent long hours on this research especially during his sabbatical year that we were both working at BMW Technology Office USA in Mountain View, CA and University of California, Berkeley. I also thank other member of our research group, Mr. Grant Mahler from Clemson University who is currently with BMW Technology Office USA in Mountain View.

Several other researchers at University of California, Berkeley also provided support and helpful suggestions to the research presented in this dissertation. I wish to express my gratitude to Prof. Pravin Varaiya, Prof. Roberto Horowitz, Dr. Anastasios Kouvelas, and Dr. Gabriel Gomes who introduced me to the interesting research area of traffic signal control. I benefited from their knowledge and expertise during our regular meetings and discussions at University of California, Berkeley.

Publications

Journal Papers

- S. A. Fayazi, A. Vahidi, G. Mahler, and A. Winckler, "Traffic signal phase and timing estimation from low-frequency transit bus data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, pp. 1928, Feb 2015.

- S. A. Fayazi, and A. Vahidi, "Crowd-sourcing Phase and Timing of Pre-Timed Traffic Signals in Presence of Queues: Algorithms and Back-end System Architecture," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 3, pp. 870-881, March 2016.

- A. Kouvelas, J. Lioris, S. A. Fayazi, and P. Varaiya, "Maximum pressure controller for stabilizing queues in signalized arterial networks," *Transportation Research Record: Journal of the Transportation Research Board*, 2421(1), p. 133-141, 2014.

- Y. Parvini, A. Vahidi, and S. A. Fayazi, "Heuristic versus Optimal Charging of Super-capacitors, Lithium-Ion, and Lead-Acid Batteries:An Efficiency Point of View," Under Review *IEEE Transactions on Control Systems*.

Patents

- A. Vahidi, S. A. Fayazi, G. Mahler, and A. Winckler, "Systems and Methods for Estimating Traffic Signal Information," US Patent number. US9183743 B2, Publication Date Nov. 2015.

Conference and Poster Session

- S. A. Fayazi, A. Vahidi, and A. Luckow "Optimal Scheduling of Autonomous Vehicle Arrivals at Intelligent Intersections via MILP," Under Review *in American Control Conference (ACC)*, 2017.

- A. Kouvelas, J. Lioris, S. A. Fayazi, and P. Varaiya, "Max-pressure controller for stabilizing the queues in signalized arterial networks," *Transportation Research Board 93rd Annual Meeting*, No. 14-5440. 2014.

- S. A. Fayazi, "A MILP-based Vehicle-Intersection Coordination under the Autonomous Vehicle Environment," *ME Graduate Student Poster Session and Conference*, Mechanical Engineering Department, Clemson University, Nov. 2016.

- S. A. Fayazi, and N. Wan, "Arterial Traffic Estimation Using Vehicular Probe Data," *Top 3 Presentation in Third ME Graduate Student Poster Session and Conference*, Mechanical Engineering Department, Clemson University, Jan. 2015.

- S. A. Fayazi, N. Wan, S. Lucich, A. Vahidi, and G. Mocko, "Optimal pacing in a cycling time-trial considering cyclists fatigue dynamics," *in American Control Conference (ACC)*, 2013, pp. 6442 6447.

- N. Wan, S. A. Fayazi, H. Saeidi, and A. Vahidi, "Optimal Power Management of an Electric Bicycle based on Terrain Preview and Considering Human Fatigue Dynamics," *in American Control Conference (ACC)*, 2014, pp. 3462 3467.

Technical Reports

- S. A. Fayazi, and A. Vahidi, , "Back-End Architecture of the Crowdsourcing System," A technical report submitted to BMW Group Technology Office USA, Mountain View, CA, Submission date Sep. 2013.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Research Motivation and Background

A connected vehicle environment, as shown in Figure 1.1, enables the vehicles equipped with computing and wireless communication devices to ”talk” not only to each other but also to surrounding infrastructure. The vehicle-to-infrastructure (V2I) communication capability can be bidirectional through communication technologies such as short-range radios or cellular networks. In one direction of communication, the vehicle information communicated can provide transportation agencies with traffic data to make roads less congested; alternatively, intelligent traffic signals can use data coming from connected vehicles to optimize their signal phase and timing. In the other direction of communication, the infrastructure information communicated to connected vehicles can help drivers manage the time, speed, and route of their travel to reduce travel delay [1]. In addition, V2I can improve safety by informing travelers of dangerous situations especially while approaching signalized intersections.

In 2013, there were 32,719 fatalities and 2.31 million injured people involved in 5.7 million total number of crashes, according to U.S. DOT’s National Highway Traffic Safety Administration or NHTSA [2]. Furthermore, the cost of traffic congestion for U.S. economy is estimated more than $120 billion in 2012 [3]. Because of these facts, the U.S. DOT’s Intelligent Transportation Systems Joint Program Office (ITS JPO) [4] has focused on connected vehicle research to leverage the capabilities of wireless technology to reduce or eliminate crashes and achieve mobility benefits. In conjunction with automotive industry, test beds have been developed to see how vehicles, roadside infrastructure, and back-end systems work together. While the test environments for connected vehicle research can be interstate roadways, arterials, and signalized/

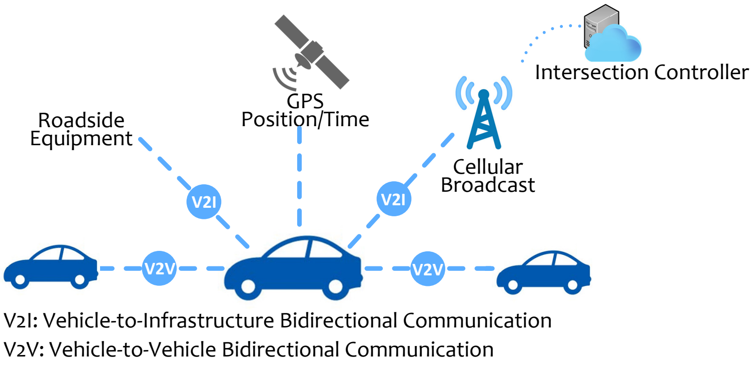

Intersection Controller
GPS
Position/Time
Roadside
Equipment
Cellular
Broadcast
V2I
V2I
V2V
V2V
V2I: Vehicle-to-Infrastructure Bidirectional Communication
V2V: Vehicle-to-Vehicle Bidirectional Communication

Figure 1.1: Possible communication modes of a GPS-enabled connected vehicle including: a V2V local broadcast, a short-range V2I communication to roadside equipment, and a cellular infrastructure based V2I communication.

unsignalized intersections or a combination of them, the research proposed in this dissertation considers the connected vehicles at signalized intersections only.

Traffic signals have been an indispensable element of our transportation networks since their inception, and are not likely to change form or function in the foreseeable future [5]. While traffic signals ensure safety of conflicting movements at intersections, they also cause much delay, wasted fuel, and tailpipe emissions. Frequent stops and goes induced by a series of traffic lights often frustrates drivers. However, the connectivity provided by connected vehicle applications can improve this situation. This can be accomplished by real-time transmission of the status of vehicles for use by intelligent traffic signals (intersection controllers) and/or by providing the status of traffic lights (Signal Phase and Timing or SPaT) to in-vehicle computing devices. The former is the subject of the second part of this dissertation and the latter is the subject of the first part of this dissertation.

1.1.1 Signal Phase and Timing Estimation in Connected Vehicle Environment

As described previously in this chapter and as shown in Figure 1.2, in one direction of V2I communication, the SPaT information can be communicated to the connected vehicles, tailored to meet the needs of the in-vehicle driver assistance systems. However, in arterial driving, the switching pattern of traffic signals are complex and unknown. This makes accurate travel time estimation or optimal routing often impossible even with modern traffic-aware in-vehicle navigation systems. Much of these difficulties arise due to the lack of information about the current and future state of traffic signals. In an ideal situation where the state of a light's timing and phasing is known, the speed could be adjusted for a timely arrival at green (Velocity Advisory System) [6]. One can expect considerable fuel savings in city driving with such predictive cruise

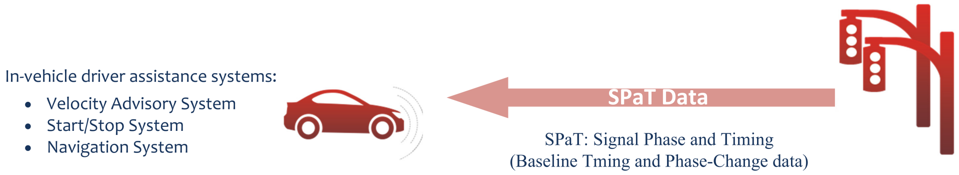

In-vehicle driver assistance systems:
Velocity Advisory System
Start/Stop System
Navigation System
SPaT Data
SPaT: Signal Phase and Timing
(Baseline Tming and Phase-Change data)

Figure 1.2: Communicating the status of traffic lights (SPaT) to connected vehicles, to be used by in-vehicle driver assistance systems.

control algorithms as shown in [6] and [7]. When idling at red becomes unavoidable, knowledge of remaining red time can determine if an engine shut-down is worthwhile (Start/Stop System). A collision warning system can benefit from the light timing information and warn against potential signal violations [8]. Future navigation system that have access to the timing plan of traffic lights, can find arterial routes with less idling delay [9] and can also provide more accurate estimates of trip time.

The main technical challenge to deploying such in-vehicle functionalities is in reliable estimation and prediction of Signal Phase And Timing (SPAT): Uncertainties arising from clock drift of fixed-time signals, various timing plan of actuated traffic signals, and traffic queues render this a challenging and open-ended problem. Direct access to signal timing plans and real-time state of the light is prohibitively difficult due to hundreds of local and federal entities that manage the more than 330,000 traffic lights across the United States alone [10]. Even when such access is granted, much effort and time must be spent in structuring information from various municipalities in standard and uniform formats. The more recent emphasis on Dedicated Short Range Communication (DSRC) technology for communicating the state of traffic signals to nearby vehicles has safety benefits, but requires heavy infrastructure investments and even then is limited by its short communication range.

To overcome some of these difficulties, in the first part of this dissertation we propose an alternative approach, that relies on vehicle probe data streams, for estimating a signal's phase and timing. In recent years several research groups have shown that mobile phone or vehicle probe data can be effectively utilized for estimation of traffic flow [11, 12, 13]. Today many traffic information providers, such as Google, INRIX, and Waze use data from vehicle and cellular phone probes, as well as other means, to estimate the severity of traffic on highways nearly in real-time. However such algorithms perform relatively poorly in arterial networks because traffic signals induce complex queue and stop and go dynamics. Some more recent work has focused on estimating queue lengths [14] and on determining location of traffic signals and stop signs [13] through use of vehicle probe data. What seems to be missing from the literature is a systematic attempt to derive SPAT information from available vehicle data streams.

Unfortunately, currently one cannot expect high update rates from public fleets that broadcast their information, nor is there a proliferation of vehicle probes. Most existing ones only provide event-based updates, for example at a time of a crash or air-bag deployment. Interesting data sources such as San Francisco taxi cab data available through the cab-spotting program [15] have update rates of only once per minute. More frequent updates are available through NextBus, a service that provides a real-time XML feed of GPS time stamp, position, velocity, and several other attributes of transit buses of a few cities in North America [16]. Some instances of this feed, such as San Francisco MUNI stream, have update rates on the order of twice per minute. And one can be certain that intersections along a bus route get traversed by a bus every few minutes during the day. An open question that we try to address in the first part of this dissertation is how much, statistical patterns in such low-frequency data can reveal about the state and parameters of traffic lights. This determines what the minimum achievable is; as higher frequency probe data becomes available in the future, more accurate estimates of parameters of traffic signals can be obtained.

In case of a low-frequency data source, less challenge is expected in the estimation procedure if we only use the probe data that appears to be less influenced by heavy traffic and delay in queue. However, this eliminates a large portion of data; and any SPaT estimation method that filters out huge amount of data is subject to error. For this reason, a queue dissipation formulation is required so that the influence of queue delay is considered in SPaT estimations.

1.1.2 Arterial Traffic Signal Optimization with Connected Vehicles

Investing in the belief that harmony between connected cars and traffic control infrastructure can revolutionize the traffic scene in smart cities, we expand the idea given in previous subsection to include two-way communication between moving connected vehicles and upcoming traffic lights. This is the subject of the second part of this dissertation and enables the traffic signals to adapt their timing as well as the approaching connected vehicles adapts their velocity for a timely arrival at signalized intersections. Adaptive signal control systems are able to adjust their timings and the lengths of signal phases in real-time in response to prevailing traffic conditions. The input data is currently from traditional detectors (loop detectors and video detection) but it has the potential to be provided within a connected vehicle environment. This environment, as previously mentioned in this chapter and as shown in Figure 1.3, allows the geographical data (positions, headings, and speeds) of connected vehicles to be wirelessly transmitted in real-time to traffic signal controllers [17], whereas traditional detectors mostly rely on point detection which provide a very limited information such as passage of a vehicle at a fixed location [18, 19].

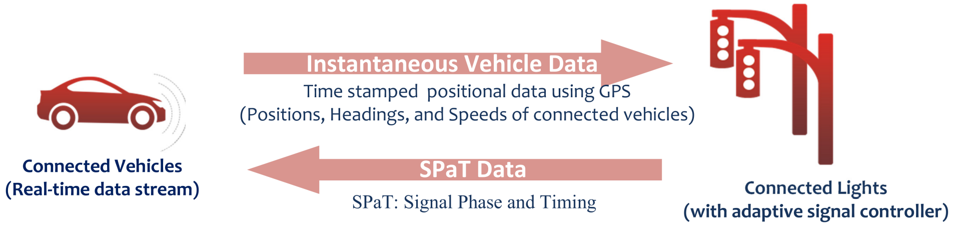

Instantaneous Vehicle Data
Time stamped positional data using GPS
(Positions, Headings, and Speeds of connected vehicles)
SPaT Data
SPaT: Signal Phase and Timing
Connected Vehicles
(Real-time data stream)
Connected Lights
(with adaptive signal controller)

Figure 1.3: Two way communication for communicating the probe data of connected vehicles to adaptive traffic signal controllers, for signal phase and timing optimization.

Autonomous vehicles can further benefit from traffic signal information because they not only process the incoming information rather effortlessly but also can precisely control their speed and arrival time at a green light. The situation can get even better with 100% penetration of autonomous vehicles since a physical traffic light is not needed anymore as shown in concept papers by [20, 21, 22]. Also because autonomous cars have much faster reaction times than human driven vehicles, the intersection controller can rapidly switch between phases [23].

However, the major challenges in this research are first to formulate an intelligent intersection control that is responsive to prevailing traffic conditions and second to create a versatile live testbed of intersection controllers that more intelligently sense and route traffic and communicate to vehicles with cellular connectivity. A real-time simulation testbed is safe, reproducible and resources-saving [24]; and this makes the simulation option suitable for analysis on the performance of vehicles connectivity, platooning, and vehicle-intersection coordination. Furthermore, a simulation testbed is inexpensive, does not need traffic lights and test drivers, and makes the data logging much easier comparing to experimental testbeds. On the other hand, by utilizing real test vehicles in an experimental testbed not only the fuel consumption can be measured more accurately but also the real dynamics of a vehicle can be analyzed. Combining the advantages of the two aforementioned testbeds, a Vehicle-In-Loop (VIL) configuration seems a preferable strategy in early stages of a development where cost and safety are a priority. A Vehicle-In-Loop (VIL) configuration needs a microscopic traffic simulation model, one or more real vehicles, and a communication between the intersection controller and all the vehicles (simulated and real vehicles). The microsimulation needs to be real-time because of the real vehicles in the loop. The real vehicles needs to be equipped with a geographical positional data logger as well as a fuel consumption tracker. And the simulated vehicles have to replicate the same communication protocol as the real vehicles use to interact with the intersection controller. These requirements need to be considered in VIL design.

1.2 Research Contributions

If traffic signals of a city are connected to Traffic Management Center (TMC), then access to the real time state of the traffic lights may be granted either directly from local and federal entities, or indirectly through third party data providers. Nevertheless, what we are proposing in the first part of this dissertation as a crowdsourced-based SPaT estimator is a complementary solution in situations where timing information is not available directly from a city's Traffic Management Center.

Our proposed solution obtains deterministic knowledge of SPaT information of pre-timed signals using only low-frequency probe vehicle data. The input probe data can be gathered from connected vehicles of any kind reporting at least their GPS coordinate, and velocity at a timestamp, although this research uses the public feed of the San Francisco's GPS-enabled buses. In [25, 26], we have shown that it is possible to estimate, fairly accurately, cycle times and duration of reds and greens for pre-timed traffic lights traversed by buses using a few days worth of aggregated bus data. Furthermore, we also estimate the green-initiations (start of greens) in real-time by monitoring movement of buses across intersections. The results are encouraging, given that each bus sends an update only sporadically ($\approx$ every 200 meters) and that bus passages are infrequent (every 5-10 minutes).

The obtained SPaT information is ultimately fed into an in-vehicle computing device. The implemented system, as shown in Figure 1.4, is actually capable of receiving SPaT data directly from Traffic Management Centers as well as from a crowdsourcing server. The system architecture of transferring a TMC's data into an in-vehicle system was verified by a previous student of our group. As mentioned previously, the main contribution of this dissertation is a crowdsourced-based SPaT estimator.

As shown in Figure 1.4, the input probe data is collected by the crowdsourcing server. The data is then recorded in a database so that the same server can access it to estimate a collection of traffic signal phase and timing information (SPaT). The traffic signal information of each phase of each intersection is then accessed by a cloud-based server, specially configured for the infrastructure-to-vehicle (I2V) communications via wireless cellular networks, such as 4G/LTE. In fact, the cloud-based server allows for fluctuations in number of connected vehicles requesting SPaT information. A Web Server is also set up for initialization, maintenance and ground-truth verification purposes.

In [27], we have extended our SPaT estimation methodologies to include the influence of queue delay. In this extension, it is statistically shown that a considerable number of vehicle passes occur in heavy traffic and excluding them will negatively influence accuracy of SPaT estimation algorithms. We have pro-

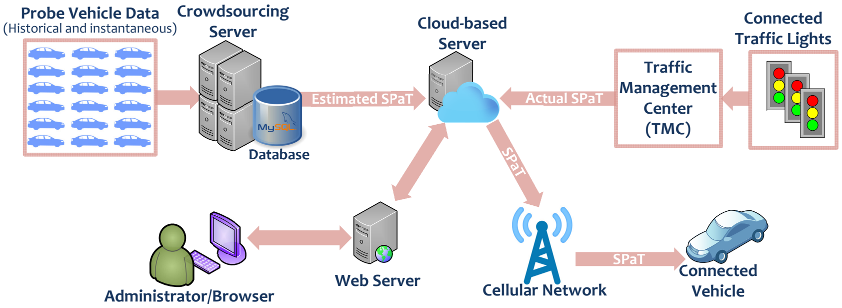


Figure 1.4: The overall hardware architecture of the implemented back-end system to provide the status of traffic lights to in-vehicle systems.

posed a queue dissipation formulation which is compared and found to be consistent to other formulations used in related works [28], [29].

In the second part of this dissertation, we add a bidirectional vehicle-to-infrastructure (V2I) communication capability so that the moving connected vehicles can wirelessly transmit their geographical data in real-time to the upcoming connected intersection controllers. As shown in Figure 1.5, we propose an intelligent intersection control algorithm that processes autonomous vehicle instantaneous data, create a live picture of evolving traffic conditions, and optimally schedule vehicles arrivals and adjusts traffic signal status. As the proposed algorithm is presented for an autonomous driving environment (100% penetration rate of equipped vehicles) no physical traffic signal is needed. The intersection controller of Figure 1.5 is not considered for central coordination of many intersections. This controller uses only the data around a certain area of the intersection; and as a result, is considered easier to implement and more cost-effective [30].

Specifically, we will create three key components in communication, control, and experimental evaluation: i) a scalable mechanism allowing a large number of vehicles to subscribe to the intersection controller, ii) a vehicle-intersection coordination scheme to anticipate vehicle arrivals and guiding them into fast moving platoons; we successfully reduce this coordination problem to a Mixed Integer Linear Program (MILP), and iii) a Vehicle-in-the-Loop (VIL) test bed with a real vehicle interacting with the intersection control cyber-layer and with our customized microsimulations in a virtual road network environment. The proposed MILP-based controller receives information such as location and speed from each subscribing vehicle and advises vehicles of the optimal time to access the intersection. The access times are computed by periodically solving a MILP with the objective of minimizing intersection delay, while ensuring intersection safety and considering each vehicle's desired velocity.

In order to demonstrate the feasibility of communicating connected vehicle instantaneous data through cellular networks communication technologies and also to evaluate the efficiency of our proposed intelligent intersection control, a testbed is set up as shown in Figure 1.6. Our evaluation is conducted at an isolated test area in a Vehicle-In-Loop (VIL) simulation framework, involving microscopic traffic simulation, wireless cellular communication, and fuel estimation model. The simulation environment, as shown in Figure 1.6, is set up in such a way that not only it is real-time and replicates real vehicles movements in arterial corridors but also communicates to the intersection controller same way the real connected vehicles do. The real vehicles and the simulated ones interact with each other at a virtual intersection.

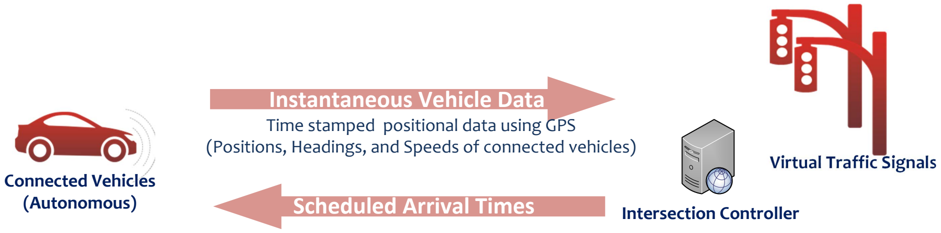

Instantaneous Vehicle Data
Time stamped positional data using GPS
(Positions, Headings, and Speeds of connected vehicles)
Connected Vehicles
(Autonomous)
Scheduled Arrival Times
Intersection Controller
Virtual Traffic Signals

Figure 1.5: Vehicle-intersection coordination under the autonomous vehicle environment.

As no autonomous test vehicle is available at this stage of the work, a location-based in-vehicle virtual driver assistant is needed to guide the driver of the connected test vehicle for a timely arrival at intersection. Furthermore, in order to estimate the fuel consumption of our test vehicle another in-vehicle application is sought which can access the On-board diagnostics (OBD) port of the vehicle. However, an in-vehicle implementation of these systems requires a close partnership with car makers or OEMs that is not always possible. As a result, applications running on a smart-phone is a reasonable alternative. We have implemented the aforementioned applications on iPhone iOS device. A new fuel estimation method is also proposed and incorporated into the iOS application.

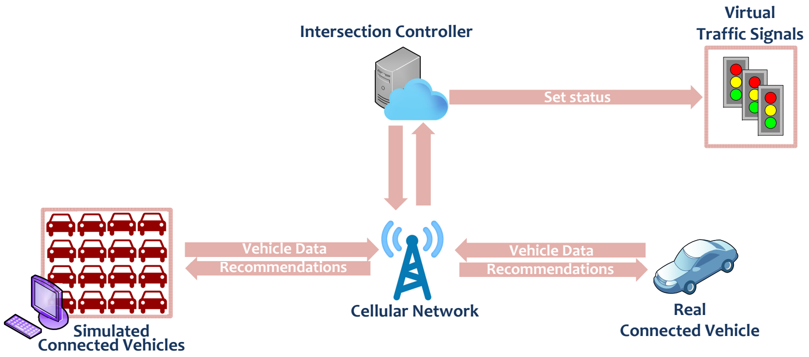


Figure 1.6: The overall hardware architecture of the implemented back-end system to provide a two-way communication between smart intersection controllers and connected vehicles.

1.3 Thesis Outline

In the first part of this dissertation, signal phase and timing estimation methods are developed for traffic lights in a connected vehicle environment. The goal is to communicate these estimations to connected vehicles for a timely arrival at intersections even if the traffic light is not connected and its real-time timing and phasing is unknown. The outline of the first part of this dissertation which includes chapters 2-5 is as follows: First in Chapter 2, the proposed methodology for estimation and prediction of Signal Phase and Timing (SPaT) information from low-frequency probe data is presented for pre-timed traffic signals. The estimated information consists of two parts: First is traffic signal baseline timing that includes cycle time, phase lengths (red and green intervals), and signal offset changes. Second is phase-change (sync) data, that is green-initiation or start-of-green. The probe data influenced by the heavy traffic and the delay in queues are investigated later in Chapter 3 in such a way that accuracy of the SPaT estimations is ensured even in presence of queues. It will be shown in this chapter that by using our proposed queue dissipation formulations, more accurate phase-change predictions are achieved comparing with the results of Chapter 2. The back-end of the whole implemented system is presented in Chapter 4. This includes the hardware architecture of the system as well as the software techniques of processing the incoming crowdsourced data, estimating SpaT information, and delivering the SPaT information to a connected test vehicle through I2V communications. In Chapter 5, a summary of the research findings and the potential future works are presented.

The second part of the dissertation is organized as follows: First, Chapter 6 introduces the vehicle-intersection coordination problem in simple words; and our proposed formulation is presented followed by conversion to a simple linear constrained optimization problem (mixed-integer linear program or MILP). In Chapter 7, we explain our evaluation approaches via microsimulations and with real vehicles interacting with the intersection control cyber-layer and a virtual road network environment. The evaluation results are given in Chapter 9; and the fuel consumption reduction potential of the proposed intersection control is also reported. Our proposed method to estimate the real vehicle's fuel consumption rate through vehicle diagnostic data is explained in Chapter 8.

Part I

Signal Phase and Timing Estimation in Connected Vehicle Environment

Chapter 2

Crowdsourcing Phase and Timing of Pre-Timed Traffic Signals from Low-Frequency Vehicle Probe Information

2.1 Introduction

In order to achieve a complete solution to providing individualized SPaT information directly to vehicles, alternative approaches are needed in situations where the phase and timing information is not available from a city's Traffic Management Center (TMC) or directly from infrastructure at intersections. In this chapter we propose an alternative approach, that relies on vehicle probe data streams, for estimating a pre-timed signal's phase and timing.

In recent years, an increasing interest in obtaining SPaT information is obvious in the related publications in the literature but most of them rely on other sources of data. A traffic signal predictor is also proposed in [31] but needs to be trained by a traffic controller's data log; it is obvious that the raw data would be logged only for few traffic controllers and if available, it is prohibitively difficult to access that data. Using the cameras of windshield-mounted mobile phones, Koukoumidis et al. in [7] present a speed advisory software service that detects the current phase of signals; however, the mobile nodes need a database from TMC for the signal settings of fixed-time traffic signals or the Support Vector Regression (SVR) prediction models for actuated signals. In [32] the probability of a light being green is found over a planning horizon but by assuming that the baseline timings and schedules are already available. A Virtual Trip Line (VTL) technology based method is also conducted by Hao et al. in [33] to estimate the signal timing parameters.

To the best of authors' knowledge, to date, three works by Kerper et al. [34], Cheng et al. [35] and Chuang et al. [36] are deterministic approaches related to our proposed approach, although, those require high frequency probe data sets. The simulation results in [34, 35] are based on the assumption that the penetration level is high and the full velocity profiles of the vehicles are available through high-frequency probe data. The only real dataset is a dataset used by Cheng et al. including detailed vehicle trajectory recorded every one-tenth of a second [37]; and [34] assumes that frequency of data update is $\approx$1Hz. The SPaT estimation methodology presented by Cheng et al. assumes that the acceleration rate of the vehicles are also reported which is not available through regular tracking devices; furthermore, the start of greens and start of reds are the only parameters of the signalized intersection that are estimated by them. Chuang et al. [36] use smartphones installed in vehicles to collect velocity profiles at a sampling rate of 1 Hz. They have implemented the crowdsourcing part directly on smartphones which decreases the number of reporting events sent from smartphones thorugh Internet connectivity; however, the implications on smartphones' battery usage (which is not addressed by the authors) remains questionable.

Furthermore, the studies by Kerper et al. [34], Cheng et al. [35], and Chuang et al. [36] are only applicable to signals that their timings are fixed during the day. Nevertheless, the SPaT estimation proposed in this paper also operate on pre-timed signals control that may have different signal timings for segments of the day such as AM-peak, PM-peak, and off-peak.

In addition to the aforementioned deterministic approaches, in [38], Zhu et al. use the maximum a posteriori (MAP) estimation and a joint optimization algorithm to estimate the state of a traffic light, but the system's capability in estimating the next phase-change of a signal (which is needed for most in-vehicle applications) is not addressed by the authors yet.

Our proposed solution in this chapter works fairly accurately with low-frequency vehicular probe data streams. The accuracy is experimentally evaluated for a selection of pre-timed traffic lights in the city of San Francisco, CA by utilizing a real-time data feed of San Francisco's public buses as an example data source. In this chapter, after a short description of the utilized data stream in Section 2.2, we explain reconstruction of the approximate trajectory of a probe vehicle between each two update points in Sections 2.3 and

2.4. Section 2.5 presents our methodology and results for estimation of red time, and cycle time of a traffic signal based on available and reconstructed probe vehicle data. We also discuss the potential for extracting other attributes such as an estimate of the signal clock time (green initiations) in Section 2.6 and 2.7, changes in a signal's offset and schedule in Section 2.8, and probability of green in Section 2.9. We will compare our estimates versus the ground truth measurements at an intersection in the city of San Francisco in Section 2.10.

2.2 Description of the Data Feed

The results in the first part of this dissertation are based on probe vehicle data which can be gathered from vehicles of any kind reporting at least their GPS coordinate, and velocity at a timestamp. As an example data source, the probe data from bus movements in the city of San Francisco is used. The bus data feed is provided by NextBus [16] for a number of cities in North America through eXtensible Markup Language or XML [39]. The attributes of interest are position and velocity of each bus along with their time stamp and the bus identification number. Also the bus route data and location of bus stops are extracted from the same data stream. A map of bus (and light rail) routes in San Francisco in Figure 2.1 is constructed by aggregating GPS updates from all buses within a twenty-four hour period. The focus on this paper is only on a few bus routes to show the feasibility of the proposed ideas.

Figure 2.2 shows example data from a portion of bus route 28 along Park Presidio Boulevard in the city of San Francisco. This is an aggregation of 2478 bus passes over an entire month. While each bus sends only four or five updates along the shown stretch of the route, the aggregated data is very revealing and correctly depicts the location of intersections and bus stops. Figure 2.3 shows the maximum and minimum

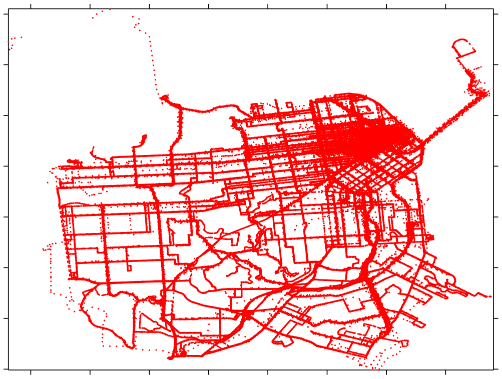


Figure 2.1: Aggregated plot of all bus (MUNI) updates for a period of 24 hours in the city of San Francisco.

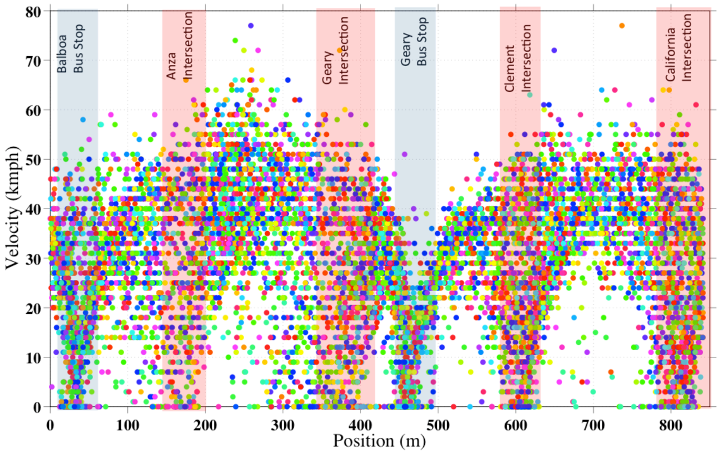
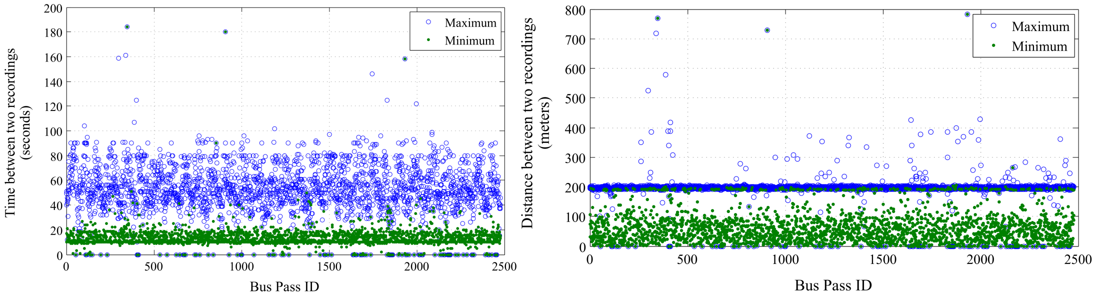


Figure 2.2: Scatter plots of San Francisco Route 28 bus updates over one month (September 2012). A total of 2478 bus passes are shown.



(a)

(b)

Figure 2.3: Maximum and minimum distance and time between two updates of San Francisco Route 28 buses over one month (September 2012) along the short portion of Park Presidio Blvd depicted in Figure 2.2.

distance and time between two updates of each bus pass and for every one of the 2478 bus passes. According to this data, the updates do not seem to be at regular time or distance intervals. Time updates are anywhere between every 10 seconds up to every 80 seconds or sometimes more. However there is a strong concentration of data at 200 meters distance intervals which indicates that most updates happen every 200 meters. From these update rates it seems that slower buses update at shorter distance intervals based on a time threshold (90 sec).

2.3 Data Transformation

We would like to estimate if a bus was stopped at an intersection, how long it was stopped, and at what time it left the intersection. We hope by aggregating this information for many buses we can estimate the duration of a red phase, the cycle length, the start of a green phase, and perhaps more. But because the update points for each bus are sporadic, we need to approximate a bus trajectory between each two update points. This is actually a data transformation process where low frequency probe data are transformed and consolidated into vehicle trajectories.

The most beneficial trajectory for our purposes is the trajectory which includes a stop at red signal. Figure 2.4 demonstrates the reconstructed velocity-time trajectories that include such a stop; and they may be used to estimate the green-initiation, the red interval, the cycle time, and perhaps more. There is also statistical patterns in travel trajectories with no stop and with constant acceleration that will be discussed later in Section 2.9.

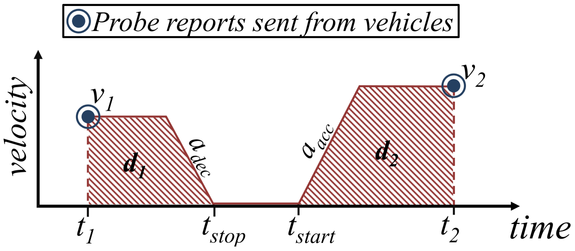


Figure 2.4: The probe reports fitted to the desired velocity-vs-time trajectories (full trajectory with a stop at red).

The Data Transformation process first checks the consistency of the identified probe vehicle passes with the trajectories shown in Figure 2.4. If successful then the process records the estimated trajectory to be used later by SPaT estimation processes. We also filtered out passes with low upstream velocity (less than 15 km/h for results in this chapter only), to ensure that the influence of heavy traffic is minimized on signal timing estimation.

Each identified vehicle pass is a series of three-tuple $[v_i, t_i, x_i]$ shown by filled circle points in trajectories of Figure 2.4; where v_i is the reported velocity, t_i is the timestamp of the report, and x_i is the distance between each location report and the upstream point of the Intersection-Phase (the distance is calculated using Haversine Formula [40]). The points with index $i = 1$ are the reports sent right before the stop-bar of the intended intersection; and the points with index $i = 2$ are the reports sent right after the stop-bar of the intended intersection.

Each reconstructed travel trajectory consists of a vector of $[velocity, time, position, a_{acc}, a_{dec}]$ where a_{acc} and a_{dec} are the average acceleration and deceleration as denoted in Figure 2.4. We use deceleration and acceleration of 2.2 m/s^2 and 1.0 m/s^2 respectively; we will later show how to obtain these values in next section.

The equivalent velocity-position trajectory is demonstrated in Figure 2.5. The variable x_{signal} is the location of the light or more specifically the stop-bar. The variables $d_1 = x_{signal} - x_1$ and $d_2 = x_2 - x_{signal}$ are areas under the velocity-time curve.

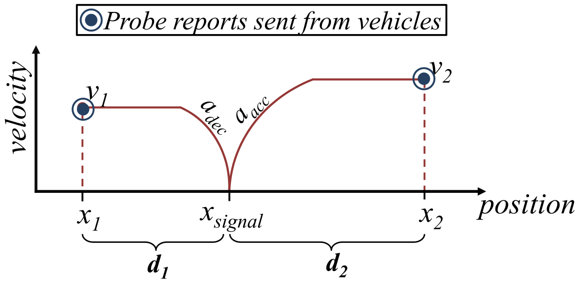


Figure 2.5: Equivalent velocity-vs-position trajectory (applicable to Figure 2.4).

The major extractable information from reconstructed trajectories is the time that a waiting vehicle starts moving at green (t_{start}) and the time that a moving vehicle comes to a stop at red (t_{stop}) estimated as follows:

$$t_{start} = t_2 - \max\{\frac{d_2}{v_2} - \frac{v_2}{2a_{acc}}, 0\} - \frac{v_2}{a_{acc}} \tag{2.1}$$

$$t_{stop} = t_1 + \max\{\frac{d_1}{v_1} - \frac{v_1}{2a_{dec}}, 0\} + \frac{v_1}{a_{dec}} \tag{2.2}$$

where the function max(.) in Equation (2.1) decides whether the estimated trajectories of Figure 2.4 include a constant velocity movement after the vehicle accelerates at green-initiation, and the function max(.) in Equation (2.2) decides whether the estimated trajectories of Figure 2.4 include a constant velocity movement before the vehicle comes to a full stop at a red light. The duration of red “observed” by a particular bus can be estimated as:

$$t_{red} = t_{SoG} - t_{stop} + \frac{v_1}{a_{dec}} \tag{2.3}$$

where $\frac{v_1}{a_{dec}}$ is the time it takes a bus to come to a full stop after the driver detects the signal is red, and t_{SoG} is the start-of-green (green initiation) which is approximately equal to t_{start} assuming the bus stops at stop bar. Aggregating t_{red} for a sufficiently large number of bus passes will later lead to an estimate of total red duration of a phase.

In the above calculations we assumed that acceleration and deceleration of buses were known and constants. We show next how probe data is used to approximate the average acceleration and deceleration of the bus fleet. We also demonstrate that t_{red} is not highly sensitive to reasonable variations in the value of acceleration.

2.4 Crowdsourcing Acceleration and Deceleration of Buses

Because of data sparsity, it is not possible to estimate the acceleration or deceleration of an individual bus. However velocity-position data from many buses shows a trend in start/stop trajectory as seen in Figure 2.2. For instance, at the Geary bus stop where a majority of buses come to a full stop, one can observe a clear slow-down and speed-up trend which can be used to estimate an average value for a bus deceleration and acceleration, later shown in Figure 2.6. To simplify the future steps of this work, we assume that deceleration to a stop and acceleration from a stop for a bus are constants and not functions of velocity. Hence the velocity while accelerating from a stop at a signal can be related to the distance traveled as follows:

$$v^2(x) = 2\bar{a}_{acc}(x - x_{signal}) \tag{2.4}$$

where $\bar{a}_{acc}$ is the average acceleration which is to be estimated from data. A similar equation can be written for a deceleration interval. By defining $y = x - x_{signal}$, $\psi = v^2(x)$, and $\theta = \frac{1}{2\bar{a}_{acc}}$ Equation (2.4) can be reorganized in the following linear parameterized form:

$$y = \theta\psi \tag{2.5}$$

Several data points can be stacked in a least-square approach to estimate the parameter θ and therefore $\bar{a}_{acc}$. As seen in Figure 2.6 there are several outlier data points that will skew the estimation result. So in the least square estimation, we have ignored the data points (in red) below a certain acceleration/deceleration profile (shown by dashed curves) to reduce the influence of outliers. Figure 2.6 shows the resulting curve fit for both deceleration and acceleration. The estimated deceleration is 2.2 m/s^2 and the estimated acceleration is 1.0 m/s^2. These values are consistent with bus acceleration measurements reported in [41, 42][1].

[1] The maximum sensitivity of t_{red} estimate in Equation (2.3) to variations in acceleration (also similarly deceleration) can be found to be:

$$\delta t_{red} = -v_2 \frac{\delta a_{acc}}{a_{acc}^2}$$

and because v_2 is at most around 20 m/s for a city bus and a_{acc} and a_{dec} are greater than 1 m/s^2, even a 20% error in approximation of a_{acc} ($\delta a_{acc}/a_{acc} = \pm 0.2$) results in a maximum error of 4 seconds for t_{red}. The error is much smaller in most places where v_2 is much less than 20 m/s.

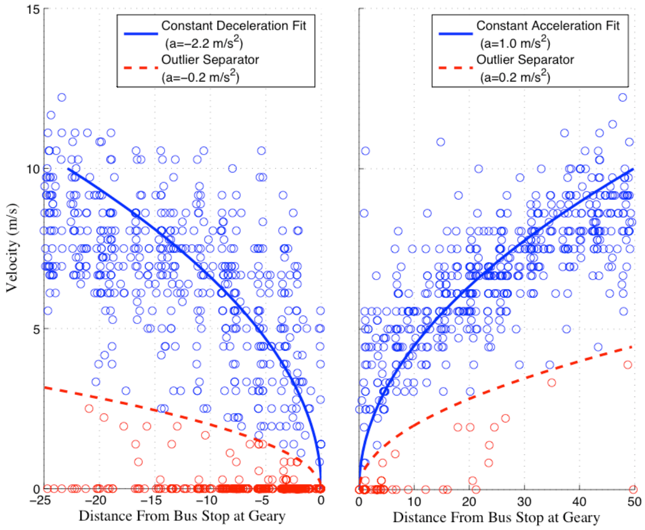


Figure 2.6: Estimation of average deceleration and acceleration of buses during stop and start using probe data.

2.5 Estimating Baseline Timing

The goal in this section is to determine if the baseline timing for lights can be obtained by *offline* aggregation and averaging of crowd-sourced bus data. In particular, we are interested in determining the duration of reds/greens of a phase and the cycle time of a traffic signal. Later we will investigate if a signal's clock time and schedule changes can be calculated. But we note that mere knowledge of baseline schedule, obtained offline and using only historical data, has statistical value even when a signal's clock-time is unknown. See for example [43] in which the baseline schedule of a light is used to predict the chance of a future green for an eco-driving application.

While we have results from several intersections in different locations in San Francisco, in the rest of this paper we focus on results for a segment of Van Ness street, between Lombard and Bush intersections. This is a sometimes congested street and therefore suited to test our proposed algorithms under (relatively heavy) city traffic conditions. Additionally, we have access to the actual signal timing cards of intersections of Van Ness and therefore can verify the validity of our estimates. Most intersections on this segment of Van Ness are fixed time intersections with the same cycle time and red duration throughout all days of the week. For most of these traffic signals, only offset times change during rush hour schedule, that could be estimated as we show later in this paper. We aggregate one month worth of data (September 2012) from two bus routes, route 47 and route 49, in the southbound direction totaling 4289 bus passes. This data is used to estimate signals' cycle time and the timing of the phases controlling southbound traffic on Van Ness, as explained next.

2.5.1 Estimating Duration of a Red Phase

For each bus pass we follow the procedure explained in Section 2.3 and for those that had stopped at a red, the observed red time is calculated via Equation (2.3). Aggregating this data provides an estimate of the duration of red for the corresponding phase. For example for the southbound phase on Van Ness street at Lombard intersection, there remained 347 bus passes after applying the procedure described in Section 2.3 to the 4289 total passes. Figure 7.3 presents the observed red for these 347 passes in two forms: The histogram of observed reds in the first subplot has a maximum of 68 seconds which is an upper bound estimate to duration of red phase. The second subplot shows the observed reds at different hours of a day for an entire month. During early morning hours (midnight-6am) and late night hours (7pm-11pm) where the queue lengths are expected to be shorter, we observe a maximum observed red of 60 seconds. This corresponds well to the actual timing of this intersection: According to the city timing cards, this intersection has a 90 second cycle time split to 60 seconds of red, 3.5 seconds of yellow, and 26.5 seconds of green for the southbound phase. Note also that many bus drivers may treat a yellow as red increasing their observed red time to a maximum of 63.5 seconds.

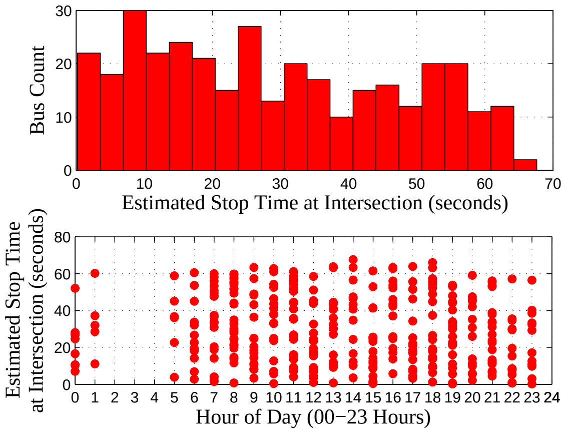


Figure 2.7: Stop time at red by each probe vehicle a) histogram b) stop time at different times of day (Southbound through phase on Van Ness Street at Lombard Intersection).

We repeated this process for a few other intersections on Van Ness and the results are summarized in Table 2.1. In most cases the red estimates are very close to the actual red. This is while, unlike Lombard Intersection, many of these intersections had a short red interval and a green-wave design that allowed most buses to pass through their green period; thus offering a smaller number of usable data points[2].

[2] A part of the larger error at Broadway intersection may be due to the steeper slope of Van Ness street at Broadway intersection which is not taken into account in crowdsourcing acceleration and deceleration of the buses.

2.5.2 Estimating Cycle Times

For fixed-time signals with phases that repeat cyclically, the time between green-initiations of a phase must be an integer multiple of the cycle time[3]. An approximation for a green-initiation can be obtained using Equation (2.1), i.e. the clock time that a bus starts accelerating from a stop at red. The difference between two consecutive approximations of green-initiations, based on bus movements, then must be an "almost" integer multiple of the cycle time, as shown schematically in Figure 2.8. Let's denote the time between approximated green-initiations as b_g, therefore,

$$b_g(j) = t_{start}(j+1) - t_{start}(j) \tag{2.6}$$

For a given cycle time C, we can then calculate the remainder of division of b_g and C as follows:

$$\mathrm{mod}_C(b_g) = b_g - \mathrm{round}(b_g/C)C \tag{2.7}$$

where the function round$(.)$ rounds its argument to the nearest integer and the function $\mathrm{mod}_C(.)$ is a modified definition of remainder of division by C that allows negative values. For example $\mathrm{mod}_{10}(12) = 2$ and $\mathrm{mod}_{10}(8) = -2$.

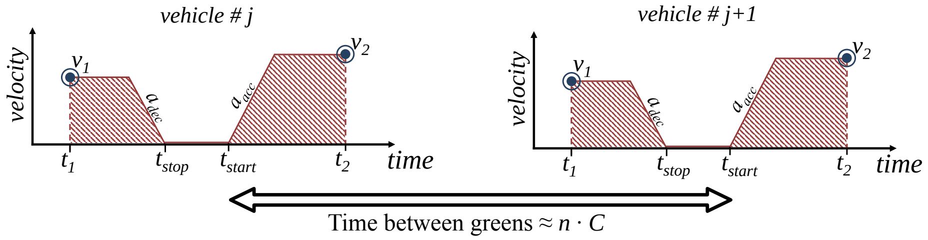


Figure 2.8: Time between consecutive green-initiations must be an integer multiple of cycle time for a fixed-cycle traffic signal.

We expect $\mathrm{mod}_C(b_g)$ to be close to zero on average, if the cycle time is fixed at C and signal clock drift between two qualifying bus passes is small. Therefore we propose to approximate C by solving the following optimization problem:

$$\bar{C} = \arg\min_C \sum_{j=1}^{n} \left(\frac{\mathrm{mod}_C(b_g(j))}{C/2} \right)^2 \tag{2.8}$$

where it is assumed there are $n+1$ qualifying bus passes during the interval of interest and therefore n

[3] Note that due to a signal's clock drift this may not be true for green-initiations that are far apart.

calculations of b_g. Observing that $-\frac{C}{2} < \text{mod}_C(.) \leqslant \frac{C}{2}$, we normalize the remainders by $C/2$ to ensure all values of C generate equivalent costs.

Because a signal cycle time is normally an integer in practice and has a limited range, one can conveniently solve the above optimization problem by trying every feasible C. We tried integer values between 1 and 120 seconds when determining cycle time of signals on Van Ness. To reduce the influence of signal clock drift we limit the choice of b_g to those within a few hours, e.g. 5 hours for results in this paper. Using one month worth of data, the estimated cycle time for Lombard intersection was 90 seconds, perfectly matching its actual value. This is visually illustrated in Figure 2.9 with histograms of $\text{mod}_C(b_g)$ for Lombard Intersection for four different values of C. As it can be seen, for $C = 90$ seconds, the histogram peaks strongly around zero despite various sources of uncertainty, i.e. unknown queue lengths and traffic conditions and approximations made in reconstructing bus trajectories. In the fourth subplot, we also observe small bumps near the tail ends; later in Section 2.8, we explain that these bumps are direct results of change in signal offset times during rush hour schedules.

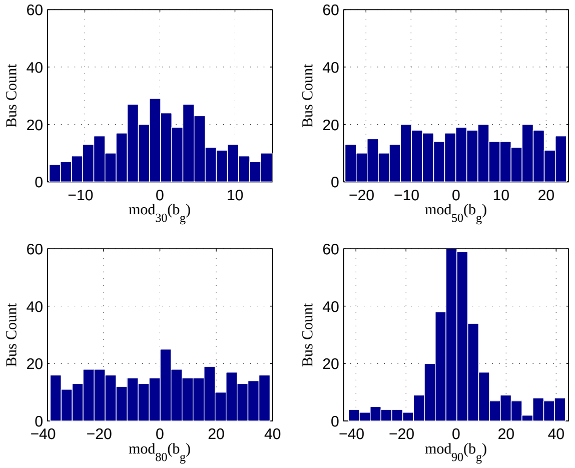


Figure 2.9: Deviation of approximated time between green-initiations from multiples of example cycle times. At the actual cycle time of $C = 90$ seconds, a clear peak can be observed.

Table 2.1 summarizes cycle estimates for a number of other intersections along Van Ness. For most, the estimated and actual cycle times are identical. For Washington Intersection, our proposed algorithm estimates the cycle time at exactly half of its actual value. This is partly due to lack of enough qualifying bus passes for this intersection. There were only 94 bus passes that qualified the filters for Washington as compared to 347 passes for Lombard Intersection. Also we were not able to obtain meaningful results for Bush intersection which is an actuated intersection with two different cycle times. Bush Intersection had also

Table 2.1: Red and cycle time estimates for a few southbound phases through Van Ness street, calculated using data from bus routes 47 and 49 gathered for September 2012.

Intersection	Actual Red (seconds)	Estimated Red (seconds)	Actual Cycle (seconds)	Estimated Cycle (seconds)	Qualifying Passes (count)
Lombard	60	60	90	90	347
Filbert	31.5	30	90	90	170
Green	31.5	35	90	90	86
Broadway	36	42	90	90	133
Washington	31.5	32	90	45	94
Bush	31.5/38.5	38	75/90	NA	41

very few (41) qualifying bus passes, as it was mostly green to buses traveling southbound.

2.6 Estimating Phase-Change (Green-Initiation)

Equation (2.1) can be used to estimate the time t_{start} that each bus left the intersection. However, a parameter called startup lost time (Δt_{lost}) which is the average time taken for a bus waiting at stop bar to react to a signal changing to green, needs to be taken into account for green-initiation estimation as in Equation (2.9). It is later found in Section 2.10 that the Δt_{lost}=6 seconds results in best estimations.

$$t_{SoG} = t_{start} - \Delta t_{lost} \tag{2.9}$$

2.7 Predicting Phase-Change (Green-Initiation)

For real-time in-vehicle applications, it is important to have a prediction of the start of future green (or red) phases. Predicting the start of a green (green-initiation) is a challenging problem: even for fixed-time signals that have fixed cycles, periodic projection of green-initiations can be inaccurate due to signal clock drift throughout a day. To address this problem, we propose to continuously estimate the start of a green phase based on the movement of buses that accelerate from a stop at an intersection. As a result, a moving average of the most recent estimated green-initiations is used to predict the next transition to green. More specifically, because of C-periodicity of a fixed-time light within each schedule, we can map the latest estimates of green-initiation to a single reference interval $[-\frac{C}{2}, \frac{C}{2}]$ by applying the mod_C operator, e.g. for the

i^{th} qualifying bus pass:

$$t_i = \mathrm{mod}_C(t_{SoG}(i)) \tag{2.10}$$

It must be emphasized that the t_{SoG}(i) estimations should be first adjusted by an offset which is usually added to the timings during the rush hour schedule. This is done by simply adding our estimated offset (as presented in next section) to the estimated t_{SoG}(i).

We can then create an average estimate of the green-initiation in this reference interval. Note that, a simple "linear" average will, in general, produce an erroneous estimate due to the cycle periodicity. See for examples the schematic in Figure 2.10 where four estimates of green, mapped to the linear interval, and their true average are shown on a straight line. As seen in this example, the correct average does not fall between the individual greens. The periodicity can be better visualized if the time axis is wrapped onto a circle shown in Figure 2.10. Each green-initiation can then be represented by a vector with angle $\theta_i = \frac{2\pi}{C} t_i$ on the circle. The average angle, $\bar{\theta}_{SoG}$, is determined by the direction of the vector sum of all individual vectors:

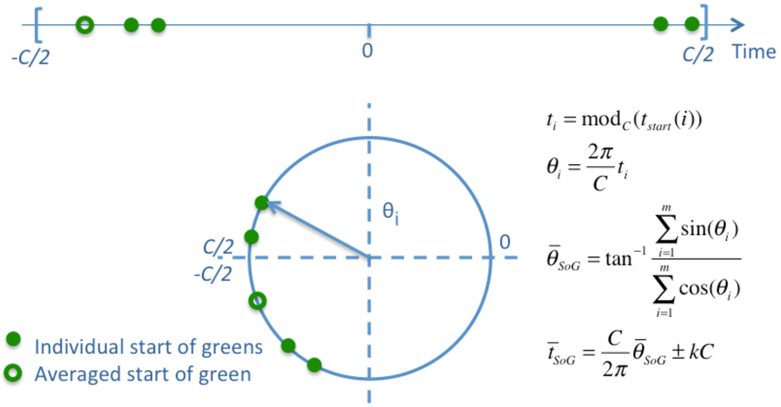

-C/2
0
C/2
Time
$t_i = \mathrm{mod}_C(t_{start}(i))$
$\theta_i = \frac{2\pi}{C} t_i$
θ_i
C/2
-C/2
0
$\bar{\theta}_{SoG} = \tan^{-1} \frac{\sum_{i=1}^{m} \sin(\theta_i)}{\sum_{i=1}^{m} \cos(\theta_i)}$
● Individual start of greens
○ Averaged start of green
$\bar{t}_{SoG} = \frac{C}{2\pi} \bar{\theta}_{SoG} \pm kC$

Figure 2.10: Schematic: Green-initiations mapped to a reference C-periodic interval for calculating the average and standard deviation of green-initiations.

$$\bar{\theta}_{SoG} = \tan^{-1} \frac{\sum_{i=1}^{m} \sin(\theta_i)}{\sum_{i=1}^{m} \cos(\theta_i)} \tag{2.11}$$

here m represents the number of samples used to calculate the moving average. The average start of the green is obtained by mapping back, the average angle to the time axis:

$$\bar{t}_{SoG} = \frac{C}{2\pi}\bar{\theta}_{SoG} \pm kC \quad k \in \mathbb{Z} \tag{2.12}$$

The variance of this estimate is then obtained based on the minimum cyclic distance to the average, equivalently calculated by:

$$\sigma^2_{SoG} = \frac{1}{m}\sum_{i=1}^{m}(\text{mod}_C(t_i - \bar{t}_{SoG}))^2 \tag{2.13}$$

We will show later in Section 2.10 that, in some instances, the accuracy of $\bar{t}_{SoG}$ can be enhanced, if we selectively choose samples that produce smaller variances. In other words with n latest samples, we propose to calculate $\bar{t}_{SoG}$ and σ_{SoG} for all possible combinations of $m < n$ samples and select the one with the minimum variance.

2.8 Estimating Changes in Signal Schedule

The traffic signals that we have considered on Van Ness street have 3 different schedules. While cycle times remain constant across multiple schedules for these intersections, each signal's offset with respect to other signals and also with respect to a reference clock switches as the schedule changes. For example at Lombard intersection and during weekdays, the start of the cycle is moved backward by 34 seconds at 6 AM and at 3 PM and moved forward at 10 AM and 7 PM. It is essential to estimate the change in offset and time of this change, if we are to solely rely on crowd-sourced data for predicting the start of a green. Here we report a couple of methods that were relatively successful in estimating time of change and amount of offset.

2.8.1 Estimating Time of a Schedule Change

We propose to detect a change in signal offset/schedule by keeping track of green-initiations and detecting when a green-initiation shifts off significantly from its periodic prediction. A smaller value of variance calculated in Equation (2.13) indicates that the corresponding m estimates of green-initiation are consistent with each other and multiple of C seconds apart. Right after a schedule change when the green-initiations are shifted by the offset times, the variance is expected to temporarily increase, until it is corrected by newer estimates of green-initiations. Jumps in the value of variance can then be indications of a change in signal schedule/offset times.

To test this hypothesis, we combined three months worth of data and calculated the variance of the moving average as a function of time of day[4]. Figure 2.11 shows the results for the intersection with Lombard for every day of the week. One can see clear jumps in the value of variance at 6 and 10 AM, and at 3 and 7 pm on a weekday. These correspond to the times that the signal schedule changes. For some days of the week there is also a large spike at around 8 AM; these spikes do not correspond to a schedule change, but perhaps are results of heavier traffic at that time. The plots for weekends do not have major spikes, which is consistent with the single schedule that is in effect on weekends. We conclude that spikes that happen recurrently on all weekdays are considered to correspond to signal schedule change while non-recurrent spikes may be due to heavy traffic.

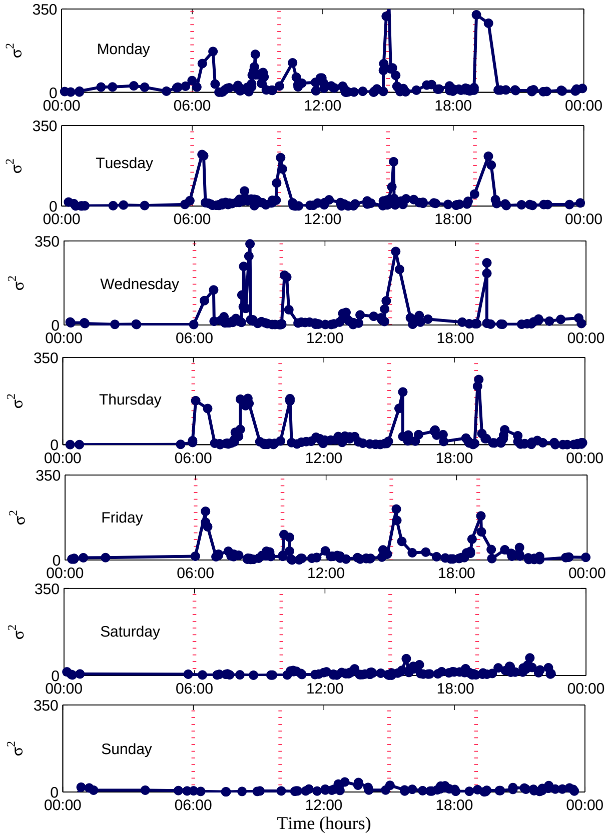


Figure 2.11: Variance of moving average estimate of green-initiation at different times and days of the week for Lombard intersection. The jump in variance corresponds, most often, to the change in signal schedule at 6 and 10 AM and 3 and 7 PM (shown by dashed vertical lines) on weekdays.

[4] A first attempt to only use a couple of weeks worth of data had many gaps due to sparsity in qualifying bus passes.

2.8.2 Estimating Signal Offset

In the histogram corresponding to $C = 90$ seconds in Figure 2.9, there were small bumps near the tail ends that were not explained in Section 2.5.2. Using the method of Expectation Maximization (EM) [44] we fitted a Gaussian mixture model to the histogram in Figure 2.9 and the result is plotted in Figure 2.12. EM found three distinct Gaussian clusters with parameters shown in Table 2.2. The major cluster is centered almost at zero, which was expected; and the two minor clusters are centered at almost ±30. These correspond closely to the 34 second shift in timing of the signal during a schedule change. We have further verified this hypothesis, by identifying time of days at which $\text{mod}_{90}(b_g)$ exceed ±30 seconds. In nearly all cases, this happens across multiple schedules, enforcing our hypothesis that the tail bumps are due to signal offset. In this case, the mean of this minor clusters can be used as an estimate to the amount of schedule offset.

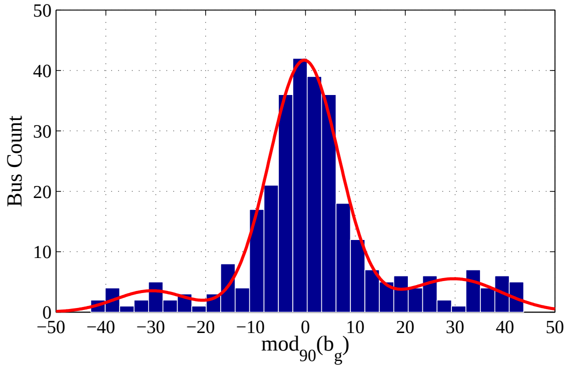


Figure 2.12: A Gaussian mixture model fitted to data of Figure 2.9 using the Expectation Maximization Algorithm. The peaks at tail ends correspond to the change to signal offset when schedule changes.

Table 2.2: Parameters of the Gaussian Mixture Fit to histogram of Figure 2.9.

mean (μ)	standard deviation (σ)	weight (π)
-30.78	7.32	0.07
-0.24	7.02	0.79
29.79	9.32	0.14

2.9 Direct Estimation of Green Intervals and Probability of Green

So far, all of our analysis has been based on movement of buses that had stopped at an intersection. We filtered out bus passes that had no intersection delay, e.g. those that cruised through a green. This approach discards a substantial amount of data, in particular for phases that either are often green or are timed in a green wave. But there is useful information that can be extracted from passes during a green: It is possible to interpolate a point in time that a phase was green based on the bus data before and after an intersection. Looking at Figure 2.13 and given the two update tuples $[t_1, x_1, v_1]$ and $[t_2, x_2, v_2]$ across one intersection, we propose the following steps:

- **Step 1:** To determine whether a bus stopped at an intersection or not, we propose to approximate the intersection delay, t_d, by subtracting projected travel time from actual travel time as follows:

$$t_d = (t_2 - t_1) - \frac{x_2 - x_1}{(v_1 + v_2)/2} \tag{2.14}$$

where $t_2 - t_1$ is the actual travel time and $\frac{x_2 - x_1}{(v_1+v_2)/2}$ is the estimated travel time if the velocity of the bus had changed linearly between v_1 and v_2.

- **Step 2:** Determine instances for which intersection delay calculated via Equation (2.14) is zero[5]. A zero value for t_d indicates (with high likelihood) that the bus passed through a green and moreover, its acceleration between two update points remained constant, as shown in Figure 2.13.

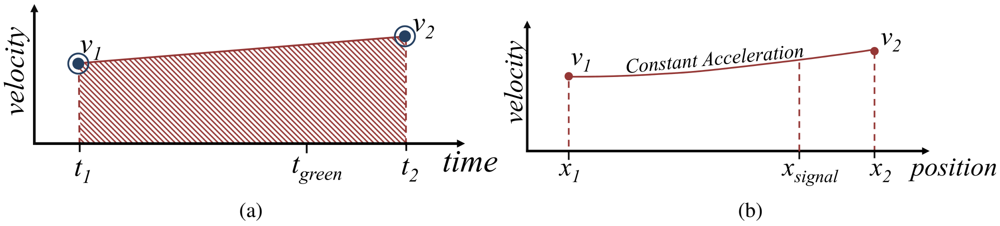


Figure 2.13: The probe reports fitted to the desired trajectory passing through a green light with constant acceleration. (a) velocity-vs-time (b) velocity-vs-position.

- **Step 3:** Interpolate between update times t_1 and t_2 to determine the point in time at which the signal was green. For the constant acceleration case, we have:

$$x_{signal} = x_1 + v_1(t_{green} - t_1) + \frac{1}{2}a(t_{green} - t_1)^2 \tag{2.15}$$

where $a = \frac{v_2 - v_1}{t_2 - t_1}$ is the constant acceleration between two update points. Here t_{green} denotes a time at

[5] We used a small threshold and accepted values sufficiently close to zero.

which the signal was green which is the feasible solution to the above quadratic equation:

$$t_{green} = t_1 + \frac{-v_1 + \sqrt{v_1^2 + 2a(x_{signal} - x_1)}}{a} \tag{2.16}$$

- **Step 4:** Ideally we would like to aggregate all point calculations of t_{green} to estimate intervals of green. For signals with fixed and known cycle time C, this can be done by mapping all values of t_{green} onto a reference interval $[0,C]$.

We carried out the above process for Lombard Intersection and the result is shown in the first subplot of Figure 2.14. When mapping all green times to a single interval, we have accounted for known changes in signal schedule. The second subplot is a histogram highlighting the concentration of points. In the ideal situation when a signal had no clock drift and repeated the same state at the exact same time every day, this mapping would result in an interval of green exactly matching signal's green time; i.e. 26.5 seconds for Lombard. But since the signal clock drifts, and also due to errors in reconstructing bus kinematics, the plotted green interval has a wider range than the actual green time. However there is much stronger concentration of mapped greens in the middle as shown by its histogram. This time period, and periods cyclically mapped forward, are where the probability of green is the highest. Even in the absence of any further crowd-sourced data, this probabilistic information is useful for many in vehicle applications (see [43] for instance).

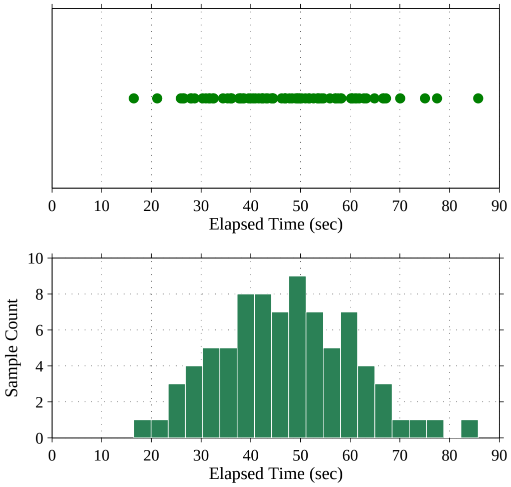
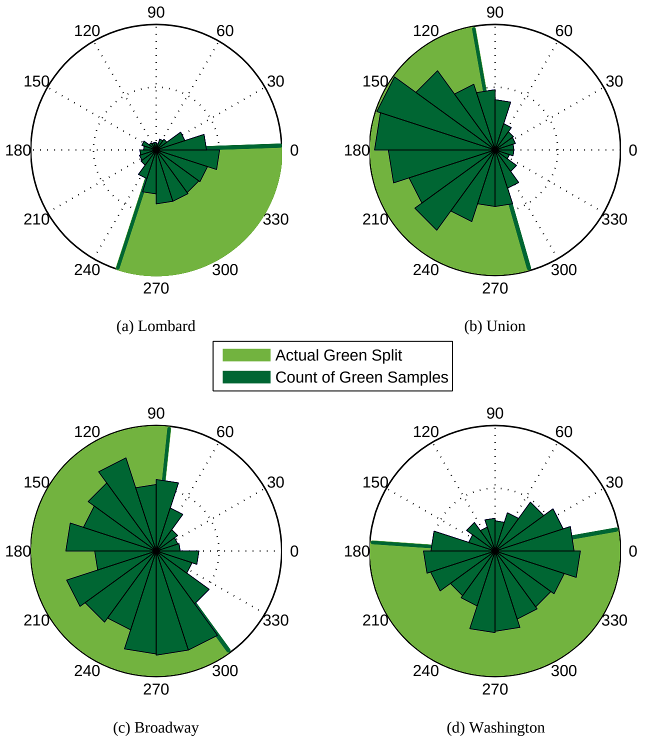


Figure 2.14: Green times mapped to one cycle interval. Southbound through phase on Van Ness Street at Lombard Intersection with cycle time of 90 seconds. Actual red was 60, actual green 26.5, and yellow 3.5 seconds.

Because of the cyclic periodicity, the data can be better visualized if mapped onto a polar histogram in which one revolution corresponds to one cycle time. Figure 2.15 shows such polar histogram plots for four different intersections along Van Ness. The height of each triangle represents the number of green samples within that triangle interval. Also shown by shaded areas on these plots are the actual green intervals, as observed and recorded in ground truth observations. It can be seen that the actual and crowd-sourced estimates of green interval match relatively well. The differences can be attributed to signal clock drift and also to errors in generating the crowd-sourced estimates.



(a) Lombard

(b) Union

(c) Broadway

(d) Washington

Figure 2.15: Crowd-sourced and actual green times mapped to one circular cycle interval in polar histograms (Southbound through phase on Van Ness Street at four different intersections).

2.10 Ground Truth Verification

To determine the accuracy of our estimates, in particular the green-initiations, we arranged a session of on-site ground truth tests at the intersection of Lombard and Van Ness streets on June 6, 2013. Between the hours of 7 AM and 4 PM, we recorded the actual start of a green of the southbound phase on Van Ness almost every 15 minutes as the ground truth. This was done with the aid of a computer program that upon a key press would log the time as synchronized with the NIST time server [45]. The human observer's reaction time was determined to be less than 0.3 seconds which is sufficiently accurate for the purpose of this study.

Concurrently, the green-initiations were predicted using the bus data feed and based on the procedure explained in Section 2.6 and 2.7. This was done in real-time via a crowd-sourcing backend server. The XML updates from routes of interest are continuously parsed and the data is written to a SQL data server. Another computational node constantly monitors the data to estimate green-initiations and records it back on the SQL server. We could monitor the agreement between actual green-initiations and crowd-sourced green-initiations, in real-time, via a PHP web-interface.

After each qualifying bus pass, new estimates for green-initiations were generated using i) the last data point only, ii) minimum-variance average of 3 samples chosen out of last 6 data points, and iii) minimum-variance average of 2 samples chosen out of last 4 data points. Note that crowd-sourced estimate of greens are sparse in time due to the fact that the bus data that qualifies our filters is infrequent. Therefore in between two actual estimates, the green-initiations are cyclically mapped using the estimated cycle time of the traffic light. Also the change in signal offset during schedule change is accounted for in this process. The estimated values for green-initiations are then compared to the actual ground readings of the green-initiations[6].

Figure 2.16 demonstrates the error between the crowd-sourced and actual green-initiations. The jumps in error plots in Figure 2.16 correspond to the times when a new qualifying bus pass occurs. The drift in between is due to the actual drift of the signal clock and is not a by-product of crowd-sourcing. The root-mean-square and maximum error of each estimation approach are summarized in Table 2.3. It can be observed that the minimum variance estimates are reasonably close to the actual timing with an RMS error of around 2.5 seconds. The estimate that was based on only last sample was more prone to error in this case.

Table 2.3: Root-mean-square and maximum estimation error for green initiations

Estimation Method	RMS Error (Sec.)	Max. Error (Sec.)
Last data point	8.0	24.3
3 out of 6 data points	2.6	7.7
2 out of 4 data points	2.5	8.2

[6] When comparing the estimated values of green-initiation to the observed ground-truth, we noticed that the error is inclined to the negative side. This is due to the value of a parameter called lost time (Δt_{lost}) which is the average time taken for a waiting bus to react to a signal changing to green, assuming there is no queue in front. This lost time is used as follows and as previously given in equation (2.9) to adjust the estimated green-initiation:

$$t_{SoG} = t_{start} - \Delta t_{lost}$$

We varied the value of t_{lost} to find a value that achieves the minimum RMS error in Fig. 2.16. We found that $t_{lost} = 6$ seconds results in minimum RMS error and included it in the results shown in Figure 2.16 and in Table 2.3

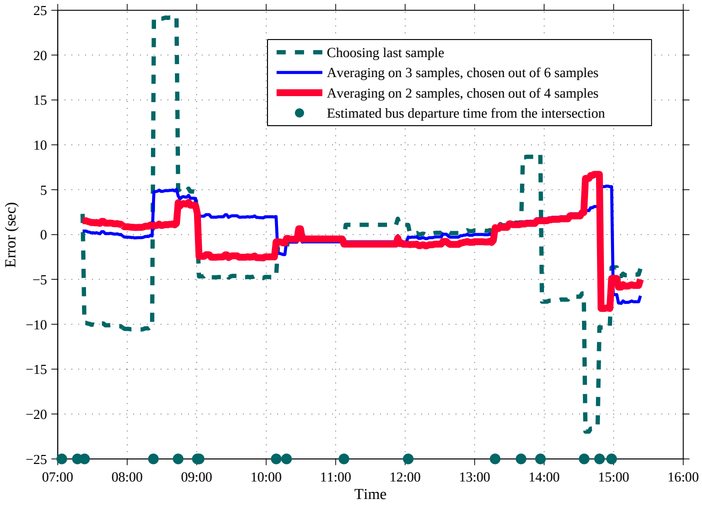


Figure 2.16: The error between crowd-sourced and actual green-initiations for the Van Ness southbound phase at Lombard intersection as recorded on June 6, 2013. Green circles highlight times of qualifying bus passages.

2.11 Acknowledgment

This research was sponsored by a research award from BMW Group Technology Office USA, Mountain View, CA; and it was also sponsored in part by the National Science Foundation grant number CMMI-0928533. I would like to thank members in BMW Technology Office's research team, Mr. Andreas Winckler and Mr. Grant Mahler for their insight and suggestions on the ideas proposed here. I also wish to thank Mr. Michael Smith of NextBus for the data he provided and Mr. Nainfeng Wan for his assistance with the Expectation Maximization algorithm. Eventually, my special thanks and acknowledgments to Prof. Ardalan Vahidi, my PhD advisor, for his support and supervision. We spent long hours together discussing the theoretical basis as well as collecting the ground truth data in the city of San Francisco for this part of the dissertation.

Chapter 3

Crowdsourcing Phase and Timing of Pre-Timed Traffic Signals in Presence of Queues

3.1 Introduction

This chapter describes the crowdsourcing-based system for phase and timing estimation of pre-timed traffic signals. As mentioned in previous chapters, the input crowd is a real-time feed of sparse probe vehicle data and the output is an estimated collection of Signal Phase and Timing (SPaT) information. Nevertheless, different from the previous chapter, the approach described in this chapter ensures the accuracy of the SPaT estimations even in presence of queues. This was achieved by investigating the probe data influenced by the heavy traffic and the delay in queues.

Only the departure pattern (discharge) of the queue is studied in this chapter. Several queue discharge models such as [46, 47, 48] have the potential to be used in our application. Cheng et al. in [35, 49] used the traffic shockwaves in order to model the queue discharge time. The shockwaves theory, also used in [46, 47], was originally extended by Stephanopolos and Michalopoulos in [48] to the signalized intersection applications. Kerper et al. in [34] claim that they use the queue discharge model of [28] with no explanation provided; however, the way they benefit from this model is not clear. The queue discharge model presented in this chapter is different from the aforementioned models in that it is based on an average discharge headway model. There is no evidence so far that the presented model implies any significant advantage to other models; however, it will be shown later in this chapter that the proposed queue dissipation formulations have a slightly better correlation with the collected ground truth data.

The SPaT estimation in previous chapter (Chapter 2) did not consider the influence of queue delay, and it was based on filtering out the vehicle passes that appeared to be influenced by heavy traffic and long queues. This is why the position in queue was not considered in the methodologies presented in that chapter. However, a considerable number of vehicle passes occur in heavy traffic and excluding them will negatively influence accuracy of SPaT estimation algorithms. In Chapter 2 many of the movements during heavy traffic period were filtered, reducing the number of available data points, causing estimation errors during heavy traffic conditions. This is why the approach of Chapter 2 is suitable for intersections that have light traffic condition with occasional short periods of heavy traffic throughout the day.

In this chapter, first ,transforming the probe data influenced by the heavy traffic into desired vehicle trajectories is explained in Section 3.2. The detailed crowdsourcing methodologies that consider delay in queues are explained with estimation results in Section 3.3. Section 3.4 describes in detail how SPaT estimations are affected by idling periods in queue. Finally, more ground truth verifications are provided to evaluate the SPaT estimations, and the queue dissipation formulations.

3.2 Data Transformation

This section describes the data transformation process where low frequency probe data are transformed and consolidated into vehicle trajectories. It will be shown that the most beneficial trajectory for our purpose in this chapter is the trajectory that not only includes a stop at red signal but also includes a probe report sent in queue. Figure 3.1(b-c) demonstrates the reconstructed velocity-time trajectories that include such a stop and a probe report sent in queue; while Figure 3.1(a) demonstrates the reconstructed trajectory not influenced by delay in queue and used in previous chapter.

The Data Transformation process first checks the consistency of the previously identified probe vehicle passes with the trajectories shown in Figure 3.1. If successful then the process records the estimated trajectory to be used later by SPaT estimation processes. The reports sent while waiting in a queue, if available, are denoted with index q. Figure 3.1(b) and (c) have both a probe report sent in queue with approximately zero reported velocity.

The equivalent velocity-position trajectory is demonstrated in Figure 3.2. The variable x_q, if avail-

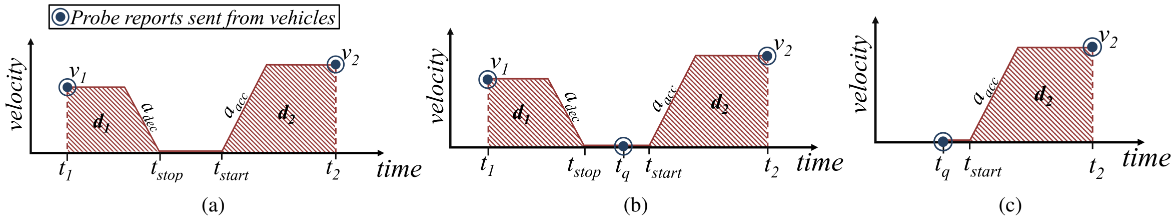


Figure 3.1: The probe reports fitted to the desired velocity-vs-time trajectories:
(a) full trajectory with a stop at red but not influenced by delay in queue.
(b) full trajectory with a stop at red and with a probe report sent in queue.
(c) partial trajectory with a stop at red and with a probe report sent in queue.

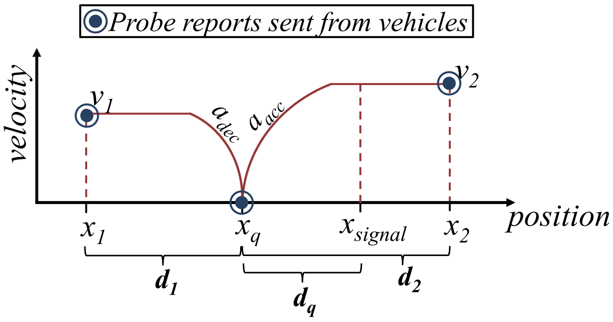


Figure 3.2: Equivalent velocity-vs-position trajectory. (applicable to Figure 3.1(b) and also to Figure 3.1(a) and (c) with modifications).

able, is the reported position of the vehicle while waiting in queue and x_{signal} is the location of the light or more specifically the stop-bar. The variables $d_1 = x_q - x_1$ and $d_2 = x_2 - x_q$ are areas under the velocity-time curve, and d_q is *position in queue*. It should be emphasized that *position in queue* is different from the term *queue length* used in literature such as in [50].

The major extractable information from reconstructed trajectories is the time that a waiting vehicle starts moving at green (t_{start}) and the time that a moving vehicle comes to a stop at red (t_{stop}) estimated as in Equation (2.1) and Equation (2.2). If the vehicle updates cannot be fit into any of the possible trajectories of Figure 3.1 with stop, then it does not necessarily mean that the vehicle has never stopped. It just shows that the actual travel trajectory is not fittable to any of the desired trajectories.

Table 3.1 reports the percentage of total vehicle passes in a year which are fittable into the desired trajectories. It can be seen that by not ignoring the probe data that are influenced by queue delay, a greater number of the vehicle passes will be available to the SPaT estimation algorithms.

Table 3.1. Counts of passes during one year at four intersections.

	Total Identified passes	Pass counts fittable to Fig. 3.1 (a)*	Pass counts fittable to Fig. 3.1 (b)	Pass counts fittable to Fig. 3.1 (c)
Lombard Intersection	49361	4476 (9.07%)	428 (0.87%)	5295 (10.73%)
Green Intersection	49343	1532 (3.10%)	103 (0.21%)	897 (1.82%)
Vallejo Intersection	49325	4468 (9.06%)	221 (0.45%)	1553 (3.15%)
Broadway Intersection	46938	3225 (6.87%)	153 (0.33%)	5543 (11.81%)

* passes captured by the method described in Section 2.3 of previous chapter.

3.3 SPaT Estimation and Prediction

The methods described in this section cover the Phase-Change (Green-Initiation) Estimation, and Baseline Timing Estimation in presence of queues.

3.3.1 Green-Initiation Estimation

When a traffic light changes to green, drivers should wait for the queue in front to move before they can start moving at green. As shown in Figure 3.3, this waiting time is denoted by $\Delta t_{waiting}$ in this paper and depends on the discharge rate of the queue as well as the time taken for the vehicle drivers to react to the signal changing to green.

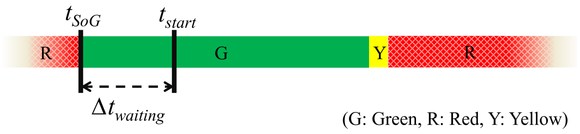


Figure 3.3: $\Delta t_{waiting}$ is the waiting time for moving after the green-initiation or Start-of-Green (t_{SoG}).

As a result, based on estimates of $\Delta t_{waiting}$ and t_{start}, the green-initiation (t_{SoG}) can be estimated as:

$$t_{SoG} = t_{start} - \Delta t_{waiting} \tag{3.1}$$

However, depending on which travel trajectory the probe data can be fitted to, there are two approaches to estimate green-initiation:

- First approach (as used in Chapter 2) uses the probe data that appears to be less influenced by heavy traffic and delay in queue. For this reason only the probe reports fittable to the trajectory of Figure 3.1(a) (or Figure 2.4) with high upstream velocity (v_1) are selected to estimate the time t_{start}. Because it is assumed that there is no long queue in front, an average value of $\Delta t_{waiting}$ for the first few vehicles in queue is used in green-initiation estimation. Please note that this average was denoted as Δt_{lost} in Chapter 2 and its value was estimated as 6 seconds in our application for transit buses.

- Second approach (as proposed in this Chapter) uses the probe reports that reveal the position of the probe vehicle in queue. This actually includes the reports that are fittable to the trajectories of Figure 3.1(b) or (c) where at least one probe report sent while in queue is available. Knowing the position of the vehicle in queue, the queue waiting time ($\Delta t_{waiting}$) can be estimated using the queue clearance time model explained later in Section 3.4. This approach is expected to be more accurate than the first one in persistent heavy traffic conditions, mainly because actual position in queue is available.

3.3.2 Signal Schedule Change Estimation

As described in previous chapter (Section 2.8), an offset (t_{offset}) is usually added to the timings during the rush hour schedule. Our proposed approach not only estimates the time that the schedule of a traffic light changes but also estimates the offset that is added to the timings. The process described in Section 2.8 is explained here again but in a different way and by using a different data source (probe reports that reveal the position of the probe vehicle in queue).

First, the green-initiations estimated using the second approach of previous subsection are sorted based on the weekday and the time of the day. Then, the variance of the average (σ^2) of few consecutive green-initiations (e.g. 5 green-initiations) is calculated; and according to Figure 3.4 this process is repeated for the next consecutive green-initiations till the whole list of green-initiations of each weekday is covered. By putting the calculated variances together, a trajectory is constructed which demonstrates the change of the variance with respect to time of each week-day. Please see Figure 3.5 for a sample plot of the variance trajectory.

The spikes in the variance trajectory plot of Figure 3.5 actually show the times that a signal schedule is changed. The more probe data is used, the sharper the spikes are. As a result, the probe data collected of almost ten months is used in depicting Figure 3.5, although fewer months of collected probe data is also enough to have detectable schedule change spikes. The variance trajectories of Figure 2.11 include some extra and misleading large spikes due to heavier traffic in the middle of rush hour; however, considering the

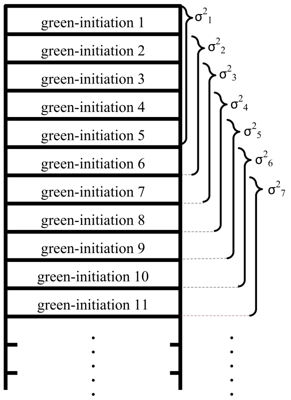


Figure 3.4: Extracting the variance (σ^2) trajectory from the green-initiations.

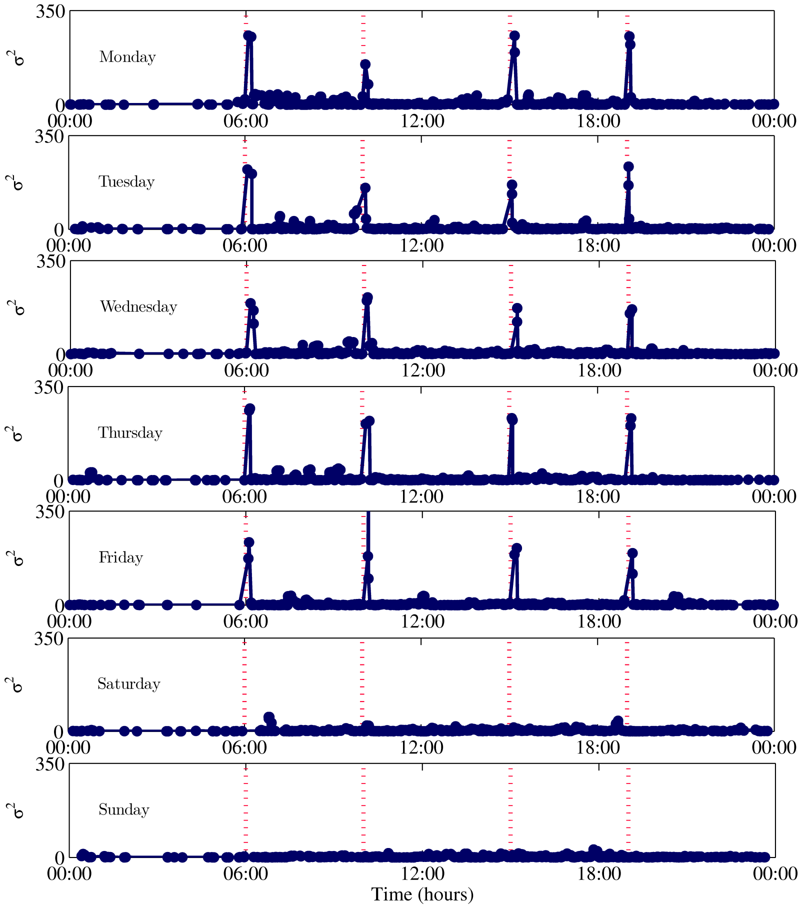


Figure 3.5: The variance of estimated green-initiations for Lombard intersection. The actual schedule-changes happen at the dashed vertical lines which are comparable to the jumps in the trajectories.

influence of queue waiting time on SPaT estimation, all the spikes in Figure 3.5 are solely the results of the schedule change.

3.3.3 Green-Initiation Prediction

The green-initiation prediction is the process of predicting the next transition to green for the desired Intersection-Phase. This process was explained in detail in Section 2.7; and it is repeated here in summary as follows:

The next green-initiations should be continuously predicted using the most recent reconstructed trajectories of vehicles that accelerate at green. As a result, a moving average of the most recent estimated green-initiations is used to predict the next transition to green. However, three adjustments should be conducted before averaging the estimated green-initiations:

- First, the green-initiation estimations should be adjusted by the estimated offset if they happened during the rush hour schedule change. This is done by simply adding t_{offset} to the estimated t_{SoG}. In this way, all the green-initiation estimations are synchronized to a same time reference.
- Second, the adjusted green-initiation estimations are mapped to one cycle interval before being averaged.
- Third, a filter is used to filter out the outliers and wrong estimations. As described in Section 2.7, this filter selectively chooses the green-initiation estimations that produce smaller variances.

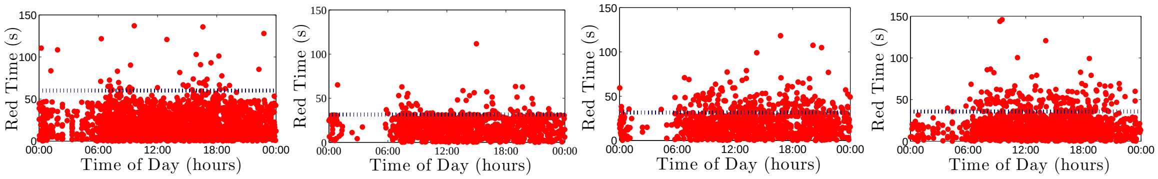
After conducting the aforementioned three adjustments, the average of these green-initiations is calculated ($\bar{t}_{SoG}$) which is used to predict the next green-initiation after current time. The improvements in green-initiation prediction are demonstrated later in Section 3.5.1.

3.3.4 Red Split Estimation

The duration of red "observed" by a particular vehicle can be calculated using the trajectories of Figure 3.1(a-b) as previously given in Equation (2.3). The trajectory of Figure 3.1(c) is also used in calculating the observed red interval by verifying Equation (3.2) if $t_{SoG} > t_q$; where t_q is the timestamp of the zero-velocity probe data sent while waiting in queue.

$$t_{red} = t_{SoG} - t_q \tag{3.2}$$

Figure 3.6 shows scatter plots of t_{red} calculated using the aforementioned equations for four intersections in San Francisco. It is expected that the maximum of the aggregated calculations would be actually an upper bound estimate to duration of the actual red phase.



(a) Lombard Intersection

(b) Green Intersection

(c) Vallejo Intersection

(d) Broadway Intersection

Figure 3.6: The red time observed by vehicles throughout the day of ten months at intersections along VanNess St. (the actual intervals were available through city timing cards and are shown by dashed horizontal lines).

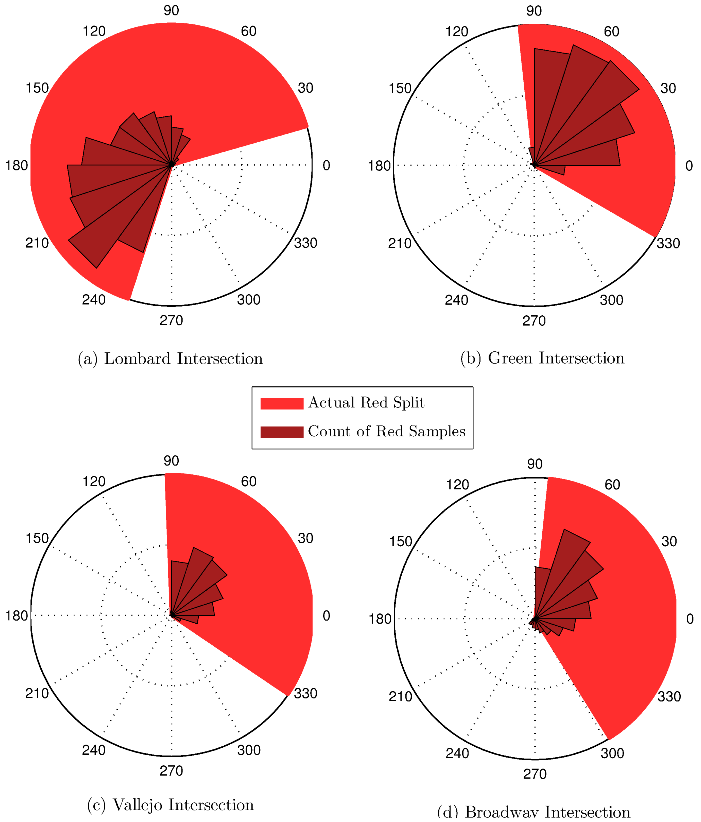
3.3.5 Red-Probability Estimation

This section demonstrates how to extract the probability distribution of red signal by aggregating the probe reports that are sent from the vehicles waiting in queue at red signal. The method proposed in this section completes the method of Section 2.9 in extracting the probability of green. For this purpose, the timestamps of the zero-velocity reports that has been sent while waiting in queue are collected (more specifically the t_q timestamps of Figure 3.1(b-c)). Nevertheless, the t_q timestamps do not necessarily denote the times at which the signal is red; and the condition of $t_q < t_{sog}$ should be verified before collecting t_q as a timestamp sample corresponding to red signal.

In order to synchronize all the collected t_q timestamps to a same time reference, the estimated offset (t_{offset}) is added to the timestamps that has occurred during the rush hour schedule change. The goal here is to aggregate all these adjusted t_q timestamps so that a probability distribution can be achieved for intervals of red phase. However, before aggregating the adjusted timestamps, all the timestamps should be mapped to one cycle interval by Equation (3.3) where C is the cycle time of the desired Intersection-Phase estimated by the method explained in Section 2.5.2.

$$t_{q,mapped} = t_q - \text{round}(t_q/C)C \tag{3.3}$$

Equation (3.3) maps all timestamps of t_q onto a reference interval of [0,C] in Unix-Time. These mapped timestamps are all aggregated and can be plotted in polar histograms such as Figure 3.7. As shown in this figure, the interval [0,C] can be mapped to an interval of [0,2π] because of the cyclic periodicity. The longer each triangle of histograms is, the more red samples it includes. The shaded portions of the cycle time are the actual red intervals which are depicted according to the city timing cards and the ground truth observations. The histograms represent the probability distributions of red intervals which match very well with the actual red intervals.



(a) Lombard Intersection

(b) Green Intersection

(c) Vallejo Intersection

(d) Broadway Intersection

Figure 3.7: Polar histogram of the red signal timestamps compared to the actual red splits (the data was collected for ten months at four intersections along Van Ness street)

3.4 Queue Waiting Time

As explained in Section 3.3.1, the key feature of the SPaT estimatior in heavy traffic conditions is inclusion of an estimate of the wait time in queue after green-initiation. The following subsections respectively describe: how to formulate this waiting time based on the expected queue clearance time, how to find an estimate of the queue clearance time via a queue discharge model, and how to estimate the unknown parameters of the model.

3.4.1 Waiting Time Formulation

Let's assume that a probe vehicle is the N^{th} vehicle waiting in a queue at red, as shown in the time-space diagram of Figure 3.8. Then, the clearance time, denoted by $\Delta t_{clearance}$, is the discharge time that it takes all of the N waiting vehicles to pass and leave the stop-bar after the start of the green signal. However, as it is plotted in Figure 3.8, the clearance time of N queued vehicles consists of two parts: the waiting interval that it takes the N^{th} vehicle to start moving after green-initiation ($\Delta t_{waiting}$) plus the interval that it takes that vehicle to travel all the way up to the stop-bar and cross the stop-bar (Δt_{travel}). As a result, the following formulation is proposed here to estimate the waiting time in queue:

$$\Delta t_{waiting} = \Delta t_{clearance} - \Delta t_{travel} \tag{3.4}$$

With the queue clearance time and the estimated travel time in hand, then it is quite straightforward to compute the waiting time and finally the green-initiation (t_{SoG}). This is demonstrated in Figure 3.9 where

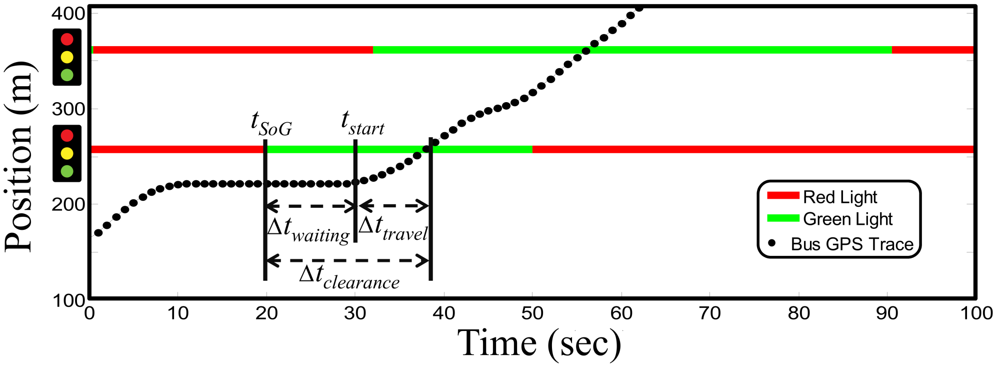


Figure 3.8: The time-space diagram of a probe vehicle (bus) waiting for moving after the start of the green signal; the clearance time is the time from t_{SoG} to the instant at which the probe vehicle passes the stop-bar.

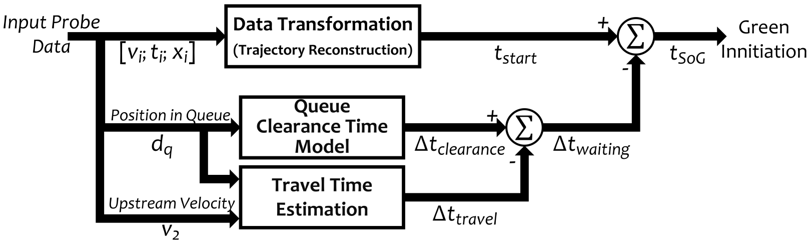


Figure 3.9: Green-Initiation estimation knowing the position of the probe vehicle in queue.

Δt_{travel} is calculated[1] using Figure 3.2; and the expected $\Delta t_{clearance}$ is derived form a model proposed in the following subsection.

3.4.2 A Model for Queue Clearance Time

The clearance time of queued vehicles can be represented by the summation of discharge headways. Figure 3.10 shows the average discharge headway, according to the specifications given in [51]. The $Headway(n{=}1)$ is the interval between green-initiation and the time that rear wheels of first vehicle cross the stop-bar, the $Headway(n{=}2)$ is the interval between the first vehicle and the second vehicle leaving the stop-bar, and so on. As a result, the summation of headways, as introduced in [52] and as given in Equation (3.7), equates to the time from green-initiation to the instant at which the N^{th} vehicle of the queue crosses the stop line.

$$\Delta t_{clearance} = \sum_{n=1}^{N} Headway(n) = hN + \sum_{n=1}^{N} \Delta_n \tag{3.7}$$

Due to the start-up reaction and acceleration, the headways for the first few vehicles are greater than h and are shown as h+Δ_n in Figure 3.10 where Δ_n is the incremental headway for the n^{th} vehicle [51]. In this paper, the incremental headways are assumed to decrease exponentially with the position in queue. As a result, an empirical formulation for the queue clearance time is achieved by rephrasing Equation (3.7) as:

$$\Delta t_{clearance} = hN + \Delta_1 \sum_{n=1}^{N} \mathrm{e}^{-(n-1)} \tag{3.8}$$

[1] The Δt_{travel} is the expected travel time between x_{signal} and x_q:

$$\Delta t_{travel} = \max\{\frac{d_q}{v_2} - \frac{v_2}{2a_{acc}}, 0\} + \frac{v_s}{a_{acc}} \tag{3.5}$$

where v_s is the velocity at stop-bar (x_{signal}) and can be a value equal or lower than v_2 as follows:

$$v_s = \sqrt{2a_{acc} \times \min\{d_q, \frac{v_2^2}{2a_{acc}}\}} \tag{3.6}$$

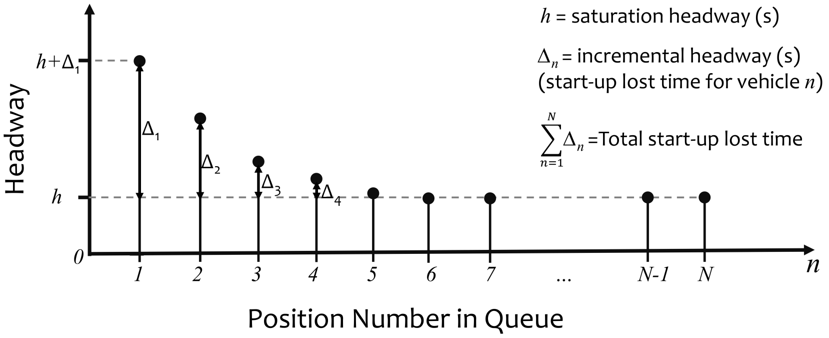


Figure 3.10. Average Discharge Headway.

It must be noted that if the heavy traffic passes are excluded (same as Chapter 2) then the average of $\Delta t_{waiting}$ interval for the first vehicle in queue (6 seconds) can be used in all signal timing estimations.

3.4.3 Parameter Estimation

The clearance time model provided in Equation (3.8) is in fact a linear combination of saturation headway (h) and the first incremental headway (Δ_1). As a result, Multiple Linear Regression model (MLR) can be used to estimate these parameters. However, there are two challenges in using linear regression for this purpose:

First, the regression variable is the vehicle position number in queue (N) which is not available. However, it can be estimated by Equation (3.9), where L_v is the average distance that a vehicle occupies in queue (20 ft), and $\lfloor . \rfloor$ is the flooring function.

$$N = \lfloor \frac{d_q}{L_v} \rfloor + 1 \tag{3.9}$$

Second, gathering enough observational data on queue clearance time is time consuming. For this reason, an approach is proposed here which provides enough samples of queue clearance time without the need of gathering them locally at intersections. This is achieved using Equation (3.10) where the green-initiation timestamp of a sample Intersection-Phase ($t_{SoG,observed}$) was locally collected from direct observation, and t_{start} is estimated based on the reconstructed trajectories (Figure 3.1(b) or (c)). Multiple of C seconds is also included in Equation (3.10) because $t_{SoG,observed}$ is a locally collected green-initiation and might be

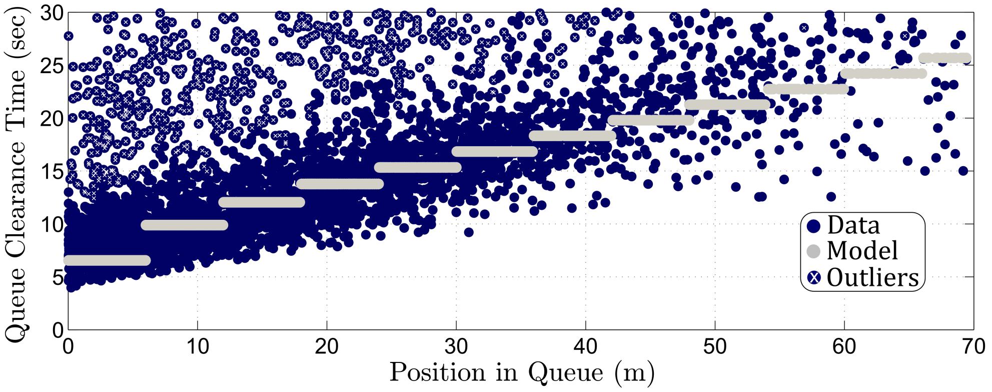


Figure 3.11: The queue clearance time model fit to data.

days before or after the estimated t_{start}.

$$\begin{aligned} \Delta t_{waiting} &= t_{start} - t_{SoG,observed} \pm kC \quad k \in \mathbb{Z} \\ \Delta t_{clearance} &= \Delta t_{waiting} + \Delta t_{travel} \end{aligned} \tag{3.10}$$

The clearance time model in Equation (3.8) is then verified to fit the aforementioned data, as shown in Fig. 3.11. However, this data shown in Fig. 3.11, represented by the blue circles, do not seem to be symmetrically distributed. The queue clearance data is skewed most probably because there are many real world factors, such as lane blockage and downstream queue spillback, that would prolong the time needed for a queue to dissipate. On the other hand, usually there is no factor that could possibly make the queue clearance time shorter than its expected value. This explains why data is skewed to the right and not to the left. As a result, in order to reduce the influence of the unwanted events such as lane blockage, not all the data shown in Fig. 3.11 was used for fitting purposes. For each one meter increment in distance to stop-bar, we had labeled the data points that were more than 1.0 times the inter-quartile range above the 75th percentiles as outliers. These outlier points were removed from data and are cross-marked in Fig. 3.11. Please note that the clock drift could also be a reason that the queue clearance calculated by (3.10) is spread out.

The estimated regression coefficients of this curve fit are h=1.47s and Δ_1=5.08s for the through movement which are consistent with the measurements in literature [28, 53]. However, the first incremental headway Δ_1 looks slightly greater than expected because of the low acceleration of buses compared to conventional passenger vehicles, and also because of our slightly different definition of headway which considers the rear wheels crossing the stop-bar instead of the front wheels. As an empirical verification of the estimated parameters, assume there is no queue in front of the vehicle then the model estimates the clearance time to be equal to h+Δ_1=6.6s. This yields a waiting time (somewhat akin to first driver's reaction time) of about 1.6s-2.6s considering that it takes a vehicle in front of queue about Δt_{travel}=4s-5s to completely pass the stop-bar. This is consistent with our observations in street and also with results in [53].

It must be emphasized that the aforementioned curve fitting was conducted only to get an idea of the queue clearance parameters values, and as it is verified in subsection 3.5.2, it is not necessary to repeat the process for every intersection-phase. However, the results are only applicable to through movement, and similar parameter estimation should be repeated for left turn or shared left/through lanes.

3.5 Ground Truth Verification

While in the previous sections most of the obtained estimations were compared with the city timing cards; this section provides more verifications for the green-initiation predictions and the proposed queue formulations based on the collected ground truth data.

3.5.1 Verification of Green-Initiation

In order to verify the accuracy of green-initiation predictions, we collected the actual green-initiations locally at a sample intersection. These time samples were actually collected by a computer program that would log the time whenever the observer pressed a key at the change of red to green. The program was synchronized to the NIST time server [45] and was used to record the actual green-initiations between hours of 2 PM and 10 PM. This period of the day was selected so that the proposed green-initiation estimator could be evaluated during the evening rush hour traffic.

Concurrent with the aforementioned ground truth data collection, the green-initiations were also predicted by crowdsourcing the probe data sent from the public buses passing over the same intersection. These predicted green-initiations are compared to the actual ground truth data collected. However, before comparing these two time arrays, the gaps between their timestamps should be filled by cyclically mapping each timestamp to the next one. The error between these predicted green-initiations and the collected actual green-initiations is shown in Figure 3.12. The error shown in solid red line is the error of the estimation method when only using data from the probe vehicles that stop at red, send a report while waiting in queue, and leave the intersection at green. These vehicle passes should fit Figure 3.1(b) or (c) though. The error shown in dashed black line is the error of the estimation approach of previous chapter that only took into account those probe vehicles that stopped at red, and left the intersection at green without sending any report while in queue. Because the position in queue is not available in this case, these vehicle passes should not be influenced by queue delay and should fit Figure 3.1(a) with high upstream velocity.

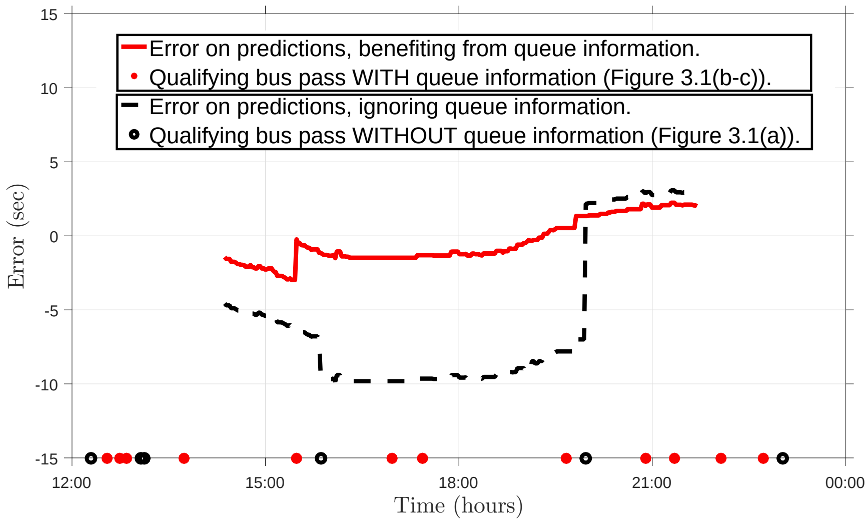


Figure 3.12: The error between the predicted and actual green-initiations (southbound phase at Lombard intersection as recorded on April 25, 2013).

As it was expected in Subsection 3.3.1, Figure 3.12 demonstrates that during the persistent heavy traffic conditions, the probe reports that include the vehicle position in queue, result in more accurate phase-change predictions compared to the reports that do not reveal any queue information.

Please note that the jumps in error plots in Fig. 3.12 correspond to the times when new qualifying bus passes occurred in this particular scenario. These times are shown with filled or open circles depending on the trajectories that the corresponding passes are fitted to. Also the drift in plotted error in between the passes is due to the actual drift of the signal clock.

3.5.2 Verification of Queue Formulations

Three ground truth data collection sessions were arranged at 10 intersections along Van Ness street, San Francisco. One of the colleagues physically sat in buses and recorded the trajectory with a GPS tracking device at high frequency. In this way, GPS location and velocity data was collected at the frequency of 1 Hz while traveling on the transit buses. Fig. 3.8 shows a sample collected high-frequency GPS trace plotted over time-space diagram, wherein the timing of the lights are plotted using the baseline timings given in the city timing cards and the locally collected green-initiation timestamps.

We first searched the aforementioned plotted GPS traces for the stops at red at any of the 10 intersections along Van Ness street. Knowing the stop-bar positions of the intersections, the plots reveal the instant at which the buses pass the stop-bar at green phases. Furthermore, the corresponding time-velocity diagrams (not shown in Fig. 3.8) reveal the instant at which the waiting buses in queues start moving at green (t_{start}). Using the aforementioned extracted timestamps, the bus actual travel time, and also the queue waiting and clearance intervals are extracted as shown in Fig. 3.8. These values are tabulated in Table 3.2 as Ground Truth (G.T.).

The estimations of the queue waiting and clearance times and the travel time are also given in Table 3.2 denoted by Estimations (Est.) which are calculated by applying the proposed queue formulations in Section 3.4 to the collected ground truth data. The Root Mean Square Error (RMSE) between the estimations and the collected ground truth data, observed at 10 intersections, was 2.68, 1.37, and 1.98 seconds for $\Delta t_{clearance}$, Δt_{travel}, and $\Delta t_{waiting}$ respectively which are accurate enough for the application in this manuscript. The box plot of the errors are also given in Table 3.2.

The correlation between the estimated values and the observed values of Table 3.2 is shown in Fig. 3.13 for queue waiting time. In the same figure, the proposed technique of waiting time estimation is compared and found to be consistent to two other formulations used in related works: First, the formulations proposed by Akçelik et al. [28] to estimate the queue departure response time for through closely-spaced intersection sites (used by Kerper et al. [34]). Second, the queue discharge shockwave speed formulations proposed by Lighthill [29] (according to the way it is used by Cheng et al. [35], Chuang et al. [36], and also [54]).

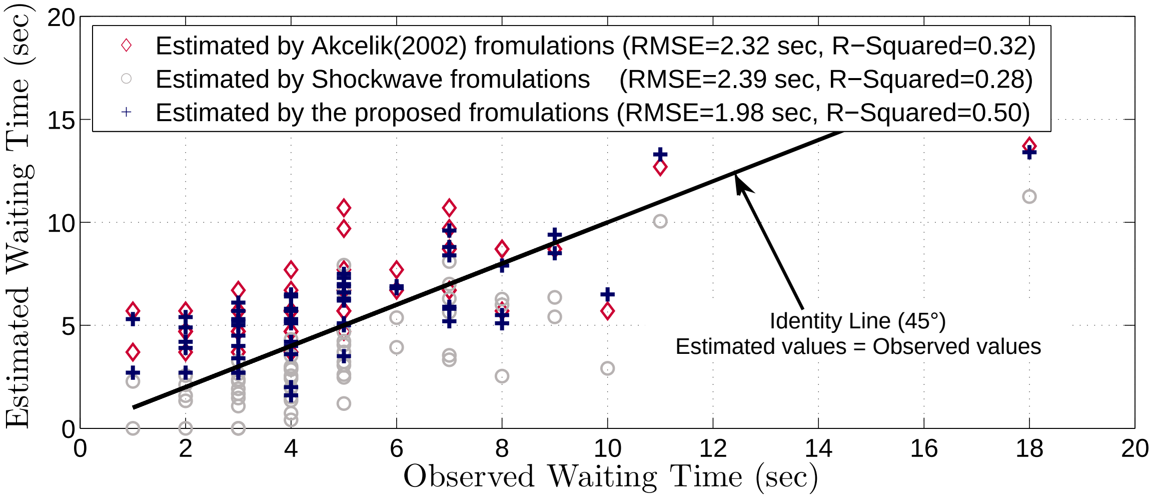
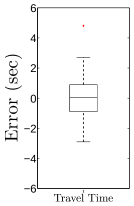
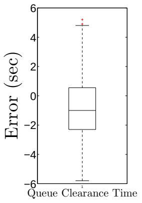
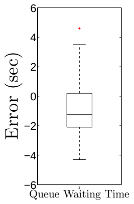


Figure 3.13: Estimated versus observed queue waiting time.

Table 3.2: Comparison Between the Collected Ground Truth (G.T.) Data and Estimations (Est.) of Travel Time, and Queue Clearance and Waiting Times for all the Intersections Combined

Position in Queue (m)	Travel Time (sec) Δt_{travel}		Clearance Time (sec) $\Delta t_{clearance}$		Waiting Time (sec) $\Delta t_{waiting}$	
	G.T.	Est.	G.T.	Est.	G.T.	Est.
0.0	3	3.9	6.0	6.6	3.0	2.7
0.0	5	3.9	6.0	6.6	1.0	2.7
0.0	4	3.9	6.0	6.6	2.0	2.7
0.0	4	3.9	7.0	6.6	3.0	2.7
2.7	6	4.5	10.0	6.6	4.0	2.0
4.9	5	5.0	9.0	6.6	4.0	1.6
7.2	4	5.4	7.0	9.9	3.0	4.5
8.0	8	6.5	13.0	9.9	5.0	3.5
8.9	7	5.7	9.0	9.9	2.0	4.2
9.0	6	5.7	10.0	9.9	4.0	4.2
9.8	5	5.9	8.0	9.9	3.0	4.0
10.1	6	6.0	10.0	9.9	4.0	4.0
10.6	8	6.0	10.0	9.9	2.0	3.9
11.2	5	6.5	8.0	9.9	3.0	3.4
12.8	7	6.4	10.0	12.1	3.0	5.7
13.1	6	6.8	10.0	12.1	4.0	5.3
14.1	5	5.6	7.0	12.1	2.0	5.4
15.2	6	6.7	7.0	12.1	1.0	5.3
15.2	6	6.7	9.0	12.1	3.0	5.3
15.8	5	6.8	8.0	12.1	3.0	5.2
16.1	6	6.9	10.0	12.1	4.0	5.2
16.5	7	6.9	12.0	12.1	5.0	5.1
16.9	9	7.0	17.0	12.1	8.0	5.1
17.0	7	7.0	11.0	12.1	4.0	5.1
17.1	6	7.2	8.0	12.1	2.0	4.9
17.4	8	7.1	13.0	12.1	5.0	5.0
17.6	6	7.1	9.0	12.1	3.0	5.0
19.0	8	7.3	12.0	13.8	4.0	6.5
19.4	5	7.3	15.0	13.8	10.0	6.5
19.5	5	7.9	9.0	13.8	4.0	5.8
20.0	15	10.2	19.0	13.8	4.0	3.6
20.4	7	7.5	12.0	13.8	5.0	6.3
21.5	7	7.6	12.0	13.8	5.0	6.2
21.8	5	7.7	8.0	13.8	3.0	6.1
22.2	11	8.6	18.0	13.8	7.0	5.2
23.5	7	8.1	11.0	13.8	4.0	5.7
23.6	9	7.9	16.0	13.8	7.0	5.9
24.7	9	8.8	14.0	15.3	5.0	6.6
26.2	7	8.5	13.0	15.3	6.0	6.9
27.0	8	8.3	13.0	15.3	5.0	7.0
28.2	7	8.4	12.0	15.3	5.0	6.9
28.8	9	8.9	13.0	15.3	4.0	6.4
31.2	8	9.4	13.0	16.9	5.0	7.5
35.8	11	10.1	17.0	16.9	6.0	6.8
36.1	11	9.9	20.0	18.3	9.0	8.5
37.6	11	9.9	18.0	18.3	7.0	8.4
40.0	11	10.5	19.0	18.3	8.0	7.9
41.8	13	12.9	21.0	18.3	8.0	5.5
42.0	11	10.2	18.0	19.8	7.0	9.6
42.4	10	10.4	19.0	19.8	9.0	9.4
44.1	13	13.2	18.0	19.8	5.0	6.6
46.7	12	11.0	19.0	19.8	7.0	8.8
52.8	17	14.3	22.0	21.3	5.0	7.3
54.0	16	17.0	23.0	22.8	7.0	5.8
67.0	13	12.4	24.0	25.7	11.0	13.3
75.0	14	13.8	32.0	27.2	18.0	13.4
Box Plot						

In addition to RMSE, Fig. 3.13 also provides the coefficient of determination R^2. The R^2 value of our proposed estimation technique is closer to 1.0 which indicates slightly better correlation with the observed values. This is achieved mainly due to taking account of incremental headways for the first few vehicles in queue as well as the downstream velocity into our proposed queue dissipation formulations. It should be emphasized that, although our obtained values for root mean square error (RMSE) and coefficient of determination (R^2) indicate more accurate estimations and better correlation with the observed values, the fact that our estimation method needs the downstream velocity (v_2) as extra information makes it difficult to conclusively claim that our queue dissipation formulation is better than the other two formulations mentioned above.

3.6 Acknowledgment

This research was sponsored by a research award from BMW Information Technology Research Center (ITRC) in Greenville, SC, USA. I am thankful for the support provided by Mr. Andre Luckow from BMW ITRC. I also thank Mr. Nainfeng Wan who traveled and collected the travel trajectories of public buses in the city of San Francisco.

Chapter 4

Back-End System Architecture

4.1 Introduction

Many in-vehicle applications have the potential to benefit from Signal Phase and Timing (SPaT) data in order to achieve better fuel efficiency, emission control, and safety features [6, 55, 56, 57]. The Velocity Advisory Systems [6, 55], and Start/Stop systems [56] are such applications with fuel efficiency benefits reported in [58, 59, 60]. Also a Collision Avoidance System [57] can benefit from SPaT information in order to foresee potential signal violations at signalized intersections. In addition to in-vehicle applications, there are also many arterial performance measurement methods that use SPaT as their input [46, 47].

The main challenges in providing real time SPaT to aforementioned in-vehicle applications are in first finding an inexpensive and reliable data communication technology, and second structuring SPaT information from various and disparate data sources in standard and uniform formats. In this chapter, we explain the implemented system architecture of how we get SPaT information from different data sources and communicate that to a connected vehicle. The whole system was developed at BMW Group Technology Office USA in Mountain View, CA; however, the contribution of this dissertation is delivering the bus-crowdsourced SPaT data to the connected test vehicle.

This chapter is also a sequel to the previous two chapters as it provides an in-depth overview of the back-end implementation of the crowdsourcing algorithms as well as the whole system developed at BMW Group Technology Office USA in Mountain View, CA. At first, Section 4.2 explains the hardware architecture of the back-end system which provides the SPaT data to connected vehicles through disparate data sources. Section 4.3 describes the software techniques of delivery of the SPaT information to a connected test vehicle through I2V communications. Section 4.4 and 4.5 explain the computational back-end nodes that specifically process the incoming crowdsourced data, and estimate SpaT information. The last section explains all the verification tools that were developed to collect ground truth data and compare it to our estimations.

4.2 System Overview

Ideally, a connected vehicle receives upcoming SPaT information from Traffic Management Centers (TMC) or directly from the intersection using Infrastructure-to-Vehicle (I2V) communication. However, for those signals which are not connected, SPaT estimation prediction is used based on crowdsourced data. For this reason, a prototype system and a test vehicle as demonstrated in Figure 4.1 were developed at the BMW Technology Office USA that is capable of building prediction from crowdsourced GPS traces as well as crowdsourced in-vehicle camera data.

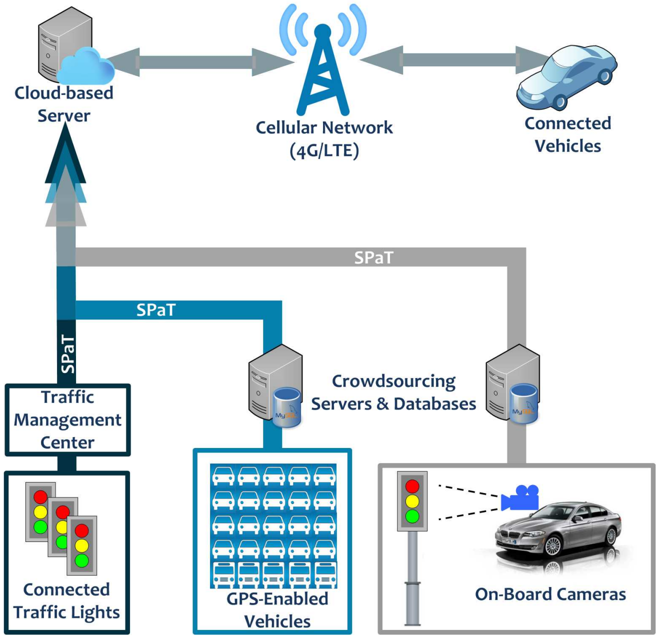


Figure 4.1: System overview of the path of data from data sources to server to a connected vehicle.

The test vehicle contains a MobilEye camera that contains an image processing system (EYE-Q) capable of detecting traffic signals [61]. From the data collected by the cameras, a database of traffic signal locations and statuses are built and stored in a database. It must be emphasized that the main contribution of the first part of the dissertation is SPaT predictions from crowdsourced GPS traces; and as provided in Section 4.6, the in-vehicle cameras were used here only to verify the accuracy of these predictions.

4.3 Software Architecture

Figure 4.2 demonstrates the software architecture of the implemented system. The inputs of the system are the different data sources, described in previous section. The output is a collection of SPaT information which can be sent to any in-vehicle application of connected vehicles that seeks SPaT information. The following subsections will describe Figure 4.2 including how the system interact with the disparate data sources and multiple connected vehicles.

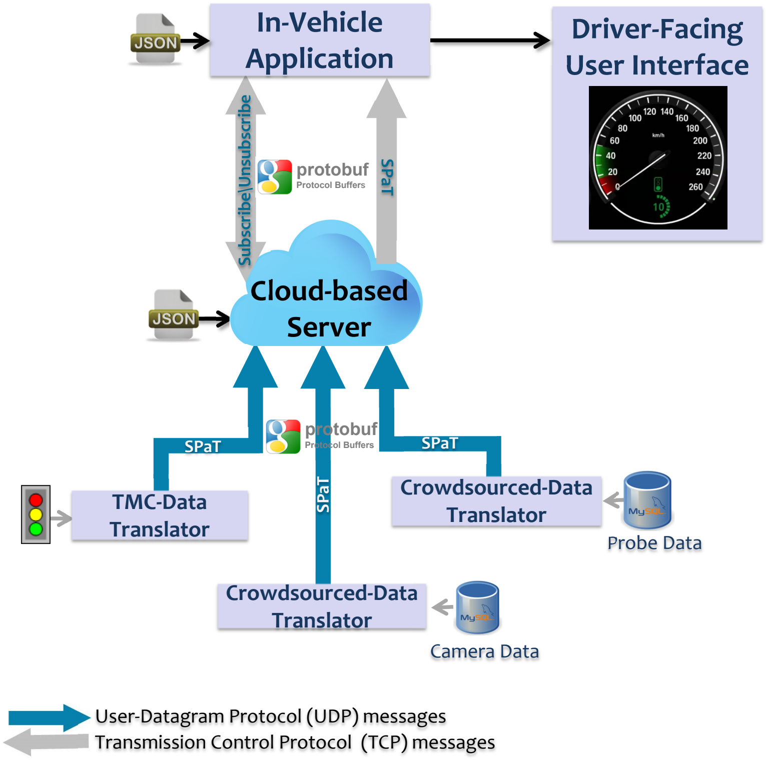


Figure 4.2: The software architecture of the traffic signal state communication through a cloud-based server.

4.3.1 Data Translation

As shown in Figure 4.2, the I2V communication happens through the cloud-based server. The cloud is actually the link between the connected vehicles and the data sources. In order to keep the design of the cloud-based server consistent, data translation is used for importing data from disparate data sources and then transmitting in one single format. In our application, the data translators shown in Figure 4.2 import data from the TMC and crowdsourced data sources and transmit the data to the cloud in a data structure represented by the Protocol Buffer (protobuf) data formats [62]. As claimed by Google and as evaluated in [63], the Protocol

Buffer leads to fast data transfer over the web comparing to eXtensible Markup Language or XML [39]. This is mainly because the Protocol Buffer uses binary format to serialize structured data.

Every time a new phase-change is predicted for an Intersection-Phase, the data translator serializes the phase-change information into the binary Protocol Buffer structure and sends the information to the cloud-based server through a User-Datagram Protocol or UDP [64] unconnected datagram sockets.

Although the delivery is not guaranteed using UDP, this is preferable to Transmission Control Protocol or TCP [65] in our application. The TCP is actually slower for sending the phase-change updates of all the intersections; and the cloud-based server cannot handle receiving all these updates in TCP. However, if the signal state communicating system is being used for a safety-focused driver assistance system such as a redlight-runner predictor, the communication technologies could be changed to ensure the vehicles receipt of all safety messages.

4.3.2 Vehicle Subscription

While the connected vehicles are traveling, the embedded in-vehicle computing devices search among the list of the intersections within a specified range of the vehicles, and they identify the most relevant Intersection-Phase to the vehicles movements. As demonstrated in Figure 4.2, the JavaScript Object Notation (JSON) [66] is used to represent this list which contains the attributes of each Intersection-Phase such as GPS location of the traffic light, entry and exit headings, and city zip code.

Afterwards, the connected vehicles send subscription requests for the identified Intersection-Phase to the cloud, as shown in Figure 4.2. An unsubscribe message is later sent from the vehicles to the cloud at the time the vehicles pass the previously relevant intersections. During the subscription period, the relevant Intersection-Phase updates are forwarded by the cloud to the connected vehicles every time an update is available.

4.4 Crowdsourcing Engine

This section presents the general back-end mechanisms for crowdsourcing GPS traces. The output of the crowdsourcing engine consists of two parts: First is traffic signal baseline timing that includes cycle time, phase lengths (red and green intervals), and signal offset changes. Second is phase-change (sync) data, that is green-initiation or start-of-green. Figure 4.3 demonstrates the mechanisms that the Crowdsourcing Server uses in back-end to predict and estimate this collection of traffic signal information. After being

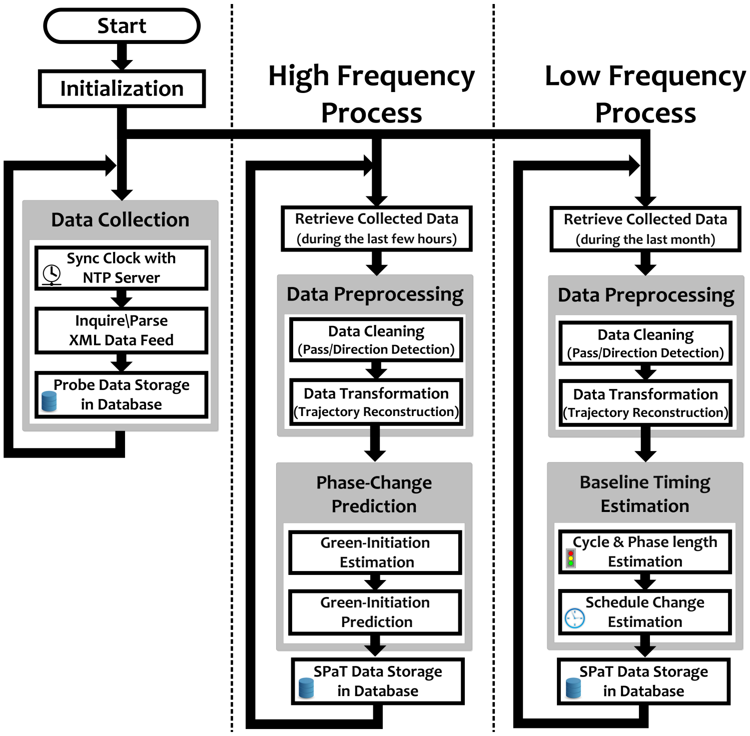

Start
Initialization
High Frequency Process
Low Frequency Process
Data Collection
Sync Clock with NTP Server
Inquire\Parse XML Data Feed
Probe Data Storage in Database
Retrieve Collected Data (during the last few hours)
Data Preprocessing
Data Cleaning (Pass/Direction Detection)
Data Transformation (Trajectory Reconstruction)
Phase-Change Prediction
Green-Initiation Estimation
Green-Initiation Prediction
SPaT Data Storage in Database
Retrieve Collected Data (during the last month)
Data Preprocessing
Data Cleaning (Pass/Direction Detection)
Data Transformation (Trajectory Reconstruction)
Baseline Timing Estimation
Cycle & Phase length Estimation
Schedule Change Estimation
SPaT Data Storage in Database

Figure 4.3: The functional architecture of the Crowdsourcing Server.

initialized, the Crowdsourcing Server goes through three processes which are separated by dashed lines in Figure 4.3:

- **Data Collection**, in which probe vehicle data is continuously collected and stored in the MySQL database.
- **High-Frequency Process (Phase-Change Estimation/ Prediction)**, in which only the green-initiations are predicted with high frequency. In fact, the green-initiation prediction is the process of predicting the next transition to green; and because of the clock drift of a traffic signal throughout a day, the next green-initiations should be continuously predicted based on the most recent probe data. In our application, every time the execution of this process cycle begins, the most updated probe data collected during the last few hours is first retrieved from the MySQL database, as shown in Figure 4.3. After preprocessing this data, the green-initiations of each Intersection-Phase are predicted and finally stored in the MySQL database.
- **Low-Frequency Process (Baseline Timing Estimation)**, in which the traffic signal baseline timings are estimated. Traffic Signals are typically re-timed infrequently; as a result, this process needs to be executed with very low frequency (once per month in our application).

4.5 Crowdsourcing Methodologies

This section presents the algorithms applied in the crowdsourcing processes of previous section. The basic steps, named in Figure 4.3 as Initialization, Data Collection, and Data Preprocessing, are described. All other algorithms involved in the SPaT estimations/ predictions were previously explained along with results in Chapter 2 and Chapter 3.

4.5.1 Initialization

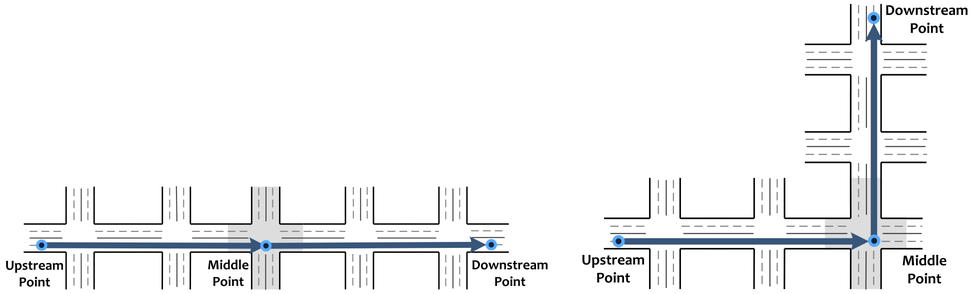
The Crowdsourcing Server initializes its objects and variables once it's fired up. The initial parameters are predefined by either the web user or the administrative user of the server. The objects such as MySQL database, intersection-phase geometry, and the probe vehicle average acceleration are initialized by the initialization function as follows:

- **MySQL database**: The connection to the database is established using Java Database Connectivity Driver (JDBC) [67] for MySQL; and the tables are created to store data sets.

- **Intersection-Phase**: The geometry of the desired intersection-phase pairs should be predefined by either the web user or the server administrator. In fact, the coordinates of three points should be defined by the user in order to define and initialize every Intersection-Phase pair. The three-point definition method covers all the possible movements at intersections. This definition includes one point on the upstream, one point on the downstream, and a middle point at the intersection center. As an example, Figure 4.4 (a) and (b) show through movement and left turn definitions respectively. Note that with additional algorithms, not described in this paper, it would be possible to automatically crowdsource the geometry instead of manually defining it.

  In fact, a path is constructed by the aforementioned points. This path is supposed to be as close as possible to the path that would be traveled by each vehicle in real world. As a result, the user should assign the points in such a way that the error between the estimated and actual paths is minimized. It is obvious that the wider the street is, the higher the imposed error would be.

- **Vehicle**: The general characteristics of the probe vehicles should be defined and initialized for crowdsourcing and plotting reasons. The estimated acceleration/deceleration given in Section 2.4 are some of the characteristics of the buses at the San Francisco Muni (sf-muni); however, they may be applicable to other agencies if similar buses are utilized.



(a) (b)

Figure 4.4: Three-Point definition of an Intersection-Phase: (a) Through movement. (b) Left turn. (The desired intersection is shown in shaded color)

4.5.2 Data Collection

A public feed of bus location and velocity data in the city of San Francisco is used here to crowd-source the traffic signal information. The feed is provided by NextBus Incorporated through eXtensible Markup Language or XML [39] which can be accessed using URLs with parameters specified in the query string [16]. Each vehicle (bus) sends a probe update every 200 meters approximately or 90 seconds, whichever comes first [25]. As shown in Figure 4.3, the Data Collection process periodically inquires the XML feed data of each route. It is crucial to set this process in such a way that its clock is automatically synchronized to a Network Time Protocol (NTP) server; in this work the clock is synchronized with the NIST time server [45] every 10 minutes.

The interval between successive XML inquiries have to be less than the minimum interval between successive probe updates; otherwise some updates sent by probe vehicles would be missed. Nevertheless, a high inquiry frequency should be avoided so as not to violate the restrictions on data usage.

4.5.3 Data Preprocessing

The first step in data preprocessing is Data Cleaning which consists of identifying the useful probe data to be mined. The second step prepares the probe data for possible use in SPaT estimations; this step is named Data Transformation and actually transforms the single probe updates to travel trajectories which are desirable for SPaT estimation purposes. These two steps can be found in Figure 4.3 and are described as follows:

A. Data Cleaning

The Data Cleaning in this manuscript refers to the process of identifying the desired probe data out of the collected probe data. This process consists of: (1) identifying the probe vehicle reports that are within the three-point definition, (2) detecting and separating each pass of each vehicle, and finally (3) discarding the vehicle passes not on the desired direction.

As described in Subsection 4.5.1, each three-point definition includes upstream and downstream parts. Here, we are interested in identifying the probe vehicle reports that have happened within either the upstream or downstream part. Figure 4.5 shows the upstream part as an example; where $d_{upstream}$ is the distance of a probe report to the upstream point, d_{middle} is the distance of the same probe report to the middle point, and $L_{upstream}$ is the length of the upstream part. It is obvious that if the condition $L_{upstream} = d_{upstream} + d_{middle}$ is satisfied then it can be concluded that the vehicle had sent the location report exactly on the straight line between the upstream and middle points. However, it is less likely for the vehicle to be exactly on the straight line between the two points, especially on wide streets. Because of this and the inevitable error in GPS position reports, the following conditions are verified instead:

$$\begin{aligned} d_{upstream} + d_{middle} &< L_{upstream} + \Delta L \\ d_{upstream} &< L_{upstream} + \Delta L \\ d_{middle} &< L_{upstream} + \Delta L \end{aligned} \tag{4.1}$$

where ΔL is a small value added to account for the street width as well as for the errors in the probe vehicle reports. A similar approach is used to verify whether a probe report is within the downstream part or not.

Finally, the distinct passes of vehicles are detectable due to the fact that each probe vehicle report is labeled with a vehicle ID number. And the direction of a distinct vehicle pass is detectable by inspecting the distance between the probe location reports and the upstream point of the intended Intersection-Phase.

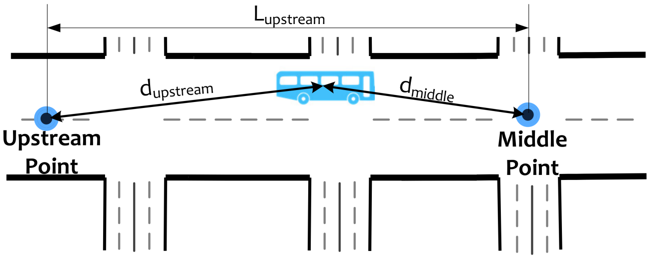


Figure 4.5: A probe location report that has been sent within the upstream part of an Intersection-Phase.

B. Data Transformation

As explained in Section 2.3 and 3.2, there should be sufficient probe data points in an identified vehicle pass for SPaT estimations. But because the utilized probe data is sparse and the consecutive data points of each pass are far away from each other, we need to approximate a vehicle trajectory between each two probe reports. This is actually a data transformation process where low frequency probe data are transformed and consolidated into vehicle trajectories. The following steps are executed in the back-end system for this purpose:

- **Step 1:** For a given intersection, we first select bus passes that have update points within a given interval before and after that intersection (in other words, within the three-point defined in Figure 4.4). For example for the Clement Intersection shown in Figure 2.2, we select bus passes that updated in [480m, 780m] position interval.

- **Step 2:** To determine if a bus stopped at an intersection, we propose to approximate the intersection delay, t_d, by the previously given Equation 2.14.

  If $t_d \leqslant 0$, we postulate that the bus had no delay and that it passed the intersection during a green interval. Otherwise, we may attribute the delay to a stop at red, which will be further confirmed in the next step.

- **Step 3:** When $t_d > 0$, we check the consistency of the trajectory shown in Figure 2.4 and Figure 3.1(a-b) with data. Using the trapezoidal geometry of the curves, we can estimate the time a bus comes to a stop t_{stop} and the time the bus leaves the intersection t_{start} as previously given in Equation 2.2 and Equation 2.1, respectively. Obviously if $t_{stop} > t_{start}$, the postulated trajectory is invalid and the associated bus pass will be discarded. When $t_{stop} \leqslant t_{start}$, we accept the trajectory as valid and estimate that the bus came to a full stop at a red light.

4.6 Ground Truth Verification Tools

Generally, the following tools were utilized to verify the SPaT estimations/ predictions as well as the proposed queue formulations based on the collected ground truth data.

4.6.1 Web-based Verification

The web-based verification tool not only enables us to do comparisons locally at the intersections but also to have an estimate of the light's actual state when we are not present at the intersections. For this reason, as previously shown in Figure 1.4, a web interface with PHP interpreter was implemented with the following features:

- **On-site Synchronization**: The user has the option to store a sample of green-initiation for a desired intersection by simply clicking on a button on web-page once the light turns green. Using the stored sample of green-initiation, the state of the traffic signal is demonstrated by a countdown counter (the left counter of Figure 4.6(a)). If the drifting of traffic light can be neglected then this counter can be used as a measure of the light's actual state.

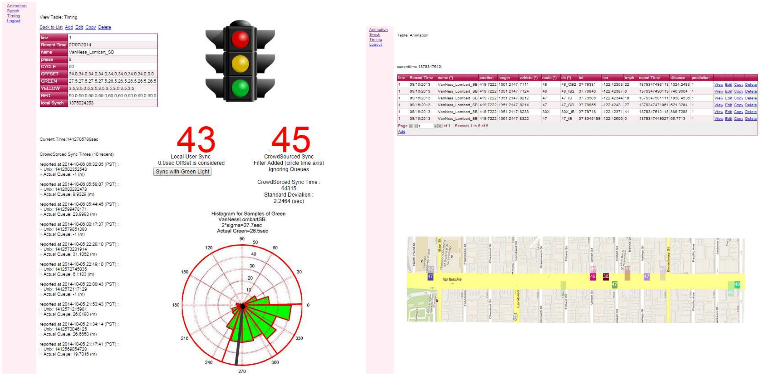
- **Visualizing Crowdsourced Data**: Using the web interface, each intersection can be monitored through the $Intersection-Dashboard$ which demonstrates the crowdsourced SPaT data as shown in Figure 4.6(a). The right countdown counter of this figure demonstrates the estimated time remaining to the end of each signal (green, yellow, or red). In order to do the countdown the web server connects to an SQL database to read the most updated predicted green-initiations as well as the baseline timing of the desired Intersection-Phase.

  In addition to the countdown counters, the Probability of Green, extracted in Section 2.9, is also demonstrated in Figure 4.6(a) through a polar histogram. The polar histogram has a smooth running second clock-hand which demonstrates the current state of the traffic light by moving counter clock wise.

- **Animation**: The web server parses the previously stored XML updates and then as shown in Figure 4.6(b), animates each probe vehicle (bus) according to its direction, position and velocity along a sample segment of Van Ness street in San Francisco. In this way, the accuracy of the parsed position and velocity of each XML update was verified during on-site ground truth tests.

4.6.2 Verification of Queue Formulations

The queue formulations proposed in Section 3.4 were verified by riding in buses and recording the trajectory at high frequency using a handheld Garmin GPS receiver. The verification results can be found in Section 3.5.2.



(a)

View Table: Timing

line	1
Record Time	07/07/2014
name	VanNess_Lombart_SB
phase	8
CYCLE	90
OFFSET	34.0,34.0,34.0,34.0,34.0,34.0,34.0,34.0,0.0
GREEN	27.5,27.5,27.5,27.5,26.5,26.5,26.5,26.5,26.5
YELLOW	3.5,3.5,3.5,3.5,3.5,3.5,3.5,3.5,3.5
RED	59.0,59.0,59.0,59.0,60.0,60.0,60.0,60.0,60.0
local Synch	1375024203

Current Time:1412705788sec

CrowdSorced Sync Times (10 recent):

reported at 2014-10-06 06:32:05 (PST) :
+ Unix: 1412602352543
+ Actual Queue: -1 (m)

reported at 2014-10-06 05:58:07 (PST) :
+ Unix: 1412600282478
+ Actual Queue: 9.9329 (m)

reported at 2014-10-06 05:44:45 (PST) :
+ Unix: 1412599478171
+ Actual Queue: 23.9993 (m)

reported at 2014-10-06 00:17:37 (PST) :
+ Unix: 1412579861383
+ Actual Queue: -1 (m)

reported at 2014-10-05 22:28:10 (PST) :
+ Unix: 1412573281914
+ Actual Queue: 31.1062 (m)

reported at 2014-10-05 22:19:10 (PST) :
+ Unix: 1412572745035
+ Actual Queue: 6.1183 (m)

reported at 2014-10-05 22:08:43 (PST) :
+ Unix: 1412572117129
+ Actual Queue: -1 (m)

reported at 2014-10-05 21:53:43 (PST) :
+ Unix: 1412571215981
+ Actual Queue: 26.8198 (m)

reported at 2014-10-05 21:34:14 (PST) :
+ Unix: 1412570046125
+ Actual Queue: 26.6658 (m)

reported at 2014-10-05 21:17:41 (PST) :
+ Unix: 1412569054728
+ Actual Queue: 19.7016 (m)

43
Local User Sync
0.0sec OffSet is considered

45
CrowdSourced Sync
Filter Added (circle time axis)
Ignoring Queues

CrowdSorced Sync Time :
64315
Standard Deviation :
2.2464 (sec)

(b)

Table: Animation

currenttime 1379347512;

line	Record Time	name (*)	position	length	vehicle (*)	route (*)	dir (*)	lat	lon	kmph	report Time	distance	prediction				
1	09/16/2013	VanNess_Lombart_SB	419.7222	1351.2147	7111	49	49_OB2	37.79331	-122.42303	22	1379347493113	1324.2483	1	View	Edit	Copy	Delete
1	09/16/2013	VanNess_Lombart_SB	419.7222	1351.2147	7124	49	49_IB2	37.79846	-122.42387	0	1379347498113	746.9664	1	View	Edit	Copy	Delete
1	09/16/2013	VanNess_Lombart_SB	419.7222	1351.2147	8212	47	47_IB	37.79586	-122.42344	18	1379347501111	1038.4536	1	View	Edit	Copy	Delete
1	09/16/2013	VanNess_Lombart_SB	419.7222	1351.2147	8214	47	47_OB	37.79955	-122.4243	27	1379347471061	621.3284	1	View	Edit	Copy	Delete
1	09/16/2013	VanNess_Lombart_SB	419.7222	1351.2147	8233	30X	30X_IB1	37.79718	-122.42371	41	1379347512118	889.7298	1	View	Edit	Copy	Delete
1	09/16/2013	VanNess_Lombart_SB	419.7222	1351.2147	8322	47	47_IB	37.8045199	-122.42506	0	1379347449027	65.7713	1	View	Edit	Copy	Delete

Page 1 of 1 Records 1 to 6 of 6
Add

Figure 4.6: The features of the web interface: (a) Monitoring via *Intersection − Dashboard* (b) Animating the probe vehicles

4.6.3 Green-Initiation Verification

- **Verification by Computer**: As it was previously explained in Section 2.10 and Section 3.5.1, in order to verify the accuracy of green-initiation predictions, we collected the actual green-initiations locally at a sample intersection by a computer program that would log the time whenever the observer pressed a key at the change of red to green. The program was synchronized to the NIST time server [45].

- **Verification by Camera**: The prediction quality of the bus-crowdsourced phase-change (green-initiation) information was verified using the on-board camera data which was collected during on-site ground truth tests. The camera only records the status of the traffic light; as a result the collected data is processed to find the signal changes and their timestamps. A comparison between detected phase-change data along the test route in San Francisco using the MobilEye camera [61] and our bus-crowdsourced phase-change predictions is reported in Table 4.1. From the camera recorded data we can calculate phase-changes to within $<$ 100ms accuracy, as a result of the frequency of camera/CAN-BUS updates, and the implementation of Network Time Protocol (NTP) synchronization on the recording computer to ensure that timestamps are valid.

  The Root Mean Square Error (RMSE) between the predicted phase-changes and the camera-detected phase-changes was 2.4 seconds. The average error is 0.98 seconds and, as shown in the box plot of the error, given in Table 4.1, the median error was 0.63 seconds.

  It is obvious that concurrent with running the laps in the test field, the crowdsourcing server is also collecting the real-time probe data and predicting the phase-changes (green-initiations) using the average of multiple observed green-initiations (please see Section 2.7 and 3.3.3). However, the average green-initiation is not necessarily relevant to the camera-detected green-initiation. In fact, the last updated green-initiation average can be few minutes to couple hours before the camera-detected one, depending on the number of the probe vehicles (buses) and the likelihood that their travel trajectories are fitted to the desirable trajectory (Figure 3.1). This time-difference between the last bus-crowdsourced and the camera-detected green-initiations is denoted as Δt as follows:

$$\Delta t = t_{SoG,cam} - \bar{t}_{SoG} \tag{4.2}$$

where $\bar{t}_{SoG}$ and $t_{SoG,cam}$ are the bus-crowdsourced and camera-detected green-initiations respectively both in Unix-Time. However, because of the cyclic periodicity in fixed-time lights, the actual error between the last bus-crowdsourced green-initiation and the camera-detected green-initiation, as given in Table 4.1, is:

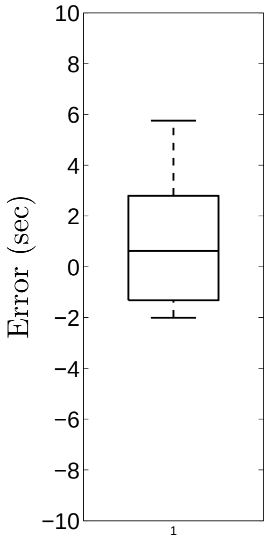
$$Error = \Delta t - \mathrm{round}(\Delta t/C)C \tag{4.3}$$

where C is the cycle time, and the function round$(.)$ rounds its argument to the nearest integer.

4.7 Acknowledgment

This work was performed in collaboration with, and supported by, BMW Group Technology Office USA, Mountain View, CA, USA. The author thank Mr. Andreas Winckler and Mr. M. Filusch of BMW Group Technology Office, USA in Mountain View, California and Mr. Grant Mahler from Clemson University for developing the in-vehicle system and the vehicle subscriptions system. Mr. Grant Mahler who spent long hours setting up the in-vehicle camera and collecting the ground truth data was invaluable to the work presented here.

Table 4.1: Difference in seconds between the camera-detected phase-changes and the bus-crowdsourced phase-change predictions.

Intersection at VanNess	Error (sec)	Box Plot
Greenwich	-1.22	
	-1.36	
	-1.53	
	-1.42	
	-1.41	
	-0.25	
Filbert	2.67	
	2.50	
	3.08	
	2.82	
	2.72	
	5.75	
	2.75	
	3.00	
Union	0.63	
Broadway	-2.01	
Pacific	-0.76	
Jackson	-0.44	
Washington	3.12	

Chapter 5

Conclusions for Part I

In the first part of this dissertation we demonstrated the feasibility of estimating timing of pre-timed traffic lights by observing statistical patterns in sparse probe vehicle data feeds. In particular we showed, for example intersections in the city of San Francisco, the feasibility of estimating cycle time, red time, start of green, and signal schedule change. This was achieved without directly estimating the queue lengths and despite traffic influence. Extensive use of data filtering/ pre-processing is elemental to the successes found at the given intersections. The obtained SPaT information is ultimately fed into an in-vehicle computing device.

A complementary approach to estimating SPaT from probe data is also proposed in this dissertation to include the influence of queue delay. In case of a low-frequency data source, less challenge is expected in the estimation procedure if we only use the probe data that can be fitted into the predefined desired trajectories. Also if we identify and remove the probe data that appear to be influenced by heavy traffic then the more complex queue formulations are not needed in estimations. However, these measures eliminate a large portion of data; and any SPaT estimation method that filters out huge amount of data is subject to error. This is mainly due to the signal clock drift throughout a day that makes it crucial to have recent SPaT.

It is also shown that adding the trajectories that have been influenced by queue delay allows us to access a larger portion of data. This is the reason that the phase-change estimation results remain accurate even during heavy traffic. Also as it was expected, more accurate baseline timing estimations are achieved if we use the trajectories that include at least one report sent while waiting in queue.

Part II

Arterial Traffic Signal Optimization with Connected Vehicles

Chapter 6

Vehicle-Intersection Coordination under the Connected Vehicles Environment

6.1 Introduction

The coordination and optimal timing of traffic signals are by nature complex problems and backed by years of research in traffic engineering and operations research. Current signal timings are mostly scheduled offline using sophisticated software packages; the optimized timings are then deployed as fix timetables for different times of the day. Many signals are actuated by traffic and have rules to override their pre-optimized timetables based on the state of their loop-detectors to reduce idling at intersections. While traffic responsive control strategies such as SCOOT [68] and SCATS [69] calculate their timing in real-time [70], they act based on the immediate state of loop-detectors. Unfortunately, in these systems the light triggers to green only when an idling queue of vehicles is already formed over its loop-detector. Furthermore, traditional detectors mostly rely on point detection which provide a very limited information such as passage of a vehicle at a fixed location [18, 19].

Smart traffic signal controllers will do more than just signaling right of ways and act intelligently as hubs that sense, route, and harmonize the flow of arterial traffic. More recent research has focused on signal to vehicle communication for improving efficiency by providing speed advisory to individual vehicles [6, 32]. In addition, the two-way communication, under the connected vehicle environment, allows the geographical data (positions, headings, and speeds) of the connected vehicles to be also wirelessly transmitted in real-time to smart traffic signal controllers [17]. Taking advantage of the two-way communication capabilities between the smart traffic signal controllers and the connected vehicles, this section presents the development of an algorithm for vehicle-intersection coordination system that not only can adjust the traffic signal timing based on the prevailing traffic conditions, but also can guide the connected vehicles for a timely arrival at intersection. The proposed algorithm is presented for an autonomous driving environment (100% penetration rate of equipped vehicles) at an intersection with no traffic lights; however, as a future research direction, it might be possible to modify the proposed algorithm to be applied to a mixed traffic consisting of autonomous-controlled and human-controlled vehicles. In a mixed traffic environment, a physical traffic light is needed, and a minimum green time needs to be considered because the signal cannot rapidly change the direction of priority without considering the reaction times of drivers [23].

In recent years, there have been a great deal of attention paid to intersection control under the connected vehicle environment. Towards the most related work, Raravi et al. [71] determined the merge sequence in which vehicles cross the intersection region by formulating an optimization problem with constraints to ensure safety. The key difference between our work and [71] is that the formulations proposed in [71] are nonlinear, and, as a result, Matlab optimization toolbox *fmincon* is used which may only give local solutions [72] and does not guarantee a global optimum [71]. In addition to this, Lee et al. [73] also employed optimal control for a cooperative vehicle intersection control. In this work, the trajectories of any two conflicting autonomous vehicles are modified to optimize an objective function: minimize the overlap of trajectories in the intersection area. This would not always be a collision-free solution, and would not always provide a feasible solution because of the complexity of the optimization formulations (the objective function and constraints are nonlinear and non-convex) [23, 74]. For this reason, [73] first tries active-set algorithm, and, if it fails, the interior point algorithm is tried next, and, finally, if both fails then genetic algorithm is employed which all adds to the complexity of the solution.

Some other works such as [75, 76] used job-scheduling techniques [77] where the intersection is considered a machine. Colombo et al. [75] view the time interval that each vehicle spends in the intersection as the length of the job to be executed on the machine. Similarly, Xie et al. [76] view clusters in the aggregate flow representation of different routes as the jobs on the machine; where clusters are a basic representation of a vehicle or group of vehicles [78]. They use an approximate dynamic programing [79, 80] procedure, called Controlled Optimization of Phases [81], to obtain a near optimal solution. Also Ahn et al. [82] translated the intersection collision avoidance to a job-shop scheduling problem assuming first-order dynamics for the vehicles. In the field of multi-agent systems (driver agents and intersection manager agents), the solution provided by Dresner et al. [83, 20, 84] is based on a reservation paradigm which allows the vehicles to reserve a block in space-time in the intersection. The solution is not optimal in the sense that it is based on the First Come, First Serve methodology, and a reservation is rejected if any part of the requested space-time block has been previously reserved or occupied by another vehicle. In addition, this reservation-based system may have some safety flaws, mainly because it relies on an extremely advanced and precise automated driving system [85]. A detailed review of cooperative intersection management systems can be found in [86].

The second part of this dissertation formulates the vehicle-intersection coordination problem as a mixed-integer linear programming (MILP) problem to coordinate the vehicle arrivals and the status of the virtual traffic signals [87]. The proposed MILP-based controller receives the probe vehicles' online information and gives each vehicle the advice of an optimal time to access the intersection. The controller assigns the access times to vehicles minimizing the stopped delay and ensuring no crashes occur at the intersection while considering the travel velocities that are desirable to the vehicles. The problem is formulated as a MILP for which commercial and open-source solvers are available. MILP has been used in various path planning applications with collision avoidance, such as airplanes and spacecrafts [88, 89, 90, 91] and autonomous vehicles [92]. In the field of vehicle-intersection coordination, Zhu et al. [74] used a lane-based traffic flow model to optimize the total travel time; however, the objective function does not consider the travel times and velocities that are desirable to the individual vehicles. The output solution of the optimization problem is the traffic flow from one lane to another lane through the conflict point model. Although it is mentioned that this output solution can be viewed as the time schedule that the vehicle moves from one end to the other end of the intersection, the vehicles' capability to make the time schedule on time is not specifically addressed by the authors in [74]. No microsimulations are provided, and only numerical case studies are provided to demonstrate the effectiveness of the proposed algorithm. This dissertation, however, provides microsimulations and real test vehicle interacting with the intersection control cyber-layer to specifically see if the connected vehicles can make the intersection controller's advice on time.

Also in the area of phase and timing optimization for standard two-phase or eight-phase controller, a set of Mixed Integer Linear Programming (MILP) formulations have been proposed in the literature. Most of these formulations can be applied to a mixed traffic consisting of autonomous-controlled and human-controlled vehicles; however, they either use off-line historical traffic data or they assume a one-way communication where the connected vehicles only report information to an intersection controller but their travel trajectory cannot be controlled. As an example, He et al. [93] used probe vehicles' on-line information to identify pseudo-platoons and found an optimal signal plan using MILP. Early works, however, have used the average flow on each link of intersections as the input to their optimization; for example, Gartner et al. [94] formulated a MILP-based delay minimization problem to determine the parameters of two-phase traffic signals including cycle time, green splits, and offsets in a signal-controlled street network. Little's MILP formulation [95], and a recent work in [96], solves the bandwidth maximization problem, and assigns optimal offsets to the standard traffic signals in a two-way arterial. Other works in this area use Model predictive control for traffic signal control problem and formulate it as a MILP problem [97, 98, 99].

As it was previously mentioned in this section, because each proposed intersection control unit communicates with approaching vehicles, it can act much more effectively than even the advanced intersection control systems that currently exist. Moreover, by encouraging platoon formations, the intersection throughput is expected to significantly increase. We propose a novel intersection control scheme at the cyber-layer to facilitate uninterrupted intersection passage and encourage platoon formation. We were encouraged by the recent simulation results presented in [100, 22] indicating the more than doubling of capacity and flow in an urban network if vehicles pass intersections in platoons. Simulations show benefits of such systems greatly increase if vehicles move in platoons, in certain cases doubling the arterial network capacity with the coordination of platoons and intersections [100]. In our proposed research, the objective of increasing intersection throughput will be formalized as an optimization problem. The optimization goal is to find the sequence and times of arrival for each vehicle such that the expected arrival times of the last subscribed vehicle in a given time window is minimized and any potential collisions is also prevented. This objective will maximize the number of vehicles that clear the intersection in a given time. Furthermore, we incorporate the desired arrival time of the vehicles into the optimization problem in such a way that vehicles would not face extreme delay or expedition compared to their desired arrival times. Our optimization method encourages platooning although it is not explicitly incorporated into our formulations.

In this chapter, first, the vehicle-intersection coordination problem is stated in simple language and a solution to an example problem is provided in Section 6.2. The notations used in the second part of the dissertation are explained in Section 6.3. Our proposed formulations for the problem and the methods to identify and handle their disjunctions are provided in Sections 6.4 and 6.5, respectively. Finally, in Section 6.6, the proposed formulations are solved by MILP and a simplified case study problem is solved only to see an example numerical output of our formulations. In Chapters 7-9, we will further evaluate our approaches via microsimulations and with real vehicles interacting with the intersection control cyber-layer and a virtual road network environment.

6.2 Problem Statement

In this section, we seek an intersection controller that harmonizes the flow of the approaching Connected Vehicles (CVs). Taking advantage of the two-way communication capabilities between the smart intersection controllers and the CVs, the controller not only can adjust the traffic signal status based on the prevailing traffic conditions, but also can guide the connected vehicles for a timely arrival at intersection. For this reason, we will set up computational servers that process connected vehicle instantaneous data and create a live picture of evolving traffic conditions. Advanced and fast optimization algorithms then optimally schedule the intersection access-times for the vehicles and adjust the status of the virtual traffic signal. Finally, the scheduled access-times (arrival-times) are sent to all approaching CVs so that they can adjusts their speed accordingly. However, the question is still open as to how to find appropriate access-times to be assigned to the vehicles in the context of traffic flow, and safety.

Figure 6.1 demonstrates a sample scheduling problem at a signalized intersection. Ignoring all the turns for simplicity's sake, we assume a two-phase/four-movement intersection. As shown in Figure 6.1(a), Phase X corresponds to a set of two traffic movements: (1) south-bound denoted as dark letter **X** or $\mathbf{X}'$ in text; (2) north-bound denoted as light letter X or X$''$ in text. Similarly, Phase O corresponds to a set of two traffic movements: (1) west-bound denoted as dark letter **O** or $\mathbf{O}'$ in text; (2) east-bound denoted as light letter O or O$''$ in text. The *y*-axis of Figure 6.1(b) indicates the remaining distance from the current position of vehicles to the intersection access point (stop position). Please note that the *y*-axis values are actually the remaining distances on four different movements which are projected on one single axis. As shown on the *y*-axis, it is assumed that at the starting time, five connected vehicles denoted by cv_2, cv_4, cv_5, cv_6, and cv_8 are traveling on Phase X and four vehicles cv_1, cv_3, cv_7, and cv_9 are traveling on Phase O. An intersection controller is sought in this chapter of the dissertation that is capable of scheduling the vehicle arrivals as shown on the horizontal axis of Figure 6.1(b), as an example. The provided sample solution assigns virtual green splits to Phase X and O (see green/red timings in Figure 6.1(b)) in such a way that all the vehicles are served without stopping at red. The solution assumes that:

- cv_2 can slow down to pass the intersection with the southbound platoon on Phase X.
- cv_3 can speed-up so the southbound platoon on Phase X does not slow down or stop at red.
- cv_8 can speed-up to catch up with the green light on Phase X.
- cv_7 and cv_9 adjusts their speed to build a westbound platoon on Phase O.

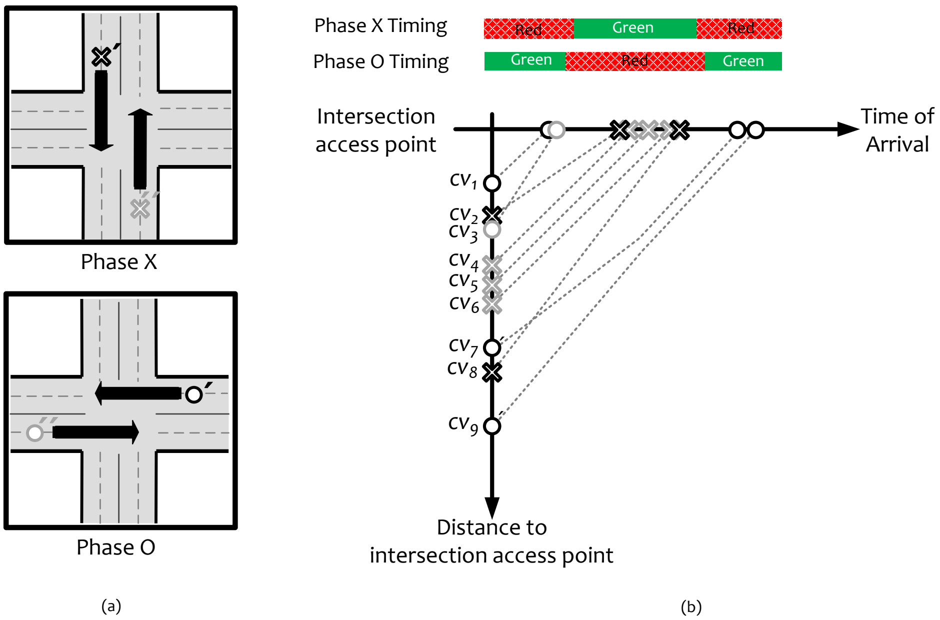


Figure 6.1: Scheduling the vehicles arrivals (an example).

6.3 Definitions and Notations

Achieving an optimal solution such as the example shown in Figure 6.1(b), requires to first formalize the problem objective as well as the constraints with which the vehicles must deal. Before providing the proposed formulations, in this section we first introduce the notations and their definitions. Table 6.1 and Table 6.2 summarize the notations that our proposed intersection controller uses to express the attributes of the intended intersection and the connected vehicles subscribed to that intersection, respectively.

Connected Vehicles. For each intersection, we will introduce a subscription process (Section 7.4) by which the approaching connected vehicles send subscription requests to the intersection control server and announce their presence as well as their intended time of arrival. We represent the list of all subscribed connected vehicles as $CV = \{cv_i\}_{i=1}^{n}$ where n is the size of CV. The list of connected vehicles is sorted by distance to the intersection where cv_1 is the closest vehicle to the intersection. The length of the vehicle is denoted by L, and is taken to be 5.0 meter [52].

Intersection. It is assumed that the intersection is square and has the width of W=10 m. As shown in Figure 6.1(a), we consider a two-phase intersection consisting of Phase X and Phase O as $\phi = \{\phi_X, \phi_O\}$. Each phase includes a set of non-conflicting movements and $M = \{O', O'', X', X''\}$ is the set of movements used in this part of the dissertation (see Figure 6.1(a)). It is assumed that all intersecting roads have the same

Table 6.1. Notations used to express intersection attributes.

	Description
W	width of the intersection (10 m)
d_{safety}	estimated distance required to stop a vehicle when a dangerous situation is detected
O'	the west bound movement
O''	the east bound movement
X'	the south bound movement
X''	the north bound movement
M	set of the movements in this dissertation $M = \{\mathrm{O}', \mathrm{O}'', \mathrm{X}', \mathrm{X}''\}$
ϕ_{X}	Phase X, the combination of non-conflicting concurrent movements $\phi_{\mathrm{X}} = \{\mathrm{X}', \mathrm{X}''\}$
ϕ_{O}	Phase O, the combination of non-conflicting concurrent movements $\phi_{\mathrm{O}} = \{\mathrm{O}', \mathrm{O}''\}$
ϕ	the combination of all phases $\phi = \{\phi_{\mathrm{X}}, \phi_{\mathrm{O}}\}$
cv	the connected vehicle subscribed to the intersection
CV	the list of all the vehicles subscribed to the intersection sorted by distance to the intersection $CV = \{cv_i \mid \quad cv_i \text{ is subscribed}, \ 1 \leq i \leq n, \ d_i \leq d_{i+1}\}$
n	the number of all the vehicles subscribed to the intersection ($n = \#CV$)

speed limit denoted by v_{max}.

Time Instances. For each vehicle approaching an intersection, we are interested in the following time instances: (1) time when the front of the vehicle enters the intersection area at the stop-bar; (2) time when the rear of the vehicle exists the intersection area; (3) time when the front of the vehicle reaches a safety distance from the intersection. As shown in Figure 6.2(a), these time instances are denoted by t_{enter}, t_{exit}, and t_{access}, respectively. In this figure, the intersection area and the safety area are shown by shaded area and solid box, respectively; and d_{safety} is the estimated distance gap required to stop the vehicle before crossing the intersection when a dangerous situation is detected. Based on [22], Equation (6.1) gives this distance gap as a function of the arterial road average speed v_{avg}:

$$d_{safety} = t_{res}.v_{avg} + \frac{0 - v_{avg}^2}{2.a_{dec,max}} \tag{6.1}$$

where t_{res} = 0.5 sec is the response time of an autonomous vehicle, and $a_{dec,max}$ = -4 (m/s^2) is the maximum declaration rate considered for passenger cars when a dangerous situation is detected. We obtain $d_{safety} \approx 38$

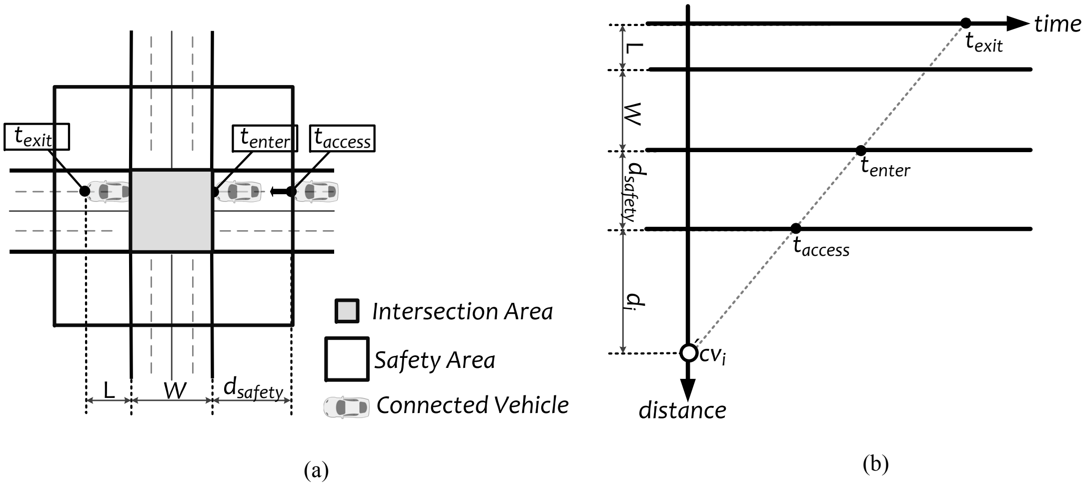


Figure 6.2: Visualization of the intersection-entering, intersection-exiting, and safety area accessing times described (a) at intersection (b) in space-time diagram.

m by setting v_{avg}=35 mph.

It should be emphasized that the entering times into the safety area are denoted by t_{access} because the access to the intersection is granted to the vehicle that is entering the safety area; and all other vehicles with conflicting trajectories must access the intersection later by a sufficient time gap. Figure 6.2(b) also visualizes the aforementioned time instances by showing the position of the vehicle head as a function of time. The distance of the vehicle cv_i to the access point is also denoted by d_i (see Figure 6.2(b)).

Vehicle Attributes. The attributes of each vehicle $cv_i \in CV$ ($1 \leq i \leq n$) that is subscribed to the intended intersection controller are described by:

$$cv_i = \langle m_i, \phi_i, d_i, v_i, t_{access,i}, t_{access,des,i}, t_{exit,i} \rangle \tag{6.2}$$

and explained in Table 6.2. Please note that in this paper, we assume that all vehicles prefer to travel at the average velocity v_{avg}=35 mph; and as a result, their distance divided by v_{avg} yields their desired access times with respect to current time (t_0=0 sec). The vehicle reports its previously assigned access time as its new desired access time ($t_{access,des,i} = t_{access,i}$) in each communication with the intersection controller. This approach reduces the chance of assigning widely varying access times to each vehicles in subsequent assignments. This is intended to minimize the deceleration and acceleration required each time the vehicle receives a new assigned access time which is more fuel efficient.

Table 6.2. Notations used to express connected vehicles.

	Description
m_i	the vehicle movement $m_i \in M$ reported from cv_i to the intersection controller
ϕ_i	the phase $\phi_i \in \phi$ that cv_i movement is associated with
d_i	the distance of cv_i to the intersection access point at current time
v_i	the velocity of cv_i at current time
$t_{access,i}$	the assigned time-stamp for cv_i to access the intersection (assigned access time)
$t_{access,des,i}$	the cv_i's desired access time
$t_{exit,i}$	the intersection-exiting time instance associated with $t_{access,i}$

6.4 Problem Formulation

In this section, we formulate the problem of how to assign access-times to vehicles so that the performances of the signalized intersection and connected vehicles are both improved compared with traditional traffic signal controls. This, in particular, means minimizing the stopped delay and ensuring no crashes occur at the intersection while considering the travel velocities that are desirable to the vehicles.

6.4.1 Assumptions

We make the following assumptions:

- The intersection controller can access the privacy-protected information of the approaching connected vehicles.
- The connected vehicle cv_i is assumed to be automated and capable of deciding on its motion to reach the safety area at the assigned access time.
- Each cv_i can decide on its preferred access time and send it to the intersection controller at the time of subscription. This desired access time ($t_{access,des,i}$) can be computed based on the average speed of the arterial road, the historical average speed of cv_i, and also the speed preferences of the passengers in the autonomous car.

6.4.2 Objective

The intersection throughput is expected to significantly increase using the proposed formulations. The major goal is to find the optimal sequence and times of arrival (t_{access}) for each vehicle such that the difference between the current time (t_0) and the expected arrival time of the last vehicle passing the intersection in a given time window is minimized:

$$\begin{aligned} \arg\min\ J_1 &= \arg\min_{t_{access,j}} (t_{access,j} - t_0) \\ \textbf{s.t.} \quad t_{access,j} &= \max(\{t_{access,1}, \ldots, t_{access,n}\}) \end{aligned} \tag{6.3}$$

Although the aforementioned objective will maximize the number of vehicles that clear the intersection in a given time, it would cause a situation where the vehicles end up traveling near the speed limit even if they do not prefer to. As a result, we state an additional optimization as given in Equation (6.4) to also take the desired intersection-accessing times of the vehicles into account. In this way, the objective is found in such a way that vehicles would not face extreme delay or expedition compared to their desired arrival-times.

$$\arg\min\ J_2 = \arg\min_{t_{access,i}} \sum_{i=1}^{n} |t_{access,i} - t_{access,des,i}| \tag{6.4}$$

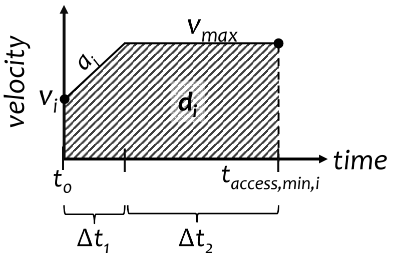
A multi-objective optimization can then be stated by the weighted sum method which is the most widely used method for this reason [101]. This method transforms multiple objectives into one function as:

$$J \ = w_1 J_1 \ + \ w_2 J_1 \tag{6.5}$$

where w_1 and w_2 are the weights that should be assigned by the intersection controller. We propose that this multi-objective optimization will lead in reductions in fuel consumption, and stopped delay, although these factors are not explicitly incorporated into the individual objective functions. At the end of this chapter, a simulation approach is employed to observe the effectiveness of the proposed formulation based on the assumptions given in Subsection 6.4.1. It will be later observed in Section 6.6 that platoon formations will also be indirectly encouraged by the proposed signal control. The detailed simulation and experimental results with measure of effectiveness (MOE) analysis will be given later in Chapter 9.

6.4.3 Constraints

Several constraints are imposed to ensure safety. The main challenge is expressing the constraints as a function of access times so that a linear constrained optimization problem can be derived at the end.



(a)

$$\Delta t_1 = \min\left\{\frac{v_{max} - v_i}{a_i}, \frac{v - v_i}{a_i}\bigg|_{v=\sqrt{{v_i}^2 + 2a_i d_i}}\right\}$$

$$\Delta t_2 = \max\left\{\frac{d_i}{v_{max}} - \frac{{v_{max}}^2 - {v_i}^2}{2a_i v_{max}}, 0\right\}$$

$$a_i = a_{acc,max}$$

$$t_{access,min,i} = t_0 + \Delta t_1 + \Delta t_2$$

(b)

Figure 6.3: Earliest access time possible based on the speed limit and maximum acceleration rates.

A. Speed limit and maximum acceleration

For each vehicle cv_i, we should consider the speed limit requirement $v_i \leq v_{max}$ as well as the maximum acceleration rate $a_i \leq a_{acc,max}$; where v_i and a_i are the velocity and acceleration rate of the vehicle, v_{max} is set based on the speed limit of the arterial road, and $a_{acc,max}$ is the maximum acceleration and is considered +3 m/s^2 in this part of the dissertation. We introduce $t_{access,min,i}$ as the earliest time that cv_i can access the intersection if it travels at maximum acceleration and maximum speed possible. Then we rephrase our aforementioned speed and acceleration constraints as:

$$t_{access,i} \geq t_{access,min,i} \tag{6.6}$$

This earliest access time, $t_{access,min,i}$, is calculated as explained in Figure 6.3. This time instance depends on the distance of cv_i to the access point (d_i) and the max(.) and min(.) functions shown in Figure 6.3 handle this dependency problem.

B. Safe gap on the same movement

Two consecutive vehicles that are traveling on the same movement (e.g. east bound) should be separated by a safe gap (headway). This is independent of the vehicles speed [22] (except at very low speeds), and actually is the minimum following time gap to avoid a rear end collision. In the first part of the dissertation (Subsection 3.4.3), the saturation headway value of h=1.47s was estimated and was found consistent with the measurements in literature for conventional cars [28]. On the other hand, the computer control in autonomous vehicles will eliminate human reaction times; and assuming a very small communication delay, a time headway of 0.2s may be sufficient for safe automatic vehicle following [102]. Considering a tolerance parameter, it is suggested in [22] that a 1 sec headway reasonably upper bounds the response time of an autonomous vehicle. However, a 1 sec headway is not sufficient at very low speeds such as when discharging from a queue. We observed in simulations that when discharging from a queue, the headway between the first vehicle and the second vehicle leaving the stop position can be as large as 2.3 sec. As a result, we default to t_{gap1} = 2.5 sec in our optimization formulations; only if it is determined that a vehicle will access intersection at a large enough speed we set t_{gap1} = 1 sec. To enforce the headway, we add the following constraint on any two consecutive vehicles traveling on the same movement:

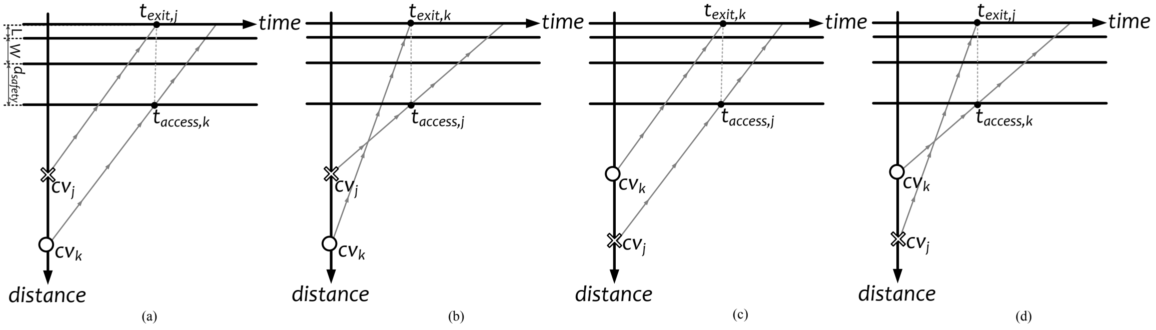
$$\begin{aligned} & t_{access,j} - t_{access,k} \geq t_{gap1} \\ \textbf{s.t.} \quad & (cv_j, cv_k) \in CV, \quad d_j \geq d_k; \\ & (m_j, m_k) \in M, \quad m_j = m_k. \end{aligned} \tag{6.7}$$

C. Safe gap on different movement

Two vehicles traveling on different phases (conflicting movements) also need to be separated by a safe gap. This time gap, if selected properly, guarantees that the second vehicle would not be within the safety area before the first vehicle leaves the intersection area. Considering two vehicles cv_j and cv_k that are on the different phases of $\phi_j \in \phi$ and $\phi_k \in \phi$ ($\phi_j \neq \phi_k$), there are four possible situations with just enough safe gap between the vehicles. These situations are shown in a space-time diagram in Figure 6.4 by projecting the vehicles trajectories on one single distance axis. It can be concluded from this figure that the following constraints cover all the possible situations where $\vee$ is the OR operator:

$$\begin{aligned} & t_{access,j} - t_{exit,k} \geq 0 \\ & \quad \vee \\ & t_{access,k} - t_{exit,j} \geq 0 \\ \textbf{s.t.} \quad & (cv_j, cv_k) \in CV; \\ & (\phi_j, \phi_k) \in \phi, \quad \phi_j \neq \phi_k. \end{aligned} \tag{6.8}$$

Although Equation (6.8) mandates at least 0 sec gap between the accessing and exiting time instances of the first and second vehicle, it does not explicitly state the gap needed between the accessing times of those vehicle. We are specifically interested in the time gap between accessing timestamps so that, at the end, we can derive a linear constrained optimization problem based on access times only. For this reason, we define



(a) (b) (c) (d)

Figure 6.4: Possible scenarios of two vehicles passing an intersection with just enough safe gap between them. (a),(d) cv_k accessing right after cv_j exiting. (b),(c) cv_j accessing right after cv_k exiting.

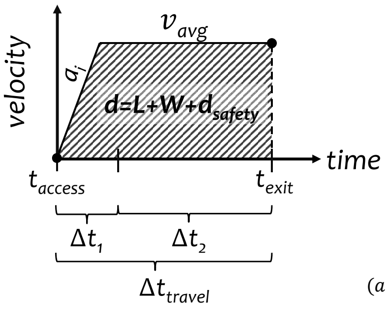
$t_{exit} = t_{access} + \Delta t_{travel}$ where Δt_{travel} is the travel time between access point and exit point of a vehicle. For simplicity's sake, we assume the intersection is square and the two intersecting roads have the same speed limits. As a result, the travel time does not depend on phase. Now, by substituting $t_{exit} = t_{access} + \Delta t_{travel}$ into Constraint (6.8), we can rephrase this constraint as:

$$\begin{aligned} & t_{access,j} - t_{access,k} \geq (t_{gap2} = \Delta t_{travel}) \\ & \quad \vee \\ & t_{access,k} - t_{access,j} \geq (t_{gap2} = \Delta t_{travel}) \\ & \textbf{s.t.} \quad (cv_j, cv_k) \in CV; \\ & \qquad\; (\phi_j, \phi_k) \in \phi, \quad \phi_j \neq \phi_k. \end{aligned} \tag{6.9}$$

where t_{gap2} is the safe gap we need between access points. The travel time between the access point and the exit point of a vehicle is equal to the time period it needs to first pass the safety area, then pass the intersection area, and finally exit the intersection completely. The longest travel time a vehicle could take is when it is stopped behind the safety area and accelerates at its assigned access time as shown in Figure 6.5(a). We set an average acceleration rate of 2 m/s^2 for the vehicles and obtain Δt_{travel}= 7.3 sec as calculated in Figure 6.5(b). We set t_{gap2}=7.5 sec in our formulations.

6.5 Handling of Removable Discontinuities

We plan to solve the formulations proposed in the previous section by linear programming. As a result, any discontinuity and disjunction in the formulations need to be removed first. There are three disjunctions found in the previous section:



(a)

$$\Delta t_1 = \min\left\{\frac{v_{avg}-0}{a_i}, \frac{v\_0}{a_i}\bigg|_{v=\sqrt{2a_i d}}\right\}$$

$$\Delta t_2 = \max\left\{\frac{d}{v_{avg}} - \frac{v_{avg}}{2a_i}, 0\right\}$$

$$\Delta t_{travel} = \Delta t_1 + \Delta t_2 = 7.3 \text{ sec}$$

$$(a_i = 2\frac{m}{s^2}, v_{avg} = 35\, mph, d_{safety} = 38m, \quad L = 5m, W = 10m)$$

(b)

Figure 6.5: The longest possible travel time between the accessing and exiting points.

6.5.1 Discontinuity in Constraint

Our first step to handle disjunctions is to identify any formulations with if-then-else statement or in ”OR” logic symbolic form. The Constraint (6.9) has discontinuity and is not linear because it includes the OR logic operator ($\vee$). The goal here is to convert this constraint into an AND-combination of two or more equations in such a way that if one equation holds true then the other equations are always redundant. The most widely known method to handle disjunctions is the big-M method that, in our application, requires a binary variable B and a parameter M [103]. For each set of Constraint (6.9) applying on two vehicles of cv_j and cv_k, we add one artificial binary variable B_i ($1 \leq i \leq$ # of constraints) to take care of the discontinuity as:

$$\begin{aligned} & t_{access,j} - t_{access,k} + M.B_i \geq t_{gap2} \\ & \quad \wedge \\ & t_{access,k} - t_{access,j} + M.(1-B_i) \geq t_{gap2} \\ & \textbf{s.t.} \quad (cv_j, cv_k) \in CV; \\ & \qquad\; (\phi_j, \phi_k) \in \phi, \quad \phi_j \neq \phi_k; \\ & \qquad\; B_i \quad binary \end{aligned} \tag{6.10}$$

where B_i can be either 0 or 1, M is a large enough number, and $\wedge$ is AND operator. If B_i=0 then the first equation of the above constraint holds true if $t_{access,j} - t_{access,k} \geq t_{gap2}$ and the second equation ($t_{access,k} - t_{access,j} \geq (t_{gap2} - M)$) is redundant and always hold true if M is big enough. If B_i=1 then the first equation ($t_{access,j} - t_{access,k} \geq (t_{gap2} - M)$) is redundant and always hold true if M is big enough, and the second equation holds true if $t_{access,k} - t_{access,j} \geq t_{gap2}$.

It is possible to predict how big M must be: either of the redundant equations discussed above need to be always fulfilled; and this requires that $M \geq t_{gap2} + t_{access,j} - t_{access,k}$. Considering the fact that t_{gap2} is small, and can be neglected comparing to M, a lower bound to M is equal to the latest access time minus the most recent access time that may be assigned to two vehicles. As a worst-case example, the first vehicle would access the intersection at as early as the current time (t_0) and the second vehicle would travel the whole subscription distance (e.g. 4 km) in a very low speed (e.g. 10 mph) and would stop waiting excessively to access the intersection (e.g. 500 sec); that totally leads to 1394 sec gap between the two vehicles and consequently we choose to set $M = (2000 \geq 1394)$.

6.5.2 Discontinuity in Cost Function J_1

The cost function J_1, introduced in Equation (6.3), is discontinuous in the sense that it includes the largest (latest) access time variable which is not assigned to any specific vehicle till we obtain the optimization results. One solution is to always assign the latest access time to the furthest subscribed vehicle that is cv_n considering the fact that the list of all subscribed vehicles (CV) is sorted by distance to the intersection. We then rephrase the optimization objective and constraint as given in Equation (6.11) where $t_{access,n}$ is the access time assigned to the cv_n and should be larger than or equal to the access times of all other vehicles.

$$\begin{aligned} &\arg\min\ J_1 = \arg\min_{t_{access,n}} (t_{access,n} - t_0) \\ &\textbf{s.t.}\quad n = \#CV \\ &\qquad\ t_{access,n} \geq (\{t_{access,1}, \ldots, t_{access,n-1}\}) \end{aligned} \tag{6.11}$$

6.5.3 Discontinuity in Cost Function J_2

The absolute value signs cannot be included in a linear programming formulation. As a result, the cost function J_2, introduced in Equation (6.4), needs to be restated. Although, this is possible by using the big-M method, we restate J_2 by adding a new so-called slack variable $\Delta t_{access,abs,i} = |t_{access,i} - t_{access,des,i}|$. Then, considering the fact that $|x| = max\{x, -x\}$ for any real number x, we can add two constraints of $\Delta t_{access,abs,i} \geq (t_{access,i} - t_{access,des,i})$ and $\Delta t_{access,abs,i} \geq -(t_{access,i} - t_{access,des,i})$ in order to ensure that our added slack variable is equal to $|t_{access,i} - t_{access,des,i}|$. In summary, the restated cost function and the added constraints are as follows:

$$
\begin{aligned}
&\arg\min\ J_2 = \arg\min_{t_{access,i}} \sum_{i=1}^{n} \Delta t_{access,abs,i} \\
&\textbf{s.t.}\quad \Delta t_{access,abs,i} \geq (t_{access,i} - t_{access,des,i}) \\
&\qquad\ \Delta t_{access,abs,i} \geq -(t_{access,i} - t_{access,des,i})
\end{aligned}
\tag{6.12}
$$

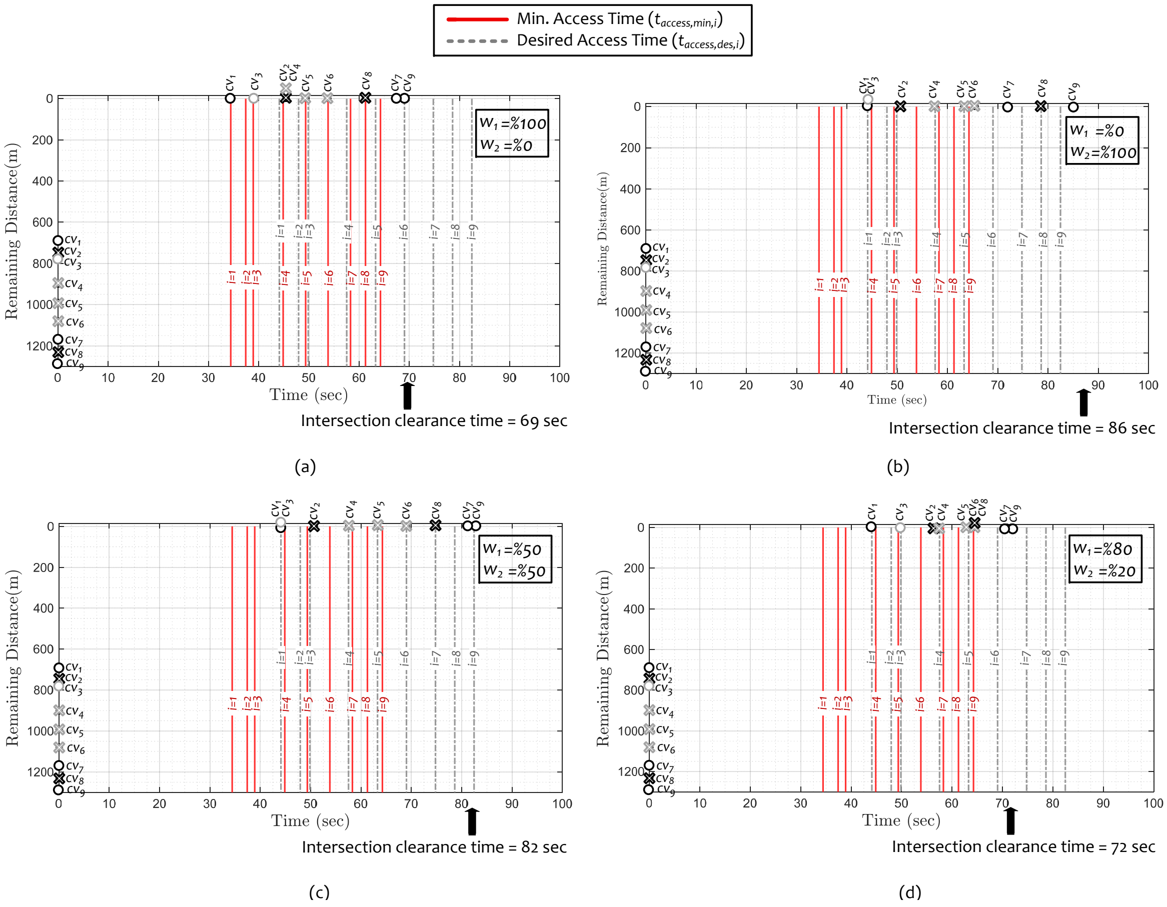
6.6 Mixed-Integer Linear Programming Case Study

The linear objective function/constraints and mixed-integer variables in the optimal solution make our formulations a Mixed Integer Linear Programming (MILP) problem for which efficient solution exists. Since the intersection control algorithm runs on powerful backend clouds, the computational complexity will be manageable in real time. Our detailed simulation and experimental results with measure of effectiveness (MOE) analysis will be given later in Chapter 9; however, this section employs a simplified case study only to observe an example numerical outputs of the proposed formulations.

To solve this MILP problem in this section, we use the *intlinprog* function in Matlab (version R2016a) from Optimization toolbox. We simulate the same example as previously shown in Figure 6.1 including n=9 connected vehicles cv_i (1 $\leq$ i $\leq$ 9). We set speed limit v_{max}=45 mph, and average arterial road speed v_{avg}=35 mph. We assume that the current state of all the vehicles is available: they are all traveling in v_{avg} and their distance to the safety area are [690 m, 750 m, 780 m, 900 m, 990 m, 1080 m, 1170 m, 1230 m, 1290 m], respectively.

Figure 6.6(a-d), demonstrates the optimal solutions to the aforementioned problem found by MILP. As shown in Figure 6.6(a), by setting w_1=%100 and w_2=%0, all the weight is given to intersection throughput improvement (J_1, Cost Function (6.11)), and, as a result, some vehicles end up traveling near the speed limit (solid red lines). On the other hand, by setting w_1=%0 and w_2=%100 in Figure 6.6(b), all the weight is given to satisfying the desired speeds of all vehicles (J_2, Cost Function (6.12)); and, as a result, the intersection clearance time was increased for 17 sec compared with Figure 6.6(a). Two compromised solutions are also demonstrated in Figure 6.6(c) and (d) for comparison purposes. For the rest of the dissertation, we set w_1=%80 and w_2= %20.

A feature of our formulations is that two vehicles that are traveling on different parallel traffic movements (e.g. cv_4 vehicle moving north bound on X$''$ and cv_2 moving south bound on $\mathbf{X}'$) can be served almost at the same time (or in other words, overlapped arrival-times can be assigned to them). Most important fea-



(a)

(b)

(c)

(d)

Figure 6.6: Scheduling the arrivals of vehicles using the proposed MILP model, solved by *intlinprog* function in Matlab R2016a (y-axis: projected remaining distance to the safety area, x-axis: assigned access times); (a) all weight given to intersection throughput improvement (b) all weight given to satisfying the desired speeds of all vehicles (c) %50 \ %50 compromised solution (d) %80 \ %20 compromised solution.

ture, that can also be identified in the Figure 6.6(a), is that since $t_{gap2} > t_{gap1}$, platoon formations will be encouraged, otherwise lone vehicles have to clear the intersection with the longer safe gap t_{gap2} with respect to opposing movements which reduces intersection capacity.

Chapter 7

Cyber-physical Test Environment

7.1 Introduction

The development of Intelligent Transportation Systems (ITS), especially intelligent traffic signal control, is difficult because these systems must be tested in the real world or near real world conditions (micro-simulations). The existing traffic signal controllers do not generally allow data communication between the controller and a simulation [104]. Moreover, they are not designed to benefit from the data streams sent from real connected vehicles. In this chapter, we aim at providing a methodology and a Vehicle-In-the-Loop (VIL) simulation platform which can incorporate most of the real time signal control methods for verification purposes.

Traffic microscopic simulation has been widely used for the purpose of evaluating ITS systems. However, the way of interfacing micro-simulations with traffic signal controllers can be different in every application based on the evaluation purposes. Stevanovic et al. describes two main methods of connecting a traffic micro-simulator with a traffic signal controller as follows [105]:

- **Software-in-the-loop (SIL) simulation**: where the micro-simulation and virtual traffic signal controller run on the same computer. As a result, no hardware interface is needed between the simulation and the virtual controller. This allows a faster simulation compared to other configurations [106]. Figure 7.1(a) demonstrates a software-in-the-loop configuration for the application of this dissertation.

- **Hardware-in-the-loop (HIL) simulation**: where a piece of network interface hardware (Physical layer interface) connects the traffic simulation engine and the traffic signal controller. The traffic simulation

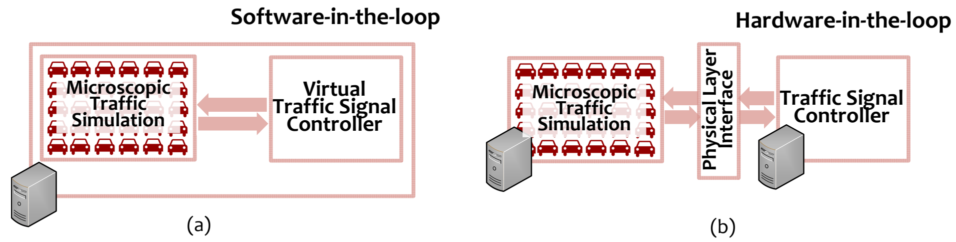

Software-in-the-loop
Microscopic Traffic Simulation
Virtual Traffic Signal Controller
(a)
Hardware-in-the-loop
Microscopic Traffic Simulation
Physical Layer Interface
Traffic Signal Controller
(b)

Figure 7.1: (a) Software-in-the-loop and (b) Hardware-in-the-loop confguration in the traffic simulation applications.

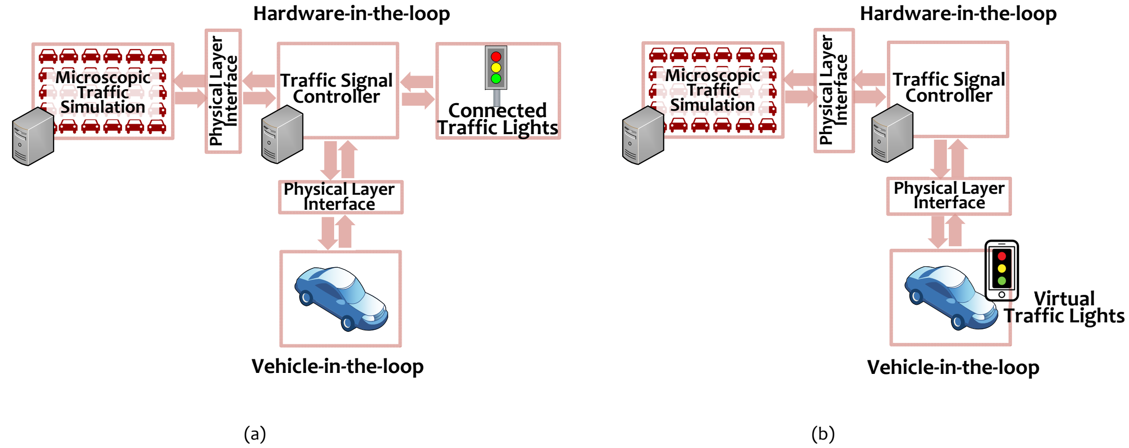
must run on a real-time basis [105]. Figure 7.1(b) demonstrates a hardware-in-the-loop configuration for the application of this dissertation.

It is possible to incorporate one or more real test vehicles into the microscopic traffic simulation at an intersection. This vehicle-in-the-loop (VIL) configuration can be included in either configurations of the HIL or SIL shown previously in Figure 7.1. However, adding a real vehicle to a SIL onfiguration requires the fast simulation speed of the SIL to be downgraded to acheieve real-time simulation. This eliminates the main advantage of the SIL configurations and, as a result, is not used here in this dissertation.

Instead, adding real vehicles to HIL configuration is advantageous for several reasons. Testing conditions in a VIL setup is configurable [107], reproducible, and repeatable [108] comparing to real traffic conditions. Moreover, VIL is a safe solution for systems, such as collision mitigation systems [24], that their testing in real test driving maneuvers is nearly impossible. It should be emphasized that the VIL concept should not be confused by Vehicle-Hardware-in-the-Loop (VeHIL), where a chassis dynamometer emulates the road interaction [109, 110].

Besides all the benefits mentioned above, VIL test setup usually requires adding virtual objects to the real scene that a driver views in the test vehicle. This can be accomplished, for example, by an optical see-through Head Mounted Display as used in [24, 111]. However, this dissertation aims at the efficiency and connectivity of traffic signal controllers at signalized intersections only. As a result, the movements at intersection can be assigned to the virtual and real vehicles in such a way that there is no need to project the virtual vehicles into the driver's gaze. A future research focusing on collision avoidance at signalized intersections would add an optical see-through Head Mounted Display as a possible visualization method for our application.

To the best of our knowledge, to date, only the traffic signal microsimulation presented by Quinlan et al. [112] is also implemented in a vehicle-in-the-loop manner. The HIL traffic signal simulations presented in [105, 113] couple external traffic signal controllers with microsimulations only, and Day et al. [106] study



(a)

(b)

Figure 7.2: Adding vehicle-in-the-loop to a hardware-in-the-loop configuration with (a) actual traffic lights or (b) virtual traffic lights.

the event-based performance measures by integrating a real world adaptive signal controller with software-in-the-loop simulated controllers.

As the last drawback of adding a real vehicle to HIL, a traffic light is needed locally at the intersection, as shown in Figure 7.2(a), to notify the driver of signal status. However, a physical traffic signal, similarly used in testbeds of [114, 115], requires at least a modem for connection to our custom traffic signal control server, and a computer for data translation between the server and the traffic signal's embedded system. This adds to complexity and cost of the testbed, and can be detrimental to reproducibility feature of the proposed VIL testbed.

In view of these issues, a virtual traffic signal is a solution as it appears on the display of our virtual driver assistant and displays the signal status to the driver (see Figure 7.2(b)). Moreover, if a vehicle is autonomous, then displaying the signal status is not even necessary. What we are proposing is actually an infrastructure-based approach to virtual traffic signals which needs wireless connection to a remote traffic signal control server in order to receive its status or SPaT data. However, the proposed concept of virtual traffic signals presented in [116, 21] and also used in [117] does not need any infrastructure-based traffic control and receives on-demand SPaT information in an ad hoc manner by V2V communication. In general, virtual traffic signals presented here and in [116, 21, 117] are all in-vehicle representations of actual traffic signals supported by the connectivity available within a connected vehicle environment.

In any VIL configuration for autonomous-targeted research, it would be very difficult to utilize an autonomous vehicle as a test vehicle for several reasons. Implementing custom algorithms into an intelligent vehicle needs access to the in-vehicle application which are not available due to proprietary reasons and consequently requires a close partnership with the manufacturer of that vehicle. Moreover, it is an expensive technology and there is great deal of challenge and research in building autonomous test vehicles from scratch because it includes guidance, sensors and sensing strategy, navigation, cartography and motion planning [118]. Last, and most important, testing legislations as well as insurance liability must be considered depending on the country and the state where the test is conducted. For example, at least $5 million in insurance coverage is required by California and Florida states; and California's Department of Licensing must permit the testing plan before it is executed [119, 120].

Because of these issues, we use a conventional vehicle as our test vehicle. In fact, considering the application of this dissertation, a conventional vehicle can be assumed as an autonomous test vehicle because:

- The objective is to test the efficiency of intersection signal control not the functionality of autonomous vehicle itself.
- The driver is guided via a virtual driver assistant so that she or he can follow the planned speed. The assistant also notifies the driver when to stop at red or start moving at green light. Being aware of the start of the green light ahead of time, decreases the driver's start-up reaction time.

An in-vehicle implementation of a virtual driver assistant system requires a close partnership with car makers or OEMs that is not possible at this stage of our research. As a result, an application running on a smart-phone is a reasonable alternative. The idea is to display the planned speed to the driver as green zone on a GPS speedometer. This way of recommendation display was adopted from [114, 115].

In this chapter, we will introduce a novel cyber-physical virtual testing environment, in which actual test vehicles at the physical layer interact with 1) simulated vehicles in a traffic micro-simulation layer, and 2) intersection controller that run and interact on a back-end cloud. First, the vehicle-in-the-loop (VIL) configuration is explained in Section 7.2. The components of our VIL setup including virtual traffic signal, and virtual driver assistant are explained in the subsequent sections. Our proposed data structure and the utilized communication technology are introduced in Section 7.4. We use SIL simulations as well as VIL simulations in this dissertation, and their results are given in Chapter 9; however, the simulation setup is explained shortly in Section 7.5 of this chapter.

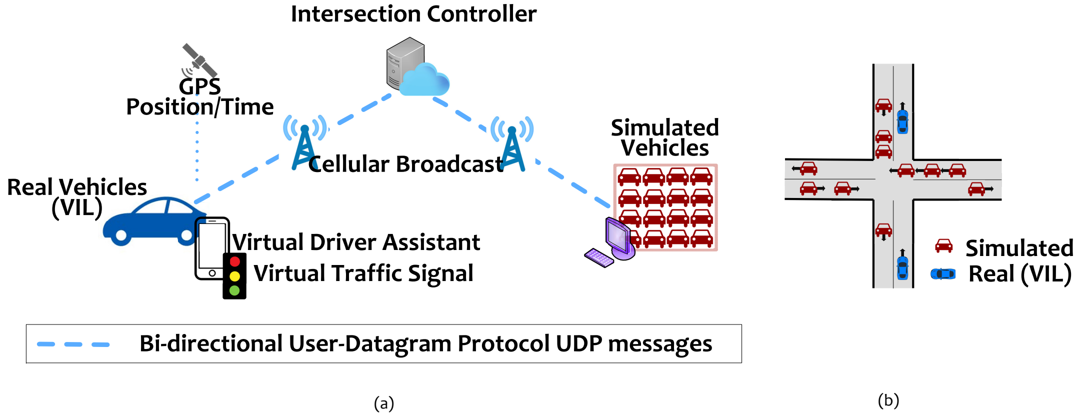


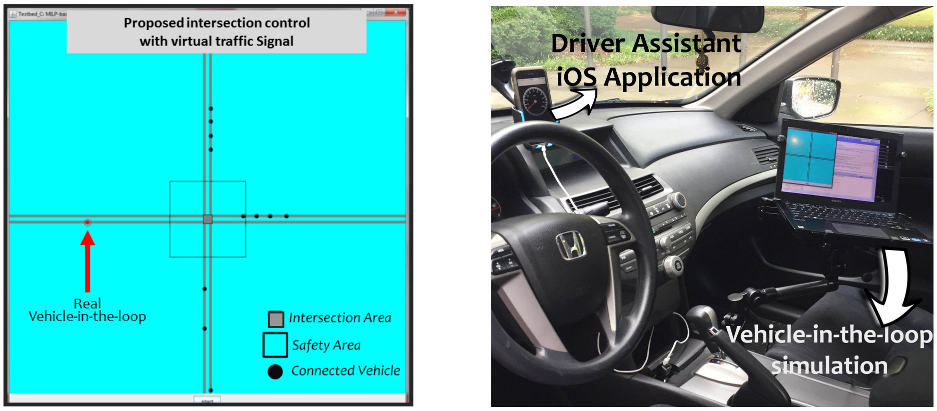
Figure 7.3: Vehicle-In-Loop (a) back-end configuration (b) visualization at intersection.

7.2 Vehicle-In-Loop Configuration

We add one single real vehicle to HIL configuration at an intersection testbed. Both simulated and real vehicles are treated similarly by the intersection controller. This, in particular, means that they all send and receive data in the same structure and format (see Figure 7.3(a)). Furthermore, they all follow the same subscription/un-subscription process that will be explained later in Section 7.4. The proposed approach addresses many limitation of a simulation only environment, while also ensures a safer environment for test vehicles because conflicting movements (and potential crashes) occur in a simulated environment.

Figure 7.3(b) demonstrates that the phases and movements are assigned to the simulated and the real vehicle in such a way that the simulated environment does not need to be visualized to the test driver while she/he is driving the real vehicle. A possible visualization method for our future research would be an optical see-through Head Mounted Display as used in [24]. Comparing to Figure 6.1(a), the real vehicle (VIL) only moves north bound on X$''$ movement and the simulated vehicles travel on all other movements of our virtual road network environment (**O**$'$, O$''$, **X**$'$). Figure 7.4(a), shows a screenshot of the real vehicle interacting with hundreds of simulated vehicles in our microsimulation environment.

If any collision happens between the VIL and a simulated vehicle, the test driver does not notice and the collision will be flagged by offline data analysis after the road test. Nevertheless, we mounted our simulator node inside the test vehicle as shown in Figure 7.4(b) so that collisions could be identified visually by our colleague who physically sat in the vehicle.



(a) (b)

Figure 7.4: Vehicle-In-Loop (a) screenshot of the vehicle interacting with the simulator (b) in-vehicle setup.

7.3 Virtual Driver Assistant

Although, in this part of dissertation, we derive optimal vehicle arrivals for the approach of autonomous vehicles, we use a conventional vehicle as our test VIL. As a result, a driver assistant is needed to guide the driver for a timely arrival at intersection. We developed an iOS virtual driver assistant specifically for the application of this dissertation that displays the appropriate speed recommendation to the driver as green zones on a GPS speedometer (please see Figure 7.4(b)). The virtual traffic signal is also shown to the driver on the same display as the speed recommendation is. The application runs on iOS devices (iPhone, or iPad) and the designed human machine interface (HMI) is explained in Figure 7.5. The HMI shown in this figure is displayed to the driver only when the upcoming intersection and phase are identified as explained in next subsection.

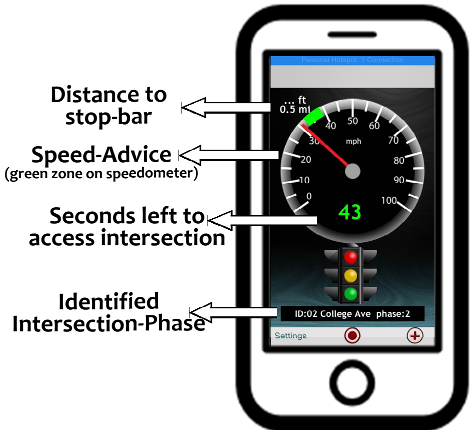


Figure 7.5: Description of HMI of the Virtual Driver Assistant and the Virtual Traffic Signal implemented on iOS iPhone device.

7.3.1 Intersection and Phase/Movement Identification

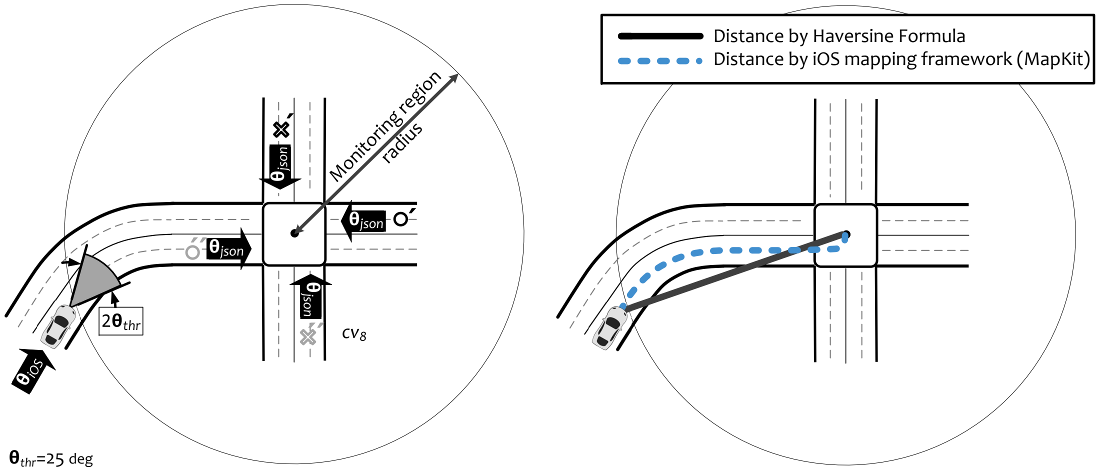
In determining which intersections and which traffic signal phases/movements are relevant to a vehicle along a trip, a list of intersections and their traffic signal phases (so called intersection-phase pairs in this dissertation) is preloaded into the virtual driver assistant (stored locally in the iOS device). The JavaScript Object Notation (JSON) [66] is used to represent this list which contains the attributes of each intersection-phase such as GPS location of the intersection center, intersection ID number, phases according to the standard NEMA phase numbering scheme, monitoring radius for each intersection, and entry and exit headings of each phase.

The implemented driver assistant is a location-based application and, immediately after the application is initiated, it starts monitoring the intended intersections with the monitoring radii predefined in JSON list. Once the vehicle enters the monitoring region (see the circular region in Figure 7.6(a)) the relevant intersection specification is loaded from the JSON file. In order to identify the most relevant traffic signal phase/movement to the vehicle direction of travel, the driver assistant compares the vectors of relevant signal phase entry headings (θ_{json}, loaded form the JSON file) with the current vehicle direction of travel (θ_{iOS}, reported by the GPS hardware of the iOS device). The difference between each phase/movement entry heading and vehicle heading is denoted as $\Delta\theta$; and if $\Delta\theta$ is less than the heading difference threshold, the phase/movement will be identified as the actual phase/movement that the vehicle is traveling towards. As shown in Figure 7.6(a), we accept a heading difference threshold of $\theta_{thr} = 25\,\text{deg}$. The pseudo-code for phase/movement identification is reported in Algorithm (1).

Please note that GPS hardware of the iOS device can report the heading (the direction in which an iOS device is pointing) as well as the course (the direction in which an iOS device is moving) [121]; as a result, we use the course information of the iOS device as the vehicle direction of travel.

Algorithm 1 Phase/Movement Identification using Heading Information

```
procedure LOAD INFO FROM JSON FILE
procedure START MONITORING INTERSECTIONS
procedure IDENTIFY PHASE/MOVEMENT IF VEHICLE $cv_i$ ENTERED AN INTERSECTION REGION
    $\theta_{iOS} \leftarrow$ iOS GPS Hardware
    for all $m \in M = \{O', O'', X', X''\}$ do
            $\theta_{json} \leftarrow$ entry heading of $m$ from JSON file
            find possible heading difference $\Delta\theta_1 = |\theta_{json} - \theta_{iOS}|$
            find possible heading difference $\Delta\theta_2 = 360\,\text{deg} - |\theta_{json} - \theta_{iOS}|$
            find heading difference $\Delta\theta = min\{\Delta\theta_1, \Delta\theta_2\}$
        if $\Delta\theta \leq (\theta_{thr} = 25\,\text{deg})$ then
            set $m_i \leftarrow m$ as the actual phase/movement of $cv_i$ return true
    return false
```


Monitoring region radius
θjson
θjson
θjson
θjson
2θthr
θiOS
cv8
θthr=25 deg
Distance by Haversine Formula
Distance by iOS mapping framework (MapKit)

(a) (b)

Figure 7.6: Virtual Driver Assistant (a) Identifying the Phase/Movement after entering the intersection monitoring region (b) Calculating the distance between the vehicle and the intersection center.

7.3.2 Speed-Planner for the Virtual Driver Assistant

Right after identifying the relevant intersection-phase, the vehicle sends a subscription request to the intersection controller (see Section 7.4 for details). The intersection controller, in return, sends the Signal Phase and Timing (SPaT) if the traffic signal is pre-timed or sends the assigned access time if the traffic signal is MILP-based. Based on the distance of the vehicle to the intersection, the appropriate speed to be followed by the driver is computed by an embedded speed-planner engine. The computed speed is then displayed to the driver as green zones on the speedometer (adopted from [114, 115]), as seen in Figure 7.5. The goal is to guide the driver for a timely arrival at intersection in almost the same way that a real autonomous vehicle would do.

The speed-planner runs locally on the iOS device, and computes a feasible trajectory to reach a goal. The goal depends on our test scenarios and is explained later in next chapter. As an example test scenario, the goal point for an autonomous vehicle connected to our MILP-based intersection controller is to reach the intersection at the assigned access time. Please note that in this dissertation, we present a simple speed-planning component that is used to navigate through urban environments with no anomalous or road hazard conditions. To simplify the model of an autonomous vehicle, we assume there is no obstacle to avoid and lane-changing is not allowed also. Furthermore, the speed-planner embedded in our virtual assistant, does not include any car following model because we have only one real vehicle-in-the-loop that is not sharing its driving lane with simulated vehicles. Even if a future work will utilize two or more real vehicles, the follower drivers will be capable of keeping a safe distance to the vehicles in front.

7.3.3 Distance Calculation

The distance between the connected vehicle and the intersection must be calculated in order to be sent to the intersection controller, and also to be used in planned-speed display. Once the vehicle enters the monitoring region, the driver assistant application uses the MapKit - mapping framework in iOS [122] to find the rout-based distance between the iOS device and the intersection center (available from JSON file). MapKit includes an API, which can provide either walking or driving directions (distances) between two points. We use the walking mode and a sample walking direction is shown by dashed line in Figure 7.6(b). Because of possible positional error associated with the device GPS location and with the intersection center coordinates, these two points may represent two locations on different sides of the road, and, as a result, the API's driving direction may include a U-turn or a reroute. This is the reason we don't use the driving mode.

Nevertheless, the aforementioned API needs to query Apple's servers each time the updated distance is needed. Although, there are no requests limits [122], the queries are server-based and the response may be returned with delay based on the server load and the network connection. For this reason, we also use Haversine Formula [40] to calculate the straight distance between the two points, as shown by solid line in Figure 7.6(b). Once the vehicle is going straight to the intersection, the Haversine and API results are approximately equal. Our virtual driver assistant detects this, stops querying Apple's server, and continue using Haversine formula till the vehicle exits the intersection area. Last, by considering the width of the intersection, the distance to stop-bar is estimated and is also shown on the top left corner of the iOS device display, as shown in Figure 7.5.

7.4 Data Structure and Communication

The V2I communication between connected vehicles (real and simulated vehicles) and intersection controller presents us with two challenges:

- **Scalability and Bandwidth**: communicating a large amount of data can be very expensive given that we use cellular networks technology. Also as the number of vehicles increases, bandwidth challenges are imposed on the server side.

- **Versatility and Universality**: generating software to transmit data to and from a variety of systems with different data streams and different implementation languages can be quite complex and very time consuming. A unified and versatile data model is sought so that minimal data translation is needed for importing data from disparate data sources.

To address the first challenge, we send and receive the information through a User-Datagram Protocol or UDP [64] unconnected datagram sockets. In addition, we propose a vehicle subscription/unsubscription process which also reduces the amount of data exchanged. To address the second challenge, we propose to serialize the data, using for instance Google Protocol Buffers. The descriptions are given in the subsequent subsections.

7.4.1 User-Datagram Protocol (UDP)

Although the delivery is not guaranteed using UDP, this is preferable to Transmission Control Protocol or TCP [65] for our application because our servers deal with small data packets sent from a large number of connected vehicles. The TCP is actually slower for exchanging the instantaneous location reports of a large number of vehicles (the real and simulated vehicles in our application); and our cloud-based server (intersection controller) may not handle receiving all these reports in TCP. It should be emphasized that concurrent with receiving a large number of probe reports, the intersection controller is also responsible to send the assigned access times or the status information of the corresponding traffic signal to the subscribed vehicles.

7.4.2 Vehicle Subscription

As it was mentioned in previous section, while a connected vehicle is traveling, our virtual driver assistant searches among a predefined list of the intersections within a specified range, and it identifies the most relevant intersection-phase to the vehicle movement. The vehicle then sends a subscription request for the identified intersection-phase to the intersection controller server and announces its intended time of arrival. An unsubscribe message is later sent from the vehicle to the server at the time the vehicle clears the intersection. Thus, data is exchanged only during the subscription period and only when new information is available from the vehicle or the intersection controller. The data exchanged during this period (transmitted from and received by the vehicle) is summarized in Figure 7.7.

It should be noted that the VIL simulation environment is designed in a such a way that the simulated vehicles follow the same subscription/unsubscription process as described above and as shown in Figure 7.7.

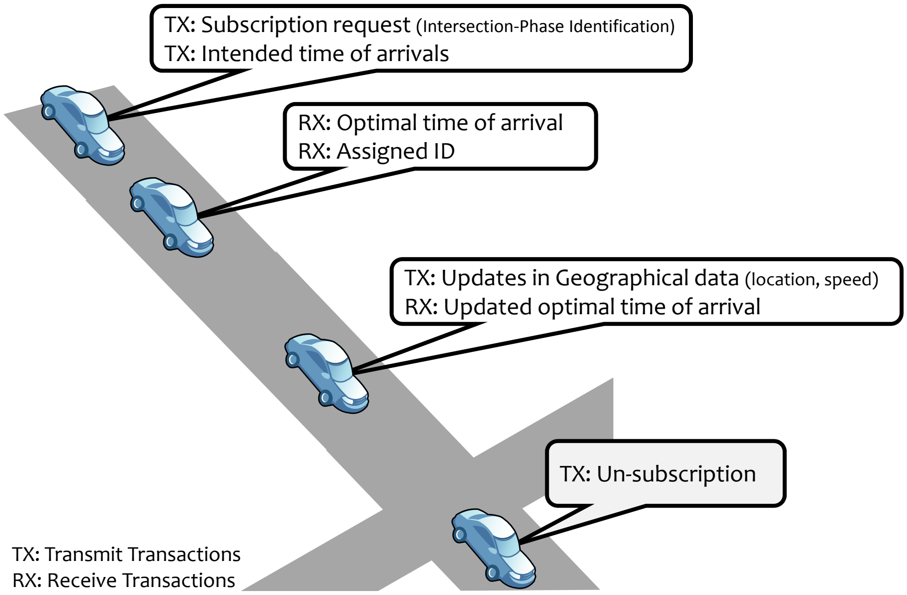

TX: Subscription request (Intersection-Phase Identification)
TX: Intended time of arrivals
RX: Optimal time of arrival
RX: Assigned ID
TX: Updates in Geographical data (location, speed)
RX: Updated optimal time of arrival
TX: Un-subscription
TX: Transmit Transactions
RX: Receive Transactions

Figure 7.7: Data exchanged between a connected vehicle approaching a signalized intersection and the intersection controller (remote server).

The only difference between the simulated vehicles and the real vehicle is that the simulated vehicles do not search among the list of the intersections to find the relevant one.

7.4.3 Google Protocol Buffers

Protocol buffers are a mechanism to serialize data. We specify the structure of the data in a protocol buffer message format [62] which can be used to transfer data between connected vehicles and our servers regardless of their implementation language. As claimed by Google and as evaluated in [63], the Protocol Buffer leads to fast data transfer over the web comparing to eXtensible Markup Language or XML [39]. This is mainly because the Protocol Buffer uses binary format to serialize structured data.

According to Figure 7.7, the data exchanged between the connected vehicles and the intersection controller lies in three categories: 1) A vehicle transmitting (Un)Subscription messages to the intersection controller, 2) A vehicle transmitting its updates (velocity, distance, and desired access time) to the intersection controller, and 3) The controller sending the assigned access time and vehicle ID to the subscribed vehicle (may be repeated to update the assigned access time). We also add an emergency message in case two or more vehicles of conflicting movements are simultaneously within the safety area. This emergency message causes all vehicle already inside the safety area to stop as soon as possible and all vehicles outside the safety are to stop behind the safety area waiting till new access times are assigned and the alert is over. All the four aforementioned messages and their data structure are shown in Figure 7.8(a,b,c,d), respectively. The contents of each message is encoded and serialized by Google Protocol Buffers, and then is preceded with a predefined preamble before being sent. It should be emphasized that the preamble is unique for each message type, and, as a result, reveals the message type that a receiver has just received. The preamble should not be serialized; otherwise, the receiver should decode the received packet by trying all possible message type which can impose a huge performance penalty.

The interesting feature of google protocol buffers is that we can add new fields to our message formats without breaking backwards-compatibility [62]; the code can still read data encoded with the old format. Moreover, our intersection controller server (developed in Java) can easily share data with applications written in different languages (e.g. our virtual driver assistant developed in Objective-C).

7.5 Traffic Microsimulation

7.5.1 Simulation Setup

The simulation tools developed in this dissertation model the autonomous vehicles as agents that decide about their desired trajectory by their individual speed-planners. A two-phase/four-movement intersection is simulated, as previously explained in Chapter 6. Each intersecting road has one lane per direction, and no turning is allowed at the intersection.

Although, the ultimate goal is to evaluate our MILP-based intersection controller in a vehicle-in-the-loop (VIL) simulation setup, we also implement a software-in-the-loop (SIL) simulation environment. The SIL-simulation determines what the maximum achievable is, as there is no communication overhead and packet drop. The SIL- and VIL-simulations are similar except that the VIL-simulations run on a real-time basis, and the simulated vehicles in the VIL-configuration communicate with the intersection controller server utilizing Google Protocol Buffer messages and UDP unconnected datagram sockets. No communication is needed for the SIL-simulation as both the intersection controller and simulated vehicle run on the same computer. Please see Figures 7.1(a) and 7.2(b) for implemented SIL and VIL simulation environments.

The SIL and VIL microsimulations were both implemented using Java and its multi-threading capability. The real-time Java simulation and the traffic signal controller interact in such a way that the system can tolerate the simulation engine failures to meet its deadlines. In other words, if the simulation is overloaded

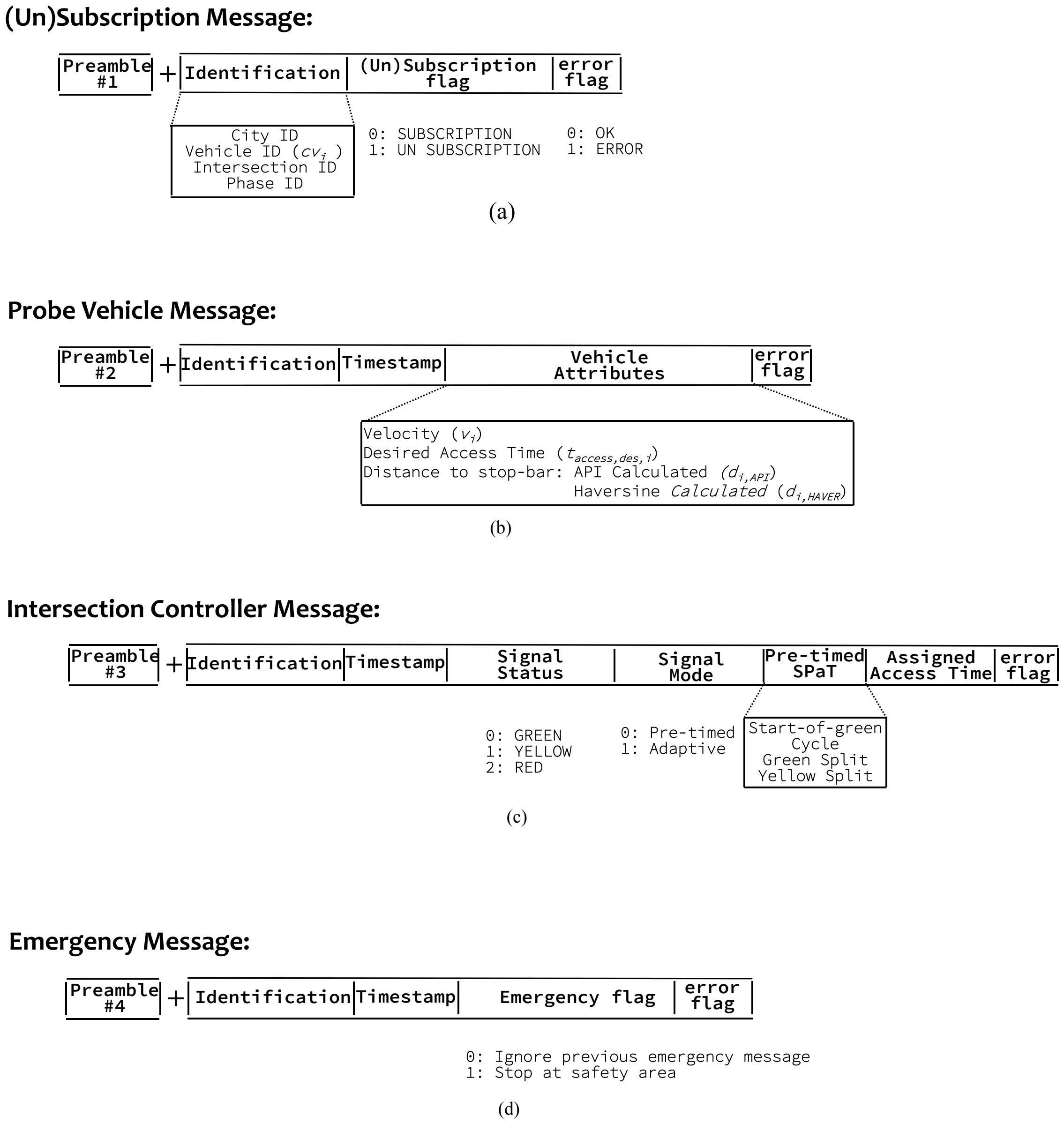


Figure 7.8: Four different message types implemented for the test environment (Protocol buffer structured messages preceded with a preamble).

and cannot model the dynamics of the simulated vehicles on time, the simulation is not interrupted and can still report the updated positional data of the vehicle but in a timestamp not at the prescheduled time.

7.5.2 Speed-Planner for the Microsimulations

Same as virtual driver assistant described in Section 7.3, the simulated vehicles compute their own trajectories by identical speed-planner engines. The speed-planners incorporated into the simulated vehicles and the one incorporated into the virtual driver assistant are similar and the main difference between them is that the speed-planning for simulated vehicles needs an autonomous car following model. The detailed speed-planning, which depends on our test scenarios, is explained in next chapter where we present out simulation and experimental results. The car following model is explained in Appendix A.

7.6 Acknowledgment

Some of the ideas presented in this section started to take shape when my advisor, Prof. Ardalan Vahidi, and I were both working at BMW Technology Office USA in Mountain View, CA. I would like to take this opportunity to thank Mr. Andreas Winckler from that office for sharing his insight and ideas, specifically, on speed recommendation display design, (un)subscription procedure, and intersection detection.

Chapter 8

Estimating Fuel Consumption using Vehicle Diagnostic Data

8.1 Introduction

To evaluate our MILP-based intersection control in improving intersection performance, a number of variables need to be analyzed during our road tests. We choose the fuel consumption of our test vehicle as the major measure of effectiveness (MOE). However, appropriate equipment are required to obtain the vehicle fuel consumption, so that the fuel consumption reduction of the implemented system can be evaluated. This dissertation obtains the vehicle's fuel consumption rate through OBDII port (On-Board Diagnostic port, version II). The On-Board Diagnostic (OBD) regulations requires the passenger cars to provide a minimum set of diagnostic information to off-board test equipment [123]. Usually, these diagnostic information do not include the fuel rate information; and, an estimation method is required to translate the real-time OBD data into the fuel flow of the engine. Some vehicles provide enhanced OBD data including the engine's fuel rate estimate but this enhanced data needs to be first evaluated, and may not be accurate [124]. Unfortunately, there is no formal fuel estimation methods suggested by relevant standards such as SAE J1979 [123] and ISO 15031-5 [125]; and diverse formulas are introduced in literature to estimate the actual fuel rate by OBDII basic data [124, 126, 127, 128, 129].

In this chapter, we propose two methods for estimating fuel consumption using the basic engine diagnostic information: 1) estimation based on engine's Mass Air Flow rate (MAF-based) 2) estimation based on engine's Fuel Injection Flow rate (FIF-based). Both methods are formulated for gasoline engines equipped with mass air-flow (MAF) sensor; however, they can be modified for engines with Manifold Pressure (MAP) sensors as well. We verified the accuracy of our methods by road tests. In each test, basic OBD data was collected by our developed iOS application that connects to a commercial Wi-Fi OBDII reader. The Vehicle used in all road tests is Honda Accord LX 2.4L 4-Cylinder SI gasoline engine with automatic transmission.

In this chapter, after a background introduction, we introduce our developed OBDII Logger application that runs on iOS devices. In Section 8.4, the methods proposed to identify the state of the engine via OBDII data are explained with on-road tests. We introduce our two methods for fuel estimation with experimental results in Sections 8.5 and 8.6. The details of the test criteria and methods used to obtain the actual fuel consumption of the test vehicle are provided in the last section.

8.2 Engine Fuel Management: Background

Engine fuel management systems have two major operating modes of closed-loop and open-loop:

Closed-loop: The main goal of the engine's Electronic Control Unit (ECU) in closed-loop mode of operation, is to control the air to fuel ratio mainly via feedback from the exhaust system. Fine tuning the air to fuel ratio by ECU, allows the vehicle's catalytic converter to reduce the pollutants contained in the exhaust gas most efficiently. The optimum ratio depends on the engine defects, and fuel impurities/compositions but is close to the stoichiometric gasoline combustion Air/Fuel Ratio of 14.64:1 ($AFR_{stoich} = 14.64$) [127]. Even in a perfect closed fuel loop engine operation, the actual air/fuel ratio (AFR) would deviate from the stoichiometric air/fuel ratio; and this deviation is usually denoted by $\lambda = \frac{AFR}{AFR_{stoich}}$.

The ECU commands to have a rich mixture ($\lambda < 1$), lean mixture ($\lambda > 1$), or stoichiometric mixture ($\lambda = 1$) based on its inputs from various sensors including the oxygen (O_2) sensor for exhaust gas. The ECU-commanded-λ can be slightly different from the actual-λ interpreted from the oxygen sensor readings [128]. The ECU-commanded-λ is available via basic OBD data, namely, OBD Commanded Equivalence Ratio (EQR). Based on the aforementioned explanations, if the actual-$lambda$ and the mass air flow into the engine (MAF) are available then a fairly accurate estimate for the fuel flow is $\frac{MAF}{14.64 \times \lambda}$.

The ECU commanded air/fuel ratio is then translated to the desired fuel mass for each cylinder, and, subsequently, the appropriate fuel injector pulse width [130]. The OBD fuel trim (FT) parameter is a numerical multiplier that denotes by what scaling factor the ECU is modifying the injectors base pulse width. The ECU modifies this base pulse width according to the percentage of oxygen present in the exhaust gases.

As an example, a fuel trim of 0% means that the injectors' pulse width needs no modification compared to the base pulse width, a positive FT of 10% means that the pulse width needs to be increased (10% more fuel than the calibrated amount) in a lean mixture condition, and a negative FT of -10% means that the pulse width needs to be decreased (10% less fuel) in a rich mixture condition. The total fuel trim actually consists of two separate OBD values: summation of 1) OBD Short Term Fuel Trim ($SHORTFT$), and 2) OBD Long Term Fuel Trim ($LONGFT$). The OBD short term fuel trim is the immediate or on-the-fly ECU's response, and it fluctuates above and below zero as the vehicle's operating conditions vary. On the other hand, the OBD long term fuel trim is learned by the deviations corrected by short term fuel trim over the operational lifetime of a vehicle; and is stored in a non-volatile memory [130]. As an example, when the short term fuel trim hits its upper/lower limit, the ECU increments/decrements the long term fuel trim by one count.

Open-loop: The closed-loop operation is not always possible. Generally, the following conditions causes an open-loop mode of operation:

- **Cold-start**: The closed-loop mode is not possible during a cold-start because the oxygen sensor should reach a certain temperature before it can be functional. The cold-start period depends on the type and temperature of the oxygen sensor and is short for the oxygen sensors equipped with built-in heaters. DeFries et al. [124] reported that their test vehicle (2009 Saturn Outlook) was running lean during cold start. This lean condition could be the result of fast idle control and spark retard technologies used in some vehicles to warm the catalyst up quickly after cold starts [124]. They obtained $\lambda = 1/0.77$ as this value best fit their dyne-measured fuel rates. We used the same value in this dissertation, although we had a different test vehicle.

- **WOT**: Wide Open Throttle (WOT) is when the gas pedal is fully or almost-fully depressed, and the engine runs in full load condition. During WOT, the mass air flow (MAF) is large, and the OBD calculated engine load value ($LOADPCT$) reaches about 100%. In WOT condition, the engine does not account for the exhaust oxygen percentage because it needs a rich mixture to reach the maximum power. The intake air/fuel ratio is a function of engine RPM [128], and can be in the range of 10 to 14 during WOT [131, 132].

- **Deceleration fuel cutoff**: Fuel is entirely cutoff by ECU when the accelartor pedal is released to cruise or decelerate. However, based on our observations, it seems that fuel is not cutoff when the vehicle is going below a certain speed or the engine is running below a certain RPM.

8.3 iOS-based OBDII Data Logger

An iOS application has been developed that logs the vehicle On-board diagnostics (OBDII) data as well as the iOS device sensor data. As shown in Figure 8.1, the OBD Log application can connect to commercial Wi-Fi OBDII readers supporting ELM327. In fact, there are several OBD protocols that none of them are directly usable by PCs or smart devices; and ELM327 is a chip programmed by ELM Electronics that acts as an interpreter bridge between these OBD protocols and our iOS device [133]. At this time, our developed iOS application is compatible with 29-bit CAN protocol (ISO 15765-4 [134]). Future releases of our smart phone application will support other protocols such as 11-bit CAN [134] and SAE J1850 [135].

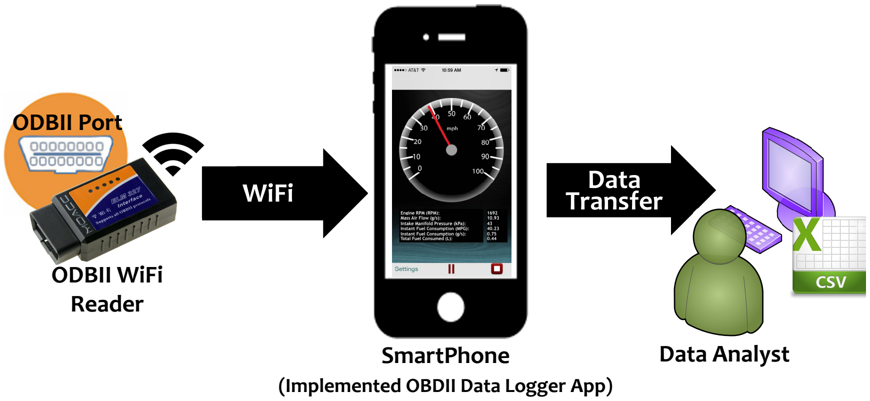


Figure 8.1: Functional architecture of the developed iOS OBD Logger App.

The OBD Log users can select up to six individual OBD data values that they are interested to monitor. The OBD Log App then requests and collects those six data values at 2 Hz max. The collected data not only is shown real-time on the display but also will be available in a Comma Separated Values file format through iTunes for further off-line data analysis (see Figure 8.1). Our proposed fuel estimation methods are incorporated into the OBD Log App using only the basic OBD data as given in Table 8.1.

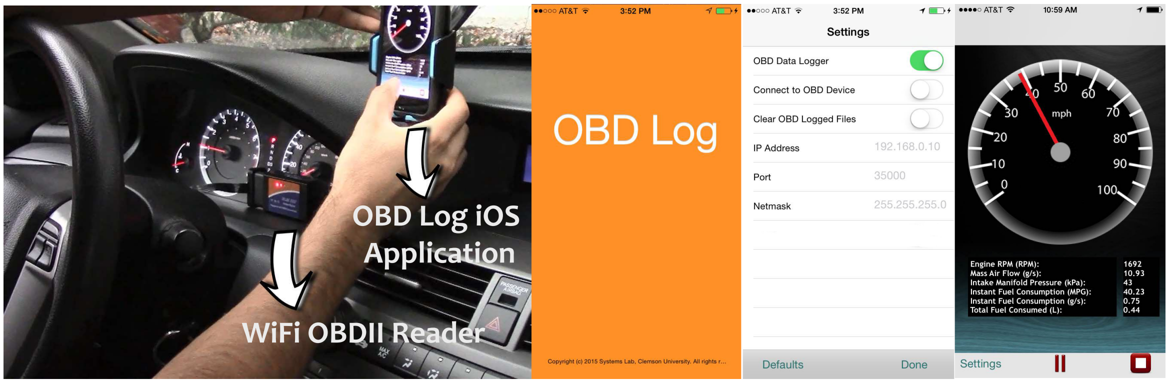
Figure 8.2(a) shows the setup in our test vehicle where the Wi-Fi OBDII reader is connected to OBD port by a cable. Figure 8.2(b) demonstrates the designed user interface for the main and settings pages. The designed user interface provides Play/Pause buttons so that the driver, driving in our experimental testbeds, can control when to collect data (iOS GPS and fuel consumption information).

Although several commercial applications are available for OBD data logging, we built a custom OBD logger from scratch because: 1) our proposed fuel consumption estimator needed to be incorporated into the application; 2) we needed to request all the possible six OBD data values in one packet and at the

Table 8.1. Basic OBD Data used for fuel rate estimation .

OBD Data	Notation	PID*
Fuel System Status	$FUELSYS$	0x03
Commanded Equivalence Ratio	EQR	0x44
Mass Air-flow (g/s)	MAF	0x10
Short Term Fuel Trim (%)	$SHRTFT$	0x06
Long Term Fuel Trim (%)	$LONGFT$	0x07
Calculated Load Value (%)	$LOADPCT$	0x04

* PIDs (Parameter IDs) are codes used to specify OBD data.



OBD Data Logger ● On
Connect to OBD Device ○ Off
Clear OBD Logged Files ○ Off
IP Address 192.168.0.10
Port 35000
Netmask 255.255.255.0

Engine RPM (RPM):	1692
Mass Air Flow (g/s):	10.93
Intake Manifold Pressure (kPa):	43
Instant Fuel Consumption (MPG):	40.23
Instant Fuel Consumption (g/s):	0.75
Total Fuel Consumed (L):	0.44

(a) (b)

Figure 8.2: The implemented OBD Log application: (a) test vehicle setup (b) designed user interface.

highest frequency possible which is not always provided by the commercial loggers; 3) Play/Pause buttons were needed to freeze or continue data collection during our road tests.

8.4 Engine State Identification

The goal of this section is to identify the engine's mode of operation based on basic OBD data. As it was mentioned as a background introduction in Section 8.2, the engine has two modes of operation, namely, closed-loop and open-loop fuel control. We are particularly interested in identifying the conditions leading to open-loop mode (non-stoichiometric operation). Identifying those conditions (cold starts, high load (WOT), and fuel cutoff) and correcting our fuel rate estimation method accordingly, will reduce the overall estimation errors. It is possible to track the engine status via OBD's Fuel System Status data, denoted as $FUELSYS$ in Table 8.1. However, the OBD standard descriptions on $FUELSYS$ [123, 125] do not discuss the possible conditions causing an open-loop operation; and we found out those specific conditions by conducting road tests. For each possible value for $FUELSYS$, we summarize the standard descriptions as well as our findings in Table 8.2.

Table 8.2. Partial identification of (non)stoichiometric conditions using OBD's Fuel System Status data ($FUELSYS$) .

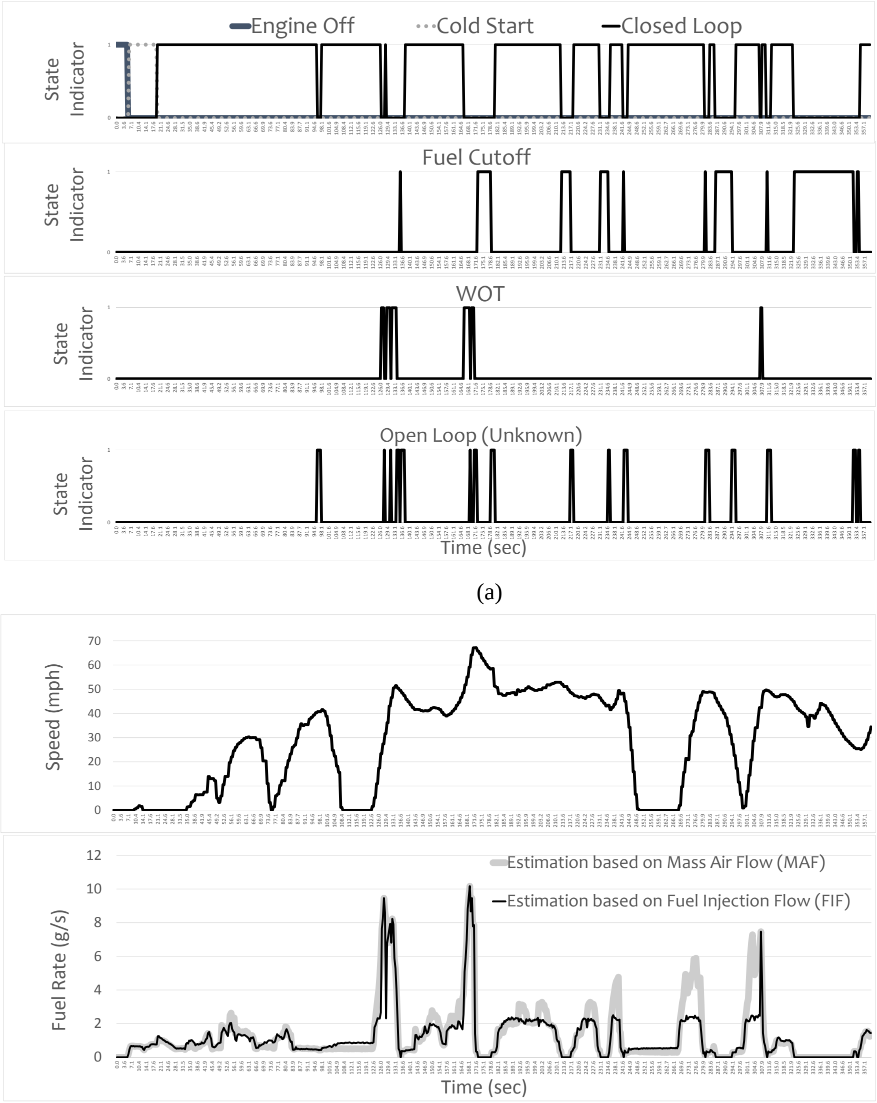
$\boldsymbol{FUELSYS}$	**OBD standard description** [123, 125]	**Road test findings**
0x00	-	Engine is Off (Open-Loop)
0x01	Open-Loop due to unsatisfied conditions	Cold start || Open-Loop due to unknown reasons
0x02	Normal Closed-Loop	-
0x04	Open-Loop due to driving conditions	Fuel cutoff || WOT*
0x08	Open-Loop with error (Fault**)	-
0x10	Closed-Loop with error (Fault**)	-

* WOT: Wide Open Throttle.

** Fault conditions never happened during road tests.

We also had other three observations during our road-tests that, if combined with the information of Table 8.2, can reveal the full working conditions of the engine: 1) Open-loop operation because of cold start only happens during the first few seconds after starting the car. Considering this fact, if $FUELSYS = 0x01$ and the run time since engine start is long enough (e.g. 60 sec) that we are sure the oxygen sensor has reached its normal operating temperature, then we flag the engine state as open-loop but by unknown reasons (Open-Loop (Unknown)). 2) During open-loop operation due to fuel cutoff, the car manufacturer feeds a default value to OBD commanded equivalence ratio, $EQR = 1.998817 \approx 2$. This is also observed by DeFries et al. in [124]; 3) High load or wide open throttle can be identified when engine is in open-loop due to driving conditions ($FUELSYS = 0x04$) and, simultaneously, air flow sensor indicates high load (e.g. $MAF \geq 50$ g/s).

Based on the aforementioned descriptions, a logic can be constructed for engine state identification using basic OBD data. We applied this logic on an OBD data set collected by our iOS data logger and plotted the engine state indicators versus time in Figure 8.3(a). This figure shows that all the indicators given in last column of Table 8.2 can be identified individually. No fault condition occurred during our road tests. It should be emphasized that WOT is when the gas pedal is fully or almost-fully depressed and, as a result, WOT does not usually occur in normal driving conditions. In this specific road test, depicted in Figure 8.3(a), we deliberately caused WOT by sudden accelerations for demonstration purposes only. Figure 8.3(b) demonstrates estimated fuel rates based on the identified engine state. Two proposed fuel rate estimation methods will be introduced later in Section 8.5 and 8.6.



(a)

(b)

Figure 8.3: Preliminary road test results: (a) engine state indicators, identifed based on basic OBD data (b) estimated fuel rate using the proposed methods.

8.5 Estimation based on Mass Air Flow rate (MAF-based)

As it was explained in Section 8.2, the stoichiometric or optimum air to fuel ratio (AFR_{stoich}) is about 14.7 for gasoline engines. However, the engine control unit constantly corrects this ratio via feedback from exhaust system. The goal of this section is to find the correction factor, namely λ, so that we can obtain the MAF-based Fuel Rate ($FR_{MAF-based}$) as given in Equation (8.1). One solution seems to be $\lambda = EQR$, where EQR is the OBD commanded equivalence ratio. However, EQR is the λ value commanded from ECU and, as observed in [128], it is slightly different from the λ measured from oxygen sensor. This causes a fuel flow estimation error that can be accumulated in cumulative fuel usage tracking. As an example, assuming $\lambda = EQR$ in all operating modes of the engine, the cumulative fuel usage was underestimated by 12.7% and 13.5% in our test drivings of 43 miles and 70 miles, respectively. In contrary to what suggested in [129], applying OBD calculated load value ($LOADPCT$) in addition to EQR did not lead to accuracy improvement, and even resulted in doubled error.

$$FR_{MAF-based} = \frac{MAF}{AFR_{stoich} \times \lambda} \tag{8.1}$$

Based on the aforementioned explanation and road tests, we propose to assume $\lambda = EQR$ only in open-loop mode of operation caused by high load (WOT) or unknown reasons. In fact, the error in commanded equivalence ratio can be ignored during these modes of operation because they do no occur as frequently as the closed fuel loop operation does. For the closed-loop mode of operation, we utilize the short and long term fuel trim data ($SHRTFT$ and $LONGFT$) as air to fuel correction factors. The fuel trims are essentially the fuel correction factors and they modify the pulse width of the fuel injector; what we are proposing here is that if ECU is modifying the injector's base pulse width for e.g. 10% more fuel because of a lean condition, then the engine is perhaps operating 10% leaner than the stoichiometric air-fuel ratio. As another example, -10% less fuel injection corresponds to -10% richer mixture than the stoichiometric mixture. As a result, we set $\lambda = 1 + \frac{(SHRTFT+LONGFT)}{100}$ in closed-loop mode of operation. This is similar to the formula presented in [127], but we have incorporated the short term fuel trim into the formulation also. The summary of our MAF-based estimation logic is given in Table 8.3 and a sample fuel flow estimation is shown in Figure 8.3(b) in a solid light-color line. Please note that Figure 8.3(b) includes some deliberate WOT intervals. The air to fuel ratio can be in the range of 10 to 14 during WOT [131, 132]; and considering a very large air flow of MAF=120 g/s, the obtained peak fuel flow rate would be about 12 g/s during WOT which is consistent with Figure 8.3(b).

We applied our test criteria, explained later in Section 8.7, to the MAF-based fuel flow estimation method. We conducted five road tests in a three months period and the results are given in Table 8.4. For comparison reasons, we applied the fuel estimation methods of [124, 126, 127] on Test-5 and we obtained 10.38%, 12.8%, and 5.8% under-estimation errors in estimated cumulative fuel usages, respectively. It should be emphasized that no explanation was found in [126] regarding the engine state identification, and the approach of [127] assumes that engine always stays in closed loop mode after cold start [136]. We added an

Table 8.3. MAF-based fuel flow estimation summary .

Identified engine state	**Assigned λ**
Engine Off || Fuel cutoff	$\lambda = \infty$ (zero fuel flow)
Cold start	$\lambda = 1/0.77$
WOT || Open-loop (Unknown) || Fault	$\lambda =$ commanded equivalence ratio (EQR)
Closed-loop	$\lambda = 1 + \frac{(SHRTFT+LONGFT)}{100}$

extra fuel cutoff identification to these two works before obtaining their fuel estimation errors.

Table 8.4. Road test results for fuel rate estimation based on mass air flow rate .

Test No.	Actual* cumulative fuel used	Engine-starts counts**	Estimated cumulative fuel used*** by proposed MAF-based method
Test 1	2.34 (Liter)	5	2.14 (Liter) (-8.5%)
Test 2	3.41 (Liter)	8	3.49 (Liter) (2.35%)
Test 3	9.26 (Liter)	21	9.11 (Liter) (-1.6%)
Test 4	11.51 (Liter)	21	10.69 (Liter) (-7.1%)
Test 5	47.03 (Liter)	53	45.17 (Liter) (-4%)

* Actual fuel usages were measured by re-filling the test vehicle's tank at gas station.

** Fuel used during engine startups are considered in the estimations.

*** Gasoline density is set 739.3 g/L in calculations.

8.6 Estimation based on Fuel Injection Flow rate (FIF-based)

Some vehicles provide enhanced OBD data which includes the engine's fuel rate estimate; however, this OEM-specific data needs to be first evaluated, and may not be accurate enough for our application. As an example, DeFries et al [124] found some incorrect fuel rates reported from a 2012 Toyota Camry's enhanced OBD fuel rate. Furthermore, not all vehicles, like our test vehicle, provide the fuel injector rate as an OEM-specific parameter. What we are proposing here is a calibration-based method that estimates the real-time fuel injection flow rate of the engine using only basic OBD data. We name the obtained fuel flow rate (FR) as Fuel Injection Flow-based or FIF-based rate ($FR_{FIF-based}$).

The proposed approach is based on measuring the base fuel injection flow of the vehicle via calibration and translating this base flow to an actual fuel flow. Base fuel rate can have different definitions based on the model formulations. Here, by base fuel injection rate (FR_{base}), we mean the summed total fuel rate (mass per second) of all injectors under maximum engine load and zero fuel trims. This base fuel flow rate is compensated by the engine ECU based on the information obtained from many sensors. We assume that the base fuel flow is adjusted by the ECU using multiplicative feedback corrections as:

$$FR_{FIF-based} = FR_{base} \times (1 + \frac{(SHRTFT + LONGFT)}{100}) \times \frac{LOADPCT}{100} \times X \tag{8.2}$$

where $1 + \frac{(SHRTFT+LONGFT)}{100}$ denotes by what scaling factor the ECU is modifying the base fuel flow rate according to the percentage of oxygen present in the exhaust gases; and $\frac{LOADPCT}{100}$ denotes by what scaling factor the ECU is modifying the base fuel flow rate according to the engine load percentage. All other possible factors contributing to fuel injection flow are denoted by X. These factors, such as engine coolant temperature and battery voltage, are assumed to have a very minimal contribution to the fuel delivery. As a result, in this dissertation, we set X=1.

By calibration, we mean the estimation of the base fuel flow rate of the vehicle (FR_{base}). In order to calibrate the model, the user should fill the fuel tank, drive the car while OBD Log application is logging data, refill the fuel tank at the same fuel dispenser, and record the total fuel used. Then the logged data is used to calibrate the model and find the best base fuel flow value that results in minimal commutative fuel estimation error for the specific vehicle.

The summary of our FIF-based estimation logic is given in Table 8.5 and a sample fuel flow estimation is shown in Figure 8.3(b) in a solid dark-color line. We applied our test criteria, explained above and also later in Section 8.7, to the FIF-based fuel flow estimation method. We conducted five road tests in a three months period and we found the best base fuel flow for each test. The obtained FR_{base} values are 2.79, 2.71, 2.78, 2.74, and 2.74, for five road tests. As a result, we set FR_{base}=2.75 g/s as our best calibration result. The estimation results using FR_{base}=2.75 g/s are given in Table 8.6. We did not find any similar calibration-based method in literature to be compared to the results obtained in this section.

8.7 Test Criteria

In this section, we describe the test criteria that we constructed to obtain the fuel estimation errors reported in previous sections of this chapter. Each test was initiated by filling the fuel tank at a filling station with automatic fuel dispensing nozzles. An automatic fuel dispensing nozzle can shut down the flow of gas

Table 8.5. FIF-based fuel flow estimation summary .

Identified engine state	**Assigned Fuel flow rate ($FR_{FIF-based}$)**
Engine Off || Fuel cutoff	$FR_{FIF-based} = 0$
Cold start	$FR_{FIF-based} = \frac{MAF}{AFR_{stoich} \times (\lambda = 1/0.77)}$
WOT || Open-loop (Unknown) || Fault	$FR_{FIF-based} = \frac{MAF}{AFR_{stoich} \times (\lambda = EQR)}$
Closed-loop	$FR_{FIF-based} = FR_{base} \times (1 + \frac{(SHRTFT + LONGFT)}{100}) \times \frac{LOADPCT}{100}$

Table 8.6. Road test results for fuel rate estimation based on fuel-injector .

Test No.	Actual* cumulative fuel used	Engine-starts counts**	Estimated*** cumulative fuel used by proposed FIF-based method
Test 1	2.34 (Liter)	5	2.30 (Liter) (-1.7%)
Test 2	3.41 (Liter)	8	3.46 (Liter) (1.5%)
Test 3	9.26 (Liter)	21	9.15 (Liter) (-1.2%)
Test 4	11.51 (Liter)	21	11.55 (Liter) (0.3%)
Test 5	47.03 (Liter)	53	47.27 (Liter) (0.5%)

* Actual fuel usages were measured by refilling tank at gas station.

** Fuel used during engine startups are considered in the estimations.

*** Estimated fuel usages were obtained by setting FR_{base}=2.75 g/s.

when the tank is almost full. The weather temperature and the fuel dispenser identity (gas pump number) were recorded at the time of filling. The test was followed by day-to-day driving for a period of time with the OBD Log application running on an iPhone device. The test was finalized by re-filling the fuel tank at the same fuel dispenser assuming that the auto fuel-shutoff sensitivity may differ from nozzle to nozzle even on the same station. We also assumed that the nozzle's auto fuel-shutoff sensitivity may be altered based on the outside temperature because the fuel shutoff is usually accomplished by mechanical valves detecting change of pressure. As a result, we scheduled the re-filling day in such a way that outside temperature would not be very different form that of the filling day. At the end, the estimated cumulative fuel usage was obtained by downloading and analyzing the OBD log files. The estimation was then compared with the amount of fuel used to refill the fuel tank.

All tests were conducted by one driver using the test vehicle as his personal vehicle for day-to-day driving. Because of this, a single test would include multiple engine-starts. Our fuel flow estimation methods do not consider the fuel used during a startup transient; however, the counts of engine-starts are reported in Tables 8.4 and 8.6 for possible future research.

8.8 Acknowledgment

I thank the technical staff members of the Mechanical Engineering department at Clemson University, Jamie Cole, Michael Justice, and Stephen Bass, for their forbearance and help throughout this long project. The staff kindly provided equipment and accommodated my request to conduct my OBD Log application development in a garage facility. Mohammad Fazelpour's help in video recording and photography is also appreciated.

Chapter 9

Experimental and Simulation Evaluation

9.1 Introduction

It is difficult to implement an appropriate benchmark for our MILP-based intersection control. The main reason is that other algorithms presented in literature, such as [71, 73, 75, 76, 83, 20, 84, 74, 82], are verified for different traffic networks and different traffic patterns and, as a result, are not useful for benchmarking and comparing our MILP-based intersection control. As a possible future work, the most relevant algorithms in literature, such as [74, 82], can be used as benchmarks only if their control programs are replicated and verified in the same manner that our proposed MILP-based algorithm is verified.

In an attempt to come up with custom benchmarks for the presented MILP-based algorithm, an extensive review of the published literature regarding planning the vehicles' longitudinal trajectories and planning their timely arrival at signalized intersections was performed through the following two deterministic paradigms:

- **Vision-based Speed-Planning:** Based on the observed current state of the signals ahead (green, yellow, or red), drivers can adjust their speed. This speed adjustment can be, for example, releasing the accelerator pedal and minimal use of braking while approaching a red light, cruising to hopefully bypass a red signal, or speeding up slightly and safely to get through a green light. In fact, these are few fuel-efficient driving techniques that any driver would use as soon as they can see a traffic signal on their way. Alternatively, these techniques would be used when traffic signals are not connected, and autonomous vehicles are equipped with a camera-based traffic signal state detection. There has been a large amount of research in the area of traffic light detection using on-board cameras and fast image/video scanning such as [137, 138]. However, this vision-based speed control is limited to the operating range of the cameras and can be affected by environmental conditions as well. Similar limitations apply to the vision range of the drivers driving conventional vehicles.

- **Communication-based Speed-Planning:** By communicating the deterministic future state of traffic signals, the connected vehicle's speed and its longitudinal trajectory can be controlled for timely arrivals at intersections. The idea of speed advisory is not new and has previously been presented in [6, 55, 139, 140, 141, 142, 143]. The communication-based speed planners can be programmed into the in-vehicle embedded systems of autonomous vehicles, or can be implemented in virtual driver assistants to manually guide drivers [58, 114].

Please note that the aforementioned speed controls can also be based on the probabilistic SPaT information. Koukoumidis et al. in [7] predict the future schedule of traffic signals based on camera detected traffic signals. Mahler et al. in [43, 32] and Apple et al. in [9] calculate the probability of green light for each traffic signal assuming that the current status updates are communicated from the city traffic management centers. Nevertheless, the probabilistic knowledge of SPaT information is not in the scope of this dissertation.

Based on the aforementioned introduction, our proposed MILP-based intersection control requires a communication-based speed-planning; however, instead of communicating the future state of traffic signals, the optimal arrival of each vehicle at intersection is communicated. The speed-planning algorithm then needs to be executed separately and locally by individual autonomous connected vehicles. The MILP-based algorithm is tested with two benchmark road tests; we study: 1) autonomous vehicles with vision-based speed-planning and 2) autonomous vehicles with communication-based speed-planning, approaching a pre-timed traffic signal control as our benchmark experimental testbeds. In all of our testbeds, we use the same virtual driver assistant, explained in Section 7.3, but, for each testbed we individually customize the information shown to the driver (driver assistant's HMI) as well as the speed-planner algorithm embedded in the driver assistant. To demonstrate that our MILP-based intersection control can improve intersection performance compared with the aforementioned benchmarks in the study, a number of variables need to be analyzed. As it was explained in previous chapter, the fuel consumption of the real test vehicle is chosen as the major measure of effectiveness (MOE).

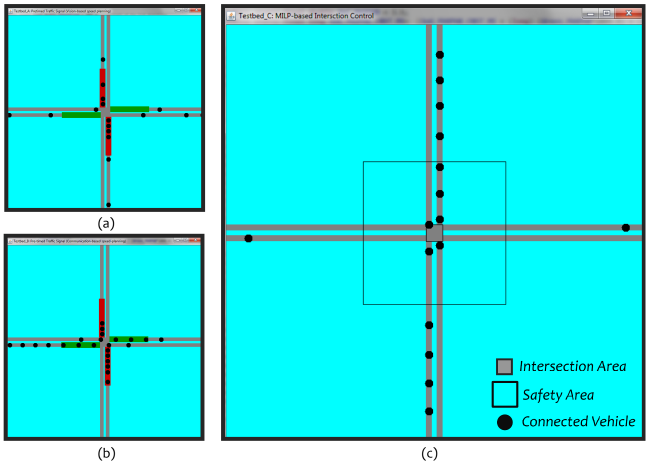

Testbed_A: Pretimed Traffic Signal (Vision-based speed planning)
(a)
Testbed_B: Pre-timed Traffic Signal (Communication-based speed-planning)
(b)
Testbed_C: MILP-based Intersction Control
Intersection Area
Safety Area
Connected Vehicle
(c)

Figure 9.1: Screenshots of the implemented simulation testbeds using Java; (a) Pre-timed signalized intersection with no communication (Testbed A) (b) Pre-timed signalized intersection with unidirectional communication and speed-advisory (Testbed B) (c) MILP-controlled intersection with no traffic signal and with bidirectional communication (Testbed C).

In this chapter, first we describe all the test scenarios and benchmarks in Section 9.2. Each test scenario requires a customized speed-planning which is explained in the same section for each test scenario. Then Section 9.3 introduces the measures of effectiveness (MOE) used to evaluate the success of the MILP-based intersection control compared with the benchmarks. The comparison results are given in last two sections via SIL and VIL simulations, respectively.

9.2 Test Scenarios

9.2.1 Testbed A: Vision-based Speed-Planning at Pre-timed Signalized Intersection

Autonomous vehicles approaching a pre-timed traffic signal control provide a baseline testbed, against which we can benchmark and compare our MILP-based intersection controller. In this specific testbed, we also assume that the autonomous vehicles are equipped with a camera-based traffic signal state detection. As it was previously mentioned in Section 9.1, this means that these vehicle can only observe the current state of the signals ahead, like the drivers of conventional vehicles do. In order to model this test environment, we set our simulation program and our virtual driver assistant in such a way that vehicles will receive the traffic signal-related speed advice from their speed-planning engines only when they are within the range of their imaginary camera. We set this range as 300 m as measured from the signal ahead. A screenshot of the implemented simulation environment is shown in Figure 9.1(a).

The goal for an autonomous vehicle equipped with a camera-based traffic signal state detection is only to adjust speed for an efficient stop at red or a safe pass though green light. As a result, the customized speed-planning algorithm for this testbed is simple with features as follows:

- In free flow, the vehicles drive at average speed v_{avg}.
- As soon as a vehicles is within the range of its imaginary camera (300 m), the current state of the simulated traffic light is fed into its speed-planner. In case of a red light, the vehicle decelerates slowly to a low cruising speed (e.g 25 mph) to hopefully bypass the red signal without stopping. In case of a green light, the vehicle maintains its speed; alternatively, if the vehicle is moving slowly and the posted speed limit allows, the vehicle speeds up slightly to get through the green light.
- While approaching the intersection, the speed-planning engines constantly estimate the distance required to stop the vehicles before crossing the intersection. In order to avoid sudden slowdowns for fuel saving, this distance is calculated considering a low to medium deceleration (2 m/s^2) as follows:

$$d_{stop} = t_{res}.v_i + \frac{0 - v_i^2}{2.a_{dec}} \tag{9.1}$$

  where t_{res} = 0.5 sec is the response time of an autonomous vehicle, v_i is the current speed of the connected vehicle, and a_{dec} = -2 (m/s^2) is the desired declaration rate considered for passenger cars in this part of the dissertation.
- Upon reaching the stop distance from the intersection (calculated by Equation (9.1)), if the light is still red even after cruising in low speed, then the vehicle decelerates to come to a complete stop at the intersection. If the light turns from green to yellow while the vehicle is within d_{stop} then the speed-planner decides to continue at the current speed through the intersection or come to a complete stop based on its distance to the intersection.

The aforementioned speed-planning algorithm was then incorporated into the simulations. Furthermore, the signal phase and timing (SPaT) information for the simulated pre-timed traffic signal was obtained off-line from SYNCHRO (Trafficware 2011) optimization program. This optimized SPaT was then used in the microsimulation to model the current state of the traffic light detected by the vehicles' imaginary camera. The SIL/VIL simulation results of this testbed are given in Sections 9.4 and 9.5.

9.2.2 Testbed B: Communication-based Speed-Planning at Pre-timed Signalized Intersection

We also consider a benchmark algorithm, namely communication-based speed-planning at pre-timed signalized intersection, where all autonomous vehicles are assumed to be able to receive the deterministic future state of traffic signals via unidirectional wireless communications. Although, the communication between the vehicles and a local signal controller can be achieved by Dedicated short Range Communication (DSRC), we assume that vehicles are connected to a signal controller server via cellular networks technology because of the limited communication range of DSRC. In order to model this test environment, we set our SIL simulation program in such a way that vehicles will receive the traffic signal phase and timing information (SPaT) when they are within 2 km of the intersection. However, we decrease this range to 1.2 km in our VIL simulation program because of the test track limitations. For our real vehicle, we set the radius of the intersection monitoring region to 1.2 km as shown in Figure 7.6; and the iOS driver assistant subscribes to the intersection and receives the corresponding SPaT as soon as the vehicle enters the intersection monitoring region. A screenshot of the implemented simulation environment is shown in Figure 9.1(b).

The goal for an autonomous vehicle that is approaching a pre-timed signalized intersection and is equipped with a communication-based speed-planning is to calculate velocity trajectories that reduce idling time at red signals and therefore improve fuel efficiency. As a result, a customized speed-planning algorithm for this testbed is implemented here by modifying the speed advisory proposed by our group in [6]. The programming details are not included here to keep the section brief, and only an example is depicted in Figure 9.2 to demonstrate our speed-planning algorithm. Figure 9.2 shows the feasible speed intervals that a vehicle traveling at a speed of v_i can follow without stopping at red signals. These speed intervals are limited to the speed limit v_{max} (e.g. 45 mph) and the minimum cruising speed possible v_{min} (e.g. 20 mph). In this dissertation, we add a constant safety buffer Δt=4 sec after each start-of-green so that a vehicle would never cross the stop-bar at red signal. Among the available speed intervals shown in Figure 9.2, we choose a target speed as close as possible to the desired average speed v_{avg}.

At this testbed, we are assuming that the autonomous vehicles have information about the instantaneous queue size when they are within 300 m of the intersection. If the queue information is available then the vertical axis of Figure 9.2 is changed from the distance to stop-bar to the distance to the rear end of the queue. In this case, the safety buffer Δt, added after each start-of-green, compensates in part for the queue dissipation time.

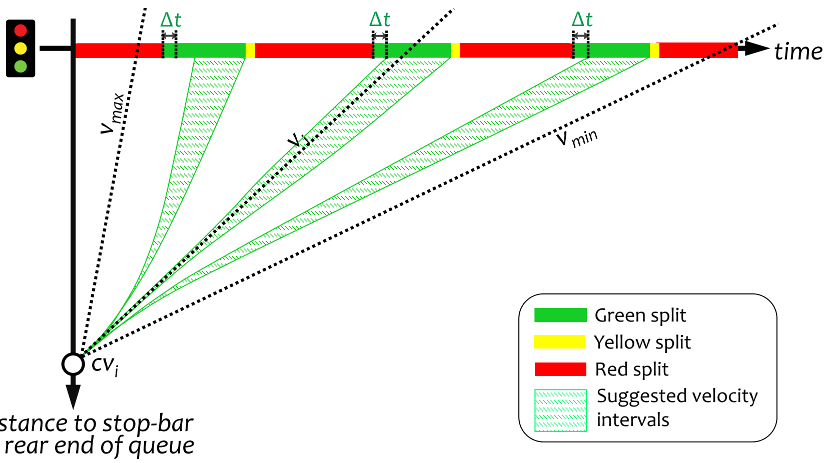


Figure 9.2: Schematics map of green splits distributed over space-time. The graphics shows how a speed-planner find the feasible velocity intervals in order to avvoid stopping at red.

The vehicles equipped with speed advisory system, form platoons to pass through green splits, although platooning is not incorporated into the speed-planning algorithm. It should be emphasized that our speed advisory algorithm does not take the vehicles that are following a platoon's lead vehicle into account. This means that the lead vehicle may adjust its speed to pass at almost the end of a green split, ignoring all following vehicles that may get stuck at the upcoming red signal.

The aforementioned speed-planning algorithm was then incorporated into the simulations as well as the iOS driver assistant. Furthermore, the same signal phase and timing (SPaT) information of testbed A was used here for the virtual pre-timed traffic signal. This SPaT (optimized by SYNCHRO (Trafficware 2011)) was communicated to the vehicles during road tests and simulations so that they could adjusts their speed based on future state of the virtual traffic signal. The SIL/VIL simulation results of this testbed are given in Sections 9.4 and 9.5. It should be emphasized that we are assuming that the autonomous vehicles, traveling in this testbed, do not have information about future queue size that they would face at their arrival times. The potential gain in traffic flow, if the future queue information is integrated in the algorithm, is left as future research.

9.2.3 Testbed C: Communication-based Speed-Planning at MILP Controlled Intersection

The testbed explained in this subsection includes our proposed MILP-based intersection control. This testbed will be benchmarked and compared against Testbeds A and B which were explained in previous subsections. Similar to Testbed B, we assume all autonomous vehicles are able to receive data via wireless communications; however, this testbed is expanded to include two-way communication between moving connected vehicles and upcoming MILP-based intersection controller. The exchanged data mainly include the access times that the signal controller server assigns to each approaching vehicle as well as the geographical data of the connected vehicles wirelessly transmitted in real-time to the MILP-based intersection controller. In order to model this test environment, we set our SIL simulation program in such a way that the simulated vehicles will subscribe to the intersection controller when they are within the range of 2 km from the intersection. However, we decrease this range to 1.2 km in our VIL simulation program because of the test track limitations. For our real vehicle, we set the radius of the intersection monitoring region to 1.2 km as shown in Figure 7.6; and the iOS driver assistant subscribes to the intersection and receives the corresponding access time as soon as the vehicle enters the intersection monitoring region. A screenshot of the implemented simulation environment is shown in Figure 9.1(c).

The goal for an autonomous vehicle connected to our MILP-based intersection controller is to reach the intersection access point at its assigned access time. As a result, a customized speed-planning algorithm for this testbed is sought to be incorporated into our simulations to guide the simulated vehicles and the real vehicle accordingly. This simplified algorithm actually plans the vehicles' longitudinal trajectories in such a way that each individual vehicle reaches the access area (safety area) at it assigned timestamp. The basic fact to be considered before explaining the algorithm is that if a vehicle's remaining travel time to the intersection safety area at current speed is equal to the remaining time to its assigned access time, then, obviously, it is not necessary to adjust its speed. As shown in Figure 9.3(a), this means that the vehicle can continue driving at its current speed if $\Delta t = t_{access,i} - t_0 = \frac{d_i}{v_i}$, where d_i and v_i are the current distance and velocity at current time t_0, and $t_{access,i}$ is the assigned access time. Most often, however, the speed-planning engine should consider acceleration/deceleration maneuvers to adjust the vehicle speed at the following two situations:

- **Acceleration**: If $t_{access,i} < t_0 + \frac{d_i}{v_i}$ then the vehicle cannot make it to the access point at its assigned access time, if it keeps driving at its current speed. As a result, the vehicle needs to accelerate to a cruising speed. As shown in Figure 9.3(b), the vehicle should keep that speed ($v_{cruise,i}$) till it reaches the intersection area. Knowing the desired acceleration rate a_{acc}, the cruising speed can be calculated by:

$$v_{cruise,i} = v_i + a_{acc}\Delta t - \sqrt{2a_{acc}(\frac{1}{2}a_{acc}\Delta t^2 + v_i\Delta t - d_i)} \tag{9.2}$$

where Δt is the remaining time to the assigned access time, d_i is the remaining distance to the access

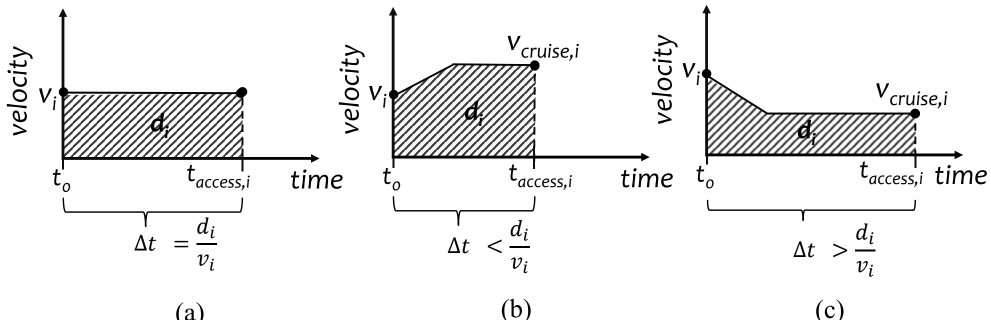


Figure 9.3: Preliminary speed-planning for vehicles that can pass the MILP controlled intersection with no stop.

point, and v_i is the vehicle's velocity at current time. We set $a_{acc} = 2m/s^2$ as the desired acceleration rate of vehicles only if it satisfies the condition of $2m/s^2 \geq \frac{2(d_i - v_i\Delta t)}{\Delta t^2}$ to avoid taking the square root of negative number in Equation (9.2); otherwise, we set $a_{acc} = \frac{2(d_i - v_i\Delta t)}{\Delta t^2}$.

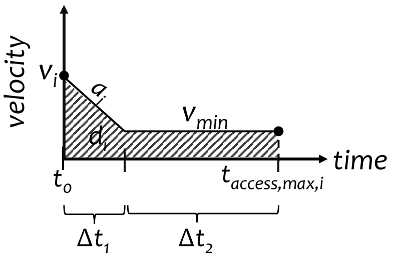
- **Deceleration**: If $t_{access,i} > t_0 + \frac{d_i}{v_i}$ then the vehicle would be early at access point if it continues traveling at the current speed. Thus, the vehicle needs to decelerate to a cruising speed ($v_{cruise,i}$), and, as shown in Figure 9.3(c), the vehicle should keep that speed till it reaches the intersection area. Knowing the desired deceleration rate a_{dec}, the cruising speed can be calculated by:

$$v_{cruise,i} = v_i + a_{dec}\Delta t + \sqrt{2a_{dec}(\frac{1}{2}a_{dec}\Delta t^2 + v_i\Delta t - d_i)} \tag{9.3}$$

where Δt is the remaining time to the assigned access time, d_i is the remaining distance to the access point, and v_i is the vehicle's velocity at current time. We set $a_{dec} = -2m/s^2$ as the desired deceleration rate of vehicles only if it satisfies the condition of $-2m/s^2 \leq \frac{2(d_i - v_i\Delta t)}{\Delta t^2}$ to avoid taking the square root of negative number in Equation (9.3); otherwise, we set $a_{dec} = \frac{2(d_i - v_i\Delta t)}{\Delta t^2}$.

The aforementioned equations do not always have solutions. In other words, it is not always possible for a connected vehicle to pass the intersection without stopping. For this reason, we also add the following two special cases to our speed-planning algorithm, in which a complete stop at intersection is inevitable. These two added cases are explained as follow:

- **Stop due to missed assigned access time**: If the access time is assigned too early such that the vehicle cannot make it, then the vehicle should be prepared to stop. If no other appropriate access time is assigned by the intersection controller while the vehicle is approaching the intersection, then the vehicle must come to a complete stop at access point because it missed its reserved access time. To be prepared



$$\Delta t_1 = \min\left\{\frac{v_{min} - v_i}{a_i}, \frac{v - v_i}{a_i}\,\middle|_{v=\sqrt{\max\{{v_i}^2 + 2a_i d_i, 0\}}}\right\}$$

$$\Delta t_2 = \max\left\{\frac{d_i}{v_{min}} - \frac{{v_{min}}^2 - {v_i}^2}{2a_i v_{min}}, 0\right\}$$

$$a_i = a_{dec}$$

$$t_{access,max,i} = t_0 + \Delta t_1 + \Delta t_2$$

Figure 9.4: Latest access time possible based on the minimum cruising speed (v_{min}) and desired deceleration rates (a_{dec}).

well ahead for this stop, we introduce $t_{access,min,i}$ in our algorithm as the earliest time that the vehicle can access the intersection if it travels at maximum acceleration and maximum speed possible. Then, the vehicle needs to be prepared for a complete stop if $t_{access,i} \leq t_{access,min,i}$. This earliest possible access time is computed locally by speed-planning engine of each vehicle in a similar way that it was implemented in the intersection controller and was previously explained in Figure 6.3.

- **Stop due to a late assigned access time**: If the assigned access time is far away in time such that even a very slow-moving vehicle reaches the intersection before that time, then the vehicle should be prepared to stop. After stopping at access point, the vehicle waits for its reserved access time before it starts to move and proceeds to the intersection. In other words, if the vehicle cannot delay its intersection access even by traveling in lowest possible velocity, then the vehicle stops behind the safety area waiting for its assigned access time. To formulate this into our speed-planning algorithm, we introduce $t_{access,max,i}$ as the latest time that the vehicle can access the intersection if it travels at minimum cruising speed (v_{min}) and maximum deceleration rate possible. Then, the vehicle needs to be prepared for a complete stop if $t_{access,i} \geq t_{access,max,i}$. The $t_{access,max,i}$ is locally computed by the speed-planning engine of each vehicle as explained in Figure 9.4. This time instance depends on the distance of the vehicle to the access point (d_i) and the max(.) and min(.) functions shown in Figure 9.4 handle this dependency problem.

The aforementioned speed-planning algorithm was then incorporated into the simulations as well as the iOS driver assistant. It should be emphasized that this algorithm is continuously applied on each vehicle; as a result, if there is an unwanted change in speed, e.g. because of a slow-moving vehicle ahead, then the algorithm updates the travel trajectory based on the new situation. Next subsection explains how the vehicles in front can affect the speed of the follower vehicles. The SIL/VIL simulation results of this testbed are given in Sections 9.4 and 9.5.

9.3 Performance Metrics (MOE)

Here, we want to analyze the impact of the MILP-based intersection control, comparing certain measures of effectiveness (MOEs) of connected vehicle that travel the same distance on the same road conditions. The MOEs studied in our SIL and VIL simulations are: (1) the intersection total number of stops, (2) the intersection total stopped delay, (3) the average stopped delay per stopped vehicle, and (4) the average travel time per vehicle. Because the OBDII data of the real vehicle can be constantly collected by our iOS application during road tests, we can also report OBD-based measures of effectiveness that are not usually reported in literature. We report the fuel consumption as a OBD-based MOE for the real vehicle in our VIL simulations. In addition, we extract the sum of engine fuel cutoff intervals using OBD logged data, and we report the percentage of fuel cutoffs during the road test. Improvement in fuel cutoff percentage means less harsh braking, prolonged deceleration without braking, and cruising at more than average speeds. The summary of the reported MOEs for SIL and VIL simulations is given in Table 9.1.

Table 9.1. MOEs reported for both SIL & VIL simulations, and OBD-based MOEs reported for the real vehicle only.

Test Configuration	MOE
SIL & VIL	Intersection total number of stops Intersection total stopped delay Average stopped delay per Stopped Vehicle Average travel time per Vehicle
VIL	Fuel consumption (liter) Fuel cutoff (%)

9.4 SIL Simulation Results

As it was explained in Section 7.5, by studying SIL simulation results, we try to determine what the maximum achievable is in a MILP controlled intersection. The simulated vehicles arrive using a stochastic generation method (negative Exponential distribution were used for 750 vehicles/hour for all four approaches); however, their arrival pattern is recorded and replayed for each testbed. In this way, the same arrival pattern was exactly replicated for each testbed. A total of 3 simulations were conducted for the testbeds (Testbeds A, B, and C (MILP)). The average and maximum speeds were set to v_{avg}=35 mph and v_{max}=45 mph, respectively. The subscription distance was set to 2 km. The measure of effectiveness (MOE) results of the SIL simulations are given in Table 9.2 for each testbed, where the performance improvements achieved by our MILP-based intersection controller are also provided comparing to Testbeds A and B. In SIL simulations for Testbed C, the mixed-integer linear programming problem was solved by IBM's CPLEX optimization package.

Table 9.2. SIL simulation results for all vehicles; and the overall performance improvements achieved by MILP-based intersection controller (Testbed C) .

MOE*	**Testbed**	**Testbed**	**Testbed**	**Gain achieved comparing to**	
(for all simulated vehicles)	**A**	**B**	**C**	**Testbed A**	**Testbed B**
Intersection traversals	2900	2900	2900	-	-
Test duration	1h 5min	1h 5min	1h 5min	-	-
Total intersection number of stops	1162	370	11	%99.1	%97.0
Total intersection stopped delay	6h 41min	4h 59min	109sec	%99.5	%99.4
Average stopped delay per Stopped Vehicle	21sec	49sec	10sec	%52.4	%79.6
Average travel time per Vehicle	2min 26sec	2min 22sec	2min 15sec	%7.5	%4.9

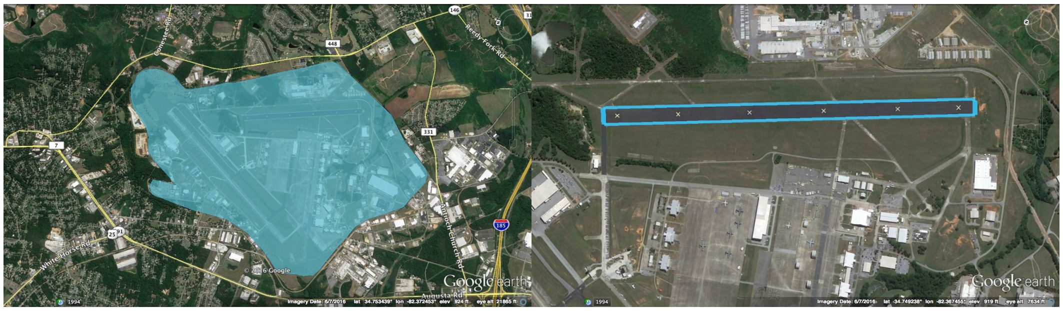
* All MOEs reported for the subscription distance (2 km).

As shown in Table 9.2, by using our MILP-based control at Testbed C, the intersection delay and number of stops were significantly reduced compared to pre-timed intersection benchmarks. Also our linear formulations did not compromise the average travel time. By using the speed advisory at Testbed B, the total number of stops at intersection (370 counts) was significantly less than that of Testbed A. However, a stopped vehicle at Testbed B, on average, stopped for a longer interval compared to the situation at Testbed A where only instantaneous traffic signal information is available within a short range from traffic signal ahead. The reason is that the vehicles in a speed advisory situation form platoons to pass through green splits; and the vehicles at the end of the platoons may get stuck at the beginning of red signals. This is why most of the stopped vehicles at Testbed B, actually stopped for almost the whole red split.

9.5 VIL Simulation Results

In this section, a MILP-based intersection control (Testbed C) is tested with two benchmarks in a vehicle-in-the-loop configuration (VIL). The detailed explanation for the VIL test environment can be found in Chapter 7. A test track at International Transportation Innovation Center (ITIC) [144] in Greenville, South Carolina was used to validate the proposed intersection control scheme in a Vehicle-In-the-Loop platform. The ITIC testing infrastructure is embedded at the South Carolina Technology & Aviation Center (SCTAC).

As shown in Figure 9.5(a), ITIC provides a 600-acre closed test site. We drove our test vehicle on a 5,500 x 300-foot controlled asphalt straightaway located at ITIC test area and shown in 9.5(b).



(a) (b)

Figure 9.5: Google satellite view of (a) International Transportation Innovation Center (ITIC) test site (b) the 5,500 x 300-foot asphalt straightaway used to execute the vehicle-in-the-loop tests.

As shown in Figure 9.6, an imaginary intersection was set up using traffic cones. A stop sign was also placed at the safety area border so that the test driver would stop at that location if a stop was asked by the virtual driver assistant. Similar to our SIL simulations, the simulated vehicles arrive using a stochastic generation method (negative Exponential distribution were used for 750 vehicles/hour for all three approaches); and their arrival pattern is recorded and replayed for each testbed. In this way, the same arrival pattern was exactly replicated for each testbed. A total of 3 vehicle-in-the-loop simulation tests were conducted for Testbed A, B, and C (MILP). The average and maximum speeds were set to v_{avg}=35 mph and v_{max}=45 mph, respectively. The subscription distance was 1.2 km. Each test consisted of 12 laps around the test track with wide U-turns at both ends of the track. After each lap, there was a different period of rest (between 10sec and 60sec) so that the obtained results would not be affected by the cyclic periodicity of the pre-timed intersection benchmarks. Each time the test driver stopped at the predefined rest location, a countdown timer showed the remaining rest period via the virtual driver assistant. The rest periods were randomly generated for the 12 laps: 60, 39, 20, 21, 45, 42, 39, 57, 14, 53, 59, and 51 seconds, respectively.

The results of the VIL simulations for each testbed are given in Table 9.3, where the performance improvements achieved by our MILP-based intersection controller are also provided comparing to Testbeds A and B. In VIL simulations for Testbed C, the mixed-integer linear programming problem was solved by IBM's CPLEX optimization package. As it was mentioned above, 750 vehicles/hour were set for three approaches of the intersection and one approach was reserved for one real test vehicle. This, and the shorter subscription distance are the reasons that there were different results obtained at test beds compared with those of Table

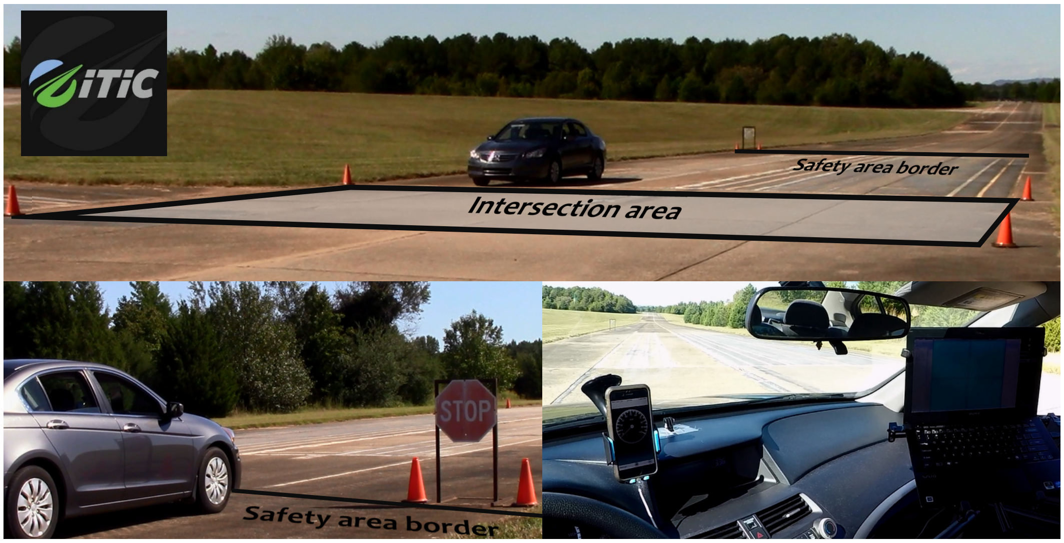


Figure 9.6: The vehicle-in-the-loop test area at International Transportation Innovation Center (ITIC).

Table 9.3. VIL simulation results for all vehicles; and the overall performance improvements achieved by MILP-based intersection controller (Testbed C) .

MOE* (for all vehicles)	**Testbed A**	**Testbed B**	**Testbed C**	**Gain achieved comparing to**	
				Testbed A	**Testbed B**
Intersection traversals	2147	2147	2147	-	-
Test duration	1h 4min	1h 4min	1h 3min	-	-
Total number of stops	900	142	0	100%	100%
Total intersection stopped delay	5h 13min	1h 56min	0sec	100%	100%
Average stopped delay per Stopped Vehicle	21sec	49sec	0sec	100%	100%
Average travel time per Vehicle	1min 36sec	1min 32sec	1min 15sec	21.8%	18.2%

* All MOEs reported for the subscription distance (1.2 km).

9.2. However, the results obtained here also showed that by using our MILP-based control, the intersection delay and number of stops were significantly reduced compared to pre-timed intersection benchmarks.

The measures of effectiveness were also extracted specifically for the real vehicle only. Using the implemented OBD data logger application, the OBD-based MOEs were also reported for the real vehicle. The results are given in Table 9.4. The test vehicle passed the MILP-based imaginary intersection 12 times without stopping that resulted in 19.5% and 18.0% benefit in fuel consumption comparing to benchmark Testbeds A and B, respectively. Benefiting from the optimal solutions of the MILP-based controller in Testbed C, the vehicle completed the 12-lap test in 51min and 11sec which was significantly shorter than the time it took in pre-timed intersection benchmarks. It should be emphasized that the vehicle also had no stopped delay in Testbed B which was due to the fact that there were no other real vehicles traveling along the vehicle's route.

Although the speed-advisory engine used in Testbed B reduced the stopped delay significantly compared to the baseline Testbed A, it did not result in significant fuel consumption reduction. The reason was that the speed-advisory imposed travel delays to the test driver by recommending very low speeds.

Table 9.4. VIL simulation results only for the real vehicle; and the overall performance improvements achieved by MILP-based intersection controller (Testbed C) .

MOE*	**Testbed**	**Testbed**	**Testbed**	**Gain achieved comparing to**	
(for real vehicle only)	**A**	**B**	**C**	**Testbed A**	**Testbed B**
Intersection traversals	12	12	12	-	-
Test duration	57min 31sec	55min 4sec	51min 11sec	-	-
Total number of stops	10	0	0	100%	0%
Total stopped delay	4min 19sec	0sec	0sec	100%	0%
Average stopped delay	26sec	0sec	0sec	100%	0%
Average travel time	1min 48sec	1min 39sec	1min 19sec	27.2%	20.8%
Fuel consumption (liter)	1.13	1.11	0.91	19.5%	18.0%
Fuel cutoff** (%)	11.6	9.1	12.2	5.2%	34.1%

* All MOEs reported for the subscription distance (1.2 km).

** This MOE is obtained by dividing the sum of fuel cutoff time intervals by the total travel time.

9.6 Acknowledgment

I would like to thank Dr. Joachim Taiber and the staff members of International Transportation Innovation Center (ITIC) for providing me with access to a closed test site in ITIC facilities in Greenville, S.C. This test site enabled me to evaluate the proposed intersection control scheme and gather data under near real world conditions.

Chapter 10

Conclusions for Part II

We proposed a novel intersection control scheme to encourage platoon formation and facilitate uninterrupted passage for autonomous vehicles at intersections. Our three key contributions are in: i) an intersection control algorithm that anticipate vehicle arrivals and guides them into fast moving platoons, ii) reducing the vehicle-intersection coordination problem to a mixed-integer linear program, and iii) developing a customized microsimulation test environment in which simulated vehicles are guided by our MILP-based controller. Microsimulation results demonstrated that our linear formulations not only minimized the intersection delay and number of stops significantly compared to pre-timed intersection benchmarks, but also ensured no crash occurred and did not compromise the average travel time.

In communication, and experimental point of view, our novel scalable mechanism allowed a large number of vehicles to subscribe to the intersection controller via cellular networks technology. This research also provided a vehicle-in-the-loop testbed with a real vehicle interacting with the intersection control cyber-layer and with the microsimulations in a virtual road network environment. In order to estimate the fuel consumption reduction of the implemented system, a new method was proposed to estimate fuel consumption using the basic engine diagnostic information.

While the proposed algorithm eliminates the need for physical traffic signals in an autonomous driving environment (100% penetration rate of equipped vehicles), the focus of a future work can be on what it will be like for a driver or a passenger in the vehicle approaching such intersections. However, it might be possible to modify the proposed algorithm to be applied to a mixed traffic consisting of autonomous-controlled and human-controlled vehicles. In a mixed traffic environment, a physical traffic light is needed, a minimum green time needs to be considered, and the minimum headway between vehicles should satisfy the human reaction time.

The proposed signal control algorithm can be improved if a scalable coordination scheme between multiple intersection controllers is incorporated into the formulations. In fact, an effective passage of platoons requires real-time coordination of neighboring intersection controllers that can be quite challenging. However, presence of intersections controllers on the same back-end node makes communication of decisions very efficient.

Appendices

Appendix A Autonomous Car Following Model

The speed-planner embedded in our virtual assistant, does not include any car following model because we have only one real vehicle-in-the-loop that is not sharing its driving lane with simulated vehicles. However, a car following model needs to be incorporated into our simulated vehicles. There has been a large amount of research in the area of car following models for vehicles driven by humans. Safety-distance models (collision avoidance models) are a class of car-following models that assume the follower vehicle always keeps a safe distance to the vehicle in front [145]. For example, the car-following model of AIMSUN software [146] is of this type, developed by P.G. Gipps [147]. In the Gipps car-following model, as one of the earliest models, the vehicle's speed is constrained to obtain a safe space headway to the vehicle in front [145]. As another example, Newell [148] presented a simplified car-following theory that is less elaborate than its predecessors [149].

To design a car following model applicable to autonomous driving, it should be noted that the computer control in autonomous vehicles will eliminate human reaction times [102]. In this dissertation, we need such a car following model to plan the simulated vehicles' longitudinal trajectories. Our model is based on the fact that if the current spacing between the follower vehicle and the lead vehicle needs to be decreased for ΔS meter then the follower should travel a distance of ΔS meter greater than that of the lead vehicle. On the other hand, if the current spacing between the two vehicles needs to be increased by ΔS meter then the follower should travel a distance of ΔS meter less than that of the lead vehicle. The time that it takes a follower vehicle to adjust its spacing depends on many factors such as its speed compared with that of the lead vehicle and also its actual spacing compared with the optimal spacing. We use the aforementioned facts to formalize our simple car following model; however, our detailed formulations are not presented here because it is not in the scope of this dissertation.

In summary, in this dissertation, the vehicles speed are constrained by (1) their desired speed in free flow, (2) the traffic signal-related speed planning, and (3) the car following-related speed planning. If the vehicle is approaching the traffic signal area then its speed is determined by traffic signal status as well as its desired speed. If there is a vehicle traveling at lower speed in front, then the vehicle speed is constrained by the car following model even if the vehicle would miss the green light or miss its assigned access time. It should be emphasized that an autonomous vehicle adopts the velocity of the leading vehicle only once it has reached the spacing predefined for that velocity.

Bibliography

[1] Intelligent Transportation Systems Joint Program Office U.S. Department of Transportation. Connected Vehicle Research in the United States. Web. `http://www.its.dot.gov/connected_vehicle/connected_vehicle_research.htm`.

[2] 2013 motor vehicle crashes: Overview. Traffic Safety Facts Research Note. Report No. DOT HS 812 101. Technical report, National Highway Traffic Safety Administration, Washington, D.C., 2013.

[3] David Schrank, Bill Eisele, and Tim Lomax. TTI 2012 urban mobility report. *Texas A&M Transportation Institute. The Texas A&M University System*, 2012.

[4] Intelligent Transportation Systems U.S. Department of Transportation. Joint Program Office. Web. `http://www.its.dot.gov/its_jpo.htm`.

[5] E. A. Mueller. Aspects of history of traffic signals. *IEEE Transactions on Vehicular Technology*, VT19(1):6–17, 1970.

[6] B. Asadi and A. Vahidi. Predictive cruise control: Utilizing upcoming traffic signal information for improving fuel economy and reducing trip time. *IEEE Transactions on Control Systems Technology*, 19(3):707–714, 2011.

[7] E. Koukoumidis, L.-S. Peh, and M. Martonosi. Signalguru: leveraging mobile phones for collaborative traffic signal schedule advisory. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 127–140. ACM, 2011.

[8] Department of Transportation. Cooperative Intersection Collision Avoidance Systems. Web. `http://www.its.dot.gov/cicas`.

[9] Jim Apple, Paul Chang, Aran Clauson, Heidi Dixon, Hiba Fakhoury, Matthew L Ginsberg, Erin Keenan, Alex Leighton, Kevin Scavezze, and Bryan Smith. Green Driver: AI in a microcosm. In *Twenty-fifth AAAI conference on Artificial Intelligence*, 2011.

[10] National Transportation Operations Coalition. National traffic signal report card. `http://www.ite.org/REPORTCARD/`.

[11] D. B. Work, O.-P. Tossavainen, S. Blandin, A. M. Bayen, T. Iwuchukwu, and K. Traction. An ensemble kalman filtering approach to highway traffic estimation using gps enabled mobile devices. In *Proceedings of 47th Conference on Decision and Control*, Cancun, Mexico, 2008.

[12] J. C. Herrera, D. B Work, R. Herring, X. Ban, Q. Jacobson, and A. M. Bayen. Evaluation of traffic data obtained via GPS-enabled mobile phones: The Mobile Century field experiment. *Transportation Research part C*, 18:568–583, 2010.

[13] A. Hofleitner, R. Herring, P. Abbeel, and A. Bayen. Learning the dynamics of arterial traffic from probe data using a dynamic bayesian network. *IEEE Transactions on Intelligent Transportation Systems*, 13:1679–1693, 2012.

[14] X. Ban, R. Herring, P. Hao, and A. Bayen. Delay pattern estimation for signalized intersections using sample travel times. *Transportation Research Record: Journal of the Transportation Research Board*, 2130:109–119, 2009.

[15] Cabspotting. http://cabspotting.org/.

[16] Nextbus. http://www.nextbus.com/.

[17] N. J. Goodall, B. L. Smith, and B. Park. Traffic signal control with connected vehicles. *Transportation Research Record: Journal of the Transportation Research Board*, 2381(1):65–72, 2013.

[18] David Kari, Guoyuan Wu, and Matthew J Barth. Development of an agent-based online adaptive signal control strategy using connected vehicle technology. In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pages 1802–1807. IEEE, 2014.

[19] L. A. Klein, M. K. Mills, and D. RP Gibson. Traffic Detector Handbook: -Volume II Report No. FHWA-HRT-06-139. Technical report, Federal Highway Administration , U.S. DOT, McLean, VA, 2006.

[20] Kurt Dresner and Peter Stone. A multiagent approach to autonomous intersection management. *Journal of artificial intelligence research*, pages 591–656, 2008.

[21] Michel Ferreira and Pedro M d'Orey. On the impact of virtual traffic lights on carbon emissions mitigation. *Intelligent Transportation Systems, IEEE Transactions on*, 13(1):284–295, 2012.

[22] Remi Tachet, Paolo Santi, Stanislav Sobolevsky, Luis Ignacio Reyes-Castro, Emilio Frazzoli, Dirk Helbing, and Carlo Ratti. Revisiting street intersections using slot-based systems. *PloS one*, 11(3):e0149607, 2016.

[23] S Ilgin Guler, Monica Menendez, and Linus Meier. Using connected vehicle technology to improve the efficiency of intersections. *Transportation Research Part C: Emerging Technologies*, 46:121–131, 2014.

[24] Thomas Bock, Markus Maurer, and Georg Farber. Validation of the vehicle in the loop (VIL); a milestone for the simulation of driver assistance systems. In *Intelligent Vehicles Symposium, 2007 IEEE*, pages 612–617. IEEE, 2007.

[25] S. A. Fayazi, A. Vahidi, G. Mahler, and A. Winckler. Traffic signal phase and timing estimation from low-frequency transit bus data. *IEEE Transactions on Intelligent Transportation Systems*, 16(1):19–28, Feb 2015.

[26] A. Vahidi, S. A. Fayazi, G. Mahler, and A. Winckler. Systems and methods for estimating traffic signal information, November 10 2015. US Patent 9,183,743.

[27] S. A. Fayazi and A. Vahidi. Crowdsourcing phase and timing of pre-timed traffic signals in the presence of queues: Algorithms and back-end system architecture. *IEEE Transactions on Intelligent Transportation Systems*, 17(3):870–881, 2016.

[28] R. Akçelik and M. Besley. Queue discharge flow and speed models for signalised intersections. In *Transportation and Traffic Theory in the 21st Century, Proceedings of the 15th International Symposium on Transportation and Traffic Theory*, Adelaide, Australia, 2002.

[29] M. J. Lighthill and G. B. Whitham. On kinematic waves. ii. a theory of traffic flow on long crowded roads. In *Proc. of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 229, pages 317–345. The Royal Society, 1955.

[30] A. Kouvelas, J. Lioris, S. A. Fayazi, and P. Varaiya. Maximum pressure controller for stabilizing queues in signalized arterial networks. *Transportation Research Record: Journal of the Transportation Research Board*, 2421(1):133–141, 2014.

[31] V. Khakhutskyy. Signal phase and timing prediction for intelligent transportation systems. M.S. thesis, Der Technischen Universitat Munschen, Munich, Germany, 2011.

[32] G. Mahler and A. Vahidi. An optimal velocity-planning scheme for vehicle energy efficiency through probabilistic prediction of traffic-signal timing. *IEEE Transactions on Intelligent Transportation Systems*, 15(6):2516–2523, Dec 2014.

[33] P. Hao, X. Ban, K. P. Bennett, Q. Ji, and Z. Sun. Signal timing estimation using sample intersection travel times. *IEEE Transactions on Intelligent Transportation Systems*, 13(2):792–804, 2012. ID: 1.

[34] M. Kerper, C. Wewetzer, A. Sasse, and M. Mauve. Learning traffic light phase schedules from velocity profiles in the cloud. In *Proceedings of the 5th International Conference on New Technologies, Mobility and Security (NTMS)*, Istanbul, Turkey, 2012.

[35] Y. Cheng, X. Qin, J. Jin, and B. Ran. An exploratory shockwave approach for signalized intersection performance measurements using probe trajectories. In *Proceedings of the Transportation Research Board 89th annual meeting*, Washington, D.C., 2010.

[36] Y. Chuang, C. Yi, Y. Tseng, C. Nian, and C. Ching. Discovering phase timing information of traffic light systems by stop-go shockwaves. *IEEE Transactions on Mobile Comp.*, 14(1):58–71, 2015.

[37] NGSIM: The Next Generation SIMulation Community by The Federal Highway Administration (FHWA) U.S. Department of Transportation. http://ops.fhwa.dot.gov/trafficanalysistools/ngsim.htm.

[38] Y. Zhu, X. Liu, M. Li, and Q. Zhang. Pova: Traffic light sensing with probe vehicles. *IEEE Transactions on Parallel and Distributed Systems*, 24(7):1390–1400, 2013.

[39] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau. Extensible markup language (xml) 1.0. *W3C Recommendation*, 2008 (`http://www.w3.org/TR/REC-xml/`).

[40] Roger W. Sinnott. Virtues of the haversine. *Sky and telescope*, 68:158, 1984.

[41] J. L. Gattis, S. H. Nelson, and J.D. Tubbs. School bus acceleration characteristics. Technical Report FHWA/AR-009, Mack-Blackwell Transportation Center, University of Arkansas, 1998.

[42] S. Yoon, H. Li, J. Jun, J. Ogle, R. Guensler, and M. Rodgers. A methodology for developing transit bus speed-acceleration matrices to be used in load-based mobile source emission models. In *Proceedings of Transportation Research Board annual meeting*, 2005.

[43] G. Mahler and A. Vahidi. Reducing idling at red lights based on probabilistic prediction of traffic signal timings. In *Proceedings of the American Control Conference*, pages 6557–6562, Montreal, Quebec, 2012.

[44] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.

[45] Official United States Time by National Institutue of Standards and Technology. http://nist.time.gov/.

[46] A. Skabardonis and N. Geroliminis. Real-time estimation of travel times on signalized arterials. In *Proc. of the 16th Int'l Symposium on Transportation and Traffic Theory*, College Park, Maryland, 2005.

[47] H. Liu, X. Wu, W. Ma, and H. Hu. Real-time queue length estimation for congested signalized intersections. *Transportation research part C: emerging technologies*, 17(4):412–427, 2009.

[48] Gregory Stephanopoulos, P. G. Michalopoulos, and George Stephanopoulos. Modelling and analysis of traffic queue dynamics at signalized intersections. *Transportation Research Part A: General*, 13(5):295–307, 1979.

[49] Y. Cheng, X. Qin, J. Jin, B. Ran, and J. Anderson. Cycle-by-cycle queue length estimation for signalized intersections using sampled trajectory data. *Transportation Research Record: Journal of the Transportation Research Board*, 2257(-1):87–94, 2011.

[50] N. Rouphail, A. Tarko, and J. Li. Traffic flow at signalized intersections: Revised monograph on traffic flow theory. Technical report, Federal Highway Administration, U.S. DOT, Washington, D.C., 2005.

[51] Highway Capacity Manual. Transportation research board. *National Research Council, Washington, DC*, 2000.

[52] R. P. Roess, E. S. Prassas, and W. R. McShane. *Traffic engineering*. Prentice Hall, 2011.

[53] M. S. Chaudhry and P. Ranjitkar. Delay estimation at signalized intersections with variable queue discharge rate. *Journal of the Eastern Asia Society for Transportation Studies*, 10(0):1764–1775, 2013.

[54] P. Hao, X. Ban, and J. W. Yu. Kinematic equation-based vehicle queue location estimation method for signalized intersections using mobile sensor data. *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, 2014.

[55] M. Li, K. Boriboonsomsin, G. Wu, W. Zhang, and M. Barth. Traffic energy and emission reductions at signalized intersections: a study of the benefits of advanced driver information. *Int'l Journal of Intelligent Transportation Systems Research*, 7(1):49–58, 2009.

[56] K. P. Sanketh, S. Subbarao, and K. A. Jolapara. I2v and v2v communication based vanet to optimize fuel consumption at traffic signals. In *Proc. of the 13th Int'l IEEE Conf. on Intelligent Transportation Systems (ITSC)*, pages 1251–1255, Madeira Island, Portugal, 2010.

[57] M. Maile and L. Delgrossi. Cooperative Intersection Collision Avoidance System for Violations (CICAS-V) for avoidance of violation-based intersection crashes. In *Proc. of the 21st Int'l Conf. on the Enhanced Safety of Vehicles (ESV)*, Stuttgart, Germany, 2009.

[58] Audi Travolution Project. https://www.audi-mediaservices.com.

[59] S. A. Fayazi, S. Farhangi, and B. Asaei. Fuel consumption and emission reduction of a mild hybrid vehicle. In *Proc. of the 34th Conf. of IEEE Industrial Electronics (IECON)*, pages 216–221, Orlando, Florida, 2008.

[60] S. A. Fayazi, S. Farhangi, and B. Asaei. Power delivery co-ordination to meet driver's demand in a mild hybrid vehicle with automated manual transmission. In *Industrial Electronics, 2008. IECON 2008. 34th Annual Conference of IEEE*, pages 327–332, Nov 2008.

[61] Mobileye. Mobileye development and evaluation platforms. `http://www.mobileye.com/technology/development-evaluation-platforms/`.

[62] Google Protocol Buffer. `https://developers.google.com/protocol-buffers/docs/overview`.

[63] G. Kaur and M. M. Fuad. An evaluation of protocol buffer. In *Proceedings of the IEEE SoutheastCon 2010*, pages 459–462, Charlotte-Concord, NC, USA, 2010.

[64] J. Postel. User datagram protocol. *STD 6, RFC 768*, 1980 (`http://www.ietf.org/rfc/rfc0768`).

[65] J. Postel. Transmission control protocol. *STD 7, RFC 793*, 1981 (`http://www.ietf.org/rfc/rfc0793`).

[66] D. Crockford. The application/json media type for javascript object notation (json). *RFC 4627*, 2006 (`http://www.ietf.org/rfc/rfc4627`).

[67] JDBC Driver for MySQL. http://dev.mysql.com/downloads/connector/j/.

[68] P.B. Hunt, D.I. Robertson, R.D. Bretherton, and M Cr Royle. The scoot on-line traffic signal optimisation technique. *Traffic Engineering & Control*, 23(4), 1982.

[69] PR Lowrie. The sydney coordinated adaptive traffic system-principles, methodology, algorithms. In *International Conference on Road Traffic Signalling, 1982, London, United Kingdom*, number 207, 1982.

[70] Christina Diakaki, Markos Papageorgiou, and Kostas Aboudolas. A multivariable regulator approach to traffic-responsive network-wide signal control. *Control Engineering Practice*, 10(2):183–195, 2002.

[71] Gurulingesh Raravi, Vipul Shingde, Krithi Ramamritham, and Jatin Bharadia. Merge algorithms for intelligent vehicles. In *Next Generation Design and Verification Methodologies for Distributed Embedded Control Systems*, pages 51–65. Springer, 2007.

[72] Thomas Coleman, Mary Ann Branch, and Andrew Grace. Optimization toolbox. *For Use with MATLAB. Users Guide for MATLAB*, 5, 1999.

[73] Joyoung Lee and Byungkyu Park. Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment. *Intelligent Transportation Systems, IEEE Transactions on*, 13(1):81–90, 2012.

[74] Feng Zhu and Satish V Ukkusuri. A linear programming formulation for autonomous intersection control within a dynamic traffic assignment and connected vehicle environment. *Transportation Research Part C: Emerging Technologies*, 55:363–378, 2015.

[75] Alessandro Colombo and Domitilla Del Vecchio. Efficient algorithms for collision avoidance at intersections. In *Proceedings of the 15th ACM international conference on Hybrid Systems: Computation and Control*, pages 145–154. ACM, 2012.

[76] Xiao-Feng Xie, Stephen F Smith, Liang Lu, and Gregory J Barlow. Schedule-driven intersection control. *Transportation Research Part C: Emerging Technologies*, 24:168–189, 2012.

[77] Michael L Pinedo. *Scheduling: theory, algorithms, and systems*. Springer Science & Business Media, 2012.

[78] Allen Hawkes. Traffic control with connected vehicle routes in surtrac. 2016.

[79] S. A. Fayazi, N. Wan, S. Lucich, A. Vahidi, and G. Mocko. Optimal pacing in a cycling time-trial considering cyclist's fatigue dynamics. In *2013 American Control Conference*, pages 6442–6447. IEEE, 2013.

[80] N. Wan, S. A. Fayazi, H. Saeidi, and A. Vahidi. Optimal power management of an electric bicycle based on terrain preview and considering human fatigue dynamics. In *2014 American Control Conference*, pages 3462–3467. IEEE, 2014.

[81] Suvrajeet Sen and K Larry Head. Controlled optimization of phases at an intersection. *Transportation science*, 31(1):5–17, 1997.

[82] Heejin Ahn and Domitilla Del Vecchio. Semi-autonomous intersection collision avoidance through job-shop scheduling. *arXiv preprint arXiv:1510.07026*, 2015.

[83] Kurt Dresner and Peter Stone. Multiagent traffic management: An improved intersection control mechanism. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 471–477. ACM, 2005.

[84] Kurt Dresner. Autonomous intersection management. PhD thesis, University of Texas at Austin, 2009.

[85] Dave McKenney and Tony White. Distributed and adaptive traffic signal control within a realistic traffic simulation. *Engineering Applications of Artificial Intelligence*, 26(1):574–583, 2013.

[86] L. Chen and C. Englund. Cooperative intersection management: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 17(2):570–586, Feb 2016.

[87] S. A. Fayazi, A. Vahidi, and A. Luckow. Optimal scheduling of autonomous vehicle arrivals at intelligent intersections via MILP. In *2017 American Control Conference*. Under Review, 2017.

[88] Antonio Alonso-Ayuso, Laureano F Escudero, and F Javier Martín-Campo. Collision avoidance in air traffic management: a mixed-integer linear optimization approach. *Intelligent Transportation Systems, IEEE Transactions on*, 12(1):47–57, 2011.

[89] Francesco Borrelli, Dharmashankar Subramanian, Arvind U Raghunathan, and Lorenz T Biegler. MILP and NLP techniques for centralized trajectory planning of multiple unmanned air vehicles. In *American Control Conference, 2006*, pages 6–pp. IEEE, 2006.

[90] Arthur Richards, Tom Schouwenaars, Jonathan P How, and Eric Feron. Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming. *Journal of Guidance, Control, and Dynamics*, 25(4):755–764, 2002.

[91] Kieran Forbes Culligan. *Online trajectory planning for uavs using mixed integer linear programming*. PhD thesis, Massachusetts Institute of Technology, 2006.

[92] Esten Ingar Grøtli and Tor Arne Johansen. Path planning for UAVs under communication constraints using SPLAT! and MILP. *Journal of Intelligent & Robotic Systems*, 65(1-4):265–282, 2012.

[93] Qing He, K Larry Head, and Jun Ding. Pamscod: Platoon-based arterial multi-modal signal control with online data. *Transportation Research Part C: Emerging Technologies*, 20(1):164–184, 2012.

[94] Nathan H Gartner, John DC Little, and Henry Gabbay. Optimization of traffic signal settings by mixed-integer linear programming: Part i: The network coordination problem. *Transportation Science*, 9(4):321–343, 1975.

[95] John DC Little. The synchronization of traffic signals by mixed-integer linear programming. *Operations Research*, 14(4):568–594, 1966.

[96] G. De Nunzio, G. Gomes, C. Canudas de Wit, R. Horowitz, and P. Moulin. Speed advisory and signal offsets control for arterial bandwidth maximization and energy consumption reduction. *IEEE Transactions on Control Systems Technology*, PP(99):1–13, 2016.

[97] Md Abdus Samad Kamal, Jun-ichi Imura, Tomohisa Hayakawa, Akira Ohata, and Kazuyuki Aihara. Traffic signal control of a road network using MILP in the MPC framework. *International journal of intelligent transportation systems research*, 13(2):107–118, 2015.

[98] Md Abdus Samad Kamal, Jun-ichi Imura, and Tomohisa Hayakawa. Network-wide optimization of traffic signals using mixed integer programming. *Journal of robotics and mechatronics*, 26(5):607–615, 2014.

[99] Shu Lin, Bart De Schutter, Yugeng Xi, and Hans Hellendoorn. Fast model predictive control for urban road networks via MILP. *IEEE Transactions on Intelligent Transportation Systems*, 12(3):846–856, 2011.

[100] Jennie Lioris, Ramtin Pedarsani, Fatma Yildiz Tascikaraoglu, and Pravin Varaiya. Doubling throughput in urban roads by platooning. *accepted in the IFAC Symposium on Control in Transportation Systems*, 2015.

[101] Il Yong Kim and OL De Weck. Adaptive weighted sum method for multiobjective optimization: a new method for pareto front generation. *Structural and multidisciplinary optimization*, 31(2):105–116, 2006.

[102] CC Chien and P Ioannou. Automatic vehicle-following. In *American Control Conference, 1992*, pages 1748–1752. IEEE, 1992.

[103] Michel Berkelaar, Kjell Eikland, and Peter Notebaert. Reference guide to open source (mixed-integer) linear programming system (version 5.1.0.0 dated 1 may 2004). available from: http://lpsolve.sourceforge.net/5.1/.

[104] Li Sheng. *The Interactive Hardware-in-loop Simulation System for Traffic Control System Development*. PhD thesis, University of Akron, 2005.

[105] Aleksandar Stevanovic, Ahmed Abdel-Rahim, Milan Zlatkovic, and Enas Amin. Microscopic modeling of traffic signal operations: Comparative evaluation of hardware-in-the-loop and software-in-the-loop simulations. *Transportation Research Record: Journal of the Transportation Research Board*, (2128):143–151, 2009.

[106] Christopher Day, Joseph Ernst, Thomas Brennan Jr, Chih-Sheng Chou, Alexander Hainen, Stephen Remias, Andrew Nichols, Brian Griggs, and Darcy Bullock. Performance measures for adaptive signal control: Case study of system-in-the-loop simulation. *Transportation Research Record: Journal of the Transportation Research Board*, (2311):1–15, 2012.

[107] Dipl-Ing Sebastian Schwab, Dipl-Ing Tobias Leichsenring, M Sc Marc René Zofka, and Dipl-Inform Tobias Bär. Consistent test method for assistance systems. *ATZ worldwide*, 116(9):38–43, 2014.

[108] Björn Blissing, Fredrik Bruzelius, and Johan Ölvander. Augmented and mixed reality as a tool for evaluation of vehicle active safety systems. In *RSS2013 Road Safety and Simulation-International Conference. October 23-25, 2013. Rome, Italy.*, 2013.

[109] Olaf Jeroen Gietelink. *Design and validation of advanced driver assistance systems*. TU Delft, Delft University of Technology, 2007.

[110] Olaf Gietelink, Jeroen Ploeg, Bart De Schutter, and Michel Verhaegen. Development of advanced driver assistance systems with vehicle hardware-in-the-loop simulations. *Vehicle System Dynamics*, 44(7):569–590, 2006.

[111] Yvonne Laschinsky, Von Neumann-Cosel, Mark Gonter, Christian Wegwerth, Rolf Dubitzky, A Knoll, et al. Evaluation of an active safety light using virtual test drive within vehicle in the loop. In *Industrial Technology (ICIT), 2010 IEEE International Conference on*, pages 1119–1112. IEEE, 2010.

[112] M Quinlan, A Tsz-Chiu, J Zhu, N Stiurca, and P Stone. Bringing simulation to life: A mixed reality autonomous intersection. iros. In *IEEE/RSJ International Conference on*, 2010.

[113] Pengfei Li and Pitu B Mirchandani. A new hardware-in-the-loop traffic signal simulation framework to bridge traffic signal research and practice. 2016.

[114] H. Xia, K. Boriboonsomsin, F. Schweizer, A. Winckler, K. Zhou, W.B. Zhang, and M. Barth. Field operational testing of eco-approach technology at a fixed-time signalized intersection. In *Proceedings of the 2012 15th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 188–193. IEEE, 2012.

[115] A. Weber and A. Winckler. Advanced traffic signal control algorithms, appendix A: Exploratory advanced research project: BMW final report. Technical Report CA13-2157-B, Caltrans Division of Research, Innovation and System Information, Sep. 2013.

[116] Michel Ferreira, Ricardo Fernandes, Hugo Conceição, Wantanee Viriyasitavat, and Ozan K Tonguz. Self-organized traffic control. In *Proceedings of the seventh ACM international workshop on VehiculAr InterNETworking*, pages 85–90. ACM, 2010.

[117] Manuel Nakamurakare, Wantanee Viriyasitavat, and Ozan K Tonguz. A prototype of virtual traffic lights on android-based smartphones. 2013.

[118] T Lozano-Perez, Ingemar J Cox, and Gordon T Wilfong. *Autonomous robot vehicles*. Springer Science & Business Media, 2012.

[119] Muhammad Azmat. *Impact of autonomous vehicles on urban mobility*. PhD thesis, Institut für Transportwirtschaft und Logistik, WU Wien, 2015.

[120] James Barker, Sam Mendez, Evan Brown, Tim Billick, and Justin Glick. Technical and Legal Challenges : An Overview of the State of the Art in Autonomous Vehicle Technology and Policy. Technical report, Autonomous Vehicles Team , University of Washington School of Law, Washington: Technology Law and Policy Clinic, 2013.

[121] Apple Inc. iOS developer library; location and maps programming guide. `https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/LocationAwarenessPG/Introduction/Introduction.html`, accessed May 2016. [Online].

[122] Apple Inc. iOS developer library; the mapkit framework reference. `https://developer.apple.com/library/ios/documentation/MapKit/Reference/MapKit_Framework_Reference/`, accessed May 2016. [Online].

[123] SAE Standard. E/E diagnostic test modes. *SAE J1979*, 2006.

[124] T.H. DeFries, M.A. Sabisch, and S. Kishan. Light-duty vehicle in-use fuel economy data collection: Pilot study. Technical report, International Council on Clean Transportation., Washington, DC, 2013. `http://www.theicct.org/sites/default/files/ICCT-131108%20-%20ERG%20-%20In-Use%20FE%20Pilot-%20V8FInal.pdf`.

[125] ISO standard. ISO 15031-5. In *Communication between vehicle and external equipment for emissions-related diagnostics - Part 5: Emissions-related diagnostic services*, 2006.

[126] Ilya Kolmanovsky, Kevin McDonough, and Oleg Gusikhin. Estimation of fuel flow for telematics-enabled adaptive fuel and time efficient vehicle routing. In *ITS Telecommunications (ITST), 2011 11th International Conference on*, pages 139–144. IEEE, 2011.

[127] Vitor Ribeiro, Jose Rodrigues, and Ana Aguiar. Mining geographic data for fuel consumption estimation. In *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*, pages 124–129. IEEE, 2013.

[128] Adriano Alessandrini, Francesco Filippi, and Fernando Ortenzi. Consumption calculation of vehicles using obd data. In *20th International Emission Inventory Conference- Emission Inventories-Meeting the Challenges Posed by Emerging Global, National, and Regional and Local Air Quality Issues*, 2012.

[129] ICCT fuel economy data collection pilot study. Technical report, International Council on Clean Transportation, 2013. http://www.theicct.org/sites/default/files/TNM_ICCT_FE_Data_Collection_Pilot_Study_ProjectReport_Final2.pdf.

[130] Gary Thomas Pepper. Methods and system for determining consumption and fuel efficiency in vehicles, August 10 2010. US Patent 7,774,130.

[131] Stephen G Russ, Edward W Kaiser, Waiter O Siegl, Diane H Podsiadlik, and Kathy M Barrett. The effect of air/fuel ratio on wide open throttle HC emissions from a spark-ignition engine. Technical report, SAE Technical Paper, 1994.

[132] MF Abdul Rahim, MM Rahman, and RA Bakar. Cycle engine modelling of spark ignition engine processes during wide-open throttle (WOT) engine operation running by gasoline fuel. In *IOP Conference Series: Materials Science and Engineering*, volume 36, page 012041. IOP Publishing, 2012.

[133] ELM327, OBD to RS232 interpreter. *ELM Electronics*, 2012. Available online: www.elmelectronics.com/DSheets/ELM327DS.pdf (accessed on May 2016).

[134] ISO standard. ISO 15765-4 CAN. In *Road vehicles – Diagnostic communication over Controller Area Network (DoCAN) - Part 4: Requirements for emissions-related systems*, 2011.

[135] SAE Standard. Class B data communications network interface. *SAE J1850*, 2006.

[136] Vitor Daniel Ferreira da Cunha Ribeiro. Mining geographic data for fuel consumption estimation. M.S. thesis, University of Porto, Porto, Portugal, 2013.

[137] Jesse Levinson, Jake Askeland, Jennifer Dolson, and Sebastian Thrun. Traffic light mapping, localization, and state detection for autonomous vehicles. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5784–5791. IEEE, 2011.

[138] Nathaniel Fairfield and Chris Urmson. Traffic light mapping and detection. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5421–5426. IEEE, 2011.

[139] Behrang Asadi and Ardalan Vahidi. *Predictive use of traffic signal state for fuel saving*, pages 484–489. 12 2009.

[140] Sindhura Mandava, Kanok Boriboonsomsin, and Matthew Barth. Arterial velocity planning based on traffic signal information under light traffic conditions. In *Intelligent Transportation Systems, 2009. ITSC'09. 12th International IEEE Conference on*, pages 1–6. IEEE, 2009.

[141] R. K. Kamalanathsharma and H. Rakha. Agent-based simulation of ecospeed-controlled vehicles at signalized intersections. *Transportation Research Record: Journal of the Transportation Research Board*, (2427):1–12, 2014.

[142] Hesham Rakha and Raj Kishore Kamalanathsharma. Eco-driving at signalized intersections using V2I communication. In *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pages 341–346. IEEE, 2011.

[143] A. Lawitzky, D. Wolherr, and M. Buss. Energy optimal control to approach traffic lights. In *Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Tokyo Big Sight, Japan, November 2013.

[144] International Transportation Innovation Center (ITIC). Web. http://www.itic-sc.com.

[145] Johan Janson Olstam and Andreas Tapani. Comparison of car-following models. Technical report, Swedish National Road and Transport Research Institute, 2004.

[146] AIMSUN Microscopic Traffic Simulator. TSSTransport Simulation Systems. `http://www.aimsun.com`.

[147] Peter G Gipps. A behavioural car-following model for computer simulation. *Transportation Research Part B: Methodological*, 15(2):105–111, 1981.

[148] Gordon Frank Newell. A simplified car-following theory: a lower order model. *Transportation Research Part B: Methodological*, 36(3):195–205, 2002.

[149] Soyoung Ahn, Michael J Cassidy, and Jorge Laval. Verification of a simplified car-following theory. *Transportation Research Part B: Methodological*, 38(5):431–440, 2004.