# Cooperative Perception and Use of Connectivity in Automated Driving

Dissertation

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

Mustafa Ridvan Cantas, M.S.

Graduate Program in Electrical and Computer Engineering

The Ohio State University

2022

Dissertation Committee

Prof. Levent Guvenc, Advisor

Prof. Benjamin Coifman

Prof. Keith Redmill

Prof. Bilin Aksun Guvenc

Abstract

Recent developments in connected and autonomous vehicles (CAV) improve traffic safety and fuel efficiency and take away the driving burden partially or completely from the driver. CAVs are improving the traffic safety using their on-board sensors such as camera, lidar, radar and ultrasonic sensors. While these sensors are effective in sensing the objects in their field of view, CAVs can also sense other road users by utilizing communication modems, and learn more about the traffic patterns such as the signal phase and timing (SPaT) information of a traffic light at an intersection. One recent approach to boost capabilities of CAVs is the sharing of perceived target detections with other road users. This practice significantly increases the situational awareness of connected road users. Since the cooperative perception concept is still in early stages, the development of use case scenarios and capabilities of this concept are still active research areas. Therefore, a Cooperative Perception (CP) architecture and CAV functionalities are developed in this research to improve the traffic safety, fuel economy, and ride comfort. Their effectiveness is demonstrated with use case scenarios. The developed CP architecture relies on Joint Probability Data Association (JPDA) multi object tracking algorithm to track detected objects and create CP messages. Then, with the simulations, it is shown that situational awareness of the road users increased significantly, thereby improving their traffic safety. Later, another use case scenario for CP is developed to improve Green Light Optimized

Speed Advisory (GLOSA). In this use case, the vehicle not only relies on SPaT and MAP messages but also relies on the shared CP messages by a smart intersection. As a result, two different algorithms are developed to utilize infrastructure CP messages. While the first approach to generate speed advisory was to create a rule-based solution, the second approach utilizes a Deep Deterministic Policy Gradient (DDPG) reinforcement learning agent to control the vehicle. The developed approaches showed promising fuel efficiency and ride comfort advantages. Finally, as part of the CAV functionalities, lateral and longitudinal controllers are designed to aid the driver whenever needed. While the designed lateral controller has lane centering and path following functionalities, the designed cooperative adaptive cruise control reduces time gap between the ego vehicle and the target vehicle to being followed utilize roads more efficiently and improve fuel economy for platoons. The designed CAV subsystems can be used as standalone functionalities to improve safety, efficiency and comfort of the passengers or they can be used as an enabling part of a highly automated vehicles.

Dedicated to my family.

Acknowledgments

Throughout the writing of this dissertation, I have received a great deal of support and assistance.

I would first like to thank my advisor, Prof. Levent Güvenç, whose expertise was invaluable in formulating the research problems and methodology. You provided me with the tools that I needed to choose the right direction and successfully complete my dissertation.

I would like to thank my dissertation committee members Prof. Benjamin Coifman,

Prof. Keith Redmill, and Prof. Bilin Aksun-Güvenç for their valuable feedback on my research. I would like to extend special thanks to Prof. Bilin Aksun-Güvenç for her valuable guidance throughout my studies in the Automated Driving Lab.

I would like to thank my supervisor Prof. Greg Chapman for his valuable mentorship throughout my teaching experience.

I would like to acknowledge my colleagues Santhosh, Sukru Yaren, Ozgenur, Xinchen, Sheng, Karina, Pedro, Nitish, Hongliang, and Evan in the Automated Driving Lab for their wonderful collaboration. I want to thank you all for the opportunities I was given to further my research.

I would also like to thank my supervisor Gopi Chandra Surnilla and colleagues Arpita Chand, Dr. Hao Zhang, Dr. Martin Summer, and Dr. Archak Mittal from my research

internship at Ford Motor Company. Your collaboration has sharpened my thinking and brought my work to a higher level.

The research presented in Chapter 3 and Chapter 5 of this dissertation is conducted as part of academic training at Ford Motor Company in 2019 and 2021 respectively.

Finally, I would like to thank my parents Ceylan and Sultan Cantas, loving wife Nermin Cantas, and dearest son Mert Cantas for their support and sympathy throughout my studies. I could not have completed this dissertation without your support.

## Vita

2009.....B.S. Electrical and Electronics Engineering

Yeditepe University, Istanbul, Turkey

2012 .....M.S. Electrical and Electronics Engineering

Bilkent University, Ankara, Turkey

2009 – 2012.....Graduate Research & Teaching Associate

Electrical and Electronics Eng. Department, Bilkent University, Ankara, Turkey

2013 – 2016 .....Embedded Hardware Design Engineer,

Otokar Defense and Automotive Industry Inc., Sakarya, Turkey

2019 & 2021 Summers ..... Research Intern

Ford Motor Company, Research and Innovation Center, Dearborn, MI

2016 – 2022.....Graduate Research & Teaching Associate

Electrical and Computer Eng. Department, The Ohio State University, Columbus, OH

## Publications

K. M. Cime, M. R. Cantas, P. Fernandez, B. A. Guvenc, L. Guvenc, A. Joshi, J. Fishelson, and A. Mittal, "Assessing the Access to Jobs by Shared Autonomous Vehicles in Marysville, Ohio: Modeling, Simulating and Validating", in SAE WCX Conference, 2021.

S. Y. Gelbal, M. R. Cantas, B. A. Guvenc, and L. Guvenc, "Virtual and Real Data Populated Intersection Visualization and Testing Tool for V2X Application development", in SAE WCX Conference, 2021.

O. Kavas-Torris, S. Y. Gelbal, M. R. Cantas, B. A. Guvenc, and L. Guvenc, “Connected UAV and CAV Coordination for Improved Road Network Safety and Mobility”, in SAE WCX Conference, 2021.

M. R. Cantas, A. Chand, H. Zhang, G. C. Surnilla, and L. Güvenç, “Data association between perception and V2V communication sensors”, CoRR, vol. abs/2101.08228, 2021. arXiv: 2101.08228.

M. R. Cantas and L. Guvenc, “A Customized Co-Simulation Environment for Autonomous Vehicle Algorithm Development and Evaluation”, in SAE WCX Conference, 2021.

K. M. Cime, M. R. Cantas, G. Dowd, L. Guvenc, B. A. Guvenc, A. Mittal, A. Joshi, and J. Fishelson, “Hardware-in-the-Loop, Traffic-in-the-Loop and Software-in-the-Loop Autonomous Vehicle Simulation for Mobility Studies”, in SAE WCX Conference, 2020.

S. Y. Gelbal, M. R. Cantas, B. A. Guvenc, L. Guvenc, G. Surnilla, H. Zhang, M. Shulman, A. Katriniok, and J. Parikh, “Hardware-in-the-Loop and Road Testing of RLVW and GLOSA Connected Vehicle Applications”, in SAE WCX Conference, 2020.

O. Kavas-Torris, M. R. Cantas, S. Y. Gelbal, B. A. Guvenc, and L. Guvenc, “Fuel Economy Benefit Analysis of Pass-at-Green (PaG) V2I Application on Urban Routes with STOP Signs”, International Journal of Vehicle Design, vol. 83 No.2/3/4, pp.258 - 279, 2020.

O. Kavas-Torris, M. R. Cantas, S. Y. Gelbal, and L. Guvenc, “Performance Evaluation of the Pass-at-Green (PaG) Connected Vehicle V2I Application”, in SAE WCX Conference, 2020.

O. Kavas-Torris, M. R. Cantas, K. Meneses Cime, B. Aksun Guvenc, and L. Guvenc, “The Effects of Varying Penetration Rates of L4-L5 Autonomous Vehicles on Fuel Efficiency and Mobility of Traffic Networks”, in SAE WCX Conference, 2020.

J. H. Lee, J. Lee, K. Jeong, S. Yoo, B. Lee, S. Kim, L. Guvenc, and M. R. Cantas, “Development of a New Neutral Coasting Control Utilizing ADAS and GPS”, SAE International Journal of Connected and Automated Vehicles, vol. 2, no. 12-02-02-0004, pp. 69–77, 2019.

X. Li, S. Zhu, S. Y. Gelbal, M. R. Cantas, B. A. Guvenc, and L. Guvenc, “A unified, scalable and replicable approach to development, implementation and HIL evaluation of autonomous shuttles for use in a smart city”, in SAE WCX Conference, 2019.

M. R. Cantas, S. Fan, O. Kavas, S. Tamilarasan, L. Guvenc, S. Yoo, J. H. Lee, B. Lee, and J. Ha, “Development of Virtual Fuel Economy Trend Evaluation Process”, in SAE WCX Conference, 2019.

M. R. Cantas, S. Y. Gelbal, L. Guvenc, and B. A. Guvenc, “Cooperative Adaptive Cruise Control Design and Implementation”, in SAE WCX Conference, 2019.

M. R. Cantas, O. Kavas, S. Tamilarasan, S. Y. Gelbal, and L. Guvenc, “Use of hardware in the loop (HIL) simulation for developing connected autonomous vehicle (CAV) applications”, in SAE WCX Conference, 2019.

H. Lee, J. Lee, S. Yoo, K. Jeong, B. Lee, S. Kim, L. Guvenc, M. R. Cantas, S. Tamilarasan, and N. Chandramouli, “Utilization of ADAS for Improving Performance of Coasting in Neutral”, in SAE WCX Conference, 2018.

M. R. Cantas and L. Guvenc, “Camera Based Automated Lane Keeping Application Complemented by GPS Localization Based Path Following”, in SAE WCX Conference, 2018.

S. Zhu, S. Y. Gelbal, X. Li,M. R. Cantas, B. Aksun-Guvenc, and L. Guvenc, “Parameter space and model regulation based robust, scalable and replicable lateral control design for autonomous vehicles”, in2018 IEEE Conference on Decision and Control (CDC), IEEE, 2018, pp. 6963–6969.

S. Y. Gelbal, S. Tamilarasan, M. R. Cantas, L. Guvenc, and B. Aksun-Guvenc, “A connected and autonomous vehicle hardware-in-the-loop simulator for developing automated driving algorithms”, in 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE, 2017,pp. 3397–3402.

## Fields of Study

Major Field: Electrical and Computer Engineering

Table of Contents

List of Tables

List of Figures

# Chapter 1. Introduction

## 1.1 Background and Motivation

Automotive companies, and their suppliers are focusing on the development of highly automated vehicle technologies, considering the potential of autonomous vehicles in improving traffic safety, energy efficiency, and mobility. Aligning with these efforts, SAE published the levels of driving automation from level 0 (no automation) to level 5 (full vehicle autonomy) [1]. This standard also defines the responsibilities of drivers and the automated driving features at each autonomy level. As the aim of the autonomous vehicles is to improve the safety of its occupants, full control of the autonomous vehicle for levels 4 and 5 requires higher situational awareness of the system in order not to compromise the safety of occupants. Motivated by the need to improve situational awareness of vehicles, this research focuses on developing connected and automated driving solutions.

To start, the perception of the surrounding vehicles and environment information using only range sensors is challenging for automated vehicles. It is not possible to detect vehicles with rage sensors when the objects are outside of the Field of View (FOV). This research aims to maximize the situational awareness of all the vehicles in a mixed traffic environment with the use of combined range sensors and communication technologies. As an example scenario, the mixed traffic shown in Figure 1 consists of vehicles without any

detection or communication capability (no sensing), connected vehicles, and connected vehicles with range sensors, such as connected and autonomous vehicles (Figure 1).

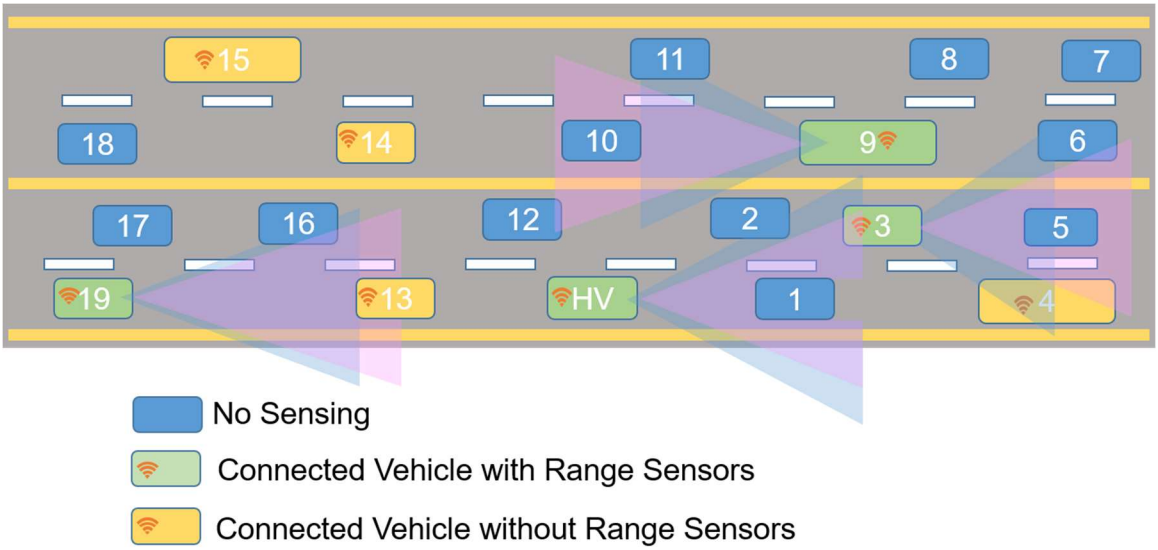


Figure 1 Mixed traffic scenario visualization for the defined problem

In the presented scenario, if the vehicles share what they perceive via their communication sensors, in other words with Cooperative Perception (CP), the situational awareness of the traffic participants increases significantly. This is shown in Figure 2, where vehicles 4, 13, 14 and 15 are connected vehicles without range sensors and broadcast only their own BSM values. The host vehicle (HV) and vehicles 3, 9, 19 are connected vehicles that are also equipped with range sensors. They share not only their own BSMs but also generate and share BSM data for the detected vehicles which do not have a V2X unit. Therefore, thanks to cooperative perception capability of some neighboring vehicles, the host vehicle (HV) is now aware of all the vehicles on the road except vehicles 7, 8, 17 and 18. As can be seen

from this simple illustration, the situational awareness of the HV can be improved significantly by utilizing cooperative perception.

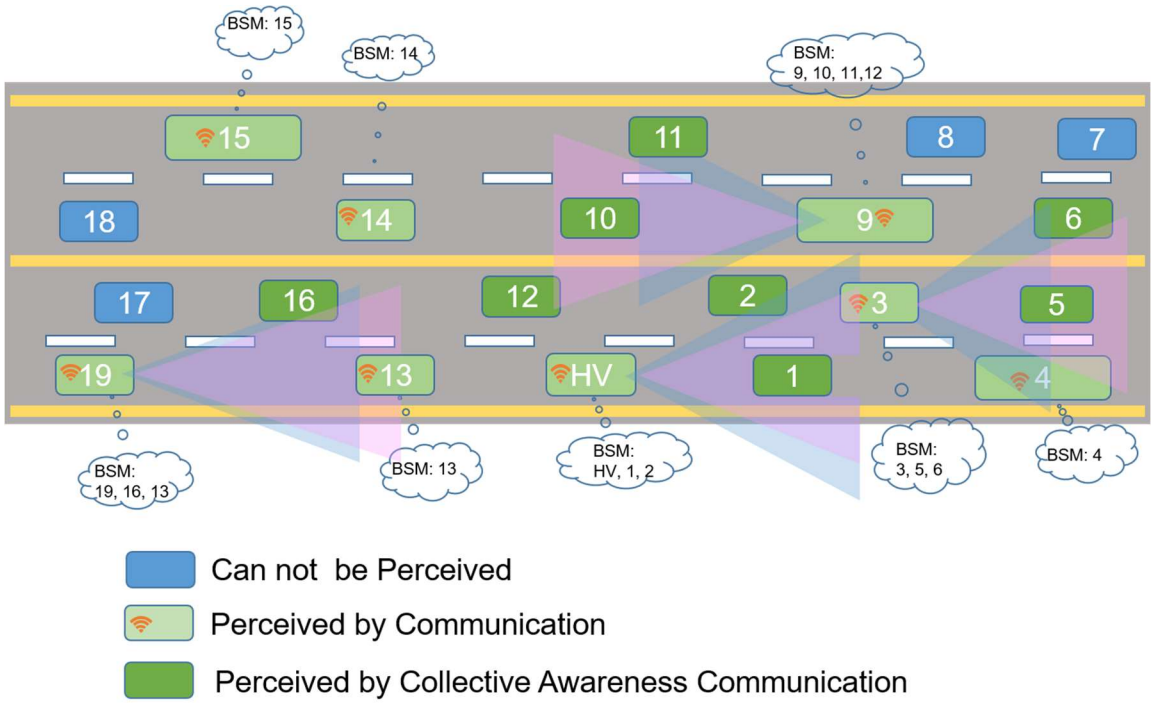


Figure 2 Cooperative Perception in a Mixed Traffic Simulation Environment.

As cooperative perception enables cooperative driving, its use is being standardized by the SAE [2]. A summary of the relationship between cooperation type and levels of automation is given in Figure 3. Motivated by the need of cooperative driving solutions, a cooperative perception architecture is developed in this research to improve the situational awareness of vehicles by sharing what is perceived. This solution corresponds to Class A (Status-sharing) level of Cooperative Driving Automation (CDA) cooperation class.

| | | SAE Driving Automation Levels | | | | | |
|---|---|---|---|---|---|---|---|
| | | No Automation | Driving Automation System | | Automated Driving System (ADS) | | |
| | | **Level 0** *No Driving Automation (human does all driving)* | **Level 1** *Driver Assistance (longitudinal OR lateral vehicle motion control)* | **Level 2** *Partial Driving Automation (longitudinal AND lateral vehicle motion control)* | **Level 3** *Conditional Driving Automation* | **Level 4** *High Driving Automation* | **Level 5** *Full Driving Automation* |
| **CDA Cooperation Classes** | **No cooperative automation** | (e.g., Signage, TCD) | Relies on driver to complete the DDT and to supervise feature performance in real-time | | Relies on ADS to perform complete DDT under defined conditions (fallback condition performance varies between levels) | | |
| | **Class A: Status-sharing** *Here I am and what I see* | (e.g., Brake Lights, Traffic Signal) | Limited cooperation: Human is driving and must supervise CDA features (and may intervene at any time), and sensing capabilities may be limited compared to C-ADS | | C-ADS has full authority to decide actions<br><br>Improved C-ADS situational awareness beyond on-board sensing capabilities and increased awareness of C-ADS state by surrounding road users and road operators | | |
| | **Class B: Intent-sharing** *This is what I plan to do* | (e.g., Turn Signal, Merge) | Limited cooperation (only longitudinal OR lateral intent that may be overridden by human) | Limited cooperation (both longitudinal AND lateral intent that may be overridden by human) | C-ADS has full authority to decide actions<br><br>Improved C-ADS situational awareness through increased prediction reliability, and increased awareness of C-ADS plans by surrounding road users and road operators | | |
| | **Class C: Agreement-seeking** *Let's do this together* | (e.g., Hand Signals, Merge) | N/A | N/A | C-ADS has full authority to decide actions<br><br>Improved ability of C-ADS and transportation system to attain mutual goals by accepting or suggesting actions in coordination with surrounding road users and road operators | | |
| | **Class D: Prescriptive** *I will do as directed* | (e.g., Hand Signals, Lane Assignment by Officials) | N/A | N/A | C-ADS has full authority to decide actions, except for very specific circumstances in which it is designed to accept and adhere to a prescriptive communication | | |

Figure 3 "Relationship between classes of CDA cooperation and levels of automation" [2]

As Cooperative Perception (CP) solution can enable several cooperative driving solutions, the sample ones are implemented to demonstrate the benefits achieved. In the first application, it is shown that the shared CP messages from other vehicles can increase the situational awareness of the host vehicle and improve the safety of vulnerable road users. Another advantage of the automation of the driving task is to be able to control the vehicle speed profile for better fuel economy and mobility. Therefore, in the second application, a use case scenario with smart intersections is presented. It is shown that if the perceived traffic information at a smart intersection is shared via CP messages, the connected vehicles can modify their speed profiles when they are approaching the intersection. Thus, the vehicles can improve their energy consumption by considering the traffic conditions at the intersection. Since the goal of this research is to contribute to cooperative and automated driving, as part of the vehicle automation, lateral and cooperative longitudinal controllers are developed to improve safety and comfort level of the road users.

### 1.2 Contributions of Dissertation

This dissertation builds connected and autonomous vehicle blocks which will enable cooperative driving and a better situational awareness as well as better fuel economy. To achieve these goals, first a cooperative perception architecture is proposed to improve the situational awareness of connected vehicles. The main components of the proposed architecture are object detection and tracking, and creation of CP messages. While there are many research results on CP architectures, the existing solutions are still in their early stages. Therefore, the proposed modular CP design is one of the few implementations of

CP in the literature which will help improve the situational awareness of connected and autonomous vehicles.

Integrating connected vehicle solutions to vehicles which are already equipped with other ADAS sensors brings the data association problem. When multiple sensors or sources (CP, V2X, camera, radar, ultrasound etc.) are used simultaneously to make threat assessment, vehicles should be able to identify which detections are coming from the same target and which ones are new tracks. Therefore, in this work a Mahalanobis Distance track to track association algorithm is developed and implemented in connected vehicles equipped with a smart camera and DSRC communication radio. With the proposed implementation, the data association between the tracks from camera and DSRC sensors can be efficiently used for associating the tracks reported by CP messages with the objects detected by the on board sensors.

Another main contribution of the dissertation is the designed Green Light Optimized Speed Advisory (GLOSA) with Traffic Preview. While conventional GLOSA systems are promising for improving fuel efficiency and reducing wait time at the traffic intersections, they do not consider the other traffic between the traffic light and the ego vehicle. Therefore, the advisory speed announced by conventional GLOSA systems is not accurate when there is a slower traffic in-front of the ego vehicle. To address this issue, two different solutions are proposed in this dissertation to utilize the smart intersection CP messages for generating more accurate speed advisories. Both of the proposed systems rely on spatial and temporal speed predictions generated from the CP messages. While the first proposed solution is rule based, the second one is a reinforcement learning controller which controls

the vehicle so that vehicle passes at the green light by avoiding unnecessary acceleration and decelerations. The proposed approaches show significant fuel economy benefits, and more comfortable ride.

Finally, to achieve higher autonomy, lateral and longitudinal controllers are designed. The designed lateral controller is a robust PD controller which considers the changes within the vehicle weight and speed operation range. The designed controller relies on the look-ahead controller approach and the performance of the designed controller is used interchangeably for both lane centering and path following tasks. The designed controller is tested in HIL simulation. For longitudinal control, a string stable Cooperative Adaptive Cruise Controller is designed and implemented. The designed controller is tested on an experimental vehicle.

### 1.3 Scope and Organization

This research explores connected and autonomous vehicle applications which enable better fuel economy, and safer transportation practices. In the upcoming chapters, the main components designed in this research to enable future connected and autonomous vehicles are presented. The scope of the following chapters is presented in this section.

In Chapter 2, the developed Cooperative Perception architecture is presented. This chapter details the designed CP architecture which employs lidar object detection algorithm and a Joint Probability Data Association (JPDA) filter object tracking algorithm. The section also covers the coordinate transformation and information on broadcasting practices for the tracked objects. Then, the practicality of the designed system is demonstrated with an example application.

In Chapter 3, a track-to-track data association algorithm for associating camera and V2X targets is presented. As a simple and effective method, the Mahalanobis Distance based Track to track data association algorithm is introduced and implemented. The designed system is then tested with the real-world data for two different scenarios.

In Chapter 4, the implementation of Pass at Green (GLOSA) algorithm in Hardware in The Loop (HIL) simulation is presented. The formulation of GLOSA and the Hardware in the Loop system used are explained. Then, the fuel efficiency benefit of the implemented algorithm is shown.

In Chapter 5, a new approach is presented to show how traffic preview can be employed to improve GLOSA performance. In this chapter, a representative traffic preview prediction approach which relies on cooperative perception data acquired from smart infrastructure is presented. Then, two GLOSA algorithms with traffic preview are given. While one of the developed algorithms is rule based, the other one uses reinforcement learning to control the vehicle. The effectiveness of both of the algorithms are shown with simulations.

In Chapter 6, a longitudinal controller is presented as part of the cooperative automated driving functionalities. As the longitudinal controller, a Cooperative Adaptive Cruise Controller (CACC) design is presented . The designed CACC controller is tested both in HIL simulation and also experimentally with the automated vehicle research platform of the Automated Driving Lab at The Ohio State University. It has been shown that the designed system can follow the vehicle in-front with the desired 0.6s time-gap.

In Chapter 7, a robust lane centering application is presented as part of the lateral controller task for the automated driving functionality. The designed system can use both lane lines

and GPS way points to keep the vehicle at the center of the road. The designed solution is tested in the HIL simulator.

As the final chapter, the summary of the dissertation and potential improvements and future research directions are presented in Chapter 8.

# Chapter 2. Cooperative Perception

Connected and Autonomous vehicles are improving traffic safety by using their on board sensors and communicated data. Recent studies on Cooperative Perception (CP) applications show that they improve the situational awareness of road users by enabling the sharing of perceived information. In this chapter, a cooperative perception architecture and its one implementation are presented. To implement the CP realistically, a customized game engine-based simulation environment is developed [3]. Then the multi object tracking algorithm of Joint Probability Data Association (JPDA) tracker is used to track objects and create CP messages. To demonstrate the effectiveness of the CP, a use case scenario is created and simulated. With the simulations, it is shown that situational awareness of the road users increases significantly, which improves the traffic safety of road users. This work demonstrates how to track and create CP messages and how they improve the situational awareness of road users. The presented work can be used as a starting point for implementing CP system in a vehicle or in smart infrastructure.

## 2.1 Background

Today's vehicles are getting more automated then ever thanks to the advances in the automotive sensors and computational resources. While most of the automation comes in the form of driver assistance technologies, the automotive industry is racing towards the goal of developing fully autonomous vehicles. Since the automated driving task is a safety critical task, it is essential to utilize multiple sensors as a redundancy measure to overcome shortcomings of sensors. Therefore, the typical sensors used for vehicle automation include combination of lidar, camera, radar, and ultrasonic sensors [4]. As an addition to these

sensors, vehicle to everything (V2X) communication is another sensing technology which is proven to be effective for increasing the situational awareness of vehicles. This technology is especially effective in providing information about traffic participants that are not in line of sight. This technology relies on the communication channel between the vehicle and other road users. Considering the number of sensors deployed on automated vehicles and smart infrastructure, a new opportunity arises to enhance the connected vehicle applications significantly. By sharing what they see, the vehicles and smart infrastructure could share the information of non-communicating road users which are detected by their sensors. Therefore, connected vehicles will be aware of the other road users which are normally outside their perception range even if they are not communicating their own status. Hence, CP increases the situational awareness of vehicles which allows automated vehicles to perform threat assessment much earlier and increase safety. Additionally, the information received from CP can be another redundancy in the system or it can be used to improve detection accuracy.

As explained above, the use of Vehicle-to-Everything (V2X) communication as an extra sensor significantly increases the total field of view of the vehicle. Extension of the field of view helps to eliminate accidents due to the occlusion of obstacles/objects. Also, sharing the detected objects through V2X communication (cooperative perception) can increase the effectiveness of existing V2X applications [5]–[7]. For example, the authors of [6], implemented a cooperative perceptions system, which has a camera and Dedicated Short Range Communication (DSRC) onboard unit (OBU). They showed that when the target/remote vehicles are reported by the cooperative perception system, V2V

applications, such as Left Turn Assist (LTA), Intersection Movement Assist (IMA), and Blind Spot Warnings (BSW) still perform well even though the cooperative perception data processing introduces delays.

It is worth noting that, although the cooperative perception concept is relatively new, it has been studied from different perspectives by the researchers. Also, it is being standardized by both the European Telecommunications Standards Institute (ETSI) [8] and the Society of Automotive Engineers SAE [9], [10].

One of the main problems in cooperative perception is efficient utilization of communication bandwidth. Considering that each vehicle can broadcast what they perceive, the targets can be broadcasted more than once, creating unnecessary occupation of the communication channel. To address these issues several redundancy methods are highlighted in the ETSI in detail [8]. The common approach for the redundancy measures is to keep useful data based on a metric. Some of the metrics considered for reporting a target via cooperative perception are, confidence level of the detections, distance of the detected objects, and dynamics of the objects. One should note that, although each of these mitigation approaches helps to reduce the transmitted data size, they all have to compromise by losing some important data.

As an application of cooperative perception, Shan et al. [11] demonstrate the use case scenario for cooperative perception where connected vehicles can communicate with an Intelligent Roadside Unit (IRSU) and avoid potential accidents with Vulnerable Road Users (VRU)s. They experimentally tested their system in a scenario where they share the tracked object information with autonomous vehicles to demonstrate the effectiveness of

their application [11]. In [12] Gunther et al. implemented a cooperative perception application and in a simulation environment, they showed that cooperative perception can be used to improve reaction time of the vehicles to obstacles on the road. Checking the literature, it can be seen that the common consensus to realizing cooperative perception is by sharing detection level information rather than sharing the raw data. This is because sharing raw data requires large communication bandwidth and high processing power especially considering the increasing number and quality of sensors on the vehicles. However, as stated in [8] if the communication channel resources permits, like future ^G communication, it might be possible to include raw sensor data in the cooperative perception messages.

As cooperative perception is still far from being a mature field and its benefits are promising to improve overall situational awareness of connected vehicles, a cooperative perception architecture is proposed in this work. To simulate the developed architecture, a customized simulation environment is developed. In the created simulation environment, the potential benefit of cooperative perception is demonstrated.

## 2.2 Simulation Environment

There are many commercial and non-commercial simulators that can be used to develop autonomous vehicles. In this work, Carla is selected because of its flexible architecture, which allows extending the simulator capabilities [13]. In Figure 4, one can see how this simulator can be extended with other simulation environments. Most of the shown co-simulation capabilities are supported by Carla. For example, ROS bridge, Autoware, Sumo, and Vissim co-simulation environments are directly supported by Carla. Also, one can

create new maps and simulation environments with MATLAB RoadRunner and import them to Carla. Considering that Carsim, MATLAB, and Carla software can communicate with Unreal Engine, it is possible to create co-simulation environments where the vehicle dynamics are simulated in MATLAB or Carsim in a Carla simulation. Finally, by using the property of MATLAB for running Python scripts, one can utilize MATLAB - Carla co-simulation. For the presented work, we decided to use a subset of the shown simulation architecture. Since the demonstrated work will mostly focus on cooperative perception applications, we are primarily interested in range sensor data, such as lidar. Therefore, our co-simulation subset consists of MATLAB – Carla simulation environment with Sumo co-simulation. While the MATLAB interface is used to transfer sensor data to MATLAB, with the addition of Sumo co-simulation we can also inject traffic into our simulation environment.

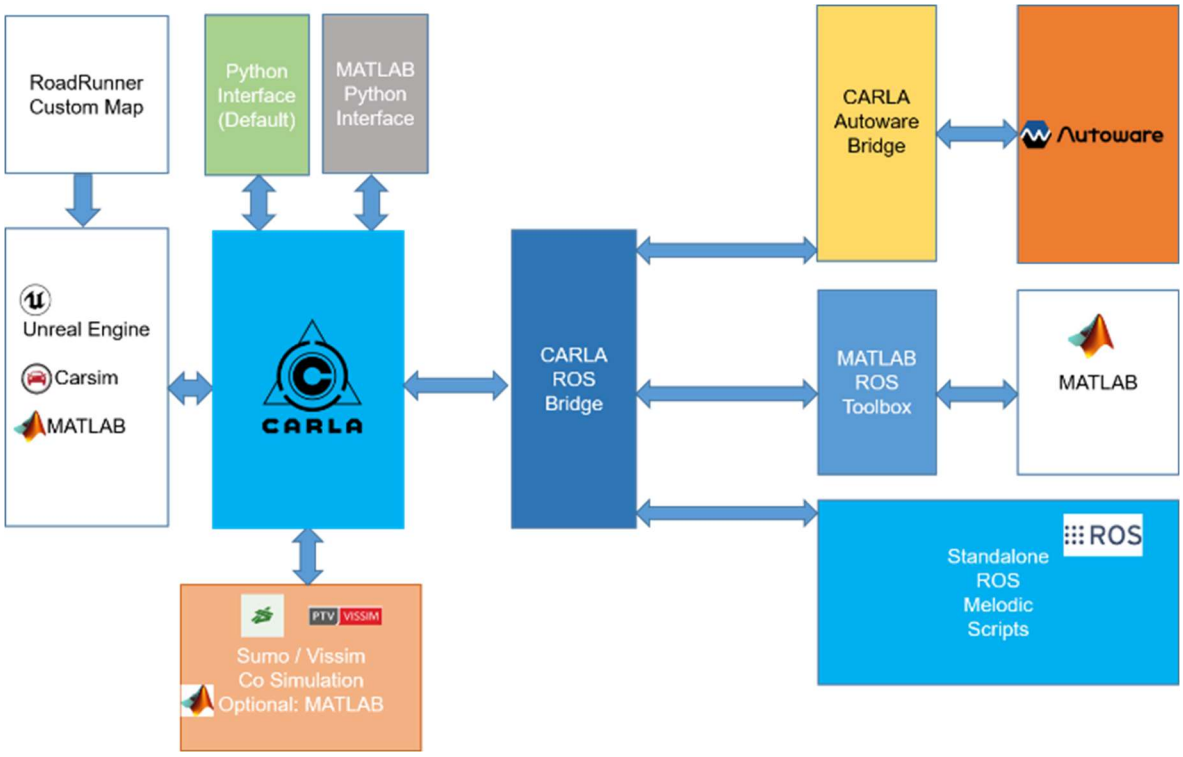


Figure 4 Carla Simulator extension diagram [3].

### 2.2.1 Python Interface

The main interface for the Carla simulator is Python scripts. By using PythonAPI, it is possible to select maps, change the weather, spawn vehicles, pedestrians, and other actors, control the actors, and more. In our application, we set our simulation environment using our custom Python script based on the target task, such as sensor fusion or car following.

### 2.2.2 ROS Interface

The Robot Operating System (ROS) is a robotics middleware which eases managing multiple processes in a system. It provides services to exchange common messages between processes and lets developers control hardware. ROS also allows users to create packages that can be reusable in different systems. The latest version of Carla (version

9.13) supports both ROS and ROS2 by having a ROS bridge. In this work, the ROS interface is used to connect to Autoware and MATLAB to collect data and interact with the Carla simulator. Autoware can be utilized for path planning, localization, object detection, and lateral & longitudinal control tasks. The implemented Carla-Autoware co-simulation is presented in Figure 5.

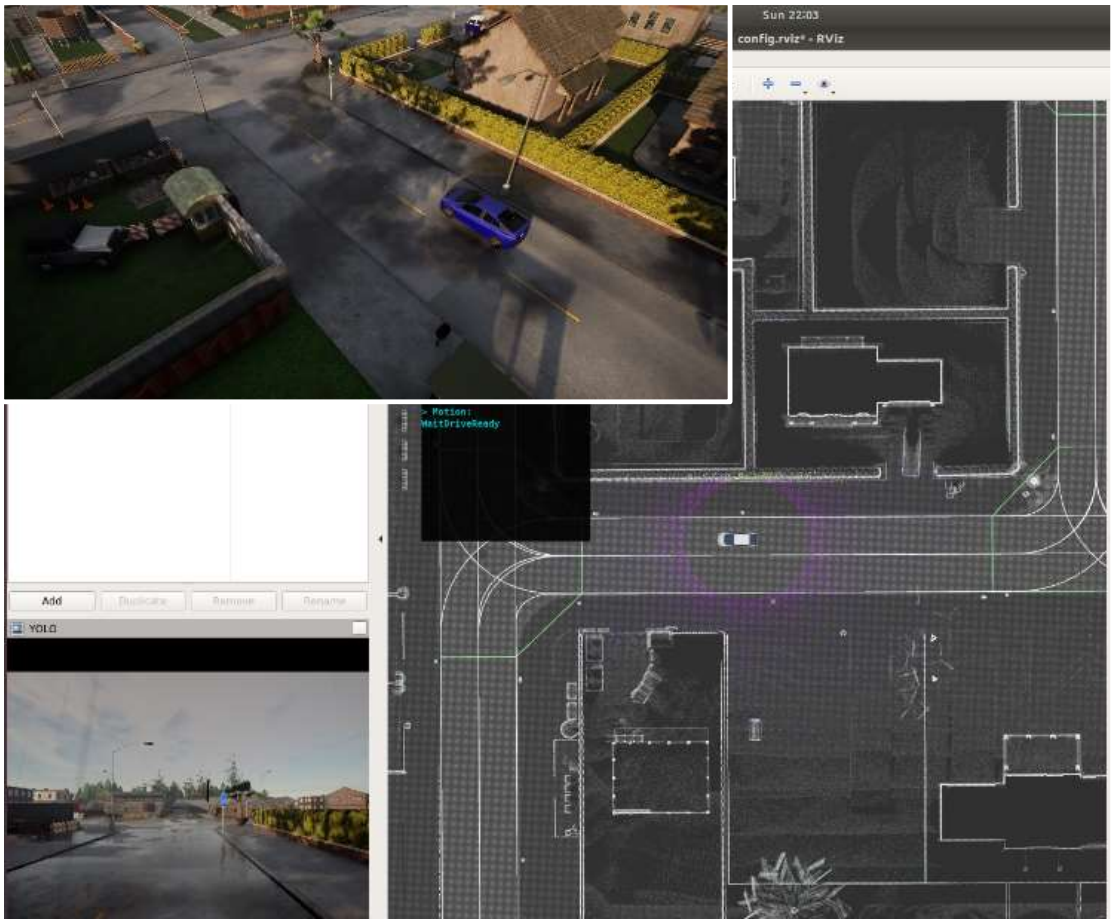


Figure 5 Top Left: Carla simulator interface. Background: Snapshot of Rviz ROS interface to Carla Autoware integration showing point cloud map, HD map, LiDAR, and camera sensor outputs [3].

### 2.2.3 MATLAB Interface

As explained in previous sections, there are two main interfaces for Carla, which are the ROS and Python script interfaces. Since MATLAB supports both of these interfaces, MATLAB - Carla simulation can be realized by using one of them. While Python scripting is supported directly within MATLAB scripts, to interact with the Carla ROS interface, the MATLAB ROS toolbox is used. We built our MATLAB Carla integration based on [14]. One should note that the MATLAB ROS integration is limited by the implemented features in the Carla ROS Bridge. On the other hand, the Python interface in MATLAB provides much more flexibility for customizing the simulation environment.

### 2.2.4 Traffic Simulation

Carla creates traffic around the host vehicle by randomly spawning vehicles. For evaluating the potential mobility or fuel-efficiency benefits of connected and autonomous vehicles, enhanced traffic simulations can be used to create realistic traffic simulation. Some of the examples of these applications can be seen in [15]–[17]. Among many of the micro traffic simulators, Sumo and PTV Vissim are the most commonly preferred simulators in use. They are also supported by Carla. In Figure 6, a screenshot from our sample Sumo- Carla co-simulation implementation can be seen. The Sumo road network is shown on top and the corresponding Carla road environment is shown at the bottom in Figure 6.

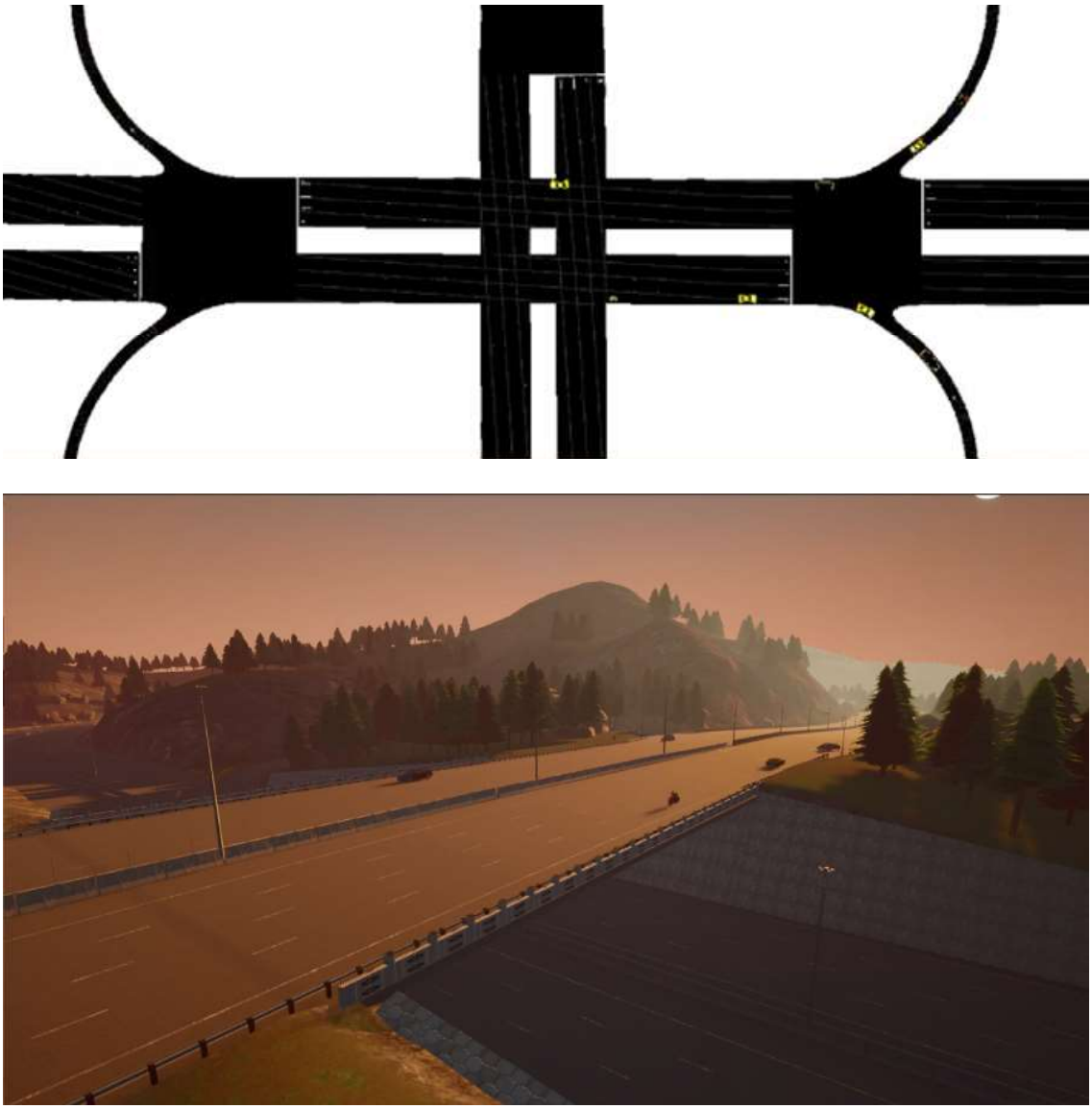

Figure 6 Carla – Sumo Integration [3].

### 2.2.5 Vehicle Dynamics

Vehicle dynamics in Carla are modeled using the NVIDIA PhysX Vehicle, and which does not accurately represent realistic vehicle dynamics. However, to develop and analyze control algorithms for AVs, the use of a realistic vehicle dynamics model is required to

develop algorithms which can handle the dynamic behavior of the vehicle under consideration. In some cases, having the full vehicle model with powertrain also helps to develop more accurate fuel-efficiency algorithms. To address this, it is proposed to extend the simulation by integrating it with CarSim as shown in [18] or MATLAB vehicle models. Models developed in MATLAB can be generated using its recent Vehicle Dynamic Toolbox, or it can be a completely custom model. In the Vehicle Dynamics co-simulation environment, vehicle dynamics would be simulated in Carsim or MATLAB, while the photo-realistic simulation environment and other simulated sensor data come from Carla.

## 2.3 Cooperative Perception System Architecture

The designed and implemented Cooperative Perception system architecture is shown in Figure 7. While it is possible to fuse from more than one sensor data and share what is perceived, as part of a proof-of-concept architecture, a 360-degree lidar sensor is selected as the main sensor for this work. The lidar outputs point cloud raw data. Then the in the bounding box detector block selects object cluster and puts a bounding box around them. Then these are sent to the Multi Object Tracker to be associated with existing tracks or to create new ones. False detections are also removed in the Multi Object Tracker stage. From the track list, BSM messages are created as Cooperative Perception Message (CPM) by converting detection locations to global coordinate system. One can easily improve the effectiveness of the system by fusing more than one sensor instead of using only lidar.

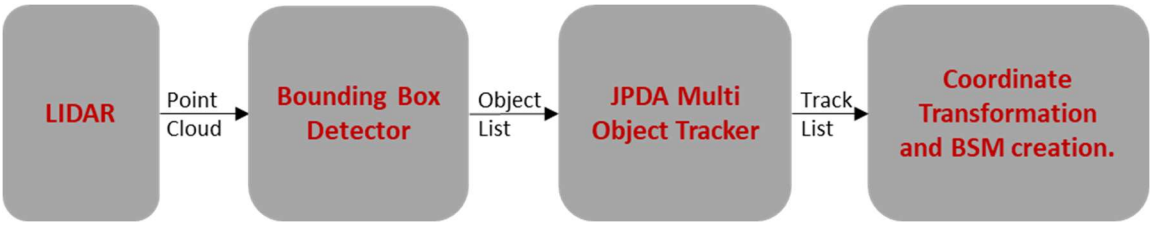


Figure 7 Cooperative Perception System Architecture.

### 2.3.1 Lidar sensor and Multi Object Tracking

One of the main reasons for choosing lidar over other sensors is its accuracy and its 360° field of view. For the implementation of the architecture in Figure 7, the simulation environment presented in section 2.2 is used. In the Carla simulation environment, the vehicles can be equipped with custom sensor combinations. In the presented work, the vehicles are customized such that they have 360° field of view lidars and a forward camera. The lidars on the vehicle are modeled based on a 64 channel Ouster OS0 digital lidar [19] which has 90° vertical and 360° horizontal field of view. The lidar can measure 1,310,720 points per second.

The object detection from raw lidar data can be achieved in multiple ways including neural network models such as PointPillars [20], Frustum Pointnets [21]. In this work, the object detection is achieved by processing the raw point cloud data using the nearest neighborhood clustering algorithm. While the code for this approach is given in [22], the final implementation is done in MATLAB as in [23].

Obstacle detection from raw lidar point cloud is done in three main steps:

1- Points outside the region of interest are removed.

2- Ground plane points are detected and removed.

3- Remaining point clouds are clustered and a bounding box for each cluster is created.

Among these three steps the first step is very straightforward. One can simply remove the raw points which are not of interest to the operation of the vehicle. This helps to improve the speed of the data processing and removes unwanted detections such as the roof of the ego vehicle from the raw data measurement.

As a next step the road surface or ground plane is segmented. For this purpose, the Random Sample Consensus, RANSAC algorithm for 2D plane detection can be utilized. RANSAC is an iterative algorithm to determine outliers in data. The detailed explanation of the RANSAC algorithm can be seen for finding outliers in a 2D data in [24]. Since the lidar data is a 3D, our ground plane can be considered as the data that we are interested in, and the obstacles can be considered as the outliers. The pseudocode of the iterative ground plane segmentation is given below in Algorithm 2.1 [25]. The implementation [22] results of the algorithm is shown in Figure 8. In the figure, one can see that the green points are the part of ground plane (inliers) founded by the RANSAC algorithm and the red points are the part of obstacles on the road.

**Algorithm 2.1: RANSAC Ground Plane Detection**

***Inputs:*** *Raw Point Cloud Data, max number of iterations, distance tolerance*

***Output:*** *Ground Plane (Road), Outliers (Obstacles)*

*for (max number of iterations)*

*Select 3 random data points from the point cloud*

*Fit a plane to selected points*

*Log the inliers (data points that are within the distance tolerance from the plane)*

*Form the ground plane with the plane max number of inliers.*

*Segment out remaining data points(obstacles) as obstacles.*

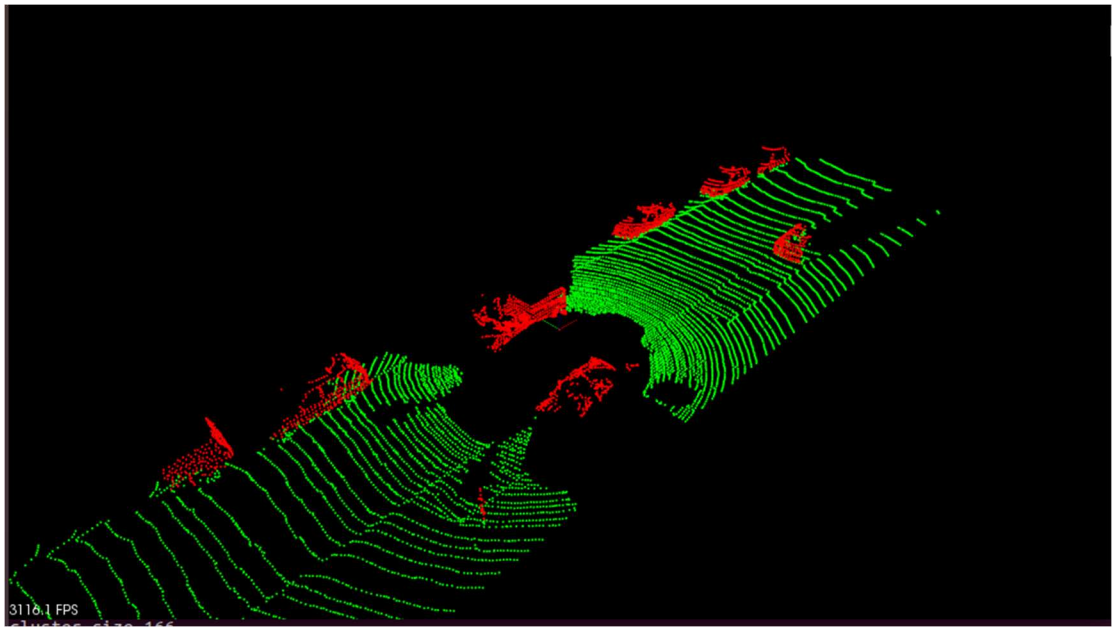


Figure 8 Ground plane segmentation from raw lidar point cloud data.

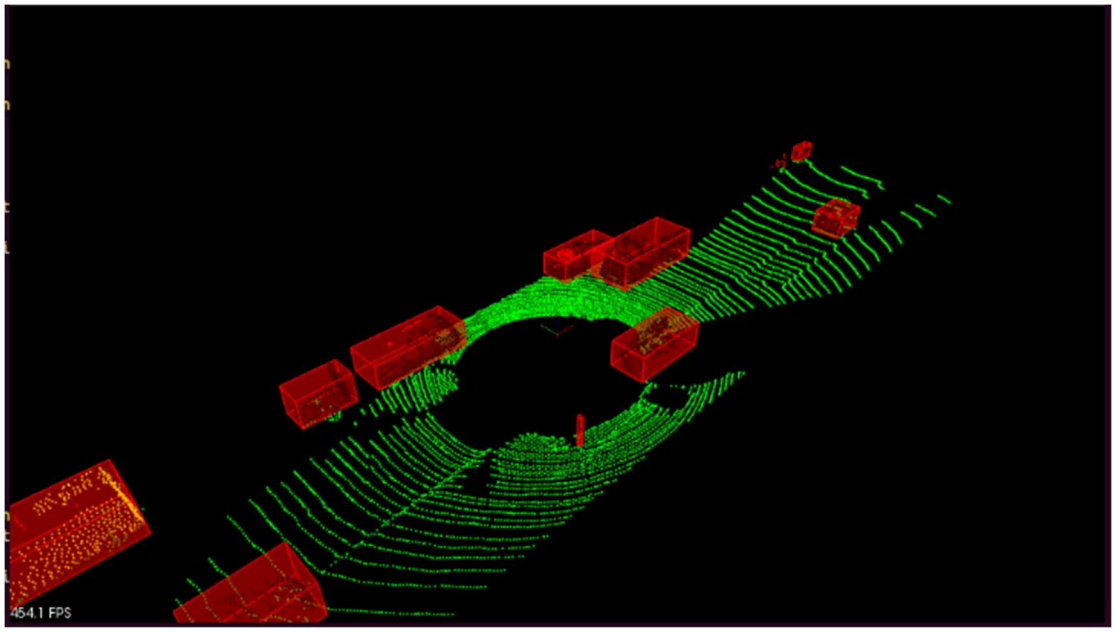


Figure 9 Obstacle clustering and bounding box creation.

After segmentation of the ground plane, one needs to cluster each obstacle in the remaining point cloud data. The points which are close to each other can be clustered using Nearest Neighborhood algorithm or Euclidean Cluster [26] algorithm which clusters the points whose Euclidean distance to each other is within the predetermined distance tolerance. To increase the speed of this algorithm, one can use the k-d tree algorithm [27]. K-d tree algorithm starts searching from the root data point then partitions the search space to narrow down the search space. Once clusters are formed, bounding boxes are created for the clustered objects. The created clusters and corresponding bounding boxes using the implementation in [22] are presented in Figure 9.

In the last step, the detected bounding boxes are fed to a Joint Probability Data Association (JPDA) Multi Object Tracker from the MATLAB Sensor Fusion Toolbox to track targets

[28]. The JPDA tracker has the functionality to create and maintain tracks based on their age. When the tracker receives the measurement list, the tracker updates the existing tracks with the measurements which lie in the validation gate of the existing tracks. While the JPDA tracker can handle more than one sensor, to simplify the architecture, the solution is presented only for the lidar detections. An example of the JPDA object tracking from lidar data can be seen in Figure 10.

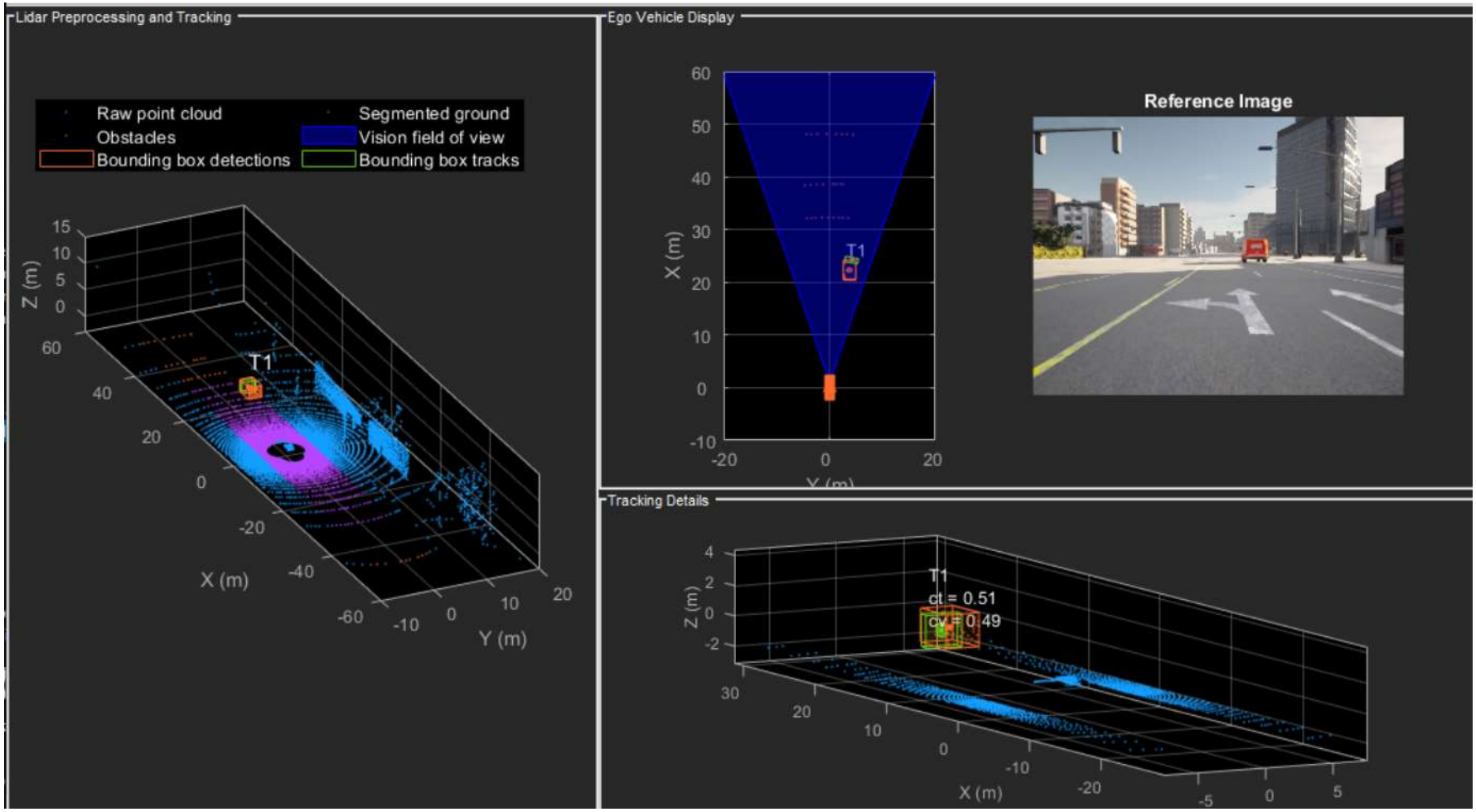


Figure 10 JPDA object tracking example from lidar raw sensor data.

### 2.3.2 Coordinate Transformation & BSM Creation

After detecting the target vehicles with range sensors (such as lidar), position, speed, and heading information of the target vehicles should be shared with other vehicles. While the range sensor measurements and GPS sensor measurements are fused at the vehicle coordinate system, the position of the targets needs to be converted to the geographic coordinates to be transmitted by the V2X communication module. One can refer to Figure 11 to see Universal Transverse Mercator (UTM) and the vehicle coordinate systems. The coordinate transformation between the local and UTM coordinate systems requires the knowledge of the host vehicle position, heading and speed information, which can be received from the GPS of the V2X communication module.

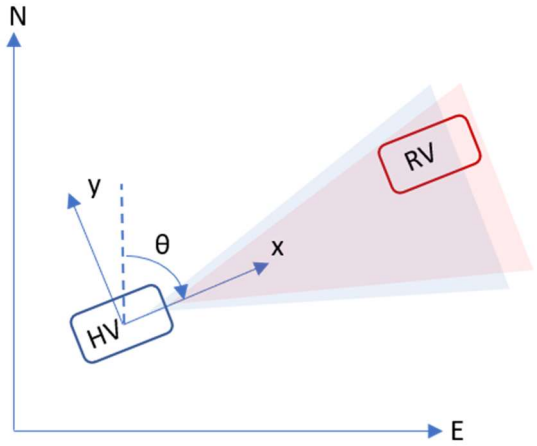


Figure 11 UTM and vehicle coordinate systems. Transparent regions in front of the host vehicle (HV) show the FOVs of the range sensors.

Figure 12 shows the coordinate transformation process used in the designed system. First, remote vehicles (RV) are detected at the vehicle coordinate system. Then, their position is converted into the UTM coordinate system by rotating the relative position of the target vehicles and adding the rotated relative positions to the UTM coordinates of the host vehicle. The rotation matrix is given in Equation 2.1. The resultant UTM coordinates then transferred to the geographic coordinate system. For the UTM coordinate system to geographic coordinate system conversion, a MATLAB function in [29] is used.

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{2.1}$$

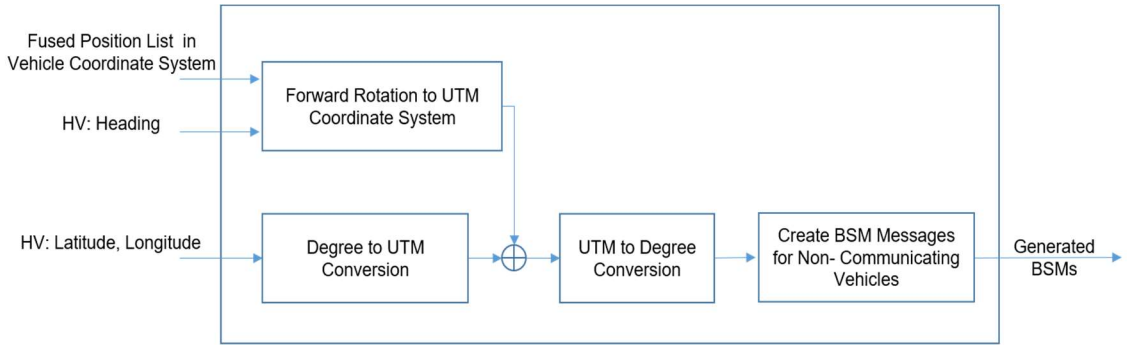


Figure 12 Coordinate transformation & BSM generation process [3].

After converting UTM coordinates to global coordinates, BSM messages should be created for the tracked vehicles. One should note that BSMs should be created only for the vehicles that are not broadcasting their BSMs. This requirement can be met by using a track -to-track data association algorithm to identify if the tracked vehicles are already broadcasting their own BSMs. An example of the algorithm will be explained in the Chapter 3.

## 2.4 Pedestrian Scenario

Motivated by the capabilities of the cooperative perception architecture, a use case scenario for the cooperative perception algorithm is constructed in the customized simulation environment. A screenshot of the scenario is shown in Figure 13. In this scenario, a pedestrian is already in the cross walk, but she is not visible to the host vehicle because the perception of the host vehicle is obstructed by a large truck. Since the truck is equipped with a cooperative perception sensor module, it can increase the situational awareness of the host vehicle and help avoid a serious accident. In this scenario, it is shown that the host vehicle is aware of the pedestrian as long as she is detected by the lidar sensor on the truck and shared with other vehicles through cooperative perception.

Figure 13 Right bottom foreground: Host vehicle camera image, Background: scenario image from an external camera showing view that is obstructed by truck [3].

## 2.5 Summary

In this section a cooperative perception architecture and its application to improve the safety of vulnerable road users is presented. The presented architecture consists of lidar object detection and a JPDA multi object tracker. The tracked object information can then be converted to the global frame to share the location and speed information of detected objects. Also, to simulate the CP applications, a customized simulation environment was developed. Then, a practical scenario is simulated in the designed simulation environment. By tracking the pedestrian location with the lidar sensor and sharing its location, the situational awareness of the host vehicle is improved significantly.

Another possibility for implementing the cooperative perception is placing the range sensors at the intersections where most of the interactions between the vulnerable road users and vehicles occur. The intersections equipped with range sensors and communication modems are called smart intersections and they can share the location and speed information of vulnerable road users and vehicles at the intersection. In Chapter 5, an improvement of the Green Light Speed Advisory application is presented by utilizing the data shared by smart intersections.

# Chapter 3. Data Association

The connectivity between vehicles, infrastructure, and other traffic participants brings a new dimension to automotive safety applications. Soon, all the newly produced cars will have Vehicle to Everything (V2X) communication modems alongside the existing Advanced Driver Assistant Systems (ADAS). It is essential to identify the different sensor measurements for the same targets (Data Association) to use connectivity reliably as a safety feature alongside the standard ADAS functionality. Considering that the camera is the most common sensor available for ADAS systems , an experimental implementation of a Mahalanobis distance-based data association algorithm between the camera and the Vehicle to Vehicle (V2V) communication sensors is presented in this chapter. The implemented algorithm has low computational complexity and the capability of running in real-time. One can use the presented algorithm for sensor fusion algorithms or higher-level decision-making applications in ADAS modules.

## 3.1 Background

Many automotive companies invest in connected vehicle applications, such as Left Turn Assist (LTA), Intersection Movement Assist (IMA), and Collison Avoidance (CA). While these communication-based driver-assist technology applications aim to reduce the number of accidents on the road, some existing driver-assist technologies aim to avoid accidents by detecting the threats using range sensors such as camera and radar. It is crucial to accurately identify if both systems refer to the same target to warn the driver in a potential collision scenario. This problem is called a data association problem. This chapter focuses on identifying the association between the two different sensor measurements of a camera

(range sensor) and a Vehicle to Vehicle (V2V) communication modem. After the data association step, a higher-level module, i.e., threat assessment module, would use the data association results to take necessary safety precautions.

There are three main categories for the data association problem which are used for tracking and sensor fusion applications. The first one is the measurement-to-measurement association, which is also called the track initiation problem. The second one is the measurement to track association, which is also called a track maintenance problem. The final one is the track-to-track association problem, which is also called the track fusion problem. Among these, this work focuses on the implementation of the track-to-track data association.

This part will summarize some of the data association work in the literature. Some of the commonly used perception/range sensors in ADAS are camera, lidar, radar, and ultrasonic sensors. Different working principles of these sensors result in different resolutions, ranges, and detection rates depending on the sensor's physical characteristics and environmental conditions. The strength, weaknesses, and working principles of these sensors are presented in [4]. Connected and autonomous vehicles use Vehicle to Everything (V2X) [30] communication as another sensor alongside the existing perception/range sensors, which improves connected vehicles' self-awareness.

In autonomous vehicles, multi-sensor multi-object tracking modules handle sensor fusion tasks, which increases the accuracy of detections by combining the measurements from multiple sensors. One essential stage of the multi-sensor multi-object tracking is the data association. In [31], the authors developed a track to track data association algorithm,

which compares the Mahalanobis distance between tracks from multiple sensors. Similarly, in [32], the authors used minimum Mahalanobis distance with the Chi-square test for their application's association problem. In [33], [34] the Hungarian Algorithm is used for data association. Alongside these methods, there are also some other probabilistic data association algorithms in the literature for target tracking. Some of these are Probabilistic Data Association Filter (PDAF), Joint PDAF (JPDAF), and Multi Hypothesis Tracking (MHT) algorithm. In PDAF, instead of using hard assignments, multiple detections inside the validation gate are considered to update the current track [35]–[37]. The PDAF is more appropriate for single track scenarios than multi-track scenarios. Therefore, JPDAF is developed for multiple target scenarios [38]. The JPDAF works by calculating the joint probability distribution between the tracks and measurements. Other methods used for multi-object tracking data associations are Multiple Hypothesis Tracking (MHT) [39] and Markov Chain Monte Carlo Data Association (MCMCDA) [40], [41] and Markov Decision Process [42]. The application area of these data association algorithms can be extended to autonomous vehicles as well [1]. Some of the examples of data association applications in autonomous vehicle research can be seen in [1], [42], [43].

As for the computational load, Mahalanobis distance is the simplest algorithm, which is calculated by the square of the error with respect to its covariance [44]. In PDAF, it is assumed that only one target is tracked, and it is already initialized. PDAF algorithm calculates data association probabilities for each measurement to target track which are used in the PDAF tracking algorithm [44]. The Joint Probability Data Association Filter (JPDAF) application is the extended version of PDAF. JPDAF jointly computes the

measurement to target associations for multiple established tracks. Another algorithm to be considered is Multi Hypothesis Tracking (MHT) Algorithm. MHT algorithm creates multiple data association hypotheses to track the targets in a cluttered environment. This allows the algorithm to postpone challenging data association decisions to a time when more data is available [45]. The main drawback of this algorithm is that it is costly to maintain the growing number of data association hypotheses. As we look at the comparison of the aforementioned data association and target tracking algorithms with data association functionality, it is also important to differentiate the type of the data association task used. Most of the algorithms in the presented literature, such as PDAF, JPDA, and MHT [39], [45] and others, are appropriate for measurement to track association, and they are used to track the targets with a filter stage. Considering that this research focuses on the implementation of a track-to-track data association rather than target tracking, a Mahalanobis distance-based data association algorithm is preferred.

In the rest of the chapter, first, the experimental setup and overall architecture with an explanation of the algorithm used is presented. Then, the results and conclusion sections conclude the chapter.

## 3.2 The Experimental Setup and System Architecture

This section describes the overall architecture of the designed and implemented data association architecture. Our experiment setup is used to depict the general architecture. In our experiment, a host vehicle (HV) and two remote vehicles (RV_1 and RV_2) are used. HV is equipped with a camera and V2V communication onboard unit (OBU), and remote vehicles are fitted with only V2V OBUs. The video stream from the host vehicle and high

accuracy RTK GPS measurements for all the vehicles are recorded for ground truth creation purposes. The HV data association system architecture is shown in Figure 14, which consists of measurements, coordinate transformation (for V2V measurements), synchronization and filtering, buffering, and track to track association modules. When reading the rest of this section, readers can refer to Figure 14 to see each module's relationship with other modules.

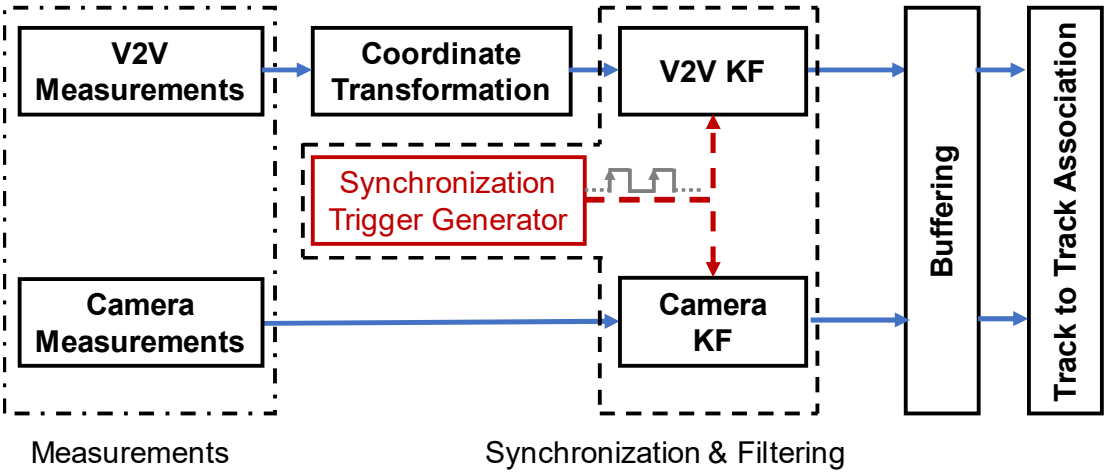


Figure 14 The designed data association architecture for the host vehicle. The architecture consists of measurement, coordinate transformation, synchronization & filtering, buffering, and track to track association modules [46].

## 3.3 Measurements and Associable Parameters

Most of the new vehicles have a camera sensor as a standard ADAS sensor. Although some of the vehicles also have radar, it is generally available for higher-end vehicles with Adaptive Cruise Control functionality. Therefore, this study focuses on Track to Track Association implementation between the most common ADAS sensor, the camera, and the

V2V communication modem. If required, one can quickly adapt the same algorithm to include more than two sensors. In our experimental vehicle architecture, the host vehicle is equipped with an automotive-grade smart, forward-looking camera. The measurement rate of the camera used is around 40 Hz with a 100° field of view. The camera can detect other vehicles on the road and track them. Therefore, the camera has its own object management algorithm, and it publishes the tracked objects' IDs alongside their spatial position, dimensions, type, relative speed, and bearing angle. The host vehicle is also equipped with a V2V OBU, which transmits and receives Basic Safety Messages (BSM) to communicate with other traffic participants. The full definition of BSM, which is a message set broadcasted by each connected vehicle at 10Hz is given in [47]. This message set has the ID, global position, heading, speed, dimensions, and path history of the vehicle. Although both sensors offer other measurements, the measurements shown in Table 1 are comparable with each other. While this setup is the minimum requirement for the host vehicle, remote vehicles do not necessarily have to have this whole setup. They only need to have V2V OBUs to transmit their BSMs.

| Camera | V2V |
|---|---|
| Id | Id |
| Processor time | UTC Time |
| Lateral Distance | Latitude |
| Longitudinal Distance | Longitude |
| Relative Heading | Heading |
| Relative Speed & Speed | Speed |
| Length | Length |
| Width | Width |

Table 1 Some of the camera and V2V measurement parameters

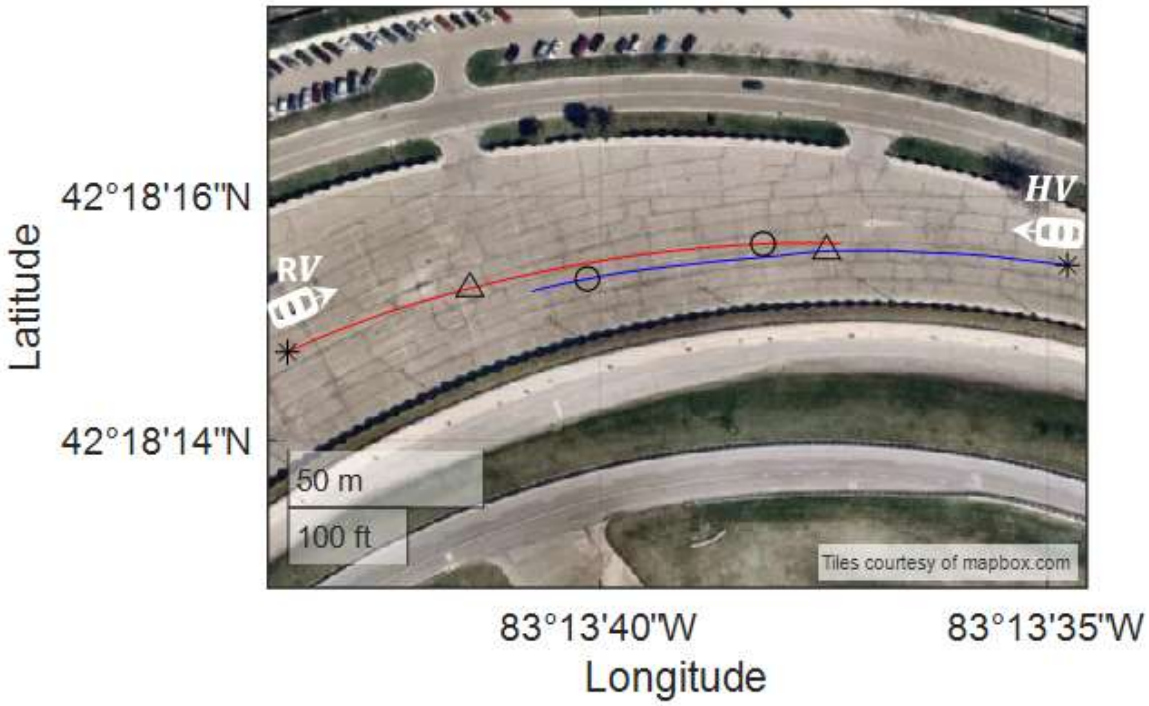


Figure 15 Paths were followed by the host and remote vehicles in the global coordinate system. "*" represents the starting location of the paths. Vehicles start to move from the '*' mark and proceed to 'Δ' and 'o', successively. For example, at timestep t, both vehicles will be at 'Δ' on their path.

It is essential to determine associable parameters between the two automotive sensors considered here: the camera and the V2V modem, to implement the data association module. This task aims to find common and comparable parameters between these two sensors, which will help identify an ADAS target. In Table 1, the relevant measurement parameters for the automotive-grade smart camera and V2V module are listed. We conducted experiments to verify which one of these parameters is associable. In the experiments, a host vehicle and a remote vehicle drive in opposite directions towards each

other in a slightly curved path, as shown in Figure 15. Vehicle starting location and positions with five-second intervals are marked with '*', 'Δ', and 'o' successively on their trajectories to reflect vehicle motion through time. After having all measurements in the host vehicle coordinate system, measurement parameters are analyzed, to be used for the data association task. It is observed from the plots in Figure 16 that the relative distance/offset measurements from the camera and V2V are comparable to each other. Similarly, relative heading between the Host Vehicle and Remote Vehicle and lateral and longitudinal velocities from the camera and V2V sensor are also comparable with each other. Therefore, these parameters are chosen as associable parameters. However, the Remote Vehicle (length and width) size measurement from the camera do not match with actual dimensions published via V2V communication. Therefore, vehicle size measurements are not considered as reliable parameters for the data association task.

### 3.4 Coordinate Transformation

In the constructed architecture, the measurements from the V2V sensor are in the global coordinate system, whereas the measurements from the camera are in the vehicle coordinate system. In Figure 17, one can see the global and local coordinate frames for the host and remote vehicles. In order to associate measurements between the HV sensors, both measurements must be in a common coordinate system. For this purpose, the V2V measurements representing the remote vehicle positions are transformed into the host vehicle coordinate system. Thus, it becomes possible to compare the camera measurements with the V2V measurements. The coordinate transformation mentioned requires the knowledge of the Host Vehicle (HV) and Remote Vehicle (RV) global coordinates and

heading which are measured by their V2V modems. In the translation step, the global coordinates (latitude, longitude) of the HV and RV are transformed into Universal Transverse Mercator (UTM) grid coordinates in (3.1) where the deg2utm function is used from [48]. Then, the relative distance between the HV and RV is calculated in the North and East directions as given in (3.2) and (3.3). In the coordinate transformation step, the calculated relative distances in the global coordinate system are transformed into the HV coordinate system in (3.4). Finally, the relative distance $d_r$ between the host and remote vehicle can be calculated as given in (3.5).

$$[N, E, utmzone] = \text{deg2utm}(lat, long) \tag{3.1}$$

$$\Delta E = E_{RV} - E_{HV} \tag{3.2}$$

$$\Delta N = N_{RV} - N_{HV} \tag{3.3}$$

$$\begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} \cos(\theta_{HV}) & -\sin(\theta_{HV}) \\ \sin(\theta_{HV}) & \cos(\theta_{HV}) \end{bmatrix} \begin{bmatrix} \Delta E \\ \Delta N \end{bmatrix} \tag{3.4}$$

$$d_r = \sqrt{{p_x}^2 + {p_y}^2} \tag{3.5}$$

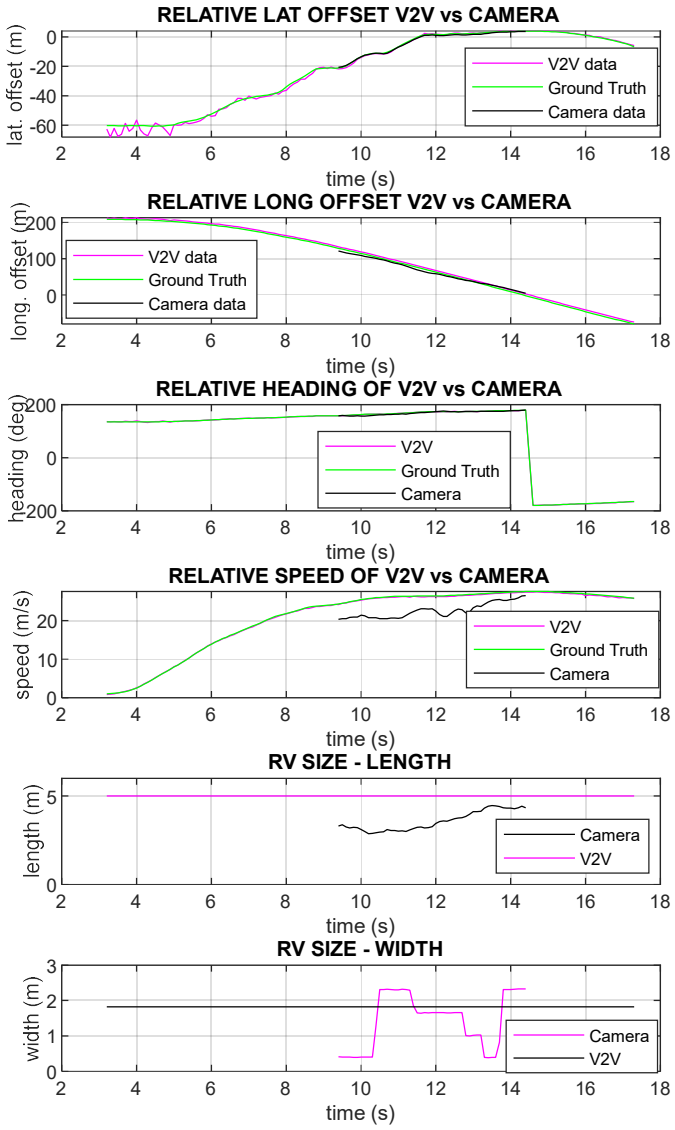


Figure 16 Camera and V2V sensor measurement performance comparison for when HV and RV are approaching each other, as shown in Figure 15. Measurements are received via camera and V2V sensors, while the ground truth data is acquired from high precision RTK GPS [46].

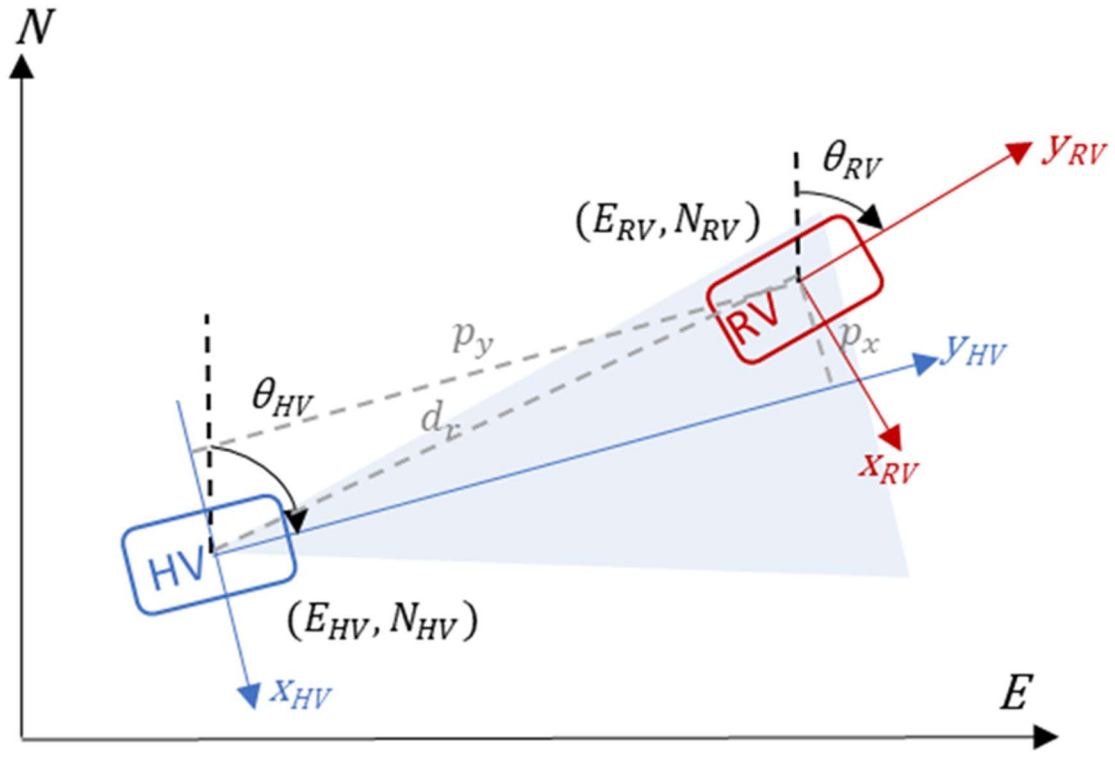


Figure 17 Sample host vehicle and remote vehicle coordinate systems with respect to the global coordinate system. These coordinate systems are used for coordinate transformation of V2V measurements from global coordinate system to host vehicle coordinate system. The blue transparent region represents the field of view of the camera [46].

### 3.5 Synchronization and Filtering

For filtering the camera and V2V track measurements, Kalman Filters (KF) are used for each sensor type. The purpose of having the filtering stage is to estimate the state of the tracks at the desired time step. It should be noted that V2V position measurements are transferred to the vehicle coordinate system before they are fed to the Kalman Filter. Thus,

all the filtering and tracking tasks are achieved in the vehicle coordinate system. The prediction and update stages of the Kalman Filter algorithm are given in Algorithm 3.1.

Algorithm 3.1: Kalman Filter

Prediction:

$$x' = Fx + u$$

$$P' = FPF^T + Q$$

Measurement Update:

$$y = z - Hx'$$

$$S = HP'H^T + R$$

$$K = P'H^T S^{-1}$$

$$x = x' + Ky$$

$$P = (I - KH)P'$$

In the Kalman Filter Algorithm the state vector is:

$$x = \begin{pmatrix} p_x \\ p_y \\ v_x \\ v_y \end{pmatrix} \tag{3.6}$$

where p is position and v is velocity. $u$ is process noise which is assumed to be a zero mean Gaussian process. The state prediction function $F$ for a 2D constant velocity linear motion model is the 4x4 matrix in:

$$\begin{pmatrix} p_x{}' \\ p_y{}' \\ v_x{}' \\ v_y{}' \end{pmatrix} = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ v_x \\ v_y \end{pmatrix} \tag{3.7}$$

The measurement function for camera $H\ is$ the 2x4 matrix in:

$$\begin{pmatrix} p_x \\ p_y \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ v_x \\ v_y \end{pmatrix} \tag{3.8}$$

Q and R represent the covariance matrices for processes noise and measurement noise, respectively. While the measurement process noise for GPS is directly retrieved from Horizontal Dilution of Precision (HDOP) and values, from GPS measurements, measurement noise for camera estimated based on collected data from previous experiment runs.

To accurately track each track from camera and V2V measurements, each track for each sensor is individually tracked with V2V KF and Camera KF (Figure 14) whenever a new measurement is received at the rate of sensor measurements. However, one should be careful when associating measurements from sensors with different measurement time stamps. Associating measurements without synchronization can result in false data association. Therefore, in our design, data synchronization is done between the camera and V2V OBU sensor measurements. The synchronization is realized periodically with a 10 Hz trigger. The synchronization trigger generator block in Figure 14 triggers the camera and V2V Kalman Filters simultaneously at each rising edge of the trigger signal to generate

state estimations for each sensor. For track-to-track data association purposes, the resulting state estimations are used.

## 3.6 Buffering

In the data association task, if the targets are well separated from each other, it is easier to associate the measurements which are originating from the same target. However, if the targets are getting closer to each other with high spatial uncertainty in the measurement, it is hard to differentiate them from each other. Buffering is introduced to enhance track to track data association performance by creating a track history for each sensor measurement and comparing the distance between two tracks over the buffer size. This is especially useful for eliminating false associations due to high spatial uncertainties for short time intervals. An example of where the buffering improves the data association performance is illustrated in Figure 18. In Figure 18, the vehicles detected by the camera are positioned almost on top of each other. The ambiguity in the data association process is eliminated by considering the path history of targets. Thus, the correct data association assessment is performed. The buffering integration into the track-to-track data association will be introduced in next section.

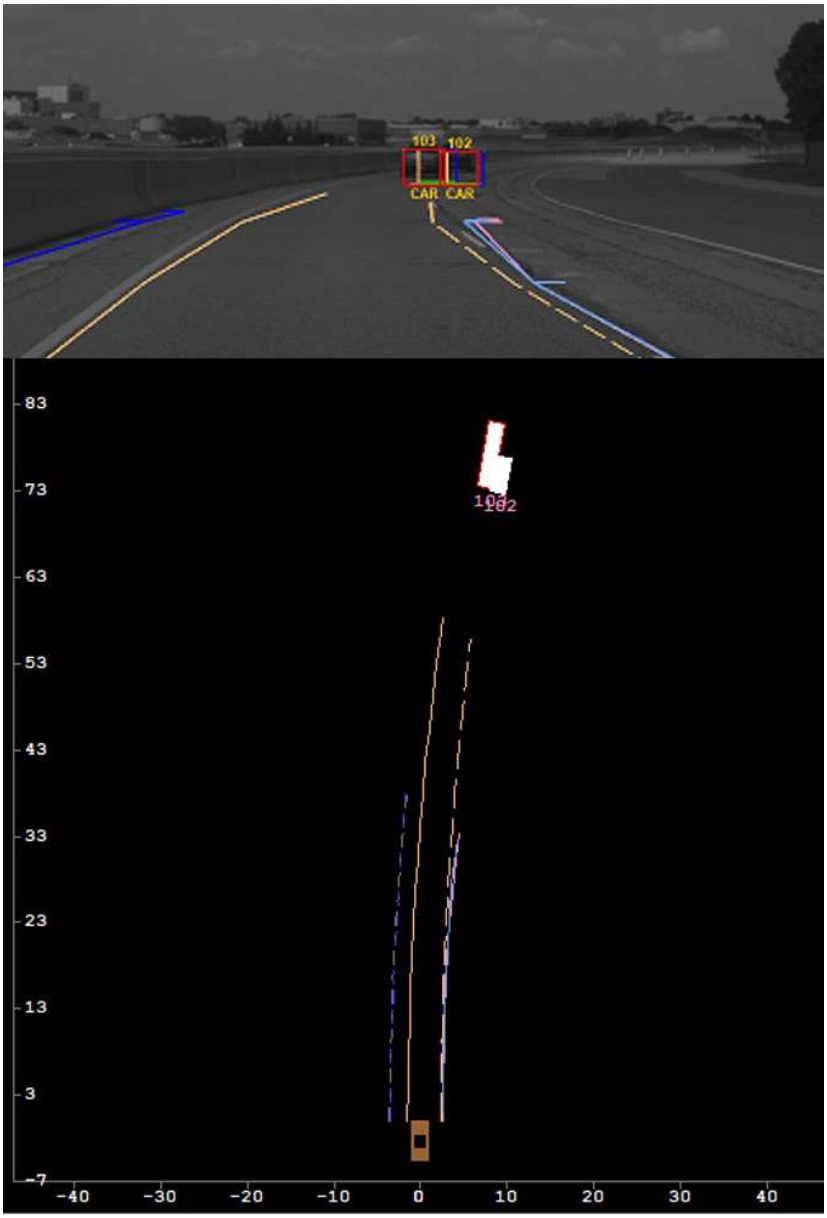

Figure 18 Sample scenario where the camera measurements for remote vehicles have high spatial uncertainty. In this scenario, HV follows two RVs. One of the RV is traveling in front of the other RV on a curved track. HV camera measurement for these vehicles falsely shows these vehicles almost on top of each other [46].

## 3.7 Track to Track Data Association

The implemented Track to Track Data Association module in Figure 14 is designed based on the method proposed in [31]. In this module's development, two sensors are considered S1: camera and S2: V2V modem. Each of these sensors outputs a list of detections $A = \{a_1, a_2, \dots, a_n\}$ and $B = \{b_1, b_2, \dots, b_m\}$. The designed module finds the detections from each sensor for the same target and associates them. Since the comparison of the detections location history would improve the data association performance, detections over time are buffered in the Buffering Module to form tracks. Thus, we can define a track as the state estimate vector of a single target over a time interval, where each element of the vector corresponds to a state estimate at the corresponding measurement time. The algorithm used for this task, which uses position similarity for data association, is presented below.

Algorithm 3.2: Track-to-Track Association

1: Concatenate the tracks of all the sensors.
2: Numerate the concatenated tracks from 1 to N.
3: Create an N x N matrix for the Track to Track Distances (TTTD) between the tracks.
    Set the cells over the diagonal entries to infinity (∞) in order to avoid repeating the calculations.
    Set cells that represent the distance between the same sensor measurements to ∞.
    Set the remaining cells to Mahalanobis Distance between the corresponding tracks.
    Set the cells to ∞, where the distance is larger than a defined threshold.
4: while there is any value other than ∞ do:
    Choose the minimum value in the matrix (row, column).
    if corresponding tracks not in a cluster do:
        Create a new cluster from these two clusters
    else if one of the tracks is in an existing cluster do:
        Add the other track into the existing cluster
    else do:
        Nothing
    Set the selected column and row cells to infinity for the corresponding sensors.
5: if there is a track which is not in any cluster do:
    Form new clusters, for tracks that are not in existing clusters.

In the presented algorithm, the Mahalanobis Distance calculation between two tracks $(a, b)$ is performed over the history size of $n$ to measure the position similarity between two tracks as in (3.9) & (3.10).

$$D_k^{(a,b)} = \frac{1}{n}\sum_{i=0}^{n-1} d_k^{(a,b)} \tag{3.9}$$

where,

$$d_k^{(a,b)} = \sqrt{(X_k^a - X_k^b)^T (P_k^a + P_k^b)^{-1} (X_k^a - X_k^b)} \tag{3.10}$$

Here $X_k$ is the state estimate and the $P_k$ is the covariance matrix of the state estimate for the corresponding tracks at time step $k$. The covariance matrices are acquired from the Kalman Filter state estimation for each sensor.

In Figure 19 and Figure 20, an example of the matrix representation of the Track to Track Association (TTA) Algorithm stages 3 and 4 are shown. In the example shown, the V2V has two measurements, and the camera has four target detections. In these figures, Tracks are named as $T_{ij}$ where $i$ is the sensor ID, and $j$ is the detection number. For example, $T_{12}$ represents the second V2V detection. Similarly, since the camera is the second sensor $T_{23}$ represents the third detection of the camera sensor.

| | T11 | T12 | T21 | T22 | T23 | T24 |
|---|---|---|---|---|---|---|
| T11 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| T12 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| T21 | 4.31 | 20.61 | ∞ | ∞ | ∞ | ∞ |
| T22 | 17.22 | 2.92 | ∞ | ∞ | ∞ | ∞ |
| T23 | 8.97 | 23.60 | ∞ | ∞ | ∞ | ∞ |
| T24 | 11.38 | 25.18 | ∞ | ∞ | ∞ | ∞ |

| | T11 | T12 | T21 | T22 | T23 | T24 |
|---|---|---|---|---|---|---|
| T11 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| T12 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| T21 | 4.31 | ∞ | ∞ | ∞ | ∞ | ∞ |
| T22 | ∞ | 2.92 | ∞ | ∞ | ∞ | ∞ |
| T23 | 8.97 | ∞ | ∞ | ∞ | ∞ | ∞ |
| T24 | 11.38 | ∞ | ∞ | ∞ | ∞ | ∞ |

Figure 19 Stage 3 example for V2V & camera TTA in Algorithm 3.2. In the left matrix, Track to Track Distances (TTTD) is formed. In the right matrix, entries larger than the threshold are set to "∞" [46].

Stage 4: Loop

| | T11 | T12 | T21 | T22 | T23 | T24 |
|---|---|---|---|---|---|---|
| T11 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| T12 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| T21 | 4.31 | ∞ | ∞ | ∞ | ∞ | ∞ |
| T22 | ∞ | **2.92** | ∞ | ∞ | ∞ | ∞ |
| T23 | 8.97 | ∞ | ∞ | ∞ | ∞ | ∞ |
| T24 | 11.38 | ∞ | ∞ | ∞ | ∞ | ∞ |

Clusters:*{T12, T22}*

| | T11 | T12 | T21 | T22 | T23 | T24 |
|---|---|---|---|---|---|---|
| T11 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| T12 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| T21 | **4.31** | ∞ | ∞ | ∞ | ∞ | ∞ |
| T22 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| T23 | 8.97 | ∞ | ∞ | ∞ | ∞ | ∞ |
| T24 | 11.38 | ∞ | ∞ | ∞ | ∞ | ∞ |

Clusters: *{T12, T22},{T11, T21}*

Figure 20 Stage 4 example for V2V & camera TTA in Algorithm 3.2. In the left matrix, by selecting the lowest entry, the first data association cluster is selected. After setting the corresponding column and rows of the selected cluster to "∞", the second cluster is selected [46].

## 3.8 Results

Two different experiments are conducted to demonstrate the effectiveness of the implemented algorithm. These two scenarios look at the data association task for car-following and Intersection Movement Assist (IMA) scenarios. Figure 21 illustrates the travel direction and layout of the vehicles for both scenarios.

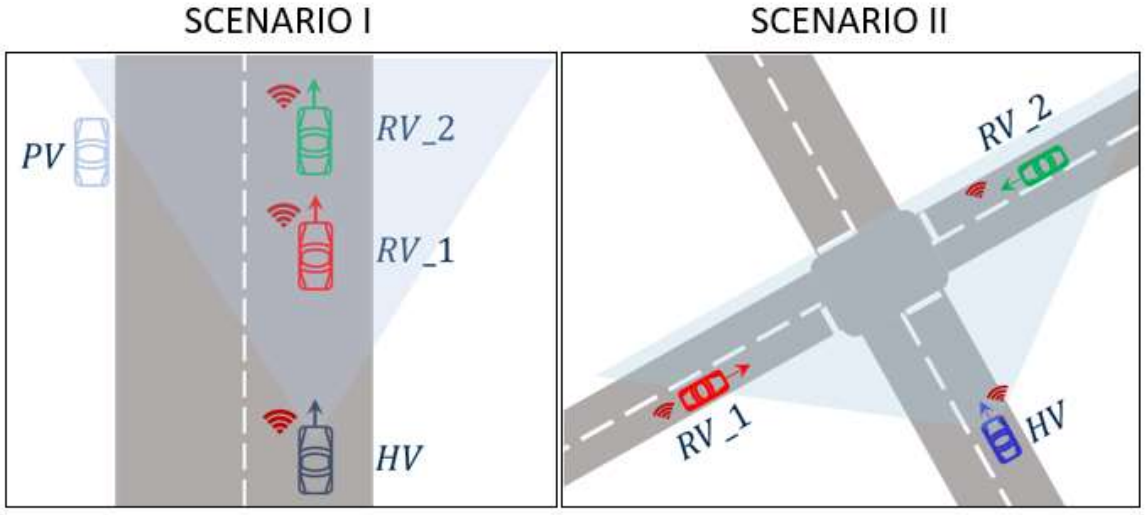


Figure 21 Left: Car following scenario (Scenario I). Two RVs are following each other, and HV follows both of them. Right: Intersection Movement Assist (IMA) scenario (Scenario II). All the test vehicles approach an intersection simultaneously. Host vehicle stops at the intersection while RV_1 and RV_2 pass the intersection. Blue transparent regions represent the camera field of view [46].

### 3.8.1 Scenario I: Car Following

In the first conducted experiment, target vehicles are traveling in front of the host vehicle. Their distance with respect to the host vehicle and with respect to each other changes throughout the experiment Figure 21 (left). In the figure, RV_1 and RV_2 represent the

remote vehicles, PV represents the parked vehicle, and HV represents the host vehicle. Parked vehicles on the track are not transmitting any V2V messages, but the camera can detect them. As for sensors, only the host vehicle has a forward-looking camera capable of detecting the target vehicles and their positions. All three vehicles are equipped with V2V modems. From time to time, remote vehicles travel side by side, but mostly all the vehicles follow each other in the test track shown in Figure 22 , which consists of small curves and straight parts.

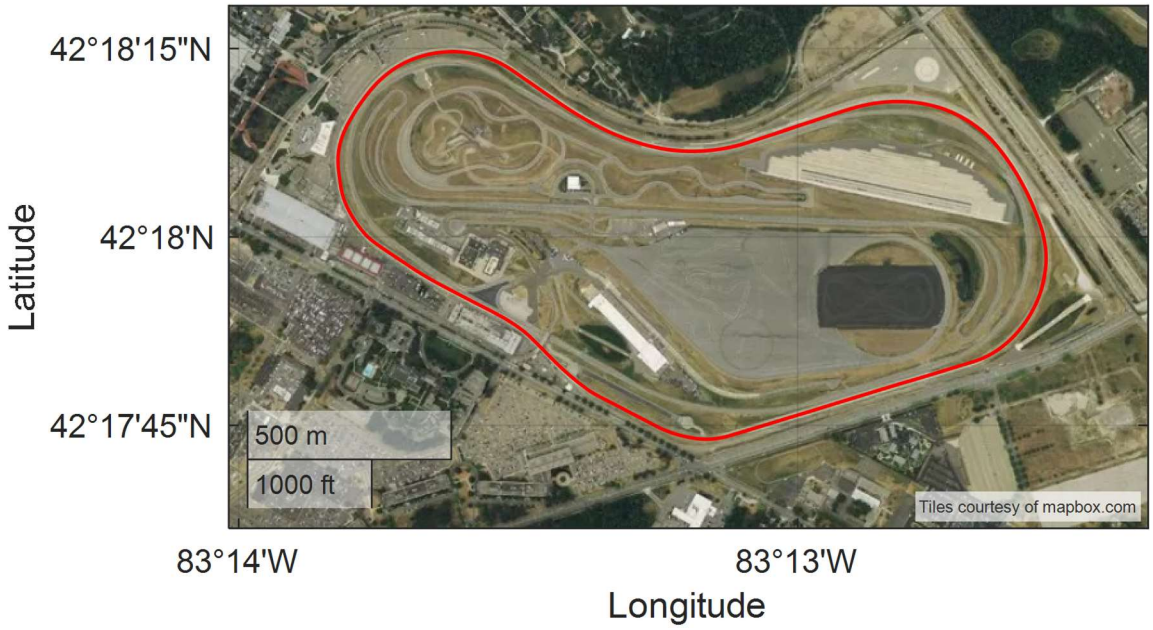


Figure 22 Test track and recorded trajectory of host vehicle for the car-following scenario (Scenario I) as shown in Figure 21. The track consists of straight and curved paths [46].

At each measurement time step, the camera outputs detected objects as a list by identifying each detection with an ID number. Time to time, the camera changes the ID for the same target when an occlusion occurs. V2V BSM messages for remote vehicles are received continuously and they maintain the same IDs throughout the entire experiment.

By considering these characteristics of the received measurements, we created ground truth IDs for camera detections of RV_1 and RV_2. It is created by comparing the camera detection IDs with the recorded video stream. In Figure 23, the upper part represents the RV_1 camera detection IDs, and the middle part represents the RV_2 camera detection IDs over time. The ground truth for the camera IDs corresponding to the remote vehicles is shown with blue color, and the result of the data association is marked with yellow. As can be seen from Figure 23, most of the data associations are performed successfully. Most of the failures occur for RV_2. This is because the RV_2 is generally partially occluded from the camera and the accuracy of the camera distance measurement is affected significantly for the occluded objects.

To assess how closely the detected objects are associated, a data association confidence level is introduced (3.11).

$$Conf = min(0, 100 * \frac{th-D}{th}) \tag{3.11}$$

Where $th$ is the selected inter-vehicle distance threshold, and D is the distance between the two tracks. This equation will yield a 100 percent confidence level if the two measurements are exactly the same. On the other hand, if the measurements are separated from each other, it gets closer to 0 confidence. If the two measurements are separated more than the set $th$ distance, the confidence will be 0, showing that selected measurements are not associated. In Figure 23 it is seen that the confidence level for the remote vehicle RV_1 is between 90-100 percent. On the other hand, for the RV_2, the confidence level varies between 45-100 percent when it is detected. The larger confidence level variation occurs because the RV_2

is mostly occluded by RV_1, HV's camera could not measure the position of RV_2 accurately.

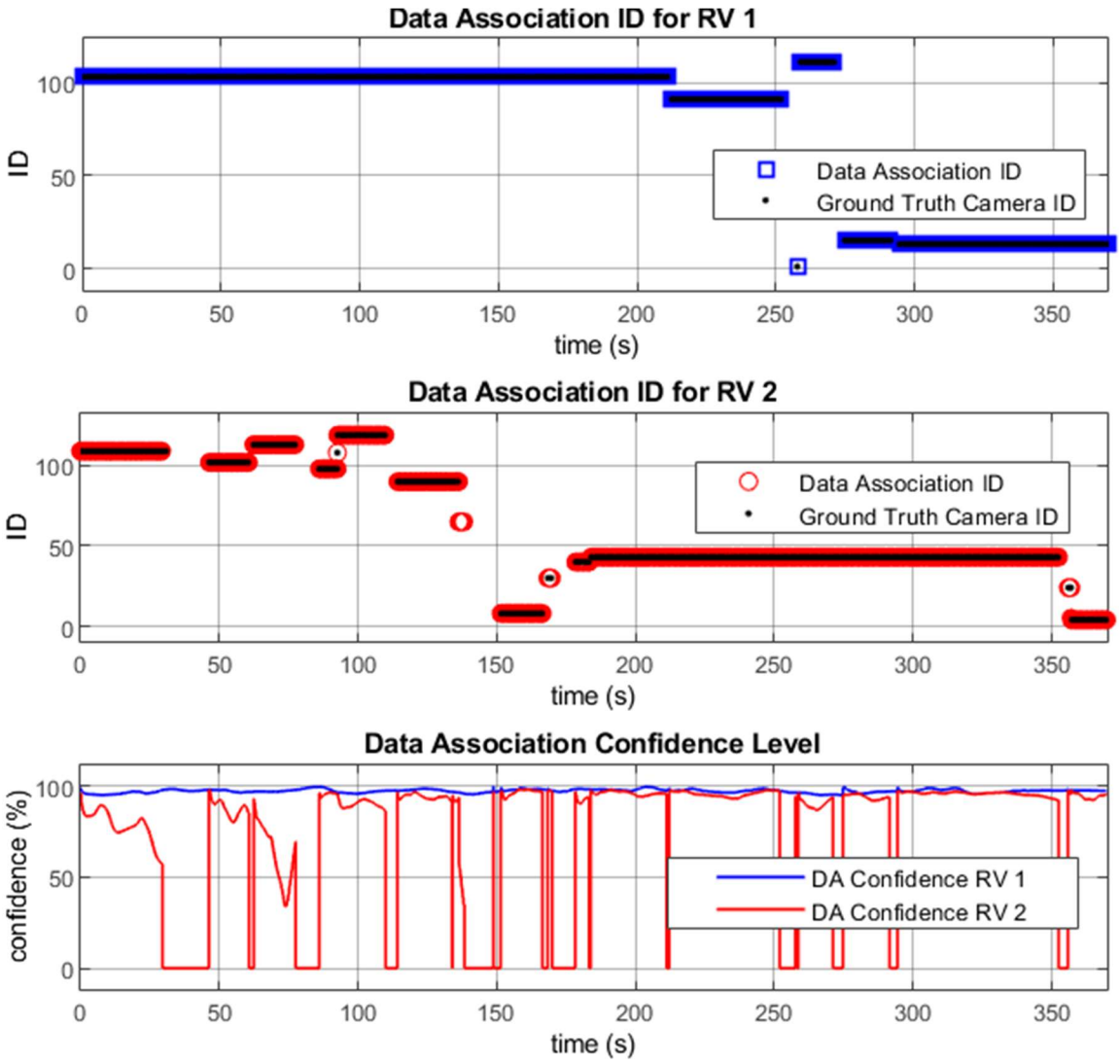


Figure 23 Scenario 1: Track to Track Association V2V - Camera id assignment for car following scenario. The data association algorithm outputs the corresponding camera detection IDs for each RV. Top two images represent the associated camera detection IDs

for RV_1 and RV_2, respectively. The bottom graph shows the data association confidence levels with respect to time [46].

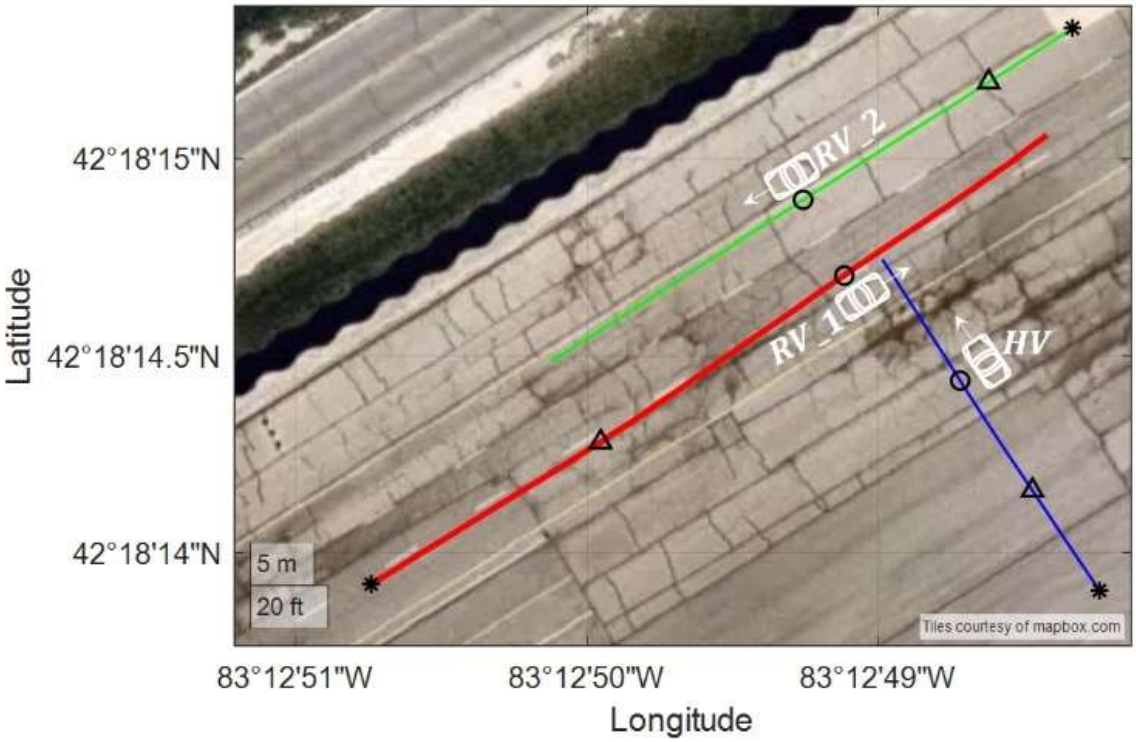


Figure 24 Recorded trajectories of test vehicles for the IMA scenario (Scenario II). Vehicles heading and position representations for occlusion instance are shown with car images at the positions marked with 'o'. '*' represents vehicles' starting points, and 'Δ' represents vehicles' positions at an intermediate time instance [46].

Track Matching Accuracy (TMA) [32] is used as another performance measure for the TTA association. It is defined as the percentage of correct matching decisions taken by the TTA algorithm among all the test cases. In the conducted experiment, camera measurements were able to be associated with the V2V measurements for RV_1 with 100 percent TMA.

However, for the second remote vehicle, TMA drops to 98.8 percent.

### 3.8.2 Scenario II: Intersection Movement Assist (IMA)

In the second scenario, HV, RV_1 and RV_2 experimental vehicles travel towards an intersection from different directions. In the scenario, the host vehicle aims to match V2V detections with the camera detections to issue an appropriate warning by comparing ADAS outputs from V2V and camera-based systems. In V2V ADAS applications, this scenario corresponds to Intersection Movement Assist (IMA) application. As a V2V safety application, IMA warns the host vehicle driver if there is a high collision chance in an intersection [49]. In the experiment, the HV is moving to the intersection from the southbound of the road. The remote vehicles RV_1 and RV_2 are moving to the intersection from westbound and eastbound, respectively. When the remote vehicles come to the intersection, they continue traveling alongside each other on separate lanes. On the other hand, the host vehicle stops at the intersection. The layout of the experiment and the vehicle recorded positions are shown in Figure 21 (right) and Figure 24. In Figure 24, the vehicles' positions in three different time instances are marked on their trajectories to show travel history. Vehicles start from the '*' mark and proceed to 'Δ' and then to 'o'. At the 'o' mark RV_1 starts to block RV_2. The camera frame for this instance can be seen in Figure 25.

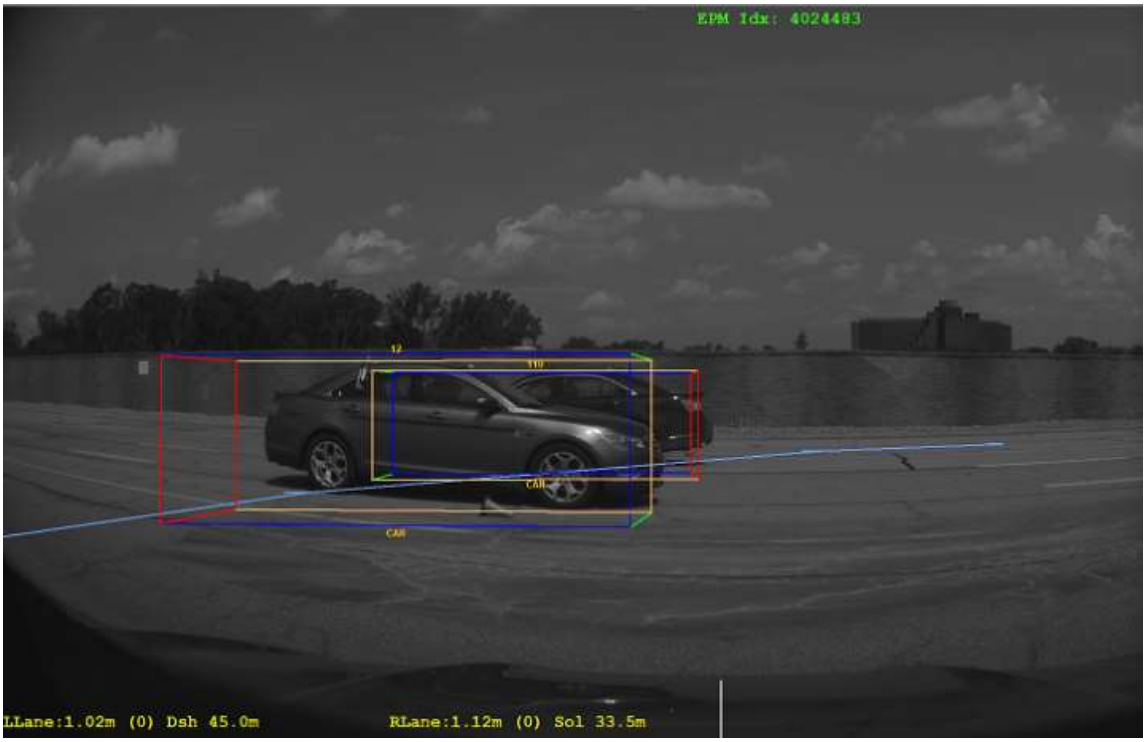

Figure 25 Captured camera frame just before the RV_1 occludes RV_2 in Scenario 2. In this scenario, HV stops at the intersections while RV_1 and RV_2 travel across the intersection in opposite directions [46].

In Figure 26, one can see the V2V camera data association results. While the camera detection ID for the RV_1 is 12 throughout the experiment, RV_2 got two different IDs: 110 and 92. RV_1 blocks RV_2 for a short time interval (Figure 25) when the remote vehicle paths seem to cross each other. After RV_2 becomes visible to the host vehicle, the camera assigns a new ID to RV_2. Data association results in Figure 26 show that the developed method is implemented successfully. All the data association results match with the ground truth for this scenario.

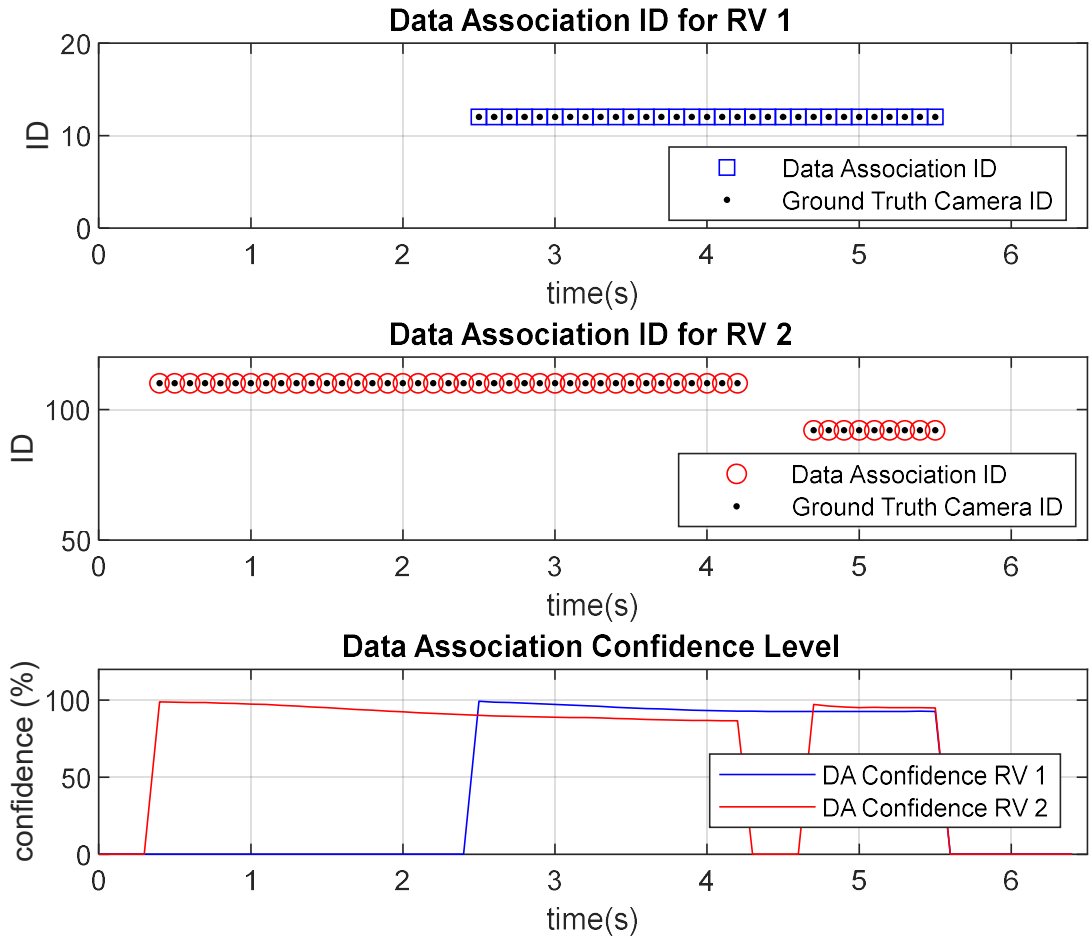


Figure 26 Scenario 2: Track to Track Association V2V - Camera id assignment for IMA experiment. The data association algorithm outputs the corresponding camera detection IDs for each RV. Top two images represent the associated camera detection IDs for RV_1 and RV_2 respectively. The bottom graph shows the data association confidence levels with respect to time[46].

## 3.9 Summary

The designed and implemented data association method aims to identify measurements for the same target from different sensors. Although this method can be generalized and applied to different sensors like radar and lidar, this work focused on the V2V modem and

camera measurements. As a first step, associable parameters for the considered sensors were investigated. It was found out that the location, relative heading, and relative speed measurements from these two sensors are comparable, and they can be used for the data association tasks. Among these parameters, the location measurements were the main parameters used in the implemented algorithm. On the other hand, speed and relative heading parameters were also used as another gating to prevent any false data association assessment. This chapter contributes to the literature by presenting an experimental implementation of V2V and camera data association method. While the presented method was experimentally tested and shown to be effective for even curved roads and intersections, it is required to conduct more experiments with a higher number of vehicles in more complex scenarios before the deployment of the algorithm. As another improvement for the developed method, one can drop the buffering stage for V2V and use the path history information, which is available in the BSM message set.

# Chapter 4. Hardware in the Loop Simulation for Pass at Green Function Validation and Development

Many smart cities and car manufacturers have been investing in Vehicle to Infrastructure (V2I) applications by integrating the Dedicated Short-Range Communication (DSRC) technology to improve the fuel economy, safety, and ride comfort for the end users. For example, Columbus, OH, USA has placed DSRC Road Side Units (RSU) to about 100 traffic lights which publish traffic light Signal Phase and Timing (SPaT) information. With DSRC On Board Unit (OBU) equipped vehicles, people will start benefiting from this technology. In this work, to accelerate the V2I application development for Connected and Autonomous Vehicles (CAV), the use of a Hardware in the Loop (HIL) simulator with DSRC RSU and OBU is presented. The HIL simulator environment is employed to implement, develop and evaluate V2I connected vehicle applications in a fast, safe and cost-effective manner. The simulator allows realistic, real-time evaluation of mobility and fuel economy benefits over simulated actual routes in a safe lab setting before actual deployment in an experimental vehicle. To show the capabilities of the HIL simulator, a Pass at Green (PaG) method, which lowers the idling time at the signalized intersections and improves fuel economy, is simulated.

## 4.1 Background

Transportation has become one of the main contributors of undesired emissions [50], and fuel economy studies have taken a new level with the increase of vehicle autonomy. Research has been conducted on various topics that include green driving [51], clean energy [52], [53], advance traffic control [54], [55] and vehicle control [56].

With the development of V2I communication and control technologies, eco-driving control which attempts to smooth the speed profile of vehicles promptly became a hotspot of research. Optimizing the vehicle speed trajectory to minimize its fuel consumption can significantly enhance the vehicle fuel efficiency. For example, it is shown in this chapter that up to 7% fuel efficiency improvement could be achieved for the multiple traffic light scenario under ideal conditions. This result might vary for different traffic and road conditions. Techniques used for this purpose are called Eco-Driving and this optimization tool predicts the future constraints that the vehicle will be subject to and generates a speed profile that is fuel-optimal. This prediction of future constraints was impossible until vehicle connectivity was introduced. In addition to the safety benefits, this technology promises valuable information to the vehicles and traffic controllers such as speed-acceleration-brake status and signal phasing and timing (SPaT) information. This information can be leveraged to develop spatial and temporal constraints to optimize the vehicle trajectory to achieve maximum fuel efficiency.

Many eco-driving methods have been proposed in the literature. Saboohi developed an eco-driving strategy, which can reduce the fuel consumption by optimizing the speed and gear ratio of the vehicles [57]. Mandava proposed an arterial velocity planning algorithm which could minimize the acceleration/deceleration rates to reduce the fuel consumption and emissions of an individual vehicle [58]. Kamal used Model Predictive Control (MPC) to improve eco driving performance in varying traffic environment [59]. Rakha designed an eco-driving framework that has detailed microscopic fuel consumption and emission

models in the objective function [60]. Mensing utilized the dynamic programming approach to optimize the speed trajectories of vehicles [61].

This chapter focuses on extending our previous HIL simulation environment [62] with an actual Traffic Light Controller, DSRC Road Side Unit and DSRC On Board Unit and a fast prototyping controller to embed fuel efficient eco-driving algorithms. Created simulation environment runs the virtual road and car model in the HIL hardware while emulating the actual SPaT communication with the traffic light controller. Thus, one can employ the HIL simulator to evaluate the achieved fuel efficiency benefit with the developed algorithms in a safe environment before actual deployment. As an example, an application Pass at Green algorithm, which minimizes idling time at the traffic light will be simulated.

## 4.2 Pass at Green Algorithm

### 4.2.1 Overview

In the Pass at Green (PaG) application, the algorithm uses the Signal Phasing and Timing (SPaT) information from an upcoming traffic light and modifies the vehicle speed, so that the vehicle can pass at the green light phase and does not have to stop and wait for the traffic light to turn green. The PaG algorithm used in this section is based on the work reported in [63] and [64]

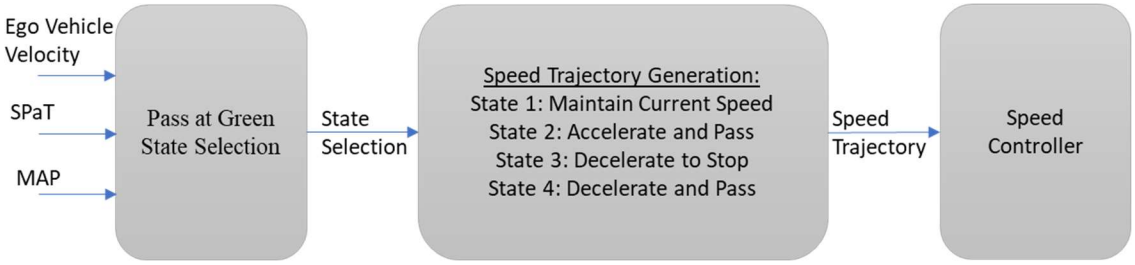


Figure 27: Structure of Pass at Green algorithm (adapted from [63])

As seen in Figure 27, when the vehicle is in the range of the DSRC range of the RSU, the Pass at Green State Selection block receives the available SPaT information and the position of the traffic light using OBU. For this simulation environment only SPaT is broadcasted through RSU. The position of the traffic light is simulated in the HIL. Based on the measured speed of the vehicle and these received signals, the State Selection block chooses the most fuel efficient and applicable state for passing the intersection. Based on the selected state, the Speed Trajectory Generation block generates the desired velocity trajectory for the selected state. The trajectory generation provides a smooth velocity profile between the current velocity and the recommended velocity for the vehicle. The trajectories are generated based on piecewise trigonometric functions that ensure drivability and fuel economy [63]. Generated velocity trajectory is sent to the vehicle speed controller to follow this desired speed. Once the vehicle crosses the traffic light in green, the desired speed generated by the Pass at Green algorithm is discarded to give the control back to the autonomous vehicle speed generator to reach to the original desired speed. In this work, an Intelligent Driver Model (IDM) is used to control the vehicle in case there is no upcoming traffic light. The driver selection behavior for the Pass at Green algorithm implementation is summarized in Equation 4.1.

$$Driver = \begin{cases} PaG, & when\ traffic\ light\ is\ in\ range \\ IDM, & else \end{cases} \quad (4.1)$$

### 4.2.2 Definition of States

There are four main states for managing the longitudinal behavior of an autonomous vehicle at signalized intersections in the PaG model. By employing the information coming from the DSRC systems installed on the vehicle and traffic lights, it is possible to manage

how to approach a signalized intersection in a fuel-efficient manner. These four states can be listed as follows:

**State 1:** Vehicle maintains its current speed and catches the Green phase of the traffic light to pass.

**State 2:** Vehicle accelerates to a set speed and catches the Green phase of the traffic light to pass.

**State 3:** Vehicle decelerates to a stop, stops at the traffic light at Red, then passes when the traffic light turns Green.

**State 4:** Vehicle decelerates to a set speed and catches the Green phase of the traffic light to pass.

4.2.3 State Selection

As stated earlier, State Selection is done based on the SPaT information from the traffic light, traffic light location and the velocity of the vehicle at the instant it is in the DSRC range of the RSU. There are 2 different outcome sets based on the status of the traffic light. For the 1st case, if the traffic light phase is already green when the vehicle enters the communication range of the RSU, then PaG has the necessary information of when the traffic light will turn red. Using this information, the vehicle speed can be modified to make the vehicle pass the traffic light before the light turns red by choosing either State 1 or State 2. If the vehicle speed cannot be modified to pass at the current green light, then State 3 is selected, so that the vehicle decelerates and comes to a stop smoothly, waits for the traffic light to turn green, then accelerates smoothly to the reference speed.

For the 2nd case, if the traffic light phase is already red when the vehicle enters the communication range of the traffic light, then it has information about when the red light will end, as well as when the next green light phase will begin. Depending on how much the vehicle has to accelerate or decelerate to catch the next green light, the State Selection is carried out and State 1, State 3 or State 4 is chosen, by taking the acceleration/deceleration and jerk limits, as well as the speed limit into account. If the traffic light state phase changes during one state, the state can be updated with the new available SPaT information for a more fuel-efficient state, if desired.

## 4.3 HIL Simulator

### 4.3.1 HIL Simulator Setup

Hardware in the Loop Simulator is a tool to simulate the vehicle dynamics and interaction of the vehicle with other traffic and infrastructure in real-time. Since the HIL simulator runs in real time and has CAN and Ethernet interfaces, it is possible to employ actual hardware and software components into the simulation. Thus, the simulations done in the HIL simulator provide a realistic environment to test main functionalities of the designed system. A well-prepared HIL simulation is not only a time saver and cost effective, but also safer than doing in-vehicle experiments. After the HIL simulations, one can conduct experiments in a more reliable environment.

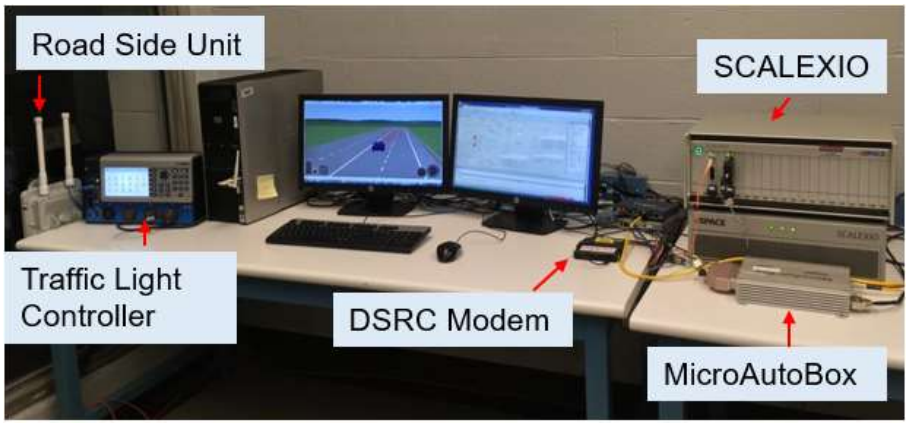


Figure 28: Connected and Autonomous Vehicle HIL simulator setup [65]

HIL simulator used (Figure 28) in this work consists of five main parts. The first part, dSPACE Scalexio HIL computer, runs the designed vehicle dynamics and environment simulation in real time. Second one is the user interface of the simulator. The computer used for user interface has both CarSim and dSPACE Control Desk software, and both are used for monitoring the chosen parameters via displays and plots. Additionally, the vehicle dynamic modelprogram CarSim can also show a real-time video of the simulation. One of the main parts of the simulator is the dSPACE MicroAutoBox controller. Since the MicroAutoBox is also used as a controller in the vehicle, developed control software can be used in the actual vehicle with minor modifications. The Traffic Light Controller in the HIL setup determines how long each Red, Green and Yellow phase is going to last, depending on the traffic density and time of the day. The SPaT messages generated by the Traffic Light Controller are sent to the Road Side Unit (RSU). The RSU can transmit the SPaT messages to vehicles equipped with DSRC OBU. In this HIL Simulator, the messages transmitted by the RSU are received by the DSRC modem to be used in the Pass at Green

model. One can see the data exchange and communication protocol structure between each part of the simulator in  Figure 29.

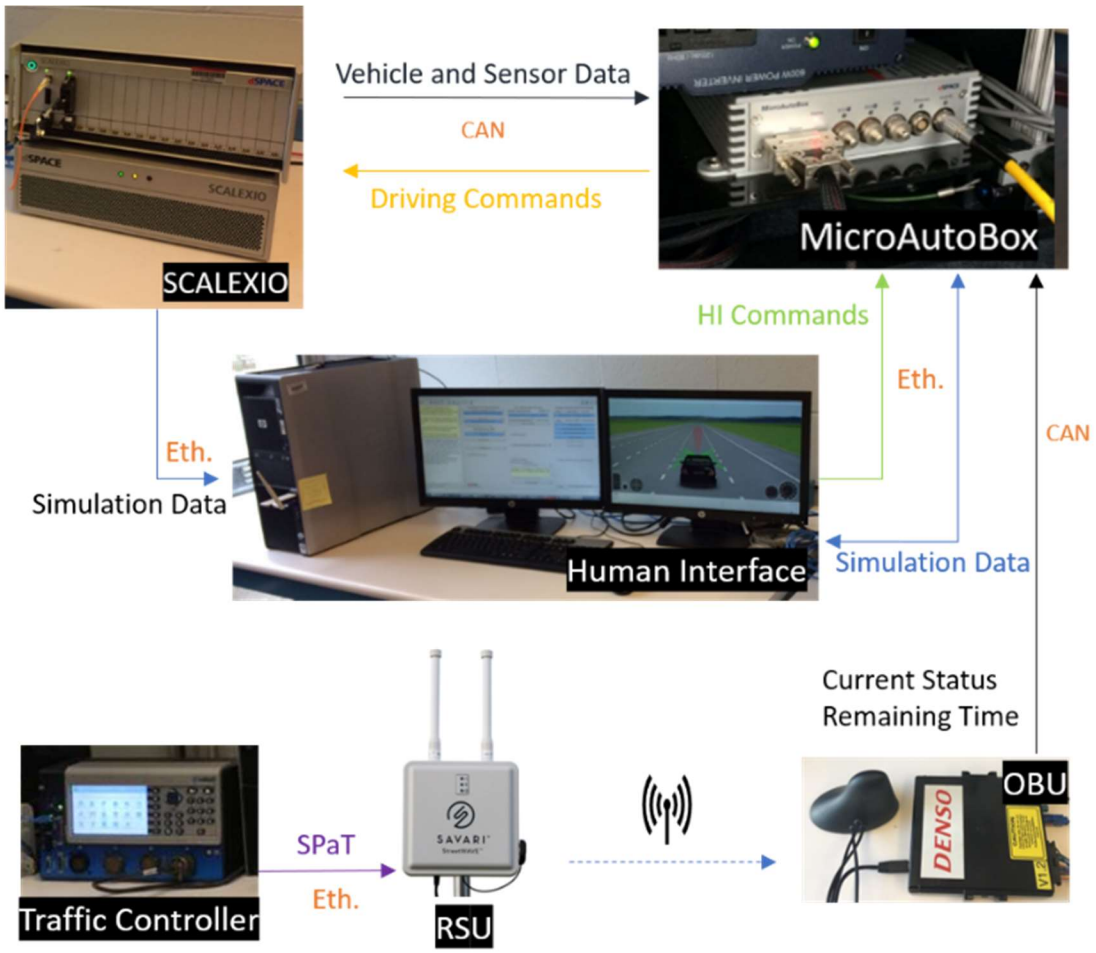


Figure 29: HIL Simulation data flow [65]

The HIL simulation block diagram is seen in Figure 30. In a real setup, the vehicle and traffic lights would be able to communicate through DSRC radios. For this simulation setup, two different drivers are used to simulate the driving behaviors of the vehicles as Intelligent Driver Model (IDM) and Pass at Green drivers. Finally, CarSim vehicle is driven by a Proportional Integral Derivative (PID) controller to follow recommended speed

profile provided by IDM or Pass at Green to evaluate the fuel consumption. The fuel consumption of the vehicle is calculated internally inside the chosen generic D-Class sedan vehicle model in CarSim.

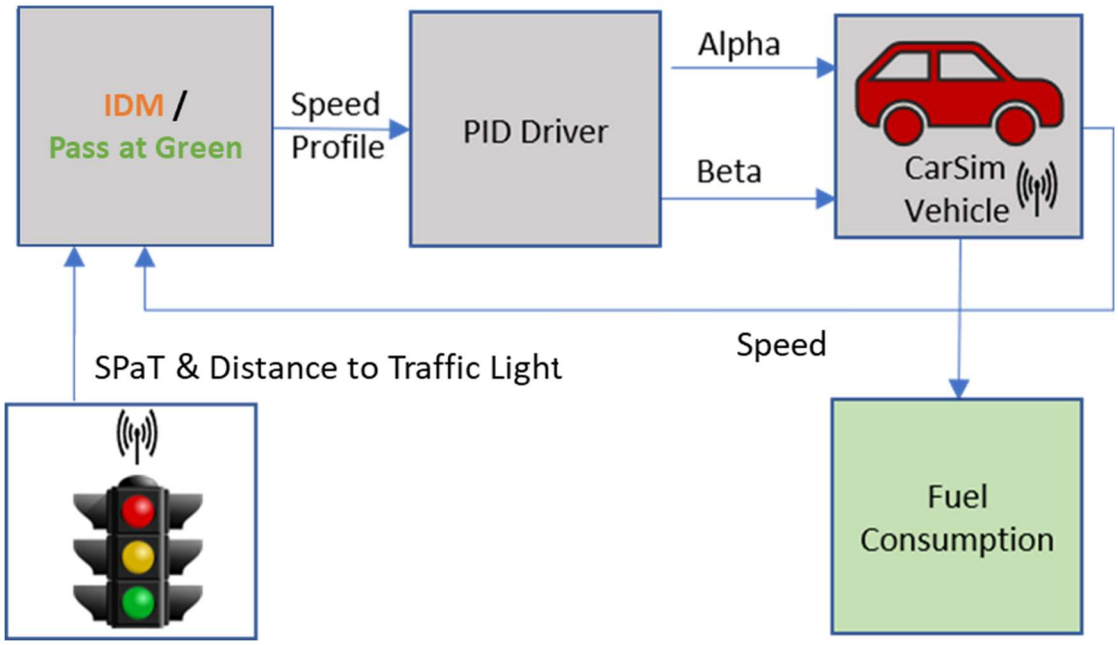


Figure 30: Simulation block diagram [65]

### 4.3.2 Infrastructure Creation in HIL Simulator

To simulate the behavior of vehicles in a simulation environment as accurately as possible, the roads that actual vehicles are tested should be generated in the simulation environment. To do that, route information needs to be acquired from a map with the global positioning system (GPS) points and elevation information. OpenStreetMap is a source to get maps online, and the maps can be downloaded and processed [66]. OpenStreetMap provides the longitude and latitude information of the roads, as well as speed limits and traffic lights and STOP signs.

It is possible to improve the fuel economy of a vehicle by previewing the road slope and modifying the speed of the vehicle accordingly. In this chapter, the simulation environment used is CarSim, a simulation tool to simulate the performance of passenger vehicles and light-duty trucks. One of the routes that were generated in the software environment can be seen in Figure 31 (marked with a sloid black line for the route). This route was put into OpenStreetMap to get the portion of the map that is of interest.

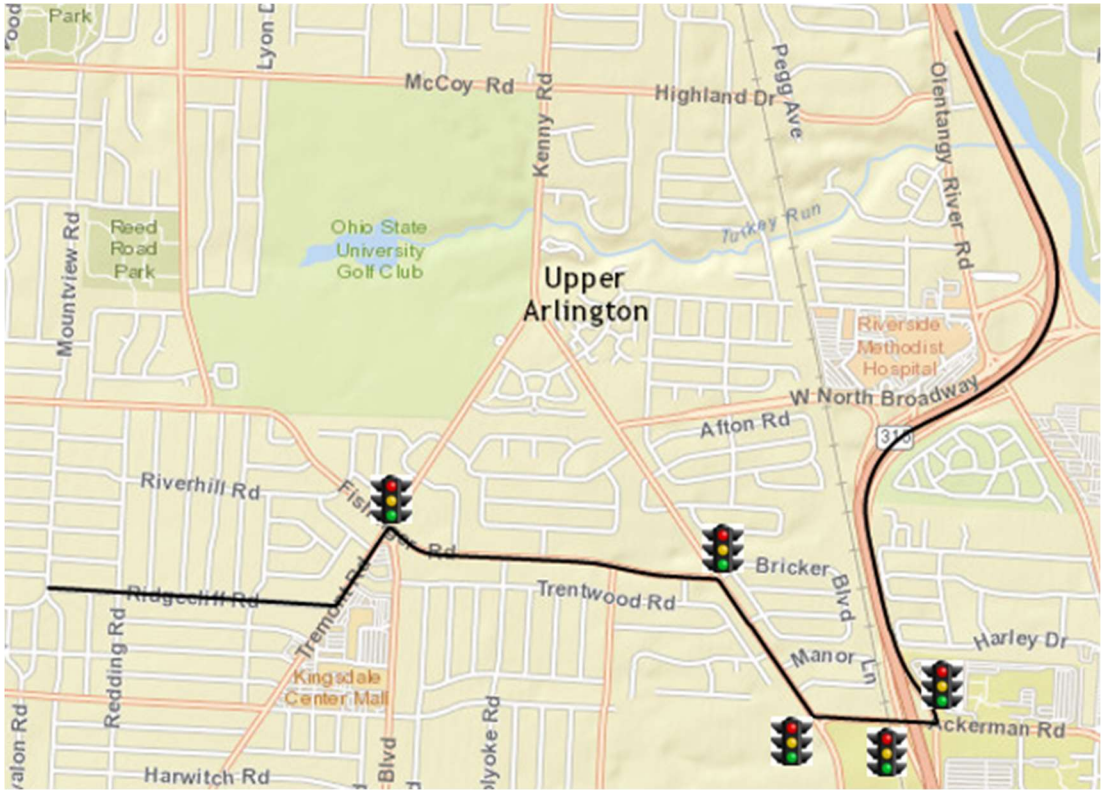


Figure 31: Test route on the map with traffic lights [65]

A real-life traffic simulation was run on microscopic traffic simulator SUMO [67] with different traffic density states; no traffic, low-density traffic, medium-density traffic and high-density traffic. Using SUMO, another set of road GPS information, as well as locations of the traffic lights and STOP signs were acquired.

Additionally, the speed limit information for the road was acquired from SUMO. The maximum speed limit for the route vs cumulative distance travelled by the vehicle can be seen below. There are 5 traffic lights on the road, and the speed limit ranges from 11.2 m/s (~25 mph) to 29 m/s (65 mph) (Figure 32). The route consists of roads in urban areas, as well as part of a highway.

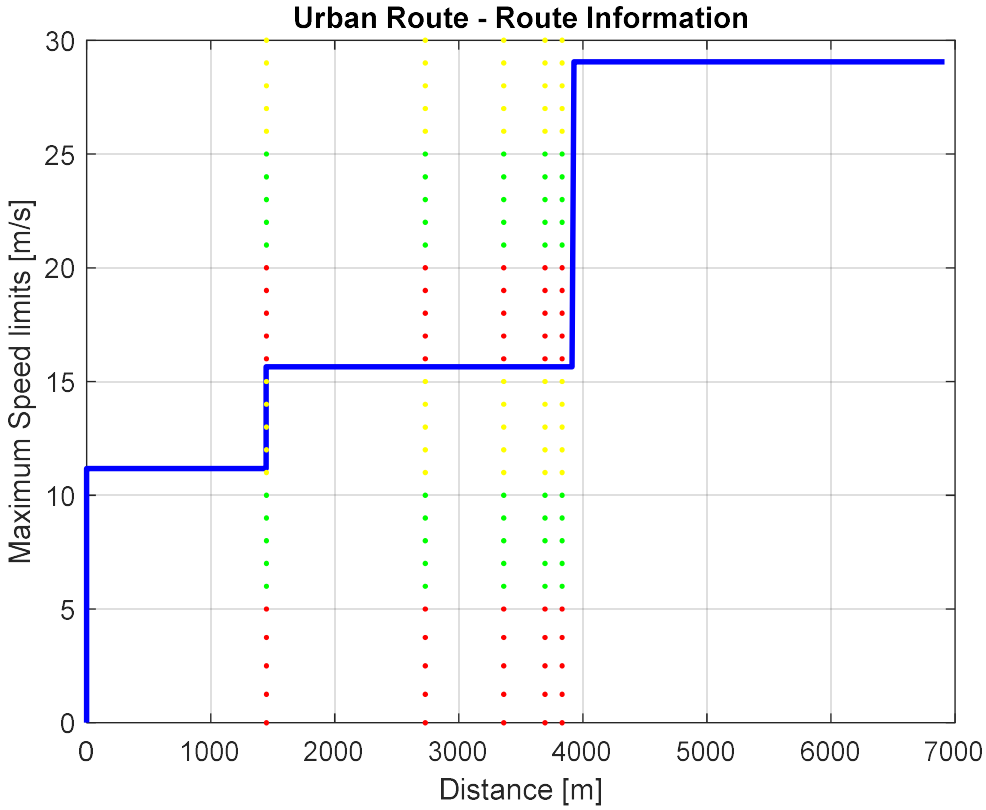


Figure 32: Speed limit vs Distance [65].

For the CarSim part, after getting the necessary GPS waypoints and elevation information for the route, they were put into the simulator environment in CarSim to design the road. Afterwards, appropriate road shapes and lane lines were added to the road, as well as the

traffic lights. Then route was visualized using the preview tool in the CarSim environment (Figure 33).

Figure 33: Visualization of traffic light approaching state in CarSim [65].

## 4.4 Simulation Methods and Results

In the HIL simulation environment, selected urban-highway mixed route is created by importing all the relevant features of the road. To evaluate the effect of the Pass at Green algorithm for the chosen route, two sets of simulations are conducted. In the first case, the IDM driver is used in the simulation to drive the vehicle to simulate a regular driver. In the second case, V2I communication is introduced into the system. When the vehicle is in the communication range of the traffic lights, speed profile is acquired from the Pass at Green algorithm to make sure that the vehicle passes the traffic light with a more fuel-efficient speed profile. Additionally, the Pass at Green algorithm tries to prevent the vehicle from making unnecessary braking maneuvers and idling at traffic lights. Comparison of these

two cases for different states of the Pass at Green algorithm can be seen in Figure 34 - Figure 36.

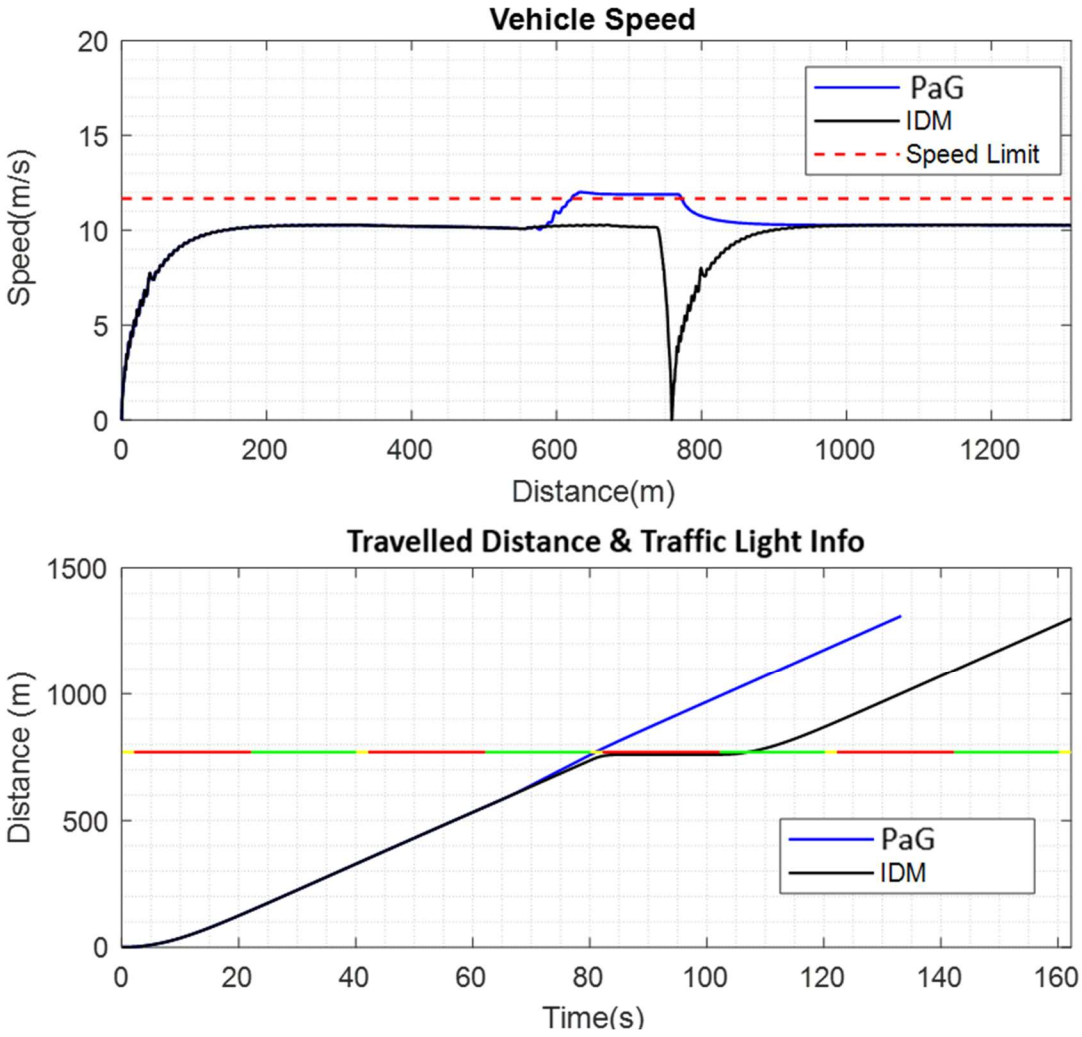


Figure 34: State 2 speed profile [65]

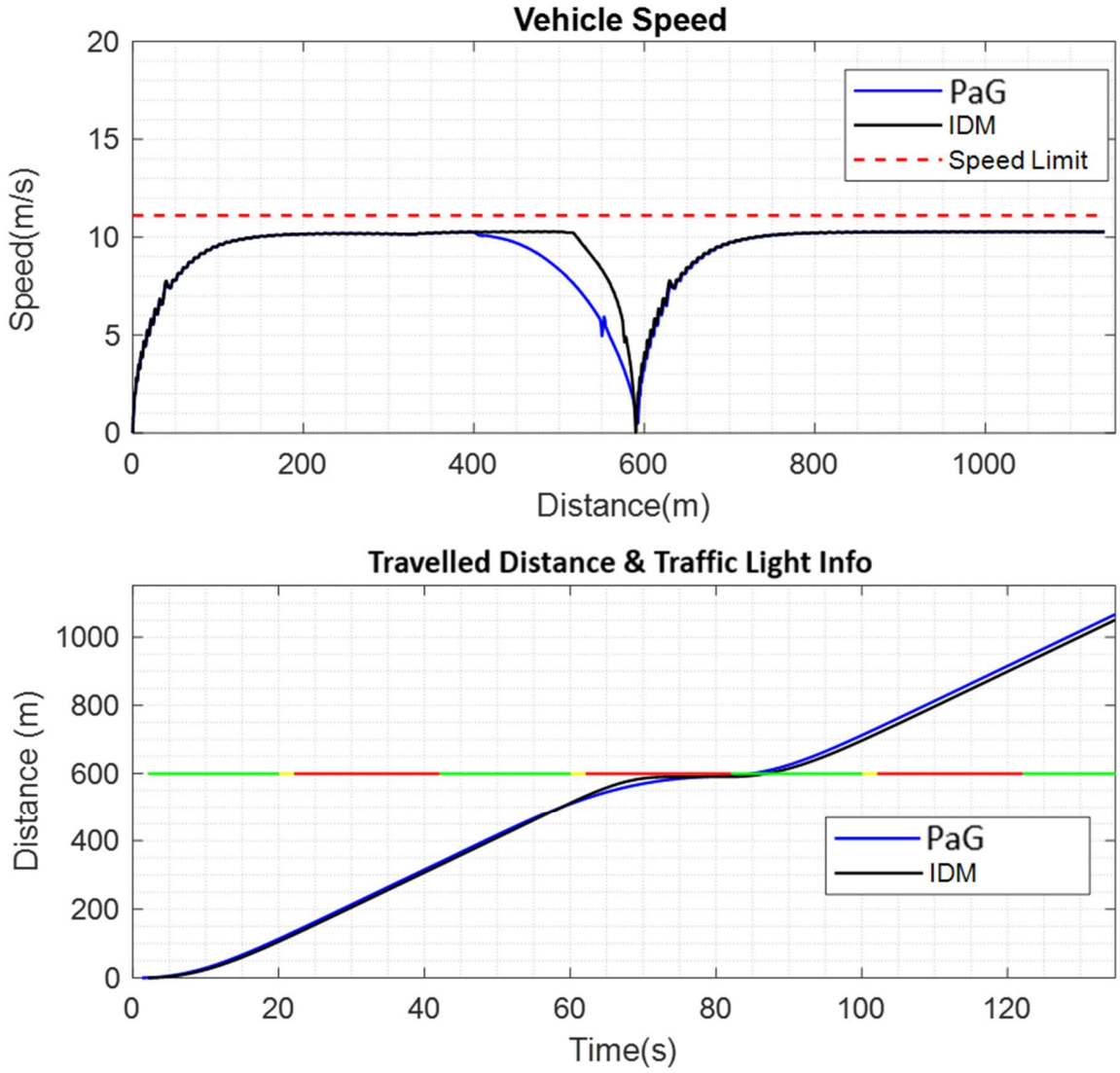


Figure 35: State 3 speed profile [65]

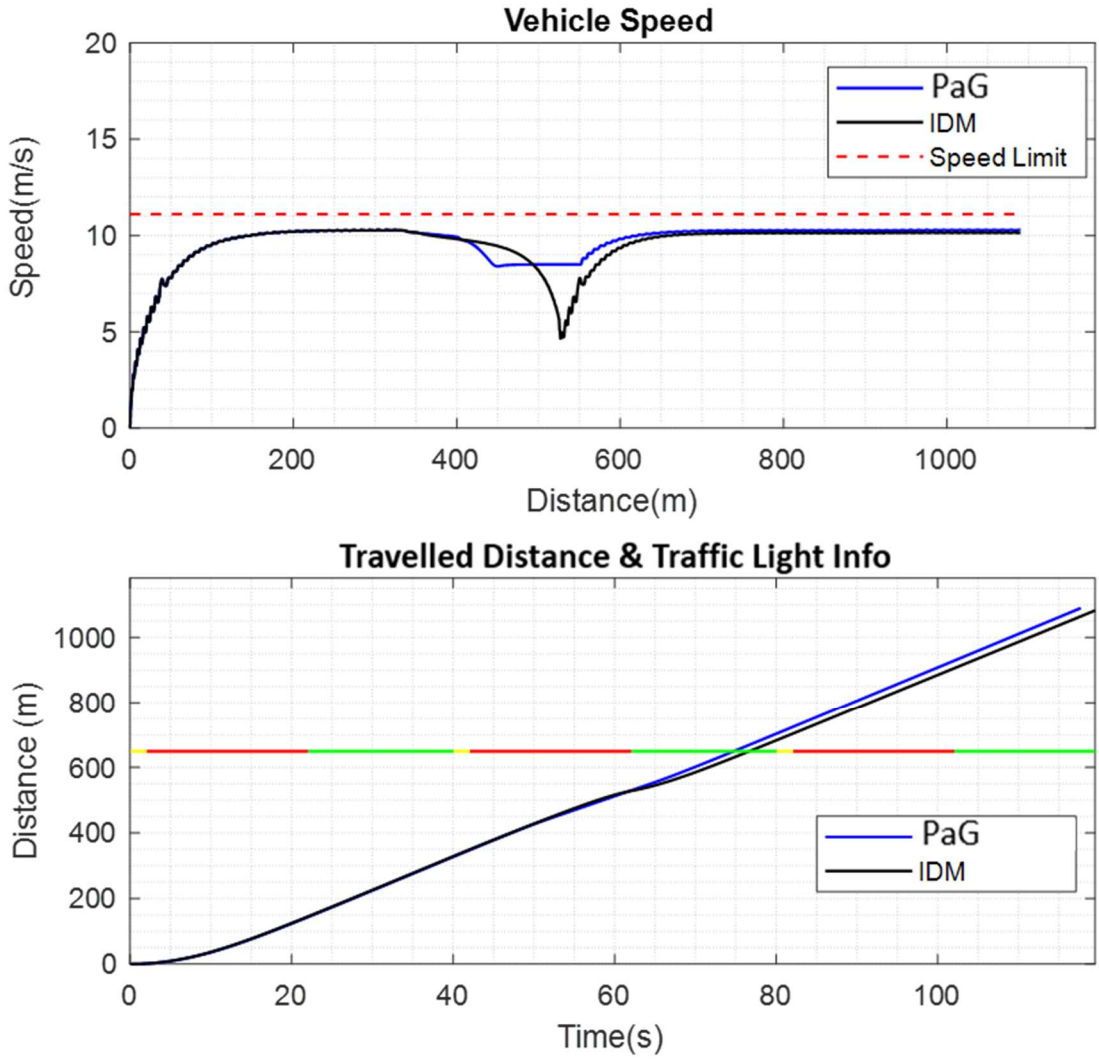


Figure 36: State 4 speed profile [65]

The Intelligent Driver Model (IDM) [68] is a deterministic car following model. The IDM can generate speed profiles that replicate human driving behavior in car following or speed following scenarios. In this chapter, IDM has been modified to have the added capability of responding to traffic light phases, as well. In case of red traffic light, a stationary vehicle where the traffic light originally positioned is virtually inserted to the simulation as a target

vehicle, so that the host vehicle stops at red. When the traffic light turns green, this virtual target is removed from the simulation, so that the host vehicle does not try to stop. Thus, the IDM driver can obey the traffic light status like a real human driver would.

As mentioned in the Pass at Green algorithm section, there are 4 states in the Pass at Green algorithm. Each state was simulated with a single traffic light to show fuel economy benefit and speed profile set by the Pass at Green algorithm.

State 1 (the cruising case), where the vehicle maintains its speed at the time the algorithm chooses which state, the speed set by the IDM is also equal to the speed recommended by the Pass at Green algorithm. Since the speed at which the vehicle passes the traffic light is the same for both Pass at Green and IDM, there would not be any fuel economy comparison between the two models. Therefore, State 1 was not simulated.

State 2 (accelerate and pass case) was simulated (Figure 34), where the vehicle accelerates to a higher speed while staying within the acceleration and jerk limits of the Pass at Green algorithm, as well as attaining a speed lower than the speed limit of the route. For the Pass at Green algorithm, when the vehicle is within the DSRC range of the RSU, the host vehicle accelerates to a higher speed than its current speed. After accelerating and passing the traffic light, the vehicle decelerates to its original speed, the speed at the instant when the State 2 has been activated. The IDM speed for this case was to follow a reference speed of 11.2 m/s (25 mph) before reaching the traffic light, come to a stop at the red light, then wait for the next green light, then finally, pass the traffic light. The fuel economy benefit for the vehicle when it follows the Pass at Green strategy rather than the IDM was 42.72%.

Another advantage that Pass at Green had over the IDM model for this state was the total travel time. As seen in the 2nd plot in Figure 34, the Pass at Green was able to utilize the SPaT information of the traffic light to accelerate and pass, whereas IDM did not have this information and had to make the vehicle stop at the upcoming red light. Therefore, Pass at Green algorithm also helped with total travel time by decreasing it significantly as compared to the IDM model.

State 3, decelerate and stop, then pass case, was simulated (Figure 35). For this case, when the vehicle was in the range of the DSRC modem, the speed of the vehicle was set to drop to zero smoothly, whereas the IDM for this case decelerated with a sharper slope. The fuel economy benefits due to Pass at Green algorithm compared to the vehicle travelling with IDM speed was 2.56% for this state.

State 4, decelerate and pass case, was simulated (Figure 36), where the vehicle decelerates to a lower speed, while staying within the deceleration and jerk limits. The IDM speed for this case was to follow the reference speed of 11.2 m/s (25 mph) until the vehicle arrives at the DSRC range of the RSU. The IDM model noticed that the traffic light phase was red way later than Pass at Green had, thus IDM tried to decelerate rapidly. When the traffic light turned green, IDM accelerated just as rapidly to reach the reference speed. Pass at Green, on the other hand, utilized the SPaT information of the upcoming traffic light to decelerate smoothly knowing that the light phase was going to turn green soon after the red phase, maintained the low speed, and then accelerated after passing the traffic light at green. With the Pass at Green algorithm, 3.78% fuel economy benefit was achieved as compared to the vehicle travelling with IDM speed.

In Table 2: Simulated fuel economy values for IDM and Pass at Green., how the fuel economy changes depending on the DSRC range of the modem is summarized. Depending on the DSRC range value, the fuel economy ranges between 2.56% and 47.72%.

| PaG State | IDM Reference | | Pass at Green (PaG) | | FE improv. (%) |
|---|---|---|---|---|---|
| | Total Fuel (gal.) | FE (mpg) | Total Fuel (gal.) | FE (mpg) | |
| 2 | 0.0305 | 26.73 | 0.0213 | 38.15 | 42.72 |
| 3 | 0.0263 | 26.97 | 0.0256 | 27.66 | 2.56 |
| 4 | 0.0239 | 28.3 | 0.0231 | 29.37 | 3.78 |

Table 2: Simulated fuel economy values for IDM and Pass at Green.

To simulate a more realistic case with multiple traffic lights, the simulation route shown in Figure 31 is modelled in the HIL simulator shown in Figure 28. Two simulation scenarios are considered. First the simulation vehicle driven by the IDM (Figure 37). In this case the IDM will try to reach the desired speed which is set to 3 km/h lower than the speed limit for the simulated route section. The modified IDM model will also let the vehicle obey traffic lights when they change state. For the first traffic light, the driver stops for red. For the second traffic light, the driver detects yellow traffic light state and brakes to decelerate but since it is too late to stop, it passes at yellow phase. This driver behavior can be considered as dangerous. Similarly, for the rest of the traffic lights, it either stops or decelerates until the traffic light phase turns to green. This leads to unnecessary deceleration and fuel consumption.

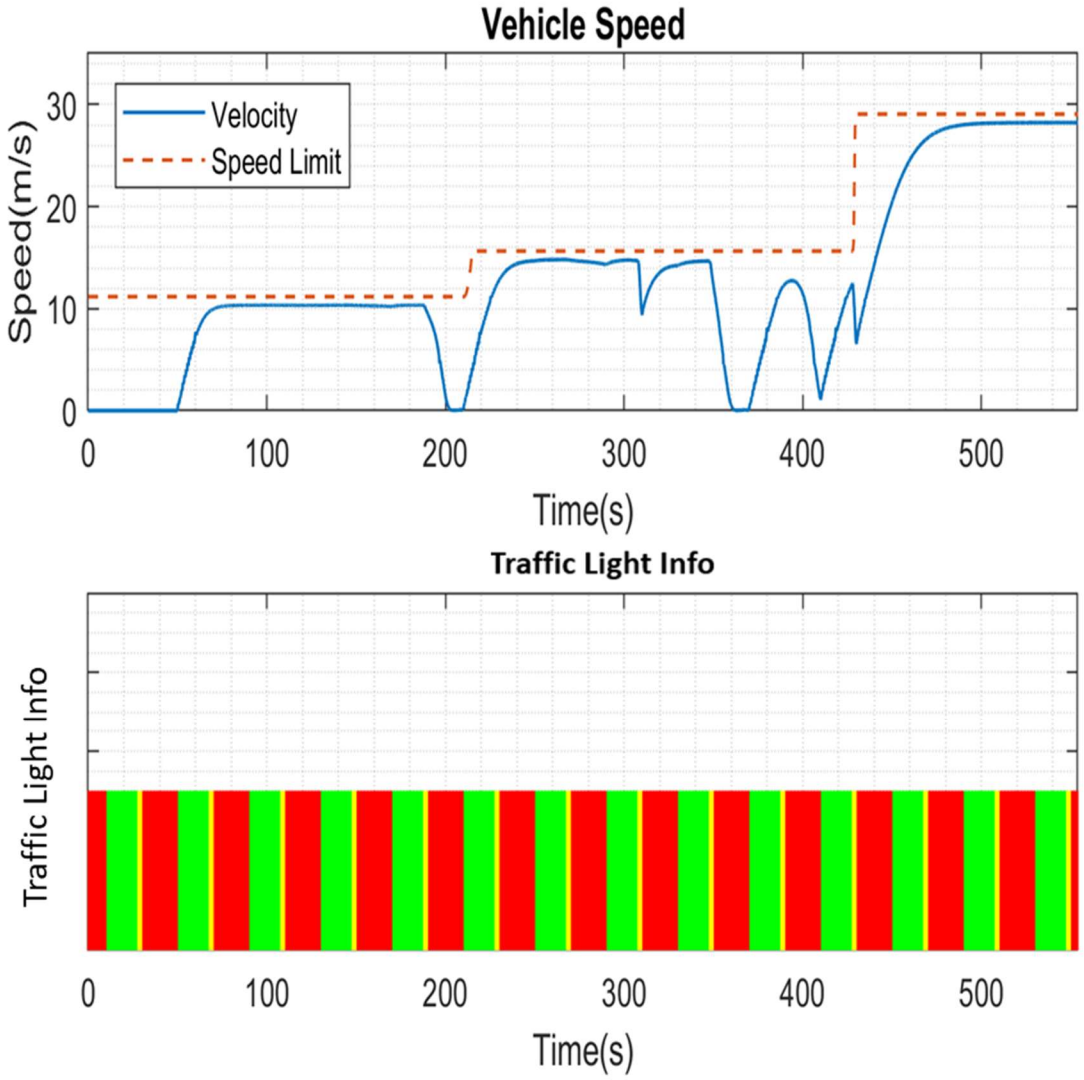


Figure 37: Simulation of multiple traffic lights with different speed limit conditions and IDM driver [65]

Similarly, in the second simulation scenario, vehicle is driven in the same route but this time the vehicle communicates with the RSU connected to the traffic light controller using DSRC modem. Thus, the informed autonomous vehicle can manage its speed in a more fuel efficient and safe manner. Simulation results for the Pass at Green driver can be seen

in Figure 38. As it can be seen from the figure, the vehicle safely stops for the first two traffic lights because it is not able to catch the green light in the green light interval without violating the speed limits. For the 3rd traffic light, it accelerates and passes at green instead of stopping at red. Similarly, for the 4th traffic light, the vehicle decelerates when the light is red to wait for it to turn green and pass at green by slowing down. In the same case, a human driver would slow down more to stop at the red light because the driver would see the red light. Finally, it passes at green by maintaining its speed at the last traffic light.

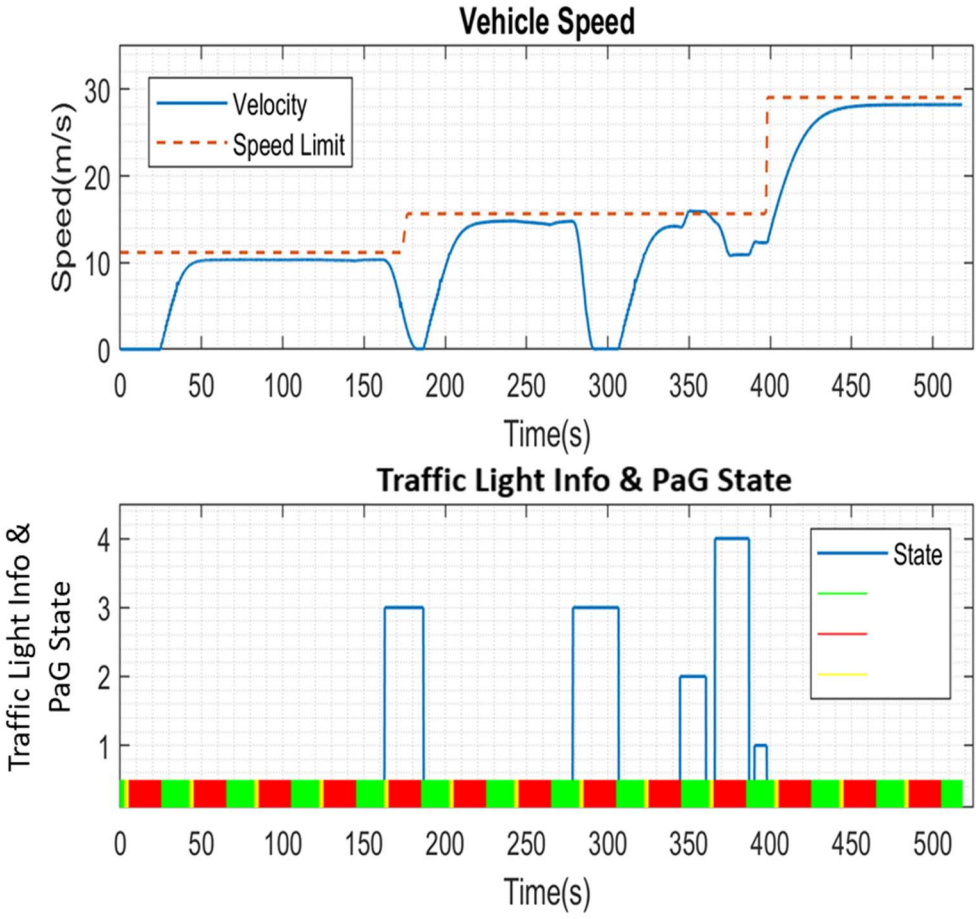


Figure 38: Simulation of multiple traffic lights with different speed limit conditions and Pass at Green model [65]

In Figure 39, the Pass at Green and IDM speeds with respect to distance, as well as the total distance travelled by the vehicle with respect to time can be seen. In the 2nd plot in Figure 39,it is seen that the total trip time for Pass at Green was 490 seconds, whereas the total trip time for IDM was 499 seconds. Therefore, it can be concluded that Pass at Green decreases the total trip time, as well helping with fuel economy and drive comfort.

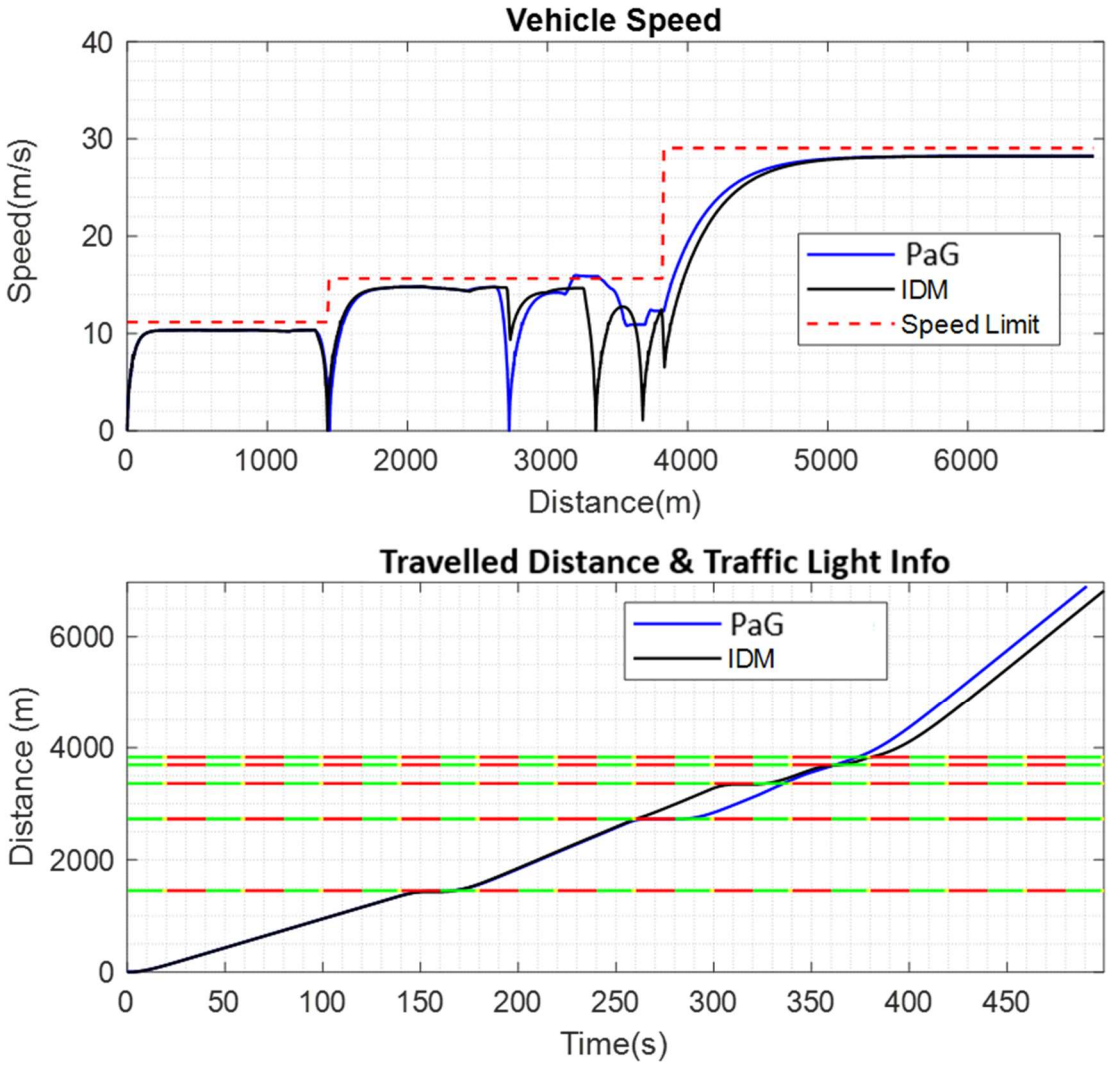


Figure 39: Comparison of IDM with Pass at Green velocities with respect to distance

As far as fuel economy is concerned, the vehicle travelling with Pass at Green model has a 7.01% fuel economy benefit compared to the vehicle travelling with the IDM driver (Table 3).

| Method | Total Fuel (gal.) | FE (mpg) | FE improv. (%) |
|---|---|---|---|
| IDM | 0.1009 | 42.51 | |
| Pass at Green | 0.0943 | 45.49 | 7.01 |

Table 3: Fuel economy comparison of IDM and Pass at Green models for a realistic road simulation

4.5 Summary

This chapter presented the HIL simulation for modifying the speed profile of a connected and autonomous vehicle at signalized intersections. The developed simulation environment speeds up the V2I application development in a safe and realistic environment with automotive grade hardware and software. To show the capabilities of the HIL simulator, the Pass at Green algorithm**,** which lowers the idling time at signalized intersection**s** and improves fuel economy has been simulated**.** By simulating the human driver behavior with IDM and comparing it with the Pass at Green model, this chapter demonstrated the possibility of fuel efficiency improvement by using DSRC technology (Table 3).

# Chapter 5. Green Light Optimized Speed Advisory (GLOSA) with Traffic Preview

By utilizing the vehicle to infrastructure communication, the conventional Green Light Optimized Speed Advisory (GLOSA) applications give speed advisory range for drivers to travel to pass at the green light [69]. However, these systems do not consider the traffic between the ego vehicle and the traffic light location, resulting in inaccurate speed advisories. Therefore, the driver needs to intuitively adjust the vehicle's speed to pass at the green light and avoid traffic in these scenarios. Furthermore, inaccurate speed advisories may result in unnecessary acceleration and deceleration, resulting in poor fuel efficiency and comfort. To address these shortcomings of conventional GLOSA, in this chapter, the utilization of cooperative perception messages shared by smart infrastructures to create an enhanced speed advisory for the connected vehicle drivers and automated vehicles is proposed. Two different algorithms were designed by utilizing the available traffic preview (Signal Phase and Timing (SPaT), MAP, and Collaborative Perception Messages), predicted traffic preview from these messages, and measurements from onboard range sensors. While in the first algorithm, the vehicle is controlled with a rule-based approach, a reinforcement learning-based approach is used in the second algorithm. The designed algorithms are then simulated in a simulation environment created in a Simulink. Simulation results demonstrated the effectiveness of the developed algorithms with better fuel efficiency performance and more comfortable ride performance. The GLOSA with Traffic Preview application presented here is a case study of the cooperative perception concept presented in Chapter 2. Cooperative perception examples in Chapter 2

were simulation based as it is difficult to run experiments since multiple cooperating vehicles are needed. A smart intersection does this cooperative messaging automatically as it monitors all incoming traffic and broadcast BSMs for all those vehicles, enabling a basic implementation of cooperative perception.

## 5.1 Background

One of the first applications of the vehicle to infrastructure communication is called Green Light Optimized Speed Advisory (GLOSA). However, early versions of GLOSA did not utilize the queue information [63], [70]. Therefore, when vehicles are approaching to an intersection, the advisory speed announced by the conventional GLOSA models is not accurate, forcing drivers to adjust their speed based on their intuition. While the idea of utilizing smart intersections for queue detection is not new [71], in this dissertation, two different methods are proposed to use the collective perception messages broadcasted by the smart intersections to predict the traffic preview.

One can use different traffic preview models from the literature [72]–[75] to predict the traffic flow of the road of interest. However, this work mostly focuses on developing a GLOSA application with a traffic preview, assuming that the traffic preview prediction is available. Therefore, instead of creating an entirely new prediction algorithm, simple representative traffic preview prediction methods are developed to explore the potential benefit of traffic preview. Then two different driver models which utilize traffic preview are developed. Finally, this use case scenario (GLOSA with Traffic Preview) is evaluated

for different parameters to identify the benefits of the developed algorithms. Fuel efficiency and comfort metrics are chosen as the main parameters for the evaluation.

Current technology makes it possible to collect and share traffic data such as vehicle speed and trajectories and traffic flow rates. One can classify the traffic data into three different layers, namely macro-layer, micro-layer, and nano-layer. The macro-layer data consists of traffic flow data, and it is typically shared by map providers such as Google Maps, HERE Maps, and similar for a large traffic network. It is updated approximately every minute and can be used for route planning applications from point A to B. Micro layer traffic data is the traffic flow information between route segments such as two intersections. This data is updated around one-second intervals, and it is provided by map providers as well. The micro-layer data can be used to predict arrival time and refine route selection. Finally, the nano-layer traffic data can be defined as high-definition traffic data in the form of traffic flow or speed, location, and direction of each vehicle at an intersection. The nano-layer data can be potentially broadcasted by connected vehicles or smart intersections at approximately 10 Hz. This study mostly focuses on the utilization of nano-layer traffic flow data. The frequent update rate and detail of the nano-layer traffic data enable many vehicle control applications, including GLOSA, in which energy consumption, travel time, and passenger comfort improve. The nano-layer data can be used either to create an advisory speed for the human drivers or automated vehicles. This chapter presents two different methods that utilize the traffic preview information in a GLOSA application. While the first method relies on rule-based methodologies, the second method relies on a

reinforcement learning model to provide more accurate speed advisories to drivers than the conventional GLOSA and forms one of the major contributions of this dissertation.

### 5.1.1 Traffic Preview

Some traffic preview data is already known by the vehicle to infrastructure communication or the range sensors on the ego vehicle. Some of the information coming from the vehicle to everything (V2X) communication can be listed as traffic light signal phase and timing (SPaT), the layout of the intersection (MAP), location and speed of each vehicle as part of the cooperative perception message (CPM) in the form of basic safety message (BSM). In addition, distance and speed of the target vehicles can be acquired from the onboard range sensors such as radar, lidar, and stereo cameras. While this information is known, the future speed distribution for the road of interest is not known. One can statistically predict the future speed distribution, and then it can be used for developing new automated driving algorithms, such as GLOSA or traffic light management applications. This work utilizes the predicted traffic preview to develop an advanced GLOSA. However, this work does not focus on the development of traffic preview prediction. Therefore, in the first approach, an intelligent driver model (IDM) based microsimulation environment is used for creating a representative prediction of the traffic preview. The developed traffic preview microsimulation environment accepts SPaT and MAP messages as input from the traffic light alongside the cooperative perception messages (CPMs). By initializing the simulation with the input messages, the micro simulator runs the simulation for a predetermined traffic preview horizon every 100ms. Each simulation calculates the predicted passing time for target vehicles at the green light for each vehicle reported in CPMs.

### 5.1.2 Intelligent Driver Model

Intelligent Driver Model (IDM) [76] is a mathematical model for the car-following behavior of human drivers. The formulation of the intelligent driver model is given in equations 5.1-5.3.

$$\dot{x}_\alpha = \frac{dx_\alpha}{dt} = v_\alpha \tag{5.1}$$

$$\dot{v}_\alpha = \frac{dv_\alpha}{dt} = a\left(1 - \left(\frac{v_a}{v_0}\right)^\delta - \left(\frac{s^*(v_a, \Delta v_a)}{s_a}\right)^2\right) \tag{5.2}$$

$$s^*(v_a, \Delta v_a) = s_0 + v_a T + \frac{v_a \Delta v_a}{2\sqrt{ab}} \tag{5.3}$$

where:

$v_0$: the velocity the vehicle would drive at in free traffic

$s_0$: a minimum desired net distance

$T$: the minimum possible time to the vehicle in front

$a$: the maximum vehicle acceleration

$b$: a positive number

The original IDM formulation does not consider the traffic light. In this work, IDM is modified so that it obeys traffic lights following the car in front. Considering that the IDM can adjust its speed according to the vehicle in front, a rule base modification is made to make the IDM act as required by the traffic light. If there is an upcoming red traffic light, a virtual stopped vehicle is inserted into the simulation at the traffic light location. Then IDM calculations are performed in both of the two different cases. In the first case, the target vehicle is selected as the vehicle in front, and in the second case, to account the red traffic light the virtual stopped vehicle at the traffic light is selected as the target vehicle. Between two calculations, the minimum acceleration command is chosen to be applied to the simulated vehicle.

### 5.1.3 Comfort

According to [77], the ride comfort of passengers heavily depends on the magnitude of acceleration and its time derivative (jerk). While different studies suggest different comfort ranges for these parameters, the common consensus is that lower jerk and acceleration magnitudes result in better ride comfort. Jerk upper limit for discomfort in the literature ranges between 0.5 $m/s^3$ - 0.9 $m/s^3$. Acceleration upper limit for discomfort in the literature ranges between 1 m/s2 - 1.47 m/s2.

### 5.1.4 Smart Intersection

To fully benefit from the connected vehicle environment, it can be claimed that all traffic participants should communicate with each other. Unfortunately, considering that the penetration rate of connected vehicles on our current roads is significantly low, the connectivity functionality of the vehicles cannot be utilized efficiently. However, with

smart intersections, the feasibility of vehicle-to-everything V2X communication applications can be enhanced significantly. Smart intersections are equipped with range sensors such as cameras, lidar, and radar to detect and track vehicles and other vulnerable road users within the intersection zone. Then, the roadside communication unit (RSU) broadcasts the position and motion information of tracked vehicles and VRUs on behalf of them, as BSM messages. Therefore, a vehicle traveling at a smart intersection can be aware of other vehicles, which are not communicating, through V2I communication as if they are. In Figure 40, one can see the typical smart intersection environment. In the depicted intersection, vehicles are approaching from westbound to the intersection. The demonstrated intersection is equipped with a camera (or other sensors, such as lidar and radar or their combinations) to track the detected vehicles' speed, location, and heading. Then, position and motion information of tracked vehicles are encoded as BSMs to be broadcasted. The collection of these BSMs can be considered as a Collaborative Perception Message. The ego vehicle communicates with the smart intersection to receive SPaT, MAP, and CPM messages in the presented scenario. In addition to the communication sensor, the ego vehicle is also equipped with a range sensor to detect the target vehicle for automated longitudinal driving.

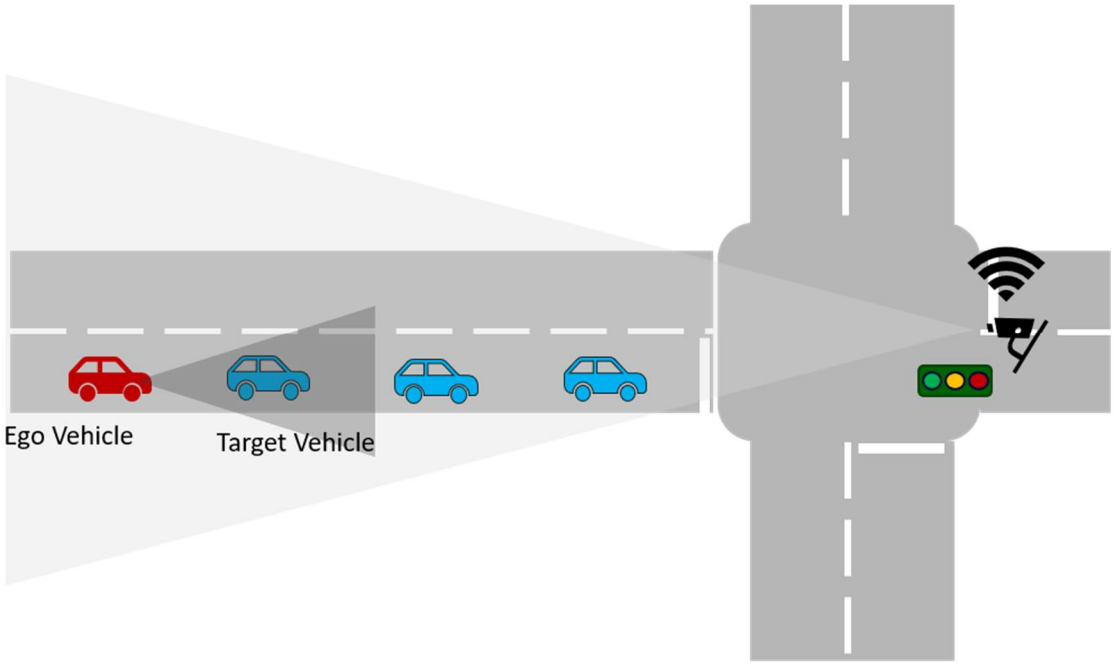


Figure 40 Smart intersection layout. Red vehicle is ego vehicle equipped with V2X onboard communication modem and range sensor. Smart Intersection is fitted with a camera (or range sensors, such as radar or lidar) alongside the V2X communication roadside modem. Smart intersection broadcasts SPaT, MAP, and CPM [78].

### 5.1.5 Green Light Speed Advisory

A simple conventional GLOSA system is designed and implemented to establish a baseline for evaluating the performance of the developed algorithms. A similar scenario to one shown in Figure 40 is constructed in the MATLAB simulation environment created to simulate the baseline GLOSA scenario. In the simulation, four vehicles are placed into the intersection, and all the vehicles are controlled with a modified IDM which drives the vehicles by considering traffic light status. The IDM controller of ego vehicle also receives GLOSA advisories to adjust the speed of the ego vehicle to pass at green when there is no

traffic. In the simulation, the initial positions (m) and speeds (m/s) of the vehicles are set as [0, 50, 105, 120] and [8, 15, 0, 0] where the traffic light is located at 130 m and the ego vehicle is the one located at 0 m. Two of the vehicles between the ego vehicle and traffic light wait for the traffic light to turn green, and the third vehicle and ego vehicle are approaching towards the traffic light. This simulation setup will be used throughout the chapter for simulations to ensure a consistent comparison between different simulation runs. The simulation result for the described baseline scenario is shown in Figure 41. As seen from the plots, GLOSA does not consider the traffic between the ego vehicle and the traffic light. Therefore, the maximum speed advised by the conventional GLOSA system (GLOSA max) is not accurate. In the simulated scenario, the ego vehicle accelerates to reach maximum speed; however, when it encounters the target vehicle, it has to slow down because the target vehicle is also slowing down for the stopped vehicles at the intersection. As a result, lack of traffic preview resulted in unnecessary accelerations and decelerations.

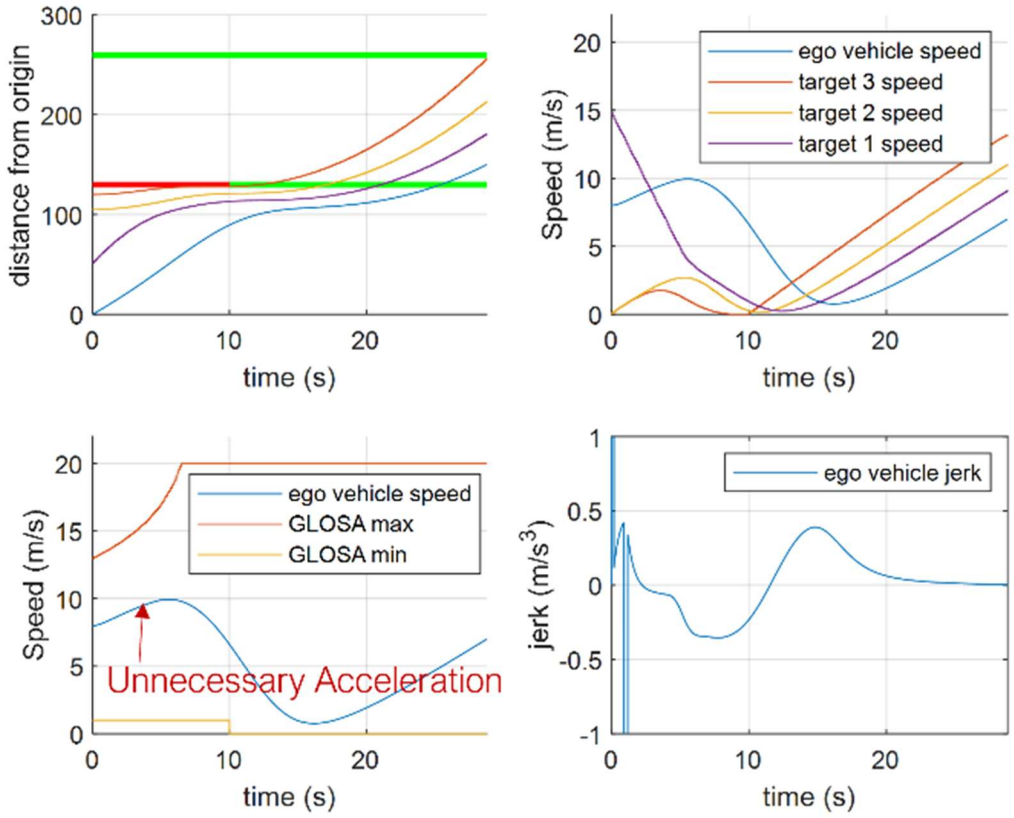


Figure 41 Baseline scenario: ego vehicle is driven with IDM and baseline GLOSA [78].

## 5.2 A Representative Traffic Preview Prediction Method 1: Passing Time Prediction for Target Vehicle

As demonstrated in the previous section, GLOSA without traffic preview results in unnecessary acceleration and decelerations. While the effect of traffic can be compensated by human drivers intuitively, with a traffic preview prediction, it is possible to provide more accurate advisory speeds to drivers and automated vehicles. This work concentrates on how a traffic preview prediction can be used for designing a more precise GLOSA. As traffic prediction is not the main focus of the work, instead of more complex prediction

models, a simulation-based simple representative traffic preview model is developed. The developed preview model is given in Algorithm 5.1.

**Algorithm 5.1: Passing Time Prediction for Target Vehicle**

***Inputs:*** *SPaT, MAP, CPM*

***Output:*** *Predicted passing time of target vehicle*

***1*** *Get current speed and locations of vehicles and SPaT, MAP*

***2*** *Initialize each vehicles position and speed with an IDM*

***3*** *Simulate the created vehicles for the prediction horizon*

***4*** *Log the time for the target vehicle to Pass at Green*

As summarized in Algorithm 5.1, the designed Passing Time for Prediction for Target Vehicle accepts SPaT, MAP, and CPM messages as input. It outputs the target vehicle's predicted passing- time (*PT*) at the green traffic light phase (Figure 42).

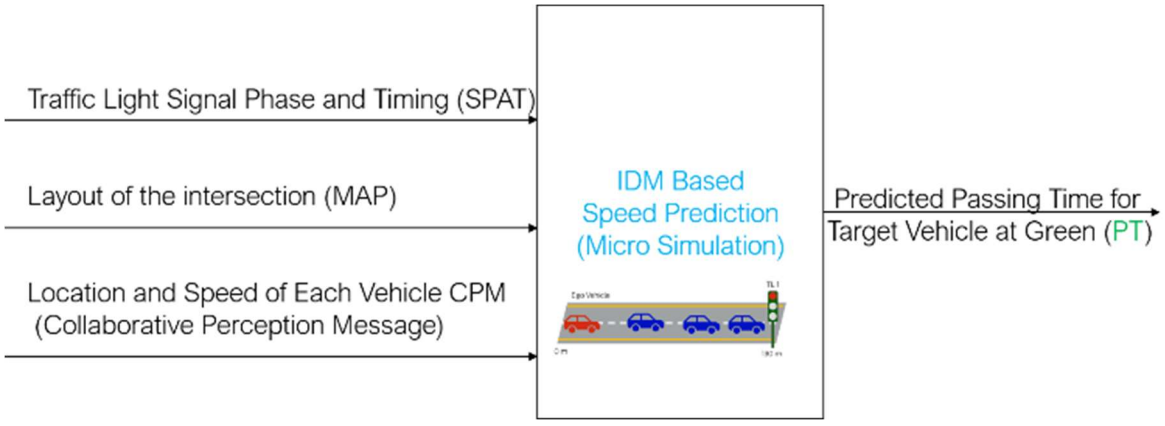


Figure 42 Traffic Preview Prediction unit utilizes the SPaT, MAP, and CPM to predict passing time (at the green light) for each vehicle reported in CPM [78].

The prediction is performed every 100ms. Vehicles reported in the CPM are initialized with an IDM with the reported initial speed and locations for each prediction step. Then for the selected time horizon, the vehicles with IDM are simulated such that they follow the vehicle in front (if there is one) and obey traffic lights. The traffic light is simulated based on the SPaT message. Once the target vehicle (vehicle traveling in front of the vehicle) passes the traffic light, the passing time is logged as the output (predicted passing-time for the target vehicle). In Figure 43, one can see the simulated locations of detected vehicles for the prediction horizon of 30s. In the demonstrated example, the smart intersection detects four vehicles, including the ego-vehicle (vehicle under test). Based on the simulation results, the target vehicle (traveling in front of the vehicle) passes the traffic light located at 130m at 22s.

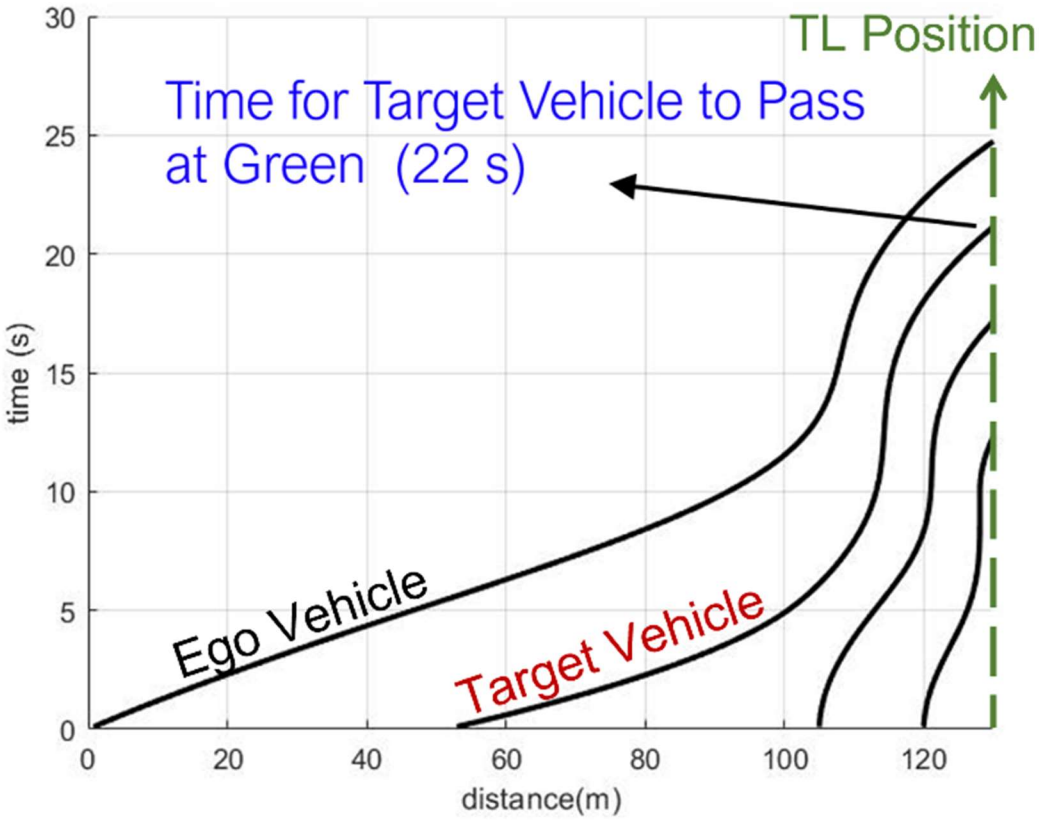


Figure 43 Visualization of predicted vehicle locations with respect to time by using the micro traffic simulation. By logging the passing time for each vehicle, the module outputs the predicted passing time for each vehicle [78].

The designed GLOSA with traffic preview (GLOSA-TP) system is the modified version of the base GLOSA model. In GLOSA-TP, alongside SPaT, MAP, and ego vehicle speed and location, the predicted passing time of the target vehicle (*PT*) is used as another input. Passing time, then, is converted to an advisory speed for the driver with Equations 5.4 and 5.55. Finally, using equation 5.4, passing time (PT) is converted to advisory speed with traffic preview, Traffic Advisory Speed (TAS).

*TAS=Distance to TL / (PT + Safe Time Headway)* (5.4)

Then as in equation 5.5, a new Predicted Speed Advisory (PSA) is selected as the minimum of GLOSA max and TAS as

*PSA=min (GLOSA max, TAS)* (5.5)

In the implemented scenario, if the PSA is lower than the vehicle's current speed, the vehicle is not allowed to accelerate. The simulation result for the GLOSA-TP is shown in Figure 44. As seen from the figure, the vehicle approaches the traffic intersection with a much smoother speed profile while avoiding unnecessary acceleration and decelerations, resulting in a more comfortable ride than the baseline scenario.

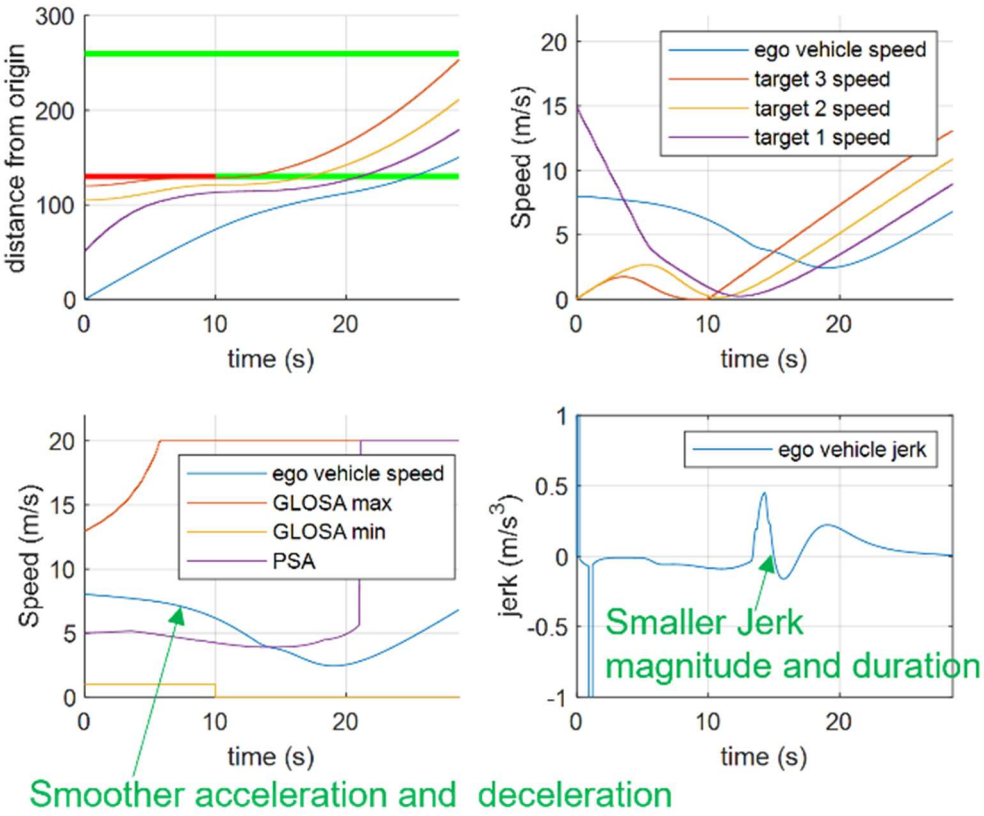


Figure 44 GLOSA-TP scenario: ego vehicle driven with IDM with the PSA received from GLOSA-TP [78].

To demonstrate the fuel efficiency benefit of the designed algorithm, the resulting speed profiles from the baseline scenario and the GLOSA-TP scenario are feed in to a simulation model of a conventional vehicle with an internal combustion engine powertrain [79] to simulate fuel consumption. While the baseline scenario completed the scenario with 29.27 miles per gallon (MPG) fuel efficiency, the proposed method showed 39.37 MPG fuel efficiency. Therefore, for this specific scenario, a 35% fuel efficiency improvement is shown with the prosed method.

### 5.2.1 Real-World Data Simulation

The developed algorithm is also tested with real-world data. One of the earliest smart intersections in the US is located at Marysville, OH [80], [81]. This intersection is equipped with cameras and a DSRC vehicle to infrastructure (V2I) communication roadside unit (RSU) as shown in Figure 45. The smart intersection broadcasts the detected vehicles and pedestrians speed and location information as BSM messages at 10 Hertz. The intersection also broadcasts the layout information of the intersection with the MAP message and also SPaT information. The developed algorithm is simulated with the data recorded from this specific intersection to demonstrate the potential benefits of the proposed algorithm. As compared to the baseline algorithm, the proposed algorithm resulted in less speed deviation (Figure 46). Also, in the baseline application, the ego vehicle had to stop for the traffic light. On the other hand, with the proposed algorithm, the vehicle passes the intersection at the green light without stopping. With the proposed algorithm, a 41% fuel efficiency performance is observed.

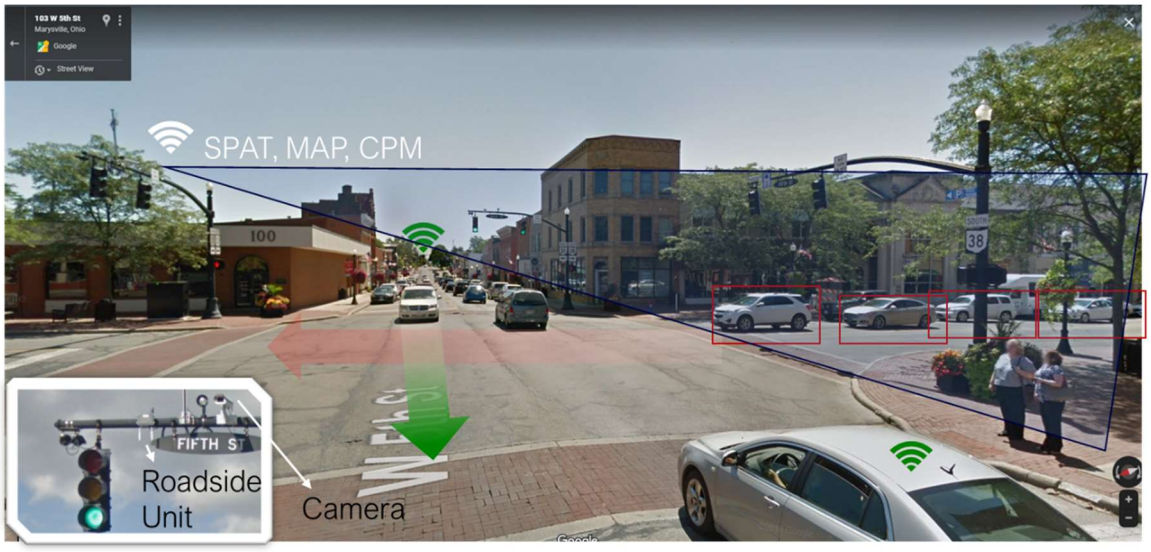


Figure 45 Marysville, OH smart intersection layout. Background image is retrieved from Google Street Views.

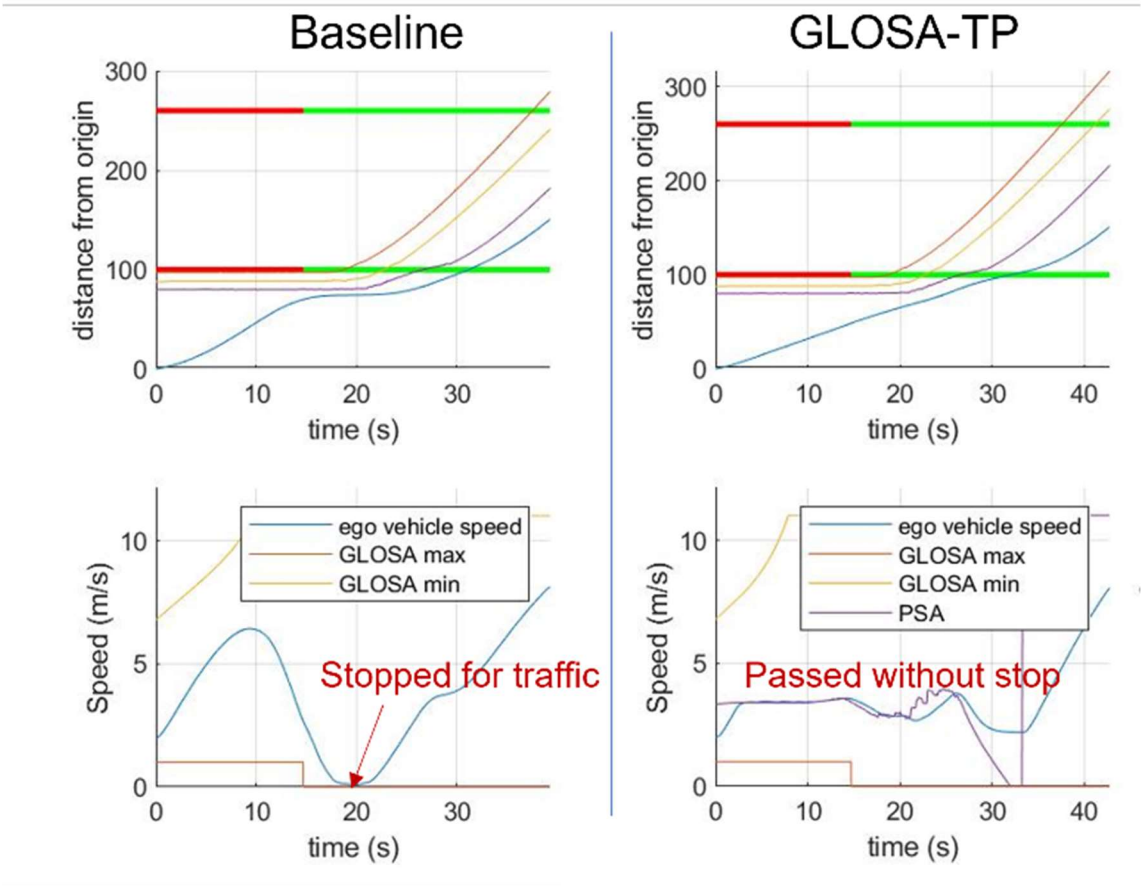


Figure 46 Left: Baseline simulation with real-world traffic data. Right: GLOSA-TP simulation with real-world traffic data [78].

### 5.3 A Representative Traffic Preview Prediction Method 2: Spatial & Temporal Speed Prediction

As another method, the traffic preview is predicted as temporal and spatial speed prediction for the predetermined time and location horizons. For the prediction, the same IDM-based microsimulation environment is used. Similar to the previous method, the inputs of the microsimulation are SPaT, MAP, and CPMs. The microsimulation is run for each prediction step for the determined time and distance horizon by utilizing the aforementioned inputs as initial states. From the simulated vehicle trajectories, the

predicted speed information of each CPM vehicle is placed into the time distance grid, as shown in Figure 47. At the boundary of the grid, the speed information can be determined by utilizing the SPaT information and the speed limit. The algorithm for creating the temporal and spatial speed prediction is given in Algorithm 5.2. The output of the algorithm for one example time step is shown in Figure 47.

**Algorithm 5.2: Spatial & Temporal Speed Prediction**

***Inputs:*** *SPaT, MAP, CPM*

***Output:*** *Spatial & Temporal Speed Prediction*

***1*** *Get current speed and locations of vehicles and SPaT, MAP*

***2*** *Initialize each vehicle position and speed with an IDM*

***3*** *Simulate the created vehicles for the prediction horizon*

***4*** *Log the speed predictions on the spatial & temporal grid.*

***5*** *Determine grid boundary values using SPaT and Speed Limit*

***6*** *Interpolate the values for unknown grid points from known data points.*

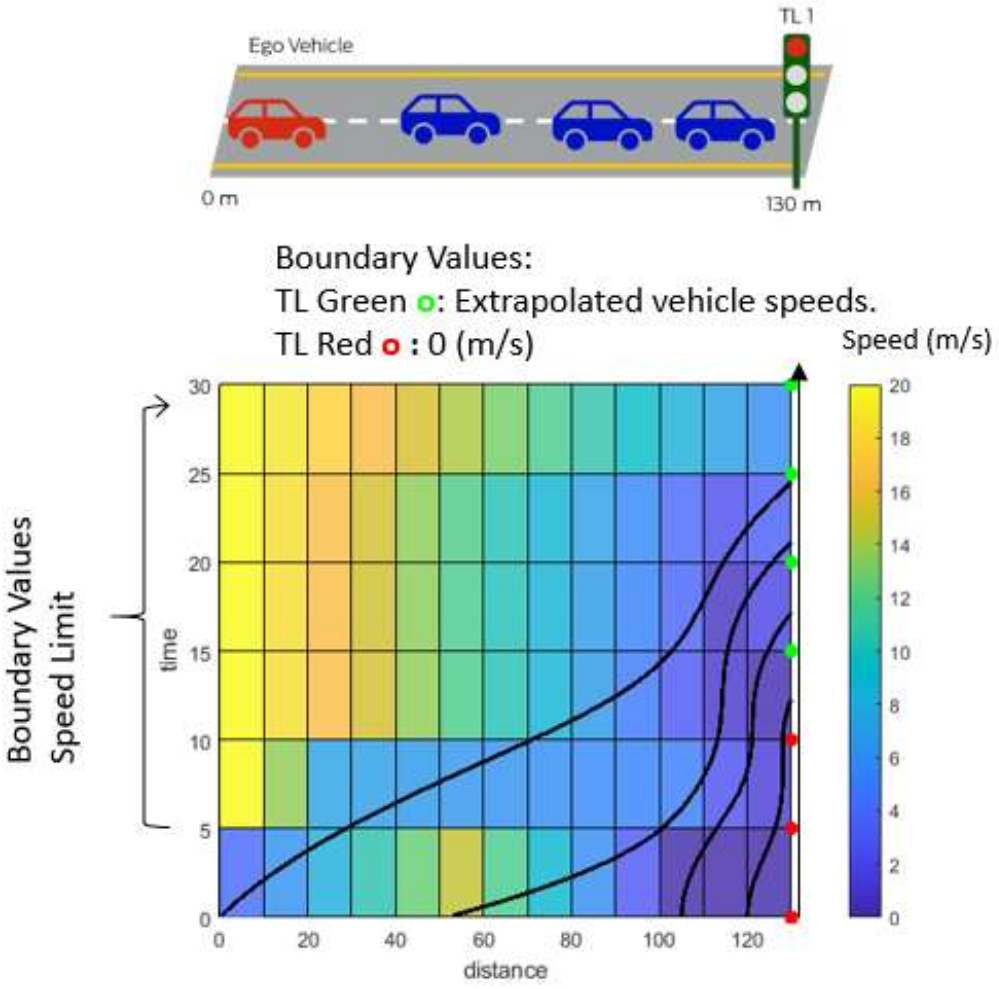


Figure 47 The temporal and spatial speed prediction distribution for selected horizon (distance range 0-130m, time range 0-30s)[78].

## 5.4 GLOSA-TP with Reinforcement Learning

As a second method, GLOSA-TP is designed with a reinforcement learning model approach. A reinforcement learning model is a machine learning-based control algorithm that takes actions to maximize the rewards collected over time. In the proposed use case scenario, instead of providing an advisory speed to the vehicle, a more general spatial and temporal speed prediction is fed to the reinforcement learning model to enable vehicle

control by optimizing the fuel economy and comfort of the passengers. For this purpose, a Deep Deterministic Policy Gradient (DDPG) reinforcement learning (RL) agent [82] is trained in the developed simulation environment to maximize the cumulative reward over time.

The action (output) of the designed reinforcement learning actor is selected as the acceleration command. Observations for the designed reinforcement learning agent consist of the predicted spatial and temporal speed distribution, GLOSA minimum and maximum speeds, ego vehicle speed, location, acceleration, jerk, and the maximum speed calculated for the safe car following time gap. The agent is rewarded when the vehicle maintains the vehicle's speed within the desired speed range by GLOSA as given in Equation 5.6. On the other hand, if the vehicle decelerates, the agent is penalized to ensure fuel efficiency as given in Equation 5.7. Similarly, the RL agent is penalized for higher jerk magnitudes to avoid an uncomfortable ride as given in Equation 5.8. During the training the reinforcement learning agent aims to maximize the cumulative reward. The designed RL system architecture is shown in Figure 48.

$$R_{speed} = \begin{cases} if\ (V_{ego} > V_{max}) & -(V_{max} - V_{ego})^2 \\ if(V_{GLOSA_{min}} < V_{ego} < V_{max}) & 0.1 * V_{ego} \\ if(V_{ego} < V_{GLOSA_{min}}) & -(V_{ego} - V_{GLOSA_{min}})^2 \end{cases} \quad (5.6)$$

$$R_{deceleration} = -2 * d^2 \quad (5.7)$$

$$R_{jerk} = -0.1 * jerk^2 \quad (5.8)$$

Visualization of the speed cost and rewards and jerk cost are shown in Figure 49 and Figure 50 respectively.

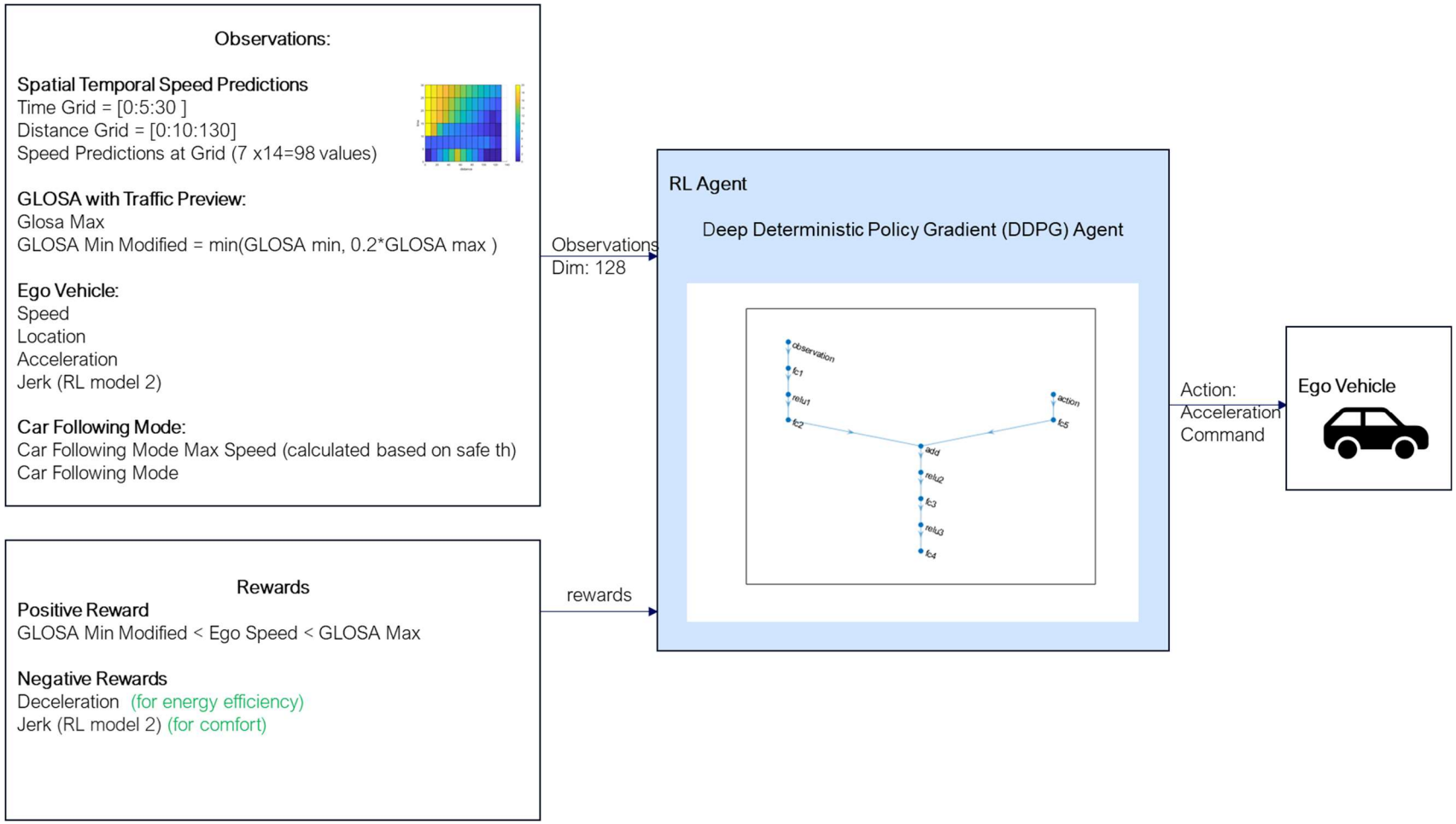


Figure 48 Reinforcement learning model system architecture.

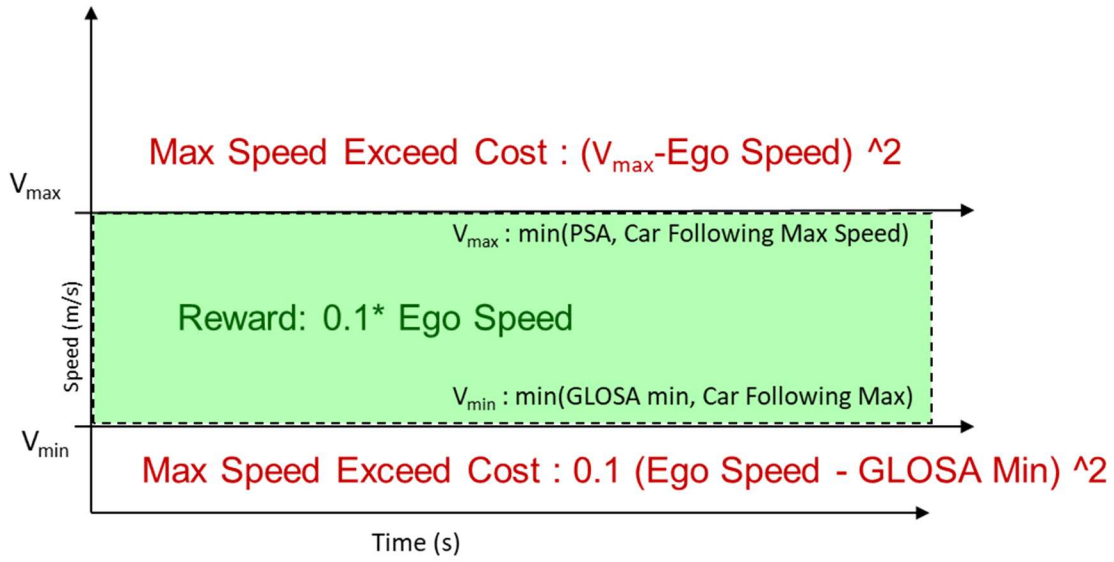


Figure 49 Visualization of speed related costs & rewards.

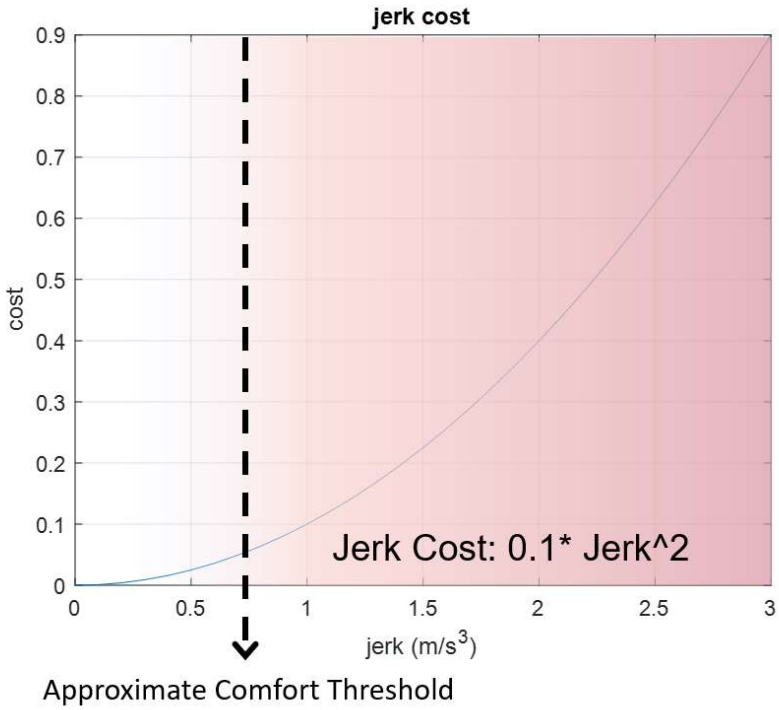


Figure 50 Visualization of jerk cost [78].

As stated earlier, the designed reinforcement learning agent architecture is an actor-critic neural network. While the actor neural network consists of four hidden layers, the critic network consists of five hidden layers. These hidden layers consist of fully connected layers followed by a rectified linear unit (RELU) layer. Each of the hidden layers are formed from 480 neurons. The architecture of the reinforcement learning model agent can be seen in Figure 51. The designed DDPG actor is a significantly redesigned version of a DDPG Adaptive Cruise Controller agent presented in [83].

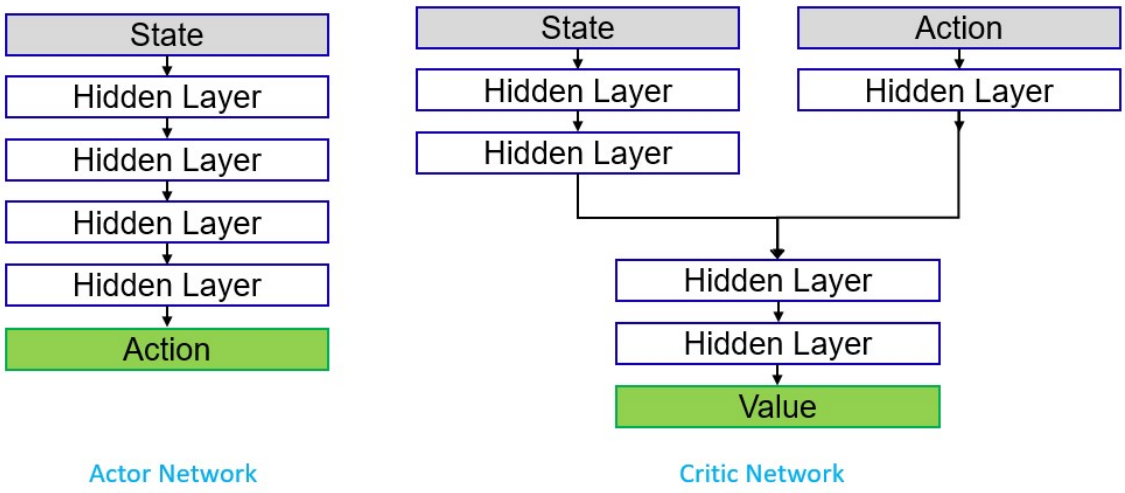


Figure 51 Reinforcement agent actor and critic networks [78].

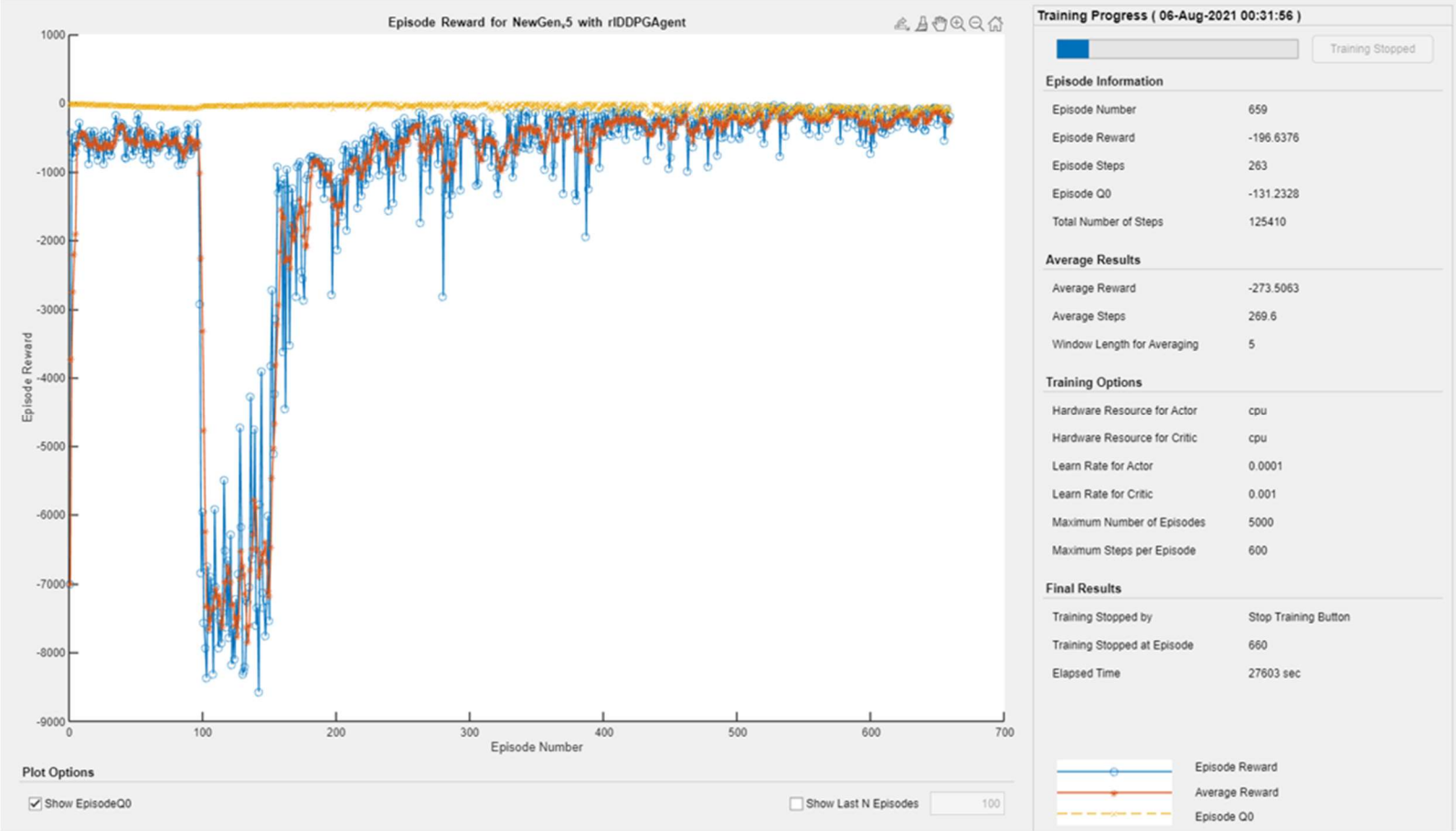


Figure 52 Training statistics for the designed RL model.

### 5.4.1 Simulation Results

The designed reinforcement learning model is trained with two different reward/cost input sets (Figure 52). While the first training set did not have the jerk as one of the cost inputs, in the second training set, the jerk cost is fed to the agent to improve the comfort level of passengers. Since the agent trained with the jerk cost is expected to increase the ride comfort by reducing jerk, the remaining simulations will be demonstrated with the agent trained with the jerk cost.

When we compare the performance of the reinforcement learning model with the baseline algorithm, we can see that it outperforms the baseline by providing a smoother speed profile. Finally, the designed GLOSA-TP models were compared with the baseline model for different initial ego vehicle speeds. The initial speed of the ego vehicle is varied to diversify the scenarios. For each scenario, the logged fuel economy results are shown in Table 4. The fuel efficiency benefit of both algorithms (GLOSA-TP rule-based and RL based model) is also shown in the bar graph in Figure 53. As can be seen from these results, both of the algorithms improve the fuel efficiency compared to the baseline.

| **Initial Speed** | **GLOSA (Baseline)** | **GLOSA with Traffic Preview** | | **RL w Jerk Cost** | |
|---|---|---|---|---|---|
| | FE (MPG) | FE (MPG) | Benefit (%) | FE (MPG) | Benefit (%) |
| 2 | 22.39 | 24.4 | 9 | 25.37 | 13 |
| 4 | 23.05 | 26.1 | 13 | 27.95 | 21 |
| 6 | 25.65 | 29.18 | 14 | 32.59 | 27 |
| 8 | 29.27 | 39.37 | 35 | 41.06 | 40 |
| 10 | 32.37 | 49.01 | 51 | 53.48 | 65 |
| 12 | 44.91 | 55.52 | 24 | 61.86 | 38 |
| 14 | 54.42 | 59.66 | 10 | 57.44 | 6 |
| 16 | 68.47 | 77.8 | 14 | 69.91 | 2 |

Table 4 Fuel economy benefits of developed GLOSA with traffic preview algorithms [78].

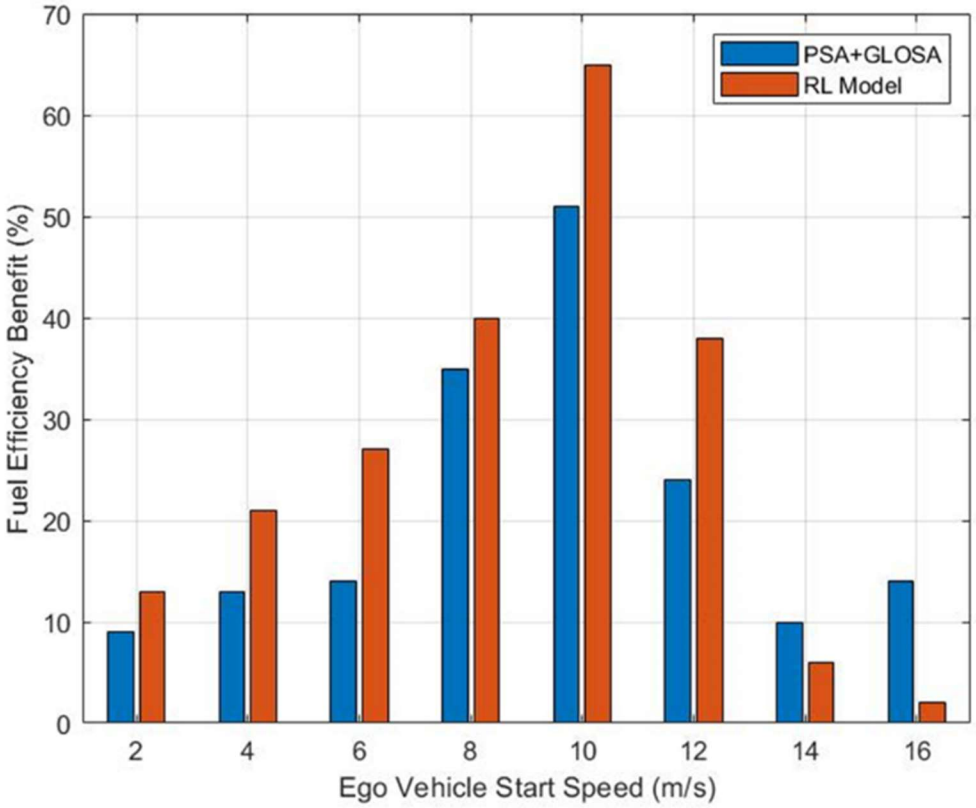


Figure 53 The fuel economy benefits of developed algorithms as compared to the baseline scenario [78].

## 5.5 Summary

Considering that conventional GLOSA systems do not consider traffic queue information, this work presents two different approaches for generating speed advisory for connected vehicle drivers and automated vehicles. Both algorithms rely on predicting the future traffic flow characteristic at a smart intersection. Representative traffic prediction models for each algorithm that predicts vehicle future speed and the location at a smart intersection, were developed. Then, the prediction models were used to evaluate potential benefits of the developed GLOSA-TP algorithms. Then, with simulations, it was demonstrated that the

developed GLOSA-TP algorithms can improve fuel efficiency and comfort of passengers by avoiding unnecessary acceleration/deceleration and jerk. The presented fuel efficiency benefits reflect the fuel efficiency trend for different scenarios rather than highly accurate fuel efficiency analysis. The main fuel efficiency benefit will be achieved by avoiding unnecessary stops and thus avoiding unnecessary accelerations after the stop or unnecessary idling at the traffic light. To demonstrate the effectiveness of the algorithms more realistically, one needs to extend the simulation network and extend the simulation scope by analyzing the fuel consumption of vehicles for a time period that includes the departure of the vehicles from the traffic light. This was not done in this chapter as fuel consumption analysis is not the main topic of research here. GLOSA with Traffic Preview is only presented as a case study of cooperative perception and the real data in the smart intersection in Marysville does not have data for vehicles after they pass the traffic light.

# Chapter 6. Longitudinal Control in Connected and Automated Driving

In this chapter a design and implementation of a Cooperative Adaptive Cruise Controller (CACC) on an autonomous vehicle platform (2017 Ford Fusion) is presented. This is used for car following tasks in connected and automated driving. The developed CACC controls the intervehicle time gap between the target vehicle and ego vehicle using a feedforward PD controller. In this design, the feedforward information is the acceleration of the target vehicle which is communicated through Dedicated Short-Range Communication (DSRC) modems. This chapter explains the detailed architecture of the designed CACC with hardware and methods used for the both simulation and experiments. Also, an approach to overcome detection failures at curved roads is presented to improve overall quality of the designed CACC system. The presented results indicate that CACC improves the car following performance of the ego vehicle as compared to the classical Adaptive Cruise Controller.

## 6.1 Background

With the recent advancements in automotive sensors, cars are becoming more autonomous by making use of these new technologies. The Advanced Driver Assistant systems such as the Adaptive Cruise Control (ACC) system do not only ensure the safety but also increase the comfort of travel. A well-known longitudinal control method, Cooperative Adaptive Cruise Control (CACC), which is an ACC system supported by the Dedicated Short-Range Radio Communication (DSRC) technology that allows Vehicle-to-Vehicle (V2V) communication, enables the achievement of lower time gaps. Reducing the time gap distance between two vehicles can significantly increase the capacity of roads. Also,

platooning multiple vehicles has the potential to improve the fuel efficiency of the vehicles by avoiding unnecessary accelerations and decelerations, by reducing the air drag experienced by the following vehicles. Motivated by these advantages, in this chapter, the design and implementation of CACC on an autonomous vehicle platform (2017 Ford Fusion) with experimental results is presented.

The designed CACC maintains the desired constant-time gap better than the well-known Adaptive Cruise Control (ACC). Thus, it is possible to reduce the gap for CACC. In truck platooning, smaller time gap results in higher fuel efficiencies by reducing the air drag resistance. Similarly reducing the gap time would increase the capacity of the highways significantly by improving the traffic flow rate [84]. Motivated by these advantages of CACC over ACC, in this chapter, a Cooperative Adaptive Cruise Controller design process for the autonomous vehicle platform is presented.

Adaptive Cruise Controllers are already being used in production vehicles under different names. A comprehensive literature review for ACC systems is done in [85]. Adaptive Cruise Controller aims to maintain the time gap constant during car following maintaining string stability. However, ACC cannot use low time gap values since it would result in rear end collision in case of sudden speed changes in traffic and can not damp out shock waves very well [86]. In ACC a small time gap causes string instability by amplifying the disturbances in the upstream direction. Using DSRC communication, one can improve the car following performance by reducing the time gap without breaking the string stability [87]. This car-following model is called Cooperative Adaptive Cruise Control. Some of the earlier work on CACC can be seen in [87]–[93]. In [87] authors presented their CACC

design methodology by considering the string stability requirements and they experimentally validated their design. One of the early implementations of CACC was done under California PATH program [88], [89]. In 2011 several research institutes formed a CACC platoon at Grand Cooperative Driving Challenge (GCDC). Two of the CACC implementations in this challenge can be seen in [91], [94]. In [92], authors presented their design for car-following with CACC and approaching maneuver controller. In [93], authors presented multi vehicle look ahead CACC simulation results which shows that the multi vehicle look ahead in CACC improves its performance.

The next section will explain the designed CACC structure. Following that the simulation environment with target vehicle modeling and simulation results for the two vehicle car following scenario will be presented. Then, the experimental vehicle set up with the explanation of sensors will be explained. In the Perception section, in-lane vehicle detection algorithm will be explained. Finally, the chapter will be concluded with experimental results and their comparison with simulation results.

## 6.2 CACC Architecture

The control structure of the designed CACC system is shown in the block diagram in Figure 54. The designed control system is similar to the one designed and shown to be string - stable in [87]. Since the vehicle does not have built-in ACC the low-level controller is designed as a gain-scheduled PI controller. As an upper level controller, a PD controller with a feed-forward controller is used. To sustain the string stability, a constant time gap spacing policy is employed [95]. The input of the feedforward controller is the acceleration

of the target vehicle which is transmitted through DSRC radio communication. Formulation of the spacing policy is given in,

$$\Delta x = x_1 - x_2 - l \tag{6.1}$$

$$\Delta x_{desired} = V_{hos}\ T_{hw} + standstill\ distance \tag{6.2}$$

where $l$ is the length of the target vehicle, $x_1$ and $x_2$are the position of the target and ego vehicle, $T_{hw}$ is the desired time-gap and $V_{host}$ is the speed of the ego vehicle. The designed PD controller minimizes the spacing error $e$ which is given as

$$e = \Delta x - \Delta x_{desired} \tag{6.3}$$

Gains of the PD controller are chosen as $k_P = {k_D}^2 = {w_K}^2$ where the $w_K$ is chosen to be close to the bandwidth of the low-level closed-loop bandwidth. The feedforward controller is designed in the same as it was designed in [91]. Formulation of the feedforward controller is given in Equation 6.4, where $1/\tau$ represents the desired closed-loop bandwidth.

$$F = \frac{\tau s+1}{T_{hw}s+1} \tag{6.4}$$

The architecture of the CACC system is illustrated in Figure 54.

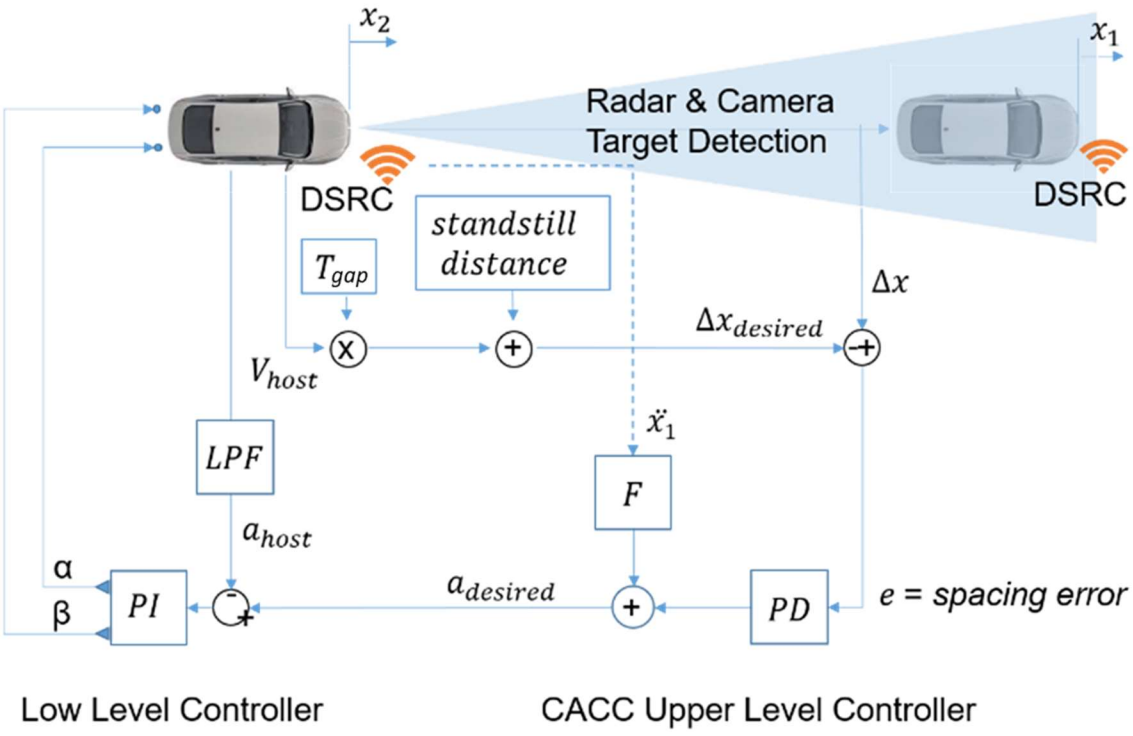


Figure 54 Cooperative Adaptive Cruise Controller (CACC) block diagram [96]

6.3 Simulation Environment

Development of the initial CACC model is done in CarSim-Matlab co-simulation environment [97]. CarSim is a vehicle simulation environment with the capability of simulating the dynamics of the vehicle. It can also simulate the target vehicle as a kinematic object. In the simulation, the target vehicle is driven by an Intelligent Driver model. By changing the desired speed and/or acceleration limits for the target vehicle, one can create different driving scenarios using Intelligent Driver Model (IDM) [76]. The formulation of the IDM was previously given in Equations 5.1-5.3.

The IDM car-following model is commonly used in traffic simulations for simulating driving behavior of the human driver in traffic. In this case, the intelligent driver model is used to model a human driver for the target vehicle. In CarSim, one can also create realistic roads by importing the GPS trajectory of the route. The simulation of the radar and camera is also possible by using the virtual sensors offered in CarSim. Figure 55 shows the visualization of the car following scenario simulation with a radar field of view.


Figure 55 CarSim CACC Simulation visualization [96]

After creating the simulation environment which replicates the structure shown in Figure 54 simulations run for two different scenarios: ACC and CACC. As the initial evaluation, the target vehicle in the created simulation environment first accelerates to a set speed of 20 km/h then it changes set speed to 25km/h, and finally it stops. In the simulations, the ego vehicle follows the target vehicle with 1 sec time gap. As it can be seen from simulation results in Figure 56, CACC follows the target vehicle much better. Although both of the

speed controllers maintain the time gap, CACC time gap follows the set value (1 second) more accurately.

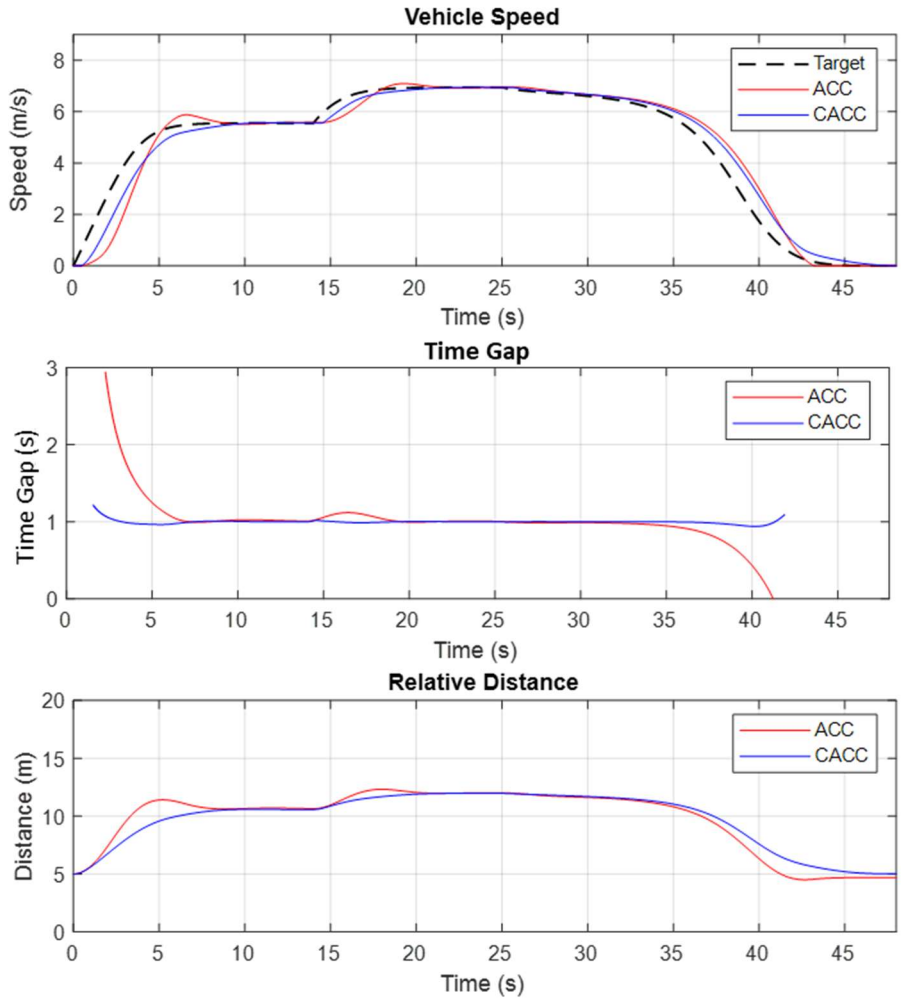


Figure 56 CarSim ACC and CACC simulation results for 1 second time gap with an IDM driven target vehicle [96]

In another scenario, in order to show the performance of the CACC in a more realistic scenario, the target vehicle speed and acceleration profiles over time are collected by driving the experimental vehicle in an urban route environment. By replaying the recorded data during the simulation, the real world driving experience with a sudden acceleration

and braking behavior of the target vehicle in an urban environment is simulated. Similar to the previous simulation results, CACC follows the desired time gap of 0.6s much better as compared to ACC (Figure 57). Especially for sudden changes in speed of the target vehicle, CACC responds much better and keeps the desired spacing more accurately.

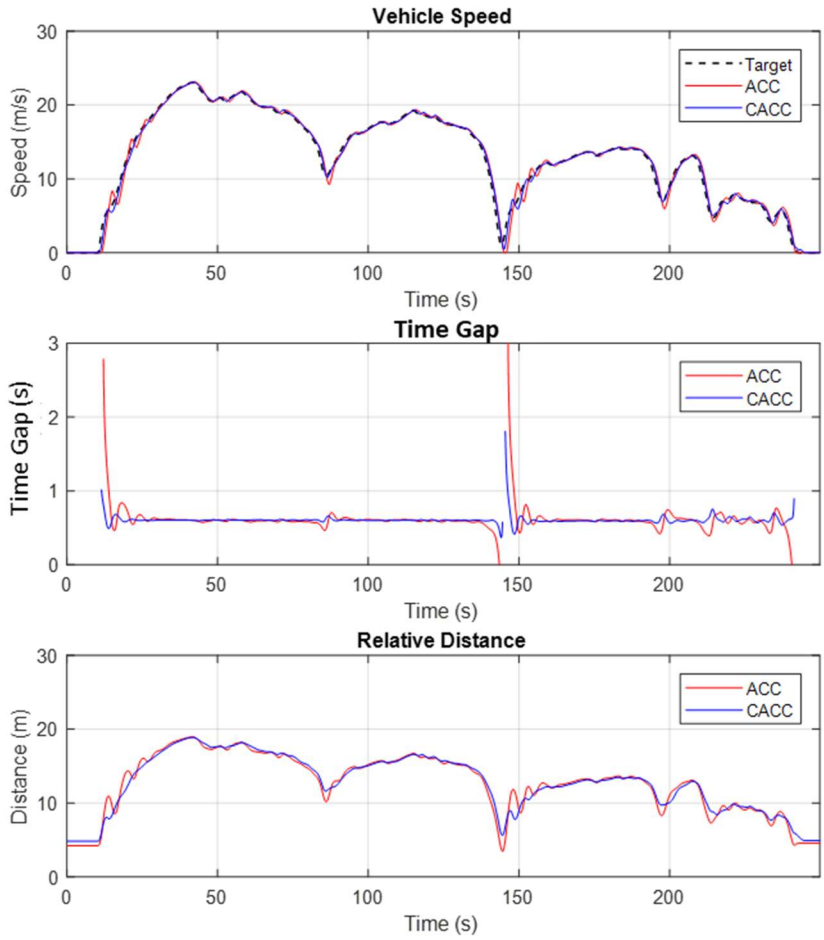


Figure 57 CarSim ACC and CACC simulation results for 0.6 second time gap with a human driven target vehicle data [96]

## 6.4 Experimental Vehicle


Figure 58 Autonomous vehicle development platform of Automated Driving Lab, at The Ohio State University [96].

For the experiments, a 2017 Ford Fusion with drive by wire capability is used (Figure 58). Since the vehicle is not equipped with an ACC, the throttle and brake actuation are realized through CAN bus messages. For the longitudinal motion measurements, speed and acceleration measurements on the vehicle CAN bus are used.

As the electronic control unit, a dSpace MicroAutoBox II (Figure 59) is employed due to its easy prototyping property with high-performance real-time system implementation capabilities. As it can be seen from the block diagram, all equipment in the vehicle is connected to the MicroAutoBox controller. All the measurements coming from the sensors, vehicle CAN bus are processed in the controller and throttle and brake commands based

on the embedded algorithm are sent to the vehicle. All the algorithms and the data parsing coming from the sensors are programmed using Simulink blocks and they are embedded into the MicroAutoBox controller. As the user interface, a portable computer with ControlDesk application is used.

To detect the objects on the road, the vehicle is equipped with a 76.5GHz Delphi forward looking radar which can track up to 64 objects and give their positions and relative velocity information. The radar is a combination of both long and middle range radars. Radar is connected to both MicroAutoBox and the laptop. While the data coming from the radar is parsed and processed in the MicroAutoBox, detections can be seen in real time for diagnosis purposes using DataView software. In order to visually validate the radar detections, a forward-looking webcam is connected to the laptop. DataView software can overlay the detections to the video stream acquired from this webcam (Figure 60).

A black and white monocular smart camera from Mobileye (Figure 61) is used to detect lane lines on the road to determine in lane vehicles among detected targets via radar. This camera can detect the lane line markers on the road and provides the lane line information in the form of 3$^{rd}$ order polynomials. Coefficients of the lane line polynomials are available on the CAN bus alongside the road curvature information.

The test vehicle is also equipped with two Denso WSU (Wireless Safety Unit) 5900 DSRC modems to communicate with the target vehicle. In the CACC scenario, the target vehicle broadcasts its acceleration alongside the Basic Safety Message (BSM) [47]. While the first modem is receiving the target vehicle acceleration, the second modem on the vehicle is used to transmit the acceleration of a virtual target vehicle.

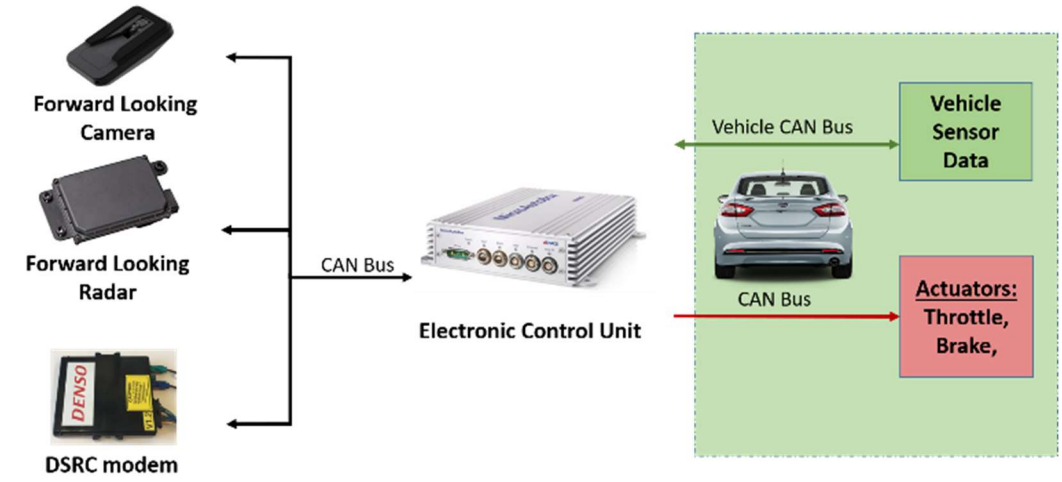


Figure 59 CACC experimental vehicle hardware block diagram[96]


Figure 60 Sample Radar Detection Visualization in real time [96]

## 6.5 Perception

A radar is used to detect the positions of the target vehicles. The downside of this method is that the radar only provides the position and speed information of the detected objects. Since the radar does not know the lane information on the road, it cannot distinguish the vehicles which are in the lane and which are not. From the radar, an ACC target information is available, which is selected by the radar using the speed of the vehicle, steering angle, and yaw rate information of the ego vehicle. Although this target detection is valid most of the time, since the exact algorithm for choosing the ACC target vehicle is not known and since the availability of the ACC target depends on the algorithm used in the radar another method was developed to detect the vehicles in the lane.

According to Zhang et.al [98], there are two main difficulties in detecting the target vehicle using radar. First, differentiation of the lane change or curve entry/exit behavior is challenging. Second, when the vehicle in the next lane goes into the curvature, it can be misclassified as in the ego lane. To overcome these difficulties, a camera and radar are used together. While the lane boundary information comes from the camera, objects detections are acquired from the radar. For each time step, acquired detections are sorted by their longitudinal range. Then, each detected object is checked to see whether it is in the ego lane or not by comparing its lateral position with respect to the lane boundaries. The block diagram of this system can be seen in Figure 61. One should note that if at least one of the lane is not visible to the camera, it is required to create the lane boundaries synthetically. If only one of the lane lines is available, the other lane boundary is created using the available lane boundary information and the lane width. If both of the lane lines

are not available, it is assumed that the vehicle moves straight, and the target object is searched within a window where the width of the window is equal to the lane width.

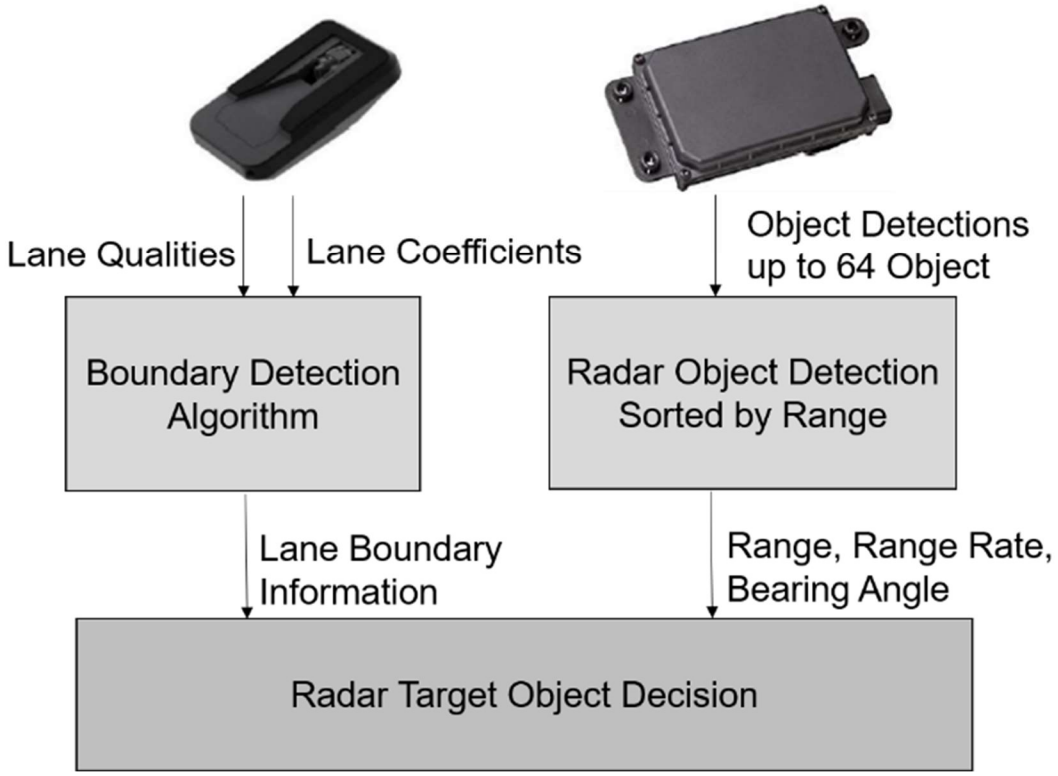


Figure 61 In lane vehicle detection structure with camera and radar combination [96]

Radar can provide the positions of the detected objects in polar coordinates (Figure 62). The measurements acquired from the radar for each object are listed with their range ($r_i$), bearing angle ($\alpha_i$) parameters and range rate. These coordinates are converted into the Cartesian coordinate system using basic trigonometric equations. Since the radar is placed in front of the front bumper and the origin of the radar coordinate system is chosen as the center of the radar, measurements acquired from the radar are converted to the longitudinal distance ($x_i$) and the lateral distance ($y_i$) of the target objects from the center of the front bumper (Equations 6.5 and 6.6).

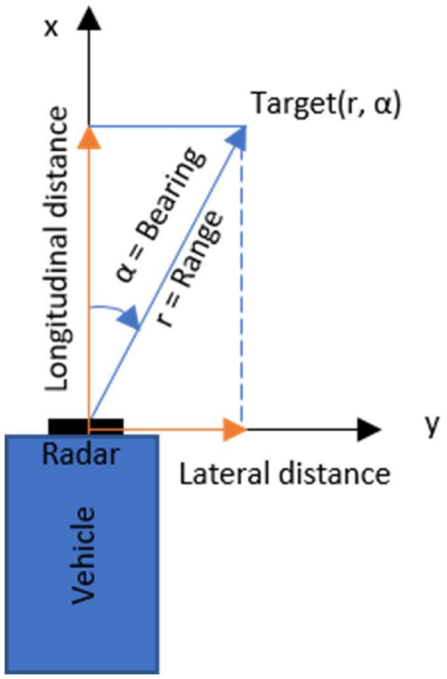


Figure 62 Radar detection coordinate system [96].

$$Longitudinal\ distance = x_i = r_i * \cos(\alpha_i) \tag{6.5}$$

$$Lateral\ distance = y_i = r_i * \sin(\alpha_i) \tag{6.6}$$

The camera in the vehicle is centered at the top of the windshield. It provides the lane line definitions as a third order polynomial in the camera coordinate system, where the origin of the coordinate system is the location of the camera. The curve fitted to the left and right lanes are given in

$$y^l(x) = {a_0}^l + {a_1}^l * x + {a_2}^l * x^2 + {a_3}^l * x^3 \tag{6.7}$$

$$y^r(x) = {a_0}^r + {a_1}^r * x + {a_2}^r * x^2 + {a_3}^r * x^3 \tag{6.8}$$

where $y$, $x$ represents the lateral and longitudinal positions of the points on the fitted curve. Each $a_i$ represents the coefficients of the fitted curve to the lane lines. Here r and l superscripts differentiate the curve fits for left and right lane, respectively. At the final step,

by knowing the positions of the targets and lane boundaries, the closest target in the lane can be chosen as the in-lane target. For this purpose, firstly, all the objects are sorted by their longitudinal distances. Then, each detected object's coordinates are translated to the camera coordinate system by adding the distance between the camera and the radar in the longitudinal direction ($\Delta x$) as

$$x_{new\_i} = x_i + \Delta \mathrm{x} \tag{6.9}$$

$$y_{new\_i} = y_i \tag{6.10}$$

Inserting the new lateral distance of the target objects into the Equations 6.7 and 6.8, the left (LB) and right boundaries (RB) of the lane at the distance of the target object are calculated as

$$LB_i = {a_0}^l + {a_1}^l * x_{new\_i} + {a_2}^l * {x_{new\_i}}^2 + {a_3}^l * {x_{new\_i}}^3 \tag{6.11}$$

$$RB_i = {a_0}^r + {a_1}^r * x_{new\_i} + {a_2}^r * {x_{new\_i}}^2 + {a_3}^r * {x_{new\_i}}^3 \tag{6.12}$$

If the lateral distance of the $i^{th}$ detected object is between the calculated lane boundaries for the longitudinal distance of the $i^{th}$ object, this object is considered to be an in-lane possible target using

$$(LB_i < y_{new\_i} < RB_i) \tag{6.13}$$

Among all in-lane possible targets, the closest vehicle in the longitudinal direction is accepted as the in-lane target vehicle. A sample experimental result for this method is presented in Figure 63. The lateral position of the target vehicle with respect to the center of the vehicle and lane boundaries are shown in the plot. One can see from the experimental result that the target vehicle is detected even in the curved section of the road accurately.

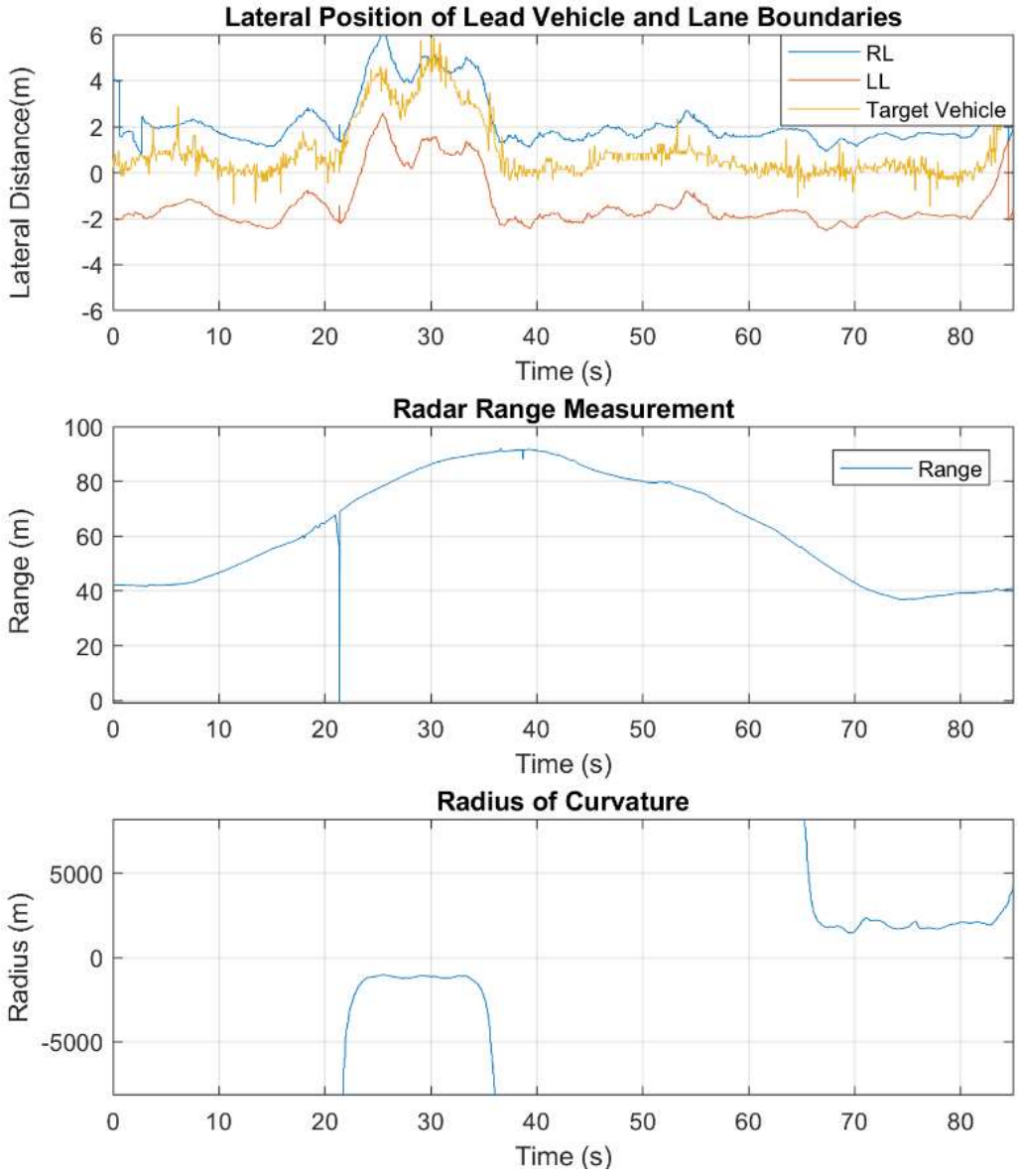


Figure 63 In lane vehicle detection experimental results [96].

## 6.6 Experimental Results

In the experiments, the target vehicle speed profile is chosen to be the same as the simulation target vehicle speed profile. The target vehicle profile is generated in real time with the IDM driver, similar to the simulation environment. The virtual target vehicle

accelerates to 20 km/h and 25 km/h consecutively and then it stops. In the CACC scenario, the simulated acceleration values for the target vehicle are broadcasted through DSRC OBU and they are received by another OBU for the ego vehicle. One can see the experimental results for the ACC and CACC for 1 second gap time overlaid onto the simulation results in Figure 64. The simulation results match with the experimental results. The small mismatches between the experiment and CarSim simulation are caused by the fact that the CarSim vehicle model is not an exact model of the experimental vehicle. In the CarSim simulation, a generic D class vehicle model is used. In response to the speed changes in the target vehicle speed profile, the CACC speed controllers start accelerating and deceleration faster as compared to ACC using the target vehicle acceleration information coming from the DSRC modem. Thus, the CACC controller can follow the target vehicle more accurately. CACC time gap following performance is much better than the performance of ACC.

Similarly, CarSim simulations and experiments are repeated for desired time gap of 0.6 s. The comparison of the simulation and experiment results are shown in Figure 65. Similar to the previous case, the simulation and experimental results are close to each other. CACC performs better while following the target vehicle with constant 0.6s time gap.

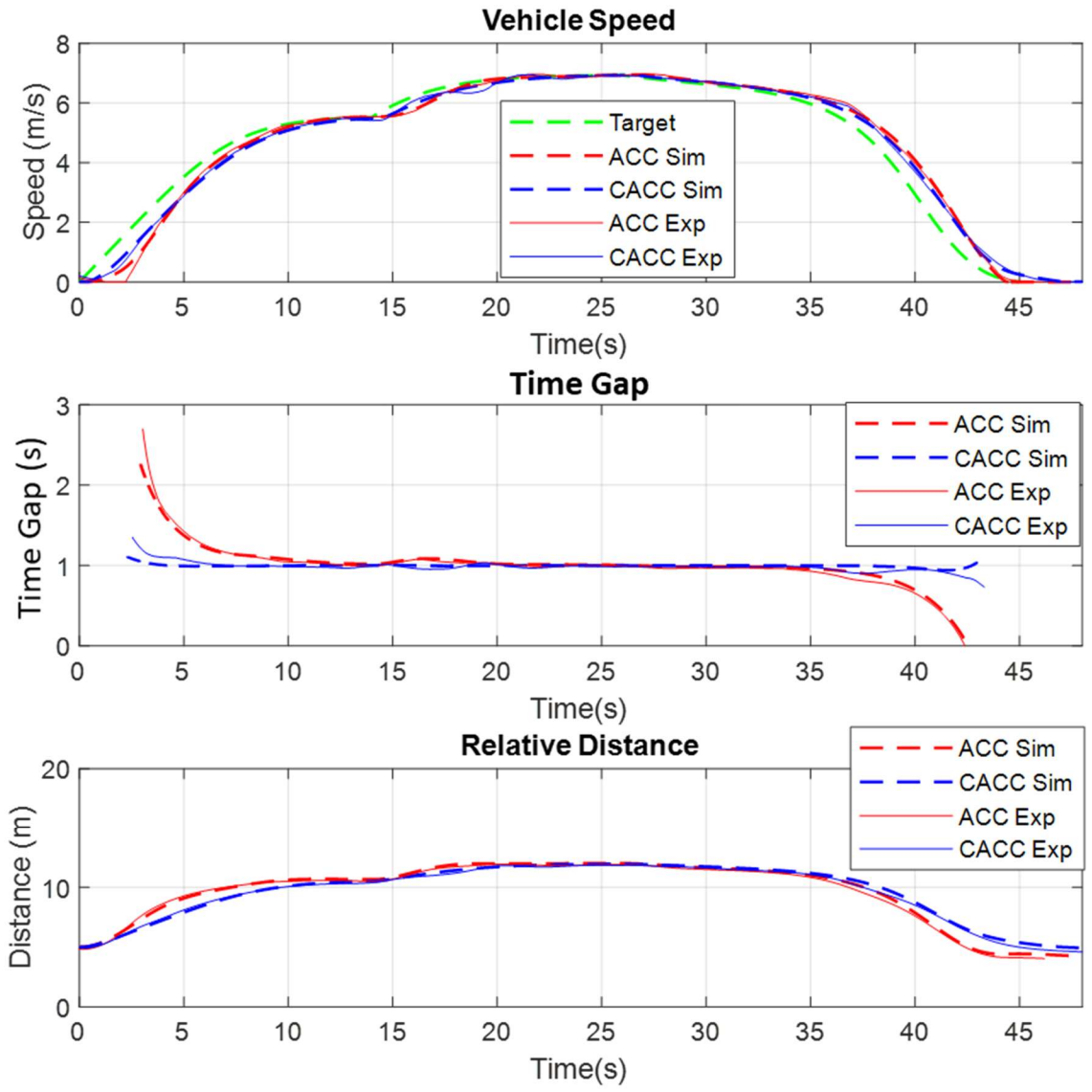


Figure 64 Comparison of ACC and CACC experimental results with simulation results for 1 second desired time gap [96]

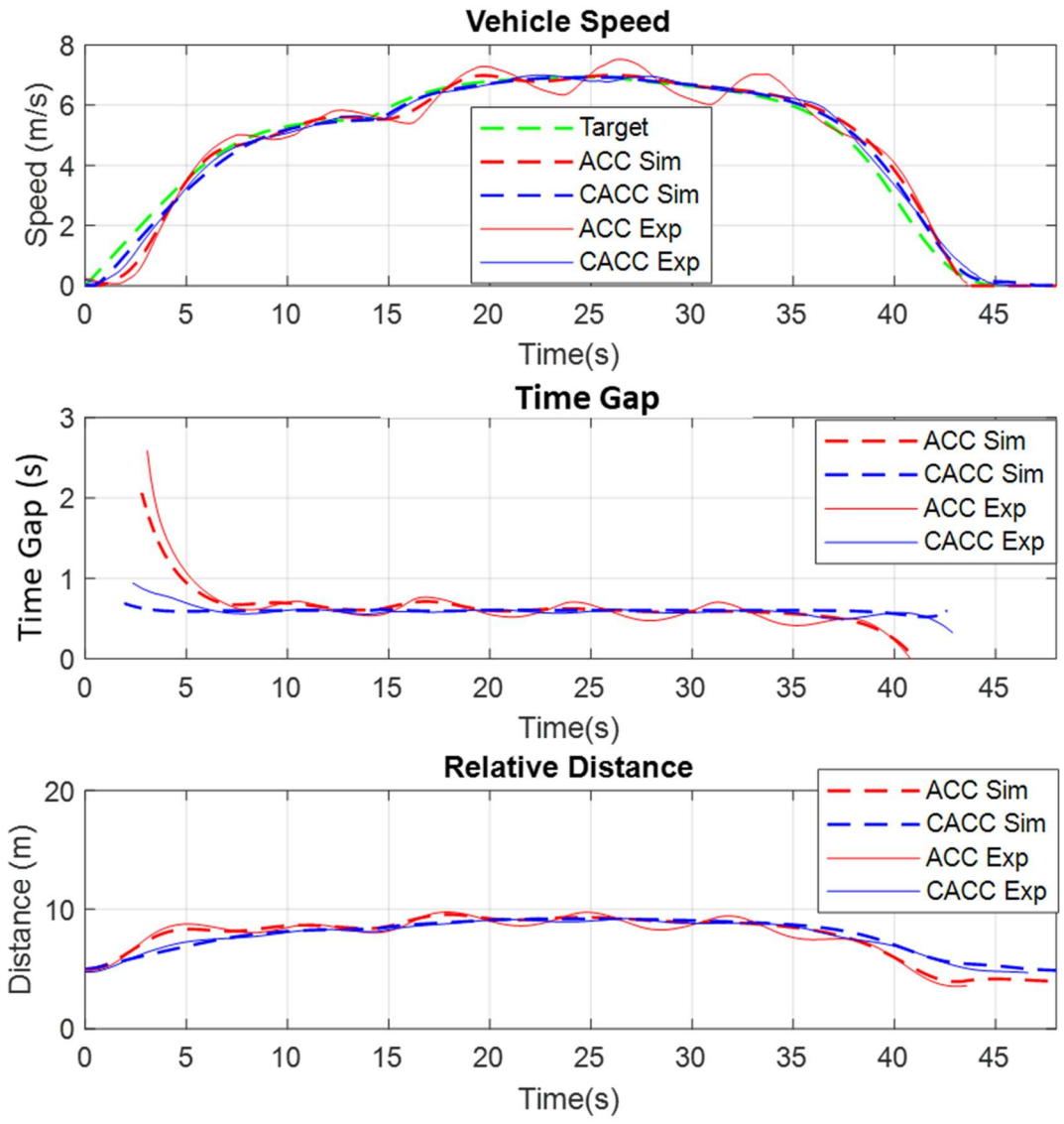


Figure 65 Comparison of ACC and CACC experimental results with simulation results for 0.6 second desired time gap [96]

In the longitudinal controller design part, the design process of ACC and CACC structure of the Automated Driving Lab at The Ohio State University is presented with simulation and experimental results. Both simulation and experimental results show that communication between the target vehicle and ego vehicle in CACC increase the car following performance significantly. This performance improvement will lead to better string stability and capacity increase on the roads.

# Chapter 7. Lateral Controller

This chapter presents lateral controller used for following the desired path automatically in automated driving. One of the main tasks of the highway chauffeur and highway pilot automated driving systems is to keep the vehicle between the lane lines while driving on a pre-defined route. This task can be achieved by using camera and/or GPS to localize the vehicle between the lane lines. However, both sensors have shortcomings in certain scenarios. While the camera does not work when there are no lane lines to be detected, an RTK GPS can localize the vehicle accurately. On the other hand, GPS requires at least 3 satellite connections to be able to localize the vehicle and more satellite connections and real-time over-the-air corrections for lane-level positioning accuracy. If GPS localization fails or is not accurate enough, lane line information from the camera can be used as a backup. In this section, a vision based lane keeping system which is aided by a GPS based path following application is developed to overcome the shortcomings of the GPS and camera sensors. The developed system has a parameter space based robust steering controller which can handle lateral motion control of the vehicle based on path tracking error detected using the GPS or camera sensor. The designed control system works for both low speed and high-speed driving scenarios and is robust to changes in vehicle mass. The results are demonstrated using the validated model of our 2017 Ford Fusion Hybrid research automated driving vehicle in Automated Driving Lab hardware-in-the-loop simulator.

## 7.1 Background

There are six automated driving levels defined in SAE International standard J3016 varying from 0 to 5 [1], where 0 represents the no automation case and 5 represents the fully autonomous driving case with no human intervention. This section will cover a lane keeping application for a vehicle which is already equipped with an adaptive cruise control. This automation level falls within Level 2 which is partial automation where the steering and acceleration of the vehicle are handled by the automated driving system but the driver is still in the loop. In the literature and in production level vehicles, there are many lane-keeping and lane departure warning applications. For instance, Tuncer et. al worked on developing a lane keeping system when the driver is inattentive [99]. In their application, a camera-based lane keeping controller was designed and simulated in a HIL simulator. Kang et. al proposed a solution for estimating the lane positions for short term lane information lost from the camera [100]. Although lane keeping applications and path fallowing applications are thoroughly studied in the literature, the failure of the existing systems would not be acceptable for a fully autonomous vehicle system. Considering the fact that many of these systems are using the camera to detect the lane lines and to localize the vehicle in the lateral direction, the failure of the camera detection would result in failure to keep being within the lane. As highlighted in the work of Yenikaya et.al [101], some of the camera detection failures can be caused by the absence of lane lines, poor lane line quality, shadow on the lane lines, or other vehicle occlusions. The camera may also completely fail to work or fail to communicate with the controller. Today, high accuracy GPS units are also available for accurate localization. For example, the GPS unit used in

our experimental vehicle OXTS xNAV550 has 1.6 m accuracy with single antenna, 0.4 m for DGPS mode and up to 2 cm for RTK mode using a base station. Also, with the use of online RTK correction services and RTK Bridge units it is possible to have RTK corrections without a base station. In the case of using RTK bridge unit, accuracy of the system is around 5cm. While RTK GPS units are very expensive today, as compared to the cameras, they are getting cheaper with the advance of the technology. Therefore, usage of a GPS based lane level path fallowing algorithm is suggested as one of the backup solutions for the camera failure cases. In the proposed solution, the GPS system is also not used solely for the lane keeping applications because it also has its own shortcomings. If the RTK corrections for the GPS are not available or the lane level map of the environment is not available, it is not possible to localize the vehicle within lane level accuracy. Therefore, a combination of the camera and GPS solution is preferred over using them alone by themselves.

## 7.2 Lateral Vehicle Model

In this section, lateral dynamics of the vehicle is modeled using the nonlinear vehicle model (Bicycle Model). In this model, the two front wheels are represented as a single front wheel and similarly, the two rear wheels of the vehicle are represented as a single rear wheel. As our test vehicle is only steerable from the front wheels, the test vehicle is modeled to be only steerable from the front wheel [102]. Forces acting on the vehicle in this model are shown in Figure 66. Lateral forces generated by the front/rear wheels, vehicle center of gravity, distance of the center of gravity from the wheels and the preview distance are represented in the figure as $F_f$/ $F_r$, $CG$, $l_f$/ $l_r$, $l_s$, respectively.

The lateral direction steering controller for the automated lane-keeping application is designed using a linearized version of the nonlinear vehicle model. Linearized state space model of the lateral motion of the vehicle is given in Equation 7.1 where $\beta$ is the vehicle side slip angle at the vehicle center of gravity, $r$ is vehicle yaw rate, $V$ is velocity, $\Delta\psi$ is yaw angle of the vehicle with respect to desired path's tangent, $\rho_{ref}$ is the road curvature, $\delta_f$ is the steering wheel angle and $\mu$ is the friction coefficient of the road. The entries $a_{11}$, $a_{12}$, $a_{21}$, $a_{22}$, $b_{11}$, $b_{12}$ used in Equation 7.1 are given in Equations 7.2-7.7, where $c_r$, $c_f$ are the cornering stiffness of the rear and front wheels, $\tilde{J} = J/\mu$ is the virtual mass moment of inertia and the $\tilde{m} = m/\mu$ is the virtual mass.

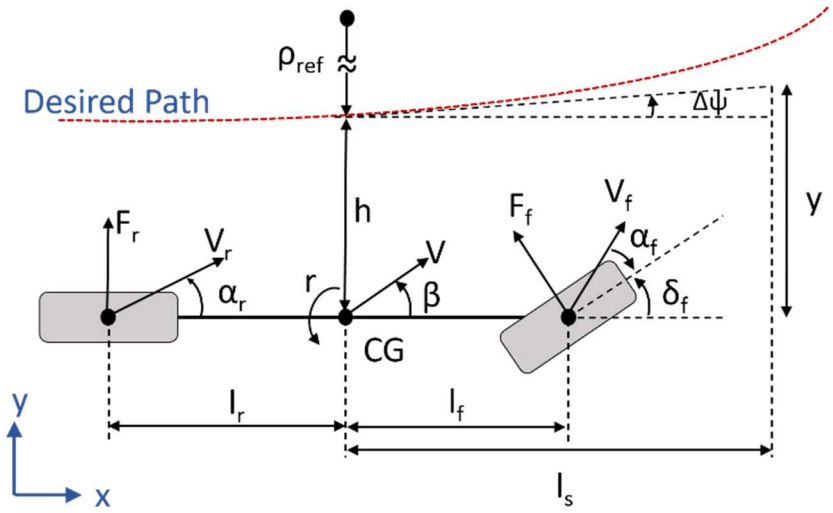


Figure 66 Lateral vehicle model for lane keeping application [103]

$$\begin{bmatrix} \dot{\beta} \\ \dot{r} \\ \Delta\dot{\psi} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 \\ a_{21} & a_{22} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ V & l_s & V & 0 \end{bmatrix} \begin{bmatrix} \beta \\ r \\ \Delta\psi \\ y \end{bmatrix} + \begin{bmatrix} b_{11} & 0 \\ b_{21} & 0 \\ 0 & -V \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_f \\ \rho_{ref} \end{bmatrix} \tag{7.1}$$

$$a_{11} = \frac{-(c_r + c_f)}{\tilde{m}V} \tag{7.2}$$

$$a_{12} = -1 + \frac{-(c_r l_r - c_f l_f)}{\tilde{m} V^2} \tag{7.3}$$

$$a_{21} = \frac{(c_r l_r - c_f l_f)}{\tilde{J}} \tag{7.4}$$

$$a_{22} = \frac{-(c_r {l_r}^2 + c_f {l_f}^2)}{\tilde{J} V^2} \tag{7.5}$$

$$b_{11} = \frac{c_f}{\tilde{m} V} \tag{7.6}$$

$$b_{12} = \frac{c_f l_f}{\tilde{J}}, \tag{7.7}$$

7.3 Lane Detection

Lane lines on the road can be used to localize the ego vehicle on the road Cartesian coordinates. Our automated driving research vehicle for experimental evaluation has a Mobileye camera which can provide the coefficients of the polynomial fit for the lane detections, lane detection availability and quality information. In this section, connected and automated driving HIL simulator with CarSim Real Time with Sensors and Traffic is used as the main development environment. The CarSim soft camera sensor in the HIL simulator provides the lane detection in the form of x, y coordinates (Figure 67). To simulate the real sensor output and extrapolate the lane detection points, two second order curves denoted by $y^l(x)$ and $y^r(x)$ are fitted to the left and right lane detection points coming from the CarSim software (Equations 7.8 and 7.9).

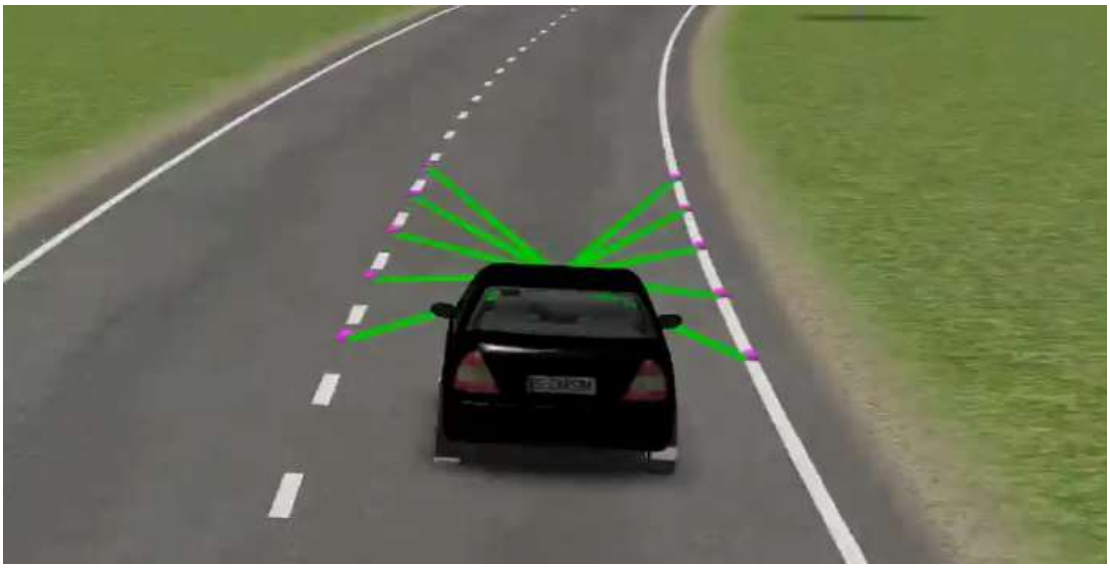

Figure 67 CarSim soft camera sensor visualization [103]

$$y^l(x) = {a_0}^l + {a_1}^l * x + {a_2}^l * x^2 \quad (7.8)$$

$$y^r(x) = {a_0}^r + {a_1}^r * x + {a_2}^r * x^2 \quad (7.9)$$

By inserting a longitudinal distance *x* into the Equations 7.8 and 7.9, one can calculate the lateral distance of the vehicle from the right and left lane lines at that longitudinal distance.

7.4 Path Generation

For generating the lane level path following map/path, the method presented in [102], [104] is used. This method requires one to drive the car at a constant speed at the center of the road and collect accurate GPS data points. These GPS waypoints can also be automatically extracted from a realistic map. Collected GPS waypoints are divided into a predetermined number of polynomial segments to capture the different characteristics of the road. These segments are represented as 3rd order parametric polynomials of a distance parameter λ, where λ changes between *i-1* to *i*, according to the number of the segment used. These polynomials are given below as:

$$X_i(\lambda) = a_{xi}\lambda^3 + b_{xi}\lambda^2 + c_{xi}\lambda + d_{xi} \tag{7.10}$$

$$Y_i(\lambda) = a_{yi}\lambda^3 + b_{yi}\lambda^2 + c_{yi}\lambda + d_{yi} \tag{7.11}$$

where $X_i$ and $Y_i$ are the path centerline coordinates for the $i$th segment. Since the polynomials fitted to two consecutive segments need to have continuity at their intersection, the polynomials can be fitted to the GPS waypoints using the constrained least squares method. However, to solve the constrained least squares problem, first, the unconstrained problem needs to be solved. The unconstrained problem can be formed in matrix form as shown below.

$$x_{data} = \Lambda n_{x,uncs} \tag{7.12}$$

$$y_{data} = \Lambda n_{y,uncs} \tag{7.13}$$

$$\Lambda = \begin{bmatrix} \bar{\lambda}^3 & \bar{\lambda}^2 & \bar{\lambda} & 1 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & \bar{\lambda}^3 & \bar{\lambda}^2 & \bar{\lambda} & 1 & \ldots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{7.14}$$

$$n_{x,uncs} = [a_{x1} \quad b_{x1} \quad c_{x1} \quad d_{x1} \quad \ldots \quad a_{xm} \quad b_{xm} \quad c_{xm} \quad d_{xm}]^T \tag{7.15}$$

$$n_{y,uncs} = [a_{y1} \quad b_{y1} \quad c_{y1} \quad d_{y1} \quad \ldots \quad a_{ym} \quad b_{ym} \quad c_{ym} \quad d_{ym}]^T \tag{7.16}$$

In the $\Lambda$ matrix, $\bar{\lambda}$ represents the entire $\lambda$ vector which ranges from *i-1* to *i*, where *i* is the number of the segment. The number of elements in $\bar{\lambda}$ is equal to the number of the data points in the $i^{\text{th}}$ segment. For the given equations, the solution of the unconstrained least square problem is given in Equations 7.17 and 7.18 as

$$n_{x,uncs} = (\Lambda^T\Lambda)^{-1}\,\Lambda^T x_{data} \tag{7.17}$$

$$n_{y,uncs} = (\Lambda^T\Lambda)^{-1}\,\Lambda^T y_{data} \tag{7.18}$$

To sustain the continuity and smoothness (continuity of the first derivative) at the segment boundaries, the constraints given below are defined.

$$X_i(i) = X_{i+1}(i) \tag{7.19}$$

$$Y_i(i) = Y_{i+1}(i) \tag{7.20}$$

$$\frac{dX_i(i)}{d\lambda} = \frac{dX_{i+1}(i)}{d\lambda} \tag{7.21}$$

$$\frac{dY_i(i)}{d\lambda} = \frac{dY_{i+1}(i)}{d\lambda} \tag{7.22}$$

$$\frac{d^2X_i(i)}{d\lambda} = \frac{d^2X_{i+1}(i)}{d\lambda} \tag{7.23}$$

$$\frac{d^2Y_i(i)}{d\lambda} = \frac{d^2Y_{i+1}(i)}{d\lambda} \tag{7.24}$$

From the equations 7.10 and 7.11, these constraints can be rewritten as shown in Equations 7.25-7.30.

$$a_{xi}i^3 + b_{xi}i^2 + c_{xi}i + d_{xi} = a_{xi+1}i^3 + b_{xi+1}i^2 + c_{xi+1}i + d_{xi} \tag{7.25}$$

$$a_{yi}i^3 + b_{yi}i^2 + c_{yi}i + d_{yi} = a_{yi+1}i^3 + b_{yi+1}i^2 + c_{yi+1}i + d_{yi} \tag{7.26}$$

$$3a_{xi}i^2 + 2b_{xi}i + c_{xi} = 3a_{xi+1}i^2 + 2b_{xi+1}i + c_{xi+1} \tag{7.27}$$

$$3a_{yi}i^2 + 2b_{yi}i + c_{yi} = 3a_{yi+1}i^2 + 2b_{yi+1}i + c_{yi+1} \tag{7.28}$$

$$6a_{xi}i + 2b_{xi} = 6a_{xi+1}i + 2b_{xi+1} \tag{7.29}$$

$$6a_{yi}i + 2b_{yi} = 6a_{yi+1}i + 2b_{yi+1} \tag{7.30}$$

These defined constraint equations are used in matrix form to convert the unconstrained problem into the constrained problem. These equations are combined into a matrix form as shown in Equations 7.31 and 7.32.

$$Fn_{x,cs} = 0 \tag{7.31}$$

$$Fn_{y,cs} = 0 \tag{7.32}$$

Finally, the solution of the constrained problem is given in Equations 7.33 and 7.34.

$$n_{x,cs} = n_{x,uncs} - (\wedge^T \wedge)^{-1} F^T [F(\wedge^T \wedge)^{-1} F^T]^{-1} F_{n,xuncs} \tag{7.33}$$

$$n_{y,cs} = n_{y,uncs} - (\wedge^T \wedge)^{-1} F^T [F(\wedge^T \wedge)^{-1} F^T]^{-1} F_{n,yuncs} \quad (7.34)$$

7.5 Lateral Deviation Calculation

The lateral controller takes the lateral deviation at a pre-defined preview distance as input and calculates the corresponding steering angle. As mentioned earlier, two different methods are used to calculate the lateral deviation in this application. The first method uses lane detections acquired from the camera and the second method uses the GPS localization and map-based waypoint information.

7.6 Lateral Deviation from the Lane Line detections:

The polynomials representing the lane lines must be parallel to one another as the lane lines are parallel to each other in a real road. Knowing this, the centerline of the road can be represented with the polynomial below in vehicle coordinates.

$$y^c(x) = a_0{}^c + a_1{}^c * x + a_2{}^c * x^2 \quad (7.35)$$

where coefficients of the polynomial which represents the centerline are given below. Here the superscript "*c*" indicates that the polynomial is a fit for the centerline of the road, and the coefficients of the polynomial are given in Equations 7.36-7.38.

$$a_0{}^c = (a_0{}^l + a_0{}^r)/2 \quad (7.36)$$

$$a_1{}^c = (a_1{}^l + a_1{}^r)/2 \quad (7.37)$$

$$a_2{}^c = (a_2{}^l + a_2{}^r)/2 \quad (7.38)$$

Inserting the preview distance $l_s$ into Equation 7.35 gives the lateral distance of the vehicle at the preview distance.

7.7 Lateral Deviation calculation from the map and GPS measurements:

When the lane detections are not available or reliable, the lateral deviation at the preview distance is calculated using the current lateral deviation and the yaw angle error with respect to the generated map. Based on the geometry shown in Figure 66 the lateral deviation at the preview distance $l_s$ can be calculated as

$$y = \mathrm{h} + \mathrm{l}_s \tan(\Delta\Psi) \tag{7.39}$$

where $h$ is the lateral deviation from the desired path at the vehicle center of gravity, $l_s$ is the preview distance and the $\Delta\Psi$ is the yaw angle of the vehicle with respect to the desired path.

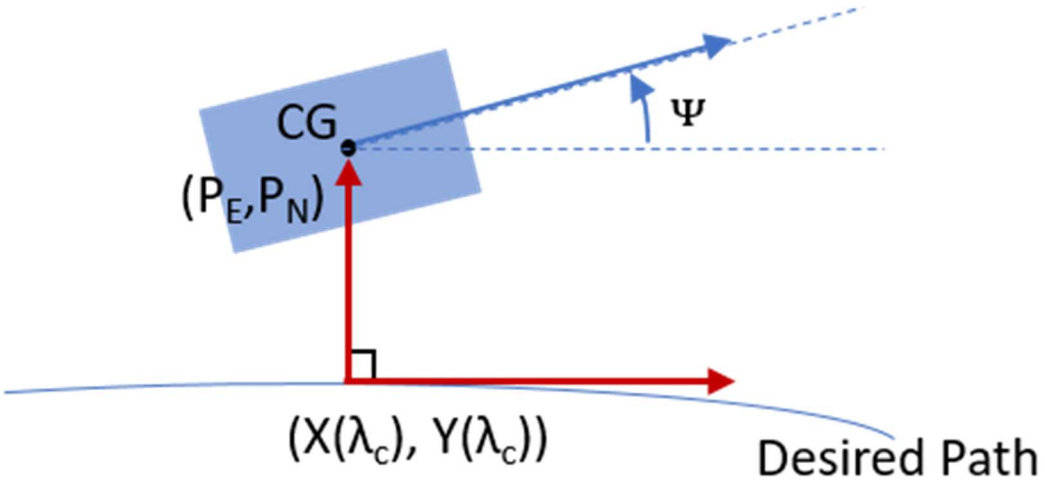


Figure 68 Position and orientation of the vehicle with respect to the desired path [103].

First, the lateral deviation of the vehicle from the generated map is calculated. Assuming the radius of the curvature is much larger than the lateral deviation of the vehicle, the shortest distance to the path can be calculated by finding the perpendicular vector to the path from the vehicle center of gravity. This means that the tangent vector of the path will

be orthogonal to the shortest vector between the generated path and the vehicle center of gravity as shown in Figure 68. Here the center of the gravity of the vehicle is represented using East and North map coordinates $P_E$ and $P_N$ respectively. Using the fact that cross product of two orthogonal vectors is zero, the solution of Equation 7.40 for $\lambda_c$ gives the closest segment position to the vehicle. One can evaluate the $x$, $y$ coordinates of the closest point on the path and the distance of the vehicle from the path by inserting $\lambda_c$ into the Equation 7.41.

$$(X(\lambda) - P_E, Y(\lambda) - P_N)\left(\dot{X}(\lambda) - \dot{Y}(\lambda)\right) = 0 \tag{7.40}$$

$$h = \rho\sqrt{(X(\lambda_c) - P_E)^2 + (Y(\lambda_c) - P_N)^2} \tag{7.41}$$

where

$$\rho = sgn(\vec{U}(3)) \tag{7.42}$$

$$\vec{U} = (X(\lambda_c) - P_E, Y(\lambda_c) - P_N, 0)\mathrm{x}\left(\dot{X}(\lambda), \dot{Y}(\lambda), 0\right) \tag{7.43}$$

If the third component of the cross product of the path tangent and distance vector is negative, it shows that the vehicle is in the inner side of the desired path and vice-versa. After finding $h$, $\Delta\Psi$ is calculated by subtracting the slope of the road at the closest point on the reference path from the yaw angle of the vehicle. Calculation of $\Delta\Psi$ can be seen in Equation 7.44.

$$\Delta\Psi = \Psi - \frac{\dot{Y}(\lambda_c)}{\dot{X}(\lambda_c)} \tag{7.44}$$

Finally, the lateral deviation at the preview distance can be calculated by inserting $h$, $l_s$ and $\Delta\Psi$ into Equation 7.39.

## 7.8 Lateral Controller Design

The parameter space based design approach given is used to design the PD controller for the robust lane-keeping controller for the overall system shown in Figure 69. The input and outputs of the system can be listed as the steering command and the lateral deviation at the preview distance, respectively. The test vehicle modeled in the state space model of Equation 7.1 has the numerical parameter values $J$=3,728 kgm$^2$, $C_f$=1.2e5 N/rad, $C_r$=1.9e5 N/rad, $l_r$= 1.5453 m and $l_f$=1.30 m where the weight of the vehicle varies between 1,700 kg and 2,000kg.

Since the vehicle operates in different load and speed conditions, the controller is designed to be able to work under these different operating conditions as is shown in Figure 70 as an uncertainty box. Also, the preview distance for higher speeds is increased as $l_s$=max($k_s v$,$l_{s\max}$) where $v$ is vehicle speed, $k_s$ is a proportional factor and $l_{smax}$ is the upper bound on the preview length. In this chapter, $k_s$ is adjusted such that preview distance changes linearly between 4 m to 7 m for the chosen operating speed range 5 m/s to 30 m/s.

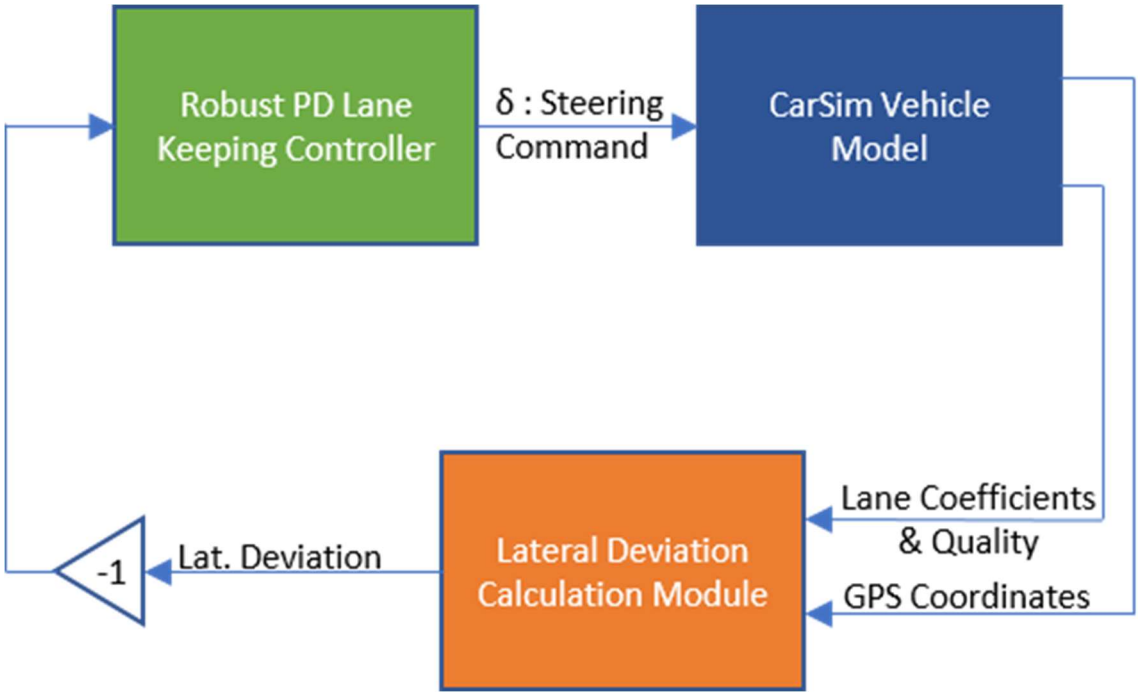


Figure 69 Lateral controller system block diagram [103]

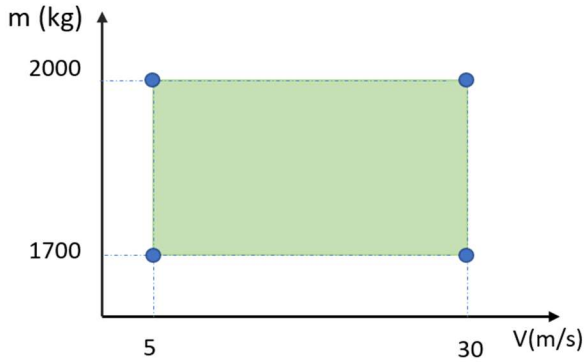


Figure 70 Lateral dynamics uncertainty box [103]

As a D-stability requirement, desired settling time, damping ratio and maximum bandwidth are chosen as 0.5 seconds, 0.7 and 19 rad/sec respectively [105]. PD controller coefficients ($K_p$ and $K_d$) are chosen as free parameters to find a solution region using the parameter space approach. A D-stability solution region is constructed for each corner of the

uncertainty box in Figure 70 and they are overlaid on top of each other to find the overall solution region as shown in Figure 71. In this figure blue, green, red, cyan, magenta colored lines show Settling Time Constraint Complex Root Boundary (CRB), Damping Constraint CRB, Bandwidth Constraint CRB, Bandwidth Constraint Real Root Boundary (RRB), Settling Time Constraint RRB, respectively. Since blue line is covered by the magenta, it is not clearly visible. Calculation of these boundaries are shown in detail in [105]. By choosing a point in this solution region, one set of $K_p$ and $K_d$ values for the PD controller are chosen as shown in the right plot in Figure 71.

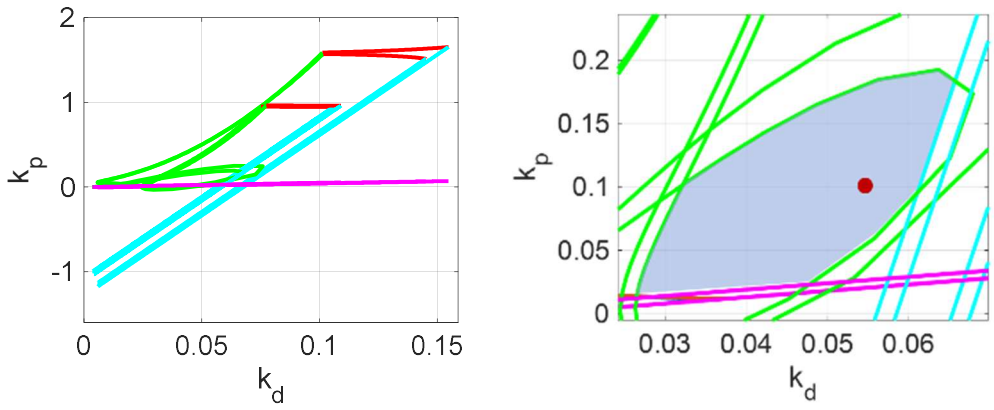


Figure 71 Left: Solution regions for the corners of the uncertainty box is plotted on top of each other. Right: The zoomed version of the left figure where intersection of the solution regions is highlighted with a gray fill and chosen solution point is shown with a red dot [103]

To be considered as a D-stable system, the poles of the system should lie within the D Stable region where it is defined by the desired settling time, the desired minimum damping ratio and the desired maximum bandwidth, all given earlier. As it can be seen from Figure

72, all of the dominant poles of the system which are marked as “x” lie in the D-Stable region for the chosen $K_p$ and $K_d$ coefficients of the PD controller.

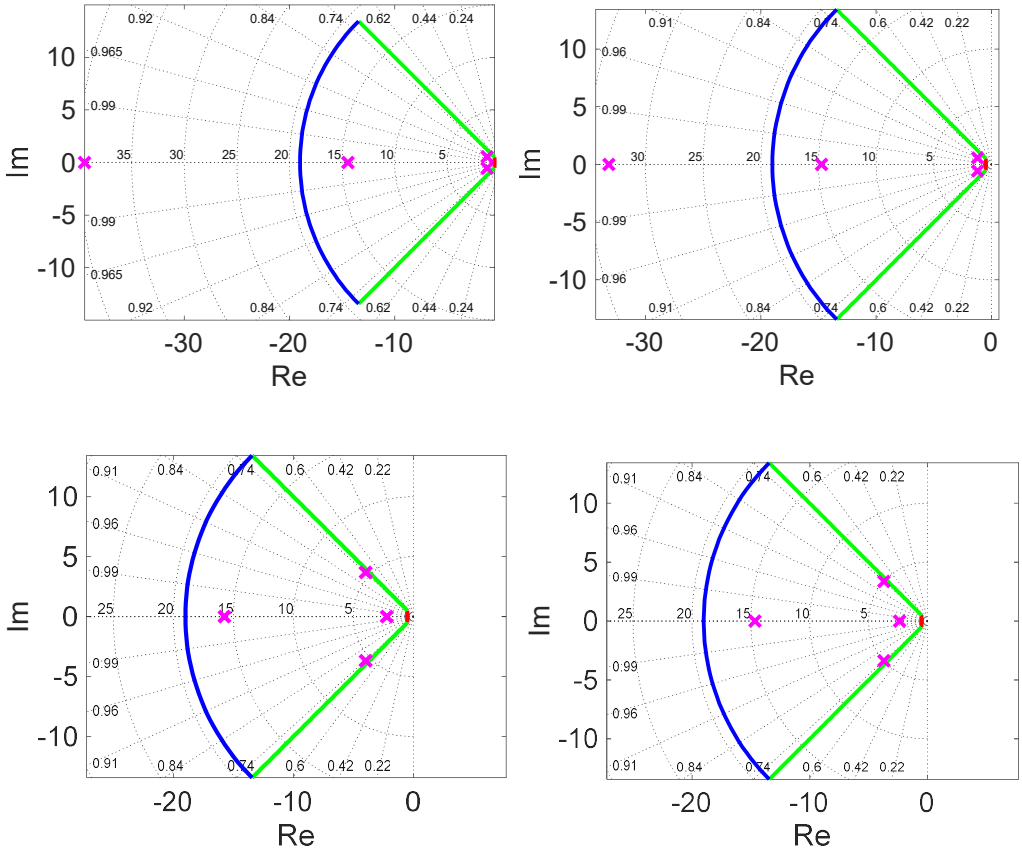


Figure 72 D-Stable region and the system pole positions in complex plane for the chosen $K_p$ and $K_d$. Top Left: 5m/s, 1700 kg $l_s$ = 4m, Top Right: 5m/s, 2000 kg $l_s$ = 4m, Bottom Left: 30m/s, 1700 kg $ls$ = 7m, Bottom Right: 30m/s, 2000 kg $l_s$ = 7m [103]

## 7.9 Simulation Results

To evaluate the performance of the system a designed lane keeping controller is tested in the HIL simulation environment which was described in Chapter 4. As a test track, a simple model of the high-speed test track in the Transportation Research Center proving ground is constructed in CarSim using its pre-recorded GPS waypoints. The top view of the TRC

testing track can be seen in the Google Maps image in Figure 73 which has two curved section between 1000m-4200m and 7200m-10100m. Since the GPS based path following localization is used as a backup solution, camera based lane keeping system is considered as the active system. In the experimental setup, the vehicle is equipped with a Mobileye camera where it can output the quality of the lane line detections. So, in the experiments the mode switching between the camera and GPS will be done based on this lane line detection quality information by the Lateral Calculation Module shown in Figure 69. If there are no reliable lane detections, the vehicle is going to switch to GPS based path following mode. Based on the road conditions, the lane quality of the system can fail anytime. To simulate the cases where the camera detection fails, the system switches to the GPS based lane keeping mode for pre-defined distance intervals (1500-1700, 5800-6000, and 9800- 10000 meters). In the first and third sections, the vehicle is travelling in the curved parts of the test track while it travels at the straight part of the road in the second interval. The designed system is tested for the different speed and vehicle mass conditions which are defined as the corners of the uncertainty box shwon in Figure 70.

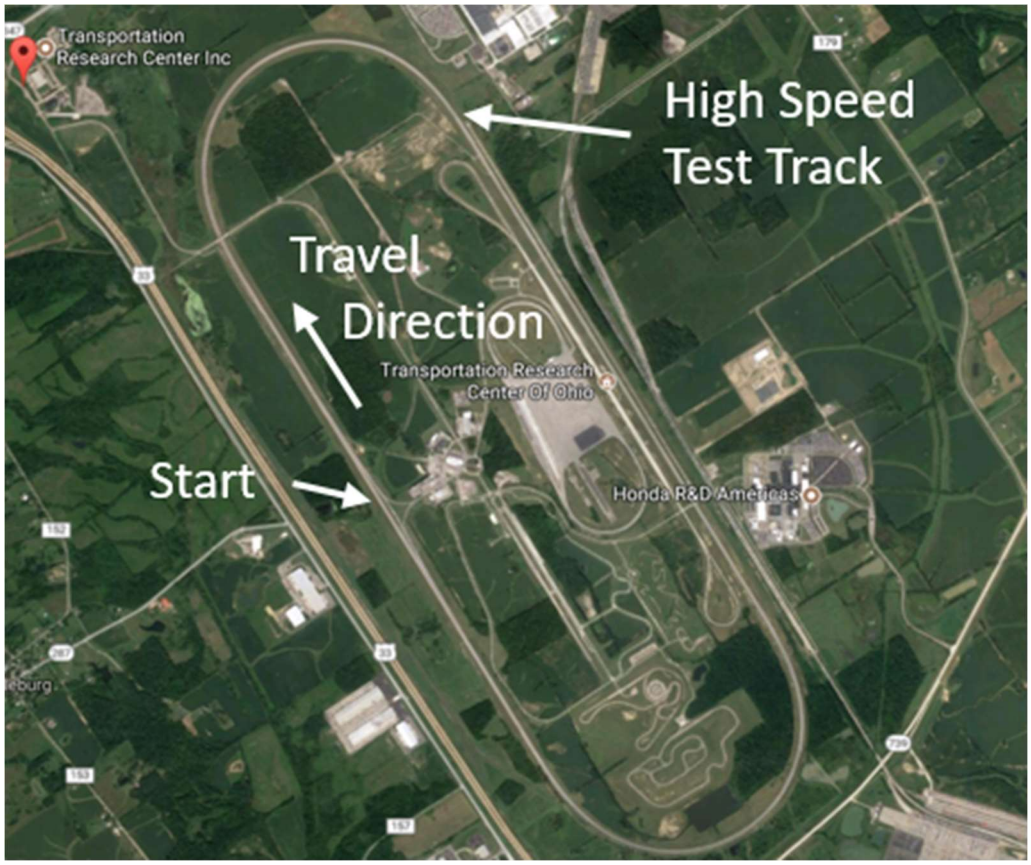


Figure 73 Top view of the TRC test tracks from Google Maps [103]

The simulation results for the designed system are shown in Figure 74.

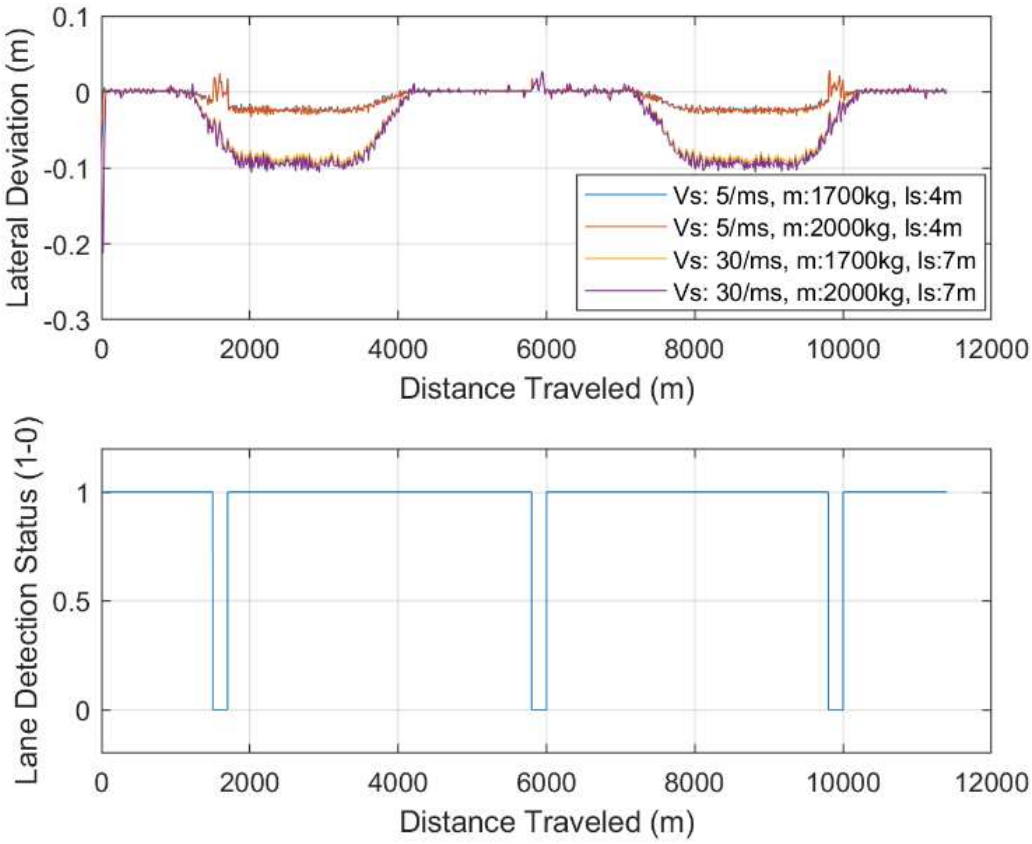


Figure 74 Simulation Results for the designed lane keeping system. Top: lateral deviation vs distance traveled for different operating conditions. Bottom: Availability of lane detection vs distance traveled [103]

When the lateral deviation graph (Figure 74) is analyzed, it can be seen that the designed system still keeps the vehicle in the lane even when the lane detection status goes to zero. Although both the vision based and the GPS based solutions have a higher error for the curved sections of the road, which increases for high speeds, this error is less than 12 cm.

## 7.10 Summary

This chapter presented the use of a GPS based lane keeping/path following application as a backup to the camera based lane keeping application. By using these two methods together, lane level control for the vehicle is sustained even when one of the sensor inputs is not available or is not reliable. This happens for instance when lane markings are missing or are very vague or not observable due to weather conditions in certain parts of the road. As it can be seen from the simulation results in Figure 74, both designed systems keep the vehicle in the lane accurately. Although increasing the operating speed increases the lateral deviation, especially for the curved parts, the deviation is still under 12 cm for the highest vehicle speed of 30 m/s. Also, the system works in different operating conditions where the vehicle weight and the speed are varied over the uncertainty box.

# Chapter 8. Conclusions

Connected and autonomous vehicle applications are re-shaping the transportation industry. With these applications automotive industry aims to increase the safety of the occupants and create more efficient and comfortable ride experiences. Considering safety, efficiency and comfort are the active problems of the intelligent transportation field. Motivated by these development some of the enabling technologies for connected and autonomous vehicle features were developed in this dissertation.

One of the least studied areas in connected vehicles is cooperative perception. In the Cooperative Perception (CP) system, connected agents not only share their own information, but also the information of agents surrounding them. In this dissertation, a cooperative perception architecture has proposed and implemented in the simulation environment. The architecture consists of JPDA object tracking algorithm and cooperative perception message generation modules. After introducing the cooperative perception architecture, two different use case scenarios were introduced. In the first use case scenario, the situational awareness of the ego vehicle is increased by utilizing CP messages broadcasted by a remote vehicle. In the presented scenario, CP messages broadcast the location and speed information of a pedestrian which is normally occluded by the remote vehicle. The second application of cooperative perception is demonstrated in a smart intersection environment. The smart intersection, alongside the SPaT and MAP messages, broadcasts the location and speed information of the approaching vehicles to the intersection. By utilizing smart infrastructure cooperative perception messages, a representative spatial and temporal traffic preview model is constructed. Then, a rule-based

algorithm and a Deep Deterministic Policy Gradient (DDPG) reinforcement learning agent model are developed to control vehicle speed when approaching the signalized intersections (GLOSA- with traffic preview). It is shown that the designed algorithms improve the fuel efficiency and comfort of the passengers.

As the number of the perception sensors increases, the situational awareness of vehicles increases. However, in multi-sensor systems same targets can be reported by different sensors which have overlapping field of view. In this case, the threat assessment modules in the car needs to identify which reported tracks from different sources originate from the same target. This is called data association problem. Since the connected vehicle applications are relatively new, the data association task between the on board sensors and communication modems have not been researched extensively. Therefore, in this dissertation, a Mahalanobis distance based track to track data association algorithm was used to solve the data association problem between camera and communication sensors. For this purpose, associable parameters were first identified for the CAV applications. By analyzing the collected data from camera and DSRC sensors, it was found out that the location, relative heading, and relative speed measurements from these two sensors are comparable, and they can be used for the data association tasks. While the location measurements were the main parameters used in the implemented algorithm, speed and relative heading parameters were also used as another gating stage to prevent any false data association assessment. Later, the developed V2V and camera track to track association algorithm was experimentally tested for two different scenarios. It was shown that the data

association algorithm is effective for even curved roads and intersections. As a future work, it is recommended to conduct more experiments with a higher number of vehicles in more complex scenarios before the deployment of the algorithm. As another improvement for the developed method, one can drop the buffering stage for V2V tracks and use the path history information, which is available in the BSM message set.

Finally, as part of this work, lateral and longitudinal controllers were developed as part of the development of autonomous driving functionalities. For the longitudinal controller both ACC and CACC architectures were developed for the Automated Driving Lab autonomous vehicle development platform. The designed CACC architecture is a PD controller with feed forward of preceding vehicle acceleration which offers a string stable solution to control a platoon of vehicles. It is shown that communication between the target vehicle and ego vehicle in CACC improves the car following performance significantly. This performance improvement leads to better string stability and capacity increase. For the lateral controller, a GPS based path following application was developed as a backup to the camera based lane centering application. By using both path following and lane centering applications together, lateral control for the vehicle was sustained even when one of the sensor (GPS or camera) inputs was not available or was not reliable.

Bibliography


[1] Society for Automotive Engineers, “SAE J3016 - Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles,” 2021. [Online]. Available: https://www.sae.org/standards/content/j3016_202104/.

[2] “J3216A: Taxonomy and Definitions for Terms Related to Cooperative Driving Automation for On-Road Motor Vehicles - SAE International.” https://www.sae.org/standards/content/j3216_202107/ (accessed Dec. 01, 2021).

[3] M. R. Cantas and L. Guvenc, “Customized Co-Simulation Environment for Autonomous Driving Algorithm Development and Evaluation,” *SAE Tech. Pap.*, no. 2021, Apr. 2021, doi: 10.4271/2021-01-0111.

[4] F. Rosique, P. J. Navarro, C. Fernández, and A. Padilla, “A Systematic Review of Perception System and Simulators for Autonomous Vehicles Research,” pp. 1–29, 2019, doi: 10.3390/s19030648.

[5] H. J. Günther, O. Trauer, and L. Wolf, “The potential of collective perception in vehicular ad-hoc networks,” *2015 14th Int. Conf. ITS Telecommun. ITST 2015*, pp. 1–5, 2016, doi: 10.1109/ITST.2015.7377190.

[6] R. Miucic, A. Sheikh, Z. Medenica, and R. Kunde, “V2X Applications Using Collaborative Perception,” *IEEE Veh. Technol. Conf.*, vol. 2018-Augus, pp. 1–6, 2019, doi: 10.1109/VTCFall.2018.8690818.

[7] B. Mourllion, A. Lambert, D. Gruyer, and D. Aubert, “Collaborative perception for collision avoidance,” *Conf. Proceeding - IEEE Int. Conf. Networking, Sens. Control*, vol. 2, pp. 880–885, 2004.

[8] ETSI, “Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Analysis of the Collective Perception Service (CPS),” *Draft TR 103 562 V0.0.16*, vol. 1, pp. 1–119, 2019.

[9] Advanced Applications Technical Committee, “V2X Sensor-Sharing for Cooperative & Automated Driving J3224.” https://www.sae.org/standards/content/j3224/ (accessed Nov. 13, 2021).

[10] V2x Vehicular Applications Technical Committee, “J2945/8 (WIP) Cooperative Perception System - SAE International.” https://www.sae.org/standards/content/j2945/8/ (accessed Nov. 13, 2021).

[11] M. Shan, K. Narula, Y. F. Wong, S. Worrall, M. Khan, P. Alexander, and E. Nebot, “Demonstrations of cooperative perception: Safety and robustness in connected and automated vehicle operations,” *Sensors (Switzerland)*, vol. 21, no. 1, pp. 1–31, 2021, doi: 10.3390/s21010200.

[12] H. J. Günther, B. Mennenga, O. Trauer, R. Riebl, and L. Wolf, “Realizing collective perception in a vehicle,” *IEEE Veh. Netw. Conf. VNC*, vol. 0, 2016, doi:

10.1109/VNC.2016.7835930.
[13] A. Dosovitskiy, G. Ros, F. Codevilla, A. López, and V. Koltun, “CARLA: An Open Urban Driving Simulator.” Accessed: Oct. 15, 2018. [Online]. Available: http://proceedings.mlr.press/v78/dosovitskiy17a/dosovitskiy17a.pdf.
[14] “darkscyla/MATLAB-Carla-Interface: Interfacing Carla with MATLAB using Python and ROS.” https://github.com/darkscyla/MATLAB-Carla-Interface (accessed Sep. 01, 2021).
[15] O. Kavas-Torris, M. R. Cantas, K. Meneses Cime, B. Aksun Guvenc, and L. Guvenc, “The Effects of Varying Penetration Rates of L4-L5 Autonomous Vehicles on Fuel Efficiency and Mobility of Traffic Networks,” *SAE Tech. Pap.*, vol. 2020-April, no. April, Apr. 2020, doi: 10.4271/2020-01-0137.
[16] M. R. Cantas, S. Fan, O. Kavas, S. Tamilarasan, L. Guvenc, S. Yoo, J. H. Lee, B. Lee, and J. Ha, “Development of virtual fuel economy trend evaluation process,” *SAE Tech. Pap.*, vol. 2019-April, no. April, Apr. 2019, doi: 10.4271/2019-01-0510.
[17] O. Kavas-Torris, N. Lackey, and L. Guvenc, “Simulating the Effect of Autonomous Vehicles on Roadway Mobility in a Microscopic Traffic Simulator,” *Int. J. Automot. Technol. 2021 223*, vol. 22, no. 3, pp. 713–733, May 2021, doi: 10.1007/S12239-021-0066-7.
[18] “Comparison between PhysXVehicle and CarSim vehicles in CARLA Simulator - YouTube.” https://www.youtube.com/watch?v=fvYcm1o8zLg&ab_channel=Ricozero (accessed Sep. 01, 2021).
[19] “OS0 Ultra-wide field-of-view lidar sensor for autonomous vehicles and robotics | Ouster.” https://ouster.com/products/scanning-lidar/os0-sensor/ (accessed Nov. 29, 2021).
[20] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, “Pointpillars: Fast encoders for object detection from point clouds,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2019-June, pp. 12689–12697, 2019, doi: 10.1109/CVPR.2019.01298.
[21] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, “Frustum PointNets for 3D Object Detection from RGB-D Data,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 918–927, 2018, doi: 10.1109/CVPR.2018.00102.
[22] “Lidar Obstacle Detection.” https://github.com/mrcantas/SFND_Lidar_Obstacle_Detection/blob/master/README.md (accessed Nov. 29, 2021).
[23] “Track Vehicles Using Lidar: From Point Cloud to Track List - MATLAB & Simulink.” https://www.mathworks.com/help/fusion/ug/track-vehicles-using-lidar.html?searchHighlight=Track Vehicles Using Lidar%3A From Point Cloud to Track List&s_tid=srchtitle (accessed Sep. 02, 2021).
[24] “Random sample consensus - Wikipedia.” https://en.wikipedia.org/wiki/Random_sample_consensus (accessed Nov. 29, 2021).
[25] “Udacity Sensor Fusion Engineer Nanodegree.”

https://www.udacity.com/course/sensor-fusion-engineer-nanodegree--nd313 (accessed Dec. 30, 2021).
[26] "Euclidean Cluster Extraction — Point Cloud Library 1.12.0-dev documentation." https://pointclouds.org/documentation/tutorials/cluster_extraction.html (accessed Nov. 29, 2021).
[27] "k-d tree - Wikipedia." https://en.wikipedia.org/wiki/K-d_tree (accessed Nov. 29, 2021).
[28] "Joint probabilistic data association tracker - MATLAB." https://www.mathworks.com/help/fusion/ref/trackerjpda-system-object.html#sysobj_tracker_jpda_sep_mw_ea30d5b5-17db-433d-8fc2-9547a1b5ff85_head (accessed Nov. 29, 2021).
[29] "deg2utm - File Exchange - MATLAB Central." https://www.mathworks.com/matlabcentral/fileexchange/10915-deg2utm (accessed Sep. 02, 2021).
[30] J. M. L. Domínguez and T. J. M. Sanguino, "Review on V2X, I2X, and P2X Communications and Their Applications: A Comprehensive Analysis over Time," no. Figure 1, pp. 1–29, 2019.
[31] A. Houenou, P. Bonnifait, V. V. Cherfaoui, A. T. A. Method, A. Houenou, P. Bonnifait, V. V. Cherfaoui, and J. Boissou, "A Track-To-Track Association Method for Automotive Perception Systems," 2012.
[32] A. H. Sakr, G. Bansal, and M. View, "Cooperative Localization via DSRC and Multi-Sensor Multi-Target Track Association," pp. 1–6, 2016.
[33] S. H. Bae and K. J. Yoon, "Robust online multi-object tracking based on tracklet confidence and online discriminative appearance learning," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 1218–1225, 2014, doi: 10.1109/CVPR.2014.159.
[34] S. Kim, S. Kwak, J. Feyereisl, and B. Han, "Online Multi-target Tracking by Large Margin Structured Learning," in *Computer Vision -- ACCV 2012*, 2013, pp. 98–111.
[35] B. K. Habtemariam, R. Tharmarasa, T. Kirubarajan, D. Grimmett, and C. Wakayama, "Multiple detection probabilistic data association filter for multistatic target tracking," *Fusion 2011 - 14th Int. Conf. Inf. Fusion*, pp. 1–6, 2011.
[36] Y. Bar-Shalom, T. Kirubarajan, and C. Gokberk, "Tracking with classification-aided multiframe data association," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 41, no. 3, pp. 868–878, 2005, doi: 10.1109/TAES.2005.1541436.
[37] Y. Bar-Shalom, T. Kirubarajan, and X. Lin, "Probabilistic data association techniques for target tracking with applications to sonar, radar and EO sensors," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 20, no. 8 II, pp. 37–54, Aug. 2005, doi: 10.1109/MAES.2005.1499275.
[38] D. Mušicki and R. Evans, "Joint Integrated Probabilistic Data Association: JIPDA," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 40, no. 3, pp. 1093–1099, Jul. 2004, doi: 10.1109/TAES.2004.1337482.
[39] C. Kim, F. Li, A. Ciptadi, and J. M. Rehg, "Multiple Hypothesis Tracking Revisited," *2015 IEEE Int. Conf. Comput. Vis.*, 2015, doi:

10.1109/ICCV.2015.533.
[40] J. Vermaak, S. J. Godsill, and P. Pérez, “Monte Carlo filtering for multi-target tracking and data association,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 41, no. 1, pp. 309–332, 2005, doi: 10.1109/TAES.2005.1413764.
[41] S. Oh, S. Russell, and S. Sastry, “Markov chain Monte Carlo data association for multi-target tracking,” *IEEE Trans. Automat. Contr.*, vol. 54, no. 3, pp. 481–497, 2009, doi: 10.1109/TAC.2009.2012975.
[42] A. Rangesh and M. M. Trivedi, “No Blind Spots: Full-Surround Multi-Object Tracking for Autonomous Vehicles using Cameras & LiDARs,” *IEEE Trans. Intell. Veh.*, pp. 1–1, 2019, doi: 10.1109/tiv.2019.2938110.
[43] C. Yi, K. Zhang, and N. Peng, “A multi-sensor fusion and object tracking algorithm for self-driving vehicles,” *Proc. Inst. Mech. Eng. Part D J. Automob. Eng.*, vol. 233, no. 9, pp. 2293–2300, 2019, doi: 10.1177/0954407019867492.
[44] Y. Bar-Shalom, F. Daum, and J. Huang, “The Probabilistic Data Association Filter: Estimation in the presence of measurement origin uncertainty,” *IEEE Control Syst.*, vol. 29, no. 6, pp. 82–100, 2009, doi: 10.1109/MCS.2009.934469.
[45] S. S. Blackman, “Multiple hypothesis tracking for multiple target tracking,” *IEEE Aerosp. Electron. Syst. Mag.*, vol. 19, no. 1 II, pp. 5–18, 2004, doi: 10.1109/MAES.2004.1263228.
[46] M. R. Cantas, S. Member, A. Chand, H. Zhang, G. C. Surnilla, and L. Guvenc, “Data Association Between Perception and V2V Communication Sensors,” pp. 1–9.
[47] “J2735E: V2X Communications Message Set Dictionary - SAE International.” https://www.sae.org/standards/content/j2735_202007/ (accessed Sep. 02, 2021).
[48] R. Palacios, “deg2utm - File Exchange - MATLAB Central.” https://www.mathworks.com/matlabcentral/fileexchange/10915-deg2utm (accessed Oct. 25, 2021).
[49] F. Ahmed-Zaid *et al.*, “VSC-A Final Report,” *Nhtsa*, no. September, pp. 1–102, 2011.
[50] “Sources of Greenhouse Gas Emissions | US EPA.” https://www.epa.gov/ghgemissions/sources-greenhouse-gas-emissions#transportation (accessed Nov. 25, 2021).
[51] T. Wada, K. Yoshimura, S. I. Doi, H. Youhata, and K. Tomiyama, “Proposal of an eco-driving assist system adaptive to driver’s skill,” *IEEE Conf. Intell. Transp. Syst. Proceedings, ITSC*, pp. 1880–1885, 2011, doi: 10.1109/ITSC.2011.6083034.
[52] T. Durbin, M. Wayne, K. Johnson, and M. Hajbabaei, “Final Report CARB Assessment of the Emissions from the Use of Biodiesel as a Motor Vehicle Fuel in California ‘ Biodiesel Characterization and NO x Mitigation Study ,’” no. x, 2011.
[53] J. Hu, Y. Shao, Z. Sun, M. Wang, J. Bared, and P. Huang, “Integrated optimal eco-driving on rolling terrain for hybrid electric vehicle with vehicle-infrastructure communication,” *Transp. Res. Part C Emerg. Technol.*, vol. 68, pp. 228–244, Jul. 2016, doi: 10.1016/J.TRC.2016.04.009.
[54] S. K. Zegeye, B. De Schutter, J. Hellendoorn, and E. A. Breunesse, “Variable speed limits for green mobility,” *IEEE Conf. Intell. Transp. Syst. Proceedings,*

*ITSC*, pp. 2174–2179, 2011, doi: 10.1109/ITSC.2011.6082833.
[55] G. Abu-Lebdeh, "Integrated Adaptive-Signal Dynamic-Speed Control of Signalized Arterials," *J. Transp. Eng.*, vol. 128, no. 5, pp. 447–451, Sep. 2002, doi: 10.1061/(ASCE)0733-947X(2002)128:5(447).
[56] M. Barth and K. Boriboonsomsin, "Energy and emissions impacts of a freeway-based dynamic eco-driving system," *Transp. Res. Part D Transp. Environ.*, vol. 14, no. 6, pp. 400–410, Aug. 2009, doi: 10.1016/J.TRD.2009.01.004.
[57] Y. Saboohi and H. Farzaneh, "Model for developing an eco-driving strategy of a passenger vehicle based on the least fuel consumption," *Appl. Energy*, vol. 86, no. 10, pp. 1925–1932, 2009, doi: 10.1016/J.APENERGY.2008.12.017.
[58] S. Mandava, K. Boriboonsomsin, and M. Barth, "Arterial velocity planning based on traffic signal information under light traffic conditions," *IEEE Conf. Intell. Transp. Syst. Proceedings, ITSC*, pp. 160–165, 2009, doi: 10.1109/ITSC.2009.5309519.
[59] M. A. S. Kamal, M. Mukai, J. Murata, and T. Kawabe, "On board eco-driving system for varying road-traffic environments using model predictive control," *Proc. IEEE Int. Conf. Control Appl.*, pp. 1636–1641, 2010, doi: 10.1109/CCA.2010.5611196.
[60] H. Rakha and R. K. Kamalanathsharma, "Eco-driving at signalized intersections using V2I communication," *IEEE Conf. Intell. Transp. Syst. Proceedings, ITSC*, pp. 341–346, 2011, doi: 10.1109/ITSC.2011.6083084.
[61] F. Mensing, R. Trigui, and E. Bideaux, "Vehicle trajectory optimization for application in ECO-driving," *2011 IEEE Veh. Power Propuls. Conf. VPPC 2011*, 2011, doi: 10.1109/VPPC.2011.6042993.
[62] Ş. Y. Gelbal, S. Tamilarasan, M. R. Cantaş, L. Güvenç, and B. Aksun-Güvenç, "A connected and autonomous vehicle hardware-in-the-loop simulator for developing automated driving algorithms," in *2017 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2017*, 2017, vol. 2017-Janua, doi: 10.1109/SMC.2017.8123155.
[63] O. D. Altan, G. Wu, M. J. Barth, K. Boriboonsomsin, and J. A. Stark, "GlidePath: Eco-friendly automated approach and departure at signalized intersections," *IEEE Trans. Intell. Veh.*, vol. 2, no. 4, pp. 266–277, 2017, doi: 10.1109/TIV.2017.2767289.
[64] O. Kavas-Torris, M. R. Cantas, S. Y. Gelbal, B. Aksun-Guvenc, and L. Guvenc, "Fuel economy benefit analysis of pass-at-green (PaG) V2I application on urban routes with STOP signs," *Int. J. Veh. Des.*, vol. 83, no. 2–4, pp. 258–279, 2020, doi: 10.1504/IJVD.2020.115058.
[65] M. R. Cantas, O. Kavas, S. Tamilarasan, S. Y. Gelbal, and L. Guvenc, "Use of Hardware in the Loop (HIL) Simulation for Developing Connected Autonomous Vehicle (CAV) Applications," *SAE Tech. Pap.*, vol. 2019-April, no. April, Apr. 2019, doi: 10.4271/2019-01-1063.
[66] "OpenStreetMap." https://www.openstreetmap.org/#map=5/38.007/-95.844 (accessed Dec. 02, 2021).
[67] P. A. Lopez *et al.*, "Microscopic Traffic Simulation using SUMO," *IEEE Conf.*

*Intell. Transp. Syst. Proceedings, ITSC*, vol. 2018-November, pp. 2575–2582, Dec. 2018, doi: 10.1109/ITSC.2018.8569938.

[68] M. Treiber and A. Kesting, "The Intelligent Driver Model with Stochasticity -New Insights into Traffic Flow Oscillations," *Transp. Res. Procedia*, vol. 23, pp. 174–187, 2017, doi: 10.1016/j.trpro.2017.05.011.

[69] M. Seredynski, B. Dorronsoro, and D. Khadraoui, "Comparison of Green Light Optimal Speed Advisory approaches," *IEEE Conf. Intell. Transp. Syst. Proceedings, ITSC*, no. October 2017, pp. 2187–2192, 2013, doi: 10.1109/ITSC.2013.6728552.

[70] M. R. Cantas, O. Kavas, S. Tamilarasan, S. Y. Gelbal, and L. Guvenc, "Use of hardware in the loop (HIL) simulation for developing connected autonomous vehicle (CAV) applications," *SAE Tech. Pap.*, vol. 2019-April, no. April, pp. 1–8, 2019, doi: 10.4271/2019-01-1063.

[71] R. Balke, Kevin; Florence, David; Feng, Yiheng; Leblanc, David; Wu, Guoyuan; Guenther, Hendrik-Joern; Moradi-Pari, Ehsan; Probert, Neal; Vijaya Kumar, Vivek; Williams, Richard; Yoshida, Hiroyuki; Yumak, Tuncer; Deering, Richard; Goudy, "Traffic optimization for signalized corridors (TOSCo) phase 1 project report," pp. 1–11, 2019, [Online]. Available: https://pronto-core-cdn.prontomarketing.com/2/wp-content/uploads/sites/2896/2020/02/TOSCo_Phase_1_Final_Report_-_CAMP_WEBSITE.pdf.

[72] B. Asadi and A. Vahidi, "Predictive Cruise Control : Utilizing Upcoming Traffic Signal Information for," *Ieee Trans. Control Syst. Technol.*, vol. 19, no. 3, pp. 707–714, 2011.

[73] Y. Jia, J. Wu, and M. Xu, "Traffic flow prediction with rainfall impact using a deep learning method," *J. Adv. Transp.*, vol. 2017, 2017, doi: 10.1155/2017/6575947.

[74] B. Bae, "Real-time Traffic Flow Detection and Prediction Algorithm: Data-Driven Analyses on Spatio-Temporal Traffic Dynamics," 2017, [Online]. Available: https://trace.tennessee.edu/utk_graddiss/4840/.

[75] T. Cui, "Short term traffic speed prediction on a large road network," 2019.

[76] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Phys. Rev. E - Stat. Physics, Plasmas, Fluids, Relat. Interdiscip. Top.*, vol. 62, no. 2, pp. 1805–1824, 2000, doi: 10.1103/PhysRevE.62.1805.

[77] I. Bae, J. Moon, and J. Seo, "Toward a Comfortable Driving Experience for a Self-Driving Shuttle Bus," *Electron. 2019, Vol. 8, Page 943*, vol. 8, no. 9, p. 943, Aug. 2019, doi: 10.3390/ELECTRONICS8090943.

[78] M. R. Cantas, G. C. Surnilla, and M. Sommer, "Green Light Optimized Speed Advisory with Traffic Preview," 2022. Under Review For SAE WCX 2022

[79] "Conventional Vehicle Reference Application - MATLAB & Simulink." https://www.mathworks.com/help/autoblks/ug/conventional-vehicle-reference-application.html (accessed Oct. 05, 2021).

[80] "ODOT, Marysville to test smart intersections, 1,200 connected vehicles to

develop road safety blueprint for U.S. cities - Columbus Business First." https://www.bizjournals.com/columbus/news/2018/11/20/marysville-to-test-smart-intersections-1-200.html (accessed Oct. 05, 2021).

[81] S. Y. Gelbal, M. R. Cantas, B. Aksun Guvenc, and L. Guvenc, "Virtual and Real Data Populated Intersection Visualization and Testing Tool for V2X Application Development," *SAE Tech. Pap.*, no. 2021, pp. 1–8, 2021, doi: 10.4271/2021-01-0164.

[82] "Deep Deterministic Policy Gradient — Spinning Up documentation." https://spinningup.openai.com/en/latest/algorithms/ddpg.html (accessed Dec. 02, 2021).

[83] "Train DDPG Agent for Adaptive Cruise Control - MATLAB & Simulink." https://www.mathworks.com/help/reinforcement-learning/ug/train-ddpg-agent-for-adaptive-cruise-control.html (accessed Dec. 02, 2021).

[84] B. Van Arem, J. G. Van Driel, and R. Visser, "The Impact of Cooperative Adaptive Cruise Control on Traffic-Flow Characteristics," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 4, p. 429, 2006, doi: 10.1109/TITS.2006.884615.

[85] L. Xiao and F. Gao, "A comprehensive review of the development of adaptive cruise control systems," *Veh. Syst. Dyn.*, vol. 48, no. 10, pp. 1167–1192, 2010, doi: 10.1080/00423110903365910.

[86] U. S. E. Of, A. In, and L. Driving, "1999 - Minderhoud and Bovy - Impact of Intelligent Cruise Control on Motorway Capacity," no. 99, pp. 1–9.

[87] G. J. L. Naus, R. P. A. Vugts, J. Ploeg, M. J. G. Van De Molengraft, and M. Steinbuch, "String-stable CACC design and experimental validation: A frequency-domain approach," *IEEE Trans. Veh. Technol.*, vol. 59, no. 9, pp. 4268–4279, 2010, doi: 10.1109/TVT.2010.2076320.

[88] R. Rajamani, H. S. Tan, B. K. Law, and W. Bin Zhang, "Demonstration of integrated longitudinal and lateral control for the operation of automated vehicles in platoons," *IEEE Trans. Control Syst. Technol.*, vol. 8, no. 4, pp. 695–708, 2000, doi: 10.1109/87.852914.

[89] R. Rajamani and S. E. Shladover, "Experimental comparative study of autonomous and co-operative vehicle-follower control systems," *Transp. Res. Part C Emerg. Technol.*, vol. 9, no. 1, pp. 15–31, 2001, doi: 10.1016/S0968-090X(00)00021-8.

[90] J. Ploeg, B. T. M. Scheepers, E. van Nunen, N. van de Wouw, and H. Nijmeijer, "Design and experimental evaluation of cooperative adaptive cruise control," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2011, pp. 260–265, doi: 10.1109/ITSC.2011.6082981.

[91] L. Güvenç *et al.*, "Cooperative Adaptive Cruise Control Implementation of Team Mekar at the Grand Cooperative Driving Challenge," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 3, pp. 1062–1074, 2012, doi: 10.1109/TITS.2012.2204053.

[92] V. Milanes, S. E. Shladover, J. Spring, C. Nowakowski, H. Kawazoe, and M. Nakamura, "Cooperative adaptive cruise control in real traffic situations," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 1, pp. 296–305, 2014, doi: 10.1109/TITS.2013.2278494.

[93] J. Meier *et al.*, "Implementation and Evaluation of Cooperative Adaptive Cruise

Control Functionalities," no. September, pp. 17–21, 2018.
[94] J. Ploeg *et al.*, "Cooperative Automated Maneuvering at the 2016 Grand Cooperative Driving Challenge," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 4, pp. 1213–1226, 2018, doi: 10.1109/TITS.2017.2765669.
[95] D. Swaroop, J. K. Hedrick, C. C. Chien, and P. Ioannou, "A Comparision of Spacing and Headway Control Laws for Automatically Controlled Vehicles," *Veh. Syst. Dyn.*, vol. 23, no. 1, pp. 597–625, 1994, doi: 10.1080/00423119408969077.
[96] M. R. Cantas, S. Y. Gelbal, L. Guvenc, and B. Aksun Guvenc, "Cooperative adaptive cruise control design and implementation," *SAE Tech. Pap.*, vol. 2019-April, no. April, Apr. 2019, doi: 10.4271/2019-01-0496.
[97] Ş. Y. Gelbal, M. R. Cantaş, S. Tamilarasan, L. Güvenç, and B. Aksun Güvenç, "A Connected and Autonomous Vehicle Hardware-in- the-Loop Simulator for Developing Automated Driving Algorithms," 2017, doi: 10.1109/SMC.2017.8123155.
[98] D. Zhang, K. Li, and J. Wang, "Radar-based target identification and tracking on a curved road," *Proc. Inst. Mech. Eng. Part D J. Automob. Eng.*, vol. 226, no. 1, pp. 39–47, 2012, doi: 10.1177/0954407011414462.
[99] Ö. Tunçer, L. Güvenç, F. Coşkun, and E. Karsligil, "Vision based lane keeping assistance control triggered by a driver inattention monitor," *Conf. Proc. - IEEE Int. Conf. Syst. Man Cybern.*, pp. 289–297, 2010, doi: 10.1109/ICSMC.2010.5642254.
[100] C. M. Kang, J. Lee, S. G. Yi, S. J. Jeon, Y. S. Son, W. Kim, S. H. Lee, and C. C. Chung, "Lateral control for autonomous lane keeping system on highways," *ICCAS 2015 - 2015 15th Int. Conf. Control. Autom. Syst. Proc.*, pp. 1728–1733, 2015, doi: 10.1109/ICCAS.2015.7364643.
[101] S. Yenikaya, G. Yenikaya, and E. Düven, "Keeping the vehicle on the road," *ACM Comput. Surv.*, vol. 46, no. 1, pp. 1–43, 2013, doi: 10.1145/2522968.2522970.
[102] M. T. Emirler, H. Wang, B. Aksun Güvenç, and L. Güvenç, "Automated Robust Path Following Control Based on Calculation of Lateral Deviation and Yaw Angle Error," Jan. 2016, doi: 10.1115/DSCC2015-9856.
[103] M. R. Cantas and L. Guvenc, "Camera Based Automated Lane Keeping Application Complemented by GPS Localization Based Path Following," *SAE Tech. Pap.*, 2018, doi: 10.4271/2018-01-0608.
[104] E. J. Rossetter, "A potential field framework for active vehicle lanekeeping assistance," no. December, p. 155, 2003.
[105] L. Güvenç, B. Aksun-Güvenç, B. Demirel, and M. T. Emirler, "Control of mechatronic systems," *Control Mechatron. Syst.*, pp. 1–200, Jan. 2017, doi: 10.1049/PBCE104E.

Appendix A. Abbreviations Table

| **Abbreviation** | **Definition** |
|---|---|
| ADAS | Advanced Driver Assistant System |
| BSM | Basic Safety Message |
| BSW | Blind Spot Warning |
| CA | Collison Avoidance |
| CACC | Cooperative Adaptive Cruise Controller |
| CAV | Connected and Autonomous Vehicles |
| CDA | Cooperative Driving Automation |
| CP | Cooperative Perception |
| CPM | Cooperative Perception Message |
| CRB | Complex Root Boundary |
| DDPG | Deep Deterministic Policy Gradient |
| DSRC | Dedicated Short Range Communication |
| ETSI | European Telecommunications Standards Institute |
| FOV | Field of View |
| GCDC | Grand Cooperative Driving Challenge |
| GLOSA | Green Light Optimized Speed Advisory |
| GPS | Global Positioning System |
| HDOP | Horizontal Dilution of Precision |
| HIL | Hardware in The Loop |
| HV | Host Vehicle |
| IDM | Intelligent Driver Model |
| IMA | Intersection Movement Assist |
| IRSU | Intelligent Roadside Unit |
| JPDA | Joint Probability Data Association |
| JPDAF | Joint Probabilistic Data Association Filter |
| KF | Kalman Filter |
| LTA | Lest Turn Assist |
| MCMCDA | Markov Chain Monte Carlo Data Association |
| MHT | Multiple Hypothesis Tracking |
| MPC | Model Predictive Control |
| OBU | Onboard Unit |
| PID | Proportional Integral Derivative |
| PDAF | Probabilistic Data Association Filter |
| PT | Passing- Time |

| | |
|---|---|
| RELU | Rectified Linear Unit |
| ROS | Robot Operating System |
| RRB | Real Root Boundary |
| RSU | Road Side Units |
| RV | Remote Vehicle |
| SPAT | Signal phase and timing |
| TAS | Traffic Advisory Speed |
| TMA | Track Matching Accuracy |
| TTTD | Track to Track Distances |
| UTM | Universal Transverse Mercator |
| VRU | Vulnerable Road User |
| V2X | Vehicle to Everything |
| WSU | Wireless Safety Unit |

Table 5 Abbreviations table