2019

MPC-BASED AUTONOMOUS DRIVING CONTROL WITH LOCALIZED PATH PLANNING FOR OBSTACLE AVOIDANCE AND NAVIGATING SIGNALIZED INTERSECTIONS

Sai Rajeev Devaragudi
Michigan Technological University, sdevarag@mtu.edu

MPC-BASED AUTONOMOUS DRIVING CONTROL WITH LOCALIZED PATH PLANNING FOR OBSTACLE AVOIDANCE AND NAVIGATING SIGNALIZED INTERSECTIONS

By

Sai Rajeev Devaragudi

A REPORT

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

In Mechanical Engineering

MICHIGAN TECHNOLOGICAL UNIVERSITY

2019

This report has been approved in partial fulfillment of the requirements for the Degree of MASTER OF SCIENCE in Mechanical Engineering.

Department of Mechanical Engineering – Engineering Mechanics

Report Advisor: *Bo Chen*

Committee Member: *Zequn Wang*

Committee Member: *Ye Sun*

Department Chair: *William W. Predebon*

Table of Contents

List of figures v

List of tables ix

Acknowledgments x

Abstract xi

1 Introduction 1

1.1 Literature Review 3

1.1.1 Vehicle Control 3

1.1.2 Motion Planning 3

1.1.3 Connected Vehicles 4

1.2 Decision-Making Process in Autonomous Vehicles 5

1.3 Research Objectives and Contributions 7

2 Path Planning 9

2.1 Obstacle Data 9

2.2 A* Algorithm 11

2.3 Trajectory Generation 15

3 Model Predictive Vehicle Control 21

3.1 Prediction Model 22

3.1.1 Longitudinal Control Model 22

3.1.2 Lateral Control Model 26

3.2 Problem Formulation 29

3.3 Closed-Loop Implementation 30

4 Simulation Results 32
4.1 Scenario Setup 32
4.2 Simulation Setup 34
4.3 Scenario 1 – Stationary Obstacle 37
4.4 Scenario 2 – Moving Obstacle 42
4.5 Scenario 3 – Multiple Obstacles 45
5 Approach and Departure at Signalized Intersections 50
5.1 Motivation 50
5.2 Data Extraction from V2I transmitters 50
5.2.1 Data Extraction from MAP Message 52
5.2.2 Data Extraction from SPAT Message 56
5.3 Implementation of Stop/Go motion 58
5.4 Simulation Setup 59
5.5 Simulation Results 63
6 Conclusions & Future Work 66
6.1 Conclusion 66
6.2 Future Work 67
7 Reference List 68
Appendix A: Structure of the MAP Message 72
Appendix B: Structure of the SPAT Message 73
Appendix C: Copyright documentation 74

List of figures

Figure 1.1: Statistics of safety concerns while driving in the U.S. [3] 1

Figure 1.2: Waymo's self-driving taxi in Arizona. Image Source: By Dllu - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=64517567 2

Figure 1.3: Flowchart depicting the decision making process in self-driving vehicles.....5

Figure 2.1: Representation of data obtained from the PreScan® sensor model 10

Figure 2.2: Possible future nodes are highlighted for a node (i, j).....13

Figure 2.3: Optimal path obtained using A* for a sample dataset.....14

Figure 2.4: Comparison of Path Planner and Trajectory Generator Output – The marked portion highlights the joining of two different Bezier curves when there is a change in the path planner output. 19

Figure 3.1: Overview of MPC based Longitudinal and lateral control model. 22

Figure 3.2: Representation of Relative Distance and Safe Distance 26

Figure 3.3: Representation of the dynamic bicycle model for lateral control [39].....28

Figure 4.1: Sample scenario that can be created on PreScan software.....33

Figure 4.2: Vehicle Dynamics Parameters for the Toyota Prius Model 34

Figure 4.3: Vis-Viewer Setup for visualizing the ego vehicle.....35

Figure 4.4: Object detected in the path of the ego vehicle through the vision sensor 36

Figure 4.5: Safe distance ($Dsafe$) versus relative distance ($DRel$) as the ego vehicle is approaching to the obstacle.....38

Figure 4.6: Command velocity ($Vcmd$) versus ego vehicle velocity ($xego$) through the object avoidance maneuver 38

Figure 4.7: Comparison of the reference trajectory and actual vehicle trajectory obtained using the MPC controller 39

Figure 4.8: Steering wheel angle (δ) to generate the object avoidance maneuver 39

Figure 4.9: Tractive Force (Ft) implemented by MPC controller 40

Figure 4.10: Ego vehicle deviating from the path to avoid collision with the obstacle 40

Figure 4.11: Ego vehicle at the maximum deviation from the path 41

Figure 4.12: Ego Vehicle rejoining the path after crossing the obstacle 41

Figure 4.13: Safe distance ($Dsafe$) versus relative distance ($DRel$) as the ego vehicle approaches the obstacle 43

Figure 4.14: Comparison of reference velocity, ego velocity ($xego$) and lead velocity ($Vlead$) while following and passing 43

Figure 4.15: Comparison of the reference trajectory and actual vehicle trajectory obtained to avoid a collision 44

Figure 4.16: Steering wheel angle (δ) for the object avoidance maneuver 44

Figure 4.17: Tractive force (Ft) implemented by the MPC controller 45

Figure 4.18: Multiple objects test scenario created using trucks 46

Figure 4.19: Command velocity ($Vcmd$) versus ego vehicle velocity ($xego$) through the multiple objects avoidance maneuver 46

Figure 4.20: Safe distance ($Dsafe$) versus relative distance ($DRel$) as the ego vehicle is approaching to the obstacles 47

Figure 4.21: Safe distance ($Dsafe$) versus relative distance ($DRel$) as the ego vehicle approaches the obstacles 47

Figure 4.22: Comparison of the reference trajectory and actual vehicle trajectory obtained to avoid both the obstacles in the path 48

Figure 4.23: Steering wheel angle (δ) for the object avoidance maneuver for both obstacles in the path.....48

Figure 4.24: Tractive force (Ft) implemented by the MPC controller 49

Figure 5.1: Representation of a sample DSRC Message Frame 51

Figure 5.2: Location of the Node List XY Data Frame in the MAP Message 53

Figure 5.3: The contents of the Generic Lane Data Frame.....54

Figure 5.4: Representation of Latitude and Longitude of the stop line 55

Figure 5.5: Representation of Signal Group ID Location.....56

Figure 5.6: Schematic of SPAT Message Data Frame 57

Figure 5.7: Contents of Time Change Details Data Frame.....57

Figure 5.8: Positioning of the V2X transmitter on the traffic signal. 59

Figure 5.9: Phase change for Signal 1 60

Figure 5.10: Phase Change for Signal 2.....60

Figure 5.11: Scenario created with signalized intersections in PreScan GUI - The red boxes highlight the position of the traffic signals. 61

Figure 5.12: V2X plugin setting on PreScan 62

Figure 5.13: Visualization of the V2I Message Packet.....62

Figure 5.14: Velocity of ego vehicle through two signalized intersections.....64

Figure 5.15: Relative distance ($Drel$) versus stopping distance ($Dstop$) through signalized intersections. 64

Figure 5.16: Tractive Force requested by the MPC controller for the stop/go motion.....65

Figure 7.1: Payload of MAP Message 72

Figure 7.2: Payload of SPAT Message 73

List of tables

Table 1: Path Planner & Trajectory Generation 17

Table 2: targetCalc($xego, xinit$) 18

Table 3: Bezier($y, ypos, yinit, xego, ind$) 18

Table 4: Vehicle parameters for Toyota Prius 33

Table 5: Simulation and model parameters for MPC controller 36

Acknowledgments

I would like to take this opportunity to thank my faculty advisor for the last two years, Dr. Bo Chen, for her able guidance throughout my Master's degree. She had given me complete freedom in choosing the research topic for my master's degree and guiding me through the process of completing my research. I admire her dedication and support in reviewing my work progress and her constant encouragement which helped me to submit a conference publication for peer review which has been accepted for publication. I appreciate her trust in me when she accepted my leave of absence and allowed me to pursue an internship in between the semester. I thank her for her able guidance throughout my time at the IMES lab and for allowing me to work on such projects which have honed my engineering skills in the field of controls.

I would also like to thank Dr. Zequn Wang and Dr. Ye Sun for serving as committee members for my Master's committee and for their support and encouragement through their review and constructive feedback on my work.

I would like to thank Siemens for providing us the Academic Partner Program Grant for PreScan® licenses. I would also like to thank Mr. Aditya Kothari and Mr. Yun Cai, Application Support Engineers at Siemens PLM for the technical support provided for using the various PreScan® toolboxes.

I am also grateful to all my colleagues who were working with me at the IMES lab and for guiding me through each step and explaining the nuances involved in any academic research. I am grateful for all the valuable feedback I received from my peers and lab mates on my work and for contributing to my work in the best of their efforts.

Lastly, I would like to thank my parents and my sister for all their emotional support throughout my graduate program which played a huge role throughout my time here at Michigan Tech. I would also like to thank my friends for their motivation and support during the course of my graduate studies.

Abstract

Connected and autonomous vehicles are becoming the major focus of research for the industry and academia in the automotive field. Many companies and research groups have demonstrated the advantages and the requirement of such technology to improve the energy efficiency of vehicles, decrease the number of crash and road accidents, and control emissions.

This research delves into improving the autonomy of self-driving vehicles by implementing localized path planning algorithms to introduce motion control for obstacle avoidance during uncertainties. Lateral path planning is implemented using the A* algorithm combined with piecewise Bezier curve generation which provides an optimum trajectory reference to avoid a collision. Model Predictive Control (MPC) is used to implement longitudinal and lateral control of the vehicle. The data from vehicle-to-everything (V2X) communication infrastructure is used to navigate through multiple signalized intersections. Furthermore, a new method of developing Advanced Driver Assistance Systems (ADAS) algorithms and vehicle controllers using Model-In-the-Loop (MIL) testing is explored with the use of PreScan®. With PreScan®, various traffic scenarios are modeled and the sensor data are simulated by using physics-based sensor models, which are fed to the controller for data processing and motion planning. Obstacle detection and collision avoidance are demonstrated using the presented MPC controller. The results of the proposed controller and the scope of the future work conclude the research.

1 Introduction

The major causes of road accidents in the United States in 2016 were reported due to driver distraction and driving-related accidents [1] as shown in Figure 1.1. One of the major ways the automotive industry and academia are focusing their research efforts to mitigate such fatalities is by heavily investing in research on autonomous and connected vehicles. By implementing Advanced Driver Assistance Systems (ADAS) technologies, majority of the driving-related accidents can be mitigated [2]. There has been an increasing demand for ADAS and self-driving technologies in the automotive industry due to increasing customer demand and stricter regulation to improve safety standards in automobiles.

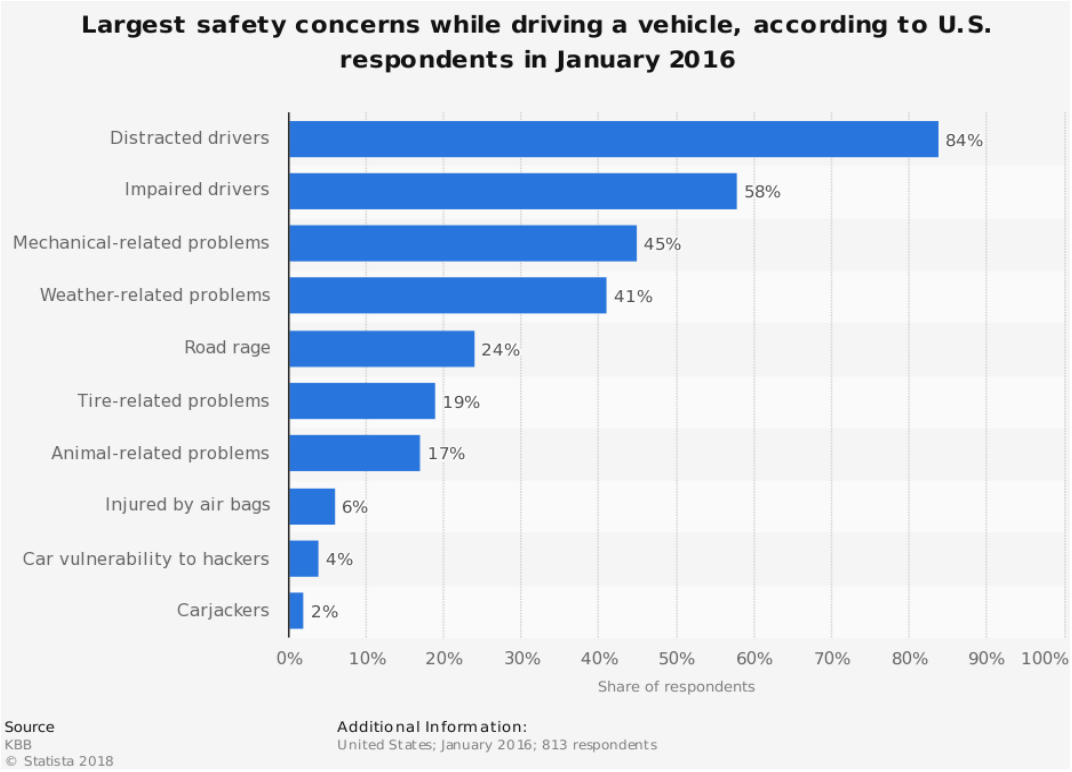

Figure 1.1: Statistics of safety concerns while driving in the U.S. [3]

The material contained in this chapter has been accepted for publication to the International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, ASME, 2019.

With the advent of Waymo One [4] self-driving taxi in Arizona shown in Figure 1.2 and Tesla's Autopilot [5], ADAS has become more popular than before. The technical breakthrough in the field of perception sensors and computational power in automobiles has advanced the level of autonomy possible in automobiles. Levels of automation in automobiles are classified based on the SAE standard referred in [6]. As per the standard the vehicles on road will be classified as SAE level 2 as it is considered to be partial automation since they require constant driver supervision of the surroundings at all times.

Figure 1.2: Waymo's self-driving taxi in Arizona. Image Source: By Dllu - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=64517567

Though there is no production vehicle that can be classified as SAE level 3 or beyond in the United States, the concept of a fully autonomous vehicle has been demonstrated in 2007 at the DARPA Urban Challenge [7]. The event requires teams to build and demonstrate a fully autonomous vehicle capable of navigating itself through numerous urban driving scenarios [8]. Six of the participant teams were able to complete the challenge without any human intervention. These six teams used various methods for path planning and motion control, which were implemented using various onboard sensors and computers. The practicality of such methods in the automotive industry can be realized now more than ever

due to the advancements in the onboard computational technology for sensor data processing.

1.1 Literature Review

This section is used to review some of the relevant literature in the field of self-driving vehicles, which provides valuable information on the aspects of decision making, path planning, and motion control, particularly for the autonomous systems that can be classified as level 3 and above.

1.1.1 Vehicle Control

Several methods have already been implemented by the automotive industry to achieve longitudinal and lateral motion control. Article [9] introduced longitudinal control to follow the lead car using vision-based sensors. Article [10] introduced methods of velocity prediction based on tire forces estimation and [11] demonstrated the implementation of adaptive cruise control (ACC) using model predictive methods. The lateral motion of a vehicle was achieved using the Pure-pursuit method [12], which is suitable for a non-holonomic system such as a vehicle steering. The pure-pursuit method was also used by three of the six finalists of the DARPA Urban challenge as reported in [7]. Another trajectory tracking methodology based on a control Lyapunov function was used in [13]. Further, a comparison of various control methods for lateral motion control was given in [14].

1.1.2 Motion Planning

The DARPA Challenge participants also demonstrated the use of various motion planning algorithms, which were deployed to facilitate the decision-making process for these autonomous vehicles. The winner of the challenge was Carnegie Mellon University's team.

The team demonstrated trajectory generation in a 4-dimensional configuration space combined with Anytime D* algorithm to achieve obstacle avoidance [15]. Stanford's team who was the runner-up team of the competition used a search strategy term called Hybrid A*, which is a variant of the A* algorithm with motion primitives for application on non-holonomic and continuous systems [16]. Virginia Tech's team finished third in the competition and used A* algorithm for route planning and a graph construction process of all the possible maneuvers. An arbitration method was implemented to select the optimum maneuver that was tracked by the controller [17]. The vehicle developed by MIT used a variation of an existing incremental tree-based search method known as Rapidly-exploring Random Tree (RRT) introduced in [18]. The variant used by MIT was called closed-loop RRT with biased sampling [19, 20]. Further motion planning algorithms have been introduced for planning the path of autonomous robots and unmanned aerial vehicles (UAV's) such as RRT* [21] and a computationally faster version of the RRT* known as the Batch Informed Trees (BIT*) as introduced in [22].

1.1.3 Connected Vehicles

The increasing availability of wireless communication technologies, which facilitates vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) via technologies such as Dedicated Short Range Communication (DSRC) [23] enables communication and data exchange from other vehicles and also real-time traffic information from road-side units (RSUs) such as signalized traffic intersections and congestion monitoring units. The improvement in the efficiency of a vehicle powertrain by optimizing the velocity profile in connected cars is demonstrated in [24]. Improved fuel economy and reduced CO_2 emissions in vehicles utilizing upcoming traffic signal information is demonstrated in [25]. Further, an algorithm to extract the traffic signal phase data from the original SPAT (Signal Phasing and Timing) messages, which is periodically broadcasted by the intersection is discussed in [26].

1.2 Decision-Making Process in Autonomous Vehicles

This section is used to describe the decision-making process that takes place in an autonomous system. Various aspects of this process have to be executed by driverless cars to complete any particular maneuver. These decisions are made based on the data from various onboard sensors such as Radar, LIDAR, camera/vision systems, Global Positioning Sensors and Inertial Measurement Units (GPS/IMU) and V2X communication modules. The data are used to automatically select an appropriate driving behavior, which then plans a motion trajectory and calculates the values of the control variables to execute the maneuver. These decision-making tasks are broadly classified into the following categories and the process chart is shown in Figure 1.3.

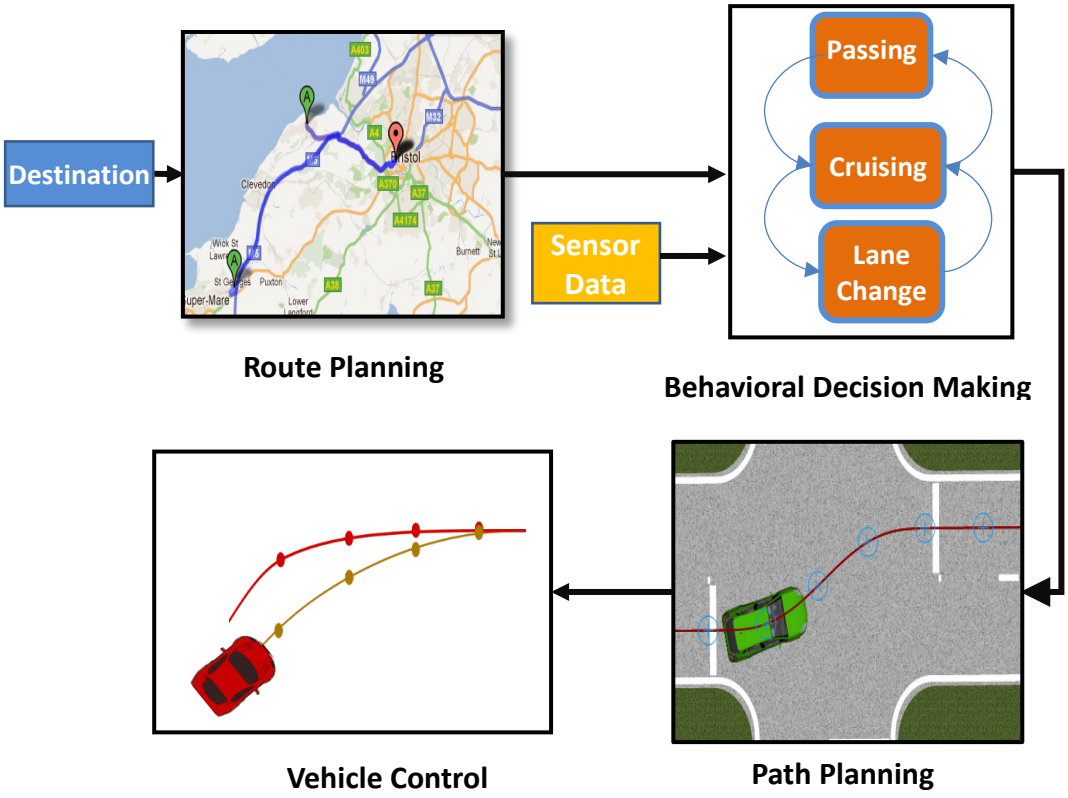

Figure 1.3: Flowchart depicting the decision making process in self-driving vehicles

1. **Route Planning:** involves selecting the shortest or most optimum route based on the distance to destination and time of travel. This also uses real-time traffic and road data to change routes accordingly. A map is used to generate the most optimum route based on the starting point and destination and this route is optimized based on the real-time data.

2. **Behavioral Decision Making:** Once the optimum route is planned, a decision making layer is used to select a driving behavior based on the sensor data, traffic information and other participants. Using the perceived data, different driving behaviors are employed for various road segments and traffic conditions. This layer of the algorithm can be tuned to closely replicate human driving behavior. Machine learning and Gaussian matrix models are usually used for prediction of surrounding vehicles' states and to plan the trajectory accordingly.
3. **Motion Planning:** Based on the selected driving behavior, a path planning algorithm is required to generate an optimum trajectory for completing the driving maneuver selected by the decision making layer. The generated path acts as a reference to the vehicle controller. The generated path has to be dynamically feasible for a non-holonomic system and should also be comfortable for the passengers. Collision avoidance strategies are also deployed in this layer.

4. **Vehicle Control:** A closed-loop feedback control system is required for controlling the longitudinal and lateral motion of the vehicle. This layer is responsible for tracking the trajectory generated by the path planner by determining the appropriate values for the control variables.

The scope of this research is limited by focusing on a novel path planning algorithm, vehicle control method, and MIL testing of the control algorithm.

1.3 Research Objectives and Contributions

Though many path planning algorithms were introduced for self-driving vehicles as mentioned in section 1.1.2, these methods were not practical to use in real-time as they are computationally expensive. The vehicles come to a complete stop and the planning algorithm is executed to find an alternate path. This solution is not practical for real-world driving situations, where the self-driving vehicle are required to immediately react to any kind of obstacle in the path. The objective of this research is to introduce a simple planning algorithm combined with a closed-loop feedback controller that is computationally efficient and can be implemented and tested in real-time.

This research proposes the use of model predictive control for throttle, brake and steering control of an autonomous vehicle. The route to be followed is assumed to be generated from the map data. The obstacle state i.e. location, orientation, and velocity is obtained using the camera and vision sensors included in the PreScan® sensor library. The A* algorithm combined with piecewise Bezier trajectory generator is implemented for real-time localized path planning and obstacle avoidance by defining the desired lateral position of the vehicle. The trajectory generator defines the desired lateral position of the vehicle. The desired longitudinal position is determined based on the user-defined velocity command and the lead vehicle velocity. Physics-based communication transceivers are used for data exchange from road-side infrastructure to determine the signal phase and stop/go motion is achieved accordingly. The controller is tested for various traffic scenarios using PreScan® to validate the controller design.

The proposed control algorithm can be implemented in real-time as it is computationally faster and more efficient to run. The control algorithm is tested using PreScan® in a fixed time step using Model-in-the-loop (MIL) methods.

To implement a computationally efficient and optimal path planning algorithm it is essential that the obstacle data obtained from the sensors are accurate. Using the obstacle data the A* algorithm is executed to find an optimal path avoiding the obstacles in the path. This path is converted into a trajectory using Bezier curves. The implementation of the proposed algorithm is further detailed in Chapter 2 of this report.

2 Path Planning

Path Planning is an important aspect of an autonomous vehicle. It can be categorized into global and local path planning. Global path planning or route planning is based on the current position of the vehicle, user-entered destination, and a map of the surrounding region. An economic route is selected to reach the destination by traversing the minimum distance. Local path planning or motion planning is localized to a particular situation based on the surrounding environment. It is usually done to change lanes or avoid obstacles on the route. Global path planning is beyond the scope of this paper and an optimal route is assumed. Local path planning is achieved with the help of the A* algorithm.

A* is a graph search method introduced as a graph traverser algorithm for an automated robot [27]. It is used to find the best path on a graph but is computationally efficient by making use of a heuristic approach [28]. The algorithm requires some predetermined information such as the current location of the ego vehicle, the obstacle location, and the target position.

2.1 Obstacle Data

The obstacle data are obtained with a physics-based model of an object camera sensor available in the PreScan® software. The data are generated on the basis of the infrastructure and the scenario that is created by the PreScan® GUI. The object camera sensor is modeled to represent a system that contains a camera and an image processing unit. This sensor determines the longitudinal distance (x_{obs}) of the object and the lateral position (y_{obs}) with respect to the sensors location on the ego vehicle. For a moving object, the sensor can be

---

The material contained in this chapter has been accepted for publication to the International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, ASME, 2019.

used to determine the relative velocity of the object with respect to the velocity of the ego vehicle.

The sensor is also able to generate coordinates of the position of an object with respect to the image captured by the camera unit. The data are used to determine the actual width of the object. Screen coordinates range from -1 to 1 in both horizontal and vertical directions and the center of the image is considered to be the origin as illustrated in Figure 2.1. Only the on-screen part of an object is taken into account and any non-visible parts of the object are omitted.

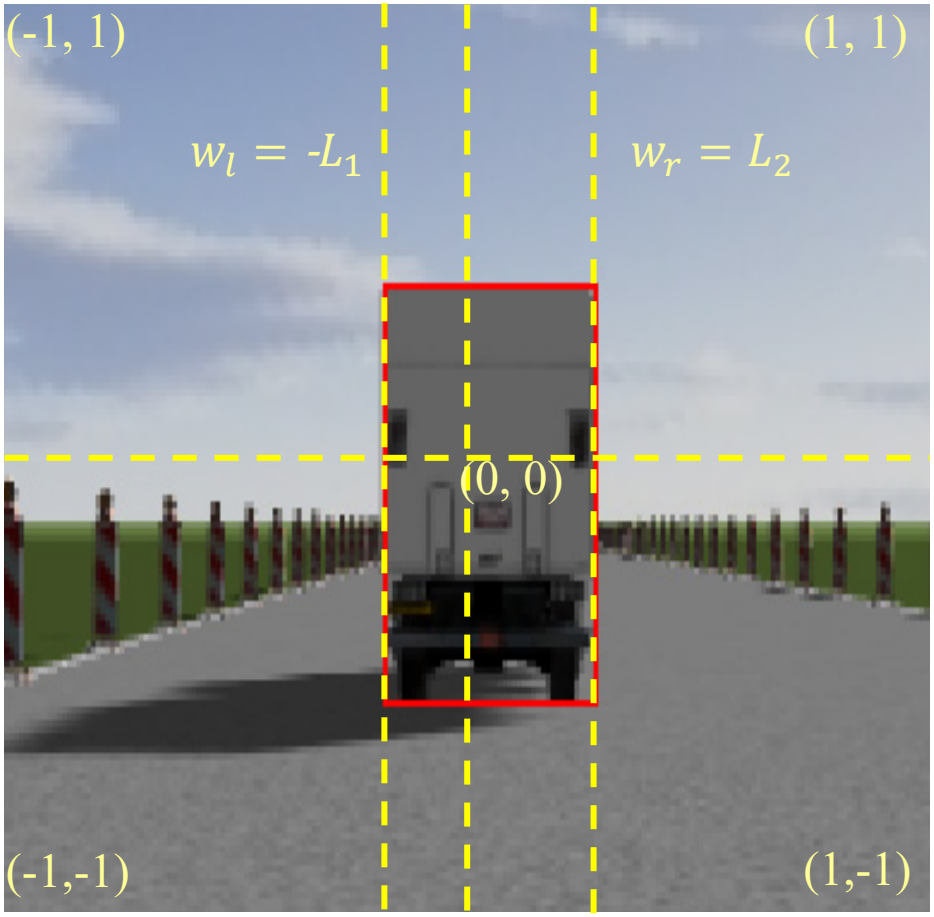

Figure 2.1: Representation of data obtained from the PreScan® sensor model

From Figure 2.1, the on-screen width ($width_{screen}$) of the obstacle can be estimated as the difference of the X-coordinate of the right edge (w_r) and the X-coordinate of the left edge (w_l) of the obstacle. The actual width of the obstacle is obtained using eqn. (2.1).

$$width_{act} = \tan\left(\frac{FOV}{2}\right) * x_{obs} * width_{screen} \tag{2.1}$$

Where $width_{act}$ is the actual width of the obstacle; FOV is the camera beam's field of view. This actual width ($width_{act}$) is used in the A* algorithm to mark the position of the obstacles. The width data combined with the obstacle position is used to plan a path for the ego vehicle such that any collision is prevented. Since the algorithm assumes the ego vehicle as a point in one of the nodes of the map, the size of the ego vehicle should also be considered.

To compensate for the size of the ego vehicle, the width of the obstacle is increased by the track width of the ego vehicle. This additional width is added only to the obstacles that are directly in front of the ego vehicle. This prevents unnecessary deviation from the path due to obstacles that are not in the path of the ego vehicle. Once all the required obstacle data is obtained, the data is fed to the path planner and a path is planned using the A* algorithm which is described further in section 2.2.

2.2 A* Algorithm

To implement the algorithm, the complete road space is converted into a grid with squares or nodes of size 1m. Each node represents the possible positions of the vehicle and a target position is calculated such that the target is saturated to a node that is 30 m ahead of the vehicle position node. This maximum distance (x_{max}) is selected based on the computational efficiency and the sensor range.

The algorithm calculates the cost function $(f(n))$ for every possible node. This cost is the sum of two costs associated for every node. The first, is the cost to reach a future node $(g(n))$ and the second term $(h(n))$ is the cost to reach the target node from the future node. The cost function is the sum of $g(n)$ and $h(n)$ represented by $f(n)$ as shown in eqn. (2.2).

$$f(n) = g(n) + h(n) \tag{2.2}$$

Consider the present path planner node, represented as (x_1, y_1) and a future node which is being considered, represented as (x_2, y_2). Let the target node for the path planner be (x_t, y_t). The cost $(g(n))$ and $(h(n))$ is determined using eqn. (2.3)

$$\begin{aligned} g(n) &= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \\ h(n) &= \sqrt{(x_t - x_2)^2 + (y_t - y_2)^2} \end{aligned} \tag{2.3}$$

This cost function is evaluated for all the possible nodes of the discretized configuration space. The nodes with the minimum cost function are connected to form the most optimal path to reach the target position. For any given node (i, j) there are eight possible successors as shown in Figure 2.2. To keep track of the nodes occupied by obstacles and the nodes for which the cost function has already been evaluated, the cost data is stored for each of the possible future nodes using lists.

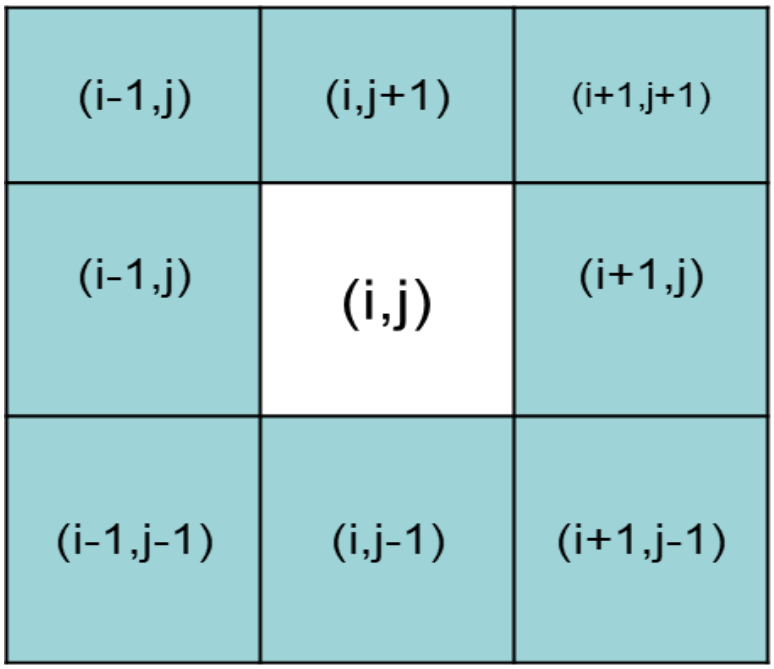

Figure 2.2: Possible future nodes are highlighted for a node (i, j)

The algorithm maintains two lists known as the *Open* and the *Close* list. The open list consists of the nodes that have been visited but successor nodes have not been explored. The close list consists of the nodes that have been visited and its successors have been found. The nodes containing any obstacle obtained via the sensor is also added to the Close list. The successor node that is yet to be explored is added to the Open list and the list is populated until an optimum successor node is found. This process is repeated until the successor node is the target node. Once the target node is reached, the optimal path is obtained by tracing back the parent nodes with the minimum cost to obtain the optimal trajectory by avoiding obstacles.

This optimum path obtained from the algorithm consists of waypoints to be followed so that the vehicle reaches the target position by avoiding obstacles. Though these waypoints represent the optimum position for the ego vehicle to be, a trajectory needs to be generated

for the ego vehicle to reach the successor node. The path obtained by the algorithm for a sample set of data is shown in Figure 2.3. It can be seen that the path obtained by the algorithm is not continuous and dynamically unfeasible due to the sharp change in angles. The trajectory needs to be dynamically feasible for a non-holonomic system such as a vehicle steering system that is continuous in nature so that it can be used as a reference for the closed-loop vehicle controller. A smoother curve would ensure smoother steering increments and decreased passenger discomfort.

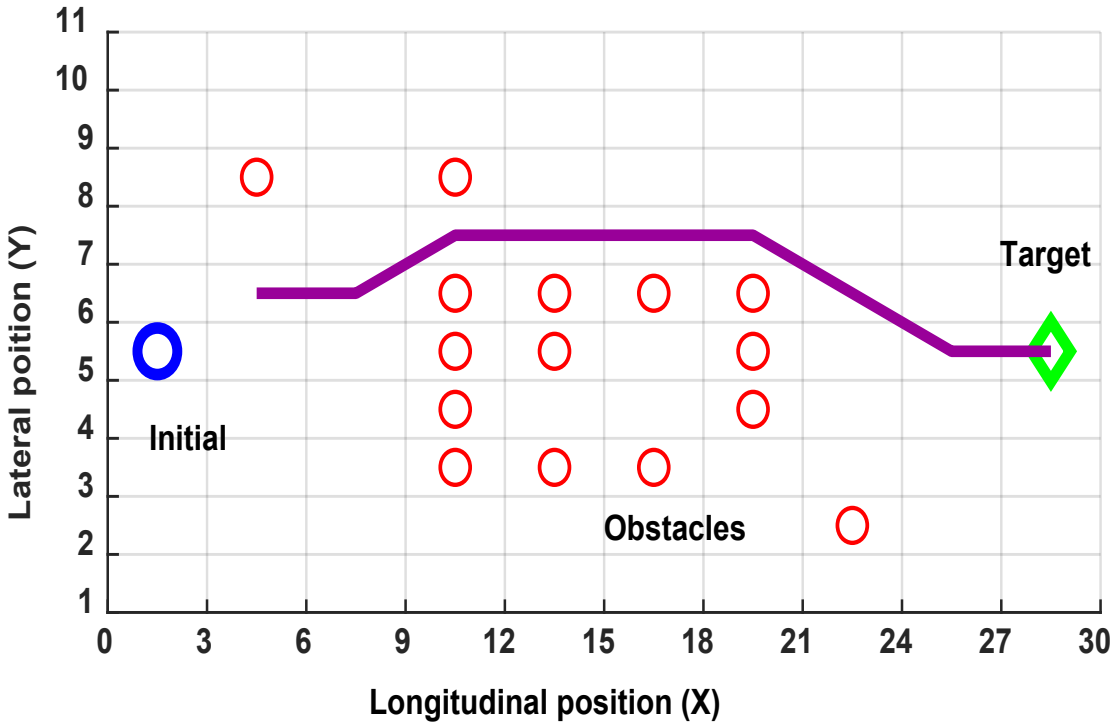

Figure 2.3: Optimal path obtained using A for a sample dataset*

To improve the smoothness of the curve, the piecewise Bezier curve approach is used in a trajectory generator function which is further described in section 2.3.

2.3 Trajectory Generation

Though the A* algorithm is computationally efficient in finding an optimal path, the path generated is not continuous and cannot be tracked by a non-holonomic system such as a vehicle steering system. To obtain a continuous curvature, a trajectory generator is used that converts the discontinuous path into a smoother curve using the Bezier curve equation [29]. A Bezier curve of degree n can be represented by eqn. (2.4).

$$P(t) = \sum_{i=0}^{n} J_i^n(t)\boldsymbol{B_i}, \quad t \in [0,1] \tag{2.4}$$

In eqn. (2.4), $P(t)$ represents the Bezier curve that is bounded by the control points or waypoints represented by $\boldsymbol{B}_i$ and $J_i^n(t)$ represents the Bernstein polynomial [30] given by eqn. (2.5)

$$J_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i} \tag{2.5}$$

Using eqn. (2.4) and (2.5), a Bezier curve cannot be used for the entire path as the resulting curve may pass through the obstacles. Therefore a variation of the piecewise Bezier curve approach [31] was used to obtain a smooth and continuous curve throughout the path. Using eqns. (2.4) & (2.5) a smooth curve can be obtained for a set of every 4 consecutive waypoints obtained from the path planner. The smoothness of the curve depends on the discretization of the variable t in eqn. (2.4). For a smoother curve, a smaller discretization interval of t shall be used.

The path planner outputs are waypoints that have the desired lateral and longitudinal position. However, we are using the Bezier curve equations mentioned in eqns. (2.4) and (2.5), only for the lateral position of the vehicle. The path planner output is converted into the desired lateral deviation with respect to time using the Bezier curve function explained

in Table 3. This is done because the MPC controller also controls the longitudinal position of the vehicle by changing the speed of the vehicle based on the lead traffic and command velocity. To convert the path planner output into lateral deviation with respect to time, the frequency of the path planner is decreased and the frequency of the trajectory generator is increased as described in Table 1. This process is repeated continuously as the waypoints are updated to generate a smooth curve that can be tracked by the ego vehicle. The smoothness of the curve depends on the discretization of the variable t in eqn. (2.4). The output of the trajectory generator for a lateral deviation of 3m produced by the path planner is shown in Figure 2.4. It can be seen that the path planner output is suddenly changed as the A* algorithm is implemented such that it can only output a minimum change of 1 m in position. This 1m is further discretized using the trajectory generator by using a smaller discretization interval (t) and by increasing the frequency of the algorithm, creating a smooth curve.

The implementation of the path planner and trajectory generator is shown in the pseudocode mentioned in Table 1. The first four waypoints obtained by the path planning algorithm are passed to the trajectory generator function to convert the path to a smooth Bezier curve. The target node for the path planner is calculated by calling the $targetCalc()$ function, described in Table 2. Once all the variables are initialized, the path planner function is executed every 1 second if there is an obstacle in the path of the ego vehicle. Though for every iteration the path planner provides the complete optimal path till the target node, the trajectory generator only considers the first four nodes of the path, to generate a smooth trajectory to complete the maneuver as described in Table 3. Since the grid of the A* path planner is discretized into blocks of 1m square size, this deviation is to be further discretized to ensure a smooth steering maneuver to prevent sudden yaw angle changes. This is achieved by using a faster sampling rate for the trajectory generator as described in Table 1. Due to the faster sampling rate, smaller discretization interval for Bezier curve (t) is used as shown in Table 3. Based on the actual vehicle lateral position

(y_{pos}) and the planned vehicle lateral position (y_{points}), the direction of the curve is decided. Based on this direction, the points on the trajectory are determined using the Bezier equation and is appropriately indexed using the static variable (ind). The output (y_{ref}) is used as a reference value for the trajectory, that the vehicle has to follow and is fed to the variable (y_{ref}) as described in Table 3. . Using these reference points the change in the required yaw angle (ψ_{ref}) is calculated by taking the tangent of 2 consecutive points as shown in Table 3.

Table 1: Path Planner & Trajectory Generation

Initialization
x_{init} ← Initial longitudinal position of the vehicle
y_{init} ← Initial lateral position of the vehicle
$y_{target} \leftarrow y_{init}$
$lane_{wid}$ ← Width of lane markings
ind ← index for referencing of Bezier Curve points
Path Planning
for every one second
  $width_{act}$ ← actual width of the obstacle
  $x_{target} \leftarrow targetCalc(\dot{x}_{ego}, x_{init})$
  (x_{obs}, y_{obs}) ← Obstacle position from sensors
  if $(x_{obs} > x_{target}$ && $y_{obs} < lane_{wid})$
    $path(x,y) \leftarrow A\_Star(x_{init}, y_{init}, x_{target}, x_{obs}, y_{obs}, width_{act})$
  end
end
Trajectory Generation
for every 0.01 second
  $y \leftarrow path(y)(1{:}4)$
  $y_{prev} \leftarrow unitDelay(y)$
  y_{pos} ← actual y co-ordinate of the vehicle
  $\dot{x}_{ego}$ ← longitudinal ego vehicle velocity
  if $abs(y - y_{prev})$!= 0 **then**
    ind ← reset index variable when new points are available
    $(y_{ref}, \psi_{ref}) \leftarrow Bezier(y, y_{pos}, y_{init}, \dot{x}_{ego}, ind)$
  end
end

Table 2: targetCalc($\dot{x}_{ego}, x_{init}$)

x_{obs} ← x co-ordinate of obstacle position
t_{gap} ← Time gap to lead vehicle/obstacle
D_{static} ← Gap required to maintain between 2 static vehicles
x_{min} ← Min planning length when no obstacle is present
x_{max} ← Max planning length when obstacle is present
if $x_{obs} \; ! = 0$ **then**
  $x_{target} = x_{init} + \min((D_{static} + (\dot{x}_{ego} * t_{gap})), x_{max})$
else
  $x_{target} = x_{init} + x_{min}$
end

Table 3: Bezier($y, y_{pos}, y_{init}, \dot{x}_{ego}, ind$)

t_{disc} ← discretization interval of t
$y_{points} \leftarrow [y_{pos} : (y(4) - y_{pos})/3 : y(4)]$
t ← $[0 : t_{disc} : 1]$
ind ← $min((ind + 1), length(t))$
if $length(y_{points}) == 4$ **then**
  $p_y \leftarrow \sum_{i=0}^{3} \left(\binom{3}{i} t^i (1-t)^{3-i} * y_{points}(i) \right)$
  $y_{ref} \leftarrow p_y(ind)$
  $\psi_{ref} \leftarrow \tan\left(\frac{py(ind) - py(ind-1)}{\dot{x}_{ego} * 0.01} \right)$
else
  $y_{ref} \leftarrow y_{init}$
  $\psi_{ref} \leftarrow 0$
end

As described in the algorithms, while there is a deviation of 1m in the path, the trajectory generator runs more iterations to generate all the points of the Bezier curve. This is shown clearly in Figure 2.4 where the trajectory generator had to be run faster than the path planner to produce the same lateral deviation as that of the path planner. The first four nodes of the optimum path obtained from the path planner are also shown in Figure 2.4.

To ensure a smooth transition from a previous Bezier curve to the new curve, the actual lateral position of the ego vehicle is selected as the starting point and a curve is generated from that point using the new waypoints provided by the path planner. This is shown in the trajectory generator pseudocode shown in Table 3.

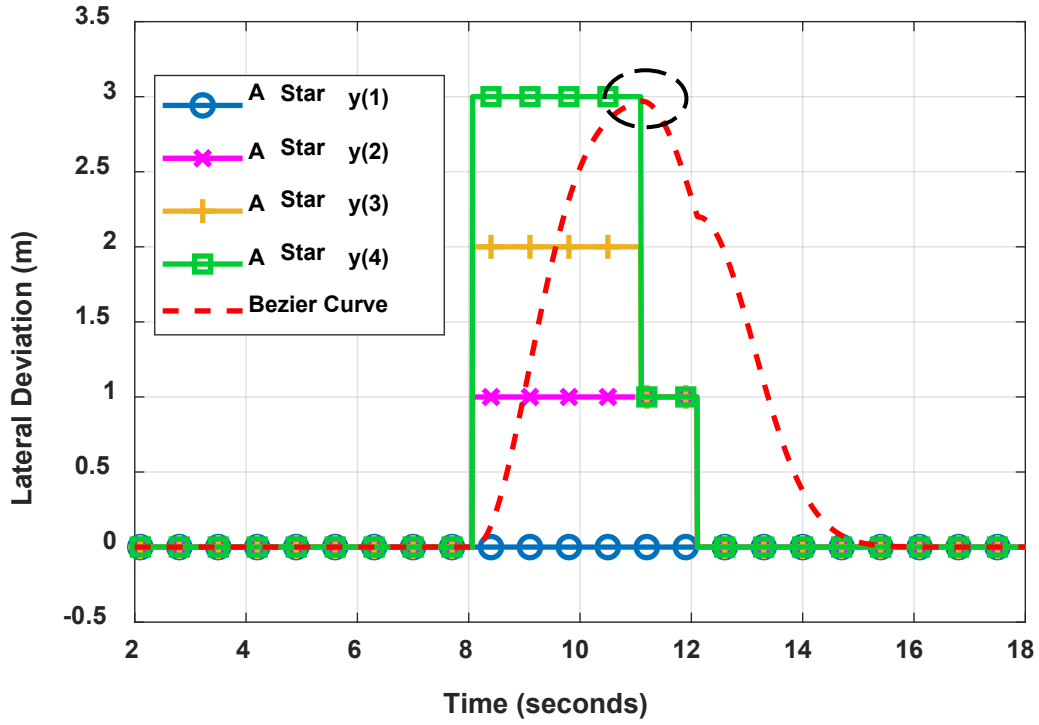

Figure 2.4: Comparison of Path Planner and Trajectory Generator Output – The marked portion highlights the joining of two different Bezier curves when there is a change in the path planner output.

This method is implemented only to determine the desired lateral position of the vehicle to avoid obstacles. By considering only the lateral motion of the path planner and by deploying a Bezier curve trajectory generator, we make sure that the reference lateral deviation output is applicable for a non-holonomic system and this reference lateral deviation can be tracked by the controller. This also allows for further control over the

longitudinal motion of the ego car, which allows the ego vehicle to follow the lead vehicle or dynamic obstacles and is also used to implement stop and go motion.

The output of the trajectory generator is passed to the Model Predictive controller as a reference for the lateral control of the vehicle. The controller is tuned such that it is capable to provide an optimum steering angle to track the reference lateral deviation. By tracking this reference trajectory, the vehicle is capable of avoiding collisions by detecting obstacles on the road and avoiding them. The prediction model used and the measured outputs and the control variables are further discussed in Chapter 3.

3 Model Predictive Vehicle Control

Model Predictive Control is used to implement lateral and longitudinal control of the ego vehicle. This method is suitable for the control of multiple-input and multiple-output (MIMO) systems by solving an optimization problem to minimize a defined cost function [32, 33]. The operating principle of MPC is to calculate the appropriate values for the control variables by solving an optimization problem to minimize a cost function [34-36]. The cost function is usually associated with the measurable states of the system being controlled and the reference to the MPC controller. The reference could be a single value for every measurable state or a trajectory for a time that is equal to the prediction horizon of the MPC controller. Using MPC, a trajectory can be tracked using the linearized prediction model. The optimization is carried out throughout the prediction horizon interval until the reference and the predicted output becomes equal at the control horizon.

By using varying weights for the various terms of the cost function, the importance of one factor over the other can be defined [37]. The rate of change of the control variables can also be controlled so that the steering and throttle changes are smooth and comfortable for the passengers. By tuning the weights and the scale factors of the MPC, the behavior and the harshness of the control can be varied. Different driving modes can be assigned with different weights and scaling factors thereby distinguishing the driving behavior from one mode to another. The first step for the implementation of the MPC controller is the formulation of the prediction model.

The material contained in this chapter has been accepted for publication to the International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, ASME, 2019.

3.1 Prediction Model

To increase computational efficiency, it is advisable to use a simple prediction model with lesser dimensions [38]. The prediction model is a linearized approximation of the vehicle plant model that is used to predict the states of the vehicle and the control variable value is appropriately calculated such that the cost function is minimized. The predicted states and the reference states are matched as the cost function decreases. Therefore a simplified linear plant model is constructed for predicting the future states in the controller. This prediction model is a combination of the longitudinal and lateral motion control equations of the vehicle. The longitudinal control model is used to calculate the normalized tractive force values while the lateral control model is used to calculate the appropriate steering angle. Figure 3.1 gives an overview of the MPC-based longitudinal and lateral control system implemented in this research.

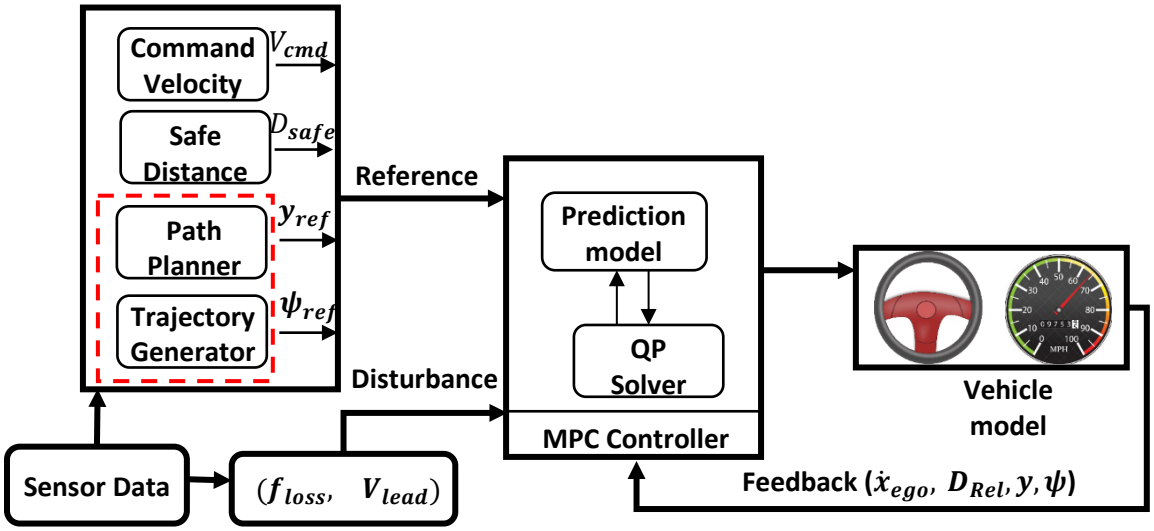

Figure 3.1: Overview of MPC based Longitudinal and lateral control model.

3.1.1 Longitudinal Control Model

For longitudinal control of the ego vehicle, the control variables are the normalized tractive force (F_t) required to achieve the acceleration/deceleration ($\ddot{x}_{desired}$) as defined by the

controller. The relation between the tractive force and the acceleration is shown in eqn. (3.1). The normalized value of the tractive force is used making the controller robust for use in any kind of powertrain system. Positive tractive force demand implies an acceleration request to the controller and negative tractive force implies braking request. The actual acceleration of the vehicle deviates from the desired acceleration with a time constant τ [39] as shown in eqn. (3.1).

$$F_t = m\ddot{x}_{desired}$$
$$\ddot{x}_{actual} = \frac{1}{\tau s + 1}\ddot{x}_{desired} \tag{3.1}$$

Equation (3.1) is used to account for the lag in the lower level throttle controller that also accounts for the lag due to the backlash in the dynamic components of the powertrain and the throttle system.

The longitudinal position of the ego vehicle is represented by x, that is measured from a reference point, which is the initial position of the ego vehicle with respect to PreScan® coordinates. The mass of the vehicle is represented by m and the frontal area of the vehicle is represented by A. Using the drag coefficient (C_d) , air density (ρ) and the coefficient of rolling friction (μ_r) the acceleration losses due to air resistance and rolling friction represented as f_{loss} is determined. The longitudinal acceleration of the vehicle ($\ddot{x}_{ego}$) is determined as shown in eqn. (3.2).

$$\ddot{x}_{ego} = \ddot{x}_{actual} - \left(\frac{f_{loss}}{m}\right)$$
$$f_{loss} = \mu mg + 0.5\rho C_d A\dot{x}_{ego}{}^2 \tag{3.2}$$

Integrating $\ddot{x}_{ego}$ over the sampling time gives the velocity of the ego vehicle ($\dot{x}_{ego}$). The controller tracks the velocity of the ego vehicle to a command reference (V_{cmd}).

The velocity output from the equations described above is a good approximation of the actual velocity output of the vehicle. Though this model is sufficient to accelerate the ego vehicle to a certain commanded velocity, a control strategy needs to be implemented to deal with any stationary or moving obstacles in front of the ego vehicle. To determine the distance to an obstacle and to control the velocity based on this data, the velocity of the lead vehicle (V_{lead}) is required, which is determined using the PreScan® sensor model.

The object camera sensor from PreScan® provides the relative velocity of the lead vehicle with respect to the ego vehicle velocity. By adding the ego vehicle velocity, the lead vehicle velocity can be determined. The relative velocity is integrated over the sample time to determine the relative distance (D_{rel}) between the two vehicles as shown in eqn. (3.3). The initial value of the distance (D_{init}) is obtained from the sensor.

$$D_{rel} = \int (V_{lead} - \dot{x}_{ego})\, dt \; + D_{init} \tag{3.3}$$

By combining eqns. (3.2) & (3.3), the prediction model for the longitudinal control is written in state-space representation with states $\boldsymbol{x} = [\dot{x}_{ego} \quad D_{rel} \quad \ddot{x}_{actual}]^T$ and the inputs to the system defined as $\boldsymbol{u} = [F_t]$. The resistance (f_{loss}) and lead vehicle velocity (V_{lead}) is introduced into the system as a measured disturbance and is updated at each sample time. This disturbance vector is defined as $\boldsymbol{w} = [f_{loss} \quad V_{lead}]^T$. The output vector of the system is $\boldsymbol{y} = [\dot{x}_{ego} \quad D_{rel}]^T$. The state space representation of the longitudinal system for the prediction model is shown in eqn. (3.4).

$$\dot{\boldsymbol{x}} = \boldsymbol{A_{long}x} + \boldsymbol{B_{long}u} + \boldsymbol{B_w w};$$

$$\boldsymbol{y} = \boldsymbol{C_{long}x} + \boldsymbol{D_{long}u}$$

where

$$\boldsymbol{A_{long}} = \begin{bmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & 0 & \frac{-1}{\tau} \end{bmatrix}, \tag{3.4}$$

$$\boldsymbol{B_{long}} = \begin{bmatrix} 0 \\ 0 \\ 1/\tau m \end{bmatrix}; \quad \boldsymbol{B_w} = \begin{bmatrix} -1/m & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix},$$

$$\boldsymbol{C_{long}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \qquad \boldsymbol{D_{long}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix};$$

The output of the system represented in eqn. (3.4) is the velocity of the ego vehicle ($\dot{x}_{ego}$) and the relative distance between the ego vehicle and the lead vehicle (D_{rel}). The reference for the ego vehicle velocity is determined from a velocity predictor function. This is just the user commanded velocity (V_{cmd}) but it is modulated when the vehicle is travelling on a curved road or is optimized to navigate through signalized intersections. The MPC control tries to reduce the difference between the reference commanded velocity and the predicted ego vehicle velocity and calculates the appropriate throttle opening value.

The relative distance is controlled using a constant time gap approach, where the controller is required to maintain a time gap of T_{gap} from the lead vehicle [40]. The distance required to maintain the given time gap is known as the safe distance (D_{safe}) that is determined using eqn. (3.5).

$$D_{safe} = D_{static} + T_{gap} * \dot{x}_{ego} \tag{3.5}$$

The static distance (D_{static}) represents the minimum gap between two stationary vehicles. It is required to maintain this gap when a vehicle is stopped due to a stop sign or a traffic light at an intersection. This safe distance is used as a reference to the MPC controller, which tries to track the relative distance to the safe distance as shown in Figure 3.2.

Figure 3.2: Representation of Relative Distance and Safe Distance

3.1.2 Lateral Control Model

To achieve lateral control, a linearized steering model is used for prediction. The steering system is approximated to a bicycle model, which simplifies the vehicle dynamics to a single track with the lateral deviation (y) and the yaw angle (ψ) as the outputs of the system [39]. A graphic representation of the bicycle model is shown in Figure 3.3. The steering angle is represented by δ and the yaw angle is represented by ψ. The distance from the CG of the vehicle to the front and rear wheels is represented by l_f and l_r, respectively. The cornering stiffness of the front and rear tires are represented by C_f *and* C_r, respectively and the moment of inertia of the vehicle with respect to the yaw axis is represented by I_z. The lateral velocity of the vehicle is given by $\dot{y}_{ego}$ and the lateral deviation is represented by y. The prediction model is implemented with the states $\boldsymbol{x} = \begin{bmatrix} \dot{\psi} & \psi & \dot{y}_{ego} & y \end{bmatrix}^T$ and inputs to the system as $\boldsymbol{u} = [\delta]$. The output of the prediction model is $\boldsymbol{y} = [y \quad \psi]^T$. The prediction model for the lateral vehicle motion in state-space representation is shown in eqns. (3.6).

$$\dot{\boldsymbol{x}} = \boldsymbol{A_{lat}x} + \boldsymbol{B_{lat}u}, \qquad \boldsymbol{y} = \boldsymbol{C_{lat}x} + \boldsymbol{D_{lat}u}$$

where

$$\boldsymbol{A_{lat}} = \begin{bmatrix} a_1 & 0 & a_2 & 0 \\ 0 & 0 & 1 & 0 \\ a_3 & 0 & a_4 & 0 \\ 1 & \dot{x}_{ego} & 0 & 0 \end{bmatrix}, \quad \boldsymbol{B_{lat}} = \begin{bmatrix} 2 * \frac{C_f}{m} \\ 0 \\ 2 * C_f * \frac{l_f}{I_z} \\ 0 \end{bmatrix}$$

$$\boldsymbol{C_{lat}} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \qquad \boldsymbol{D_{lat}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.6}$$

$$a_1 = -\frac{\frac{2 * Cf + 2 * Cr}{m}}{\dot{x}_{ego}}; \qquad a_2 = -\dot{x}_{ego} - \frac{\frac{2 * Cf * lf - 2 * Cr * lr}{m}}{\dot{x}_{ego}};$$

$$a_3 = -\frac{\frac{2 * Cf * lf - 2 * Cr * lr}{Iz}}{\dot{x}_{ego}}; \qquad a_4 = -\frac{\frac{2 * Cf * lf^2 + 2 * Cr * lr^2}{Iz}}{\dot{x}_{ego}}$$

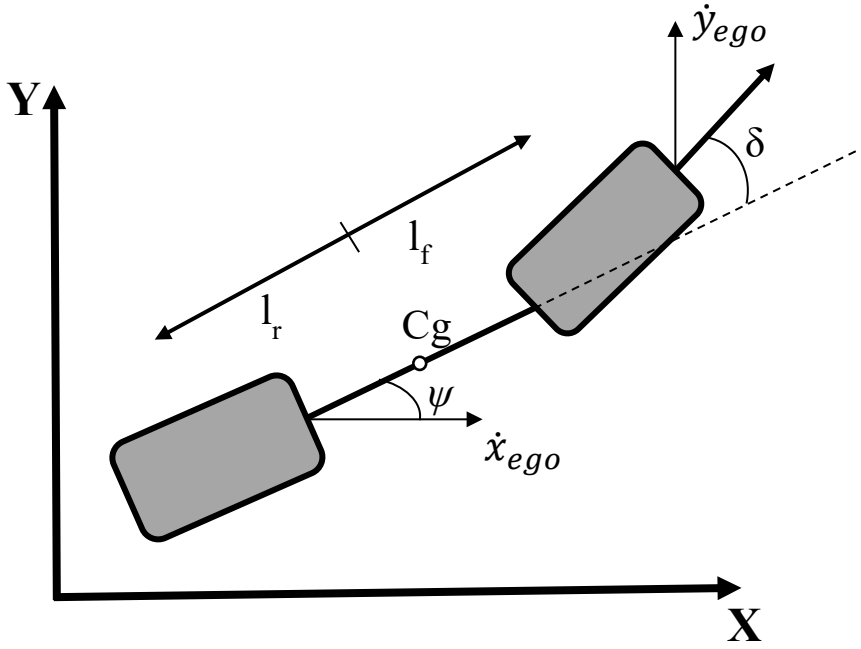

Figure 3.3: Representation of the dynamic bicycle model for lateral control [39]

To implement the prediction model in a single MPC controller, the state-space equations from eqns. (3.4) & (3.6) are combined to form a single state-space representation as shown in eqn. (3.7) with states $\boldsymbol{x} = [\dot{x}_{ego} \quad D_{rel} \quad \ddot{x}_{actual} \quad \dot{\psi} \quad \psi \quad \dot{y}_{ego} \quad y]^T$ and the control inputs as $\boldsymbol{u} = [F_t \quad \delta]^T$. The disturbance to the system is represented as $\boldsymbol{w} = [f_{loss} \quad V_{lead}]^T$. The output state vector of the system is $\boldsymbol{y} = [\dot{x}_{ego} \quad D_{rel} \quad y \quad \psi]^T$.

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{B}_{\boldsymbol{w}}\boldsymbol{w}, \qquad \boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u} \tag{3.7}$$

where

$$\boldsymbol{A} = \begin{bmatrix} A_{long} & zeros(3,4) \\ zeros(4,3) & A_{lat} \end{bmatrix};$$

$$\boldsymbol{B} = \begin{bmatrix} B_{long} & zeros(3,1) \\ zeros(4,1) & B_{lat} \end{bmatrix};\ \boldsymbol{B_w} = \begin{bmatrix} -1/m & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix};$$

$$\boldsymbol{C} = \begin{bmatrix} C_{long} & zeros(2,4) \\ zeros(2,3) & C_{lat} \end{bmatrix};\ \boldsymbol{D} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix};$$

3.2 Problem Formulation

The prediction model is converted to a discrete-time model in the controller using the first-order hold method [41] and is represented as shown in eqn. (3.8).

$$\begin{aligned} \boldsymbol{x_{t+1}} &= \boldsymbol{A_d x_t} + \boldsymbol{B_d u_t} + \boldsymbol{B_{w_d} w_t}; \\ \boldsymbol{y_t} &= \boldsymbol{C_d x_t} + \boldsymbol{D_d u_t}; \end{aligned} \tag{3.8}$$

The control variables (u_t) i.e. the tractive force and the steering wheel angle are determined by solving an optimization problem for each time step over the control horizon (t_c) with a cost function defined in eqn. (3.9).

$$\begin{aligned} \boldsymbol{\mathcal{H}}: \sum_{i=1}^{t_c} \Big(& w_1 \left| \dot{x}_{ego_i} - V_{cmd} \right|^2 + w_2 \left| D_{Rel_i} - D_{safe} \right|^2 + w_3 \left| y_i - y_{ref} \right|^2 \\ & + w_4 \left| \psi_i - \psi_{ref} \right|^2 + w_{r1} \left| \Delta F_{t_i} \right|^2 + w_{r2} |\Delta \delta_i|^2 \Big) \end{aligned} \tag{3.9}$$

In eqn. (3.9), w_1, w_2, w_3 & w_4 represent the weights of the predicted outputs and w_{r1} & w_{r2} are the weights for the rate of change of the control variables ($\Delta F_t \; and \; \Delta\delta$) i.e., the change in the tractive force and steering angle. The constraints for the optimization problem are given in eqns. (3.10).

$$D_{Rel_i} \geq D_{safe}$$

$$F_{t_{min}} \leq F_{t_i} \leq F_{t_{max}}$$

$$\Delta F_{t_{min}} \leq \Delta F_{t_i} \leq \Delta F_{t_{max}} \tag{3.10}$$

$$\delta_{min} \leq \delta \leq \delta_{max}$$

$$\Delta\delta_{min} \leq \Delta\delta_i \leq \Delta\delta_{max}$$

The minimum and maximum values for the control variables are chosen based on the vehicle and powertrain constraints. The rate of change limits is selected such that the steering angle change and the acceleration change are within the permissible limits for passenger comfort. The selected values are summarized in Table 5. MATLAB®, Simulink®, and Model Predictive Toolbox™ [40, 42] were used to solve the optimization problem with a QP solver.

3.3 Closed-Loop Implementation

The solution to the optimization problem yields the control signals i.e. the tractive force demand (F_t) and the optimum steering wheel angle (δ) required to minimize the cost function shown in eqn. (3.8). These signals are passed to the vehicle model that calculates the vehicle states, which are used as the feedback for solving the optimization problem for the next time step. The feedback signals include vehicle states such as vehicle

velocity ($\dot{x}_{ego}$), relative distance to obstacle (D_{rel}), the lateral deviation from the reference (y) and the yaw angle of the ego vehicle (ψ).

The reference for the longitudinal control is determined from the user-defined command velocity (V_{cmd}) or the velocity predictor and the safe distance (D_{safe}) calculator. The reference for the lateral control is determined from the path planner and the trajectory generator.

Simulations are carried out on the PreScan® platform to introduce traffic characteristics and to visualize the performance of the controller for varying traffic scenarios that are discussed further in section 4.

4 Simulation Results

The controller is tested for varying traffic scenarios using various static and dynamic obstacles from the PreScan® library. The platform creates a run-time environment to execute these various traffic scenarios and a model-in-the-loop setup is established where the controller is tested.

4.1 Scenario Setup

To test the control algorithm, a real-time traffic scenario is modeled using the PreScan® software. This platform allows the user to create any kinds of roads, road signs, lane markings, traffic signals, and other cars. Using Open Street Maps, the roads and environment of any place can be modeled using the software. The controller can also be tested for various weather and lighting conditions. A sample scenario that is created using PreScan® software is shown in Figure 4.1.

Various types of vehicles ranging from motorcycles to trailer trucks can be modeled and simulated as required. This research uses the Toyota Prius model available in the PreScan library as the ego vehicle. The vehicle model is tweaked to make it more representative of the actual vehicle. Values of the vehicle parameters used in the prediction model of the MPC controller is shown in Table 4. The complete vehicle dynamics parameters are shown in Figure 4.2. All the other parameters pertaining to tire force estimation and vehicle pitch and roll calculation is used in the vehicle plant model, which is used for simulation to test the controller.

The material contained in this chapter has been accepted for publication to the International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, ASME, 2019.

Figure 4.1: Sample scenario that can be created on PreScan software

Table 4: Vehicle parameters for Toyota Prius

Parameter	**Value**
Mass (m)	1650 kg
Wheelbase	2.90 m
Cornering Stiffness – Front (C_f)	66479 N/rad
Cornering Stiffness – Rear (C_r)	110068 N/rad
Moment of Inertia – yaw (I_z)	3269 kg m^2
Steering Angle (max)	-565°, 565°
CG distance – Front (l_f)	1.16 m
CG distance – Rear (l_r)	1.74 m

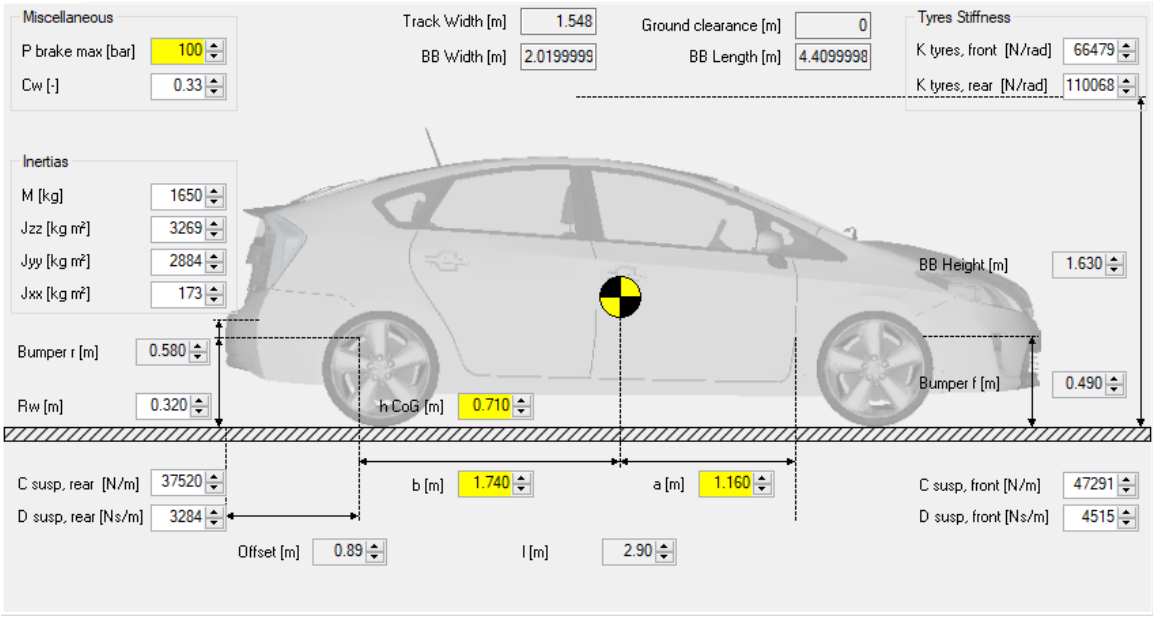

Figure 4.2: Vehicle Dynamics Parameters for the Toyota Prius Model

4.2 Simulation Setup

The working of the path-planning algorithm and the lateral controller is demonstrated using a scenario, in which a stationary obstacle is blocking the path of the ego vehicle. For simulation and controller tuning, the Toyota Prius vehicle model was used and the constraints for the steering wheel and the acceleration were decided accordingly. The rate constraints are decided such that there are no harsh pitch and roll accelerations for maximum passenger comfort. The weight coefficients in eqn. (3.9) are tuned such that the most critical factor requires more weightage as it is essential for the safe operation of the vehicle. The minimum and maximum values of the control variables and the weights for the cost function are shown in Table 5. The obstacle is detected through the vision sensors as shown in Figure 4.4

The vehicle tries to stay in the center of the path until the obstacle is detected. The path is a straight road and the goal of the experiment is to observe the maneuver planned by the ego vehicle to avoid a collision by deviating from a straight line path. The longitudinal

control algorithm modulates the throttle and brake values to control the speed of the ego vehicle such that the vehicle slows down when the obstacle is approaching and tries to reach the command velocity once the collision avoidance maneuver is complete. Cameras are added at different locations on the vehicle to visualize the movement of the vehicle. The Vis-Viewer interface is shown in Figure 4.3.

Figure 4.3: Vis-Viewer Setup for visualizing the ego vehicle

Table 5: Simulation and model parameters for MPC controller

Parameter	Value
Weighting factor (w_1)	980
Weighting factor (w_2)	125000
Weighting factor (w_3)	103931
Weighting factor (w_4)	0.1
Rate Weights (w_{r1} , w_{r2})	750, 0.09
$F_{t_{min}}$, $F_{t_{max}}$	-100%, 100%
δ_{min}, δ_{max}	-565°, 565°
$\Delta F_{t_{min}}$, $\Delta F_{t_{max}}$	-1%, 1%
$\Delta\delta_{min}$, $\Delta\delta_{max}$	-56°, 56°

Figure 4.4: Object detected in the path of the ego vehicle through the vision sensor

4.3 Scenario 1 – Stationary Obstacle

The ego vehicle with an initial velocity of 5m/s accelerates to reach the command velocity. The object is first detected when it is at a distance of 100m from the ego vehicle as shown in Figure 4.5. When the relative distance of the obstacle reaches the safe distance as highlighted by the arrow in Figure 4.5, the vehicle starts to brake to maintain a safe distance with the obstacle. When the relative distance falls below the safe distance highlighted using the dashed line, the path-planning algorithm is triggered and a maneuver is planned and executed to avoid a collision. The reference trajectory generated from the planner and the actual vehicle trajectory obtained is shown in Figure 4.7. The steering maneuver implemented by the controller is shown in Figure 4.8. Once the vehicle is at its maximum lateral deviation, the obstacle is no longer in the path of the ego vehicle and hence the obstacle is avoided as shown in Figure 4.5.

During the object avoidance maneuver, the longitudinal control algorithm is also changing the tractive force percentage demand to control the velocity of the ego vehicle based on the difference between the relative distance and the safe distance. As the relative distance is decreasing, the tractive force demand becomes negative as the vehicle is braking to maintain a safe distance from the obstacle. The tractive force values implemented by the controller is shown in Figure 4.9. As the vehicle is slowing down, the steering maneuver also takes place to avoid a collision. Once the ego vehicle deviates from the path of the obstacle i.e. at maximum lateral deviation, the tractive force is adjusted so that the ego velocity tracks the commanded velocity and stabilizes as it approaches the command velocity as shown in Figure 4.6 and Figure 4.9. The object avoidance maneuver performed by the ego vehicle can be visualized using PreScan® Vis-Viewer application as shown in Figure 4.10, Figure 4.11 & Figure 4.12.

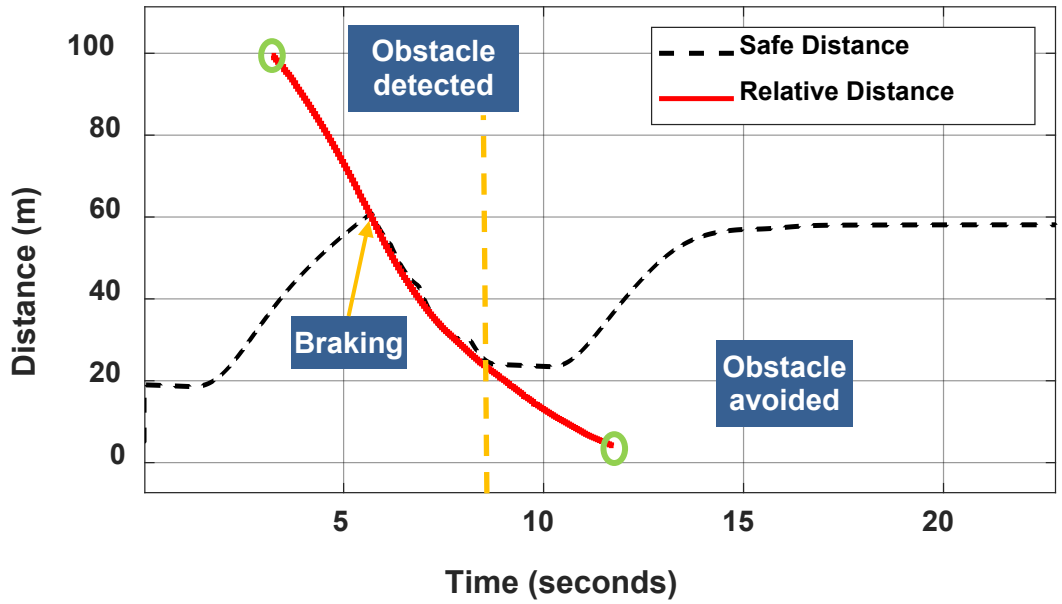

Figure 4.5: Safe distance (D_{safe}) versus relative distance (D_{Rel}) as the ego vehicle is approaching to the obstacle

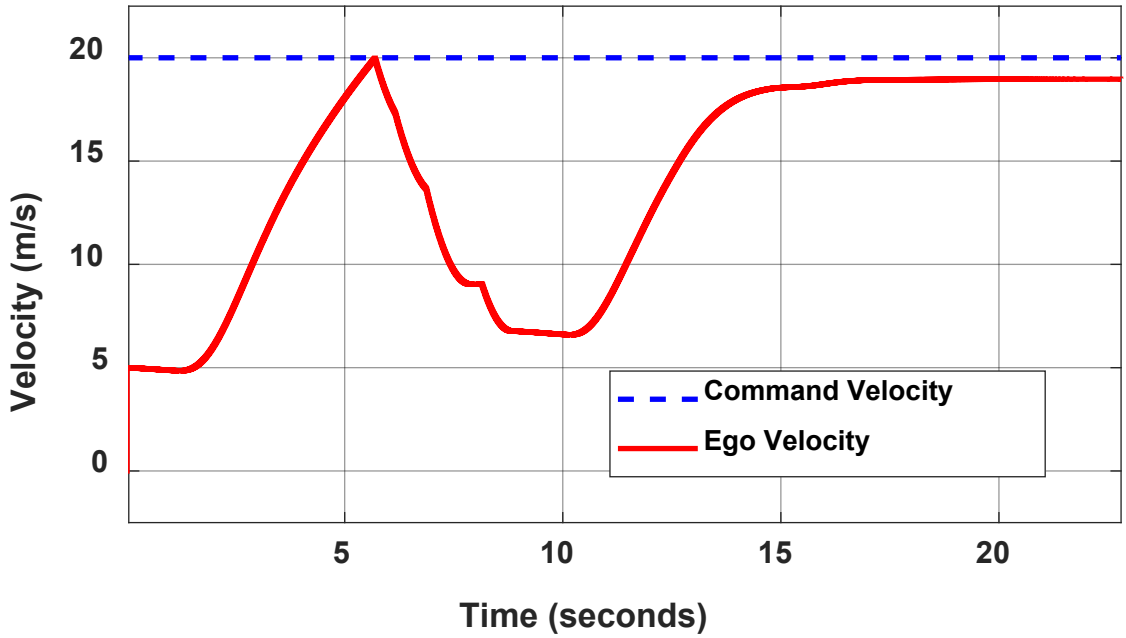

Figure 4.6: Command velocity (V_{cmd}) versus ego vehicle velocity ($\dot{x}_{ego}$) through the object avoidance maneuver

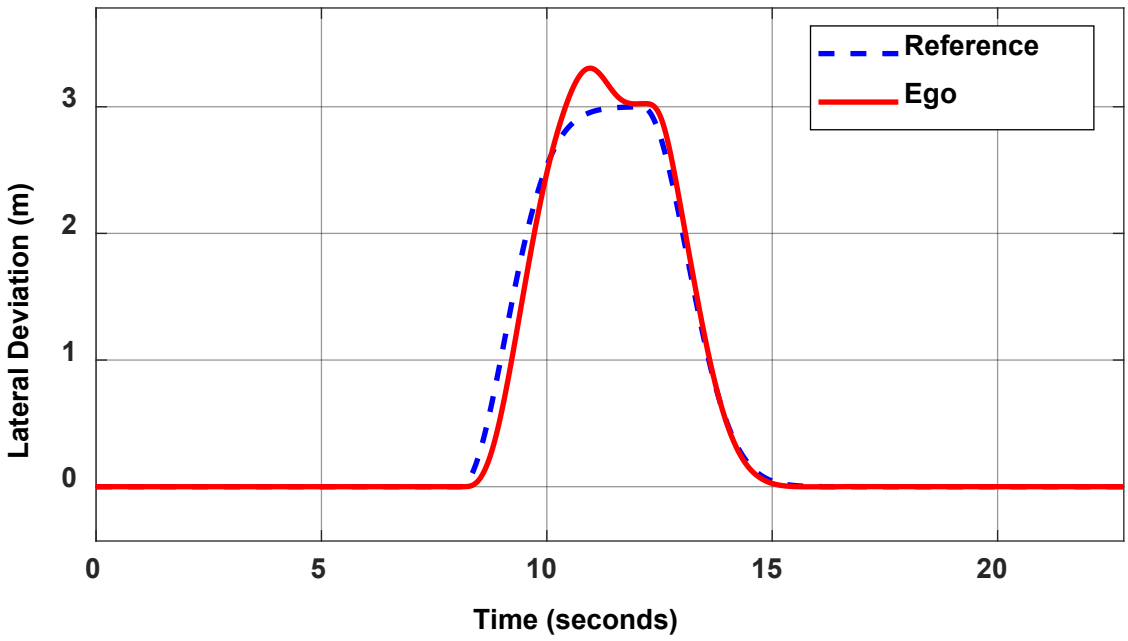

Figure 4.7: Comparison of the reference trajectory and actual vehicle trajectory obtained using the MPC controller

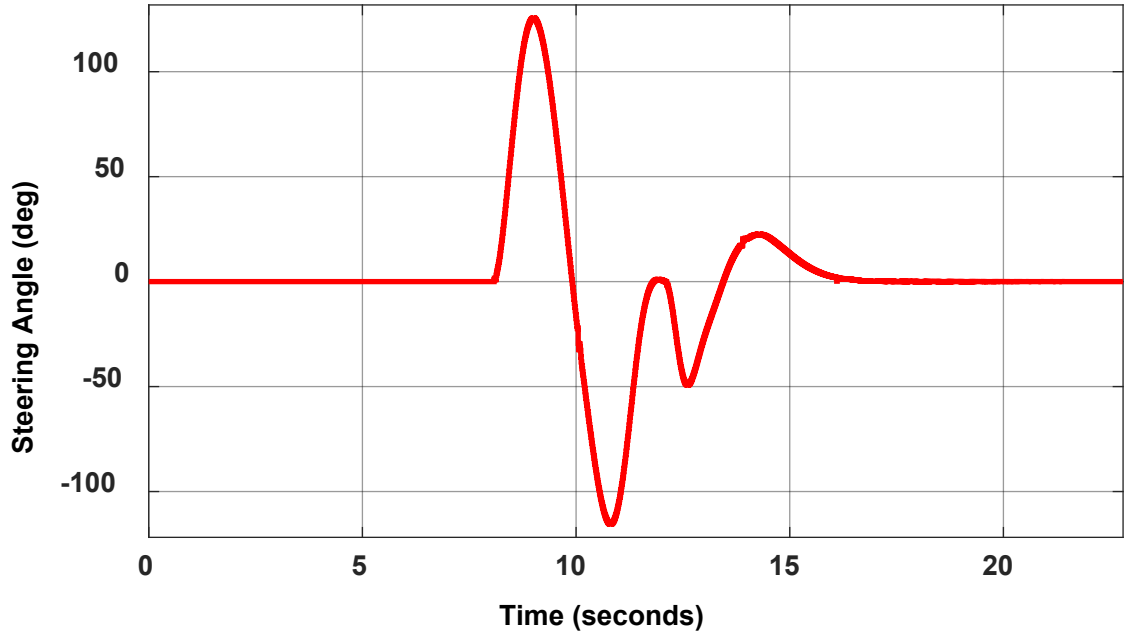

Figure 4.8: Steering wheel angle (δ) to generate the object avoidance maneuver

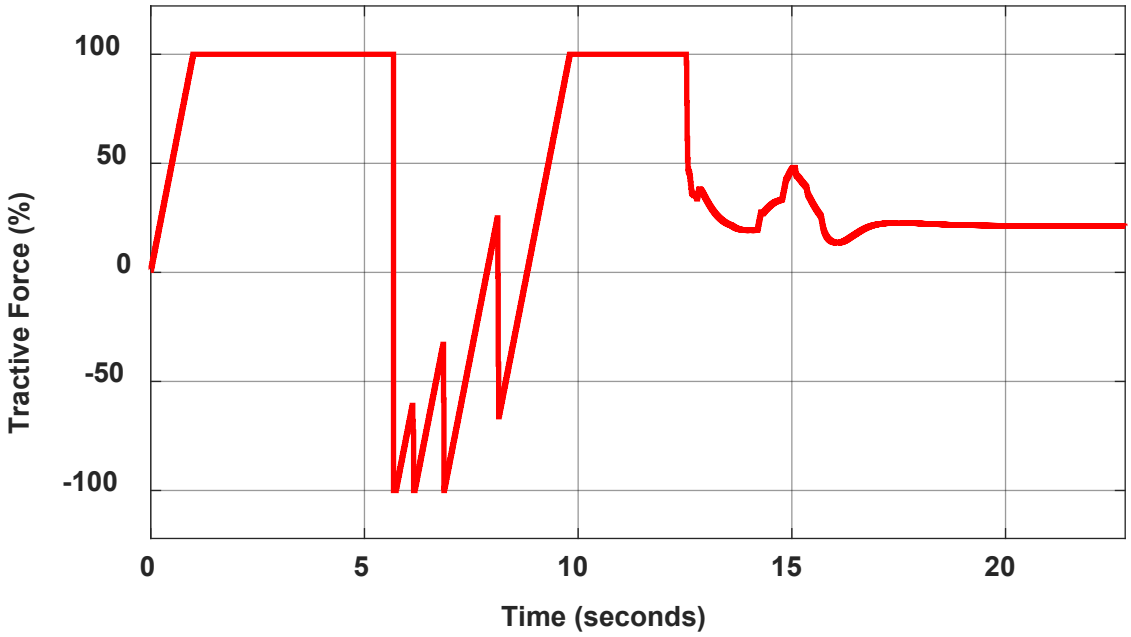

Figure 4.9: Tractive Force (F_t) implemented by MPC controller

Figure 4.10: Ego vehicle deviating from the path to avoid collision with the obstacle

Figure 4.11: Ego vehicle at the maximum deviation from the path

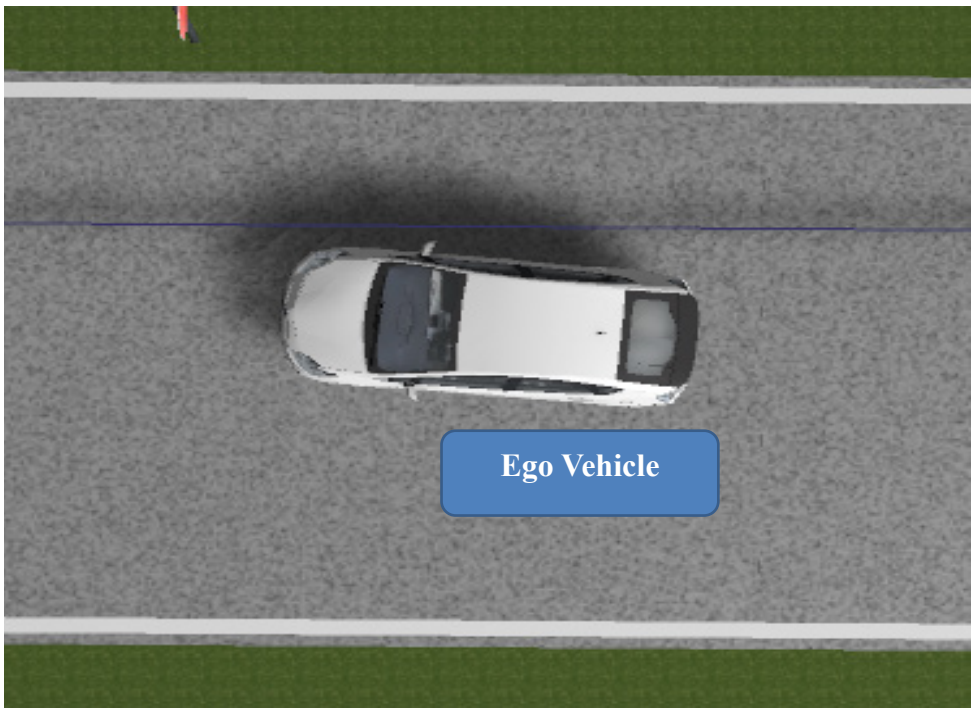

Figure 4.12: Ego Vehicle rejoining the path after crossing the obstacle

4.4 Scenario 2 – Moving Obstacle

A second test is conducted using a lead vehicle to demonstrate the working of longitudinal and lateral control in harmony. In this scenario, the lead vehicle is traveling at a constant speed of 10m/s and uncertainty is created as the lead vehicle comes to a sudden stop. The initial velocity of the ego vehicle is 5m/s. The behavior of the ego vehicle is observed as it is following the lead vehicle. As soon as the relative distance becomes less than the safe distance, braking is initiated as the controller tries to match the velocity of ego vehicle to the lead vehicle velocity to maintain a constant gap as shown in Figure 4.13 and Figure 4.14. The tractive force values obtained from the controller is shown in Figure 4.17. Once the lead vehicle stops suddenly, the relative distance decreases further and the tractive force demand becomes negative. The relative distance falls below the safe distance as highlighted by the dashed line in Figure 4.13 and the path planner creates a maneuver for the ego vehicle to pass the lead vehicle thereby preventing a collision. The reference trajectory generated by the path planner and the actual trajectory executed by the vehicle are shown in Figure 4.15 and the steering angle values implemented to complete the maneuver are shown in Figure 4.16. Once the maneuver is completed, the ego vehicle accelerates to reach the command velocity as shown in Figure 4.14 and the tractive force values used to reach this steady state are shown in Figure 4.17.

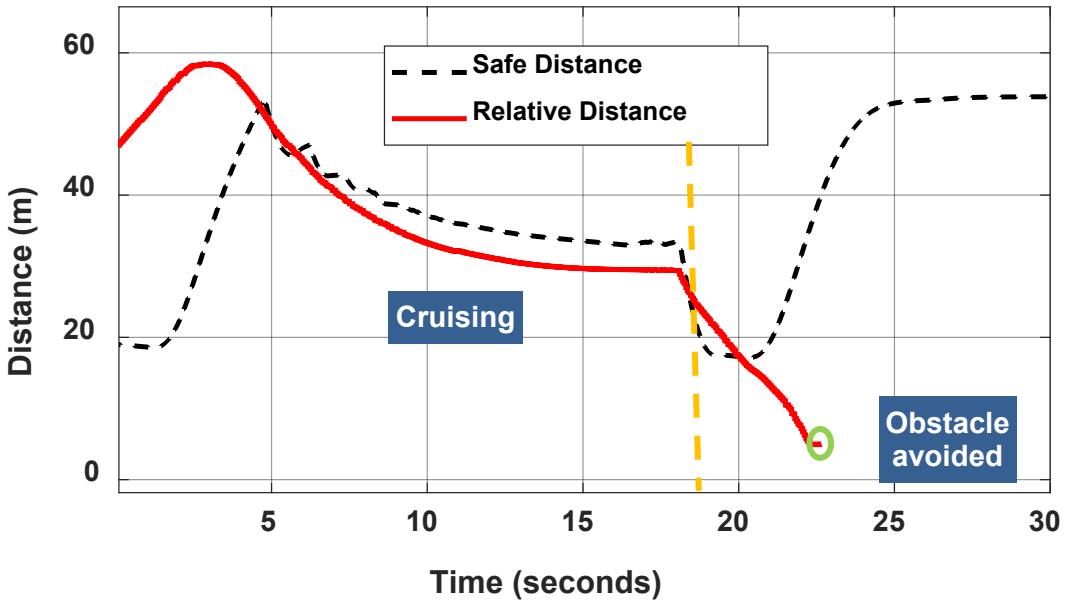

Figure 4.13: Safe distance (D_{safe}) versus relative distance (D_{Rel}) as the ego vehicle approaches the obstacle

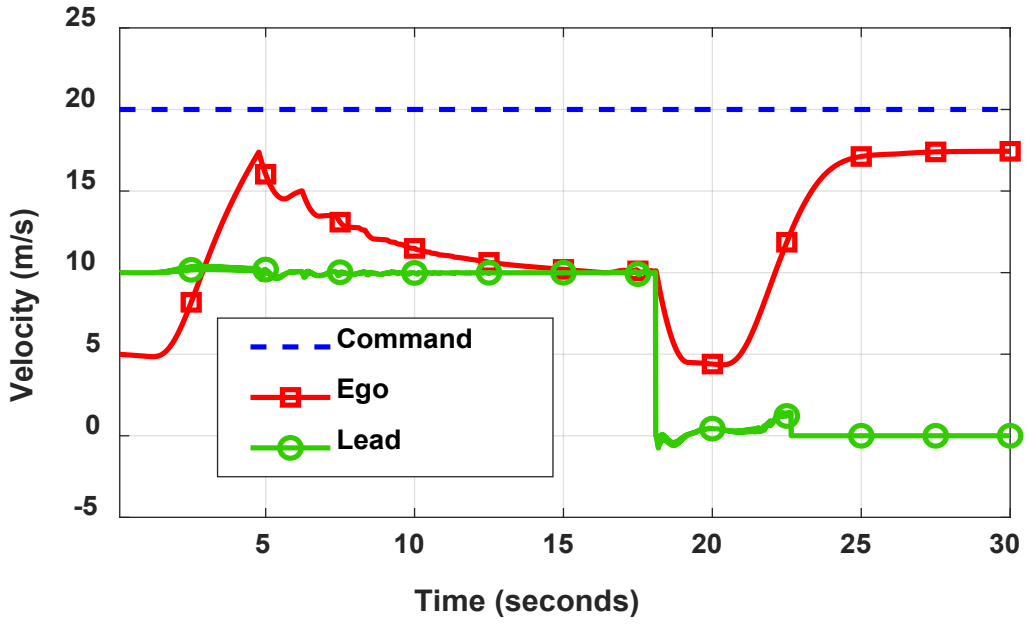

Figure 4.14: Comparison of reference velocity, ego velocity ($\dot{x}_{ego}$) and lead velocity (V_{lead}) while following and passing

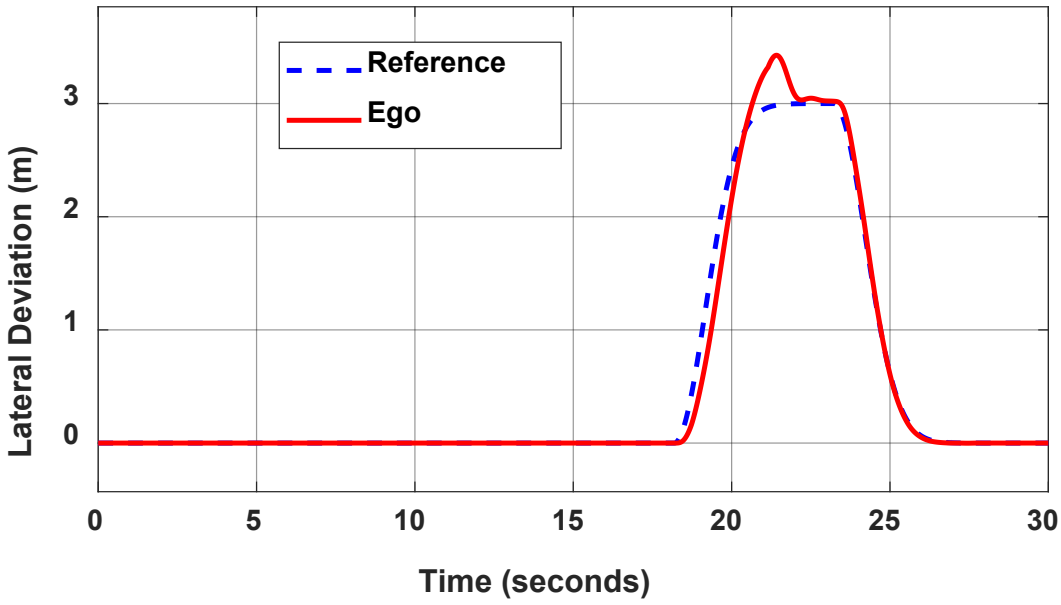

Figure 4.15: Comparison of the reference trajectory and actual vehicle trajectory obtained to avoid a collision

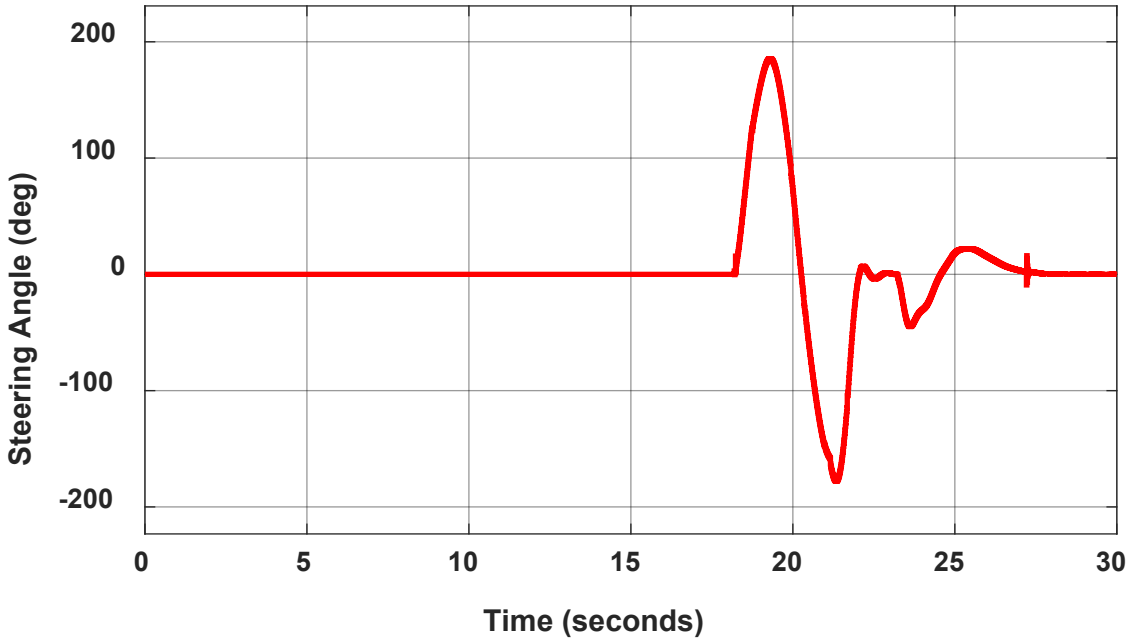

Figure 4.16: Steering wheel angle (δ) for the object avoidance maneuver

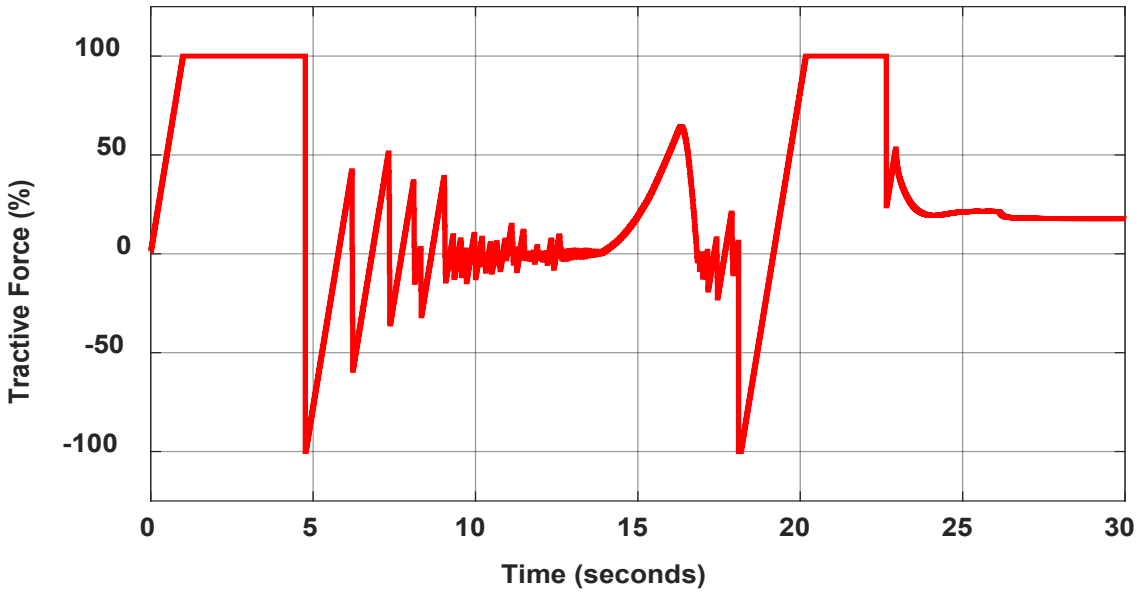

Figure 4.17: Tractive force (F_t) implemented by the MPC controller

4.5 Scenario 3 – Multiple Obstacles

In this test, a multi-lane road is used with multiple objects to demonstrate the obstacle avoidance maneuver. The obstacles are simulated using trucks rather than cars to understand the behavior of the path planner as it takes more time to pass a truck due to their long wheelbase. Three trucks are placed among which two of them are directly in the path of the ego vehicle and one of them is in an adjacent lane as shown in Figure 4.18. The horizontal line in Figure 4.18 indicates the desired path of the ego vehicle. The red dashed rectangles indicate the position of the trucks on the multi-lane road. With an initial ego velocity ($\dot{x}_{ego}$) of 5m/s the ego vehicle starts moving. The command velocity (V_{cmd}) for the ego vehicle is 20m/s as shown in Figure 4.19. The ego vehicle accelerates as it tries to reach the command velocity (V_{cmd}) due to which the relative distance (D_{Rel}) to the obstacle starts decreasing and the safe distance (D_{safe}) is increasing as shown in Figure 4.21. The vehicle starts slowing down once the relative distance (D_{Rel}) to the truck is equal to the safe distance (D_{safe}) as highlighted in Figure 4.21.

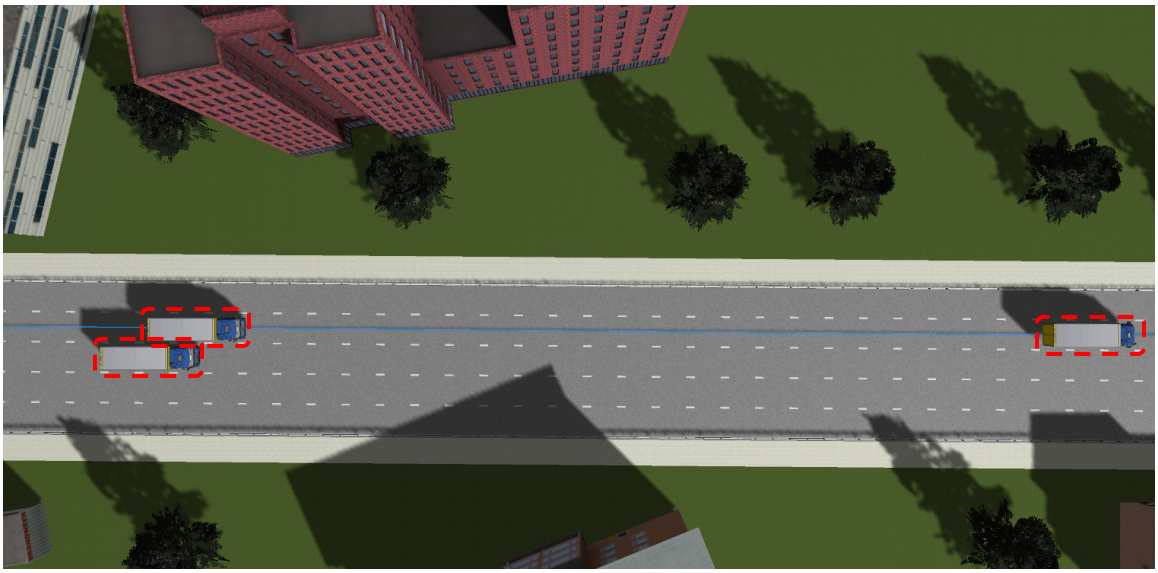

Figure 4.18: Multiple objects test scenario created using trucks

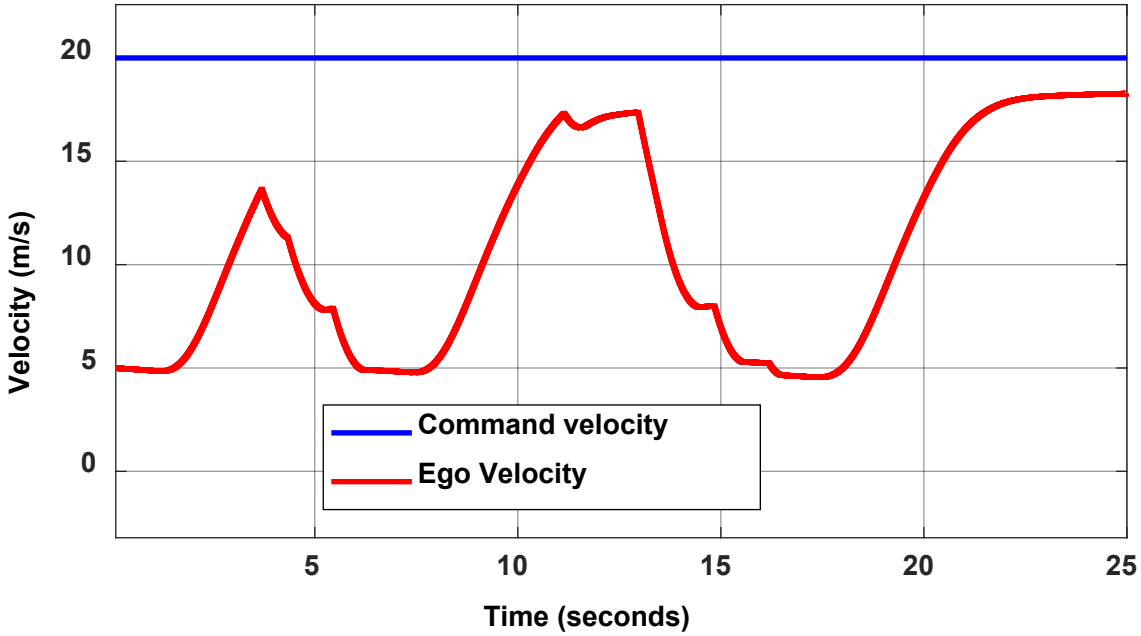

Figure 4.19: Command velocity (V_{cmd}) versus ego vehicle velocity ($\dot{x}_{ego}$) through the multiple objects avoidance maneuver

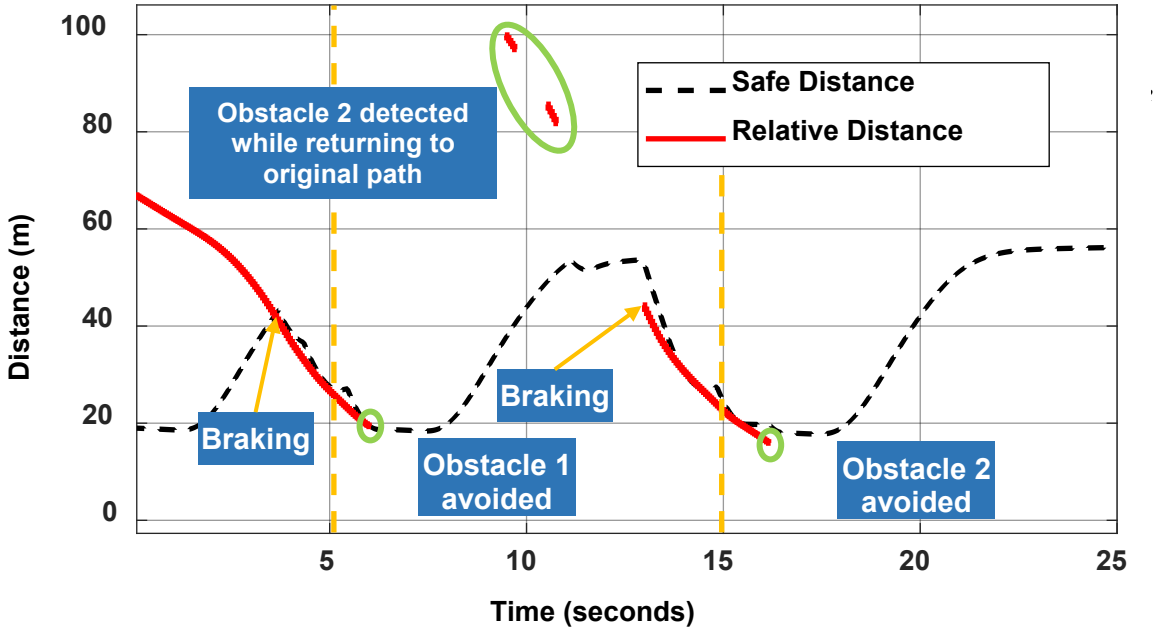

Figure 4.20: Safe distance (D_{safe}) versus relative distance (D_{Rel}) as the ego vehicle is approaching to the obstacles

The path planner and the trajectory generator are triggered when the relative distance falls below the safe distance and a trajectory is planned and the object avoidance maneuver is executed as shown in Figure 4.22. The avoidance maneuver for each of the obstacle is highlighted in Figure 4.22. It can be seen in this figure that the planner is capable of planning a trajectory for avoiding obstacles in both directions. For the first obstacle, the algorithm planned a path with positive lateral deviation due to the presence of the third truck adjacent to the lane of the ego vehicle path. The path planning algorithm takes into account the location of both these obstacles and a path is planned to avoid both the obstacles. Once the vehicle returns to the original path, the second obstacle in the path of the ego vehicle is detected and a path in a different direction is planned. The change in direction is due to the positioning of the second obstacle. To complete the maneuver with minimum lateral deviation, the path planner generates a trajectory with negative lateral deviation as shown in Figure 4.22. The steering wheel angle values (δ) calculated by the MPC controller to track the reference trajectory is shown in Figure 4.23.

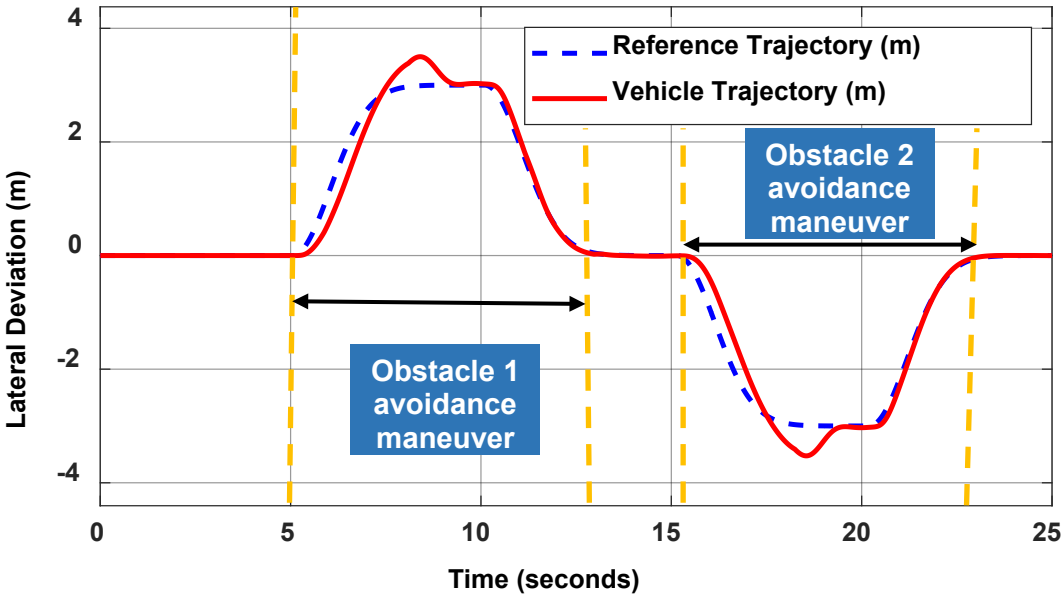

Figure 4.22: Comparison of the reference trajectory and actual vehicle trajectory obtained to avoid both the obstacles in the path

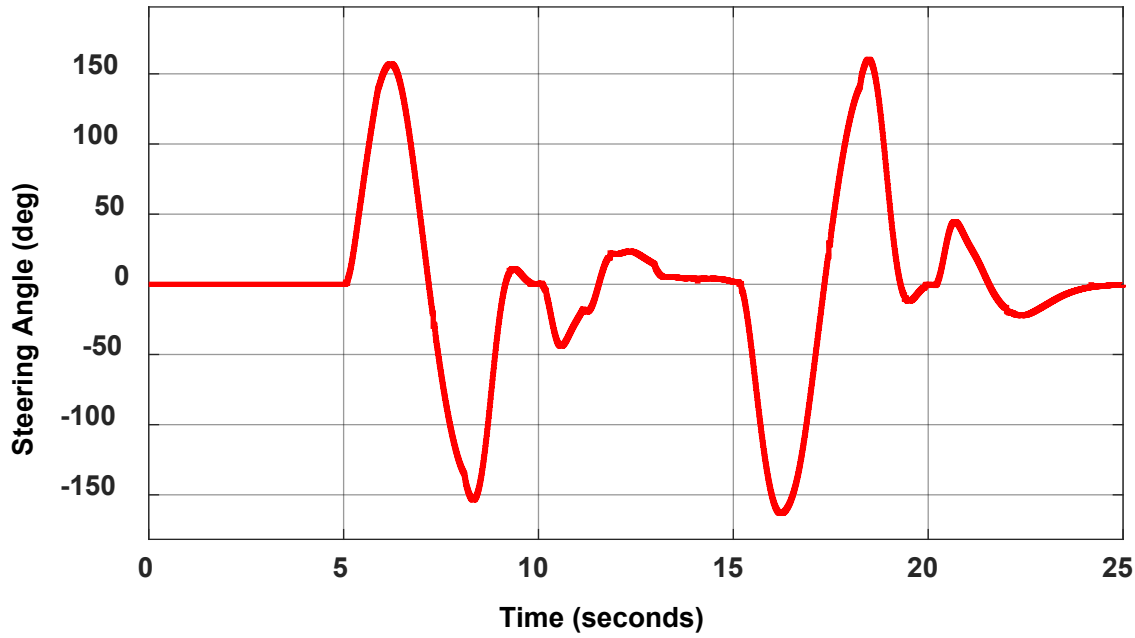

Figure 4.23: Steering wheel angle (δ) for the object avoidance maneuver for both obstacles in the path

The requested tractive force percentage (F_t) to achieve longitudinal control is shown in Figure 4.24. The braking zones for both the obstacles can be observed as highlighted in Figure 4.24.

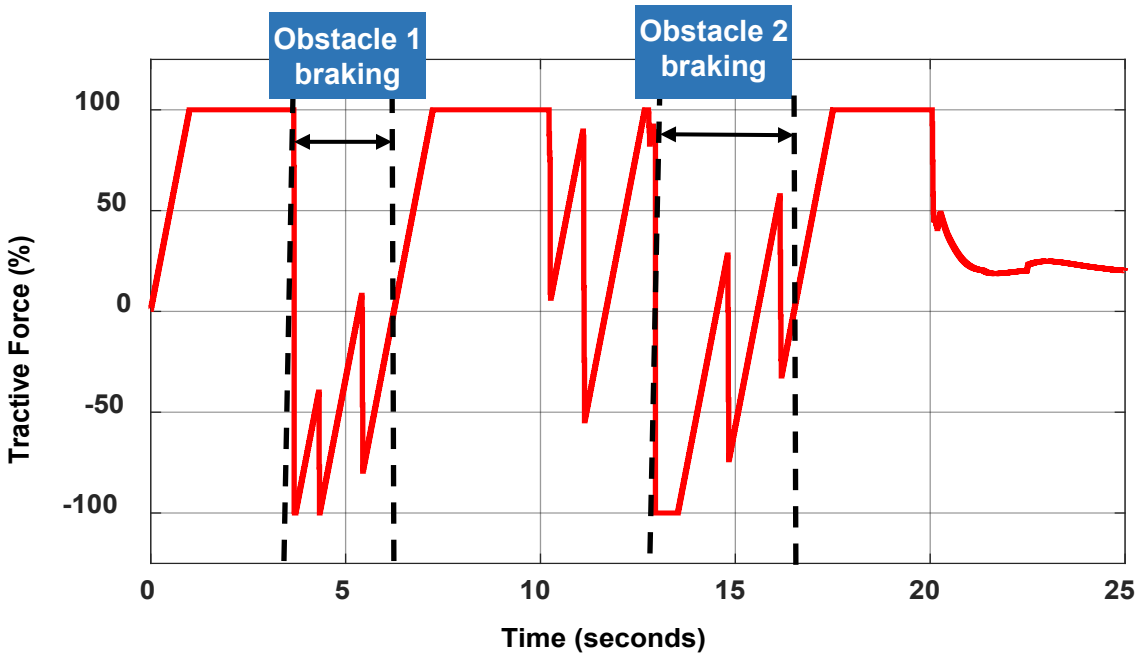

Figure 4.24: Tractive force (F_t) implemented by the MPC controller

From the above results, it can be clearly seen that by using path planning algorithms, the autonomous vehicle can safely navigate through a collision scenario or any such uncertainties. Using the connected vehicle technology, traffic signal data from road-side units are obtained via the V2X infrastructure. This data is used to achieve stop/go motion of the ego vehicle at signalized intersections. This is demonstrated in Chapter 5.

5 Approach and Departure at Signalized Intersections

5.1 Motivation

The number of vehicles on road in urban areas has been rising continuously and this directly leads to an increase in traffic congestion. This is one of the major causes of air pollution in cities as vehicles generate more emissions when they spend more time in stop and go traffic as the engine operates in idling/low speeds during which the emission levels are high. Electric vehicles could solve the issue of emissions but increased electricity consumption may lead to even more pollution as most of the electrical energy nowadays is still generated by burning coal. Increasing traffic congestions also lead to increased travel times in a stop and go traffic that is not only an inconvenience but also a cause of driver frustration, which is one of the main reasons for road accidents.

By implementing connected vehicle technologies, vehicle-to-infrastructure (V2I) communication technology can be used to transmit signal information such as signal phase and time to next phase, which can be used to stop the vehicle at a red light in a fuel efficient way. This can be further improved by modifying the algorithm in such a way that the velocity of the vehicle is adjusted such that the vehicle reaches the intersection only at a green phase. This reduces travel times and can further be improved to save energy while braking and acceleration. A model predictive approach to predict a velocity command using such traffic signal information is shown in [25].

5.2 Data Extraction from V2I transmitters

The transmitters make use of the Dedicated Short Range Communication (DSRC) technology to enable communication between the vehicles and the road-side-units. The data that is transferred is according to the SAE J2735 Message Set Dictionary standard

[43]. This standard specifies the definitive message structure and provides the message definitions.

Every DSRC message is classified into several components-

1. Message – the top level of complexity in the data structure
2. Data Frame – complex data structures
3. Data Element – smallest division of information content

The definitions of each of these components are available in the Message Set Dictionary [43]. Every Message Frame consists of several Data Frames and Data Elements. Data Frames can further be a collection of simple Data Frames or Data Elements. A figure representing a sample DSRC Message is shown in Figure 5.1. It can be seen in this figure that a Message can contain both Data Frames and Data Elements and Data Frames can further contain Data Frames and Data Elements.

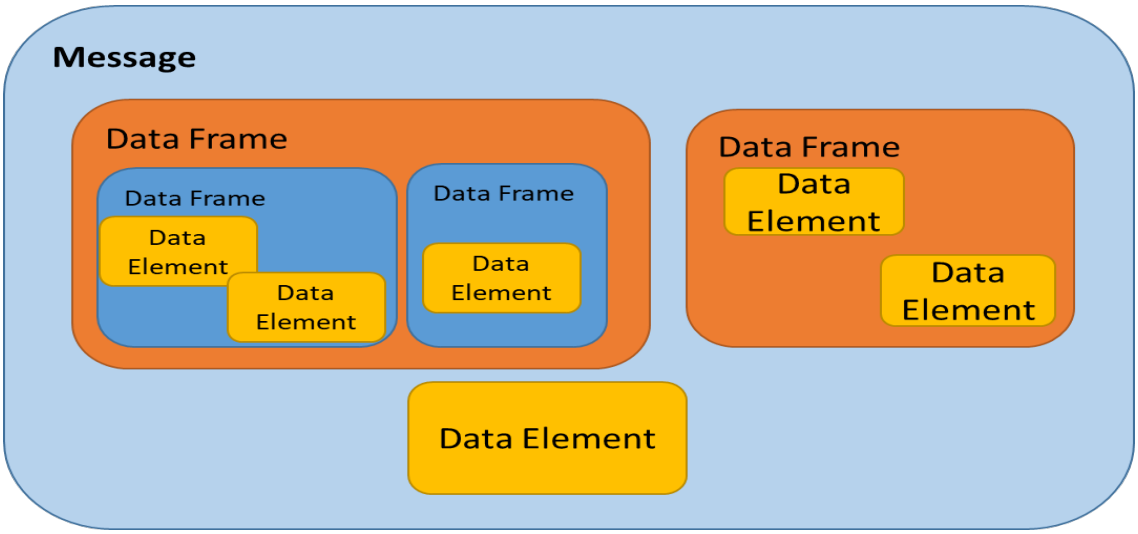

Figure 5.1: Representation of a sample DSRC Message Frame

To implement stop/go motion at signalized intersections, two messages are required to be transmitted by the road-side-unit which are received by the vehicle system with the message ID as follows

1. MSG_MapData (MAP)
2. MSG_SignalPhaseAndTiming (SPAT)

The Map Data message is used to convey one or more intersection's lane geometry maps within a single message. This message includes data pertaining to the geographical information of items such as complex intersections, road segments, high-speed curve outlines and segments of roadway. This message is also used to define the details of the indexing systems that are in turn used by other messages to relate to additional information such as the signal phase events from the SPAT message at a specific geographic location on the roadway. A complete summary of the SPAT message payload is given in Appendix A.

The Signal Phase and Timing message is used to transmit the current status of one or more signalized intersections. Along with with the MAP data, the vehicle system will be able to determine the state of the signal phasing and the time for the next phase to occur. A complete summary of the SPAT message payload is given in Appendix B.

5.2.1 Data Extraction from MAP Message

When the ego vehicle is in the range of the road-side-unit, the MAP message is used to determine the position and location of the road-side-unit. For a traffic signal, this location is actually the location of the stop line on the road. Since the MAP message may contain data for one or more intersection lanes, it is first required to determine the lane data that are applicable to the ego vehicle. This is achieved using the global position of the vehicle, which is obtained using the Global Positioning Sensor (GPS) unit.

The geometric nodes of all the lanes pertaining to an intersection are stored in the Node List XY Data Frame as described in Appendix A. The location of the Node List XY Data Frame is visually represented in Figure 5.2. Node List XY Data Frame contains the position of the center nodes of the lanes spaced at approximately 1 centimeter. The location for all

the nodes of the center point for all the lanes approaching the intersection is transmitted to the vehicle. By comparing the GPS coordinates of the vehicle with the coordinates present in the Node List XY Data Frame, the Generic Lane ID is determined for the active lane of the ego vehicle. This Lane ID is used to index additional data for the selected lane. The contents of the Generic Lane Data Frame is shown in Figure 5.3. In this figure, the solid line indicates required data frames and the dashed line indicates optional data frames.

Figure 5.2: Location of the Node List XY Data Frame in the MAP Message

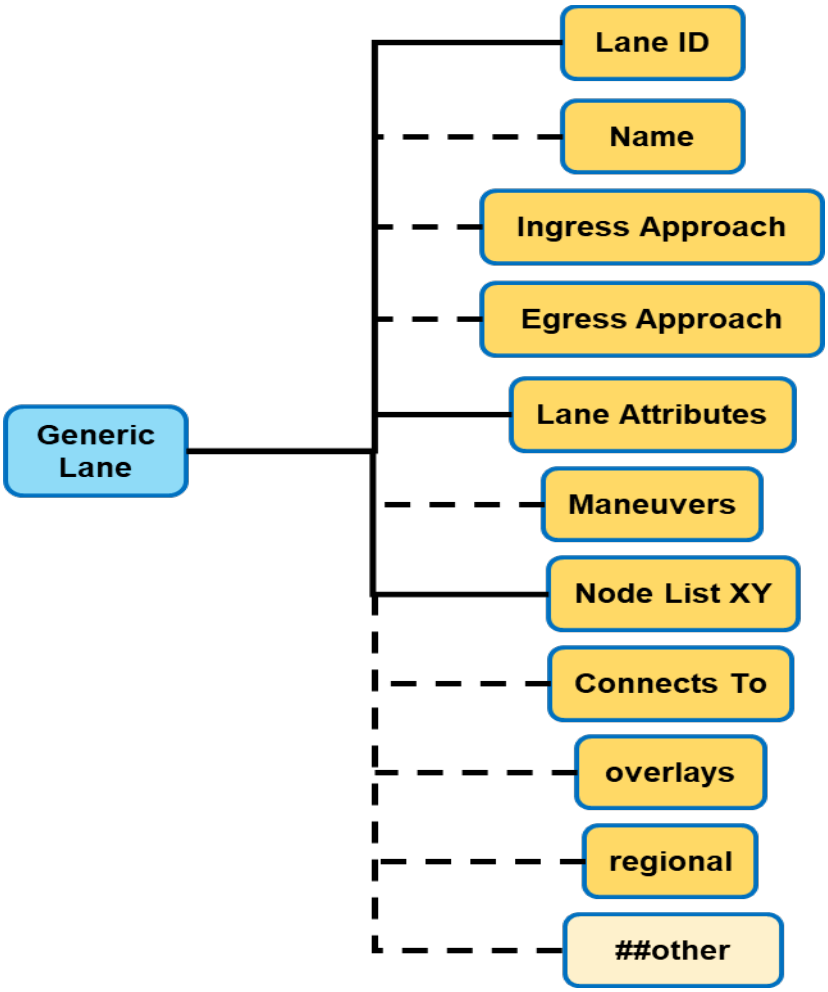

Figure 5.3: The contents of the Generic Lane Data Frame

Using the Lane ID, the corresponding data for the active lane form the MAP and SPAT messages is determined. The location of the stop line for the selected Lane ID is determined using the Intersection geometry Data Frame. The position 3D data frame is used to determine the latitude and longitude of the center of the stop line. The location of the latitude and longitude data elements are visually represented in Figure 5.4.

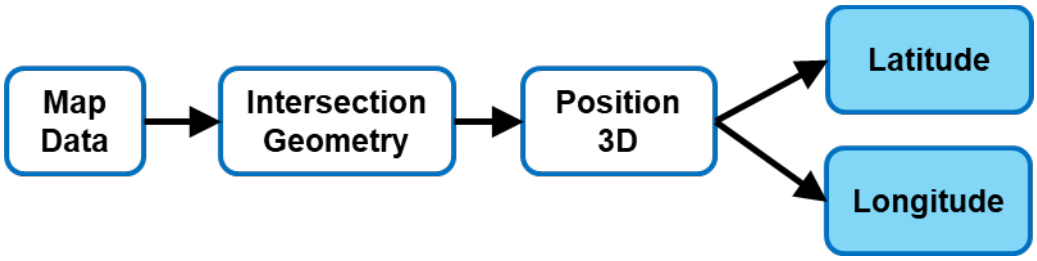

Figure 5.4: Representation of Latitude and Longitude of the stop line

The distance of the ego vehicle from the stop line represented as D_{rel} is calculated using the GPS coordinates of the ego vehicle represented by lat_{GPS} & $long_{GPS}$ and the coordinates of the center of the stop line determined form the MAP data is represented by lat_{RSU} & $long_{RSU}$. The distance to the stop line is determined using eqn. (5.1) [44].

$$D_{rel} = \sqrt{(lat_{GPS} - lat_{RSU})^2 + (long_{GPS} - long_{RSU})^2} \quad (5.1)$$

This relative distance D_{rel} is used to control the motion of the ego vehicle to implement stop/go motion, which is discussed further in detail.

Once the Lane ID for the active lane of the ego vehicle is obtained, it is further used to select the Signal Group ID data element. This Signal Group ID is used to match the lane data in the MAP message and the signal phasing data from the SPAT message. To determine the signal phasing for the active lane of the ego vehicle, the Signal Group ID is used as an index to select the appropriate signal phasing data from the SPAT message. The Signal Group ID is obtained from the MAP message using the Connects to Data Frame contained in the Generic Lane data frame as shown in Figure 5.3. The address for the Signal Group ID for a corresponding Generic Lane is visually represented as shown in Figure 5.5.

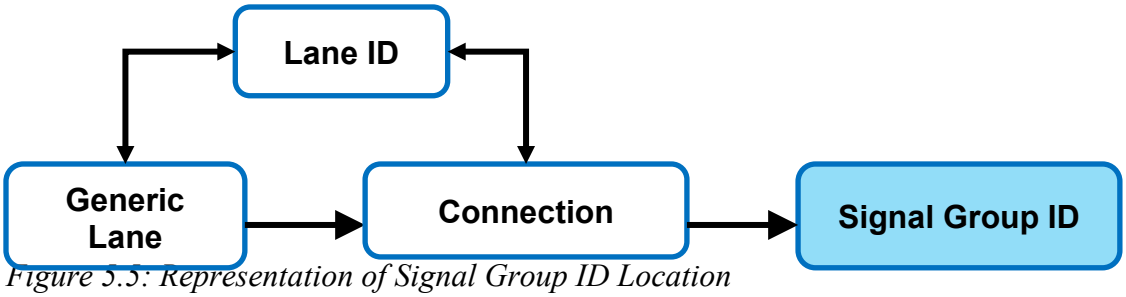

Figure 5.5: Representation of Signal Group ID Location

The extraction of the signal Phasing Data from the SPAT message is discussed in the next section

5.2.2 Data Extraction from SPAT Message

The DSRC message set transmitted by the road-side-unit contains both the MAP data and the SPAT data. SPAT data can be used to determine the current phase for each signal in the system, which is sent in the Movement Phase State Data Frame. The phase timing related information is conveyed via the Time Change Details Data Frame, which contains information such as the time an event has begun or expected to begin and the time at which the event will end latest. The SPAT message payload is shown in Appendix B.

Since the SPAT message contains information from one or more signalized intersections, the Signal Group ID obtained from the MAP message is used to index the appropriate SPAT data for the ego vehicle. The schematic of the SPAT message with the Signal Group ID and the Movement Phase State data is shown in Figure 5.6. Using the active Signal Group ID determined using the MAP data, the Movement Phase State and the Time Change Details data for the appropriate Lane is determined.

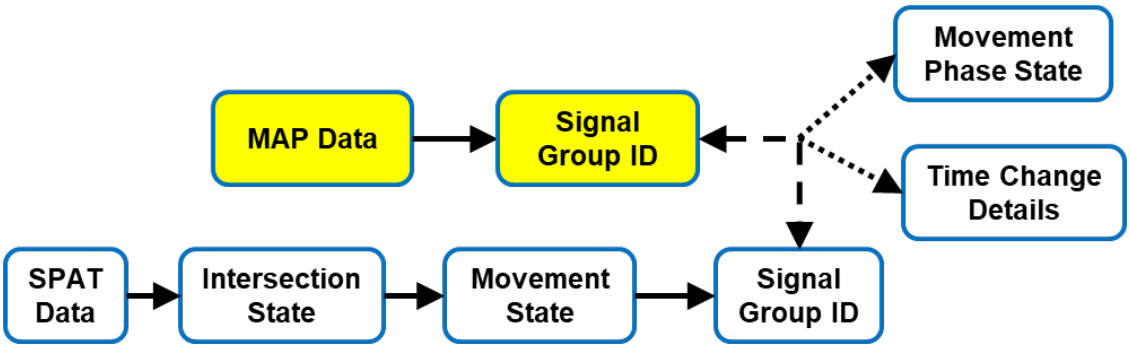

Figure 5.6: Schematic of SPAT Message Data Frame

The Movement Phase State is a Data Element that conveys the current signal phase information i.e. green/red phase. The SPAT Message is capable of transmitting other various information such as yellow or flashing red but the scope of this research is limited to stop/go motion, hence only the red and green phases are considered. The Time Change Details Data Frame consists of timing related signals such as the phase or event start time and likely end time. The contents of the Time Change Details Data frame are shown in Figure 5.7.

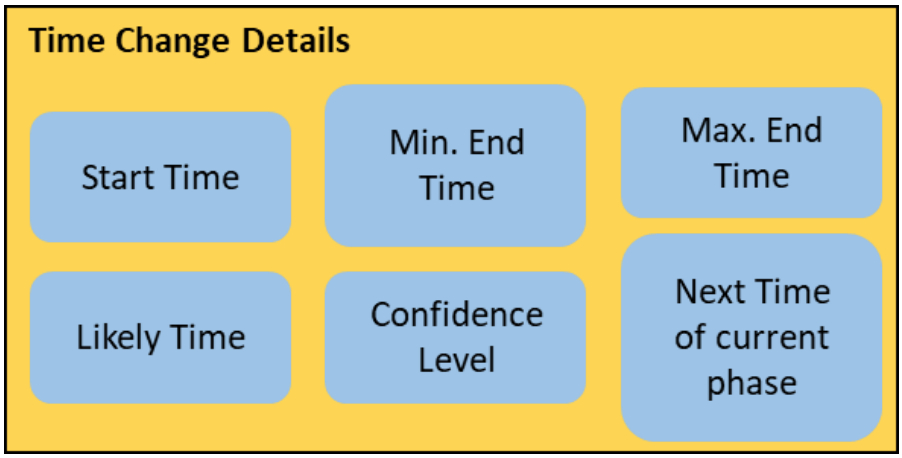

Figure 5.7: Contents of Time Change Details Data Frame

The Movement Phase State and Max. End Time data elements are used to determine the signal phase and the time to the next phase. This is one of the methods that can be used to extract the required data as per the SAE J2735 Message Set definitions.

5.3 Implementation of Stop/Go motion

The V2X signal information is used to determine if the current phase of the signal and the time for it to change. The range of the DSRC signal is usually 300m as defined by the PreScan V2X Plugin module. Though the data is available from 300m, the stopping action starts only when the signal is at a distance of 50m away from the ego vehicle. This distance was selected as it is the approximate stopping distance (D_{stop}) required to stop a vehicle when the vehicle is at a speed of 30 mph. The stopping distance (D_{stop}) is calculated using eqn. (5.2).

$$D_{stop} = T_{stop} * \dot{x}_{ego} \tag{5.2}$$

In eqn. (5.2), T_{stop} represents the time gap from the intersection at which braking is initiated. The information broadcasted from the V2I transmitter is the latitude and longitude of the device with the PreScan® global coordinates as a reference.

For a red signal phase, the relative distance to the signal (D_{rel}) is used as a reference to the MPC controller. The stopping distance (D_{stop}) tracks this relative distance thereby slowing down the vehicle as the ego vehicle changed its velocity to maintain the stopping distance from the traffic signal. Once the signal phase changes to green, the controller stops tracking the relative distance as the signal phase is green and no stopping maneuver is required. The MPC controller tries to track the command velocity by changing the tractive force demand thereby changing the ego velocity.

This process is further explained using the simulation setup and the simulation results discussed below.

5.4 Simulation Setup

A scenario is created with multiple signalized intersections in the PreScan® GUI. The signal phase and the change in signals are programmed such that the green phase and the red phase are in the 'ON' state for the same duration. After this specified time is elapsed, the signal phase changes. The orange phase in the signals are not considered and only the red and green phases are considered. The V2X transmitter is positioned on the signal as shown in Figure 5.8. The red dot in Figure 5.8 indicates the V2X transmitter. Two such traffic signals are placed at intersections, with different phase change timings are spaced at 130 m from each other as highlighted using the red boxes in Figure 5.11. The Signal Group ID of these two signals is defined in Figure 5.11.

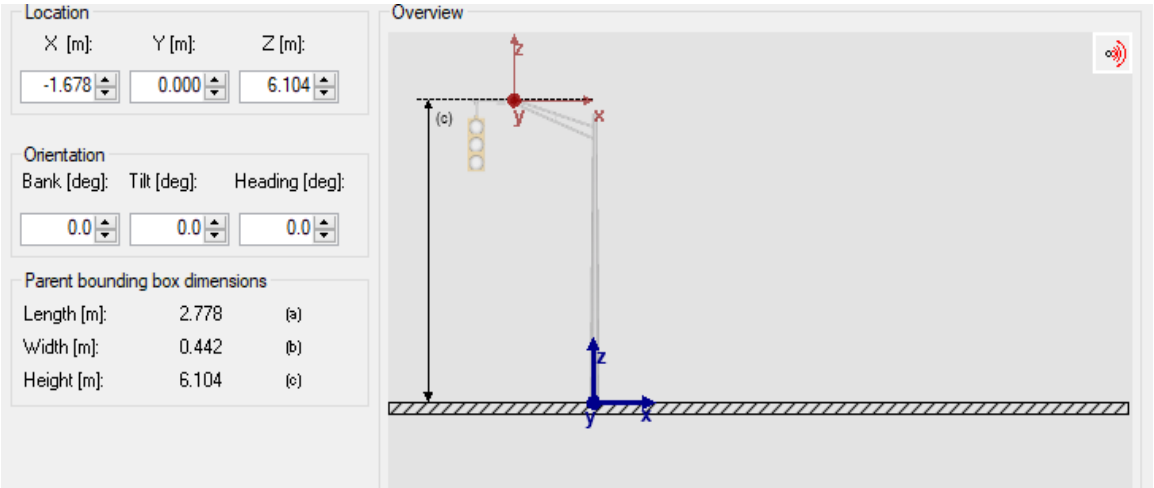

Figure 5.8: Positioning of the V2X transmitter on the traffic signal.

The simulation is set up such that the signal phase change time for both these signals is different. The phase change time for the first signal with Signal Group ID of 1 is 21 seconds and for the second signal with Signal Group ID of 2 is 10 seconds. The phase change of the signals with respect to time for signal 1 is shown in Figure 5.9 and the phase change with respect to time for signal 2 is shown in Figure 5.10.

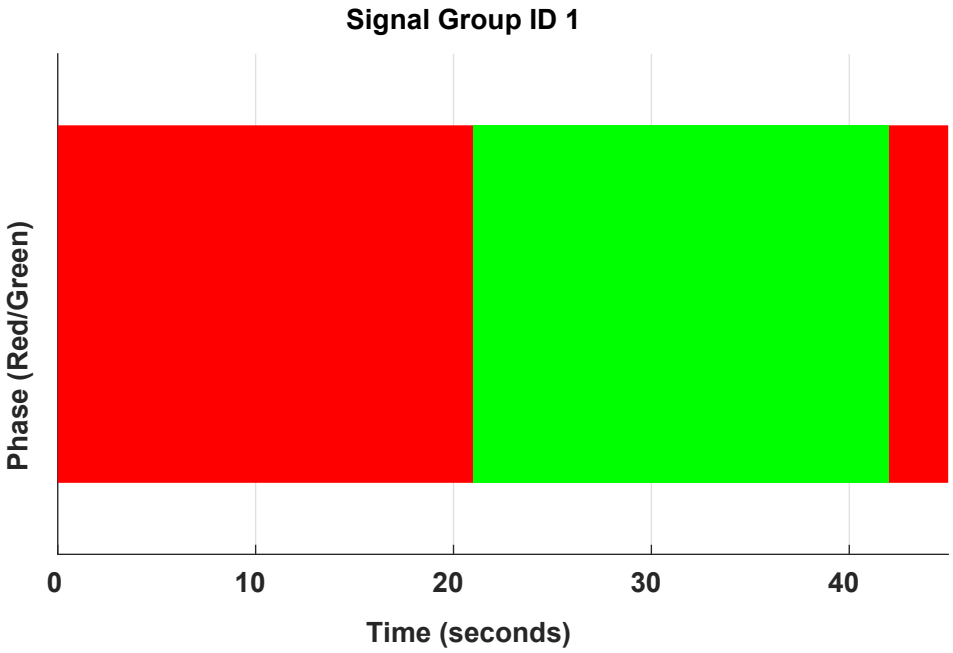

Figure 5.9: Phase change for Signal 1

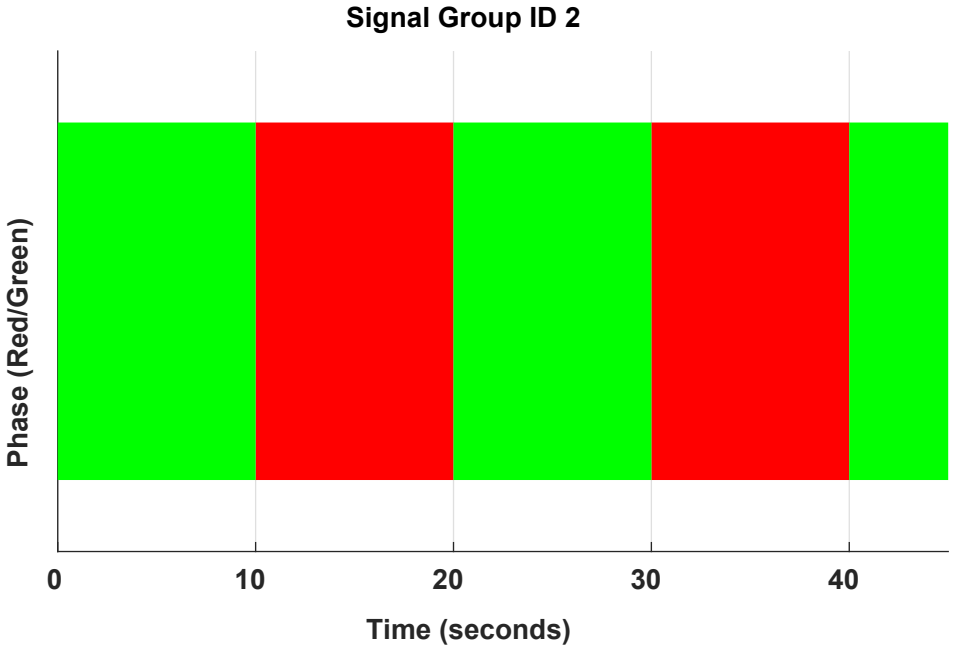

Figure 5.10: Phase Change for Signal 2

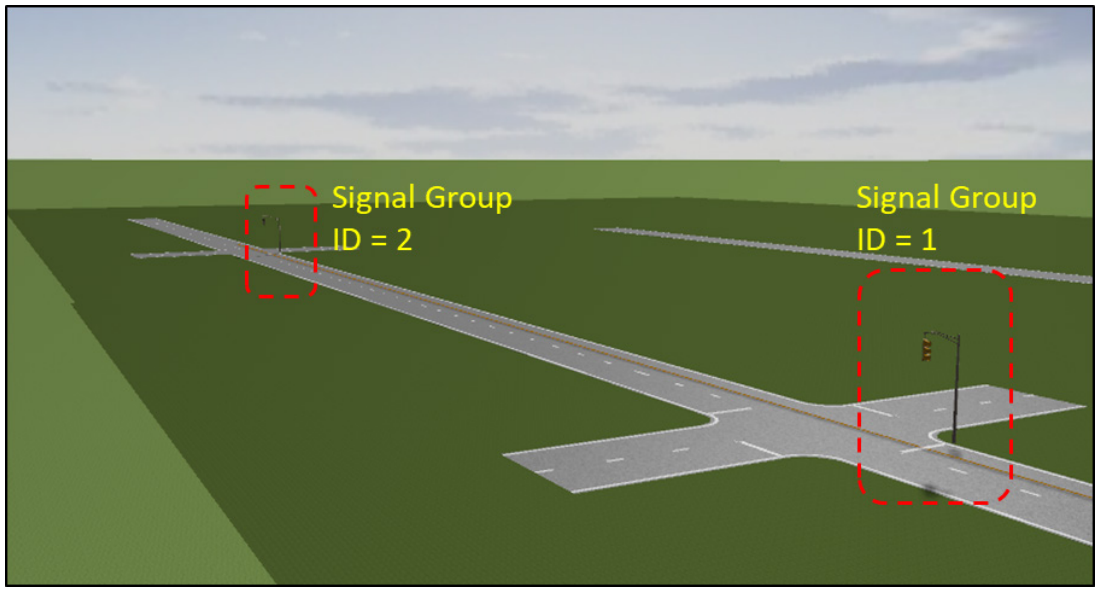

Figure 5.11: Scenario created with signalized intersections in PreScan GUI - The red boxes highlight the position of the traffic signals.

The information broadcast via the V2X transmitter is the SPAT message and the MAP message. The PreScan® V2X plugin toolbox can be used to model a signal similar to the SPAT message and the MAP message using the Generic V2X signal format as shown in Figure 5.12. The MAP and SPAT message payloads are shown in Appendix A and Appendix B, respectively, as described in the SAE – J2735 Message Set Dictionary [43].

The actual signals from the DSRC devices require preprocessing to extract the data. For simulating this preprocessed data, generic signals from the PreScan® V2X plugin toolbox are used to model the required MAP and SPAT messages. The distance is calculated using the Position signal contained in the MAP message. This is also used to determine the appropriate Signal Group ID that is applicable for the ego vehicle based on the position. The Movement Phase State data element for the corresponding Signal Group ID are used to determine the value of the current signal phase and the time when the phase changes. The messages transmitted from these devices are received by a single receiver on the ego vehicle. Generic signals are modeled to transmit only the required signals such as position, Signal Group ID & Phase state as shown in Figure 5.13. The frequency at which the data

is updated is 0.1 seconds as defined by the standard. The vehicle level controller is able to receive the data only from a predefined number of devices. If more devices are transmitting the data, it will not be able to read the data from these additional devices.

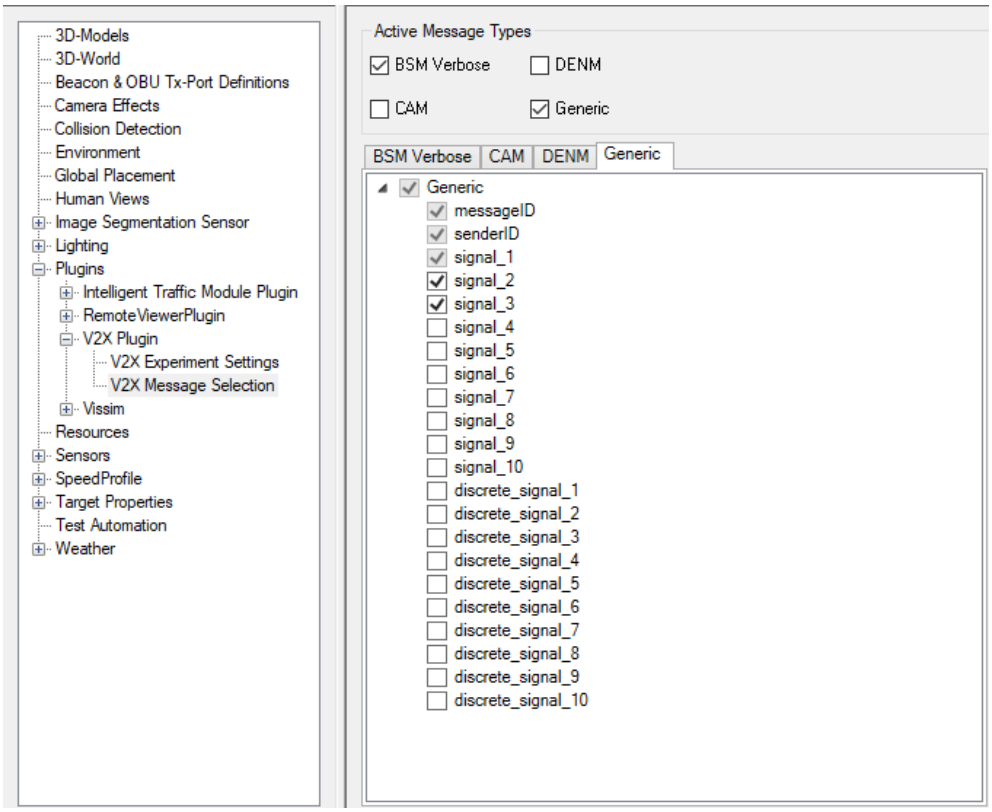

Figure 5.12: V2X plugin setting on PreScan

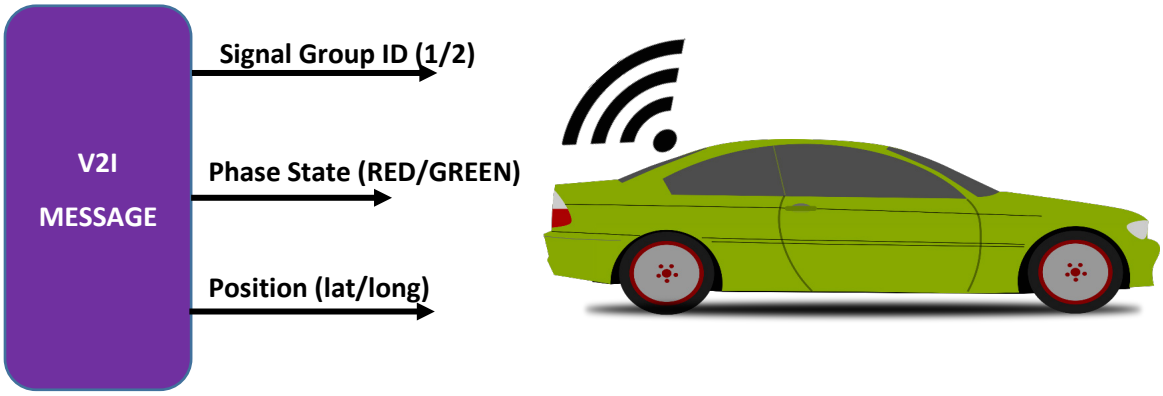

Figure 5.13: Visualization of the V2I Message Packet

5.5 Simulation Results

Using the above-mentioned traffic scenario, the stop/go motion of the ego vehicle at signalized intersections is demonstrated. The ego vehicle initially 160 m away from the first signal starts moving with an initial velocity of 5m/s as shown in Figure 5.14. The vehicle tries to attain the command velocity of 20m/s and stabilizes at this value. When the ego vehicle reaches a distance of 50m to the intersection as shown in Figure 5.15 braking is initiated and the vehicle slows down such that it stops at the intersection because of the red phase. This can be seen clearly in Figure 5.14 and the region of braking is highlighted and labeled. Once the phase changes, the controller stops tracking the relative distance (D_{rel}) as the phase is green as highlighted in Figure 5.15. This is achieved by eliminating the difference between the relative distance and the stopping distance, which makes the distance term zero in the cost function. The ego vehicle again tries to reach the command velocity and continues acceleration as shown in Figure 5.14. The first intersection is passed by now and the second intersection signals are being read the vehicle. This is because the Signal Group ID is updated from 1 to 2 as shown in Figure 5.14. Though the distance to the second signal is less than 50m, the relative distance to the signal originates only when the signal phase changes to the red phase as shown in Figure 5.15. The vehicle comes to a rest and waits till the phase changes and passes the intersection as shown in Figure 5.14 & Figure 5.15. The tractive force demanded by the controller to achieve this stop/go motion is shown in Figure 5.16.

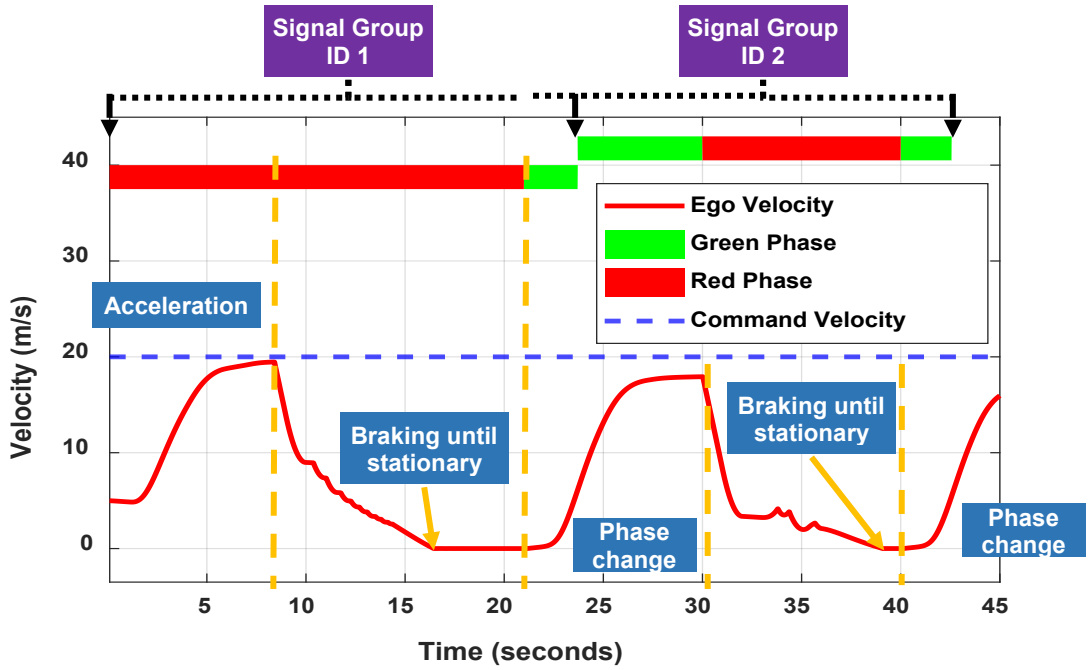

Figure 5.14: Velocity of ego vehicle through two signalized intersections

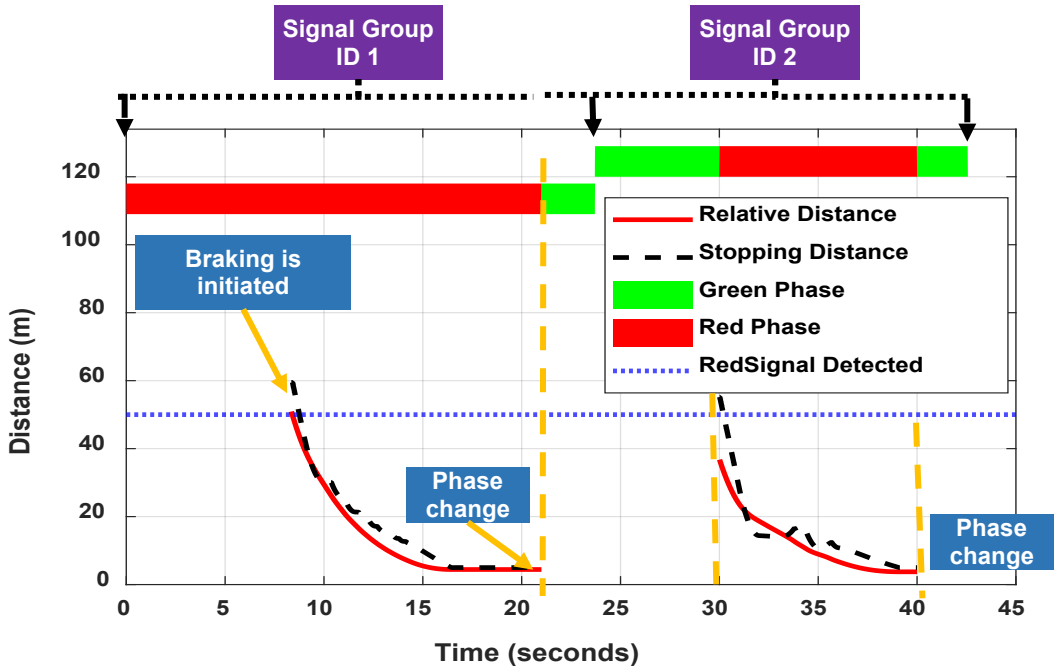

Figure 5.15: Relative distance (D_{rel}) versus stopping distance (D_{stop}) through signalized intersections.

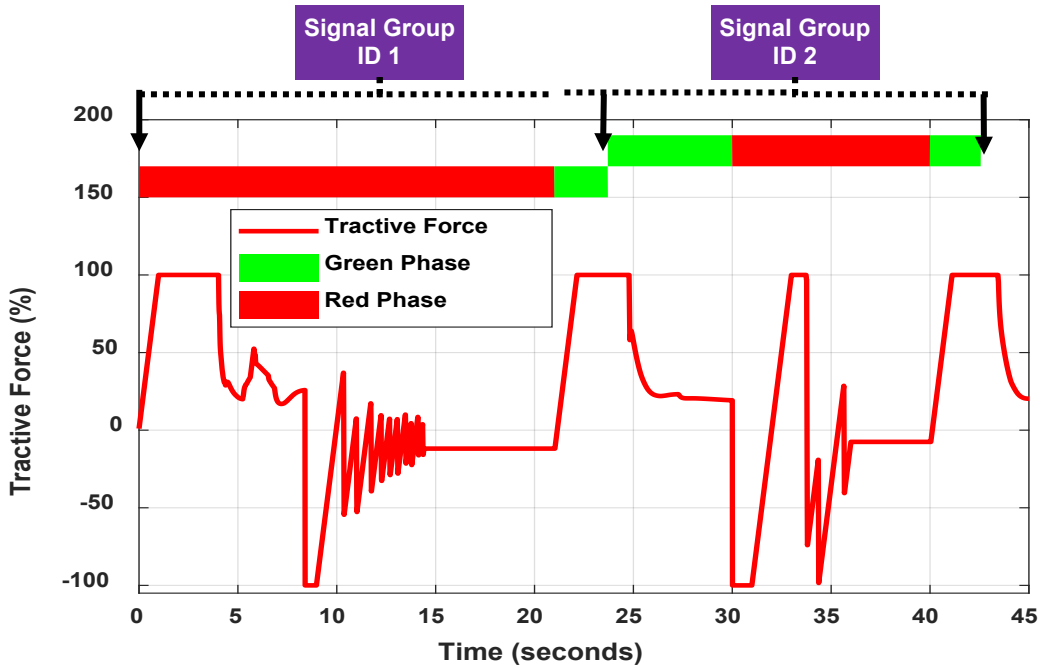

Figure 5.16: Tractive Force requested by the MPC controller for the stop/go motion

Using the SPAT data from V2X infrastructure, the navigation of the autonomous vehicle through signalized traffic intersections was made possible and stop/go motion of the autonomous vehicle is demonstrated using the PreScan Software.

By using path planning algorithms and V2X communication signals the self-driving capability of any Level 2 autonomous vehicle is increased. The conclusion for the research and the possible future work in the field are discussed in the next chapter.

6 Conclusions & Future Work

This research focusses on the implementation of computationally efficient path planning algorithms for autonomous vehicles, which can be implemented in real-time to plan an optimum path such that the autonomous vehicle can avoid obstacles during uncertainties. The control algorithm was further extended to demonstrate stop/go motion at signalized intersections with the use of data transmission using V2X communication technology.

6.1 Conclusion

In this research, we have addressed the issue of path planning and motion control of self-driving vehicles under uncertainties, such as an obstacle in the path or a traffic collision situation. With the use of the proposed A* algorithm combined with a Bezier curve trajectory generator, a model predictive controller was implemented to control the lateral motion of the vehicle. Furthermore, a constant time gap approach was used to apply longitudinal control for changing the speed of the vehicle based on the distance to a lead vehicle and commanded velocity. Using MIL methods and PreScan® simulation tool, the controller was tested for various traffic scenarios and the performance of the controller was evaluated for validation. The working of the longitudinal and lateral control algorithms was demonstrated for scenarios with static and dynamic obstacles.

Furthermore, the same control algorithm was used to implement stop/go motion in autonomous vehicles to navigate signalized intersections in urban driving scenarios. MAP and SPAT message data frames such as position and phase state were transmitted using the V2X communication devices from the traffic signal and the distance to the nearest approaching signal was determined. Using this data, stop/go motion of the autonomous vehicle was demonstrated by simulating a traffic scenario on the PreScan® GUI. Using real-time MIL testing methods, the control algorithm was tested for varying traffic phase signals and timings, and the behavior of the controller and the ego vehicle was observed.

By implementing such algorithms, the level of autonomy of self-driving vehicles can be increased during uncertainties or while navigating signalized intersections thereby mitigating and eventually eliminating human intervention in autonomous driving.

6.2 Future Work

The research in the field of autonomous vehicles is moving at a rapid pace, yet there are many new areas to be explored. Conducting similar MIL/SIL testing for more varied situations and understanding the effects of temperature and lighting can also be studied. PreScan® allows users to simulate the controller for various road grades, weather conditions, ambient lighting, and other disturbances can be introduced to validate the robustness of the controller.

The proposed path planning algorithm can be further developed and improved by testing the control algorithm using HIL methods and real-time vehicle testing. Vehicle data can be used to improve the prediction model and thereby improving the MPC controller behavior for nonlinearities. Non-Linear MPC can be implemented by improving the prediction model to include the steering and tire system nonlinearities, which can increase the quality of the controller. Further, sensor data quality can be improved by using more complex image processing and data localization algorithms to combine data from various sensors making it more accurate.

The information from the V2X communication devices can be used to further develop the control algorithm to communicate with other cars and road-side-units, which can be used to tackle various driving situations such as navigating through a non-signalized intersection or lane change assistance for highway driving. Further, the SPAT message data can be used to develop algorithms for efficient approach and departure at intersections, which can lead to energy savings and reduced trip times. All such advanced algorithms can be tested using the PreScan® simulation platform and subsequently tested on real-time HIL systems.

7 Reference List

[1] United States Department of Transportation. (2017). *TRAFFIC SAFETY FACTS*. Available: https://crashstats.nhtsa.dot.gov/Api/Public/Publication/812456

[2] U. Z. A. Hamid, K. Pushkin, H. Zamzuri, D. Gueraiche, and M. A. A. J. P. e. Rahman, "Current Collision Mitigation Technologies for Advanced Driver Assistance Systems â€“A Survey," vol. 6, no. 2, 2016.

[3] K. B. B. (n.d.). (January 14). *Largest safety concerns while driving a vehicle, according to U.S. respondents in January 2016.* Available: https://www.statista.com/statistics/669018/us-respondents-largest-safety-concerns-while-driving-a-vehicle/.

[4] Waymo. (Dec 5, 2018, Dec 12,2018). *Waymo One: The next step on our self-driving journey*. Available: https://medium.com/waymo/waymo-one-the-next-step-on-our-self-driving-journey-6d0c075b0e9b

[5] Tesla. (Dec 12). *Autopilot*. Available: https://www.tesla.com/presskit#autopilot

[6] *Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems*, 2014.

[7] M. Buehler, K. Iagnemma, and S. Singh, *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*. Springer Berlin Heidelberg, 2009.

[8] DARPA. (2007). *DARPA Urban Challenge*. Available: https://www.darpa.mil/about-us/timeline/darpa-urban-challenge

[9] P. F. McLauchlan and J. Malik, "Vision for longitudinal vehicle control," in *Proceedings of Conference on Intelligent Transportation Systems*, 1997, pp. 918-923.

[10] L. R. Ray, "Nonlinear state and tire force estimation for advanced vehicle control," *IEEE Transactions on Control Systems Technology,* vol. 3, no. 1, pp. 117-124, 1995.

[11] S. Li, K. Li, R. Rajamani, and J. J. I. T. o. C. S. T. Wang, "Model predictive multi-objective vehicular adaptive cruise control," vol. 19, no. 3, pp. 556-566, 2011.

[12] Jes *et al.*, "Pure-pursuit reactive path tracking for nonholonomic mobile robots with a 2D laser scanner %J EURASIP J. Adv. Signal Process," vol. 2009, pp. 1-10, 2009.

[13] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi, "A stable tracking control method for an autonomous mobile robot," 1990.

[14] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. J. I. T. o. i. v. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," vol. 1, no. 1, pp. 33-55, 2016.

[15] C. Urmson *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," vol. 25, no. 8, pp. 425-466, 2008.

[16] M. Montemerlo *et al.*, "Junior: The stanford entry in the urban challenge," vol. 25, no. 9, pp. 569-597, 2008.

[17] A. Bacha *et al.*, "Odin: Team victortango's entry in the darpa urban challenge," vol. 25, no. 8, pp. 467-492, 2008.

[18] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[19] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. J. I. T. o. C. S. T. How, "Real-time motion planning with applications to autonomous urban driving," vol. 17, no. 5, pp. 1105-1118, 2009.

[20] J. Leonard *et al.*, "A perception-driven autonomous urban vehicle," vol. 25, no. 10, pp. 727-774, 2008.

[21] D. Connell and H. M. La, "Dynamic path planning and replanning for mobile robots using RRT," in *Systems, Man, and Cybernetics (SMC), 2017 IEEE International Conference on*, 2017, pp. 1429-1434: IEEE.

[22] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, 2015, pp. 3067-3074: IEEE.

[23] N. Lu, N. Cheng, N. Zhang, X. Shen, and J. W. Mark, "Connected Vehicles: Solutions and Challenges," *IEEE Internet of Things Journal,* vol. 1, no. 4, pp. 289-299, 2014.

[24] B. Barik, P. Krishna Bhat, J. Oncken, B. Chen, J. Orlando, and D. Robinette, "Optimal velocity prediction for fuel economy improvement of connected vehicles," *IET Intelligent Transport Systems,* vol. 12, no. 10, pp. 1329-1335, 2018.

[25] B. Asadi and A. Vahidi, "Predictive Cruise Control: Utilizing Upcoming Traffic Signal Information for Improving Fuel Economy and Reducing Trip Time," *IEEE Transactions on Control Systems Technology,* vol. 19, no. 3, pp. 707-714, 2011.

[26] S. Ibrahim, D. Kalathil, R. O. Sanchez, and P. J. a. p. a. Varaiya, "Estimating Phase Duration for SPaT Messages," 2017.

[27] J. E. Doran, D. Michie, and D. G. Kendall, "Experiments with the Graph Traverser program," vol. 294, no. 1437, pp. 235-259, 1966.

[28] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics,* vol. 4, no. 2, pp. 100-107, 1968.

[29] M. E. Mortenson, *Mathematics for computer graphics applications*. Industrial Press Inc., 1999.

[30] G. Farin, *Curves and surfaces for computer-aided geometric design: a practical guide*. Elsevier, 2014.

[31] J.-W. Choi, R. Curry, and G. Elkaim, "Piecewise bezier curves path planning with continuous curvature constraint for autonomous driving," in *Machine learning and systems engineering*: Springer, 2010, pp. 31-45.

[32] J. M. Maciejowski, *Predictive Control: With Constraints*. Prentice Hall, 2002.

[33] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. J. A. Scokaert, "Constrained model predictive control: Stability and optimality," vol. 36, no. 6, pp. 789-814, 2000.

[34] M. Cheng, L. Feng, and B. Chen, "Nonlinear Model Predictive Control of a Power-Split Hybrid Electric Vehicle with Electrochemical Battery Model," 2017. Available: https://doi.org/10.4271/2017-01-1252

[35] P. Poramapojana and B. Chen, "Minimizing HEV fuel consumption using model predictive control," in *Proceedings of 2012 IEEE/ASME 8th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications*, 2012, pp. 148-153: IEEE.

[36] P. Poramapojana and B. Chen, "Model Predictive Control for Hybrid Electric Vehicle Energy Management," in *The 2012 International Conference on Advanced Vehicles and Integration Technology*, Changchun, China, July 16-19, 2012.

[37] L. Feng, M. Cheng, and B. Chen, "Predictive control of a power-split hev with fuel consumption and soc estimation," SAE Technical Paper0148-7191, 2015.

[38] T. Takahama and D. J. I. J. o. A. E. Akasaka, "Model Predictive Control Approach to Design Practical Adaptive Cruise Control for Traffic Jam," vol. 9, no. 3, pp. 99-104, 2018.

[39] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[40] A. Bemporad, M. Morari, and N. L. Ricker, "Model Predictive Control Toolbox™ User's Guide," 2018, Available: https://www.mathworks.com/help/pdf_doc/mpc/mpc_ug.pdf.

[41] G. F. Franklin, J. D. Powell, and M. L. Workman, *Digital control of dynamic systems*. Addison-wesley Menlo Park, CA, 1998.

[42] A. Bemporad, "Model predictive control design: New trends and tools," in *Decision and Control, 2006 45th IEEE Conference on*, 2006, pp. 6678-6683: IEEE.

[43] *(R) Dedicated Short Range Communications (DSRC) Message Set Dictionary*, 2009.

[44] J. Zheng, H. X. Liu, and J. Parikh, "Automatic Intersection Map Generation Task 10 Report," 2016-2-29 2016.

Appendix A: Structure of the MAP Message

The payload structure of the MAP message payload is shown in Figure 7.1. The data frames of importance for the proposed control algorithm are outlined in the figure with dashed lines. A more detailed definition of all the data frames can be found in the SAE – J2735 Message Set Definition Standard.

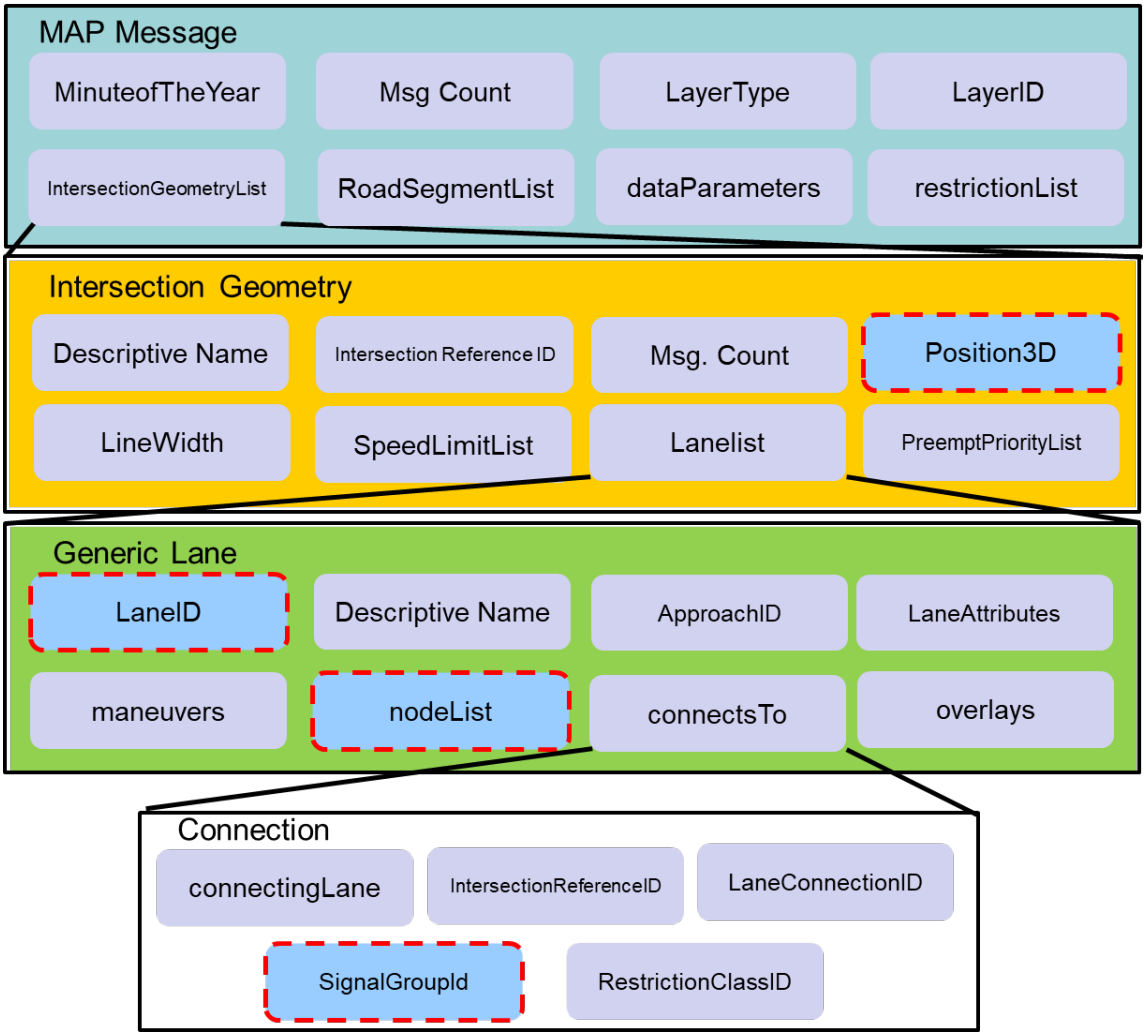

Figure 7.1: Payload of MAP Message

Appendix B: Structure of the SPAT Message

The payload structure of the SPAT message payload is shown in Figure 7.2. The data frames of importance for the proposed control algorithm are outlined in the figure with dashed lines. A more detailed definition of all the data frames can be found in the SAE – J2735 Message Set Definition Standard.

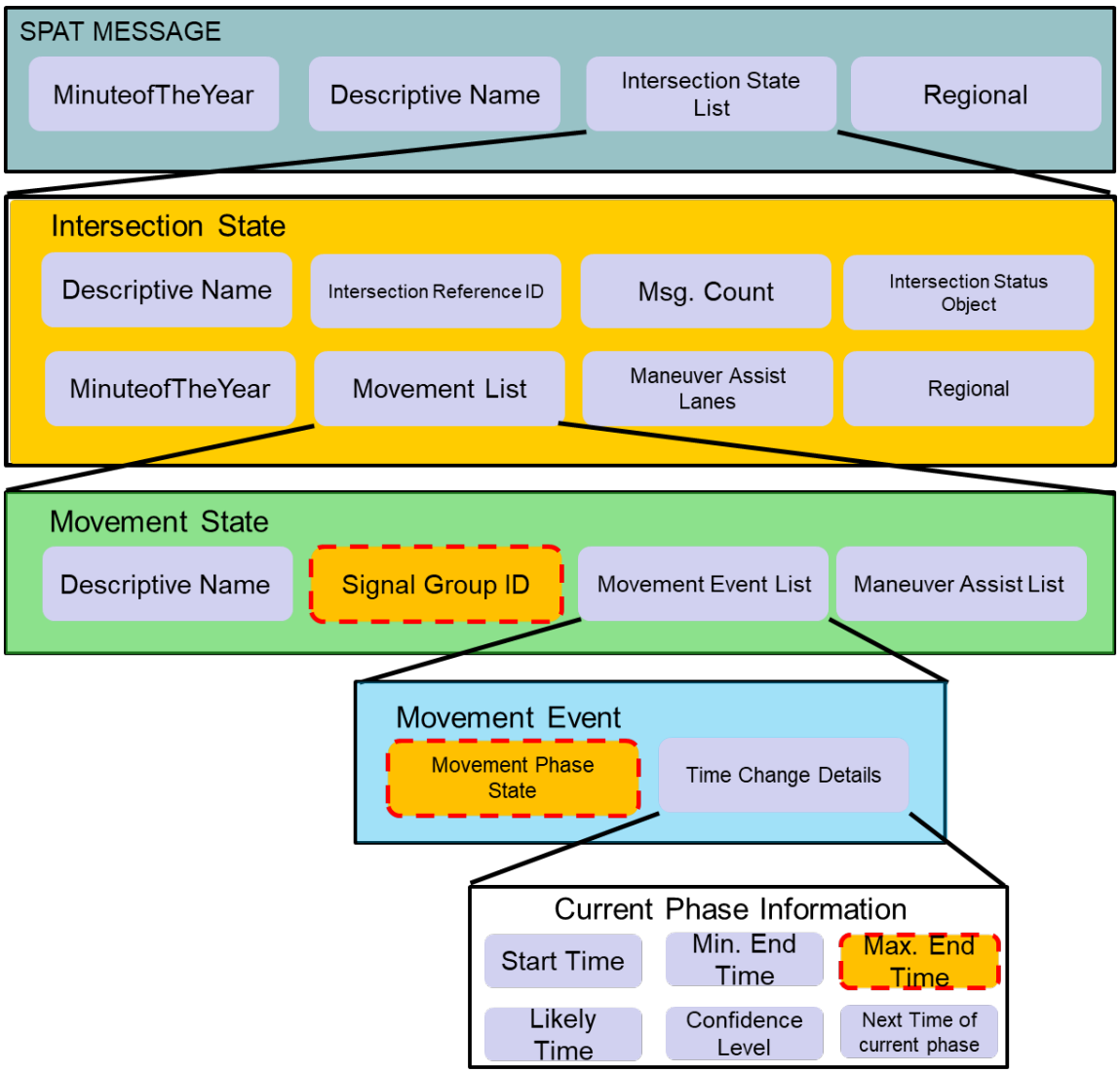

Figure 7.2: Payload of SPAT Message

Appendix C: Copyright documentation

Some of the chapters of this document have been accepted for publication to the International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, ASME. The copyright agreement has been signed and the following permission via email communication was established with the manager of the conference publications at ASME.

Regarding republication of material as Master's Report/Thesis

Nhora Cortes-Comerer <Cortes-ComererN@asme.org> Wed, Feb 20, 2019 at 1:18 PM
To: Sai Rajeev Devaragudi <sdevarag@mtu.edu>
Cc: Walter Ancarrow <AncarrowW@asme.org>

If you plan to use any of the content now, and the paper has not been accepted, you would need to annotate any references to the paper as "Under submission to the xyz conference."

Assuming the paper is accepted and presented at the conference, you would need to update the reference and copyright.

Regards,

Nhora Cortes-Comerer

Nhora Cortes-Comerer
Manager
Conference Publications
ASME
2 Park Avenue, 6th Floor
New York, NY 10016-5990
Tel 1.212.591.7099
cortes-comerern@asme.org