DITM: Control-oriented sensor fusion for bus platooning

EngD THESIS REPORT

DITM: Control-oriented sensor fusion for bus platooning

An integrated approach towards developing control and fusion algorithms

ir. Prabhat K Sharma
Oct 2023
Department of Mathematics & Computer Science

EngD AUTOMOTIVE SYSTEMS DESIGN
Track AUTOMOTIVE SYSTEMS DESIGN

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

DITM: Control-oriented sensor fusion for bus platooning

An integrated approach towards developing control and fusion algorithms

Prabhat Kumar Sharma

October 2023

Eindhoven University of Technology
Stan Ackermans Institute - Automotive/Mechatronic Systems Design

EngD Report: 2023/076

Confidentiality Status:
Open access

Partners

Siemens Industry Software B.V. Eindhoven University of Technology

Steering Group

Project Owner	: ir. Prabhat K Sharma (TU/e)
Project Manager	: ir. Riske Meijer (TU/e)
Project Mentors	: dr. ir. Igo Besselink (TU/e) | dr. ir. Jeroen Ploeg (Siemens)
Project Supervisors	: dr. ir. Erjen Lefeber (TU/e) | Xavier Borrega (Siemens)

Date October 2023

Composition of the Thesis Evaluation Committee:

Chair:	dr. M. (Mircea) Lazar
Members:	dr. ir. I.J.M. (Igo) Besselink
	dr. ir. A.A.J. (Erjen) Lefeber
	dr. ir. J. (Jeroen) Ploeg
	X. (Xavier) Borrega, MSc
	dr. ir. T.P.J. (Tom) v.d. Sande
	ir. R. (Riske) Meijer

The design that is described in this report has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct.

Date October, 2023

Contact address
Eindhoven University of Technology
Department of Mathematics and Computer Science
Automotive Systems Design
MF 5.075
P.O. Box 513
NL-5600 MB
Eindhoven, The Netherlands
+31 402743908

Published by Eindhoven University of Technology

PDEng Report 2023/076

Abstract

In this project, the lateral and longitudinal control of a follower bus in a three-bus platoon is developed. An integrated approach for a high-level track-to-track sensor fusion is employed where the requirements are derived from the control problem formulation. The developed algorithms are analyzed and verified experimentally in the Siemens Prescan simulation environment. The simulation results show that the presented algorithms are capable of not only achieving the control objectives to follow the path of the preceding/lead vehicle closely but also doing it in a comfortable manner for the passengers.

Keywords Platooning, lateral control, longitudinal control, track-to-track-fusion

Preferred reference DITM: Control-oriented sensor fusion for bus platooning. Eindhoven University of Technology, PDEng Report 2023/076, October 2023.

Partnership This project was supported by Eindhoven University of Technology and **Siemens Industry Software B.V.**

Foreword

In the innovation project 'Digital infrastructure for future-proof mobility' (DITM), research institutions and industry together develop and deploy a system architecture for a digital infrastructure consisting of critical core technologies related to localization, traffic services, digital maps, and energy supply. The underlying aim is to improve traffic safety and efficiency by supporting the implementation of a scalable Cooperative, Connected and Automated Mobility system (CCAM), including a reliable and secure energy supply. To this end, DITM adopts an integrated public-private approach, in which developments from the automotive industry, ICT, traffic management, and mobility innovation are combined.

To validate the feasibility and scalability of the architecture, several use cases are defined in DITM, involving the application of connected automated vehicles in a particular operational design domain. These use cases are developed, tested, and validated in a real and virtual validation environment. One of these use cases relates to platooning of electric buses, where both the longitudinal and the lateral motion are fully automated, employing on-board sensors and wireless inter-vehicle communications.

Prabhat was given the challenging assignment to design the perception and control system for this use case, the results of which are presented in the current report. And he tackled it energetically! Prabhat implemented a platooning controller for the longitudinal motion control and a non-linear path-following controller for the lateral motion control. In addition, path generation based on the preceding vehicle's path was designed and implemented in simulation. But planning and control was not all: perception of the environment is the enabling technology for all automated vehicles, including the current use case. Therefore, Prabhat also investigated a high-level fusion method, known as track-to-track fusion, which can combine the observations of various sensors such as camera and radar into object tracks. The resulting system behavior is analyzed in simulation, using the Prescan simulation platform.

Without any doubt, I can say that Prabhat has laid a solid foundation for the CCAM use case of DITM, providing a basis for several partners to build upon. Personally, I wished this EngD project lasted another year, not in the least because Prabhat proved to be a very friendly, cooperative, and ambitious student, due to which our progress meetings gave energy rather than consumed it.

Jeroen Ploeg,

Siemens Industry Software Netherlands

Helmond, October 2023

Acknowledgements

After investing almost two years in the EngD program at Eindhoven University of Technology, including my year-long final project in collaboration with Siemens, I would like to take this opportunity to thank and acknowledge the contribution of some of the most amazing people with whom I shared my journey. I also apologize to some people whose names I could not mention here. Their contributions are not forgotten and I will be always grateful to them.

First, I would like to thank all my final project mentors, namely, Igo, Erjen, Jeroen, and Xavier for their support and guidance throughout the project. Thank you Igo for providing me with the opportunity to work on this project and for your efforts to include VDL and other parties in this consortium project. Special thanks to Erjen and Jeroen for having so many in-depth conversations. These conversations helped me immensely in terms of expanding my horizons and finding innovative solutions to our problems. Many thanks to Xavier, Alperen, Pieter, and Koen for your support regarding the Siemens Prescan setup. I would also like to thank the AT lab colleagues, Erwin, Redmer, Bart, Wouter, Aaron, and Dhruv for their company and discussions on so many varied topics. Special mention to Bart and Redmer for fruitful discussions regarding V2V and radar data exploration.

Secondly, I thank the whole EngD family for making this journey fun, exploratory, and enriching. Thank you Riske and Ellen for taking care of the organizational aspects and facilitating our learning. Thank you to all the professional development coaches, especially Peter and Andre for their help. Thanks to my ASD/MSD EngD colleagues for the time we shared during lunch breaks, sports events, road/bike trips, group projects, birthday celebrations, and BBQs. I will cherish these memories forever. Special mention to Raj for being my lab partner throughout my final project. You certainly kept the mood light with your antics, jokes, and funny stories. Diya, I cannot thank you enough for being with me and helping me in so many ways. From enduring my blabber about the code not working to celebrating those little triumphs, you were always by my side encouraging me.

Finally, I want to thank my family, maa, papaji, and bhaiya for their unconditional love, support, and blessings, without which nothing would have been possible!

Prabhat Kumar Sharma

October, 2023

Executive Summary

The growing population is putting a lot of pressure on the existing mobility infrastructure. Frequent disruptions in public transport are becoming common nowadays. Cooperative, Connected, and Automated Mobility (CCAM) systems are becoming the need of the hour to improve traffic efficiency by increasing road throughput and reducing traveling time. In addition, these mobility solutions are supposed to improve road safety and reduce fuel consumption. To that end, a use case of city bus platooning is used as the central idea in this work.

This work aims to develop lateral and longitudinal control for the follower buses in a platoon such that they can follow the preceding/lead vehicle closely in a safe and comfortable manner. To implement these control algorithms, the motion states of the preceding vehicle are required. Various sensors are used to measure these states. However, no single sensor can measure these states directly with sufficient accuracy to be usable by the control algorithms. Thus, filtering and fusion algorithms are used to combine the information coming from different sensors to estimate the relevant motion states accurately. Another goal of this work is to develop these filtering and fusion algorithms based on the requirements from the controller design, thereby, approaching the control and sensor fusion problems in an integrated way instead of treating them in isolation.

The Siemens Prescan simulation environment is used as a platform to test and verify the developed algorithms in different scenarios. The simulation results show that the presented algorithms are capable of not only achieving the control objectives of the follower buses to follow the preceding vehicle closely but also doing it in a comfortable manner for the passengers.

Contents

1 Introduction

In this work, the design and development of the lateral and longitudinal control of a platoon of city buses and the underlying control-oriented sensor fusion algorithms are presented as a part of the final individual project for the engineering doctorate (EngD) program. The project is carried out in collaboration with Eindhoven University of Technology, and Siemens Industry Software B.V. In this chapter, the motivation for the project is first explained. Then, the main goals and the scope of the project are presented. Next, a system engineering based design approach applied for the execution of this project is discussed. Finally, the outline of the rest of the report is presented in the last section.

1.1 Background and motivation

With the increasing population and its ever-increasing demands, the current mobility infrastructure is reaching its limits. This limit is putting the brakes on the economic growth and prosperity of society in general [1]. Cooperative, Connected, and Automated Mobility (CCAM) systems are becoming the need of the hour to improve traffic efficiency by increasing road throughput and reducing traveling time. In addition, these mobility solutions are supposed to improve road safety and reduce fuel consumption [2]. To tackle these challenges and make the mobility infrastructure robust to future needs, an innovative project called Digital Infrastructure for Future Mobility (DITM) is being carried out in the Netherlands with the help of many industrial and academic partners [1]. Figure 1.1 shows different work packages, corresponding use cases, and the related partners involved in the project.

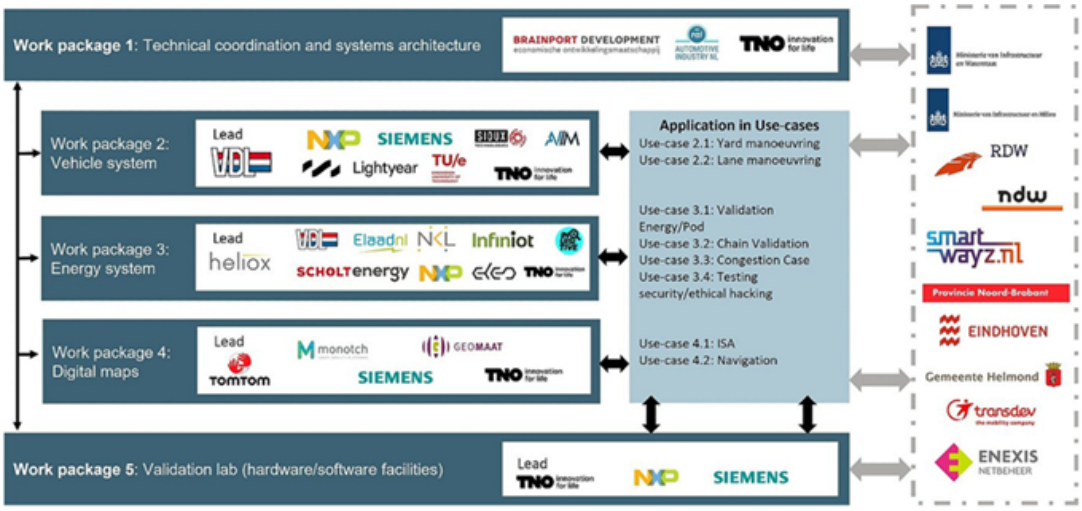


Figure 1.1: DITM work packages and related partners

This work is related to the lane maneuvering use case of work package 2, focusing on the bus platooning application. *Platooning* refers to the use of vehicle following with short-inter vehicle distance [3] [4]. This is usually achieved with the help of the onboard sensors on the vehicles and vehicle-to-vehicle (V2V) communication. Platooning is actively researched for heavy-duty commercial vehicle applications and they show significant advantages in terms of fuel savings [5]. However, city bus platooning has additional challenges like passenger comfort, frequent stop-go situations, tighter roads, etc. and there is limited literature available concerning these aspects. Apart from the above-mentioned benefits of platooning, there could be other business use cases in utilizing platooning in city buses. For instance, it can be used to manage peak traffic demands during work commute hours. It can also be used to create a rail-on-road transport mode for intra-city travel during railway maintenance and/or failure. Thus, bus platooning is included in the DITM scope.

1.2 Need for control-oriented sensor fusion

To perform cooperative and/or automated driving tasks, CCAM systems use various sensors and may rely on vehicle-to-everything (V2X) communication. For ease of articulation, any source of information, for instance, V2X communication, map data, or any other sensor data is simply referred to as sensor data in this work. The information received from these sensors is then used to understand the driving environment, localize the vehicle in the environment, plan motion, and finally control the vehicle to perform the desired driving task. Generally, one sensor is not capable of measuring all the motion states of a vehicle and even if they do, the measurements are not accurate for all the states. Thus, one sensor cannot provide accurate measurements for all the motion states necessary for control application. To that end, the sensor data from different sources are fused to make more accurate estimates. This process is called sensor fusion. There are many ways to fuse different information. The fused information should however be useful for further use, for instance, controlling the vehicle. Thus, it is important to think about sensor fusion in an integrated manner with controller design, rather than designing fusion algorithms and control algorithms in isolation. Figure 1.2 shows the workflow of an integrated approach towards control-oriented sensor fusion.

The development of control and sensor fusion algorithms in an integrated manner can be accelerated if all the related scenarios, sensors, and algorithms can be implemented virtually in a single platform. To this end, the Siemens Prescan [6] environment is used. This also allows Siemens to explore

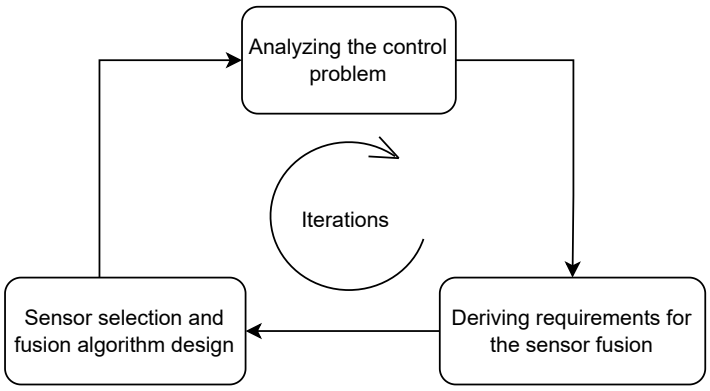


Figure 1.2: Integrated approach towards control-oriented sensor fusion

the opportunity to develop sensor algorithms within their software module based on the controller requirements.

1.3 Project goals and scope

As outlined in sections 1.1 and 1.2, the aim of this project is two-fold. The first aim is to develop vehicle control modules to implement platooning in city buses. The second aim is to derive requirements from the controllers to develop the sensor fusion algorithm in order to implement the controllers successfully. Thus, the two main goals of this design project are as follows:

1. To design and develop longitudinal and lateral control for the follower vehicles in a platoon.
2. To use a control-oriented approach to design and develop sensor fusion algorithms.

The duration of the project does not allow for developing a complete pipeline to control a platoon of buses in all possible scenarios. To that end, after discussion with the relevant stakeholders, the scope of this project is limited as summarized below:

- A three-bus platoon use case to be considered, where the lead bus is driven manually and is responsible for taking a safe path. Figure 1.3 schematically shows this use case.
- Formation and/or dissolution of the platoon is beyond the scope of this work. It is assumed that the platoon is already formed and communication is established between the vehicles.
- A homogeneous string of rigid buses, without articulation, is to be considered.
- Both structured and unstructured road infrastructure should be considered, i.e., lane markings on the roads may or may not be present or detected.
- It is assumed that the object-level measurement data is available from the sensors. Thus, object detection and tracking problem from raw sensor data is not considered in this work.
- Considering that the individual sensors can detect the same object and thus can provide track data of the same object, a high-level track-to-track fusion (T2Tf) has to be developed.
- All the information regarding ego vehicle pose (position, velocity, acceleration, heading, etc.) is known. Thus, the ego state estimation problem is not considered in this work.
- Functional safety aspects such as fail-safe operation, the safety of the intended function, etc. are also not considered in this work. Consequently, it is assumed that the wireless communication has very limited or no packet loss.

Figure 1.3: Schematic to show the use case of three bus platooning

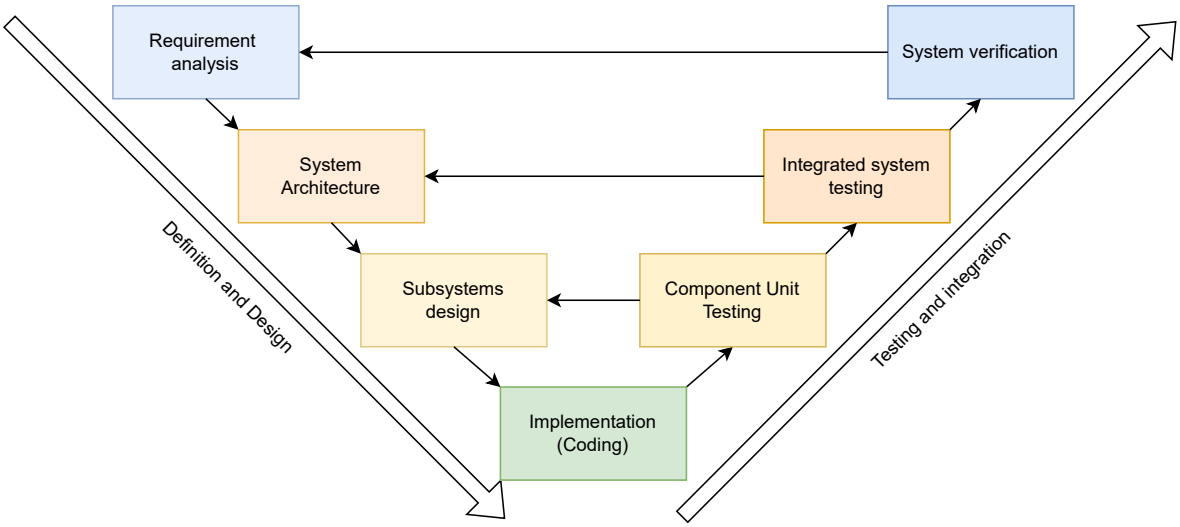


Figure 1.4: V-cycle development model

1.4 Design approach

In this section, the system engineering based design framework used for the execution of this project is discussed. First, a brief introduction of the V-model is provided which is used as the general framework for the development of various algorithms in this project. Following the V-model framework, the conceptual system design is presented which acts as the basis of implementation carried out during this project. Please refer to Figure A.1 in appendix A for the project management plan outlining the execution timeline for this project.

1.4.1 V-model

The V-model [7] is one of the most common models utilized in software development. Figure 1.4 shows different stages of the development in the V-model. The left side of the V represents the definition and design phase of the project where the conceptual design of the system is conceived based on the requirements and constraints. The system is also decomposed in this phase into subsystems for ease of implementation and to maintain important aspects of development such as modularity, traceability, etc. Once the subsystems are defined, they are implemented in the implementation phase. Next, the right side of the V is used to test and integrate these implemented subsystems. Note that the testing and integration happens both at subsystem and system level and design iterations might be required based on the testing results obtained.

In an ideal V-model approach, one should start with listing all the requirements and concerns of different stakeholders before focusing on the conceptual system design. This helps in understanding the problem better and knowing what exactly stakeholders want. However, the goals and scope of this project were well-defined, therefore, we started with the conceptual system design considering the aforementioned goals and scope as the guiding requirements and constraints. Next, the conceptual system design is discussed which lays the foundation for the implementation phase.

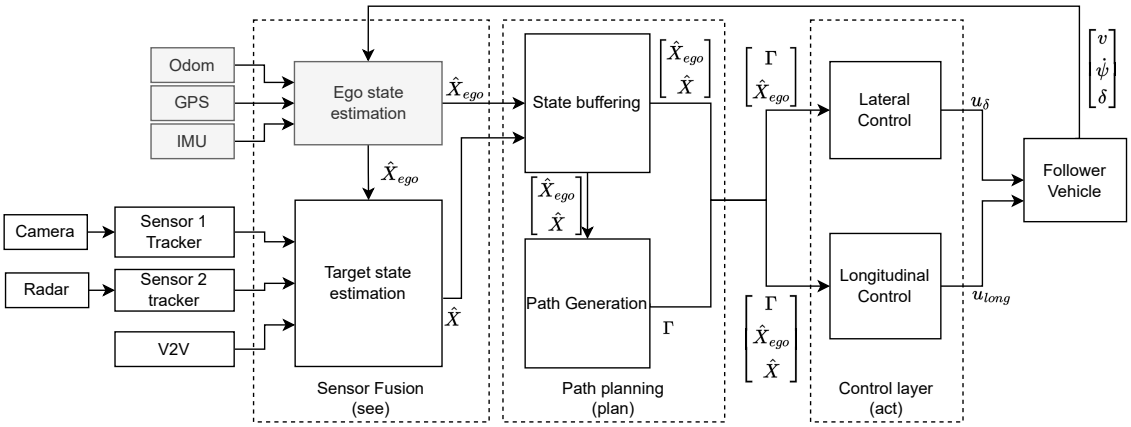


Figure 1.5: System architecture

1.4.2 Conceptual system design

Driving is a highly complex process. Human beings use multiple sensory and cognitive capabilities simultaneously to perform this task. Just like humans, automated driving systems also use the so-called 'see-plan-act' cycle iteratively. Various sensors are used to 'see' the environment. Based on this sensed information, a motion 'plan' is made to safely maneuver through the environment. Then, a controlled 'act' is used to ensure that the 'plan' is followed. A similar approach is used for the conception in this work as well. The complete software pipeline for the automated follower buses is referred to as a 'system'. The system is divided into three sub-systems, namely, sensor fusion, path planning, and controller modules corresponding to the see, plan, and act tasks. Now, each of these sub-systems is briefly explained, then the system architecture is presented showing the interaction of these sub-systems to achieve the task of the system.

The sensor fusion module is responsible for processing the information from different sensors to estimate the poses and/or states of the target $(\hat{X})$ and the ego vehicle $(\hat{X}_{ego})$. As already mentioned in section 1.3, the ego state estimation is not considered in this work and is assumed to be available. Furthermore, in the context of platooning, the *target* is the vehicle that the ego vehicle needs to follow.

The task of the path planning module is to use the ego vehicle and target state information to generate a reference path for the controllers to follow. The path may be represented as a curve Γ in 2D space. In order to make a path, this module also needs to keep a buffer of the state estimates.

Finally, the task of the controller module is to generate control commands, namely, desired longitudinal acceleration u_{long} and desired steering angle input u_δ. These commands are then provided to the vehicle's low-level controllers to maneuver the vehicle.

Figure 1.5 shows the architecture of the whole system outlining the interaction of these three modules. It also shows the expected input(s) and output(s) of each of the modules. The sensor fusion module consists of the ego state $\hat{X}_{ego}$ estimation and the target state $\hat{X}$ estimation blocks. Note that the ego state estimation block is colored gray to show that it is not developed in this work. In the path planning module, first, a buffer of $\hat{X}_{ego}$ and $\hat{X}$ is maintained. This buffer is then used in the path generation block to generate a reference path Γ. Finally, the control layer module uses the path and motion state information to generate control commands. This module consists of the lateral and longitudinal control blocks responsible for computing desired steering angle u_δ and longitudinal acceleration u_{long}

commands respectively. These commands then act on the ego vehicle changing its motion states such as velocity v, yaw rate $\dot{\psi}$, and steering angle δ to maneuver the vehicle.

As seen in Figure 1.5, these modules are interdependent on each other. Thus, it is not trivial to design them individually. To that end, the second goal of this project is used, i.e., first, the control algorithms are designed, and then their formulation acts as the requirement for the design of path generation and sensor fusion algorithms. This sets the basis for the implementation of this project which is outlined next.

1.5 Outline

Chapter 2 outlines the control problem for a platoon and describes the lateral and longitudinal controller design and their implementation in detail. The method of generating reference paths for the controllers to follow is then discussed in chapter 3. Based on the controllers and thereby path generation requirements, sensor fusion algorithms are presented in chapter 4 to estimate the pose of the preceding vehicle accurately. Chapter 5 describes the verification of all the algorithms developed in this work implemented in Siemens Prescan. Finally, conclusions are drawn from this work, and recommendations provided for future work are provided in chapter 6.

2 Controller Design

In this chapter, the vehicle controller design and its implementation are discussed. Before dwelling into the controllers, first, the vehicle model being used is described in section 2.1. Next, the lateral and longitudinal control problems are described in sections 2.2 and 2.3 respectively. Simulations are used to develop and test the control algorithms, the results of which are shown in the respective sections.

2.1 Vehicle model

Similar to most vehicle lateral control literature [8] [9], a bicycle model is used as the vehicle model in this work. It is shown in Figure 2.1. The position (x, y) of the center of the rear axle is expressed in an inertial frame $\vec{r}^{\mathrm{I}}$. Steering angle δ is used to change the curvature κ of the path followed by the rear axle and consequently, the orientation ψ of the vehicle. The planar dynamics showing the relationship between these entities is given by

$$\begin{cases} \dot{x_i}(t) = v_{long,i}(t)\cos\psi_i(t) \\ \dot{y_i}(t) = v_{long,i}(t)\sin\psi_i(t) \\ \dot{\psi_i}(t) = \kappa(t)v_{long,i}(t) \\ \kappa_i(t) = \frac{\tan\delta_i(t)}{l_i}, \end{cases} \tag{2.1}$$

where l is the wheelbase of the vehicle and v_{long} is the forward velocity in the longitudinal direction. Furthermore, the subscript i denotes the index of the vehicle. Thus, l_i and $v_{long,i}$ denote the wheelbase and longitudinal velocity of vehicle i. Additionally, it should be noted that the tire slip is not considered throughout this work, thus, the instantaneous radius R of the path traversed by the rear axle center is given by

$$R_i = \frac{1}{\kappa_i}. \tag{2.2}$$

From (2.1), v_{long} and δ can be seen as the two inputs to the vehicle required to maneuver the vehicle in the $x-y$ plane. However, in practical application, they are not the external inputs provided. Thus it is important to model the dynamics for these inputs. Taking a similar approach as in [4], the steering dynamics is modeled as the first-order dynamics

$$\dot{\delta}_i(t) = \frac{1}{\eta_{\delta,i}}\left(u_{\delta,i}(t) - \delta_i(t)\right) \tag{2.3}$$

in which $\eta_{\delta,i}$ is the time constant and $u_{\delta,i}(t)$ is the desired steering input.

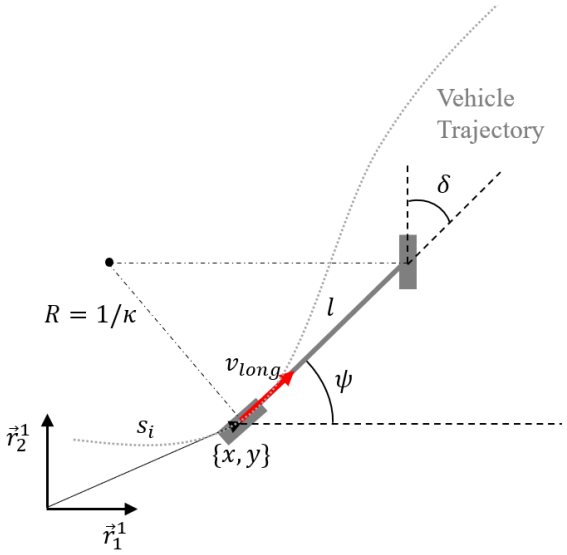


Figure 2.1: Planar kinematic bicycle model

For the longitudinal dynamics, the first-order drive-line dynamics used in [4] is used:

$$\begin{cases} \dot{s}_i(t) = v_{long,i}(t) \\ \dot{v}_{long,i}(t) = a_{long,i}(t) \\ \dot{a}_{long,i}(t) = \frac{1}{\eta_{a,i}} \left(u_{long,i}(t) - a_{long,i}(t) \right), \end{cases} \tag{2.4}$$

where $s_i(t)$ is the curvilinear distance travelled by the rear axle, $v_{long,i}(t)$ is the longitudinal velocity, $a_{long,i}(t)$ is the longitudinal acceleration, $u_{long,i}(t)$ is the desired longitudinal acceleration, and $\eta_{a,i}$ is the drive-line time constant of vehicle i.

Furthermore, the acceleration and steering inputs need to be limited to model the powertrain, brake, and steering limitations. To that end, the following limits are imposed on the desired inputs $(u_{long,i}, u_{\delta,i})$ and the actual inputs $(a_{long,i}, \delta_i)$ using saturation functions

$$\begin{cases} -\bar{a} \leq u_{long,i} \leq \bar{a} \\ -\bar{\delta} \leq u_{\delta,i} \leq \bar{\delta} \end{cases} \tag{2.5}$$

$$\begin{cases} -\bar{a} \leq a_{long,i} \leq \bar{a} \\ -\bar{\delta} \leq \delta_i \leq \bar{\delta}, \end{cases} \tag{2.6}$$

where $\bar{a} = 1.4\, m/s^2$, $\bar{\delta} = 42\,\text{deg}$.

Ideally, the limits should be put only on the actual inputs. However, putting these limits on the desired input allows the actuator dynamics to respect the limits as well.

With the vehicle model established, the design of the lateral and longitudinal controllers is discussed next.

2.2 Lateral control

The task of lateral control is to make the vehicle follow a desired path by reducing the lateral distance and heading angle errors to zero. There are many lateral controllers presented in the literature. A brief overview of these controllers and their comparison can be found in [10]. There it is shown that the simple geometric controllers based on the kinematic vehicle models perform sufficiently well in a variety of situations when compared to the controllers based on dynamic vehicle models. [11] shows the comparison of these geometric controllers and also presents experimental test results. They propose a lateral speed based controller to achieve reasonable tracking performance without sacrificing passenger comfort. The main idea of this controller is to control the lateral speed of the vehicle such that the motion of the vehicle toward the path is smooth. Following a similar approach as presented in [11], first, the kinematic bicycle model described in section 2.1 is reformulated in Frenet coordinates. Then, the control law is presented.

2.2.1 Frenet coordinate representation

The advantage of representing the vehicle model in Frenet coordinates is that it helps to describe the vehicle behavior with respect to the desired path. It also simplifies the error models and allows for deriving control laws that are intuitive and easy to understand. Figure 2.2 illustrates the Frenet frame $\vec{r}^2$ attached to a reference path and how the vehicle's pose is defined with respect to it. The reference path is described by means of a spline curve parameterized by the curvilinear distance s. Here, $X_c(s)$ is the point on this curve which is closest to the center of the rear axle and whose position is expressed in the inertial frame $\vec{r}^1$, i.e., $X_c(s) \overset{\text{def}}{=} (x_c, y_c)$.

Now the deviated path followed by the vehicle can be described in Frenet coordinates $\vec{r}^2$ by means of the lateral distance error l_e and the heading error θ_e. To that end, the center of the rear axle is considered as the reference point of the vehicle, called the control point. Consequently, l_e is the distance between $X_c(s)$ and the control point in $\vec{r}_2^2$ direction, whereas θ_e is the difference in the heading of the vehicle ψ and the tangent θ_c of the reference path at $X_c(s)$, i.e. $\theta_e = \psi - \theta_c$.

Note that the pose of the vehicle at any instance can be calculated with the Frenet state variables s, l_e, and θ_e. The dynamics of these variables are given by

$$\begin{cases} \dot{s} = v_{long} \frac{\cos\theta_e}{1-\kappa(s)l_e} \\ \dot{l}_e = v_{long} \sin\theta_e \\ \dot{\theta}_e = v_{long} \left(\frac{\tan\delta}{l} - \kappa(s) \frac{\cos\theta_e}{1-\kappa(s)l_e} \right), \end{cases} \tag{2.7}$$

with v_{long}, l, and δ being the longitudinal velocity, wheelbase, and steering angle of the vehicle as described in section 2.1, and $\kappa(s)$ is the curvature of the path at $X_c(s)$.

Next, the lateral control law to let these lateral and heading errors converge asymptotically to zero such that the vehicle follows the reference path as closely as possible is discussed.

2.2.2 Control law

The lateral control law used in this work is based on the controller presented in equation (17) in [11]. The complete derivation and stability analysis for the controller is however missing in [11]. To that

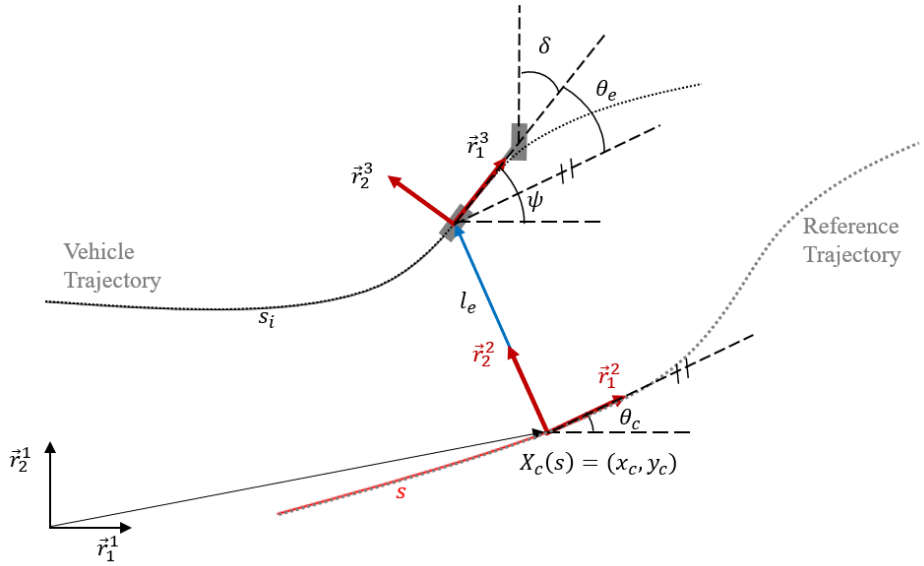


Figure 2.2: Schematic to show the reference path described by $X_c(s)$ in inertial frame $\vec{r}^1$ parameterized by the curvilinear distance s, and representation of the deviated path followed by the vehicle in the Frenet coordinates. Adapted from [4]

.

end, derivation and stability analysis are presented herein. Note that the steering dynamics (2.3) is not included in the control law derivation and its stability analysis. This is done under the assumption that steering dynamics (2.3) is much faster than lateral vehicle dynamics (2.7) to simplify the analysis.

From (2.7), the error dynamics consisting of lateral distance error and heading error can be written as:

$$\begin{cases} \dot{l}_e = v_{long} \sin\theta_e \\ \dot{\theta}_e = v_{long} \left(\frac{\tan\delta}{l} - \kappa \frac{\cos\theta_e}{1-\kappa l_e} \right). \end{cases} \tag{2.8}$$

For brevity, s is dropped from $\kappa(s)$ for the rest of this section.

Now, as mentioned in [11], the exact linearization technique [12] can be used to linearize (2.8).

To that end, let us define $\omega \overset{\text{def}}{=} \frac{\tan\delta}{l}$ as the input to the system. To perform input-output (IO) linearization [13], we can write (2.8) in the following form:

$$\begin{cases} \dot{\mathtt{x}} = \mathtt{f}(\mathtt{x}) + \mathtt{g}(\mathtt{x})\mathtt{u} \\ \mathtt{y} = \mathtt{h}(\mathtt{x}), \end{cases} \tag{2.9}$$

where

$$\mathtt{x} = \begin{bmatrix} l_e \\ \theta_e \end{bmatrix} \tag{2.10}$$

are the error states,

$$\mathtt{f}(\mathtt{x}) = \begin{bmatrix} v_{long} \sin\theta_e \\ -\frac{v_{long}\kappa\cos\theta_c}{1-\kappa l_e} \end{bmatrix} \tag{2.11}$$

is the vector representing the non-linear dynamics of the states,

$$\mathtt{g}(\mathtt{x}) = \begin{bmatrix} 0 \\ v_{long} \end{bmatrix} \tag{2.12}$$

is the vector representing the non-linear relation between the input $\mathtt{u}\,(=\omega)$ and the states, $\mathtt{y}$ is the output and $\mathtt{h}(\mathtt{x})$ is the vector representing the non-linear relation between the output and the states.

Let θ_e be the output of the system, then

$$\mathtt{h}(\mathtt{x}) = \theta_e. \tag{2.13}$$

With this formulation, we have a single input single output system that can be IO linearized by defining a new input W_1 such that

$$\dot{\theta_e} = W_1. \tag{2.14}$$

The relation between W_1 and ω directly follows from (2.8) and (2.14)

$$\omega = \frac{\kappa \cos\theta_e}{1 - \kappa l_e} + \frac{W_1}{v_{long}}. \tag{2.15}$$

It should be noted that with IO linearized θ_e dynamics, one can choose the control input W_1 such that θ_e dynamics is asymptotically stable and $\theta_e \to 0$. However, that does not ensure that the lateral error also goes to zero as even with $\theta_e = 0$, $\dot{l_e} \to 0$, leaving a constant lateral error offset, which might not be zero.

To ensure that $l_e \to 0$, the control law proposed in [11] is used. The motivation for this controller as presented in [11] is that this controller ensures a smooth approach of the vehicle towards the reference path. For completeness, the derivation for this controller is also presented herein. From (2.8), the lateral speed $\dot{l_e}$ depends on θ_e, which in turn depends on the steering input δ. Let $\hat{\dot{l}}_e$ be the desired lateral speed with which the control point (center of the rear axle) must approach the line tangent to the path at $X_c(s)$. The desired lateral speed can be given by introducing a proportional gain k_{lat}

$$\hat{\dot{l}}_e = -k_{lat} l_e. \tag{2.16}$$

From (2.8) and (2.16), the lateral speed error $\dot{l}_{e_{err}}$ can be written as

$$\dot{l}_{e_{err}} = \dot{l}_e - \hat{\dot{l}}_e = v_{long} \sin\theta_e + k_{lat} l_e. \tag{2.17}$$

To reduce this lateral speed error to zero, the controller must steer the vehicle in the direction of $\hat{v}_{long}$ as shown in Figure 2.3.

This can be achieved by introducing another proportional gain k_{head} to this lateral speed error as the input W_1 resulting in the following controller

$$W_1 = -k_{head} \left(v_{long} \sin\theta_e + k_{lat} l_e \right), \tag{2.18}$$

where k_{head} and k_{lat} are the proportional gains for heading and lateral errors. The stability analysis for this controller is presented next.

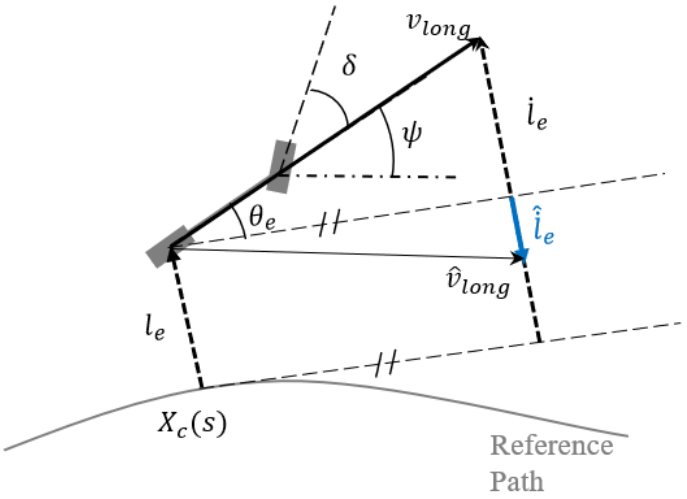


Figure 2.3: Lateral speed controller. Adapted from [11]

Stability analysis

The Lyapunov stability criterion [14] can be used to prove the asymptotic stability of the controller presented in (2.18). A Lyapunov candidate function V can be given by

$$V(l_e, \theta_e) = \frac{1}{2} l_e^2 + 1 - \cos\theta_e, l_e \in \mathbb{R}, \theta_e \in [\pi, -\pi], \tag{2.19}$$

such that $(l_e, \theta_e) = (0, 0)$ is the equilibrium point, i.e.,

$$V(l_e = 0, \theta_e = 0) = 0$$

and

$$V(l_e, \theta_e) > 0, \forall (l_e, \theta_e) \neq (0, 0).$$

Now, taking the derivative of (2.19) and using (2.8) and (2.18) results

$$\dot{V}(l_e, \theta_e) = l_e v_{long} \sin\theta_e - k_{head} v_{long} \sin^2\theta_e - k_{head} k_{lat} l_e \sin\theta_e. \tag{2.20}$$

Considering $v_{long} > 0$ and making the control gain k_{lat} longitudinal velocity dependent such that

$$k_{lat} = \frac{v_{long}}{k_{head}}, \tag{2.21}$$

equation (2.20) reduces to

$$\dot{V}(l_e, \theta_e) = -k_{head} v_{long} \sin^2\theta_e, \tag{2.22}$$

which is negative definite given $k_{head} > 0$. Thus, it is proven that the equilibrium $(l_e = 0, \theta_e = 0)$ is asymptotically stable provided $v_{long} > 0$, $k_{head} > 0$, and $k_{lat} = \frac{v_{long}}{k_{head}}$.

Controller tuning and practical considerations

Substituting W_1 in (2.15) and using the relation between ω and steering input δ, the control law for the desired steering input is given as

$$\begin{cases} u_{\delta,i} = \arctan\left(l\left(-k_{head}\sin\theta_e - \frac{k_{head}k_{lat}l_e}{v_{long}} + \frac{\kappa\cos\theta_e}{1-\kappa l_e}\right)\right) \\ v_{long} \neq 0 \\ \kappa l_e \neq 1. \end{cases} \tag{2.23}$$

The controller gains k_{head} and thereby k_{lat} are tunable parameters with $k_{lat}, k_{head} > 0$. k_{lat} can be seen as the gain that determines how aggressively the controller will try to reduce the lateral error. Thus, its value can be determined by the passenger comfort requirements translated to the lateral speed limits $\bar{\dot{l}}_e$, for example, $\bar{\dot{l}}_e = 1\,m/s$ suggested in [11]. On the other hand, k_{head} can be seen as the gain that determines how aggressively the controller will try to reduce the heading error θ_e. This may be limited by the yaw stability of the vehicle, which is not taken into account in this work. Furthermore, it is assumed that the lateral error is relatively small compared to the reference path's instantaneous radius, i.e., $|\frac{1}{\kappa}| = R >> l_e$. This ensures that $|\kappa l_e| << 1$ and thus the denominator of the third term in (2.23) does not approach zero. To avoid the issue of divisibility by zero in implementation, the desired steering value is set to zero when $v_{long} \to 0$ and/or $(1 - \kappa l_e) \to 0$

Next, performance analysis for the controller developed in this section is provided.

2.2.3 Performance analysis

To analyze the performance of the lateral controller, a simulation study is presented in this section. There are two scenarios considered for the same. First, the stability of the controller is verified when an initial lateral and heading offset with respect to a straight path is given to the vehicle. In second scenario, the tracking performance of the controller is verified with respect to a zigzag path. Various parameters used for the simulation are listed in Table 2.1

Initialization error response

In this scenario, the vehicle is given an initial lateral error of $1m$ and a heading error of $10\,\mathrm{deg}$ with respect to a straight reference path. Figure 2.4 shows the path-following behavior of the vehicle in an inertial frame. The vehicle smoothly converges to the reference path. The lateral error and sine of the heading error are shown in Figure 2.5. Both the errors converge to zero. The steering response of the vehicle is also shown in Figure 2.5. Note that the steering dynamics (2.3) is not considered for this

Table 2.1: Lateral control parameters

Parameters	Values
l	$6\ m$
$\eta_{\delta,i}$	$0.2\ [-]$
k_{head}	$1\ [-]$
k_{lat}	$10\ [-]$

simulation and it is assumed that $\delta = u_\delta$. The vehicle tries to steer sharply towards the right in order to reduce the lateral and heading errors. Furthermore, the steering remains around zero when the errors are reduced to zero asymptotically. This verifies that the lateral controller shows stable behavior.

Tracking response

In this scenario, the vehicle is given a predefined zig-zag path to follow. Initial errors are kept at zero to verify the tracking performance of the controller. Figure 2.6 shows the path-following behavior of the vehicle in an inertial frame. The vehicle follows the reference path sufficiently well. The lateral distance error l_e and sine of heading error θ_e are shown in Figure 2.7. Both errors are close to zero. However, they show 'jerky' behavior and peak at the corners with the maximum lateral and heading error of approximately $0.05\,\mathrm{m}$ and $0.1\,\mathrm{deg}$ respectively. This is due to the approximation error for calculating the minimum distance point ($X_c(s)$) from the ego vehicle position to the reference curve (see Figure 2.2). The approximation errors are higher at the corners where the curvature values are changing. These error values are, however, deemed acceptable for this work. The steering response is also shown in Figure 2.7. Note that the steering dynamics (2.3) are not included in this simulation. Thus, $\delta = u_\delta$ is used for this scenario. It can be seen that the steering values are discontinuous in the first derivative. This observation can be explained by (2.23). If $\theta_e \to 0$, $l_e \to 0$, and v_{long} is constant, it follows from (2.23), $u_\delta \propto \kappa$. The curvature values for the reference path are shown in Figure 2.8. Thus, it can be seen that the steering response indeed follows the curvature profile of the path.

Furthermore, the simulation presented herein is repeated with taking the steering dynamics into account. Figure 2.9 shows the error and steering response. The errors are slightly larger when compared to the case without including the steering dynamics. This is mainly due to the phase lag of the steering input with respect to the desired steering input. This delayed response results in larger errors, especially in the corners making the vehicle either cut or overshoot the corners. For a platoon of vehicles, this behavior is not desirable. However, for the use case of a three-bus platoon considered in this work, this error is considered acceptable.

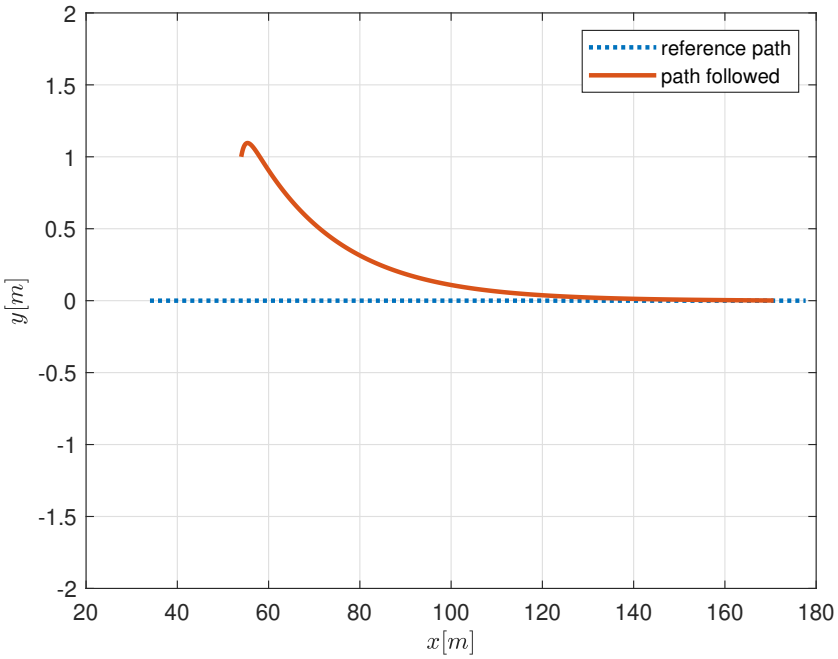


Figure 2.4: Lateral control initialization error response: $x - y$ plot

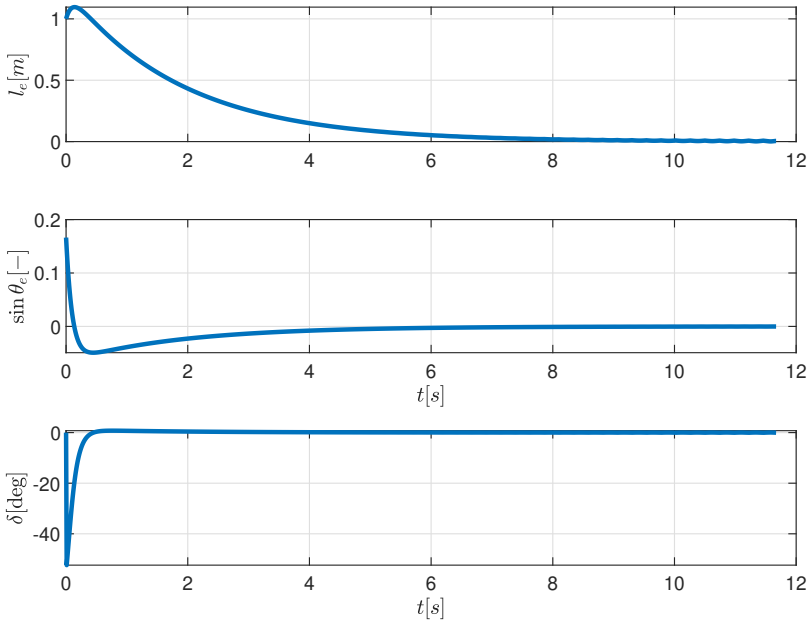


Figure 2.5: Lateral control initialization error response: error and steering plot

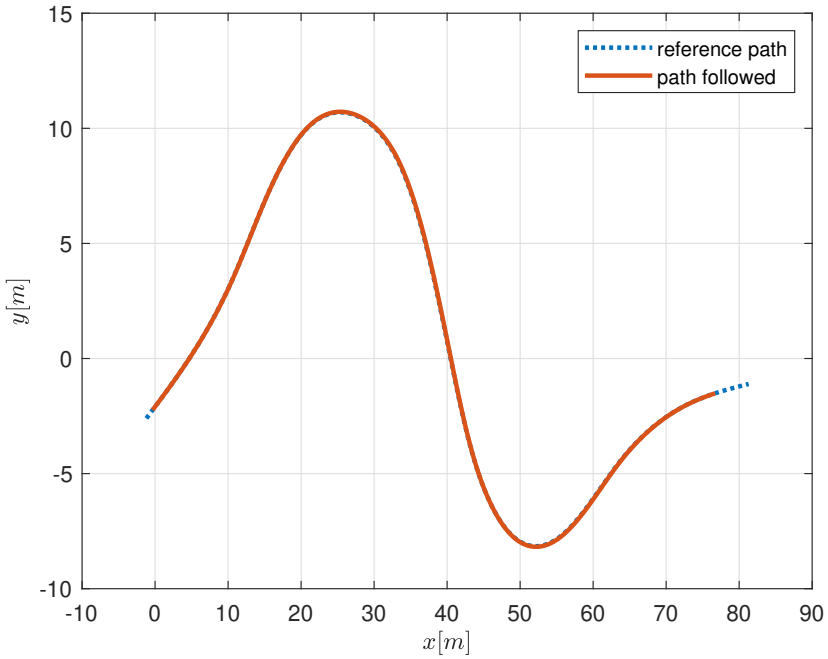


Figure 2.6: Lateral control tracking response: $x - y$ plot

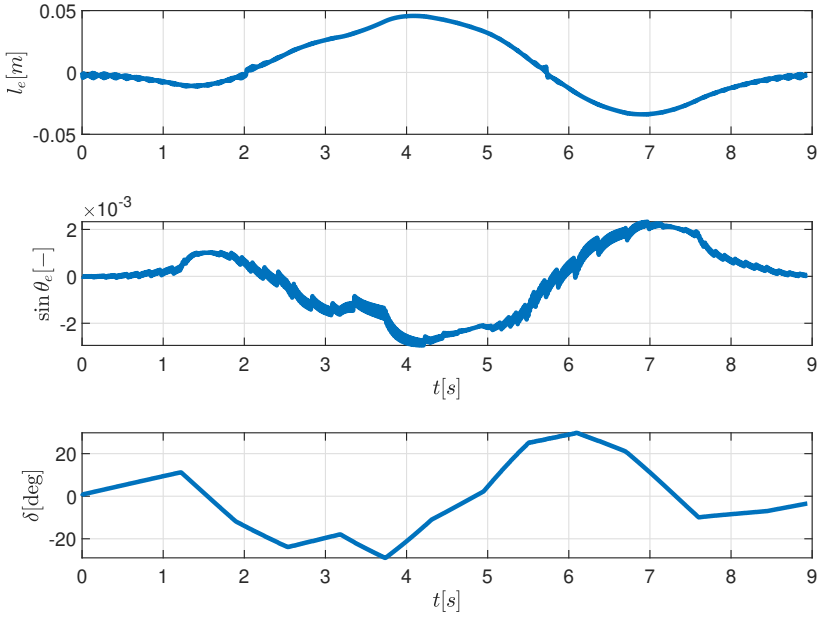


Figure 2.7: Lateral control tracking response: error plot

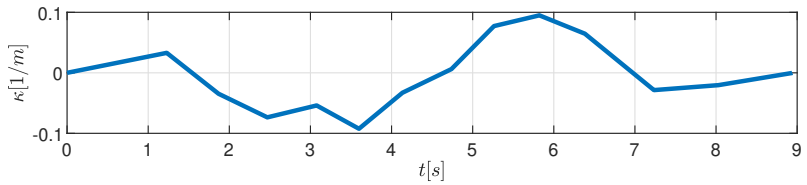


Figure 2.8: Lateral control tracking response: path curvature

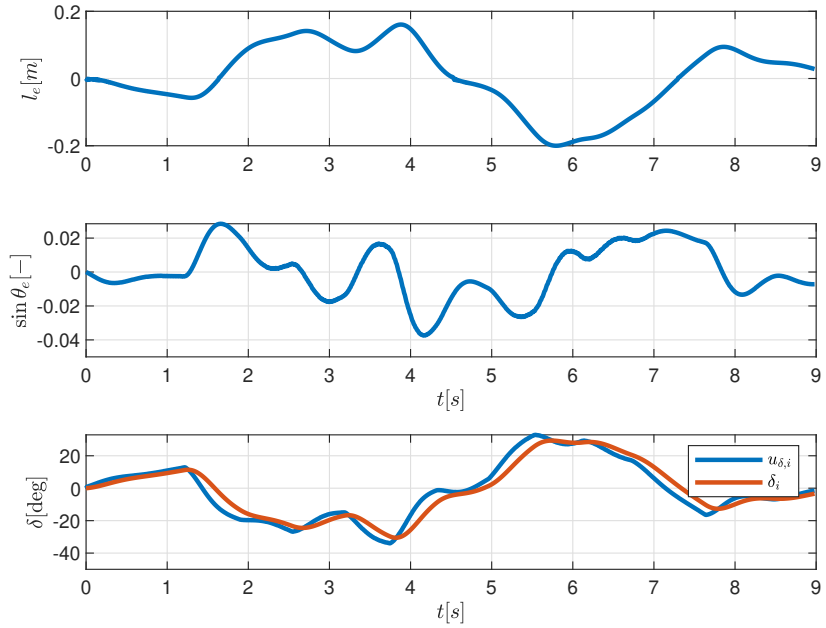


Figure 2.9: Lateral control tracking response: error plot including steering dynamics

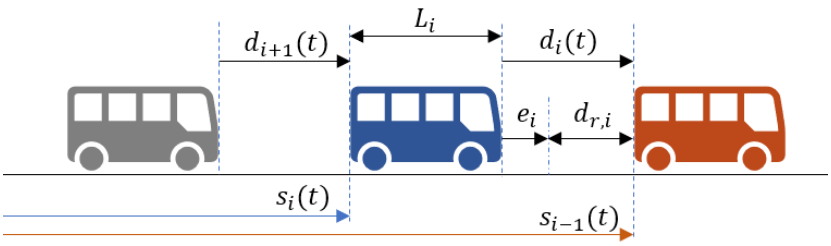


Figure 2.10: Schematic showing the spacing policy and the related errors relevant for longitudinal control of the platoon.

2.3 Longitudinal control

The goal of longitudinal control in a platoon is to maintain a safe but close gap between the vehicles. The most common spacing policy used in literature is the constant time gap [3] [4] which is given by

$$d_{r,i}(t) = r_i + hv_{long,i}(t), \tag{2.24}$$

where r_i, h, and $v_{long,i}$ are standstill distance, time gap, and the longitudinal velocity of the i^{th} vehicle respectively. Figure 2.10 shows the schematic of the spacing policy and related errors in the longitudinal control of a platoon. The actual inter-vehicle distance is calculated as

$$d_i(t) = s_{i-1}(t) - (s_i(t) + L_i)\,, \tag{2.25}$$

where L_i is the overall length of the vehicle, and s_i and s_{i-1} represent the position of the ego vehicle and the preceding vehicle, respectively. Now, the error between the desired gap and the actual gap is defined as the spacing error e_i given by

$$\begin{aligned} e_i(t) &= d_i(t) - d_{r,i}(t) \\ &= (s_{i-1}(t) - s_i(t) - L_i) - (r_i + hv_i(t))\,, \end{aligned} \tag{2.26}$$

that the longitudinal controller tries to reduce to zero.

One of the most commonly used techniques to achieve it in a string stable manner is cooperative adaptive cruise control (CACC) [3]. Interested readers can further refer to [3] for the derivation of this controller and string stability analysis. The control law for such a controller is given by

$$\dot{u}_{CACC,i} = -\frac{1}{h}u_{CACC,i} + \frac{1}{h}\left(k_p e_i + k_d \dot{e}_i + k_{dd}\ddot{e}_i\right) + \frac{1}{h}u_{long,i-1}. \tag{2.27}$$

Here $u_{CACC,i}$ is the desired CACC longitudinal acceleration for the ego vehicle, $u_{long,i-1}$ is the desired acceleration of the preceding vehicle, and k_p, k_d and k_{dd} are the control gains.

With this control law, one can compute the desired longitudinal acceleration, for instance, $u_{long,i} = u_{CACC,i}$ which can be used in the longitudinal dynamics in (2.4). However, there are some design considerations that need to be addressed in the practical implementations, which are discussed next.

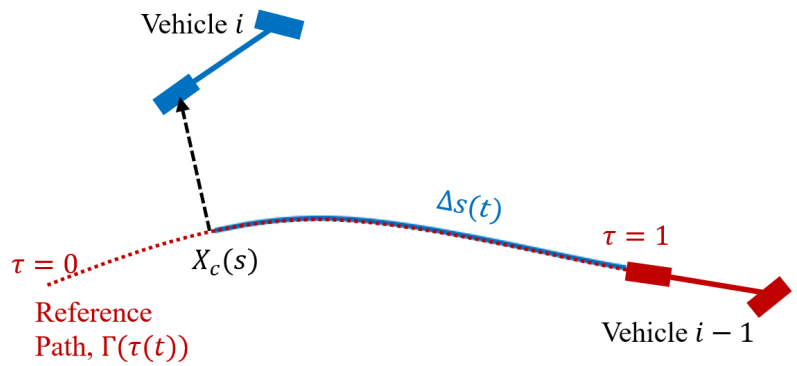


Figure 2.11: Curvilinear distance based inter-vehicle gap calculation

2.3.1 Practical considerations

There are some practical design considerations that need to be addressed during the implementation of this controller.

- V2V communication is required to communicate the desired acceleration ($u_{long,i-1}$) of the leading vehicle $i-1$. Wireless communication can be used and its applicability is demonstrated in practice in the literature as well [3]. The communication delay, however, plays a crucial role in string stability which is not considered in this work.

- The actual gap between the two vehicles is not trivial to calculate or estimate. To that end, the parameterized reference path, $\Gamma(\tau)$, traced by the predecessor is used to calculate the curvilinear distance based gap in this work. It is schematically shown in Figure 2.11. The footpoint $X_c(s)$ of the i^{th} vehicle on the reference path is used to determine the curve parameterization parameter τ (see section 3.3) at time t. Now, using the value of τ at $X_c(s)$ and at the end of the reference path (which is 1, considering τ is normalized), we can approximate the gap between the vehicles
$$d_i(t) = \Delta s(t) - L_i = (\Gamma_i(1) - \Gamma_i(\tau(t))) - L_i. \tag{2.28}$$

- The longitudinal controller (2.27) requires calculation of e_i, $\dot{e}_i$, and $\ddot{e}_i$. In practical implementation, e_i can be computed using (2.28) and (2.26). Now, taking the derivative of (2.26), we obtain
$$\dot{e}_i = v_{long,i-1} - v_{long,i} - h a_{long,i}, \tag{2.29}$$
which can be computed by measuring the longitudinal velocity of the preceding vehicle $v_{long,i-1}$. The onboard sensors on the ego vehicle or V2V communication can be used for it. The measurements are typically noisy and thus require state estimation filtering. Finally, the $\ddot{e}_i$ term is not trivial to compute. Moreover, [3] shows that the error dynamics are stable with $k_{dd} = 0$, thus, it is ignored in this work.

- It can be seen from (2.27) that the controller only accounts for the spacing error. This might lead to some undesired behavior in practical scenarios, especially in corners. Consider a scenario where the preceding vehicle exits a turn and starts to accelerate, leading to an increased spacing error. To reduce the spacing error, the follower vehicle will try to accelerate while still

being in the corner. To tackle such situations, a curvature based cruise controller (CBCC) is implemented. The idea is that the velocity of the ego vehicle, and consequently the desired acceleration $u_{long,i}$ is limited to a safe threshold, irrespective of the spacing error. The velocity threshold $\bar{v}_{long,i}$ is calculated based on the path curvature the ego vehicle is following and is given by

$$\bar{v}_{long,i} = \sqrt{\frac{\bar{a}_{lat}}{|\hat{\kappa}_{max,i}|}}, \tag{2.30}$$

where $\hat{\kappa}_{max,i}$ is the maximum curvature of the path over a finite horizon and $\bar{a}_{lat}$ is the lateral acceleration threshold acceptable in public transport. The value for $\bar{a}_{lat}$ is taken as $0.98m/s^2$ from [15].

A proportional gain $k_{cc} > 0$ is then used to calculate the desired CBCC longitudinal acceleration to limit vehicle's speed to $\bar{v}_{long,i}$ which is given by

$$u_{CBCC,i} = -k_{cc}\left(v_{long,i} - \bar{v}_{long,i}\right). \tag{2.31}$$

Now, the minimum of the two accelerations presented in equations (2.27) and (2.31) is chosen as the desired longitudinal acceleration setpoint for the vehicle, i.e.

$$u_{long,i} = \min\left(u_{CACC,i}, u_{CBCC,i}\right), \tag{2.32}$$

which can be used in longitudinal dynamics (2.4) to achieve a longitudinal control that makes vehicle i follow vehicle $i-1$ closely in a safe and comfortable manner.

Next, a simulation study is presented to show the performance of the longitudinal control.

2.3.2 Performance analysis

To evaluate the performance of the longitudinal controller, a simulation study is presented in this section. Similar to the lateral controller case, two scenarios are considered. First, the stability of the longitudinal controller is verified in a scenario where an initial spacing error is given. The second scenario is presented to verify the tracking performance of the longitudinal controller in terms of following the acceleration profile of the preceding vehicle. For both scenarios, a straight path is assumed. Thus, the gap between the vehicles can be calculated by simply calculating the Euclidian distance between them. However, in actual implementation, the curvilinear distance should be used

Table 2.2: Longitudinal control parameters

Parameters	Values
$\eta_{a,i}$	0.2 $[-]$
h	0.5 $[s]$
r_i	2 $[m]$
k_p	0.2 $[-]$
k_d	0.7 $[-]$
k_{dd}	0 $[-]$
k_{cc}	0.5 $[-]$

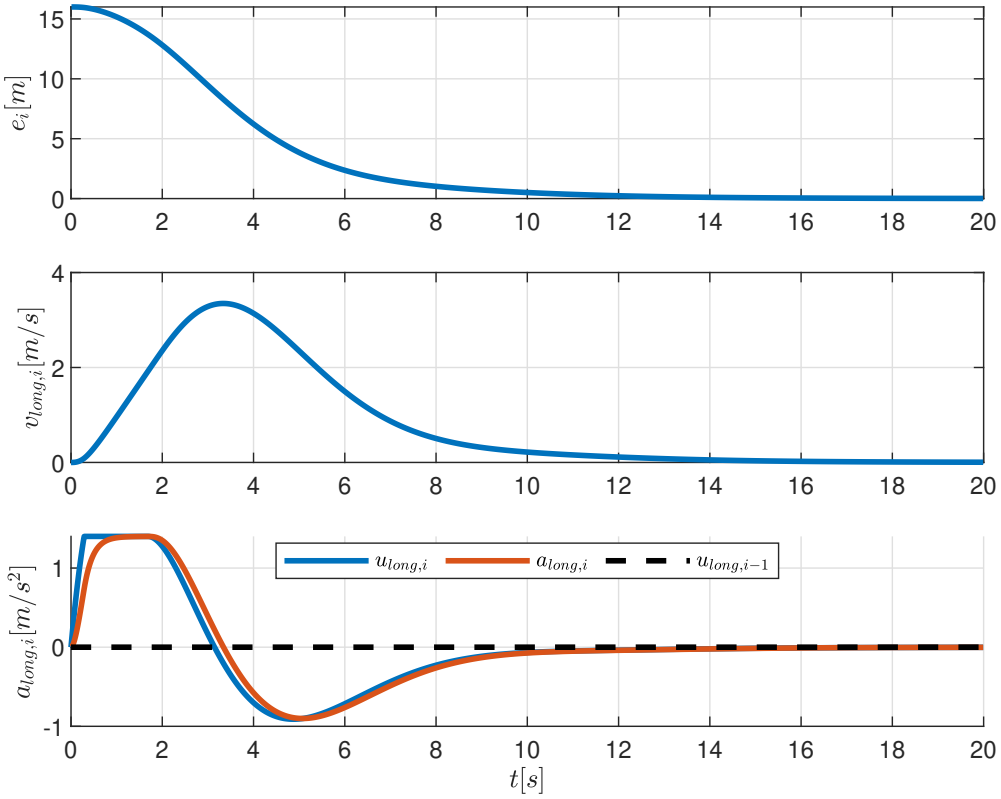


Figure 2.12: Longitudinal control: initialization error response

as depicted in Figure 2.11, which requires parameterized path traversed by the preceding vehicle. Furthermore, various parameters related to the vehicle model, spacing policy, and control gains used for the simulation are listed in Table 2.2.

Initialization error response

In this scenario, the preceding vehicle is standing at rest in front of the ego vehicle on a straight path such that there is an initial spacing error of $16m$. The longitudinal behavior of the ego vehicle in this scenario is shown in Figure 2.12. The vehicle starts to accelerate to close the gap and reduce the spacing error. Note that the first order longitudinal dynamics (2.4) is also included in this simulation along with the acceleration limit (2.6). As the gap reduces, the vehicle again decelerates to reduce the velocity again to zero such that the spacing error remains zero, showing the asymptotic stability of the controller.

Tracking response

For this scenario, both the ego vehicle and the preceding vehicle are initially moving with a constant speed of $10\ m/s$ with zero spacing error between them on a straight path. The acceleration profile

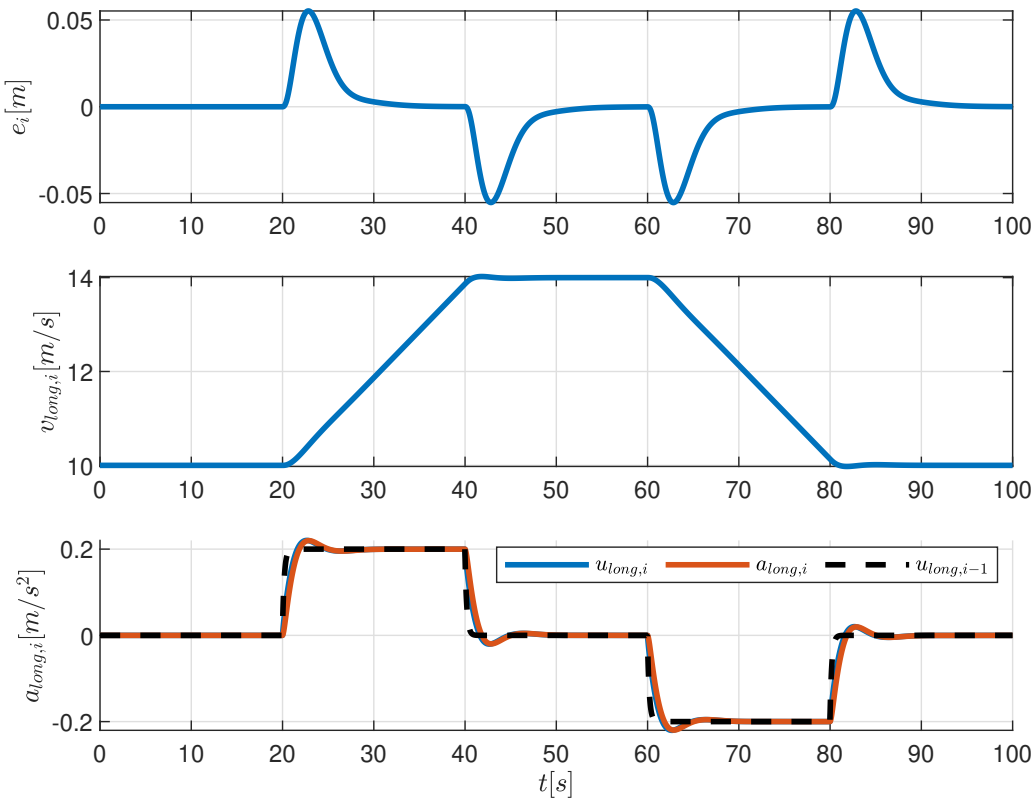


Figure 2.13: Longitudinal control: tracking response

followed by the preceding vehicle $u_{long,i-1}$ is shown in Figure 2.13. Figure 2.13 also shows the longitudinal behavior of the ego vehicle in response to the acceleration profile followed by the preceding vehicle. It can be seen that the ego vehicle follows the acceleration profile of the preceding vehicle reasonably well. The spacing error also remains zero except for the transient periods of acceleration and deceleration.

The performance of the longitudinal controller is deemed sufficient for this work and its performance in the closed-loop simulation in different scenarios is further discussed in chapter 5.

2.4 Summary

In this chapter, the design and implementation of a longitudinal and a lateral controller are discussed. The performance of these controllers is also studied using simulation. It can be seen that both controllers require a reference path to follow. Generation of reference paths in real-time is one of the challenging tasks in platooning. This is discussed in the next chapter.

3 Path Generation

In the context of the platooning use case considered in this work, the automated follower vehicles require a reference path to follow. The follower vehicles need to construct this reference path in real time. There are mainly three ways to construct the path:

1. Platoon leader's trajectory based: The trajectory of the platoon leader can be communicated via wireless communication to the follower vehicles. Thus, all the follower vehicles have the same trajectory to follow.

2. Preceding vehicle's trajectory based: In this approach, the path traversed by the immediate predecessor is used as the reference. The onboard sensors such as radar or camera can be used to estimate the position of the preceding vehicle and use this information as waypoints to fit a smooth curve through them.

3. Infrastructure based: Features in the infrastructures can also be used to generate a reference path. The road center line is one of the obvious choices for the reference path. Different methods can be used to determine the lateral position of the vehicle with respect to the road center line. Camera based computer vision techniques can be used. Some applications may also use the magnetic markers for the same purpose.

All these approaches have some advantages and disadvantages. Both the first and the second approaches are more robust and flexible when compared to the third approach as they are infrastructure independent and try to mimic the dynamic driving maneuvers of the preceding/leader vehicle, for example, avoiding an obstacle or taking a different route, etc. On the other hand, both the first and second approaches rely on safe driving behavior from the preceding or lead vehicle.

In this work, it is assumed that the lead vehicle is driven by an experienced driver and is responsible for taking a safe path. Thus, path generation based on the platoon leader's trajectory seems to be the best choice as it is road infrastructure independent and potentially the most precise considering localization in modern vehicles is fairly accurate. However, this approach is overdependent on wireless communication, which can be unreliable. Considering these aspects, path generation based on the preceding vehicle's trajectory is selected for further investigation in this work. This approach allows for designing a more modular, flexible, and reliable solution.

In this chapter, first, the path generation problem based on the preceding vehicle's trajectory is mathematically formulated. Then, a detailed explanation of the algorithms used to solve this problem is provided.

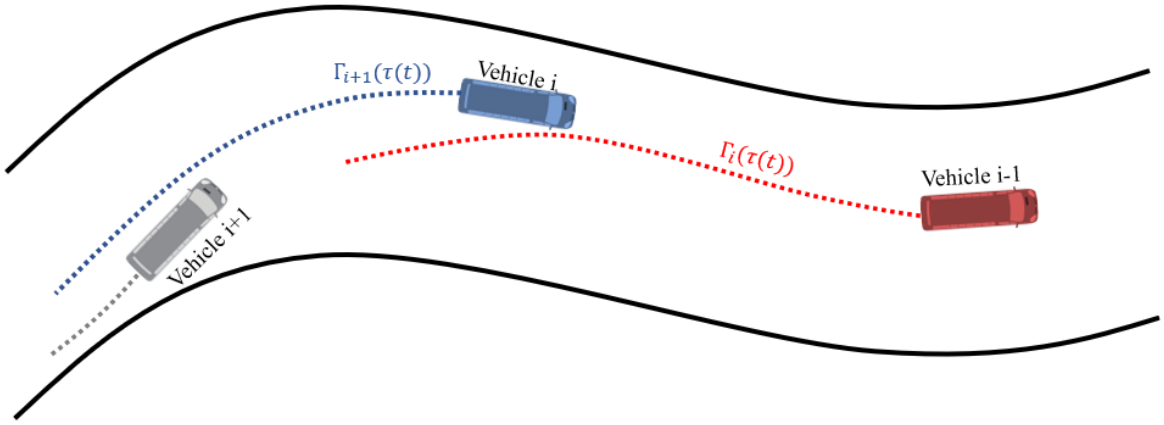


Figure 3.1: Schematic showing path following for a platoon of three vehicles

3.1 Problem formulation

Figure 3.1 schematically shows the convoy of three vehicles considered as our use case. The leader vehicle $i-1$ is manually driven. It is responsible for avoiding obstacles and taking collision-free paths. Vehicle i is just behind it and it can see the leader vehicle. Thus, vehicle i can construct the path $\Gamma_i(\tau(t))$ of its leader from its observation and use it as a reference path to follow. Furthermore, vehicle $i+1$ cannot see the leader vehicle $i-1$ as vehicle i will obstruct its field of view. To that end, vehicle $i+1$ can construct the path $\Gamma_{i+1}(\tau(t))$ driven by its immediate predecessor and use it as the reference. Ideally, this arrangement should result in vehicle $i+1$ following the exact path of vehicle i which is following the exact path of vehicle $i-1$, and thus making all the follower vehicles in the platoon follow the exact path of the leader.

Before focusing on how to construct the path for the vehicles, it is important to choose which reference point of the preceding vehicle one needs to track in order to construct the path for the ego vehicle.

Reference point to track

From the perspective of lateral control, the center of the rear axle seems to be the most obvious choice for the reference point to track as this point on the vehicle is made to follow the prescribed path in most of the lateral control literature [11] [8]. Thus, it is wise to construct a path that is followed by the preceding vehicle's rear axle center point. This choice is justified as the vehicles are usually front-steered and the rear tires point towards the longitudinal direction, consequently making them ideal for placing the speed sensor to measure the longitudinal velocity, which is used in describing the vehicle dynamics (2.1). In the case of longer vehicles like buses, considering the geometric center point as the reference point might be beneficial, taking the swept path into account [16]. In this case, however, the vehicle's side slip angle is required to estimate the longitudinal velocity which is not directly measured. Thus, the geometric center is not considered as the reference point in this work for simplicity.

On the other hand, from the sensing perspective, measuring the states (position, velocity) of the rear axle center or the geometric center is generally not possible with common onboard automotive sensors

like radar and cameras. However, utilizing the V2V communication and under the assumption of knowing the preceding vehicle's dimensions, it is possible to estimate the states of the rear axle center as described in [17]. Thus, the center of the rear axle is used as the reference point to track in this work.

Now, the following sections describe the procedure of obtaining the path traversed by the preceding vehicle assuming its position is known.

3.2 Waypoint management

Let us define the position vector of the reference point of the preceding vehicle as $p \stackrel{\text{def}}{=} (x_r, y_r)$. Estimation of this position vector is discussed in chapter 4.

Now, these position estimates p act as the waypoints in a 2D space through which the path for the ego vehicle is generated. Let $\mathbf{P}$ be the set of such j waypoints.

$$\mathbf{P} = \{p_1, p_2, ..., p_j\}, \tag{3.1}$$

where p_1, p_2, ..., p_j are the waypoints observed at time steps $k = 1, 2, ...j$.

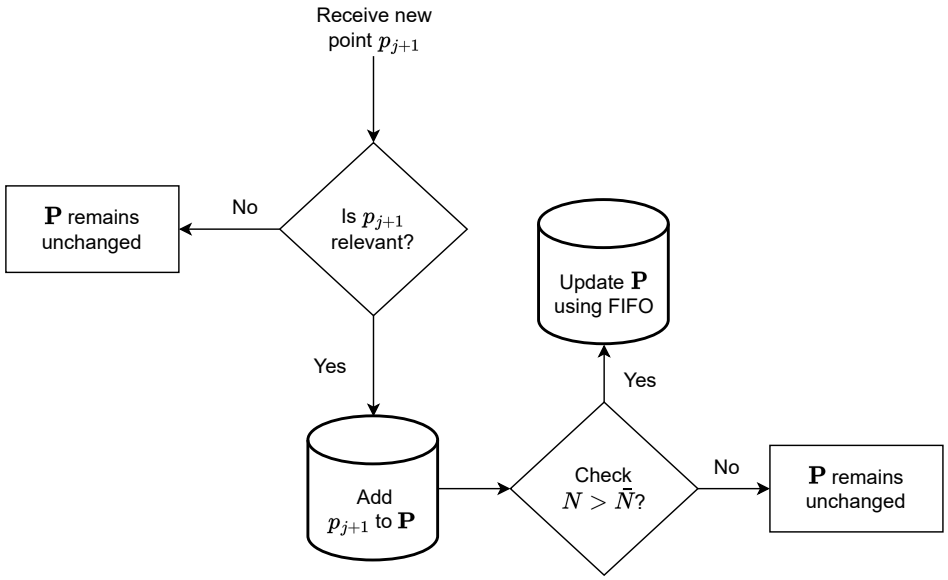


Figure 3.2: Waypoint management strategy workflow

Before moving to methods of generating a path based on these waypoints, it is important to note the following aspects:

- To ensure real-time performance and memory requirements, the number of waypoints to be handled has to be managed and should be restricted to an upper bound, say $\bar{N}$
- Not all the observed waypoints are relevant for path generation. For example, a very slow moving or standstill preceding vehicle results in a new waypoint that is very close or equal to the most recent waypoint in $\mathbf{P}$.

To incorporate these aspects, a waypoint management strategy shown in Figure 3.2 is employed. When the new waypoint p_{j+1} is received, first, it is checked whether it is relevant to append to $\mathbf{P}$. The Euclidean distance between the most recent point in $\mathbf{P}$ and the new point is calculated. If it is higher than a distance threshold, say $\bar{r}$, it is appended to the list $\mathbf{P}$. Furthermore, the number of points in the list is capped. As the number of points in the list, N, exceeds $\bar{N}$, the first-in-first-out principle is used to remove the oldest point from the list.

With this procedure, we obtain a list of points $\mathbf{P}$ that represents a polygon chain of the waypoints of the preceding vehicle in an inertial frame. These waypoints, however, cannot be used as the path for controllers as they are noisy and only C^0 continuous. Most of the controllers rely on the geometric derivatives [8] [4]. To that end, smoothing techniques are used which are discussed next.

3.3 Curve fitting and smoothing

There are mainly two techniques that can be used to fit a curve through a given set of points, namely, interpolation and approximation. With interpolation, the fitted curve passes through all the points, whereas in approximation it may not. The presence of noise in the measurements makes the interpolation technique inappropriate. Thus, the approximation technique is considered in this work. A piece-wise cubic spline approximation method based on least-squares adjustment [18] is implemented.

In this method, a parametric, two-dimensional piecewise spline is calculated. The spline is represented as

$$\Gamma(\tau) = \begin{cases} \gamma_1(\tau)\ , \tau_0 \leq \tau < \tau_1 \\ : \\ \gamma_n(\tau)\ , \tau_{n-1} \leq \tau < \tau_n \end{cases}, \tag{3.2}$$

where τ is the parameterization variable, γ_i are the n spline segments with $n+1$ strictly monotonic breaks $\tau_0 < \tau_1 < .. < \tau_n$, commonly referred as knots in literature. These spline segments are represented as

$$\gamma_i(\tau) = \begin{bmatrix} \tilde{x}(\tau) \\ \tilde{y}(\tau) \end{bmatrix} \begin{bmatrix} a_{i,0} + a_{i,1}\tau + .. + a_{i,k}\tau^k \\ b_{i,0} + b_{i,1}\tau + .. + b_{i,k}\tau^k \end{bmatrix}, \tag{3.3}$$

where k is the degree of the splines, $a_{i,0}, .., a_{i,k}$ and $b_{i,0}, .., b_{i,k}$ are the polynomial coefficients of the i^{th} spline segment.

The algorithm tries to determine the polynomial coefficients for each spline segment by minimizing the squared distance between the spline and the waypoints, using Lagrange multipliers to take the continuity constraints into account [18]. Interested readers are requested to refer to [18] for further explanation. The MATLAB implementation developed in this work is provided in appendix C.1. If λ is the required parametric continuity (C^λ) at the knots τ_i, then the polynomial coefficients can be calculated by solving a linear matrix equation of the form

$$\mathbf{AX} = \mathbf{B},\ \mathbf{A} \in \mathbb{R}^{\alpha \times \alpha},\ \mathbf{B} \in \mathbb{R}^{\alpha \times 2}, \tag{3.4}$$

with $\alpha = (k+1)n + (\lambda+1)(n-1)$. Here, matrix $\mathbf{A}$ captures the sum of squared distance terms and constraints, matrix $\mathbf{X}$ denotes the unknown coefficients and the Lagrange variables, and $\mathbf{B}$ represents

the measurements and residuals. Kindly refer to equation (91) of [18] for the exact formulation of these matrices.

Note that the computational load (mainly for calculating $\mathbf{A}^{-1}$) in solving (3.4) does not depend on the number of waypoints, rather it depends on the number of spline segments the user wants to create, the degree of the polynomial, and the parametric continuity required. However, the number of waypoints does play a significant role in spline approximation as the goodness of fit depends on the noise in the position estimates, the number of points in a spline, and design parameters like n, k, and λ [18]. Thus, these parameters need to be chosen based on the real-time performance, and continuity requirements from the controller.

3.4 Parameter selection

In this section, the selection of the design parameters related to spline curve approximation and their effect is briefly discussed.

Distance threshold, $\bar{r}$

As mentioned above, distance threshold $\bar{r}$ is used to prevent unnecessary waypoints being appended in $\mathbf{P}$, especially in a stop-go scenario. The expected measurement noise while measuring the position of a standstill vehicle can be used to determine $\bar{r}$. However, having $\bar{r}$ too high may result in discarding necessary waypoints, which can be crucial, for example, in slow cornering scenarios. In this work, $\bar{r}$ is taken as $0.5m$.

Capping threshold, $\bar{N}$

In order to approximate splines, sufficient waypoints should be available in the buffer $\mathbf{P}$. On the other hand, having too many waypoints might not be useful considering vehicles are following closely, which can result in unnecessary memory requirements. Based on [8], $\bar{N}$ is chosen to be 100 in this work.

Degree of spline polynomials, k

From the lateral control law (2.23) presented in section 2.2, it can be seen that the curvature of the path needs to be calculated. The curvature of a parameterized spline is given by [4]

$$\kappa = \frac{\tilde{y}(\tau)''\tilde{x}(\tau)' - \tilde{x}(\tau)''\tilde{y}(\tau)'}{\left(\tilde{x}(\tau)'^2 + \tilde{y}(\tau)'^2\right)^{3/2}}, \tag{3.5}$$

where $'$ and $''$ represent the derivative and double derivative respectively concerning the parameterization variable τ introduced earlier in this section. Thus, a third-degree polynomial ($k = 3$) is used in this work to ensure the calculation of the derivatives with sufficient accuracy while keeping the computation load in check.

Parametric continuity, λ

Following a similar argument presented above, λ is chosen as 2 to ensure C^2 continuity of the curve at the knots.

Number of spline segments, n

The choice of the number of spline segments depends on the number of waypoints, j, present in the buffer list $\mathbf{P}$. Theoretically, one cubic polynomial piece can be used to approximate the entire path. However, at least two spline pieces are required to utilize the continuity constraints in the approximation problem formulation presented in section 3.1 in [18]. Thus, the lower bound $\tilde{n}$ for the number of spline segments is taken as two, i.e. $\tilde{n} = 2$. In practice, one would want as many polynomial pieces as possible to represent the path precisely. However, having too many polynomial pieces can result in a path that might encompass the measurement noise, which is not desired. In this work, the upper bound on the number of splines is taken as $\bar{n} = 10$ arbitrarily.

Number of points in a spline, m

In order to calculate the coefficients of a third-degree polynomial, at least four points are generally required. However, applying continuity constraints at the knots can help in determining these coefficients with fewer points. This criterion is thus can be used to determine the lower bound on the number of points ($\tilde{m}$) required in one spline piece. So, for a spline section of second-order continuity, at least two points are required (the knot with three continuity constraints and one other waypoint), i.e., $\tilde{m} = 2$. For the upper bound on the number of points ($\bar{m}$) in a spline piece, the upper bound on the number of splines ($\bar{n}$) can be used:

$$\bar{m} = \lfloor j/\bar{n} \rfloor + j \mod \bar{n}, \tag{3.6}$$

where j is the total number of waypoints.

Minimum number of waypoints, $\tilde{j}$

We can use the lower bounds on the number of spline segments $\tilde{n}$ and the number of points in a spline $\tilde{m}$ to calculate the minimum number of waypoints $\tilde{j}$ required for approximating a spline, i.e.

$$\tilde{j} = \tilde{n}\tilde{m} = 4. \tag{3.7}$$

Thus, the algorithm waits for at least four waypoints to be available in the buffer to approximate the spline and thereby provide a reference path for the controllers.

With the above parameters, an example of a cubic spline approximation is shown in Figure 3.3. It uses a set of hundred waypoints representing the noisy estimated positions for an arbitrary curve. The approximated spline follows the ground truth reasonably well with $0.5m$ root mean squared error. Furthermore, the approximated spline is also able to avoid major measurement noise deviations, for instance, in the vicinity of $(44, 2)$ and $(22, 15)$. With this fitting, the spline approximation method proposed in this chapter is deemed satisfactory for this work.

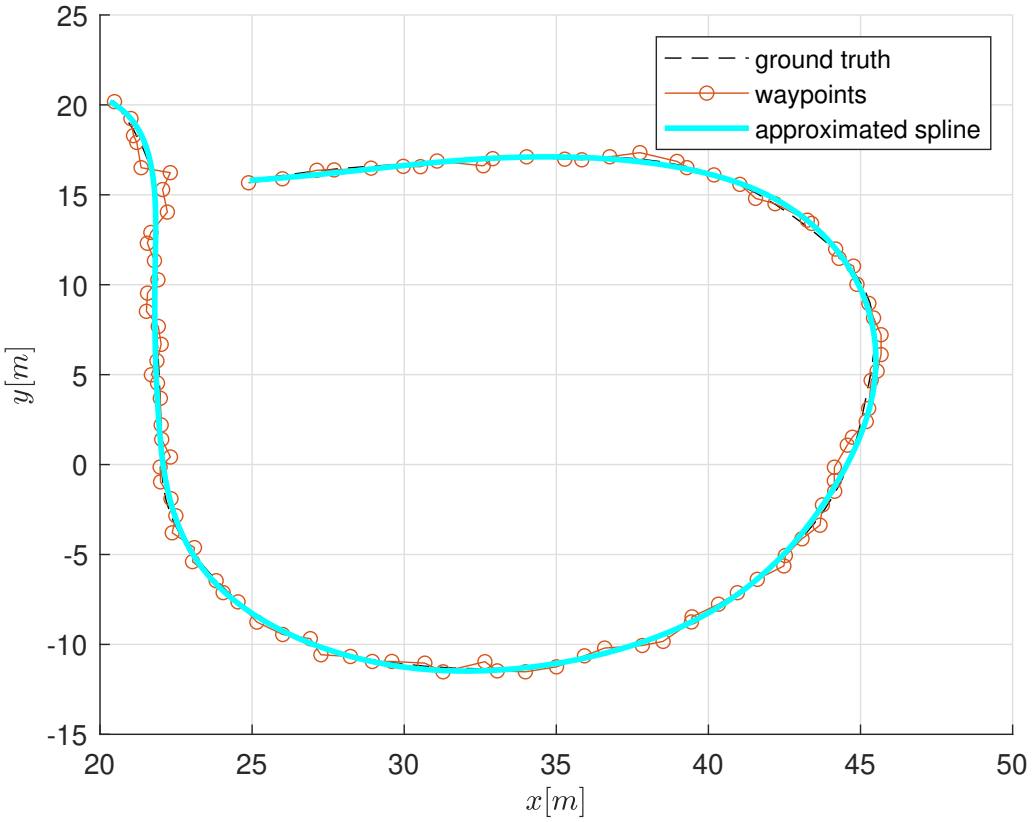


Figure 3.3: Cubic spline approximation for an arbitrary set of points based on least-squares adjustment [18] with $k = 3$, $\lambda = 2$, $n = 10$

3.5 Summary

In this chapter, the design and implementation of the path generation algorithm for follower vehicles is discussed. A cubic piecewise spline approximation method is implemented to fit a path for the ego vehicle from the position estimates of the preceding vehicle. The effects of various parameters for spline approximation are also studied.

4 State Estimation

Chapters 2 and 3 laid the theoretical foundation for implementing the automated lateral and longitudinal control of follower vehicles in a platoon. Different algorithms required to perform these control actions are also discussed in these chapters. To implement these algorithms, however, information regarding the pose of the preceding vehicle is required. This chapter discusses different algorithms to estimate this information using the onboard sensors on the vehicles. First, the requirements from previous chapters are listed to understand what information is exactly required from the perspective of controller implementation. Then, the problem of state estimation of a preceding vehicle using only onboard sensors is discussed in detail.

4.1 Control oriented requirements

In this section, the requirements from the controllers and thereby path generation are derived. There are mainly two requirements. The first requirement comes from path generation where the estimation of the position of the rear axle center is required. Recall that a buffer $\mathbf{P}$ of these position estimates is maintained in the path generation algorithm. These position estimates can either be in the relative frame or the inertial frame. However, from an implementation point of view, it is efficient to have the position estimates in the inertial frame to avoid the transformation of all the waypoints every time the algorithm is called to accommodate for the change in the ego vehicle position. Thus, the requirement from path generation is to estimate the position of the rear axle center of the preceding vehicle in an inertial frame. This path generation requirement already covers the requirement for the lateral controller used in this work as it only requires the geometric properties of the reference curve. The second requirement comes from the longitudinal controller which requires estimating the longitudinal velocity and acceleration of the preceding vehicle (referring to equations 2.29 and (2.27)). Note that we require the desired longitudinal acceleration ($u_{long,i-1}$) state of the preceding vehicle for CACC implementation. However, an acceleration estimate ($a_{long,i}$) might be useful as a fallback scenario in case of communication loss.

In a nutshell, state estimation of the position, velocity, and acceleration states of the preceding vehicle in an inertial frame is needed to successfully implement the control algorithms. To that end, the preceding vehicle's state estimation problem with onboard sensors is discussed next.

4.2 Preceding vehicle state estimation

This section describes the preceding vehicle state estimation problem using onboard sensors on the ego vehicle. Different sensors have different strengths based on their measurement principles. For

instance, radars provide accurate range measurement but due to their limited angular resolution, measure the lateral position with less accuracy. On the other hand, cameras are not accurate in range measurement but can provide better lateral position measurement due to advanced computer vision methods employed on the images captured by them. Furthermore, the velocity of the target is not measured directly from the camera. Thus, camera and radar sensors can be seen as complementary sensors with the camera being better at lateral position measurements while radar being better at longitudinal position and velocity measurements. Therefore, fusion algorithms are often employed to fuse data from different sources to produce overall more accurate state estimates[19]. Additionally, the sampling frequencies of these sensors are different, thereby producing measurements at different rates. Thus, the fusion algorithms need to account for this aspect as well.

One can utilize a state estimation filter such as the Kalman filter to fuse different state measurements by the sensors directly as presented in section 3.3 in [20]. However, state estimation and tracking algorithms are implemented in automotive sensors by their manufacturers, producing object-level filtered estimates, called *tracks*. To utilize the existing estimation and tracking pipeline of the sensors, track-to-track-fusion (T2Tf) can be used. T2Tf allows for a flexible method to fuse the track-level data from different sensors coming at different rates and forms, making it relatively easy to add and remove sensors from the sensor suit.

For T2Tf implementation, track-level measurements and corresponding covariances are required [19]. However, sensors may not produce the covariance values in practice. Furthermore, the underlying state estimation and filtering implementation for the sensors are proprietary and not shared. Thus, in this work, first, a state estimation algorithm is presented for the individual sensors. This algorithm can be treated as the local tracker for each sensor that tracks the preceding vehicle. Then, a track-to-track fusion (T2Tf) algorithm is presented which fuses the tracks from these local trackers to produce a central track of the preceding vehicle to be used by the controllers.

4.2.1 Sensor level tracking

In this subsection, the state estimation or tracking of the preceding vehicle using a sensor is discussed. First, the tracking problem is formulated mathematically. Then, the implementation of a discrete Kalman filtering based recursive filter is presented to recursively produce state estimates of the preceding vehicle.

Problem formulation

Figure 4.1 schematically shows the state estimation problem wherein the position, velocity, and acceleration of the reference point R of the preceding vehicle needs to be estimated using measurements from an onboard sensor S on the ego vehicle A. To that end, let

$$X = \begin{bmatrix} x_r & v_{x,r} & a_{x,r} & y_r & v_{y,r} & a_{y,r} \end{bmatrix}^T \tag{4.1}$$

be the state vector of the target that needs to be estimated, where $[x_r, y_r]^T$, $[v_{x,r}, v_{y,r}]^T$, and $[a_{x,r}, a_{y,r}]^T$ are the position, velocity, and acceleration vectors in the inertial frame O.

For a typical automotive sensor capable of providing object-level data, for example, a radar or a camera, we can assume that it can measure the position and velocity of the target in the sensor's

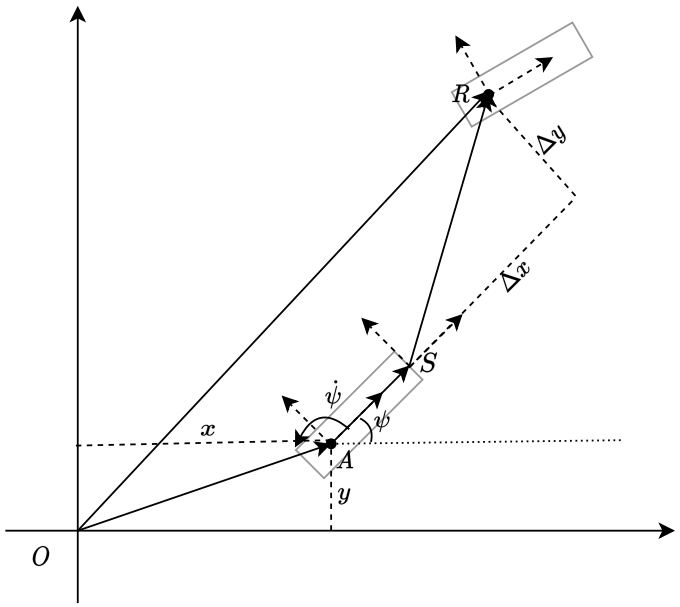


Figure 4.1: Schematic showing the transformation of reference point R from the sensor's frame to the inertial frame

relative coordinates S. To that end, let us define

$$Z \stackrel{\text{def}}{=} \begin{bmatrix} \Delta x & \Delta v_x & \Delta y & \Delta v_y \end{bmatrix}^T \tag{4.2}$$

as the measurement vector. Further, let

$$X_{ego} = \begin{bmatrix} x & y & v_x & v_y & a_x & a_y & \psi & \dot{\psi} \end{bmatrix}^T \tag{4.3}$$

be the ego vehicle states consisting of the position, velocity, acceleration, yaw, and yaw rate in the inertial frame O. The ego state estimation problem is not dealt with in this work and hence, all states are assumed to be known.

Now, the task at hand is to use Z and X_{ego} to estimate X. To that end, let us transform the measurement vector Z from sensor frame S to the inertial frame O. Figure 4.1 also illustrates this coordinate transformation. The measured position vector is given by

$$\begin{bmatrix} \tilde{x}_r \\ \tilde{y}_r \end{bmatrix} = \begin{bmatrix} x + l_s \cos\psi + \Delta x \cos\psi - \Delta y \sin\psi \\ y + l_s \sin\psi + \Delta x \sin\psi + \Delta y \cos\psi \end{bmatrix}, \tag{4.4}$$

where l_s is the longitudinal distance between the center of the rear axle of the ego vehicle and the sensor mount location. Note that it is assumed that the sensor is mounted on the longitudinal axis of the vehicle.

Taking the derivative of (4.4), the velocity vector can also be calculated

$$\begin{bmatrix} \tilde{v}_{x,r} \\ \tilde{v}_{y,r} \end{bmatrix} = \begin{bmatrix} v_x - \dot{\psi}\left(l_s \sin\psi + \Delta x \sin\psi + \Delta y \cos\psi\right) + \Delta v_x \cos\psi - \Delta v_y \sin\psi \\ v_y + \dot{\psi}\left(l_s \cos\psi + \Delta x \cos\psi - \Delta y \sin\psi\right) + \Delta v_x \sin\psi + \Delta v_y \cos\psi \end{bmatrix}. \tag{4.5}$$

Using (4.4) and (4.5), a transformed measurement vector can be defined as

$$\tilde{Z} \stackrel{\text{def}}{=} \begin{bmatrix} \tilde{x}_r & \tilde{v}_{x,r} & \tilde{y}_r & \tilde{v}_{y,r} \end{bmatrix}^T \tag{4.6}$$

which provides the position and velocity measurement of the preceding vehicle in the inertial frame.

The measurement values are noisy leading to inaccurate estimates which might be not useful for the path generation and thereby controllers. However, $\tilde{Z}$ can be used as the observation in a Kalman filtering setting in order to have an accurate estimation of the measured states and also estimate unobserved states like acceleration. This Kalman filter approach is briefly outlined next.

Kalman filtering

Bayesian filtering is used in many object-tracking applications [20]. In the context of object tracking, the Bayesian filter first predicts the states of the target object at the current time step using the estimated states at the previous time step and a motion model. Based on this predicted state, the measurement likelihood is calculated using a measurement model. This measurement likelihood is then compared with the actual measurement to give innovation which is eventually used to correct the predicted state. This process is repeated to estimate the target object states recursively.

The Kalman filter provides a closed-form solution of a Bayesian recursive filter assuming that the motion and measurement models are linear and the related noise is zero mean Gaussian [21]. Figure 4.2 shows the steps involved in a discrete-time Kalman filter. Interested readers are requested to refer to [20] for a detailed explanation and derivation of the expressions used at various steps. It is important to note that data association is one of the key aspects of Kalman filtering. Data association refers to the task of assigning which measurement is to be associated with which track. This step is crucial in a scenario with multiple targets, false positive detections, and missed detections. However, in this work, it is assumed that the sensor only provides data for the relevant target which is the preceding vehicle in our use case.

Now, taking Figure 4.2 as a reference, various aspects of the Kalman filter are presented below:

- **States**: $\hat{X}$ represents the estimated states of the preceding vehicle in this work, i.e.

$$\hat{X} = \begin{bmatrix} \hat{x}_r & \hat{v}_{x,r} & \hat{a}_{x,r} & \hat{y}_r & \hat{v}_{y,r} & \hat{a}_{y,r} \end{bmatrix}^T. \tag{4.7}$$

- **Motion model**: A (near)constant acceleration model [20] is used in this work, where the jerk (the derivative of acceleration) of the preceding vehicle is assumed as zero mean Gaussian noise. Thus, the linear motion model to predict the states $\hat{X}_{k|k-1}$ and corresponding error covariance $P_{k|k-1}$ at the current time step k using the estimated states and covariance at time $k-1$ is given by

$$\begin{cases} \hat{X}_{k|k-1} = \mathbf{F}\hat{X}_{k-1|k-1} \\ P_{k|k-1} = \mathbf{F}P_{k-1|k-1}\mathbf{F}^T + \mathbf{Q} \end{cases} \tag{4.8}$$

  which is characterized by the transition matrix $\mathbf{F}$ and the process noise matrix $\mathbf{Q}$ and they are

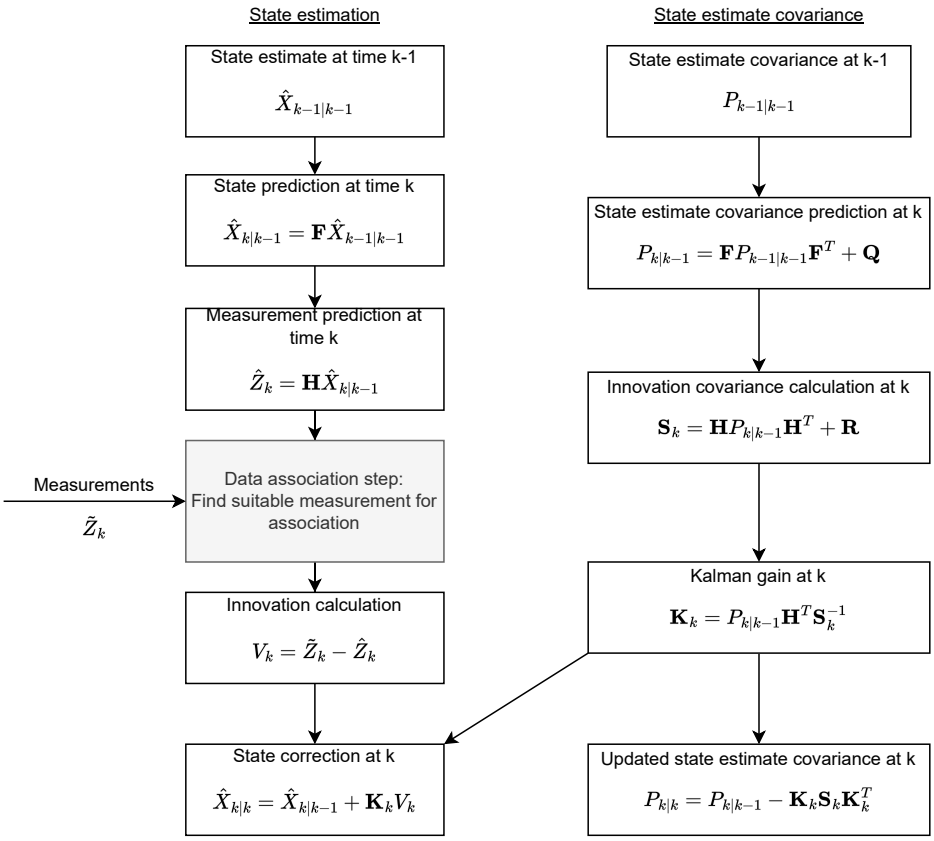


Figure 4.2: Flowchart of discrete Kalman filter recursive steps [20]

given by:

$$\mathbf{F} = \begin{bmatrix} 1 & T & T^2/2 & 0 & 0 & 0 \\ 0 & 1 & T & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & T & T^2/2 \\ 0 & 0 & 0 & 0 & 1 & T \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.9}$$

$$\mathbf{Q} = \sigma_Q^2 \begin{bmatrix} T^5/20 & T^4/8 & T^3/6 & 0 & 0 & 0 \\ T^4/8 & T^3/3 & T^2/2 & 0 & 0 & 0 \\ T^3/6 & T^2/2 & T & 0 & 0 & 0 \\ 0 & 0 & 0 & T^5/20 & T^4/8 & T^3/6 \\ 0 & 0 & 0 & T^4/8 & T^3/3 & T^2/2 \\ 0 & 0 & 0 & T^3/6 & T^2/2 & T \end{bmatrix}, \tag{4.10}$$

where T is the time difference between the current and the previous time step, i.e., $T = t_k - t_{k-1}$ and σ_Q is the standard deviation of the Gaussian noise assumed.

- **Measurement model**: The measurement model is used to calculate the likelihood of the measurement $\hat{Z}_k$ at k given the predicted state $\hat{X}_{k|k-1}$. It is given as

$$\hat{Z}_k = \mathbf{H}\hat{X}_{k|k-1}, \tag{4.11}$$

where $\mathbf{H}$ is the linear sensor model. For the measurement vector assumed in this work (4.6), it is given by

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \tag{4.12}$$

- **Innovation**: Innovation V_k represents the difference between the actual measurement received $\tilde{Z}_k$ and the measurement likelihood $\hat{Z}_k$ calculated above i.e.,

$$V_k = \tilde{Z}_k - \hat{Z}_k. \tag{4.13}$$

Furthermore, the covariance for the innovation $\mathbf{S}_k$ is given by

$$\mathbf{S}_k = \mathbf{H} P_{k|k-1} \mathbf{H}^T + \mathbf{R}, \tag{4.14}$$

where $\mathbf{R}$ is the sensor noise matrix. Writing this sensor noise matrix is not trivial. Even if we assume a zero mean Gaussian noise for the measurement vector Z in the sensor frame (4.2), the non-linear transformation to calculate $\tilde{Z}$ will result in non-gaussian noise. An unscented Kalman filter can be used to deal with this. However, in this work, a conservative approach is taken, where the noise is modeled as the zero mean Gaussian noise with sufficiently high covariance and is given as:

$$\mathbf{R} = \begin{bmatrix} \sigma_x^2 & 0 & 0 & 0 \\ 0 & \sigma_{v_x}^2 & 0 & 0 \\ 0 & 0 & \sigma_y^2 & 0 \\ 0 & 0 & 0 & \sigma_{v_y}^2 \end{bmatrix}, \tag{4.15}$$

where $\sigma_x^2, \sigma_{v_x}^2, \sigma_y^2, \sigma_{v_y}^2$ are the variance values for the position and velocity measurements. Note that the off-diagonal terms are zero. This is under the assumption that there is no correlation between the measured states.

- **Kalman update**: With the innovation V_k and its covariance $\mathbf{S}_k$ obtained above, the Kalman update step is performed to correct the predicted states and their corresponding error covariance $(\hat{X}_{k|k-1}, P_{k|k-1})$ to give the corrected estimates for the current time step $(\hat{X}_{k|k}, P_{k|k})$, i.e.,

$$\begin{cases} \hat{X}_{k|k} = \hat{X}_{k|k-1} + \mathbf{K}_k V_k \\ P_{k|k} = P_{k|k-1} - \mathbf{K}_k \mathbf{S}_k \mathbf{K}_k^T \end{cases}, \tag{4.16}$$

where $\mathbf{K}_k$ is the Kalman gain given by

$$\mathbf{K}_k = P_{k|k-1} \mathbf{H}^T \mathbf{S}_k^{-1}. \tag{4.17}$$

Using the aforementioned motion and measurement models, a Kalman filter based state estimation algorithm is implemented for both the radar and the camera sensors. With this formulation, each sensor recursively produces the state estimates $\hat{X}$ of the preceding vehicle along with the covariances P. This list of state estimates and covariances along with the time stamp forms the track of the preceding vehicle. Thus, a track can be represented as

$$\mathbf{T} = \left[\{\hat{X}_0, P_0\}, \{\hat{X}_1, P_1\}, ..., \{\hat{X}_k, P_k\}\right], \tag{4.18}$$

where k represents the time step. Next, a high-level fusion algorithm is discussed to perform T2Tf using the local tracks from different sensors.

4.2.2 Track to track fusion

Fusing two or more tracks from different sensors is a widely researched topic. [19] and [22] presents a review of different methods and architectures to perform this task. One of the simplest methods for fusion presented in [22] is the basic convex combination method. In this method, the local tracks from individual tracks are linearly combined as the weighted sum based on their covariances. This method, however, may produce suboptimal results as it ignores the cross-covariance between the tracks which might be present due to the common motion model and process noise involved in their estimates [22]. Furthermore, the implementation of this algorithm requires the tracks to be synchronized in time, which often is not the case in the actual sensors. One of the most popular T2Tf algorithms that is tractable, implementable, and produces near-optimal solutions is information matrix fusion (IMF).

The main advantage of IMF is that it does not require cross-covariance calculation, which makes its implementation tractable when compared to the exact versions of T2Tf fusion algorithms [19]. It also produces near-optimal solutions when operated at the full rate, i.e. performing fusion step at each local track update [19]. Furthermore, several versions of IMF are available in the literature catering to different implementation issues like asynchronous sensors, heterogeneous measurements, etc. Thus, IMF is selected as the T2Tf algorithm in this work.

As described in the previous section, both the sensors produce the state estimate of the preceding vehicle along with their covariances. Furthermore, these sensors operate at different frequencies. Therefore, the asynchronous, homogeneous version of IMF presented in [23] is implemented. Additionally, from an implementation point of view, it is required to have the fusion at a fixed frequency such that the path generation and thereby control algorithms can be implemented at a fixed rate in a discrete-time setting. To that end, a fusion center (FC) driven approach [23] is implemented where the fusion happens at fixed intervals.

The timeline for such an implementation is shown in Figure 4.3. The FC update steps (shown by the green squares) occur at fixed time intervals ν, i.e., if the current fusion update happens at t_f, then the previous update must have happened at $t_{f-1} = t_f - \nu$. The local tracker (LT) updates for the camera and radar are shown using red circles and blue triangles, respectively. Now, the idea is to use the *new information* obtained from each sensor in the fusion interval $(t_{f-1}, t_f]$ to update the fused states. The information state fusion expression and the corresponding covariance expression to do so is given by [23]

$$\begin{aligned} P\left(t_f \mid t_f\right)^{-1} \hat{X}\left(t_f \mid t_f\right) =& P\left(t_f \mid t_{f-1}\right)^{-1} \hat{X}\left(t_f \mid t_{f-1}\right) \\ &+ \sum_{i=1}^{N_s} \Big\{ P^i\left[t_f \mid t^i\left(t_f\right)\right]^{-1} \hat{X}^i\left[t_f \mid t^i\left(t_f\right)\right] \\ &- P^i\left[t_f \mid t^i\left(t_{f-1}\right)\right]^{-1} \hat{X}^i\left[t_f \mid t^i\left(t_{f-1}\right)\right] \Big\}, \end{aligned} \tag{4.19}$$

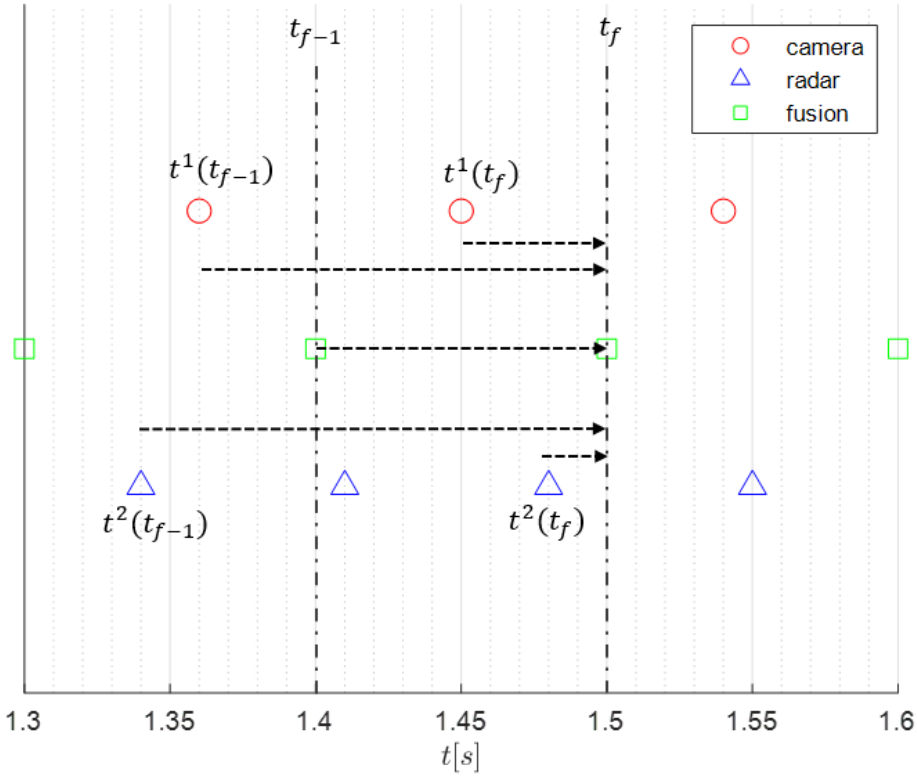


Figure 4.3: Fusion center driven asynchronous track to track fusion: timeline

$$P\left(t_f \mid t_f\right)^{-1} = P\left(t_f \mid t_{f-1}\right)^{-1} + \sum_{i=1}^{N_s} \left\{ P^i \left[t_f \mid t^i\left(t_f\right)\right]^{-1} - P^i \left[t_f \mid t^i\left(t_{f-1}\right)\right]^{-1} \right\}, \tag{4.20}$$

where N_s is the number of sensors, i represents the sensor index, $\hat{X}^i$ and P^i are the state estimate and related covariance from sensor i, and $\hat{X}$ and P are the fused state estimate and related covariance. Furthermore, $t^i\left(t_f\right)$ and $t^i\left(t_{f-1}\right)$ are the times of the most recent LT update of sensor i prior to t_f and t_{f-1} respectively. One can calculate $\hat{X}$ by using (4.20) in (4.19).

Next, we present a brief explanation of the terms presented in (4.19) and (4.20). The terms $\hat{X}\left(t_f \mid t_{f-1}\right)$ and $P\left(t_f \mid t_{f-1}\right)$ denote the FC predicted state and the related covariance obtained using the motion model (4.8) presented in the previous section, i.e.

$$\begin{cases} \hat{X}\left(t_f \mid t_{f-1}\right) = \mathbf{F}\hat{X}\left(t_{f-1} \mid t_{f-1}\right) \\ P\left(t_f \mid t_{f-1}\right) = \mathbf{F}P\left(t_{f-1} \mid t_{f-1}\right)\mathbf{F}^T + \mathbf{Q}. \end{cases} \tag{4.21}$$

Note that the time difference T required to calculate $\mathbf{F}$ and $\mathbf{Q}$ in the above expression is given by $T = t_f - t_{f-1}$. Further, the terms in the braces in (4.19) and (4.20) represent the *new information* received during the fusion interval $(t_{f-1}, t_f]$. A brief explanation for this notion of *new information* is presented herein.

Consider the terms in the braces in (4.19). In the first term, $\hat{X}^i\left[t_f \mid t^i\left(t_f\right)\right]$ represents the state

estimate from sensor i obtained at $t^i\left(t_f\right)$ propagated to the fusion time t_f. Similarly, in the second term, $\hat{X}^i\left[t_f \mid t^i\left(t_{f-1}\right)\right]$ represents the state estimate from sensor i obtained at $t^i\left(t_{f-1}\right)$ propagated to the fusion time t_f. The dotted arrows in Figure 4.3 represent these propagations. Furthermore, $P^i\left[t_f \mid t^i\left(t_f\right)\right]$ and $P^i\left[t_f \mid t^i\left(t_{f-1}\right)\right]$ represent the related covariance for these propagated states, respectively. Note that these propagations are carried out exactly the same way as it is done in (4.21) with appropriate values of T. Now combining these propagated states and their related covariance gives the latest 'information' [24] from sensor i prior to time t_f and t_{f-1}, i.e., the terms $P^i\left[t_f \mid t^i\left(t_f\right)\right]^{-1}\hat{X}^i\left[t_f \mid t^i\left(t_f\right)\right]$ and $P^i\left[t_f \mid t^i\left(t_{f-1}\right)\right]^{-1}\hat{X}^i\left[t_f \mid t^i\left(t_{f-1}\right)\right]$ represent the latest information we have from sensor i prior to t_f and t_{f-1} respectively. Thus, the difference between the two denotes the *new information* obtained from sensor i during the fusion interval $(t_{f-1}, t_f]$.

Thus, the overall expression in (4.19) and (4.20) represents a covariance-based weighted summation of the FC prediction and the new information obtained from the sensors in the fusion interval.

Note that this implementation is flexible and capable of handling different uncertainties. For example, if there is no update received from any sensor in the fusion interval, then $t^i\left(t_f\right) = t^i\left(t_{f-1}\right)$, resulting in cancellation of the terms in the braces and zero contribution from that sensor in the fusion update. Furthermore, the LT update does not need to be happening at a fixed rate. The only requirement is that the algorithm needs to have some memory to remember the state and covariances from different sensors and their respective time stamp. It should also be noted that if the fusion interval is too big, then it is possible to have multiple LT updates within the fusion interval. This can result in discarding some useful information as only the most recent LT updates prior to t_f and t_{f-1} are used. To avoid such information losses, the fusion interval should be kept as low as possible. However, computational load also needs to be considered for real-time implementation.

Next, a simulation study is provided to show the performance of the Kalman filter of individual sensors and the IMF based T2Tf fusion algorithm.

4.2.3 Results

To evaluate the performance of the state estimation algorithm described in the previous section, a simulation study is provided in this section. The MATLAB scenario generator [25] is used to create a scenario where the ego vehicle is following a preceding vehicle which is driving a circular path at a constant longitudinal speed. A camera and a radar sensor are attached to the ego vehicle to measure the relative position and velocity. These measurements are then transformed to the inertial frame. Next, these transformed measurements are used in the Kalman filter based local trackers of each of the individual sensors. Finally, the outputs of these local trackers are used in the IMF based T2Tf to produce the fused state estimates. Various parameters used in the simulation are listed in Table 4.1. The noise values for radar are based on the values provided in [26] whereas the noise values for the camera are taken based on the discussion with the stakeholder.

Figure 4.4 shows the estimated states of the preceding vehicle with camera, radar, and IMF fusion. It also shows the ground truth for reference. The estimates seem to follow the ground truth reasonably well. To quantify the performance of the local trackers and fusion, root mean square errors (RMSE) are calculated for position, velocity, and acceleration estimates. The initial phase of filter stabilization of about $2s$ is not included in the RMSE calculations. The estimation error values are listed in Table 4.2. Of the two sensors, the camera produces better position estimates while radar produces better velocity and acceleration estimates. This was expected based on the respective noise values listed in Table 4.1. Furthermore, IMF fusion produces the best estimates for all the states, even though it runs

Table 4.1: Sensor fusion parameters

Parameters	Symbol	Values
Radar tracker time interval	ν_{rad}	$0.07\ [s]$
Camera tracker time interval	ν_{cam}	$0.09\ [s]$
Fusion center time interval	ν_{fusion}	$0.10\ [s]$
Process noise standard deviation	σ_Q	$1\ [m/s^3]$
Camera position noise standard deviation	$\sigma_{x,cam} = \sigma_{y,cam}$	$0.2\ [m]$
Camera velocity noise standard deviation	$\sigma_{v_x,cam} = \sigma_{v_y,cam}$	$1\ [m/s]$
Radar position noise standard deviation	$\sigma_{x,rad} = \sigma_{y,rad}$	$0.5\ [m]$
Radar velocity noise standard deviation	$\sigma_{v_x,rad} = \sigma_{v_y,rad}$	$0.5\ [m/s]$

Table 4.2: State estimation RMSE errors

	Position $[m]$	Velocity$[m/s]$	Acceleration$[m/s^2]$
Camera	0.1567	0.4198	0.6516
Radar	0.1992	0.3032	0.5872
IMF fusion	**0.1133**	**0.2832**	**0.5666**

at a lower rate compared to the individual sensors. It can also be seen from the estimated acceleration plots in Figure 4.4 that the fusion estimates closely follow the radar-based estimates. This is expected since the camera-based acceleration estimates have a higher noise level. The position and velocity estimation errors are deemed acceptable for path generation and longitudinal control requirements. However, the acceleration estimates are still quite noisy. The acceleration values for buses in nominal conditions are in the range of $-1.4m/s^2$ to $1.4m/s^2$ [15] and having an RMSE of $0.6m/s^2$ in its estimates doesn't make it too useful. Thus, it is assumed that V2V communication provides the necessary acceleration values of the preceding vehicle. The acceleration estimates, however, can be used in fallback scenarios such as intermittent loss of communication. Such implementations are however beyond the scope of this work.

4.3 Summary

This chapter outlines the need for the preceding vehicle's state estimation based on the control algorithms developed in the previous chapters. Then, it describes the state estimation problem of the preceding vehicle using the onboard sensors. It also presents a high-level fusion algorithm that can be used to fuse the information coming from two or more sensors to provide a coherent estimate with better accuracy. A simulation study is also presented to show the performance of the fusion algorithm.

With the state estimation algorithm implemented, the whole pipeline for lateral and longitudinal control of the follower vehicles is complete. Now, the Siemens Prescan environment is used to verify the algorithms developed in this work.

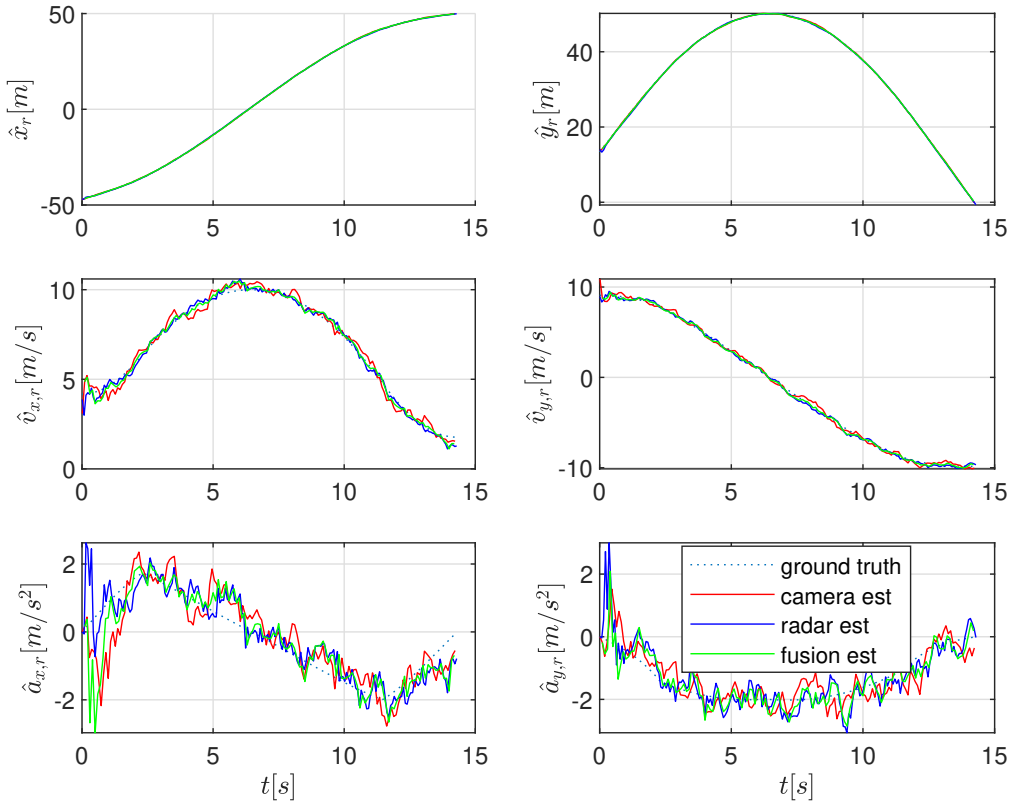


Figure 4.4: State estimation results: Camera vs Radar vs IMF

5 Experimental analysis

This chapter presents the experimental analysis to verify the performance of different algorithms presented in the previous chapters in different scenarios. The experimental analysis also provides insight into the potential limitations of the current implementation of the algorithms. Siemens Prescan is used as a simulation platform to design scenarios, set up sensors, and integrate all the algorithms. First, a short introduction to the Prescan tool is provided in section 5.1. Next, the scenarios considered in this simulation study and their motivation are described in section 5.2. Finally, the simulation results are presented in section 5.3.

5.1 Simulation environment: Prescan

Siemens Prescan is a physics-based simulation platform that can be used to develop and validate advanced driver assistance systems (ADAS) and automated vehicle functionalities. Prescan provides a streamlined methodology to verify different algorithms in a closed-loop simulation environment. This methodology consists of four steps:

- Replicate or import real-world traffic scenarios using various inbuilt elements within the Prescan library using a GUI or API. Different aspects like weather conditions, local traffic signs, road imperfections, etc. can also be modeled.
- Add sensors to the vehicles. Prescan provides a number of sensors in its library that can be used for understanding the environment around the ego vehicle. It also provides V2X communication which is crucial for the simulation of the CCAM system.
- Use the control system interface to implement control algorithms. Prescan offers integration with MATLAB/Simulink, C++, and Python platforms. These platforms can be used to develop fusion and control algorithms and visualize results.
- Run experiments once scenarios are built and algorithms are implemented using the control system interfaces. Prescan also provides a test automation feature that can help in running the simulation in various configurations of parameters automatically and log results. This can be useful for tuning various algorithms.

The above methodology is followed in this work as well. First, different scenarios are built using the Prescan GUI. The default Hino blue ribbon bus available in the Prescan library is used as the vehicle in these simulations. Unfortunately, the dynamic model is not present for the bus in the Prescan library. Therefore, a vehicle model described in section 2.1 is implemented using the vehicle parameters listed in appendix B. For sensors, Prescan provides very detailed sensor models. However, they do

Table 5.1: Time interval for different algorithms in the Simulink model

Parameters	Values
Overall simulation	$0.01\ [s]$
Radar tracker	$0.05\ [s]$
Camera tracker	$0.10\ [s]$
IMF fusion	$0.10\ [s]$
Path generation	$0.10\ [s]$
Longitudinal control	$0.01\ [s]$
Lateral control	$0.01\ [s]$

not directly provide the object-level data, which is assumed to be available in this work. To that end, the Actor Information Receiver sensor of Prescan is used, which provides the ground truth information of the actors/vehicles. A zero mean Gaussian noise is added to simulate the noisy measurements from the sensors. Thus, the result of the sensor fusion algorithm for all the scenarios is similar to what we obtained in chapter 4. Therefore, fusion-related results are not discussed in this chapter. Furthermore, V2X transceivers use the ETSI CAM [27] message type for V2V communication in the simulations presented in this chapter.

Once the scenarios are built and sensors are configured, a Simulink model is generated from the built file. All algorithms such as IMF based T2Tf, path generation, and control algorithms are then implemented in this Simulink model. Table 5.1 lists the rate at which different blocks run in the Simulink experiment.

5.2 Scenario description

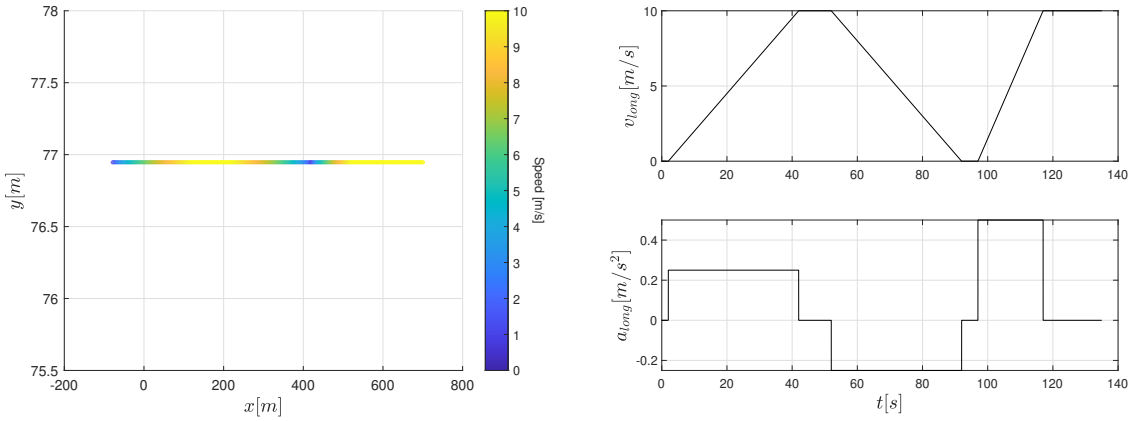
As discussed in section 1.4.2, a three bus platoon use case is considered in this work. The first bus or the leader of the platoon is driven by an experienced human driver and thus responsible for taking a safe path, whereas the second and the third buses are the automated follower bus. To mimic the human-driven lead bus (Bus 0), a reference trajectory is provided in the simulation which it follows perfectly. The follower buses (Bus 1 and 2) are equipped with sensors, path generation, and control algorithms to perform the path-following task.

Driving is a complex task and many scenarios are possible in real-world driving. Even though it is not possible to simulate all scenarios, one can define several scenarios that represent essential maneuvers required for driving a city bus. To that end, three scenarios are considered in this simulation study. The scenarios are characterized by the trajectory followed by the lead bus.

5.2.1 Start-stop-start

This scenario depicts the typical stop-go drive cycles that the city buses go through. Figure 5.1 shows the $x - y$ position plot along with the velocity profile followed by the lead vehicle. The vehicle is at rest initially. It waits for $2s$, then starts to accelerate to reach a velocity of $10m/s$. It decelerates to come to a stop again for $5s$. Then it starts to accelerate again.

This scenario intends to verify the performance of the path generation algorithm in different phases of the stop-go cycle. Different phases include the initialization phase, constant velocity phase, and



(a) $x - y$ plot

(b) Longitudinal velocity and acceleration profiles

Figure 5.1: Start-stop-start scenario

transition phase from rest to motion, or motion to rest. Furthermore, the tracking performance of the longitudinal controller is also verified in this scenario with the changing acceleration profile of the preceding vehicle. This scenario is also used for verifying the stability of the lateral control on a straight path.

5.2.2 Roundabout

In this scenario, the lead bus takes a $270\,\mathrm{deg}$ roundabout turn. Figure 5.2 shows the $x - y$ plot for this scenario with the velocity profile followed by the lead vehicle. Similar to the previous scenario, it is at rest initially. Then, it accelerates to attain a constant velocity. As it approaches the corner, it reduces the velocity to make the roundabout at a constant speed of $6m/s$. Once it exits the roundabout, it again accelerates.

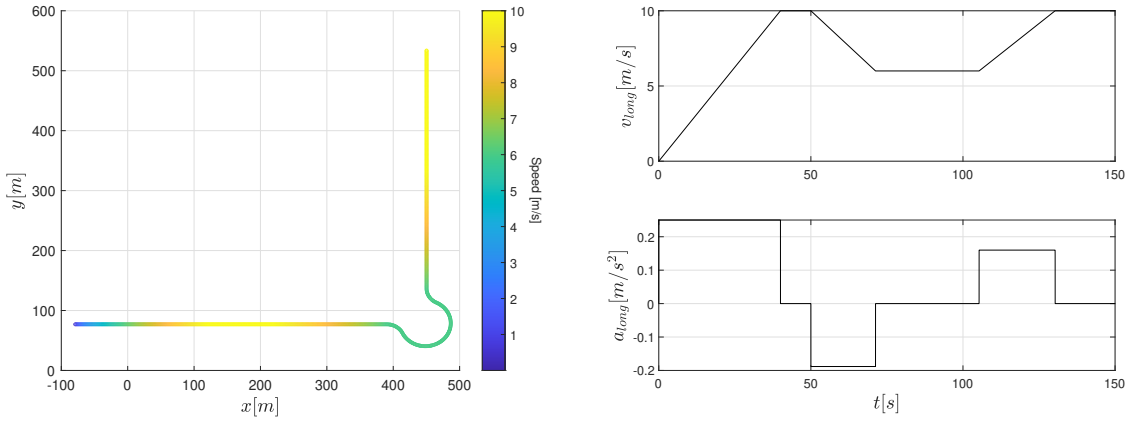
The goal of this scenario is to evaluate how well the lateral controller performs. Furthermore, the performance of the longitudinal controller is verified when the vehicle is not on a straight path and curvature-based limitations have to be imposed on the velocity of the ego vehicles such that the lateral acceleration is limited as explained in section 2.3.1.

5.2.3 Evasive maneuver

This scenario depicts an evasive action taken by the lead vehicle. Figure 5.3 schematically shows the scenario, wherein the road is partially blocked and the vehicles need to take an evasive action to avoid the obstacle.

This simulation aims to study the response from the platoon when the lead vehicle takes such an evasive action, especially in the distance domain which is critical for collision avoidance.

Next, the simulation results for these three scenarios are discussed.



(a) $x - y$ plot

(b) Longitudinal velocity and acceleration profiles

Figure 5.2: Roundabout scenario

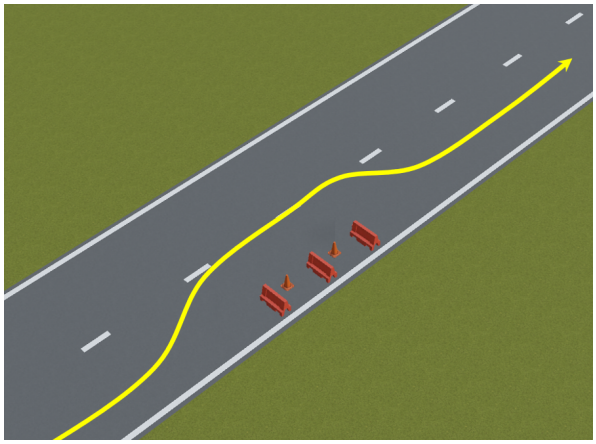


Figure 5.3: Evasive maneuver scenario schematic: The evasive path to be taken by the vehicle is shown by the yellow arrow

5.3 Simulation results and discussion

This section presents the simulation results obtained for the three scenarios described in the previous section. The simulation parameters for the vehicle model, path generation, sensor fusion, and longitudinal and lateral control algorithms are kept the same as described in their respective chapters. For clarity, the lead vehicle, the first follower, and the second follower are referred to as Bus 0, Bus 1, and Bus 2, respectively. Furthermore, they are represented with black, blue, and red colors in the plots.

5.3.1 Start-stop-start

Figure 5.4 shows the $x-y$ plot of the three buses. Both the followers traverse the leader's path closely with the maximum lateral deviation of $\pm 0.25m$. It can also be seen from Figure 5.4 and 5.1(a) that the lateral deviation is higher in the low-speed transient phases, namely, the initialization phase and

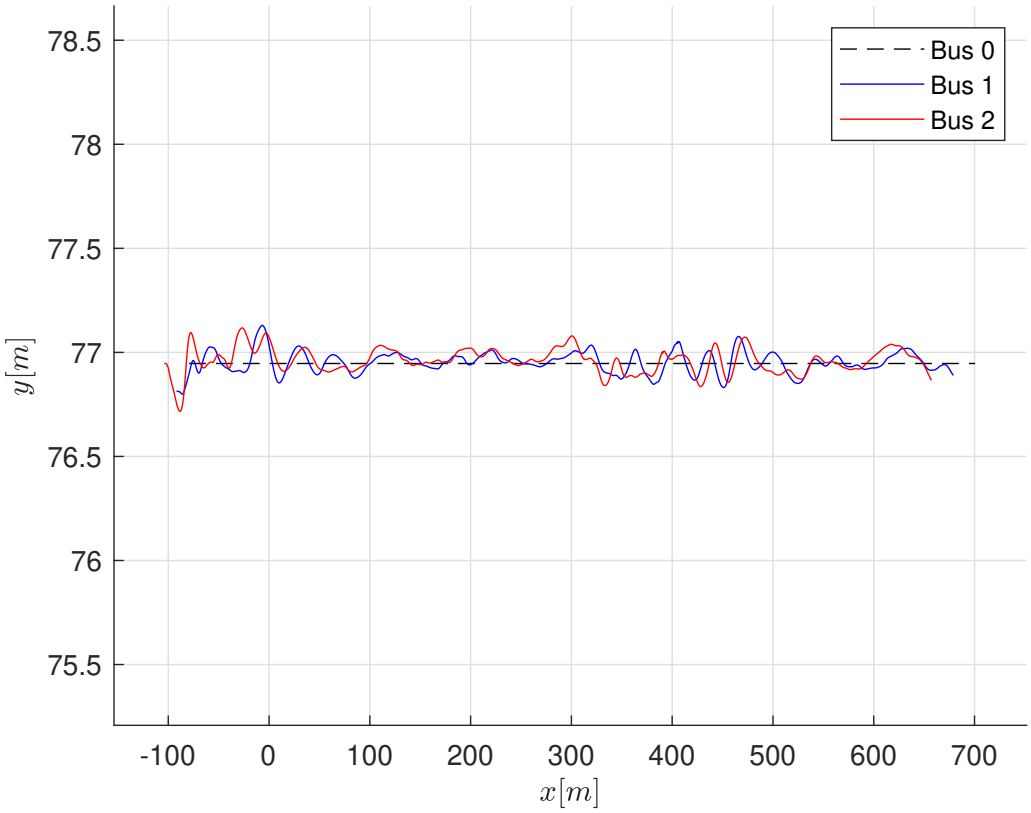


Figure 5.4: Start-stop-start: $x - y$ plot

the motion transition phase from motion to rest and rest to motion.

This observation may be attributed to the path generation inaccuracies in the low-speed transient phases. Figure 5.5 shows the approximated spline path generated by the path generation algorithm for Bus 1 in the initialization phase. It can be seen that the waypoints used for spline fitting are not equidistant. The first three waypoints are within the space of $12m$ while the next three are within $1m$. Furthermore, the effect of measurement/estimation noise becomes more prominent at lower distances. These factors may result in an inappropriate approximated spline with high curvatures as shown in Figure 5.5. A similar argument can be used to explain the path generation inaccuracy in the motion transition phase as well. Consequently, it can be concluded that there should be a speed dependency on the waypoint management and thereby path generation algorithm which is not included in the current implementation.

Next, the longitudinal tracking performance of the follower vehicles is presented. Figure 5.6 shows the longitudinal controller response for this scenario. The topmost subplot shows the spacing error (2.26) while the middle and the bottom plots show the longitudinal velocity and acceleration profiles followed by the three buses. Both follower buses follow the acceleration profile of the leader vehicle reasonably well, except for the initialization phase, where both vehicles accelerate rather aggressively. This is due to the initial buffering period that the path generation algorithm needs to have enough waypoints to generate a path. Without a path in this buffering period, the spacing error could

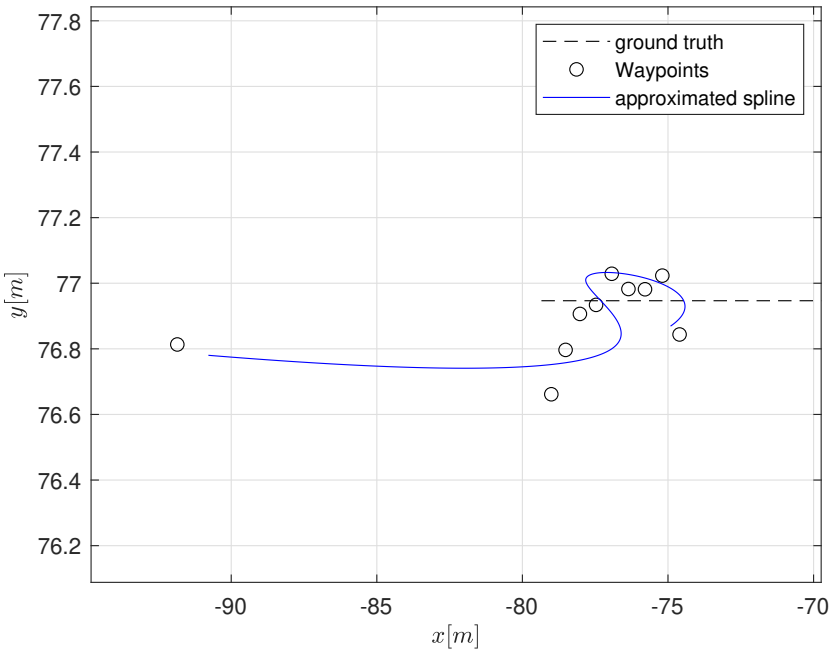


Figure 5.5: Start-stop-start: path initialization

not be calculated. This causes an initial delay in the follower's response, leading to a high spacing error, and thereby high acceleration to reduce the spacing error. One can use the radar range values directly to calculate the spacing error in the buffering period to avoid such a situation. Further, note that the acceleration value is saturated to $1.4m/s^2$ for Bus 2 (around $t = 12s$) as per the threshold value prescribed in (2.6) to meet the passenger comfort requirements.

Furthermore, the spacing error value seems to be quite noisy. A zoomed-in spacing error plot is shown in Figure 5.7. It looks like a saw tooth profile. This is due to the difference in the execution rates of the path generation and the control blocks. Referring to Table 5.1, the path generation provides an updated path every $0.1s$ whereas the longitudinal controller calculates the desired acceleration input every $0.01s$. Furthermore, zero-hold is used for the path in the longitudinal controller. This causes the controller to believe that the preceding vehicle is not moving in between the path generation updates and hence the error is reduced causing it to decelerate. However, the moment a new path update comes, the spacing error increases, prompting it to accelerate. This causes a low amplitude cyclic acceleration-deceleration cycle which can be seen in the acceleration plots in Figure 5.6.

Another interesting observation occurs when Bus 0 comes to a halt for $5s$. Even though the follower buses decelerate, they do not come to a halt completely. This can also be attributed to inaccuracy in the spacing error calculations leading to small acceleration inputs on the vehicle. However, if Bus 0 waits long enough, both follower buses come to a complete halt. This is shown in Figure D.1 in appendix D where Bus 0 remains stationary for $15s$ instead of $5s$.

Finally, the response of the lateral controller is presented for this scenario. Figure 5.8 shows the lateral controller response. The top and the middle subplots show the lateral and heading errors respectively while the bottom subplot shows the steering response of the follower buses. Note that the lateral and heading errors are computed with respect to the approximated spline generated by the path generation

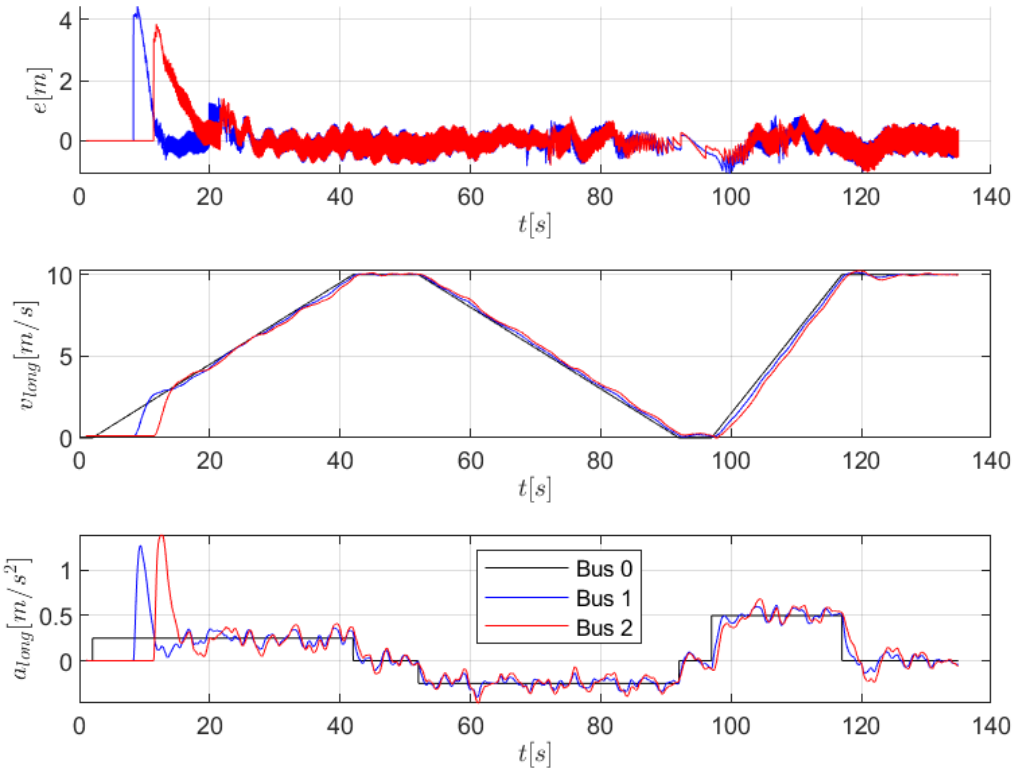


Figure 5.6: Start-stop-start: longitudinal control response

algorithm and not with respect to the actual ground truth. It can be observed that both the lateral and heading errors, and thereby steering values remain close to zero for the entire scenario except for the low-speed initialization and the motion transition phases. There are two possible reasons for this observation. Firstly, the spline approximation inaccuracy in the low-speed regions as discussed above. Secondly, the aggressive response from the lateral controller when the speed of the vehicle approaches zero, see (2.23). It can also be seen that the steering value is set to zero when velocity reaches below a certain threshold. In this work, the threshold value is $0.1m/s$. Furthermore, the lateral error seems to be following a cyclic pattern of a very low amplitude. This observation is also attributed to the spline approximation. The current path generation implementation does not take the temporal consistency [4] into consideration, i.e. no provision is provided to ensure that the part of the path remains unchanged for the overlapping time horizon between the two consecutive path updates. This results in sudden changes in the shape of the splines. Figure 5.9 shows the approximated splines for Bus 1 and its position with respect to these splines for five consecutive time-steps at an interval of $0.5s$ each when Bus 1 is in (almost)constant speed phase. It can be seen that the position of the bus is changing with respect to the spline. In the first two plots, the closest point from the bus position to the spline is on the left side of the bus, considering the bus is moving in the positive $x-$ axis direction. In the next two plots, the closest point is on the right side, resulting in a change in the sign of the lateral error.

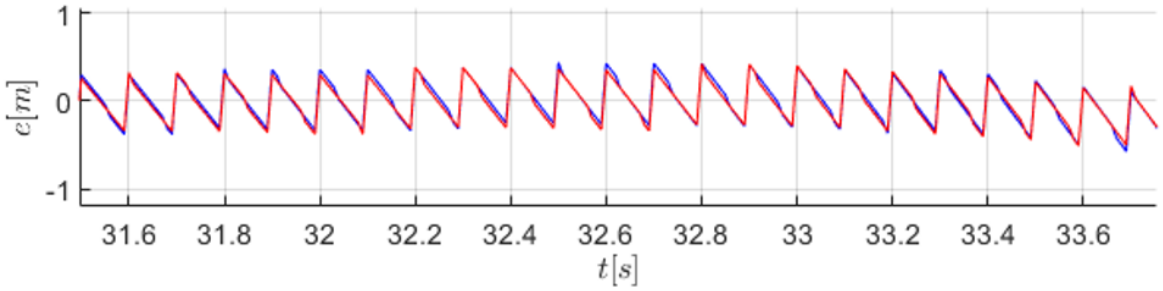


Figure 5.7: Start-stop-start: zoomed-in spacing error

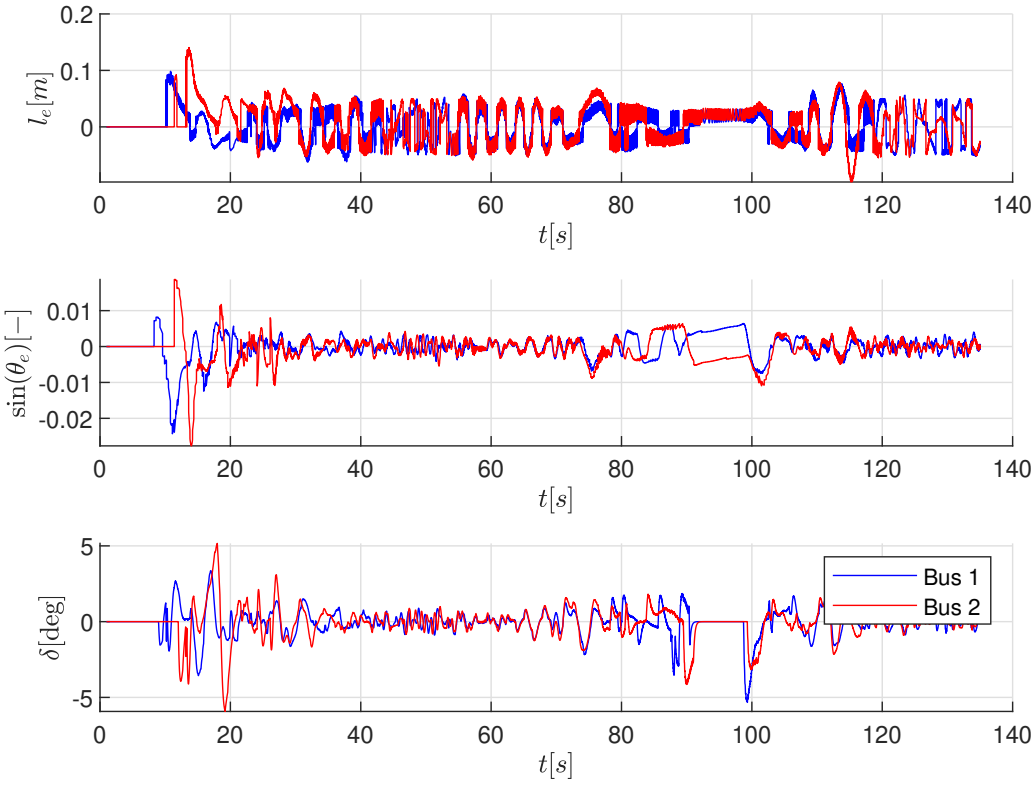


Figure 5.8: Start-stop-start: lateral controller response

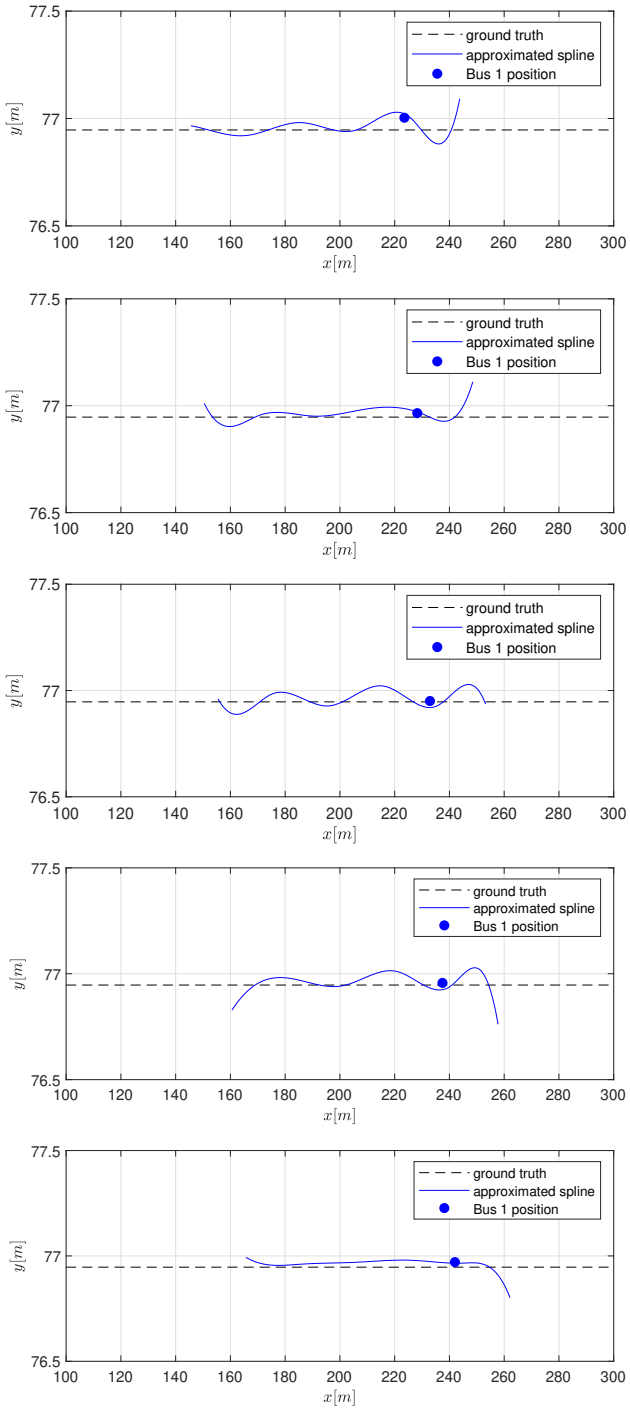


Figure 5.9: Start-stop-start: spline approximation at $t = 54.5s$, $t = 55.0s$, $t = 55.5s$, $t = 56.0s$,$t = 56.5s$ respectively in the order of top to down

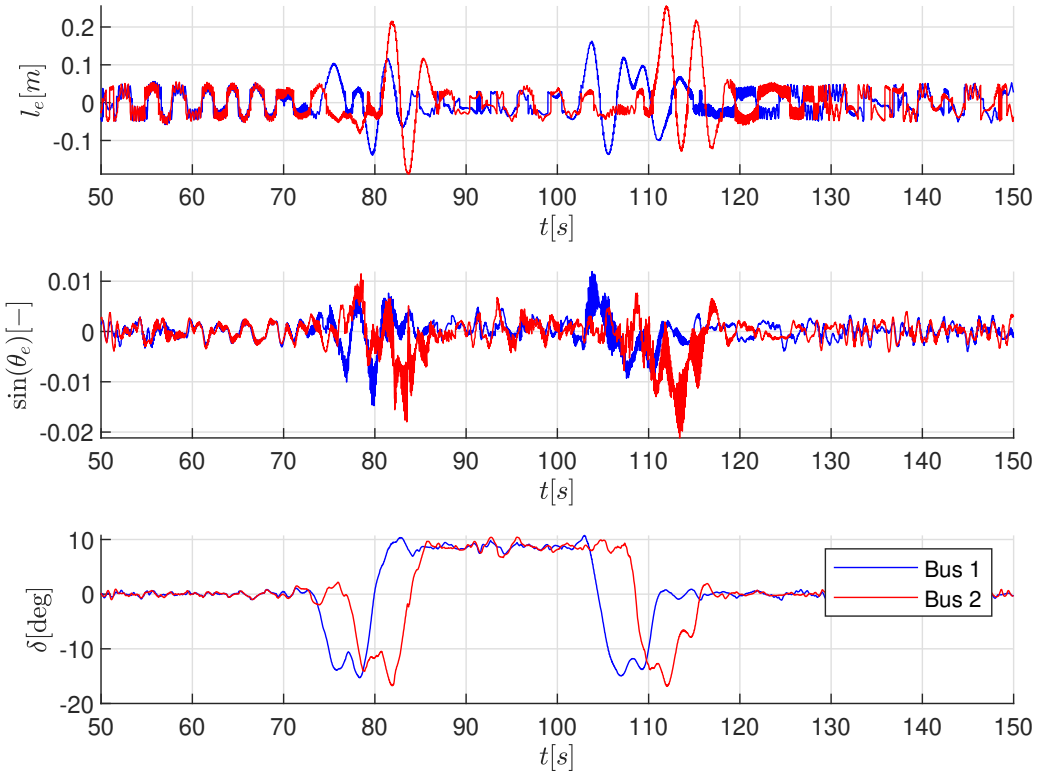
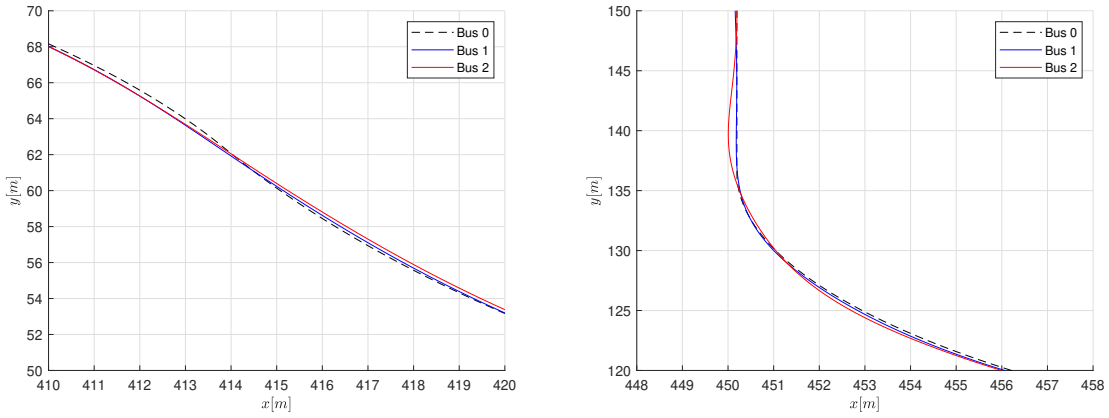


Figure 5.10: Roundabout: lateral controller response

5.3.2 Roundabout

As discussed in the previous section, the goal of this scenario is to verify the performance of the lateral and longitudinal controller in cornering. Thus, only the cornering part of the roundabout scenario is discussed here. Figure 5.10 shows the lateral controller response for this scenario. Similar to the straight path response, both lateral and heading errors are also close to zero in the constant radius turn phase of the roundabout, i.e., between times $90-100s$. The errors are however relatively higher at the entry and exit of the corners of the roundabout. This observation is in line with section 2.2.3. Thus, the same reasoning of the underlying steering dynamics of the system, i.e., the phase lag between the desired and the actual steering input, explains this observation.

The entry and exit of the corners of the roundabout seem to be the critical phases in this scenario. The lateral deviation of the follower buses with respect to the ground truth (path traversed by Bus 0) at the entry and exit of the roundabout is shown in Figure 5.11. Note that calculating the exact values for the lateral and heading errors with respect to the ground truth is not trivial as the simulation is done in the time domain whereas these errors are calculated in the distance domain, i.e., with respect to the closest point on the ground truth path from the ego vehicle. Thus, the analysis of the lateral deviation of follower buses with respect to the ground truth is done by visually looking at the $x-y$ plots. From Figure 5.11, it can be observed that the lateral deviation of Bus 2 is higher than Bus 1 with respect to the ground truth, which is expected since the reference for Bus 2 is the path



(a) Entry: driving from left to right

(b) Exit: driving from right to left

Figure 5.11: Roundabout: corner cutting and overshooting

traversed by Bus 1 which already has some deviation with respect to the ground truth. It can also be observed in Figure 5.11(b) that the deviation for Bus 2 is large at the exit of the roundabout corner, near $(450, 140)$, even though Bus 1 follows the ground truth relatively well. The reason for this observation is the spline approximation method. Figure 5.12 shows the approximated spline for Bus 2 at $t = 114s$. It can be seen that the approximated spline does not follow the waypoints closely. This observation indicates that along with the steering dynamics, spline approximation is the root cause of the high lateral deviation of the follower buses in the corners. The spline approximation method may be improved with tuning. For example, by increasing the number of spline pieces, one can obtain a spline that follows the waypoints more closely. However, that can result in spline approximations that follow the measurement/estimation noise pattern, which is also not desirable. Thus, a trade-off has to be made in the current implementation.

Furthermore, the longitudinal controller's response for this scenario is shown in Figure 5.13. It can be seen that the spacing error grows in the corner entry phase for both the followers ($75s - 80s$ for Bus 1 and $80s - 85s$ for Bus 2). This is due to the curvature-based cruise controller (CBCC) (2.31) implemented in this work to limit the lateral acceleration for passenger comfort. The radius of the path at the corner entry is $25m$. Thus, the maximum permissible speed of the vehicle based on (2.30) would be $5m/s$ which the follower buses try to attain. However, the lead bus continues to move at a speed of $6m/s$ through the entire roundabout, resulting in an increase in the spacing error. In the steady state cornering phase, the radius of the curve is $36m$, i.e. maximum permissible speed of the vehicle is $6m/s$. Thus, the spacing error remains almost constant as the follower buses maintain the same speed as the lead bus. At the exit of the roundabout, a similar situation arises as the entry of the roundabout, resulting in a further increase in the spacing error. Furthermore, the lead bus accelerates after exiting the roundabout which further increases the spacing error. However, once the follower buses are out of the corner, they accelerate to reduce the spacing error and eventually close to zero.

Note that the spacing error for Bus 2 reduces in the roundabout exit phase ($105s - 115s$). This is due to the fact that its preceding vehicle (Bus 1) decelerates to comply with CBCC. Furthermore, the spacing error between Bus 1 and 2 increases significantly when Bus 1 exits the corner when compared to the spacing error increase between Bus 1 and 0. This is due to the fact that Bus 1 accelerates significantly faster than Bus 0 at the exit of the roundabout. Note that the longitudinal acceleration is limited as

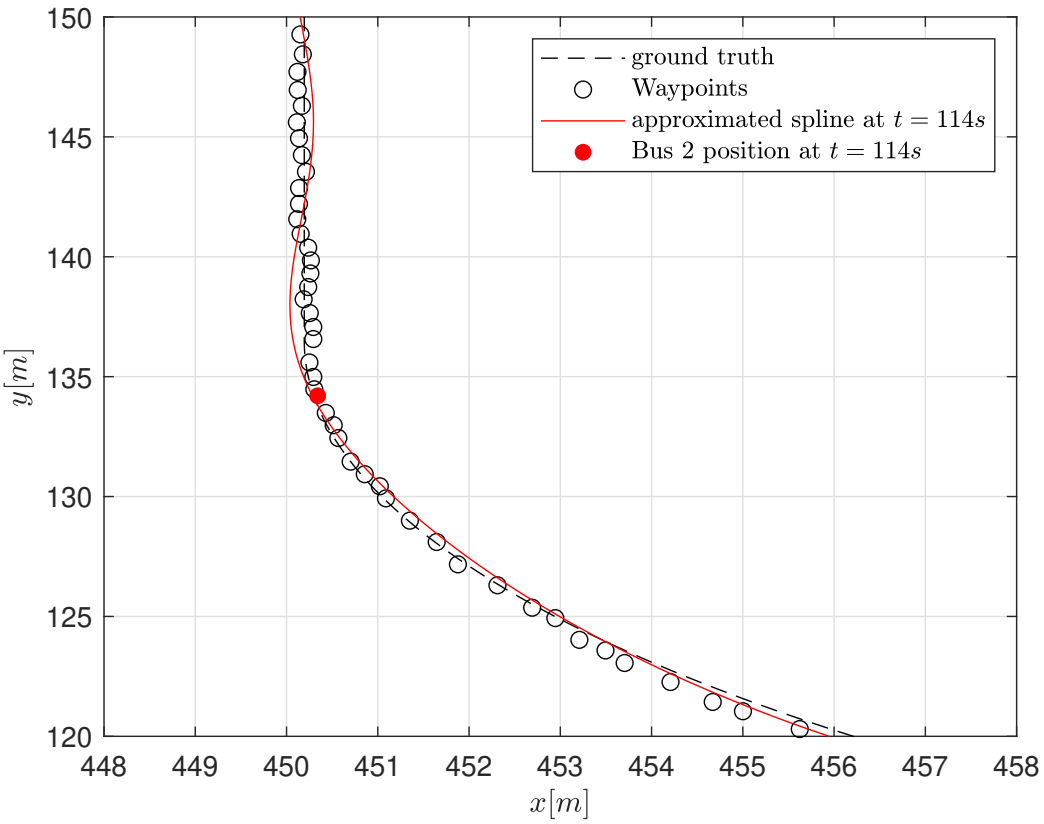


Figure 5.12: Roundabout: approximated spline for Bus 2 at $t = 114s$

per (2.6) to meet the passenger comfort requirements.

This simulation illustrates that the CACC- and CBCC-based longitudinal controller implemented in this work may have contradicting goals, especially in the corners. The CACC component tries to maintain a close but safe inter-vehicle gap whereas the CBCC component tries to limit the speed of the vehicle to respect the lateral acceleration threshold values for passenger comfort. This can result in a high spacing error as observed above. In an extreme case, the spacing error might become so large that the preceding vehicle goes beyond the measurement range of the sensors, thus losing the track. One may argue that if the lead vehicle maintains the velocity profile to respect the curvature based speed threshold for passenger comfort, this situation can be avoided. Another solution could be to formulate an objective function for the desired acceleration combining both objectives, i.e., close following and passenger comfort into account such that when spacing error becomes too high, passenger comfort is sacrificed momentarily to avoid losing track of the preceding vehicle completely.

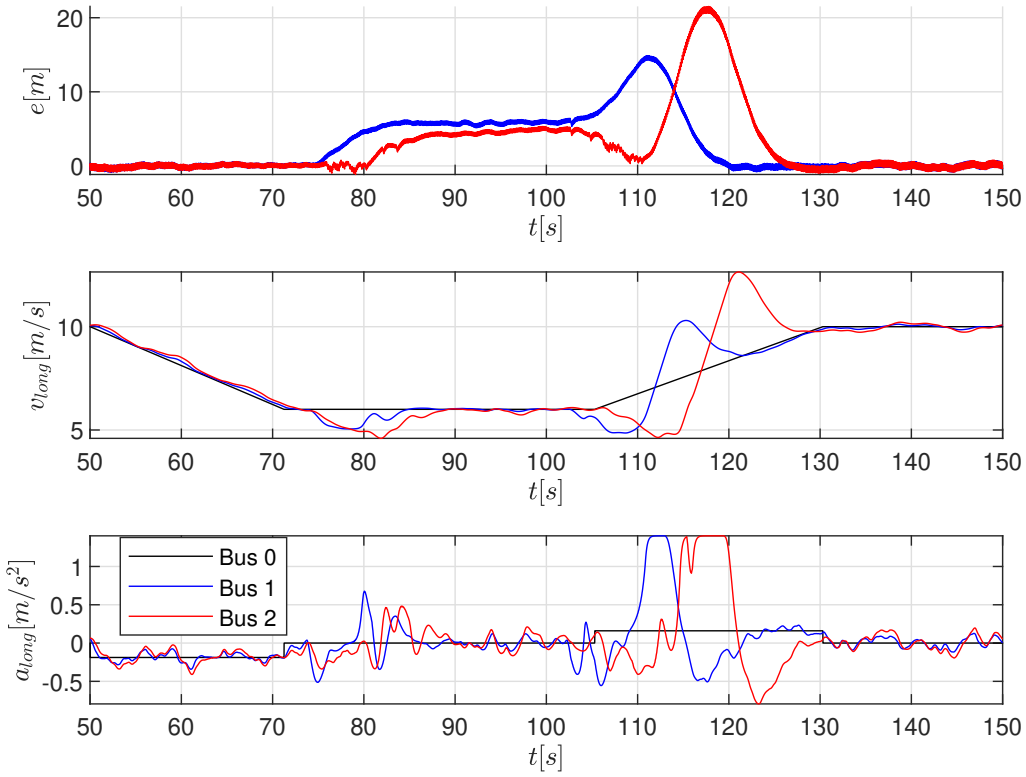


Figure 5.13: Roundabout: longitudinal control errors

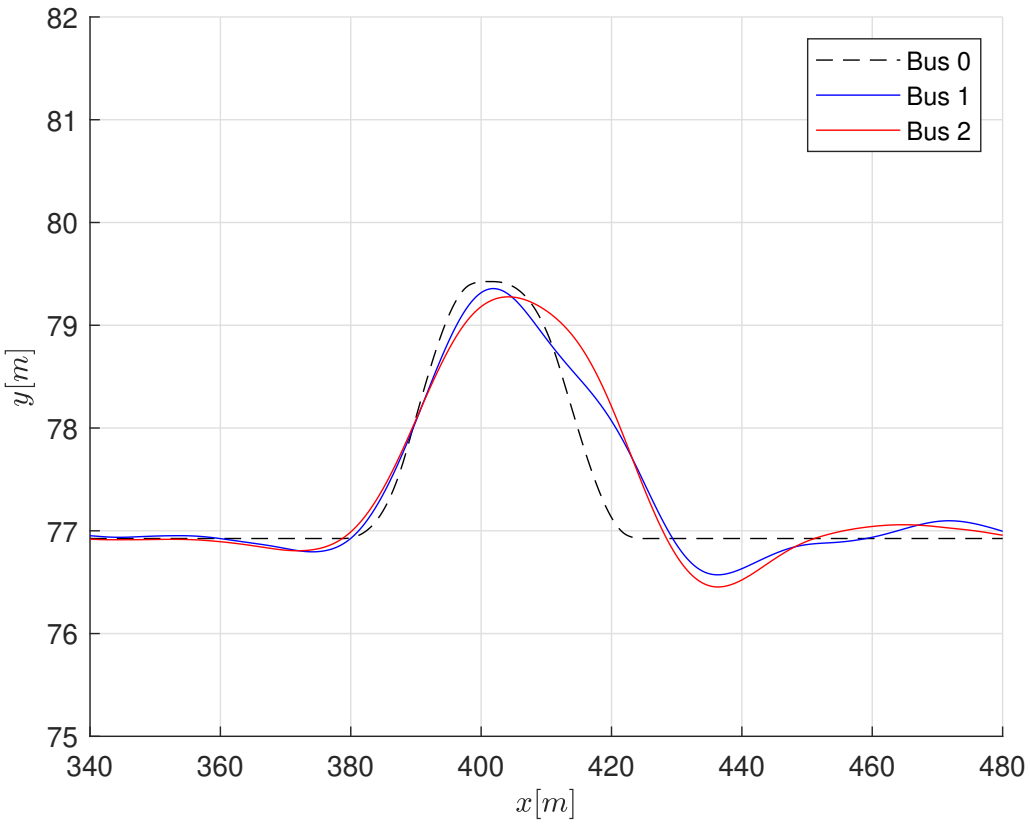


Figure 5.14: Evasive maneuver: distance domain following

5.3.3 Evasive maneuver

The $x - y$ plot for the evasive maneuver scenario is shown in Figure 5.14. Both the follower buses take the evasive path following the lead bus. However, the follower buses seem to cut the corner when approaching the peak of the lateral deviation (near $(400, 79.5)$) and then overshoot as they come out of it. This is mainly due to spline approximation in the path generation method. Figure 5.15 shows the approximated spline for Bus 1 at $t = 50s$ when it is close to the crest of the evasive maneuver. It can be seen that the approximated spline does not follow the waypoints closely. The spline approximation could not handle the abrupt change in direction and tried to fit a smoother spline. By increasing the number of splines, a better spline approximation may be obtained. However, as explained in the previous section, it might result in a spline approximation that follows the measurement/estimation noise pattern. Thus, if the lead bus provides enough margin to avoid the obstacle, the response from the follower buses can be deemed satisfactory with the current implementation.

Furthermore, it can be observed that the follower buses steer in the opposite direction initially at the start of the evasive maneuver, i.e., the region around $(370, 77)$ in Figure 5.14. This observation can also be explained based on the spline approximation shown in Figure 5.15 in that region.

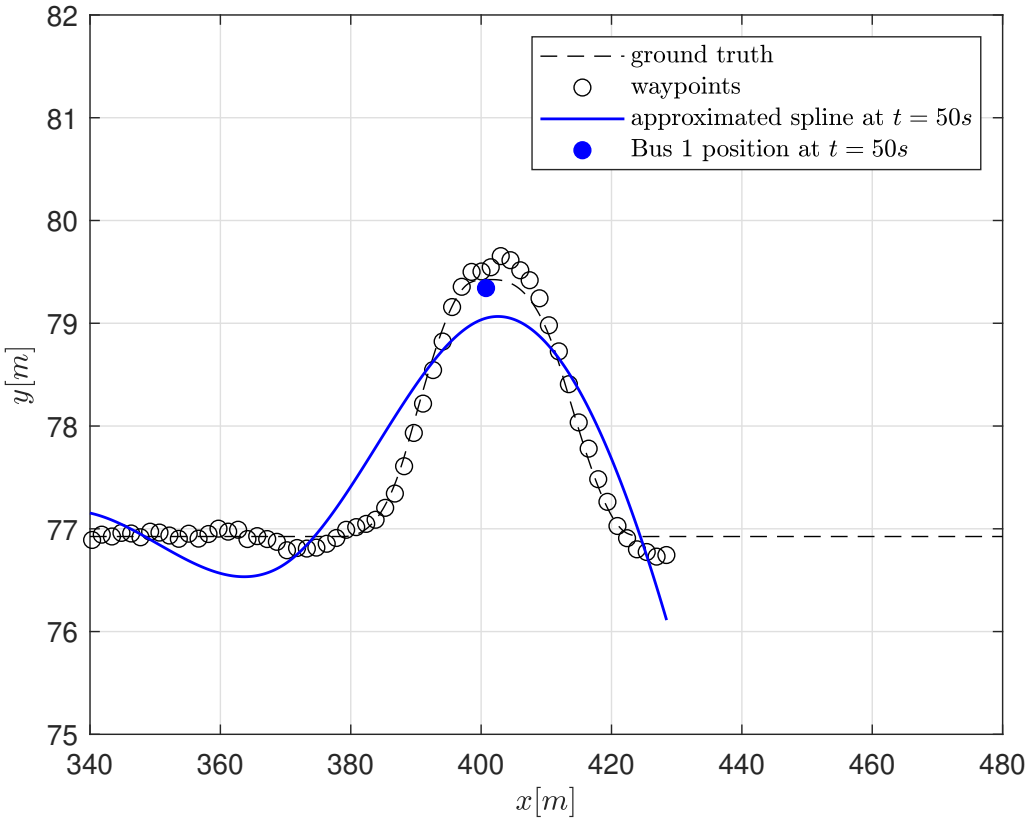


Figure 5.15: Evasive maneuver: approximated splines for Bus 1

5.4 Summary

In this chapter, the path following response of the follower buses is analyzed using the Simens Prescan simulation platform. The behavior of the follower buses seems reasonably acceptable in nominal conditions. Some interesting observations are noted mainly related to the spline approximation method of the path generation module in low speed regions and evasive maneuvers which require improvement. Furthermore, the performance of the controllers is also deemed satisfactory. The controllers are capable of not only following the preceding vehicles closely but also considering passenger comfort in their maneuvers as shown in the roundabout scenario.

6 Conclusions & Recommendations

This chapter summarizes the main conclusions drawn from this work and provides recommendations for future work.

6.1 Conclusions

This project aimed to use a control-oriented development approach for the development of sensor fusion algorithms for follower buses in a city bus platooning use case. To that end, first the lateral and longitudinal control algorithms are developed. Then, a high-level T2Tf algorithm is developed to estimate the necessary state information required by the controllers.

For the longitudinal controller, a combination of CACC- and CBCC-based control laws is used. The performance of the longitudinal controller is found to be satisfactory as it maintains a close but safe gap between the vehicles. Furthermore, it respects the passenger comfort parameters namely, permissible longitudinal and lateral acceleration threshold values. Notably, the follower vehicles respected these thresholds despite the lead vehicle violating them, as shown in the roundabout scenario presented in chapter 5.

For the lateral controller, a non-linear lateral velocity controller is used. The stability analysis for the same is also presented in this work. The lateral controller performs reasonably well in terms of tracking. Moreover, thanks to its formulation, it ensures passenger comfort by controlling the lateral velocity at which the vehicle approaches the reference path. However, there are some limitations of this controller. It is quite aggressive in the steering response at low vehicle speeds. Thus, a different approach might be required at low speeds.

Both controllers require a reference path to follow. The lateral controller requires a desired path, whereas the longitudinal controller requires the curvilinear distance over that path for spacing error calculation. To that end, a piecewise cubic spline approximation method is implemented for generating a reference path from the position estimates (waypoints) of the preceding vehicle. It performed reasonably well in most of the scenarios presented in chapter 5. However, improvements are also identified during experimental analysis. Further, it is noted that the performance of the path generation method proved to be the limiting factor in most of the observations.

To successfully implement the path generation and control algorithms presented in this work, the position, velocity, and acceleration states of the preceding vehicle are required in an inertial frame. Hence, a high-level T2Tf algorithm, called Inverse Matrix Fusion (IMF), is used. As expected, the fusion estimates are consistent and more accurate than the individual sensors. The formulation of the IMF algorithm is found to be versatile as it can handle asynchronous sensors, variable data rates from the individual sensors, etc. It is also capable of handling heterogeneous measurements. This is

useful for future developments when other sensors are used with different measurement states from one another.

Siemens Prescan is used in this work as the simulation tool to verify all developed algorithms. The simulation environment provides a vast library of sensors, vehicles, infrastructure, etc., and an interface with other development platforms, which is very useful for ADAS and CCAM system verification. However, the platform is restrictive in terms of dynamic modeling of the vehicles, and producing object-level data from the sensors.

6.2 Recommendations for future Work

There are some areas of improvement identified during this work that can be taken up in the future to implement bus platooning in practice.

- Path generation improvements: The current spline approximation method approximates the best possible spline minimizing the least squared distance error given the noisy waypoints and certain design parameters. However, the path generation for a vehicle should consider more factors such as temporal consistency, respecting vehicle dynamics, respecting road boundaries, etc. One may formulate the spline approximation problem with these factors as constraints to get a more viable reference path for following. This also calls for a method for online calculation of the goodness of fit for the generated reference path so that the control algorithms can take appropriate action to avoid aggressive responses for an improper path.
- Inclusion of delays: In the current implementation of the controllers and fusion algorithm, the effects of various delays such as communication delays, actuation delays, and data processing delays are not considered. Thus, to improve the robustness of all these algorithms, the effect of these delays should be studied.
- Producing object-level data: The common automotive sensors usually employ some clustering and filtering techniques to produce the object-level target data. The reference point for which this object-level data is produced might be different for different sensors. For example, the camera may produce the center of the bounding box that it obtains, whereas radar may produce the center of the cluster it obtains from the reflections it receives. These properties of the measurements need to be properly understood before implementing the T2Tf algorithms. This also calls for Siemens Prescan to include low-level clustering and filtering algorithms in its sensors library to produce object-level data that can be used in the development of T2Tf algorithms.
- Including dynamic behavior of the bus: In this work, a kinematic model with first-order longitudinal and steering dynamics is used to represent the vehicle model, which does not include some of the dynamic and handling behavior of the vehicle. Further study and/or experimental analysis are required to analyze these behaviors.

Bibliography

[1] Digitale infrastructuur voor toekomstbestendige mobiliteit (ditm). `https://brainporteindhoven.com/nl/ondernemen-en-innoveren/markten/mobility/programmabureau-smart-green-mobility/digitale-infrastructuur-voor-toekomstbestendige-mobiliteit`. Date Accessed: 13/11/2022.

[2] Jeroen Ploeg. Analysis and design of controllers for cooperative and automated driving. 2014.

[3] Jeroen Ploeg, Bart TM Scheepers, Ellen Van Nunen, Nathan Van de Wouw, and Henk Nijmeijer. Design and experimental evaluation of cooperative adaptive cruise control. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 260–265. IEEE, 2011.

[4] Robbin Bernard Adrianus van Hoek. Cooperative trajectory planning for automated vehicles. 2021.

[5] Robbert Janssen, Han Zwijnenberg, Iris Blankers, and Janiek De Kruijff. Truck platooning driving the future of transportation. Technical report, TNO, 2015.

[6] Simcenter prescan software simulation platform. `https://plm.sw.siemens.com/en-US/simcenter/autonomous-vehicle-solutions/prescan/`. Date Accessed: 13/07/2023.

[7] Tim Weilkiens. *Systems engineering with SysML/UML: modeling, analysis, design*. Elsevier, 2011.

[8] Georg Nestlinger, Johannes Rumetshofer, and Selim Solmaz. Leader-based trajectory following in unstructured environments—from concept to real-world implementation. *Electronics*, 11(12):1866, 2022.

[9] Zhenji Lu, Barys Shyrokau, Boulaid Boulkroune, Sebastiaan Van Aalst, and Riender Happee. Performance benchmark of state-of-the-art lateral path-following controllers. In *2018 IEEE 15th International Workshop on Advanced Motion Control (AMC)*, pages 541–546. IEEE, 2018.

[10] Jarrod M Snider et al. Automatic steering methods for autonomous automobile path tracking. *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RITR-09-08*, 2009.

[11] Salvador Dominguez-Quijada, Alan Ali, Gaëtan Garcia, and Philippe Martinet. Comparison of lateral controllers for autonomous vehicle: experimental results. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC 2016)*, 2016.

[12] Daizhan Cheng, Alberto Isidori, Witold Respondek, and Tzyh Jong Tarn. Exact linearization of nonlinear systems with outputs. *Mathematical systems theory*, 21(1):63–83, 1988.

[13] Sebastiaan v.d. Eijnden Henk Nijmeijer, Erjen Lefeber. 4DM30 Non linear control lecture notes. 2019.

[14] Hassan K. Khalil. *Nonlinear Systems*. Prentice Hall, 2002.

[15] Ksander N de Winkel, Tugrul Irmak, Riender Happee, and Barys Shyrokau. Standards for passenger comfort in automated vehicles: Acceleration and jerk. *Applied Ergonomics*, 106:103881, 2023.

[16] Lingli Yu, Yu Bai, Zongxv Kuang, Chongliang Liu, and Hao Jiao. Intelligent bus platoon lateral and longitudinal control method based on finite-time sliding mode. *Sensors*, 22(9):3139, 2022.

[17] Owen McAree and Sandor M. Veres. Lateral control of vehicle platoons with on-board sensing and inter-vehicle communication. pages 2465–2470. Institute of Electrical and Electronics Engineers Inc., 2016.

[18] Nikolaj Ezhov, Frank Neitzel, and Svetozar Petrovic. Spline approximation, part 1: Basic methodology. *Journal of Applied Geodesy*, 12(2):139–155, 2018.

[19] Xin Tian, Yaakov Bar-Shalom, D Choukroun, Y Oshman, J Thienel, and M Idan. Track-to-track fusion architectures–a review. In *Proc. Itzhack Y. Bar-Itzhack Memorial Symp. Estimation, Navigation, Spacecraft Control*, pages 200–213, 2012.

[20] Prabhat K Sharma, TPJvd Sande, Henk Nijmeijer, Robin Smit, Frank Evers, and Alexis Siagkris. Data association algorithms for multi target tracking in a probabilistic framework. 2021.

[21] Simo Särkkä. *Bayesian filtering equations and exact solutions*, page 51–63. Institute of Mathematical Statistics Textbooks. Cambridge University Press, 2013.

[22] Chee-Yee Chong, Shozo Mori, William H Barker, and Kuo-Chu Chang. Architectures and algorithms for track association and fusion. *IEEE Aerospace and Electronic Systems Magazine*, 15(1):5–13, 2000.

[23] Kaipei Yang, Yaakov Bar-Shalom, and Kuo-Chu Chang. Information matrix fusion for nonlinear, asynchronous and heterogeneous systems. In *2019 22th International Conference on Information Fusion (FUSION)*, pages 1–6. IEEE, 2019.

[24] Yaakov Bar-Shalom, X Rong Li, and Thiagalingam Kirubarajan. *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2001.

[25] MATLAB. Design driving scenarios, configure sensors, and generate synthetic data. `https://nl.mathworks.com/help/driving/ref/drivingscenariodesigner-app.html`. Date Accessed: 02/04/2023.

[26] Franciscus Nicolaas Hoogeboom. Safety of automated vehicles: design, implementation, and analysis. 2020.

[27] EN ETSI. 302 637-2 v1. 3.1-intelligent transport systems (its); vehicular communications; basic set of applications; part 2: Specification of cooperative awareness basic service. *ETSI, Sept*, 2014.

A Project management plan

Figure A.1 shows the overall project plan. We employed the V-model approach for carrying out this project. Based on this V-model, certain milestones are identified at different stages of the project.

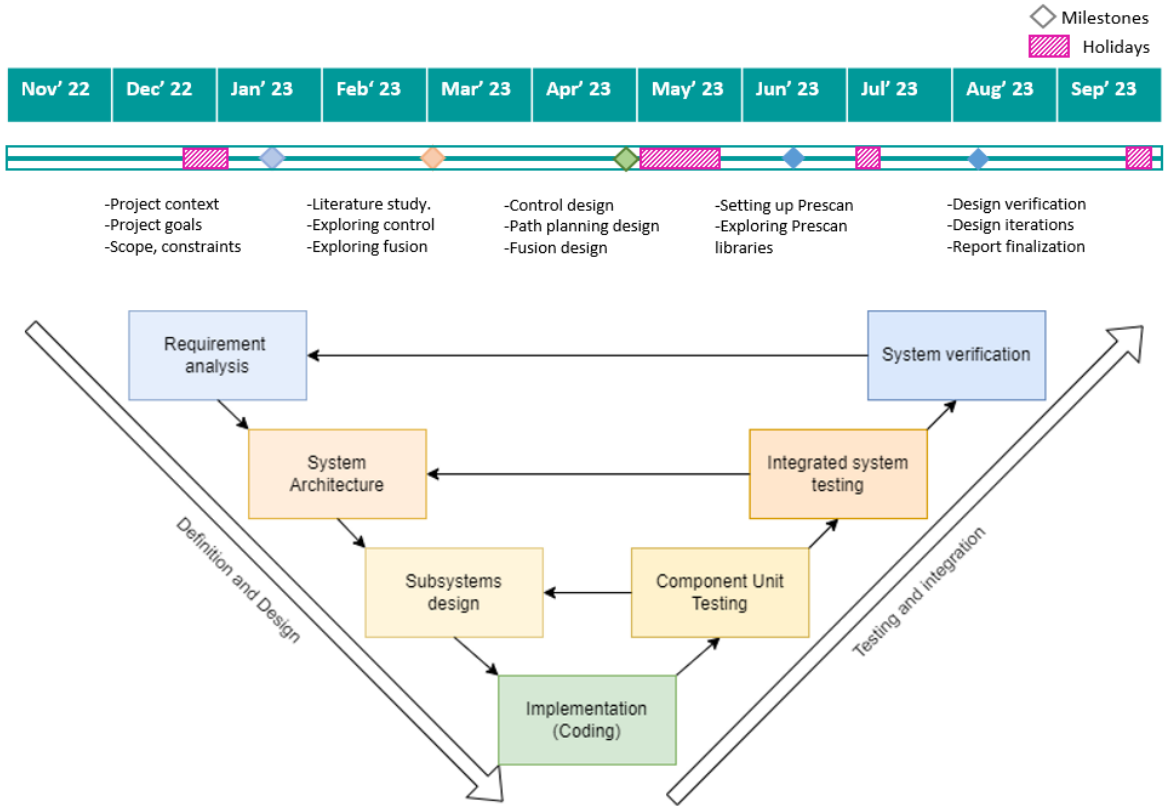


Figure A.1: Project management plan

B Vehicle parameters

Table B.1 lists the parameters used for the dynamic vehicle model for the buses considered in this work.

Table B.1: Hino bus dimensions [6]

Parameters	Symbol	Values
Wheelbase	l	$5.6\ [m]$
Rear overhang	l_r	$2.7\ [m]$
Front overhang	l_f	$2.5\ [m]$
Overall length	L	$10.8\ [m]$
Overall width	W	$2.5\ [m]$

C MATLAB code

C.1 Path generation

```
function [waypoints, kappa, s_approx, slope] = mySplines(P, n)
% Spline approximation
% Author: Prabhat K Sharma (p.k.sharma@tue.nl)
% Based on
% [1] Ezhov, N.; Neitzel, F.; Petrovic, S.
% Spline approximation, Part 1: Basic methodology

% version 1.2: curvature calculations added

% Input(s):
% p : Nx2 array of control points
% n : Number of spline segments to be formed

% Output:
% waypoint: 1000x2 array representing the smoothend curve
% kappa: curvature values at each waypoint
% s: curvilinear distance

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% t calculation: constructing the non-decreasing sequence
N = length(P);
t = nan(N,1);
t(1) = 0;
for i=2:N
    t(i) = i;
end

t = t./t(end); % normalize the t vector

% assign number of data points to each section
mj = nan(n,1);
mj(1:n-1) = floor(N/n);
mj(n) = floor(N/n) + mod(N,n);

% knot sequence
k_ind = nan(n+1,1);
k_ind(1) = 1;
k_ind(end) = N;
k = nan(n-1,1); % knot sequence initialization
for i=1:n-1
    ind = 1;
```

```
    for j=1:i
        ind = ind+mj(i);
    end
    k_ind(i+1) = ind;
    k(i) = t(ind);
end

% Observation matrix
Lx = P(:,1);
Ly = P(:,2);

% Design matrix (A)
A = zeros(N,4*n);
for i=1:n
    m = mj(i);
    A_temp = nan(m,4);
    for j = 1:m
        t_ij = t(k_ind(i)+j-1);
        A_temp(j,:) = [1 t_ij t_ij^2 t_ij^3];
    end
    if i<n
        A(k_ind(i):k_ind(i+1)-1,4*(i-1)+1:4*(i-1)+4) = A_temp;
    else
        A(k_ind(i):k_ind(i+1), 4*(i-1)+1:4*(i-1)+4) = A_temp;
    end
end
Ax = A;
Ay = A;

% Continuity matrices (C)
C0 = zeros((n-1),4*n);
C1 = zeros((n-1),4*n);
C2 = zeros((n-1),4*n);
for i=1:n-1
    t_i = k(i);
    C0(i,4*(i-1)+1:4*(i-1)+8) = [1 t_i t_i^2 t_i^3 -1 -t_i -t_i^2 -t_i^3];
    C1(i,4*(i-1)+1:4*(i-1)+8) = [0 1 2*t_i 3*t_i^2 0 -1 -2*t_i -3*t_i^2];
    C2(i,4*(i-1)+1:4*(i-1)+8) = [0 0 2 6*t_i 0 0 -2 -6*t_i];
end

C = [C0; C1; C2];
Cx = C;
Cy = C;

% P matrices
Pxx = 0.5*eye(N,N);
Pyy = 0.5*eye(N,N);

% Solving matrix equations for coefficients and lagrange multipliers
A_bar_rows = size(Ax,2)+size(Cx,1);
A_bar_cols = A_bar_rows;
A_bar = zeros(A_bar_rows, A_bar_cols);
A_bar(1:size(Ax,2), 1: size(Ax,2)) = Ax'*Pxx*Ax;
A_bar(size(Ax,2)+1:end,1:size(Cx,2)) = Cx;
A_bar(1:size(Cx,2),size(Ax,2)+1:end) = Cx';

L_bar_x = zeros(size(A_bar,1),1);
```

```
L_bar_x(1:size(Ax,2)) = Ax'*Pxx*Lx;

L_bar_y = zeros(size(A_bar,1),1);
L_bar_y(1:size(Ax,2)) = Ay'*Pyy*Ly;

X_bar = inv(A_bar)*L_bar_x;
Y_bar = inv(A_bar)*L_bar_y;

%%

n_wp = 1000; % number of waypoints
t_q = linspace(0,t(end),n_wp);
x_est = nan(n_wp,1);
y_est = nan(n_wp,1);
kappa = nan(n_wp,1);
slope = nan(n_wp,1);
s_approx = nan(n_wp,1);
s_approx(1) = 0;

j=1;
lim = k(j);
for i=1:length(t_q)
    t_temp = t_q(i);
    if t_temp≥lim && j≤length(k)
        j=j+1;
        if j<length(k)+1
            lim = k(j);
        else
            lim = k(j-1);
        end
    end

    x_est(i) = X_bar(4*(j-1)+1:4*j)'*[1; t_temp; t_temp^2; t_temp^3];
    y_est(i) = Y_bar(4*(j-1)+1:4*j)'*[1; t_temp; t_temp^2; t_temp^3];

    % Calculate the curvature analytically C = (y"x'-x"y')/(x'^2+y'^2)^(3/2)
    xdot = X_bar(4*(j-1)+1:4*j)'*[0; 1; 2*t_temp; 3*t_temp^2];
    xddot = X_bar(4*(j-1)+1:4*j)'*[0; 0; 2; 6*t_temp];

    ydot = Y_bar(4*(j-1)+1:4*j)'*[0; 1; 2*t_temp; 3*t_temp^2];
    yddot = Y_bar(4*(j-1)+1:4*j)'*[0; 0; 2; 6*t_temp];

    kappa(i) = (yddot*xdot-xddot*ydot)/(xdot^2+ydot^2)^(3/2);

    % approximated curvilinear distance
    if i>1
        s_approx(i) = s_approx(i-1) + norm([x_est(i), ...
            y_est(i)]-[x_est(i-1), y_est(i-1)]);
    end

    % Slope of the curve
    slope(i) = atan2(ydot,xdot);

end

waypoints = [x_est, y_est];
```

D Additional simulation plots

Figure D.1 shows the longitudinal controller's response for the start-stop-start scenario presented in chapter 5. It shows that the follower buses also come to a complete halt if the leader bus (Bus 0) waits for a longer duration ($15s$ instead of $5s$).

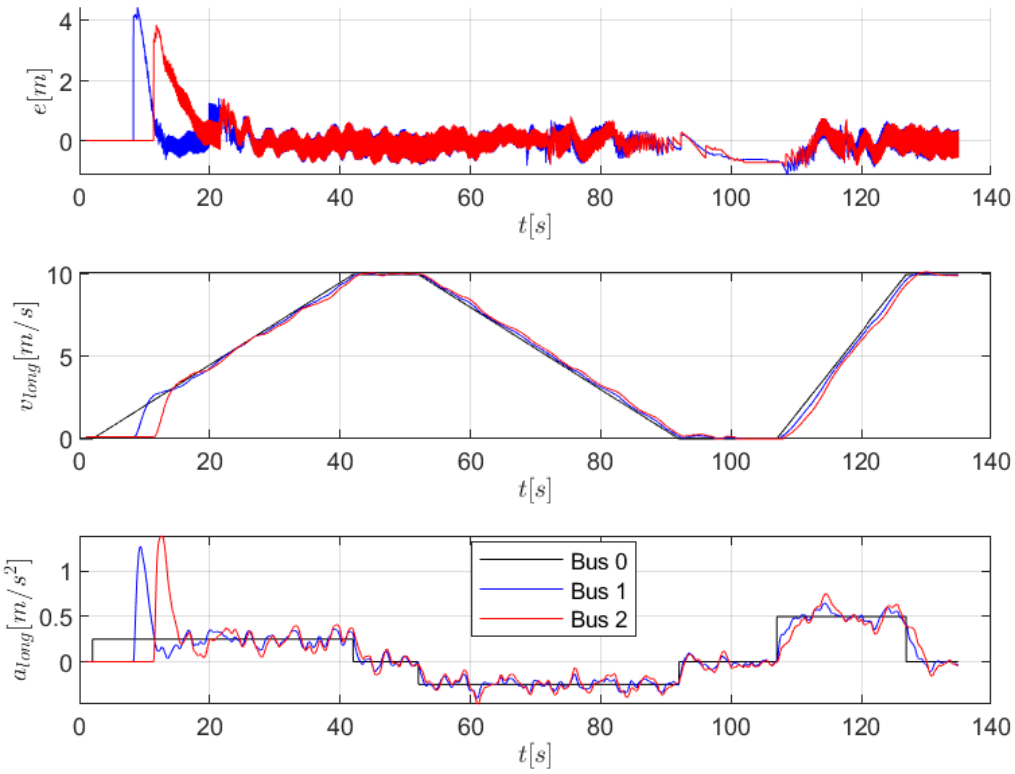


Figure D.1: Start-stop-start: longitudinal control response with $15s$ halt for Bus 0

About the Author

Prabhat Kumar Sharma received his M.Sc. in Automotive Technology (2021) with a specialization in dynamics and control at Eindhoven University of Technology (TU/e), Eindhoven, the Netherlands. During his studies, he carried out his research project concerning probabilistic data association for target tracking in autonomous vehicles at TNO, Helmond. Currently, he is pursuing his Engineering Doctorate in Automotive Systems Design at TU/e. For his individual year-long project, he is working with Siemens Industry Software B.V., Helmond. His project concerns with the design and development of longitudinal and lateral control for the city bus platooning application. He is also taking an integrated approach toward developing a high-level track-to-track fusion algorithm based on the controller's requirements. His career interests lie in cooperative and automated vehicles, mechatronics systems design, and robotics.

PO Box 513
5600 MB Eindhoven
The Netherlands
tue.nl

EngD AUTOMOTIVE SYSTEMS DESIGN
Track AUTOMOTIVE SYSTEMS DESIGN

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY