Autonomous Vehicle Social Behavior for Highway Driving

Submitted in partial fulfillment of the requirements for

the degree of

Doctor of Philosophy

in

Department of Electrical and Computer Engineering

Junqing Wei

B.S., Automation, Tsinghua University
M.S., Electrical and Computer Engineering, Carnegie Mellon University

Carnegie Mellon University
Pittsburgh, PA

May 2017

For my son Andrew Moyuan Wei

Abstract

In recent years, autonomous driving has become an increasingly practical technology. With state-of-the-art computer and sensor engineering, autonomous vehicles may be produced and widely used for travel and logistics in the near future. They have great potential to reduce traffic accidents, improve transportation efficiency, and release people from driving tasks while commuting. Researchers have built autonomous vehicles that can drive on public roads and handle normal surrounding traffic and obstacles. However, in situations like lane changing and merging, the autonomous vehicle faces the challenge of performing smooth interaction with human-driven vehicles. To do this, autonomous vehicle intelligence still needs to be improved so that it can better understand and react to other human drivers on the road.

In this thesis, we argue for the importance of implementing "socially cooperative driving", which is an integral part of everyday human driving, in autonomous vehicles. An intention-integrated Prediction- and Cost function-Based algorithm (iPCB) framework is proposed to enable an autonomous vehicles to perform cooperative social behaviors. We also propose a behavioral planning framework to enable the socially cooperative behaviors with the iPCB algorithm. The new architecture is implemented in an autonomous vehicle and can coordinate the existing Adaptive Cruise Control (ACC) and Lane Centering interface to perform socially cooperative behaviors. The algorithm has been tested in over 500 entrance ramp and lane change scenarios on public roads in multiple cities in the US and over $10,000$ in simulated case and statistical testing. Results show that the proposed algorithm and framework for autonomous vehicle improves the performance of autonomous lane change and entrance ramp handling. Compared with rule-based algorithms that were previously developed on an autonomous vehicle for these scenarios, over 95% of potentially unsafe situations are avoided.

Acknowledgments

First and foremost, I want to thank my wife, Dr. Shimeng Huang. She is always so supportive of my research. For the most challenging times when I had to work in the industry during the day and do research late at night, she took care of our family so well, making it possible for me to focus on my study. Many times when I felt frustrated and started to question my approach, she was always beside me and gave the confidence to solve the real issues in my algorithm.

I would like to thank my advisor, Professor John Dolan. He gave me the freedom to find interesting and challenging problems within the autonomous driving area. He always provides essential guidance in my research. He taught me how to be a good researcher and more importantly how to become a better person. I would never expect a better PhD advisor than him.

Thanks to my sponsor General Motors that supported me for my PhD study, especially Dr. Bakhtiar Litkouhi. This thesis is possible because of his strong support of this research topic. And thanks for his sharp and great advice that allowed this thesis to go beyond theory and link tightly to the actual challenges the industry is facing.

Thanks to all the members of the autonomous driving lab in CMU, Junsung Kim, Gaurav Bhatia, Jongho Lee, Tianyu Gu, Hyunggi Cho, Christopher Baker, Dr. Paul Rybski and Professor Raj Rajkumar; colleagues in Ottomatika and Delphi, Ludong Sun, Guchan Ozbilgin and Jonathan Wieskamp. Thanks to Wenda Xu who has spent endless time discussing algorithms with me and also built the full perception system to make the testing of my research feasible. Special thanks to Jarrod Snider, who supported this project when we worked together in CMU, Ottomatika and Delphi for the last 8 years. He taught me lot about how to make a real autonomous driving system work.

Thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Autonomous driving has been a promising research area since the 1990s. With the current state of the art in computer and sensor engineering, autonomous vehicles may be produced and widely used for travelling in the near future. Autonomous vehicles have great potential to avoid most traffic accidents caused by human driver errors, which lead to over 1,200,000 deaths and 50,000,000 injuries each year worldwide [1, 35]. They will also benefit the overall performance of the current traffic system by efficiently increasing highway capacity and decreasing traffic congestion in cities [62]. Fully autonomous driving is able to spare people from the task of driving while commuting. It is found that an average American driver spends 55 minutes a day driving a car [34]. The total time savings by implementing fully autonomous driving will provide great value to human society.

Since the 1980s, autonomous vehicle intelligence has increased from lane centering to actually driving on public roads with lane-changing capability [10, 18, 36, 44, 45, 51]. A few research platforms have shown great capability in driving on public roads [5, 28, 60]. Nevertheless, in the short term, human-driven vehicles will continue to predominate. Therefore, it is important to improve the autonomous vehicle's ability to drive intelligently on the road with surrounding human-driven vehicles [51].

Human drivers cooperate implicitly or explicitly with one another. They understand each other's intentions and make decisions based on an estimate of other drivers' intentions and behaviors. This interaction allows human drivers to behave in a socially acceptable way in many

driving scenarios, such as entrance ramps and lane changing. Without it, an autonomous vehicle may not perform reasonably in these scenarios, leading to low transportation efficiency and even undesirable situations. Therefore, to enhance driving performance on public roads with human traffic involved, it is important for an autonomous vehicle's decision-making system to be able to socially cooperate with human drivers.

1.1 Thesis Statement

Autonomous vehicles driving on public roads need to perform proper socially cooperative behavior with human-driven vehicles to achieve an acceptable level of driving safety and comfort. If an autonomous vehicle can understand human drivers' intentions and make decisions based on prediction of their future behavior, including reactions to the autonomous vehicle's potential movement, it will drive with better comfort, safety and smoother cooperation with surrounding vehicles.

1.2 Thesis Contributions

1. A novel behavior-planner integration framework that enables robust freeway autonomous driving and social cooperation with surrounding vehicles.
2. A Prediction-and Cost function-Based (PCB) algorithm for autonomous vehicles to perform social behaviors in a wide variety of scenarios.
3. A quantitative analysis of the importance of social behavior for autonomous vehicles based on experiments in simulation and on public roads. Comparing with algorithms that do not take social behavior into account, the PCB algorithm can potentially decrease the precarious cases for lane change and entrance ramps by over 90%.

1.3 Document Outline

The motivation and expected contributions of this thesis are introduced in Chapter 1. Chapter 2 covers a summary of related work that inspires the approach developed in the thesis. The autonomous robot social behavior problem and the application domain for the thesis are defined in Chapter 3.1. The autonomous vehicle platform and the mechanism of behavioral planning in the robot's software system are introduced in Chapter 4. A Prediction- and Cost-function Based algorithm for high-level planning is proposed in Chapter 5 to improve the robot's performance in cooperation with human participants. Results of applying the PCB algorithms to the autonomous driving domain are given and a new autonomous driving platform for conducting on-road experiments is introduced. In Chapter 6, the conclusions are drawn based on test results.

Chapter 2

Related Work

In this section, we summarize research related to improving the intelligence of autonomous vehicles. The abilities of several prominent autonomous driving platforms are summarized in Section 2.1, and those of current off-the-shelf vehicle driver assist systems are described in Section 2.2. State-of-the-art algorithms of motion planning, and autonomous vehicles to handle lane changes and entrance ramps are discussed in Section 2.3, 2.4 and 2.5. Prior investigations on driving behavior recognition and emulation of human driver behavior inspire this thesis, and are respectively described in Section 2.6 and 2.7, respectively. Some recent approaches to the robot social navigation problem are discussed in Section 2.8.

2.1 Autonomous Vehicle Platforms

In the 1990s, E. Dickmanns and his colleagues implemented an autonomous driving platform based on their 4-D approach to vision data processing [10, 18]. The NAVLAB project at CMU has built a series of experimental platforms which are also able to run autonomously on freeways [36, 45]. The PATH project in California showed a very promising demo of the potential functionality of the Automated Highway System (AHS) in 2003 [44]. The vehicles could form a vehicle platoon and perform lane changing with the help of a V2V communication system.

In 2004 and 2005, two DARPA Grand Challenge events that attracted the attention of both academia and industry were held with the objective of achieving fully autonomous driving in

an unstructured environment with no human-driven traffic. The robots had to drive on an unpaved road with sharp turns, narrow tunnels, etc. while avoiding obstacles. Five teams successfully completed the challenge. The major challenges for the teams were localization, real-time road shape estimation and static obstacle avoidance [46, 49]. The competition considerably furthered the technology of autonomous driving. In 2007, the DARPA Urban Challenge provided researchers another practical scenario in which to test the latest sensors, computer technologies and artificial intelligence algorithms in an urban autonomous driving environment [30, 51]. The participating vehicles could perform off-road/on-road driving, lane changing, parking and intersection handling with stop signs. Basic interaction with human-driven vehicles in urban environments was preliminarily verified in the Urban Challenge.

In 2011, Google released its autonomous driving platform [28]. Its main sensors are radars and a rotational multi-beam laser scanner installed on a customized roof-rack. The autonomous driving system has logged over 2 million miles on public roads. It needs a high-resolution 3D map acquired by driving the route in advance, which provides robust vehicle localization. In 2010, the VisLab vehicle from Parma University crossed Europe and Asia autonomously [5]. Vision technology is heavily used in this system and shows great potential. Since 2013, more and more autonomous driving platforms are built with improved sensor integration and user interface design [14, 19, 26]. Significantly more public road testing has been deployed as a way to improve the capability of autonomous cars.

These platforms contribute dramatically to achieving the goal of autonomous driving for people's everyday lives. However, there are still many limitations of these systems, such as the cost of sensors and computing platforms, the absence of user friendly interfaces and more importantly the level of intelligence of these robots. Only a few of these platforms focus on improving smoothness and perform human-like behaviors in basic scenarios such as distance keeping and lane centering. The autonomous vehicles' ability to drive in dense traffic and cooperate with human-driven vehicles also still needs to be developed.

2.2 Driving Assist Capabilities

In the automotive industry, Adaptive Cruise Control (ACC) has already been installed in some factory vehicles from major car manufacturers [61]. Some advanced ACC systems use fuzzy logic and neural network control to improve their car-following performance [52]. Stop-and-go ACC has also been implemented to control the vehicle's speed in low-speed scenarios such as congestion on freeways [53]. Using vehicle-to-vehicle communication (V2V), vehicles with ACC can keep shorter but safe distances between each other, which also improves overall highway capacity and performance [62]. In recent years, lane centering / lane keeping assist technology has also been developed to improve freeway driving safety [55]. A few prototype freeway single-lane autonomous driving platforms integrating these basic modules have been released by auto manufacturers [17, 54]. Though these platforms show potential to improve the driver's experience and safety, they can only perform single-lane autonomy, and in limited scenarios. These systems are also not able to consider the cooperation between the host vehicle and surrounding cars. Therefore they are not able to properly deal with scenarios needing interaction between cars, such as entrance ramps, cutting-in vehicles, etc. Because of all these limitations, human supervision is always required so that the human driver can take over when the system encounters scenarios it cannot handle.

2.3 Motion Planning

Motion planning research targets generating a feasible trajectory for the automated car to execute. It also serves the purpose of avoiding static and dynamic obstacles. The three main approaches are based on searching, sampling and optimization [47]. State-of-the-art algorithms will also incorperate a module to anticipate moving objects movements based on some simple assumptions such as that cars follow the road with constant curvature [29, 63]. Using such a planner, the vehicle is able to perform lane changes and obstacle avoidance with speed adjustments. However, the assumption that the moving objects follow constant speed and curvature is not valid in many practical cases. Therefore, the planner needs to replan or change the decision very often while interacting with actual surrounding human-driven traffics, which leads to poor

driving smoothness [20]. If more realistic assumptions are used about moving object, the computation expense will be too high for the computers to handle in real-time for both search-based and optimization-based algorithms. The dimension of the search space increases exponentially because these algorithm also need to consider the shape of the road, trajectory feasibility and a very wide-range of speed variations [63].

Nilsson contributed an automated driving planner based on Model Prediction Control and convex optimization [33]. It considers circumventing and avoiding surrounding vehicles and the smoothness of the trajectory. Its strength is that no action space discretization need to be applied. However, it doesn't consider how the behavior of the host vehicle could affect surrounding cars. More importantly, not all scenarios in which automated vehicles need to make decisions can be modeled as a convex cost problem.

To enable the capability for an autonomous vehicle to consider the interaction between itself and surrounding human-driven cars, the motion planning framework need to be adjusted. In this thesis, we propose that a behavioral planner should be implemented that focuses on making decisions about the interactions between autonomous vehicles and surrounding traffic.

2.4 Autonomous Lane Change Mechanism

Semi-automatic lane change on the highway has been provided by multiple car manufacturers as an advanced driving assist feature [6]. Usually, there are two main components in these systems:

- Adaptive cruise controller (ACC), which keeps a safe distance to the front car automatically. When there is no leading vehicle, the ACC will converge to the commanded speed limit.
- Lane centering controller, which keeps the car in the center of the lane or performs a lane change.

Most production or developing automated highway driving systems use radar or camera sensors to detect whether there are cars beside the host vehicle. Then the vehicle will follow simple logic to decide whether the car can move laterally or not. However in situations with slightly dense traffic or in which the autonomous vehicle needs to search for the best gap between sur-

rounding cars to perform a lane change, the rule-based lane change fails very often. It is either not able to perform a proper lane change or it will fails to safely and smoothly merge into the desired lane as human drivers are able to do.

Different automatic lane change algorithms with speed adjustment and lateral movements have been explored by multiple groups [25, 32]. As for most production systems, the occupancy and time-to-crash criteria are used to command the lane change to the lateral controller. An advanced model to emulate a safe and comfortable human lane change is proposed by [22]. It allows the vehicle to shrink its comfortable zone to merge into a relatively narrow gap. However, most of these algorithms are either based on rules or some simple prediction assuming surrounding cars will keep a constant speed [29]. This thesis shows that, because of the limited prediction capability, in more dynamic cases such as heavy traffic or when the surrounding vehicles do not follow constant speed, these algorithms will not allow the car to perform a lane change safely and comfortably.

Instead of addressing the autonomous lane change application as a planning problem, it is possible to model it as a Markov Decision Process (MDP) problem. In this case, for each state in the state space, an optimal decision is already pre-computed using MDP solvers such as dynamic programming. However, the state transition model for MDP is often unknown. Therefore, a reinforcement learning approach is proposed for lane change decision making by Shalev-Shwartz [39]. In this approach, Q-learning is applied in as simulated environment where all the vehicles need to perform multiple lane changes within a limited distance horizon. The result shows that without any hand-coding of policy, the vehicle is able to handle fairly challenging lane change situations by adjusting its speed and lateral position. However, in the training process, the algorithm assumes all the cars are running the same policy. This greatly reduces the robustness of this algorithm to different human drivers. It could even cause the algorithm to converge to a local optimum such that the policy only works in scenarios where it interacts with agents using the same policy.

2.5 Autonomous Entrance Ramp Handling

Another challenging area for basic autonomous highway driving is entrance ramp handling. A human-driven vehicle will enter entrance an entrance ramp and adjust its speed with the purpose of merging into the main road with the least interruption of the traffic flow. However, most autonomous driving system cannot understand the significance of the speed adjustment of the merging-in vehicles and therefore cannot cooperate with them as human drivers usually do with each other.

Multiple groups have recognized the inefficiency of entrance ramps and started to solve the problem using a centralized solution [21, 56]. Though, the algorithm can achieve the optimal efficiency at entrance ramps in simulation, the solution is based on the assumption that all the vehicles' speeds are controlled by a centralized algorithm and they communicate with each other with Vehicle-To-Vehicle (V2V) communication. Therefore, these systems cannot be applied to autonomous vehicle to improve their entrance ramp handling capability without adding V2V communication centralization.

To ensure safety on entrance ramps, a modified ACC algorithm is proposed in [59]. In this algorithm, when the vehicle gets close to the entrance ramp, it will not only slow down vehicles in front of it but also the vehicle merging in. Using this mechanism, the automated vehicle will be able to keep a safe distance to cars merging in. However, in cases in which the merging vehicle slows down to yield to the automated vehicle in the main lane, the automated car will also decelerate hard, which is the opposite of what the merging vehicle expects to happen.

In light of the highlighted deficienceies in prior algorithms, an automated driving algorithm to handle entrance ramp needs to be developed.

2.6 Emulating Human Behavior

For the autonomous vehicle to make decisions like a good human driver, its intelligence needs to understand quantitatively what constitutes good driving behavior. A straightforward performance metric for autonomous vehicles is their degree of similarity to human drivers.

Therefore, researchers have been developing mechanisms to allow autonomous vehicles to

emulate human driver behavior. For the distance keeping application, there are models proposed to emulate a human driver's acceleration output using as few as three parameters [16]. This model, and some further enhancement of it, worked well in microscopic traffic simulations, which focus on simulating an individual vehicle's behavior [2]. For an autonomous vehicle, this simplified model introduces some undesired human driver behaviors, such as latency caused by human response delays. Learning algorithms such as Artificial Neural Networks (ANN) have also been implemented in lane keeping and adaptive cruise control. NAVLAB at CMU implemented an ANN-based algorithm to perform steering control on freeways [36]. The ANN is trained by the front-view images of the vehicle and the human driver's steering wheel operation. The major constraint of learning algorithms is their dependency on training data. Therefore, no safety criteria can be verified in untrained scenarios.

Inverse optimal control is another technology for emulating a human behavior, and focuses on learning the objective of a human behavior [37]. It assumes that a human behaves statistically similarly to a planner that is trying to optimize some costs. The objective of Inverse Optimal Control is to find the cost term-driven human behavior and the weight for each cost term. Using this mechanism, a robot can follow the same cost criteria and perform similarly to or better than a human in a wider variety of scenarios. This approach has been applied in multiple robotics areas, including offline path planning for field robots, step planning for a quadruped robot and also control of robotic arms and hands with a high number of degrees of freedom [40]. The results show that learning from very limited human demonstration, the behavior quality of robots is greatly improved. However, this approach has only been applied to a robot planning in a static environment, where no other contacts, either human or another robot, are involved. Therefore, the interactions between a host robot and other agents in the environment are not modeled in this learning-based approach. In this thesis, we will create a set of cost functions for autonomous vehicle decision making through emulating human drivers, where the host vehicle's influence on surrounding human-driven vehicles will also be considered.

2.7 Human Behavior Prediction

Better cooperation with human drivers will result from having autonomous vehicles be able to predict future human behavior through understanding their intention. To recognize human intention, a widely used mechanism is based on Bayesian models [13]. First, a model of human behaviors given a certain intention is created. Then from the observation of actual human behaviors, the probability of intention is inferred using Bayes' Rule. Using a similar approach, researchers have shown that human driver intention in lane change, turning, stopping, etc. can be estimated with reasonable accuracy. To improve driving safety, Morris et al. developed a lane change intention prediction system based on vision and surrounding vehicle information [31]. Various features are first extracted from the perception system and then fed into a classifier, which outputs the probability of lane change a few seconds later. However, because vehicle-internal video of the driver is the main input of the system, this algorithm can only be applied to the host vehicle. Therefore, it is not suitable to predict behaviors of surrounding human driving vehicles.

To better predict human pedestrian behavior, Kitani et al. developed a hidden variable Markov Decision Process (hMDP) activity forecasting mechanism [24], which incorporates uncertainty and observation noise into the MDP model. It is assumed that a human pedestrian's behavior is driven by some goals represented by costs, including getting to a position, avoiding obstacles, etc. Therefore, observing a pedestrian's movements allows his probable destination to be estimated. Based on that, the probabilities of the future trajectories of the pedestrian are generated. This algorithm usually consumes considerable computation power and needs very accurate statistical data of how humans behave in a specific domain. The current hMDP approach also doesn't model the interaction between agents in the environment. For example, if one pedestrian is in the way of another person's path to a destination, the behavior of both pedestrians will not be accurately predicted using the hMDP model.

In this thesis, we apply the Bayesian model-based intention recognition algorithm to the autonomous driving domain to understand surrounding vehicles' intentions. Then interaction between vehicles is also integrated into the human driver behavior prediction model.

2.8 Robot Social Cooperation

As discussed in Chapter 1, to achieve better performance in human environments, robots need to actively interact with human participants. Therefore, besides the passive intention recognition of human drivers, researchers have also put efforts into developing robots that can actively cooperate with humans.

In the human-robot interaction area, researchers have developed planners for robots to navigate in an environment shared with human participants. Kirby et al. proposed a biased oval-shaped occupancy region around humans for use by the robot planner [23]. The associated cost causes the robot to perform more socially acceptable maneuvers by, for example, circumventing to the left of human pedestrians in hallways and keeping a comfortable distance from human participants. However, this approach regards human pedestrians as static obstacles with a specialized cost. This works well for indoor navigation, wherein both pedestrians and the robot move relatively slowly and the robot can easily stop within meters. But for high-speed applications, such as freeway autonomous driving, the plan horizon (proportional to the speed of the car) is usually much longer because of the high inertia of the car. Without an accurate prediction of the movement of surrounding cars, the autonomous vehicle cannot make safe and robust decisions within the expected plan horizon.

In the decision theory area, there are frameworks supporting the modeling of robot social behavior, such as POMDP [9]. Using POMDP, Broz et al. introduce human intention as a partially observable state in decision making [7]. One simulation experiment shows that in an intersection scenario, an autonomous vehicle can cooperate with another car by understanding whether it wants to yield or not and performing proper actions. An attractive feature of the POMDP-based approach is that its decision is theoretically optimal. However, because of the computational expense and difficulty in building an accurate probabilistic state transition model, it has limited extensibility. For example, if there is more than one human-driven vehicle to deal with, the difficulty of abstracting the cooperation into a POMDP model increases exponentially. In this thesis, we propose a decision making model for an autonomous vehicle to perform social cooperation that uses the probability of the movements of surrounding cars and is also adaptive to a wide variety of driving scenarios without the exponential-increase difficulty.

There are algorithms being developed for multi-agent systems in which the agents interact with each other. Two example applications are intersection handling and robot soccer [11, 12, 27]. In the intersection handling case, there could be a combination of human-driven vehicles and autonomous vehicles and communication is allowed between autonomous vehicles. Dresner et al. proposed a reservation-based algorithm and demonstrated how the traffic agents will interact with each other to achieve better throughput without using a traffic light [12]. However, for handling human-driven cars, the principal concept is for autonomous vehicles to yield to any human-operated vehicles instead of actively interacting with them as human drivers do with each other.

In robot soccer or humanoid robot control, a policy could be generated to achieve optimal performance in a simulated environment where the state transition for the robot and its interaction with the simulated physical world are well modeled [27]. However, when applying the learned policy to the real robot, there is always a difference between the simulated environment and the real environment, which causes the controller developed in simulated environment to fail. A grounded simulation learning approach is proposed to iteratively adjust the policy of the controller by observing how it works in the real environment and then adjust the simulation parameters [15]. The optimal policy is generated after each round of simulation parameter adjustment. Compared with learning purely from testing results in the real world, this method greatly speeds up the policy generation process. However, for autonomous vehicle interacting with surrounding cars, this approach cannot directly apply due to the complexity of building the environment model, which includes not only physical interaction between the robot and environment, but also social interaction between the robot and other human agents.

To make the autonomous vehicle be able to generate a plan considering the interaction between itself and the surrounding vehicles, Sadigh proposed a framework based on finite horizon optimization [38]. The human drivers are modeled to make decisions based on their own optimal controller given the robot's future action. And the robot tries to find the best plan based on the prediction using the human driver model. However, the uncertainty of human drivers behavior is not considered. This approach has the strength that they model the decision making nicely as an optimization problem. Therefore powerful tools such as dynamic programming or non-linear

optimization could be used. However, due to the modeling complexity, to make the problem solvable, strong assumptions need to be made by either very rough discretizing the action space or there is no uncertainty of human drivers behaviors.

2.9 Summary

In the last few decades, there have been considerable improvements in autonomous driving technology. Recent research and experiments have achieved promising results in testing autonomous driving research platforms on public roads. There are also some off-the-shelf driver assist systems that can help human drivers control the vehicle in certain situations. However, one of the main barriers to the application of fully autonomous driving is still the level of intelligence of the autonomous vehicle.

To achieve high-level intelligence on the road, autonomous vehicles need to behave more like good and experienced human drivers. Research indicates that a set of properly tuned cost functions can be used to robustly emulate human decision making for static environments. But further work is needed to accommodate the dynamic interaction between autonomous vehicles and surrounding traffic.

To improve the performance of cost function-based algorithms for autonomous driving, proper prediction models of surrounding vehicles are required. Previous research shows that prediction of human drivers' behaviors based on understanding their intentions will lead to more accurate prediction results. However, there is no complete integration of statistical human intention estimation, human behavior prediction and decision making yet implemented for autonomous vehicles.

One of the major differences between autonomous driving and human driving is the human drivers' capability to socially interact with surrounding vehicles by understanding and predicting surrounding vehicles' behaviors. Recent research shows promising performance with respect to human acceptance by applying social navigation algorithms to the indoor robot-pedestrian cooperation and simulated intersection domains. However, due to computational complexity and some over-simplified assumptions in these algorithms, they cannot be applied directly to

autonomous driving systems.

Last but not least, the autonomous vehicle decision making module needs to be properly interfaced with its planning and controllers. The whole system should be able to deal with varying road geometry, moving and static obstacle situations and traffic rules in real-time. Previous autonomous driving platforms are either optimized for a certain road condition or sacrifice driving performance to adapt to a wide variety of driving situations. A better autonomous vehicle decision making scheme needs to be developed to balance the efficiency of decision making with the system complexity so that it is can be practically applied to autonomous cars.

Chapter 3

Definition of Autonomous Robot Social Behavior

3.1 Problem Statement

As summarized in Section 2.9, this thesis sets out to solve the following problems in state-of-the-art autonomous driving systems

1. The importance of autonomous vehicle social cooperation with human-driven vehicles has not yet been recognized by most of the state-of-the-art autonomous driving systems. The lack of social behavior reduces the efficiency, comfort and safety of autonomous vehicles for everyday driving in situations where social cooperation is necessary, such as entrance ramps, handling cutting-in vehicles, etc. In this thesis, the autonomous vehicle social behavior problem and domain are defined, and the importance of this problem is illustrated.
2. Current autonomous vehicles do not fully comprehend human drivers' intentions or predict their future behaviors. Therefore, they fail to perform socially cooperative behaviors as human drivers do naturally in their everyday driving.
3. An algorithm needs to be developed for autonomous vehicles to perform the desired human-like social behaviors. Such an algorithm should overcome two major shortcomings of previous cost function-based approaches:
    - Instead of assuming a static environment, the algorithm should perform decision

making in a dynamic environment where the effect of the host vehicle's behavior on surrounding vehicles needs to be considered.

- Instead of computing a cost function of one deterministic prediction result, the algorithm should utilize the probabilistically estimated intentions of surrounding vehicles to take the uncertainties of their behaviors into account.

Though autonomous vehicles are able to drive on-road and execute driving tasks, their performance is still not satisfactory in situations with relative high-density traffic and the need for cooperation with human drivers, such as handling cut-in vehicles, merging cars from entrance ramps, lane changes in heavy traffic, etc.

For example, if a human-driven vehicle wants to cut into the lane of an autonomous vehicle, before the actual merging, the human driver will notify the autonomous vehicle by using its turn signal or biasing away from the center of the lane. The human driver will expect some reaction from the autonomous driver to indicate whether its behavior is accepted. However, almost all autonomous vehicles will keep driving without considering the social reaction the cut-in vehicle desired. In contrast, an experienced and cooperative human driver will follow its leader more closely if it doesn't want the car to cut in; or it may decelerate and leave a more comfortable distance to allow the vehicle to cut in.

This kind of cooperation between human drivers happens frequently and naturally. There are no written policies or rules about how human drivers should cooperate with each other. But in general among the society of drivers, individuals are able to understand each other's intention and react to each other properly. This kind of cooperation is also an important factor that enhances transportation efficiency. Failure to perform social behavior cooperation causes confusion and leads to defensive driving by humans, such as slowing down or waiting, to avoid potential accidents with other drivers.

In physiology and sociology [3], social behavior is defined as:

> "behavior directed towards society, or taking place between, members of the same species"

Since autonomous vehicles and human-driven vehicles will co-exist on the road in the near future, both human drivers and autonomous pilots can be regarded as members of a "driver

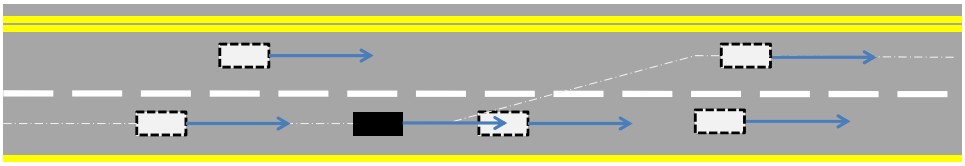


Figure 3.1: Lane Change Domain

society". In this thesis, we define the social behavior that occurs in the driving domain between drivers as "vehicle social behavior". We want to design and implement an algorithm in the autonomous vehicle, which allows it to perform cooperative "vehicle social behavior", so that the safety, comfort and traffic efficiency of the existing human driver society won't be degraded by autonomous vehicles.

3.2 Social Behavior Experiment Domains

In the automotive industry's road map towards fully autonomous vehicles, on-demand autonomy on freeways is the first step [6, 17]. Therefore, we choose this as the target application of social behavior algorithms. Lane change and entrance ramp handling are the two scenarios that autonomous vehicles need to handle in this application that may require some level of socially cooperative behaviors. The principles developed in this thesis will therefore be validated in these illustrative scenarios.

3.2.1 Lane Change

Lane changing is a very basic scenario the autonomous vehicle should be able to handle, as shown in Figure 3.1. Few of the current autonomous driving systems is able to perform smooth and cooperative lane change on public roads without human intervention in everyday traffic. Lane changing requires the autonomous vehicle to have a good understanding of surrounding vehicles' movements to make the right decision.

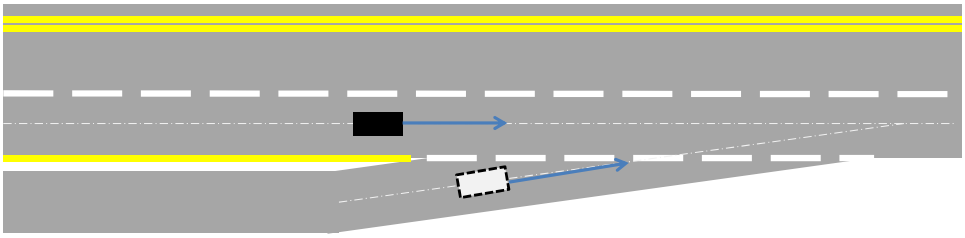


Figure 3.2: Freeway Entrance Ramp Domain

3.2.2 Entrance Ramp

Freeway entrance ramp management is chosen as a challenging scenario in which vehicles (autonomous and human-driven) need to exhibit social behavior. For the entrance ramp management system, a normal scenario is shown in Figure 3.2. The host vehicle is driving in the rightmost lane of the freeway. There will be a merging vehicle entering from the entrance ramp into the autonomous vehicle's lane. The autonomous vehicle needs to make proper speed adjustment to ensure a smooth and safe cooperation with the merging vehicle.

Chapter 4

Behavioral Planning Framework for Autonomous Vehicles

This thesis' approach to solve the described problem has two main parts. First, a novel autonomous vehicle framework with behavioral planning is designed and implemented. The autonomous vehicle's motion planner, controller and the proposed behavioral planner work together to handle a wide variety of road geometries, traffic situations, traffic rules and static obstacles on freeways. Compared with other autonomous driving software frameworks, the proposed framework efficiently constrains the search space for both on-road trajectory generation and high-level strategies of interacting with surrounding vehicles, and is hence implementable in real autonomous vehicle platforms.

4.1 Prior Autonomous Vehicle Software System Frameworks

To support the handling of a wide variety of scenarios for autonomous driving, the autonomous vehicle's intelligence system is usually of considerable complexity. Therefore, modularization is important for the autonomous vehicle hardware and software system design [43]. A hierarchical architecture is the most commonly used framework for the robot decision making and control problem [41]. It is important to understand the system framework so that the desired vehicle social behavior intelligence can be integrated into the proper system module.

In a hierarchical decision making framework, the higher-level goals are decomposed into sub-goals and executed by the lower-level controller. The hierarchical design simplifies the decision making process and makes some complicated decision making problems solvable by allowing the robot to make higher-level decisions in a heuristic way without taking into account the plan details, and to make lower-level decisions with the focus on achieving a local goal. The drawback is that the lack of detailed information while making high-level decisions may result in sub-optimality. Even when the high-level decision is problematic, it is infeasible for the lower-level controller to override the high-level decision.

In the DARPA Grand Challenge 2005, vehicles needed to drive in an unstructured environment. The Carnegie Mellon University vehicles used a top-down hierarchical architecture including pre-planning, terrain evaluation and path-shifting, obstacle detection, path tracking and control [50]. The major difficulty was to find a drivable path and avoid rocks and bushes along the road while the vehicle was driving at speed. In the system design of CMU's two Grand Challenge vehicles, there were two layers: terrain evaluation and obstacle detection. The Stanley team in the Grand Challenge used a more general system architecture with perception, planning and control, and a user interface [46]. Inside of the planning and control layer, a top-level mode selection module, path planner and a few lower-level controllers were implemented.

In the DARPA Urban Challenge in 2007, vehicles needed to avoid obstacles but also to follow traffic rules and interact with human drivers [8]. Therefore, a behavior layer was added in multiple teams' system frameworks in different forms to issue real-time planning goals that would also allow the car to follow traffic rules [30, 51]. Figure 4.1 shows the planning framework implemented in CMU's autonomous vehicle Boss for the DARPA Urban Challenge [51]. There were four primary subsystems. The perception system analyzed real-time data input from LIDAR, radar, GPS and other sensors. Mission planning optimized the path to achieve different checkpoints considering the arrival time, distance, or different required maneuvers. The behavior executive system made tactical driving decisions about such things as distance keeping, lane changing, and neighboring vehicle interactions. Motion planning generated the desired trajectory of the vehicle considering the dynamic parameters and output steering and throttle commands.

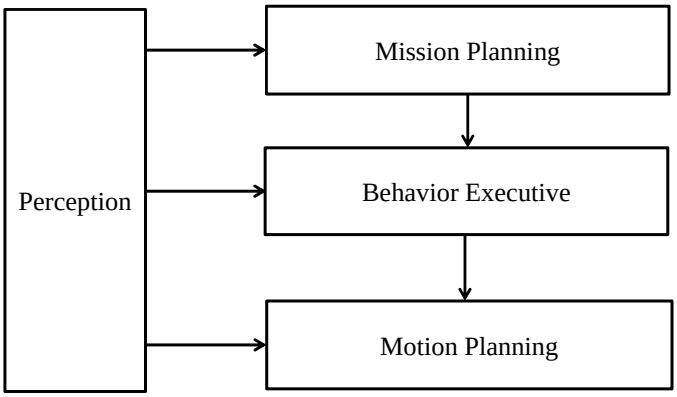


Figure 4.1: Software Framework of an Autonomous Vehicle

4.2 Autonomous Vehicle Software Framework with Behavioral Planning

Though a hierarchical architecture can successfully decompose a mission-level objective into local goals, there is a considerable performance decrease due to the lack of detailed information for the higher-level layers. For example, the behavior layer in the architecture of CMU's autonomous vehicle Boss can command the vehicle to perform a lane change when the leading vehicle is too slow. However, because of the level of abstraction of the world in this layer, it doesn't have enough information about the shape of the road. This may lead to a lane change command from the behavior layer on sharp turns, which is not human-like and usually undesirable.

To prevent such performance decrease, an improved autonomous vehicle architecture is proposed in this work. Figure 4.2 shows the planning framework of the proposed architecture. There are four main layers in this decision making system: mission planner, reference path planner, behavioral planner and motion planner. The mission planning takes charge of generating the best route from the current position of the vehicle to the destination at intersection-level. It outputs which turn or lane the vehicle should take at each intersection. Then a reference path planner will generate a reference path and speed for the autonomous vehicle with the consideration of road shape, speed limit and stop sign or traffic light state at intersections. In other words, if there are no static or dynamic obstacles such as cars, pedestrians, etc. on the road, the vehicle can just execute the reference path. The behavioral planner takes the reference path and static and dynamic

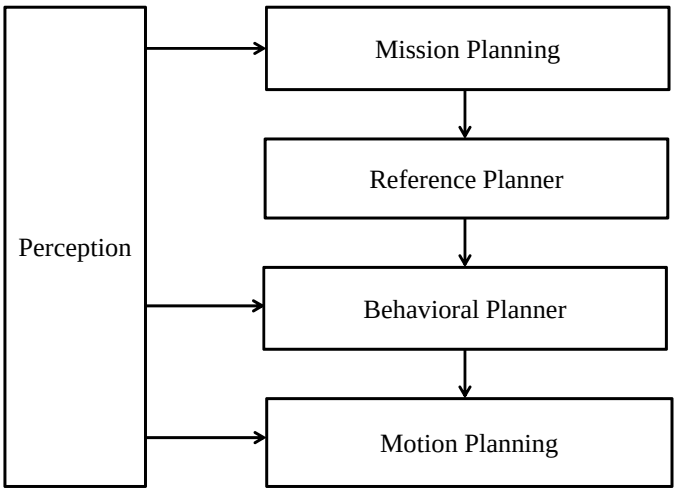


Figure 4.2: State-of-the-art Autonomous Vehicle Software Framework

obstacle information as inputs. It will try to find a set of path adjustment parameters that shift the reference path and change the reference speed so that the car can avoid obstacles properly and behave reasonably with other traffic agents on the road. The path adjustment parameters include a time profile of desired lateral bias $l_{0...t}$ and a profile of time headway parameters for distance keeping $th_{0...t}$, where t is length of the profile. The motion planner will then take the reference path and the path adjustment parameters generated by the behavior layer to reconstruct a final trajectory the car is going to execute. The path adjustment parameters will be used to generate a speed profile by forward-simulating an Adaptive Cruise Controller. Smaller th will lead to more aggressive distance keeping behavior that tends to get closer to its leader within a shorter amount of time. Using larger th, the vehicle will more conservatively approach the leader and keep further away from it. Adjusting l allows the vehicle to bias away from over-size vehicles in the neighboring lane to the host vehicle on the road. It also allows the vehicle to perform a lane change if the command of l is a full lane width.

There are multiple benefits of the proposed framework. First, the behavior layer works in a very abstract space, which is more computationally feasible for embedded computing platforms. Compared with other path planning approach, it only considers the potential path adjustment parameters rather than all trajectory details. This allows it to focus more on predicting surrounding vehicles' behaviors and socially cooperate with them. Second, the behavior layer has access to more detailed information about the road the robot is supposed to drive on, such as shape, speed

limit, future stop signs and traffic lights, etc. Taking these factors into account increases its capability of making correct and human-like decisions. Third, this architecture is also very adaptive to autonomous driving platforms which already have basic single-lane autonomous driving capability, such as General Motors' Super Cruise and Volkswagen's AutoPilot systems [17, 54]. Adding the behavioral planner will extend the capability of these systems to be able to avoid static obstacles and to perform lane change and other social behaviors with surrounding cars.

The proposed architecture minimizes the role of short-term motion planning in normal driving tasks. In situations without many unstructured static or dynamic obstacles to avoid, the behavioral planner is able to directly guide the lateral and longitudinal controllers of the vehicle to perform on-road driving tasks such as lane following, lane changing, etc. The motion planning layer, on the other hand, still acts as a high-fidelity collision checker and would respond to any situations that the behavior planning cannot handle, such as animals running cross the road, etc.

For the behavioral planner to coordinate the lateral controller and longitudinal controller of the vehicle to perform advanced social cooperative behaviors, it needs to forward-simulate the lateral and longitudinal controller models. The controller models are introduced here.

4.2.1 Lateral control

The lateral controller uses the lateral error, heading error and curvature of the detected lane marking or map-based road center line information. It is implemented based on a theoretical approach developed by Snider [42]. There are two path trackers being implemented for lateral control, a kinematic tracker and a dynamic tracker.

The kinematic tracker assumes no side-slip of vehicle tires that applies to arbitrary wheel angle. It outputs the desired wheel speed command $\dot{\delta}$, using a nonlinear feedback method described

by:

$$x_1 = s \tag{4.1}$$

$$x_2 = -\kappa'(s)e_{ra}tan(\theta_e) - \kappa(s)(1 - e_{ra}\kappa(s))\frac{1 + sin^2(\theta_e)}{cos^2(\theta_e)} + \frac{(1 - e_{ra}\kappa(s))^2 tan(\delta)}{Lcos^3(\theta_e)} \tag{4.2}$$

$$x_3 = (1 - e_{ra}\kappa(s))tan(\theta_e) \tag{4.3}$$

$$x_4 = e_{ra} \tag{4.4}$$

$$\dot{\delta} = \alpha_2(u_2 - \alpha_1 u_1) \tag{4.5}$$

$$\alpha_1 = \frac{\partial x_2}{\partial s} + \frac{\partial x_2}{\partial e_{ra}}(1 - e_{ra}\kappa(s))tan(\theta_e) + \frac{\partial x_2}{\partial theta_e}(\frac{tan(\delta)(1 - e_{ra}\kappa(s))}{Lcos(\theta_e)} - \kappa(s)) \tag{4.6}$$

$$\alpha_2 = \frac{Lcos^3(\theta_e)cos^2(\delta)}{(1 - e_{ra}\kappa(s))^2} \tag{4.7}$$

Here s is distance along the commanded reference path, $\kappa(s)$ is the curvature of the path, e_{ra} is the lateral error, θ_e is the heading error of the vehicle and δ is the current wheel angle of the car.

The dynamic tracker considers side-slip of the vehicle, but applies small-angle assumption for the wheel turning angle. The output, commanded steering angle δ_{cmd}, is determined by:

$$\delta_{cmd} = -Kx + \delta_{ff} \tag{4.8}$$

$$\delta_{ff} = \frac{m{v_x}^2}{RL}(\frac{l_r}{2c_{\alpha f}} - \frac{l_f}{2c_{\alpha r}} + \frac{l_f k_3}{2c_{\alpha r}}) + \frac{L}{R} - \frac{l_r}{R}k_3 \tag{4.9}$$

$$u_{dyn} = p_v(\delta_{cmd} - \delta_{host}) \tag{4.10}$$

Here x is the error state of the vehicle, K is the controller gain, δ_{ff} is the feedforward term, m is the mass of the vehicle, v_x is the longitudinal speed of the vehicle, R is the turning radius, L is the wheel base length, l_r is the distance from the center gravity to the rear differential, l_f is the distance from the center of gravity to the front differential, $c_{\alpha r}$ and $c_{\alpha f}$ are the rear and front tire cornering stiffness coeffients, u_{dyn} is the wheel velocity command output and p_v is the gain for tracking the desired wheel angle.

The kinematic tracker applies in urban environments, while the dynamic tracker applies mainly to highway driving. Equation 4.11 weights the output of these two controllers based

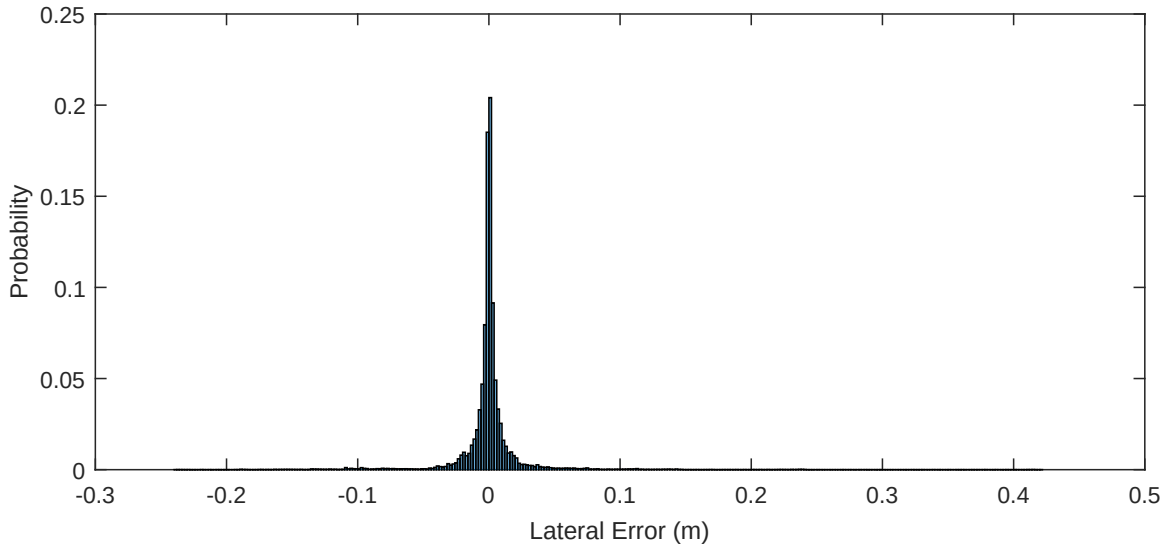


Figure 4.3: Lateral control error histogram during a 100-minute autonomous driving test

on the speed of the host vehicle.

$$k_{kin} = max(0, min(1, 0.05 * v_{host}) \tag{4.11}$$

$$u = k_{kin}u_{kin} + (1 - k_{kin})u_{dyn} \tag{4.12}$$

A statistical analysis over 100 min. of driving along different routes using an autonomous vehicle has been implemented to evaluate the performance of the tracker in lane changes and lane following. Figure 4.3 shows histogram of the lateral control error. In most cases, the lateral error can be constrained to less than 0.3m.

A more detailed road test analysis for the integrated kinematic and dynamic tracker is done on a course with both highway and urban driving, as shown in Figure 4.4. The performance of the integrated tracker is shown in Figure 4.5. It is seen that, there is a strong correlation between the lateral acceleration and lateral control error. The higher the lateral acceleration is, the bigger the lateral error would get. As long as the lateral acceleration amplitude is smaller than $0.225g$ (fast sharp turns), the controller is fully capable of keeping the vehicle in center of lane. Considering that the reference planner is able to slow down the vehicle in sharp turns, and that most of the discussed lane change or entrance ramp scenarios will happen when the required lateral acceleration is smaller than $0.1g$, response of the lateral controller can be ignored in the

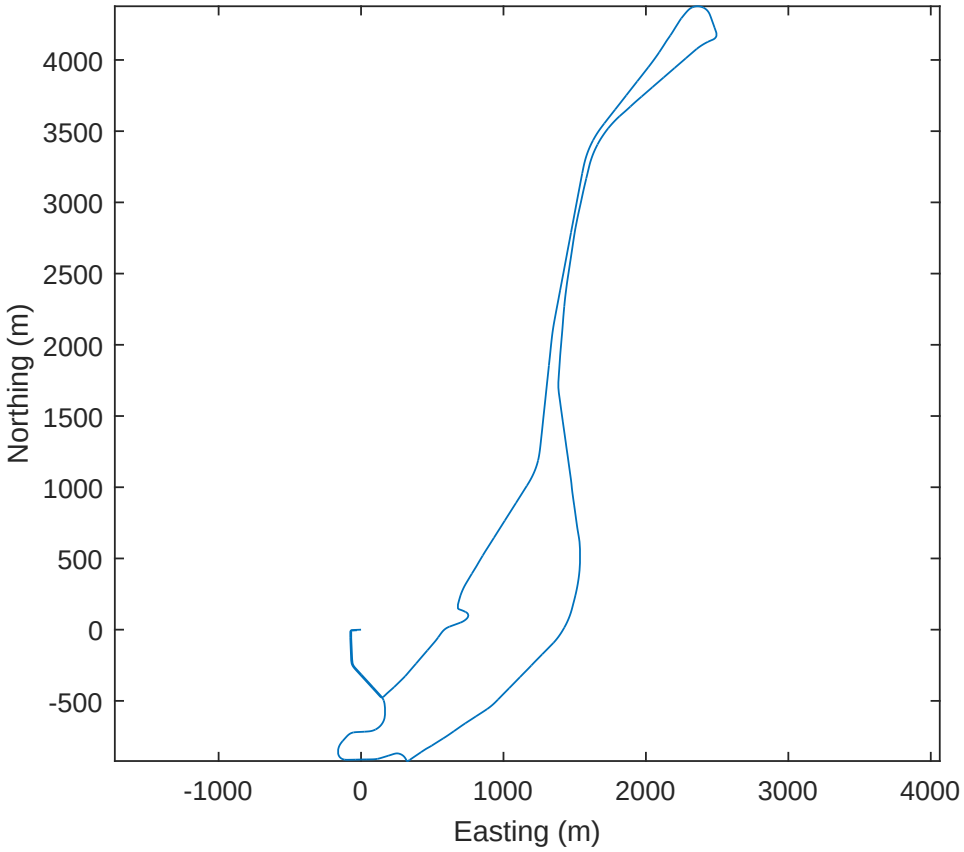


Figure 4.4: Test route for the integrated kinematic and dynamic tracker including low-speed sharp turns in an urban environment and higher speed on straight and curvy roads on the highway with some road bank change.

behavioral planning for the application scenarios that will discussed in this thesis.

4.2.2 Longitudinal control

The longitudinal controller governs both the speed of the host vehicle and its distance to the identified leading vehicle if there is one. If there is no leading vehicle, then a proportional controller is applied to generate the desired acceleration for the host vehicle to converge to the desired speed. The maximum allowed desired acceleration is shown in Figure 4.6, which is generated by observing human drivers speeding up when there are no obstacles ahead.

If there is a leading car, then a LQR controller is implemented to keep the vehicle at a desired

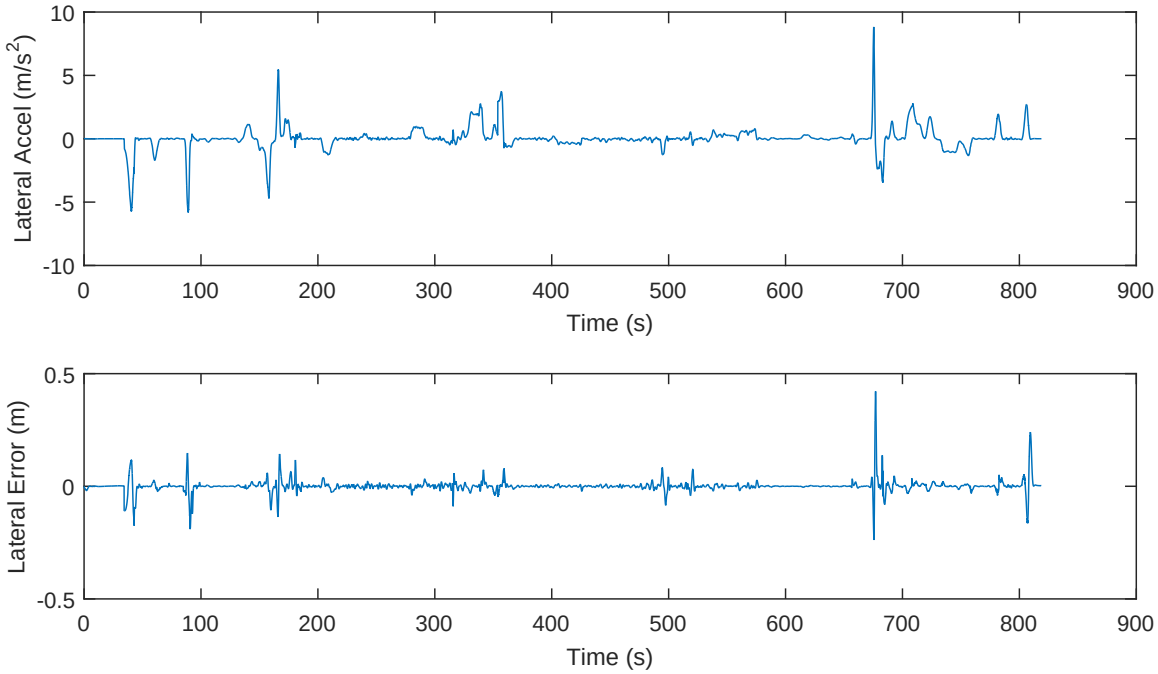


Figure 4.5: Lateral control error and lateral acceleration on the test route.

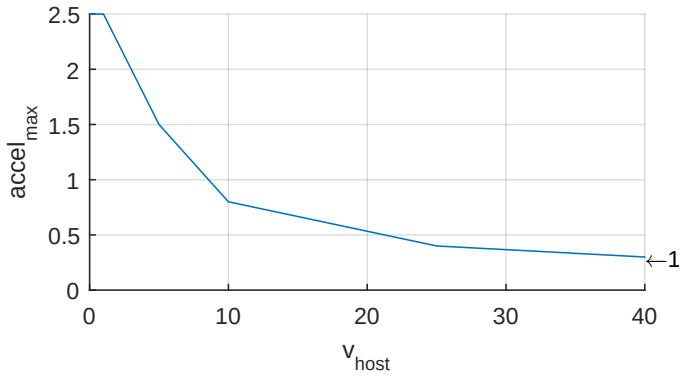


Figure 4.6: Maximum acceleration allowed for longitudinal controller at different vehicle speeds. It is generated from human driving data.

distance from the leading vehicle which is defined as:

$$d_{desired} = d_{min} + t_{hw} v_{lead} \tag{4.13}$$

$$a_l = k_d(d_l - d_{des}) + k_v(v_h - v_l) \tag{4.14}$$

$$a_l = min(a_l, a_{max}) \tag{4.15}$$

The LQR controller works well when the vehicle is around the range of steady state, which

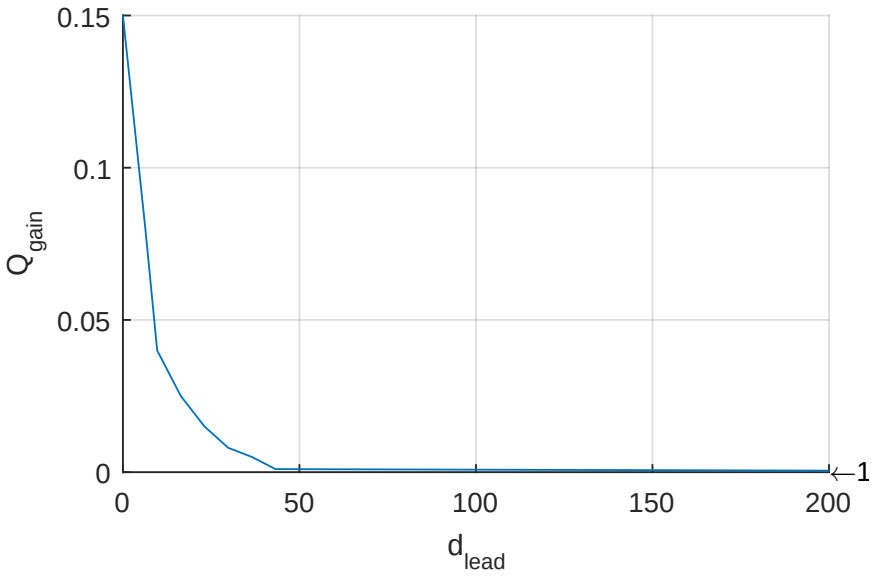


Figure 4.7: LQR gain table that makes the vehicle converge to the desired distance with gradual acceleration. The closer distance the host vehicle to its leading vehicle, the higher the gain is. This also improves the safety if the host vehicle gets too close to the leading vehicle.

means it is already following a leading vehicle at close to the desired distance. When the vehicle is still catching up with the leading vehicle, the LQR controller will tend to command a much bigger acceleration than necessary. To solve this problem, an LQR gain scheduling mechanism is designed. LQR gains are computed with different Q values at different distances away from the vehicle. Then the LQR acceleration is compared with the actual acceleration performed by a normal human driver. The final gain table is shown in Figure 4.7. The effect is that when the vehicle is further away from the leading vehicle, it doesn't weight the distance error as much as when it is getting close to achieving the desired distance to it.

Another special case that needs to be added to the standard LQR controller is the handling of cut-in vehicles. When a vehicle cuts in with similar speed but closer distance, for human drivers, it is natural to only slightly moderate speed and let the distance converge to the desired one. But due to the nature of LQR and the gain scheduling scheme, when a cut-in vehicle suddenly appears in the front, the host vehicle will command a large deceleration to keep the reference distance. To handle such scenario with more comfortable and human-like driving, a simple override rule is implemented by Equation 4.16, in which a_l is the longitudinal acceleration of the host vehicle, v_h is the velocity of the host vehicle, v_l is the velocity of the leading vehicle, d_l is the distance to

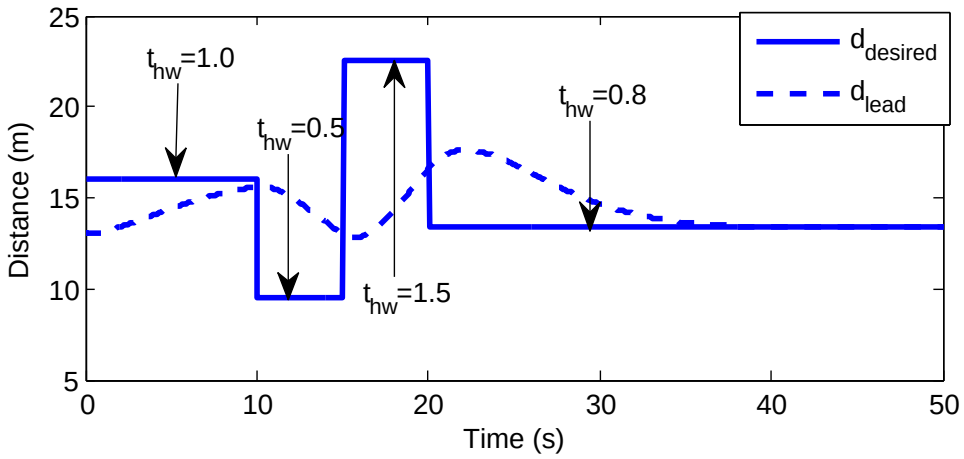


Figure 4.8: When the desired time headway t_{tw} changes, the implemented ACC algorithm of the test veihcle will accelerate and decelerate the vehicle to follow the commanded $d_{desired}$

the leading vehicle and d_{des} is the host vehicle's desired distance to its leader. This rule saturates the deceleration command when relative speed between host and cut-in vehicle is small, which indicates no threat is posed by the cut-in vehicle.

$$a_l = max(a_l, -0.7) \text{if } v_h - v_l < 0 \text{ and } d_l - d_{des} < 5 \tag{4.16}$$

Figures 4.8 and 4.9 show how the autonomous vehicle reacts when the reference time headway of the ACC controller get adjusted. In Figure 4.8, at the beginning, the host vehicle is targeting keeping a 1 seconds headway to its leading vehicle. The distance is converging to the desired distance. Then at $t = 10s$, the time headway t_{hw} changes to $0.5s$ and the host vehicle speeds up to catch up with the leading vehicle. At $t = 15s$, the time headway t_{hw} changes to $1.5s$ and the vehicle slows down to smoothly achieve the desired distance computed using t_{hw}. In the end, t_{hw} is set to $0.8s$ and the vehicle eventually converges to the desired distance using the LQR controller.

It is seen that by adjusting only the desired time headway, the autonomous vehicle is able to perform a wide range of velocity changes, which is necessary to create and merge into a gap for lane change.

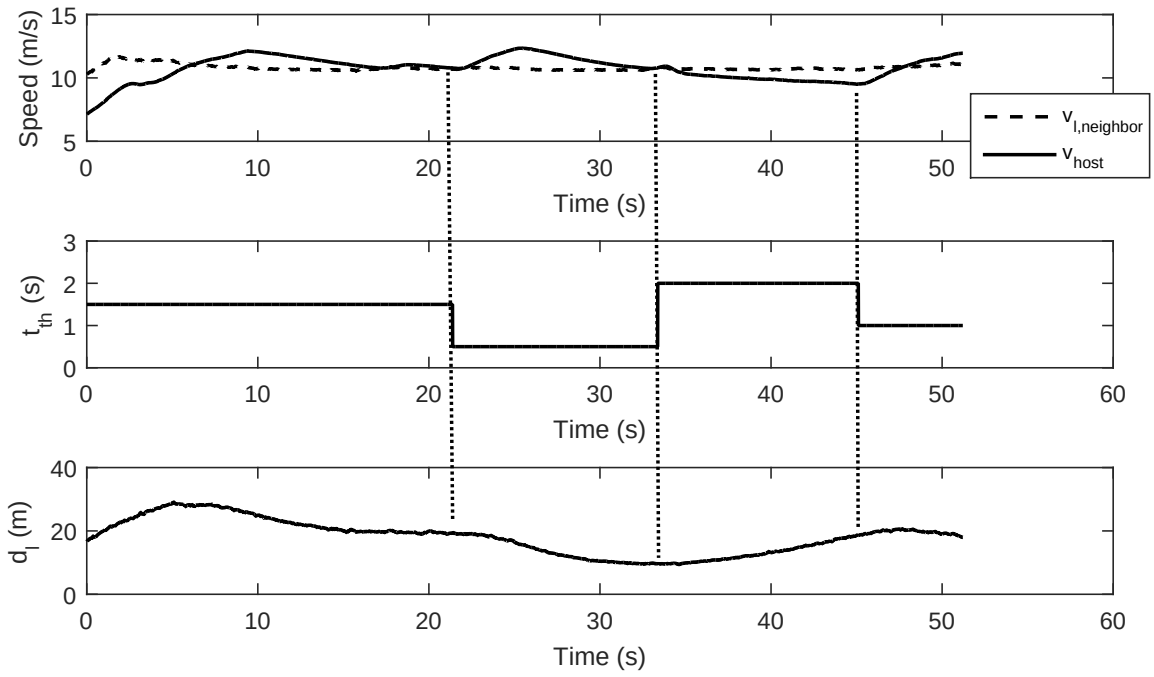


Figure 4.9: In on-road testing, the time headway t_{tw} has the same effect on the speed of the host vehicle and the distance to its leader. In the real vehicle, there is noise in executing the acceleration due to gear shifting and other powertrain-related delays and dead bands.

4.2.3 Summary

In this section, an autonomous vehicle software framework with the behavioral planning module is proposed. It improves the previous decomposition of the hierachical decision making framework. It allows the vehicle to perform adjustment of its speed and lateral position for potential social behaviors with the minimum increase of search space needed. The lateral and longitudinal controller are implemented and tested in an autonomous vehicle to show how different driving behavior can be generated using behavioral planning framework's time headway and lateral offset interface.

4.3 Design of Testing Platform

This section introduces the platform built by Carnegie Mellon University, that is designed for automated driving. This design has been implemented in multiple vehicles for this thesis's testing, including 2011 model year Cadillac SRX and 2015 Audi SQ5.

4.3.1 Actuation System for Drive-by-Wire

The stock car is not equipped with by-wire controls for operation. Several mechanical and electrical modifications were necessary to enable computer control of the required inputs that operate the vehicle. In order to support a full range of completely autonomous features, actuation and/or electronic control of the brake, throttle, steering, transmission shifting and secondary controls were implemented. This section provides a brief overview of each implementation.

Brake Actuation

The stock hydraulic brake assist system found in the SRX does not allow appropriate electronic control of braking for full-range autonomy. Actuation of the stock brake pedal assembly is achieved using an electric motor. The motor commutation is achieved via feedback from integrated Hall effect sensors and an internal incremental encoder. The incremental encoder provides feedback for braking rate control. The stock brake pedal position and fluid pressure sensors, as well as an auxiliary absolute angle sensor on the motor output, provide feedback for position control of the brake pedal. All feedback sensors are fused to produce optimal estimates of the braking system state and to provide redundancy/fault tolerance.

Throttle Actuation

Enabling computer control of the throttle does not require an additional actuator, since the throttle body of the stock SRX is already controlled electronically. The stock configuration uses accelerator pedal position sensors to generate input signals to the throttle body control module (throttle-by-wire). The modification that allows a computer to control the throttle simply simulates the pedal position sensors. A custom circuit board allows seamless switching between the pedal and computer inputs depending on the state of the Auto/Manual switch.

Steering Actuation

The stock vehicle uses a hydraulic power steering assist system without electronic control, so the steering column was modified to allow computer-controlled actuation of the steering system.

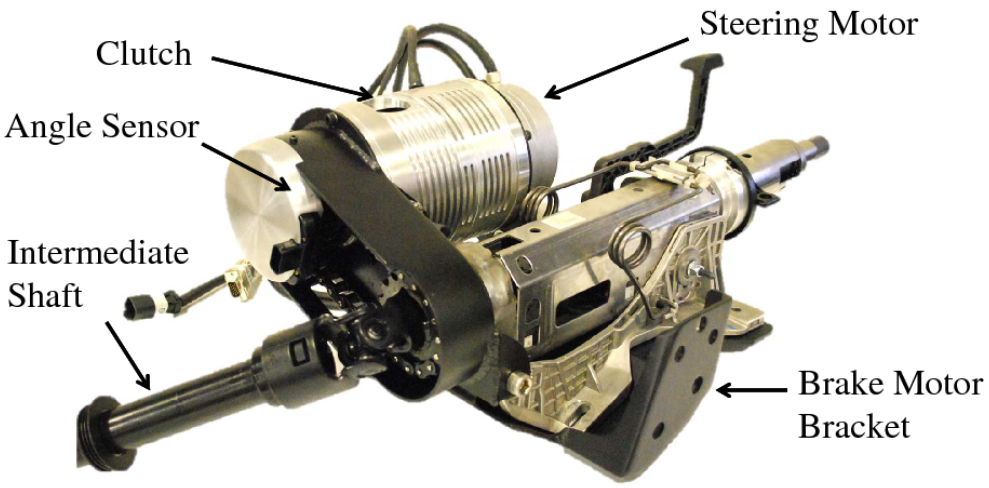


Figure 4.10: Modified Steering Column

The steering column modification is shown in Figure 4.10. Actuation of the steering system is achieved using an electric motor. The 3-phase motor commutation is achieved via feedback from the motor's internal incremental encoder, the stock steering wheel angle sensor and an auxiliary absolute angle sensor on the motor output.

An electromagnetic clutch serves as the mechanical interface between the steering actuator and the intermediate shaft of the steering system. The addition of the clutch provides two main features. First, it allows the steering wheel to be completely disengaged when the vehicle is switched to Manual mode. Second, it is sized such that the driver can easily override the steering actuation for manual takeover from Auto mode.

Gear Selection Actuation

Actuation of the transmission gear selector is achieved using an electric motor and linear actuator. The actuator is mechanically connected in line with the stock gear selector cable. The linear actuation is achieved by a coarse-pitch ball screw to allow the actuator to be back-driven with little force when not energized. The design keeps the original gear selector lever intact and operational for manual operation. A stock lever position sensor, as well as an auxiliary absolute linear position sensor on the actuator, provides feedback for position control of the transmission selector.

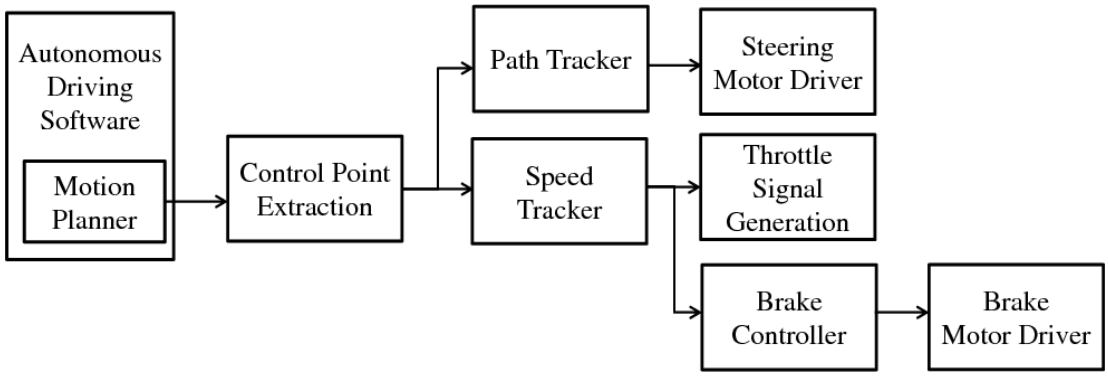


Figure 4.11: Block diagram of the Drive-by-Wire hierarchical trajectory tracking controller

Secondary controls

A variety of secondary controls are electrically modified to allow for computer control of turn signals, hazard lights, dim/bright headlights, door locks, ignition, and the horn. A custom circuit board is used to manipulate the signals of the secondary controls that are input into the body control module. The design allows for parallel operation of the controls by both computer and driver. The electrical configuration defaults to the stock version when the car is placed into Manual mode.

4.3.2 Overview of Drive-by-Wire System Control

Desired trajectories for the next 5-10 seconds are generated by the motion planning at 10Hz in the software system, including both the velocity and path information. The role of the Drive-by-Wire (DBW) system is to track the trajectory. The DBW-level controller is implemented in a dSPACE MicroAutoBox, which runs at 100Hz. A hierarchical controller is implemented to control the actuators to operate the autonomous vehicle to track the designated path, as shown in Figure 4.11.

The control point extraction module projects the vehicle's current pose onto the desired trajectory and outputs information for the controller, including cross-track error Δd, heading error Δh, desired curvature k, desired curvature rate dk and desired velocity $v_{desired}$. The velocity tracker consists of two Proportional-Integral (PI) controller outputs, the desired brake position and throttle percentage, which respectively minimize the error between current and desired vehicle velocities $v_{desired}$. The desired brake position is converted by a brake motor controller into the

instantaneous motor current command using a PI controller. A logic module switches between the brake and throttle with hysteresis to avoid operating them at the same time.

For the path tracking, a vehicle-kinematic-model-based controller is built to perform accurate lateral control when the vehicle is at low speed. It works very well even on very sharp corners (e.g. U-turns). This controller is also used to execute trajectories when the car moves in reverse.

In addition to the low-speed path controller, another controller is being implemented for high-speed path tracking with vehicle dynamics taken into account. It considers the side-drift of the vehicle and is able to control the car well in moderate-curvature turns. Both controllers output the steering wheel velocity. Smoothly merging these two controllers allows the lateral tracking error to be limited to within 30cm even for sharp turns and high-speed driving. The steering velocity command is then sent to the driver of the steering motor.

4.3.3 Power System

The autonomous vehicle actuation system usually needs more power than the stock vehicle electrical system can generate. Therefore, an auxiliary power system is installed in addition to the stock vehicle battery system, as shown in Figure 4.12. A high-output alternator which can generate at least 600W extra power when the engine is idle is installed to replace the stock alternator. A high-output alternator is installed to replace the stock alternator. A dedicated battery with a battery isolator is connected to the alternator. The isolation device prevents the risk of the auxiliary system affecting the stock car electrical system.

To generate a stable DC voltage, the power from the alternator is first inverted to AC, then converted back to DC. All of the AC power comes from an inverter/charger device. When the engine of the car is on, the device will invert the current from the alternator to AC. When the car stays in the garage with shore power plugged in, it works as a charger to charge the auxiliary battery and also directly route the AC power to the converters.

From the AC bus, two converters are installed to generate 12V (1000W peak) and 24V (1500W peak), respectively. Also, parts of the 24V bus are backed up, as shown in Figure 4.12. To support different voltages, maximum current, and redundancy requirements of the auxiliary autonomous driving system, six independent power busses are created, as shown in Table 4.1.

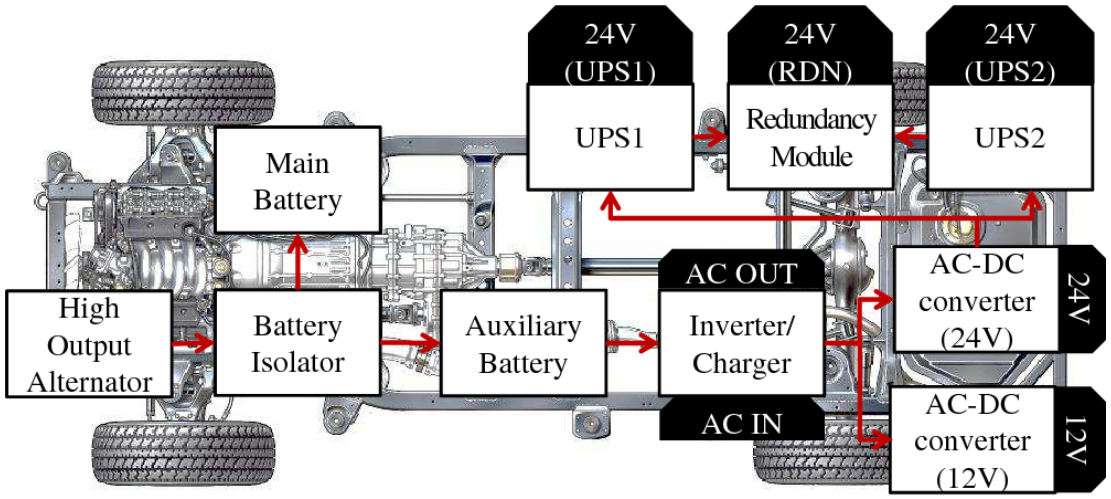


Figure 4.12: The power system design of the autonomous vehicle

Table 4.1: Power-bus Connection Diagram

Power Bus	Devices
12V	Computer system, Sensors, Localization system, Network devices
24V	CAN Gateways, RTK bridge
24V(UPS1)	Steering motor, Gear shifter
24V(UPS2)	Brake motor
24V(RDN)	dSPACE, Motor Controllers, Auto/Manual Switching
AC	Developer's laptop

Table 4.1 shows that the safety-critical DBW that controls the steering, brake and throttle is fully backed up on the 24V bus. Two 24V Uninterrupted Power Supply (UPS) UPS with 7.2AH capacity and 20A peak current are used to meet the requirements for peak current of the motor. In addition, a redundancy module merging the two backed up 24V busses ensures that even if one of the UPSs fails, the main controller of the DBW system will still be alive. The backed-up DBW subsystem ensures that even if the 24V converter fails, the auxiliary system can still keep the vehicle on the road for over 10 minutes while performing behaviors like notifying the human driver and trying to safely approach an emergency parking zone.

All of these power electronic devices are also equipped with a digital diagnostic port. Their health status is real-time-monitored by the autonomous driving system computer.

To install and better conceal the power electronic devices, a separation wall is built 6 inches behind the vehicle's rear seat. As shown in Figure 4.13, all power system components are housed

(a) Normal (b) Backseat folded down

Figure 4.13: The implementation of designed power system with consideration of space constraints, cooling and accessibility for debugging

in this concealed space and cooled by the cold air routed from the outlet for the rear seats. Without major modification to the vehicle, an isolated and backed-up power system is integrated into the stock vehicle.

4.3.4 Perception System

Localization system

High-performance localization is of great importance for fully autonomous driving. In the DARPA Urban Challenge, the Tartan Racing team proposed a pose filtering mechanism that combines the output of global localization and local lane marker features together [51]. Based on the pose filtering approach, a hybrid system consisting of both global and local localization was implemented. The global localization system focuses on determining the vehicle's absolute

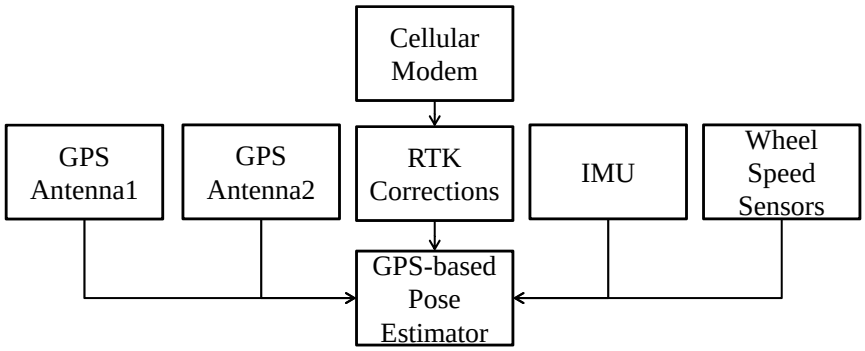


Figure 4.14: The design of the autonomous vehicle global localization system

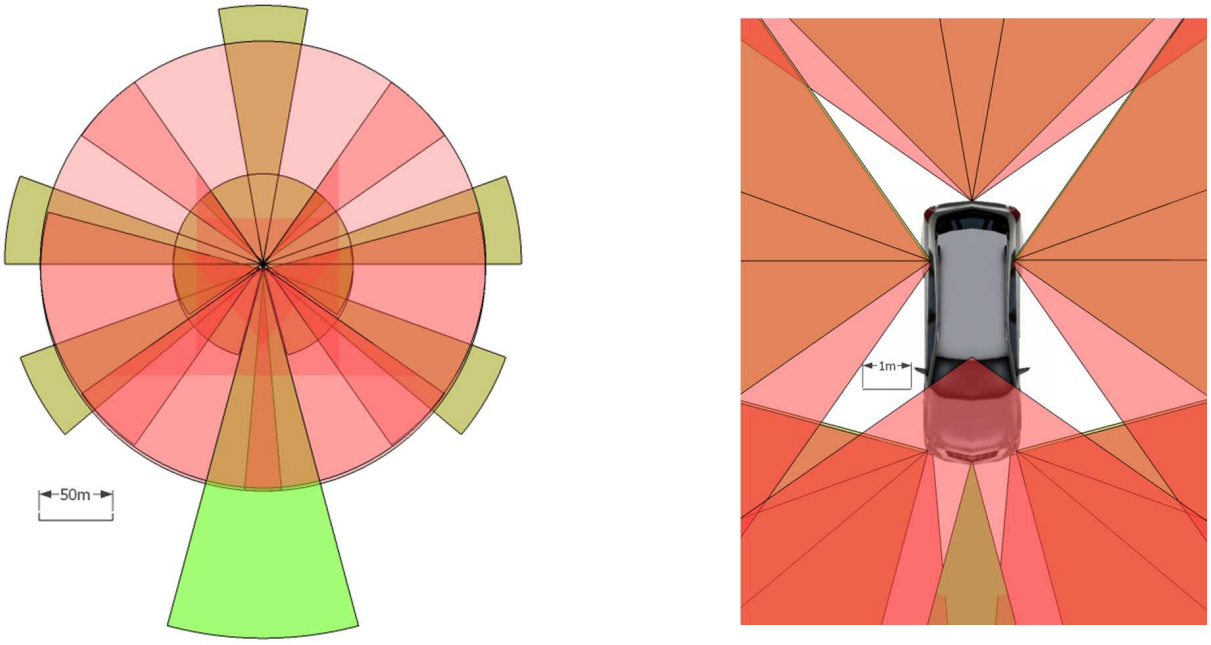
coordinates on the earth. GPS antennas, wheel speed sensors, IMU and additional cellular-transmitted RTK correction data are fused by a GPS pose estimator as shown in Figure 4.14. This system is able to localize the vehicle with an error as small as 1.5cm RMS when the cellular network is available and the GPS antenna can receive good signals. When the satellites' signals become unavailable for a short period of time, the IMU is still able to keep the localization error relatively small ($< 1.0m$) for a few minutes using dead reckoning.

The meter-level error from the global localization system is usually not good enough for the car to keep in the center of a lane autonomously. An additional few meters of error can be expected from the geo-information system's map. Therefore, a local localization system is implemented to compensate the error by using lane marking and road shape features to estimate the lateral and longitudinal distances between the vehicle and the lane center from the map. This information is fused with the output of the global localization to provide a higher-accuracy estimate of the vehicle pose. More importantly, the pose generated by this mechanism has an accurate relative position between the vehicle and the road, which enables the car to drive correctly in the middle of the road.

Environmental Perception System

In autonomous driving, vehicles need to make decisions based on the perception of the surrounding environment, including static and dynamic obstacles. The perception system performance will greatly affect the system's overall ability and robustness. There are three main types of sensors used by autonomous vehicle platforms: radar, LIDAR and cameras. Radar has been widely used in automotive applications for decades. It has the ability to detect obstacles' positions and speeds directly. However, radar outputs are usually not informative enough to estimate obstacle shape.

In the DARPA Challenges in 2005 and 2007, LIDAR sensors showed great potential for automotive applications. The data provided by LIDAR are usually a cloud of 3D points dense enough for the shape of cars, pedestrians, curbs, undrivable areas and more information to be extracted. However, LIDAR's cost, sensing range, robustness to weather conditions and velocity measurement accuracy are not as good as radar's. Cameras are the most cost-effective sensor



(a) (b)

Figure 4.15: Perception system coverage with all sensors integrated. a) Maximum ranges b) Sensor Field-of-View (FoV) and coverage close to the car

Table 4.2: Installed Sensors' Specifications

Sensor Type	Number	FoV (°)	MaxRange(m) / Resolution	Update Rate(Hz)
LIDAR	6	$85 - 110$	200	50
Radar1	1	$12 - 30$	250	12.5
Radar2	5	20(near) 18(far)	60(near) 175(far)	20
Video Camera	1	30	1360x1024	30
FLIR Camera	1	36	640x480	24

for automotive applications. Camera data are very informative, especially for detecting lane markings, signs and traffic lights with texture and color features. However, false-positive and recognition failure rates are much higher for vision than for LIDAR and radar.

Based on the features of these three sensor types, a hybrid perception system was designed to fuse the sensor data, exploiting their strengths and mutually compensating for their weaknesses. Table 4.2 shows the sensors used by the SRX and their main specifications.

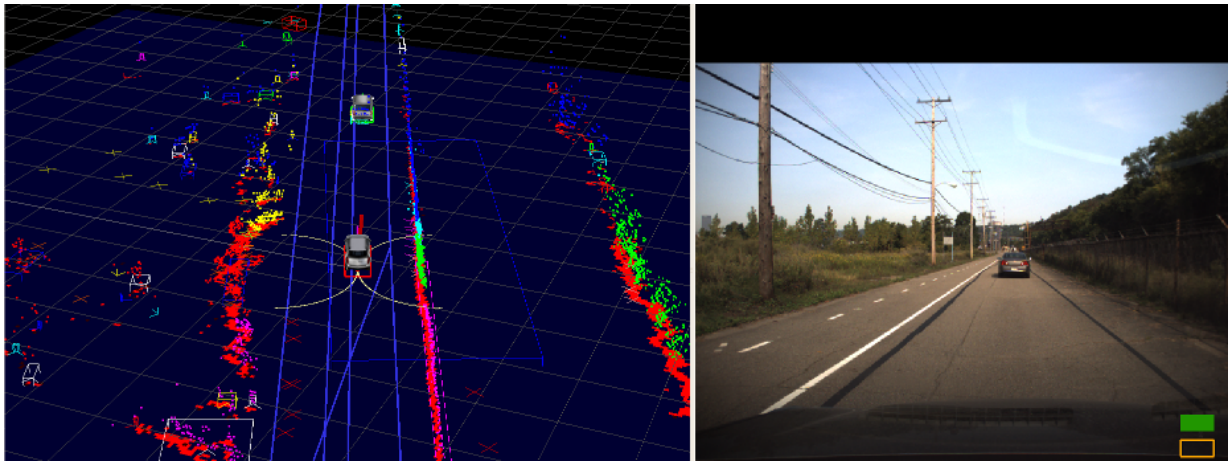


Figure 4.16: Data visualization of the integrated LIDARs, radars and camera

As shown in Figure 4.15(a), there are multiple LIDAR and radar sensors installed, giving the vehicle 360-degree coverage around the car. Based on this design, any obstacle within 200 meters will be covered by at least one type of sensor. Obstacles within 60 meters will be detected by at least two different types of sensors. This configuration maximizes the sensor's ability at long range. It also provides redundancy for obstacles close to the car, which are usually more important for the autonomous vehicle's decision making. The LIDAR & radar can also compensate for each other's shortcomings and minimize sensing failures for obstacles within 60 meters.

Figure 4.15(a) shows how multiple LIDARs and Radars are installed in pairs around the car, allowing them to see obstacles from the same point of view. This makes the LIDAR & radar data more consistent with each other and leads to higher perception performance. Cameras with vision-based recognition are used to assist the LIDAR/radar fusion system and also provide traffic light detection, work zone detection and help other assistance features.

In an autonomous vehicle sensor layout, blind spots can hardly be avoided. The blind spots for this platform are shown in Figure 4.15(b). Because of the integration of multiple wide-FOV sensors surrounding the car, the blind spots are small enough that no vehicle can be overlooked. Assisted by the tracking algorithm of the perception system, the actual effect of the blind spots can be further reduced.

Vehicular Communication as a Sensor

Autonomous vehicles require various sensors such as LIDAR, radar, cameras, etc. to understand their surroundings; however, such sensors can be easily obstructed by nearby obstacles, preventing long-range detection. This can be improved by using sensor data from other vehicles equipped with a vehicular communication device. Our platform uses a DSRC (Dedicated Short-Range Communications) modem to communicate with other similarly equipped vehicles. It can also talk to any infrastructure equipped with a DSRC capability. As a proof-of-concept implementation, our platform can retrieve detailed traffic light information via DSRC from DSRC-equipped traffic lights.

Sensor installation

For both safety and appearance reasons, the perception system should be well concealed and integrated into the vehicle. Therefore, the described platform avoids additional exterior sensor mounting racks, which greatly change the look and aerodynamics of the car.

The final sensor installation is shown in Figure 4.18. It shows that even with 15 sensors installed on the vehicle, the appearance and driving performance of the vehicle are not much influenced. Compared with previous research vehicle platforms, which have larger-scale and

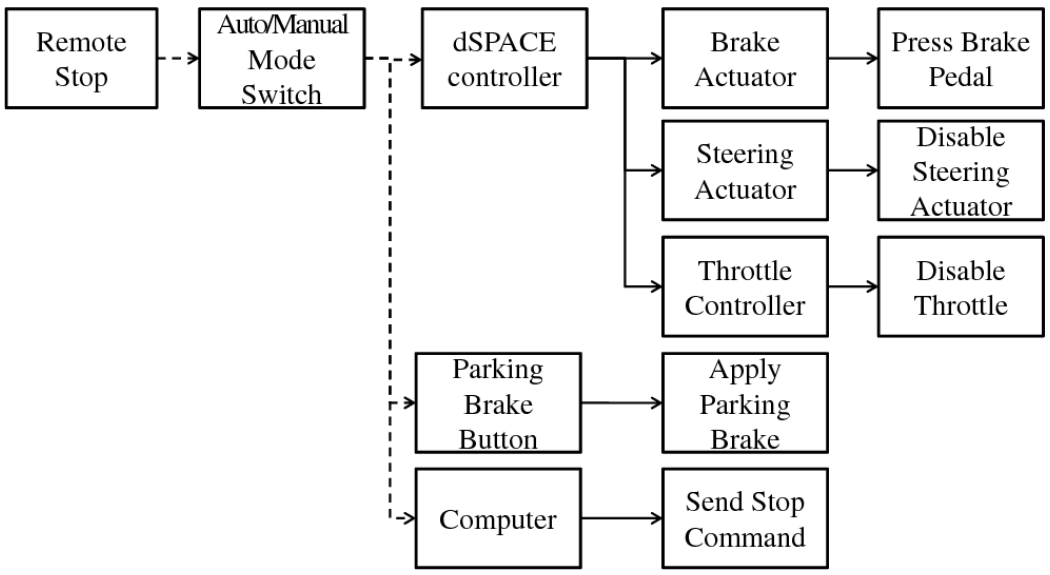


Figure 4.17: Remote stop mechanism of the platform

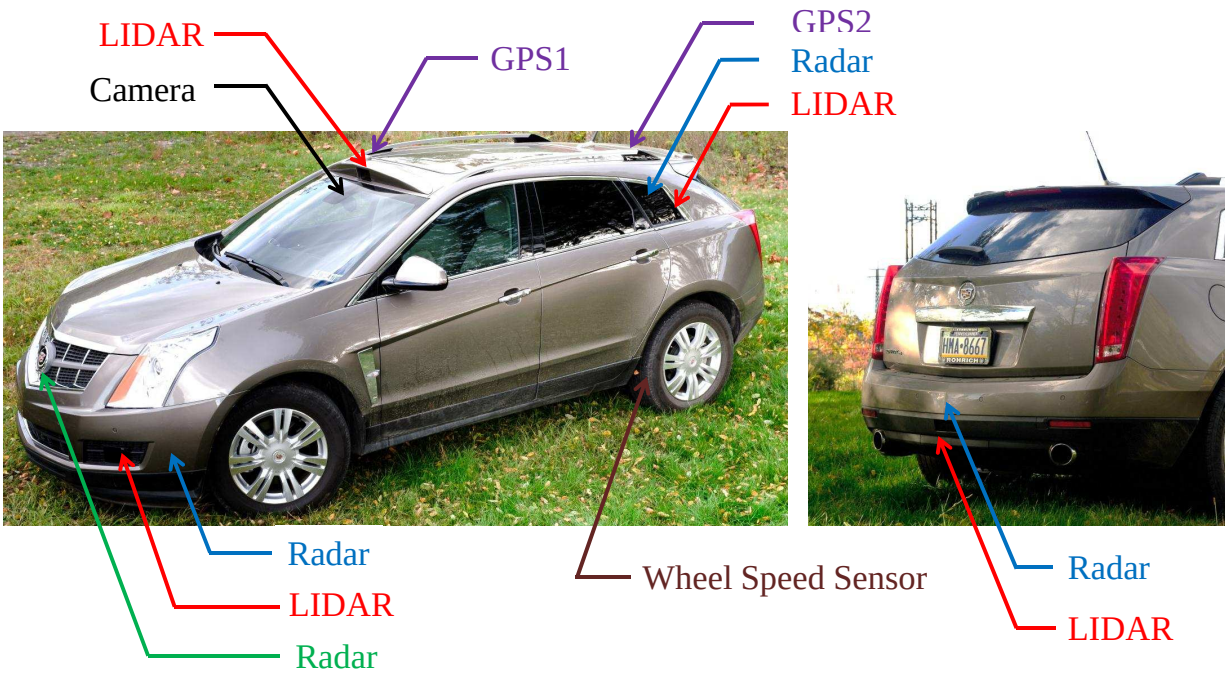


Figure 4.18: Sensor installation

non-automotive-grade sensors on top of the car, the proposed sensor configuration and installation options are more practical for future volume-produced autonomous vehicles. A typical data acquisition result of the LIDAR, radar and camera is shown in Figure 4.16.

4.3.5 Safety System

The vehicle platform hardware is equipped with a three-layer safety system: remote stop, auto/manual switch and emergency stop. The remote stop is used when there is no one in the car. Virtual valet is one target application, wherein the vehicle drops off users and then drives away to find a parking spot. As shown in Figure 4.17, when the remote stop is activated, the vehicle will disable both steering and throttle controllers, press the brake pedal and also apply the parking brake. This ensures that even if one braking mechanism fails, the car can still be commanded to stop.

The second layer is the auto/manual switching mechanism, shown in Figure 4.19. The most important feature of the switching module is that when the vehicle is in manual mode, the additional actuators are isolated from the stock vehicle through clutch, brake releasing and signal selection relays. Therefore, by disengaging the whole auxiliary system, the vehicle will become an unmodified car. The physical disengagement greatly improves driving safety when the plat-

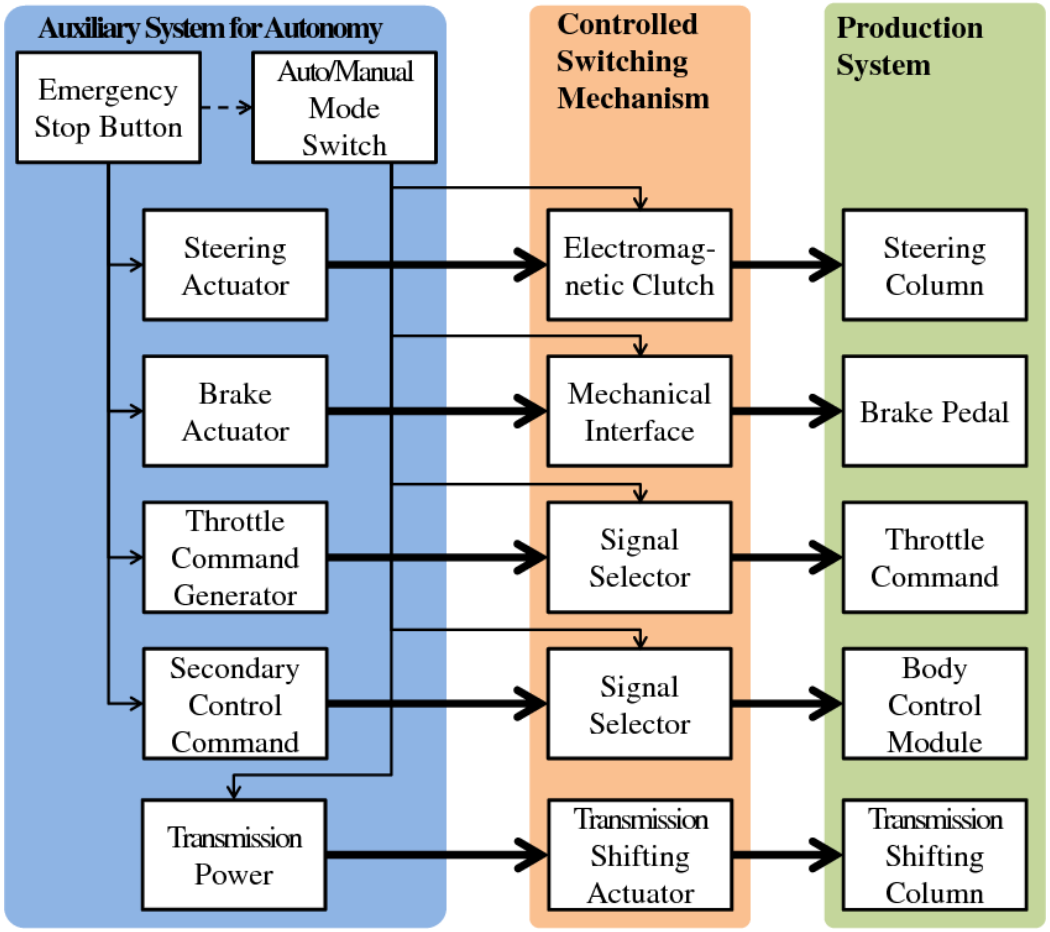


Figure 4.19: Switching mechanism between autonomous mode and manual driving mode

form is in manual mode.

The third layer, the emergency stop system, is designed to disable the auxiliary autonomous driving system entirely. The AC power from the inverter/converter is shut off by pressing the emergency stop button on the dashboard of the car, easily accessible. The backed-up power for the DBW system is also disconnected. This makes the vehicle platform return directly to manual mode, which is the nominal state. The three-layer safety subsystem is fully integrated and tested in the vehicle.

4.3.6 Computing Platform

Computer Configurations

The hardware components of the SRX have been carefully chosen to satisfy the following requirements: (i) the temperature of the computing hardware must remain under its maximum operating temperature even inside its small and closed space; (ii) the computing hardware must offer enough computation resources to support complicated intelligent algorithms; and (iii) the vehicle must tolerate a single point of failure of the computing system.

Since one of our high-level requirements is that the automated vehicle should look as *normal* as possible, we decided to use small-scale-factor computers, which can be installed underneath the carpet in the trunk area. Four Intel® Core™2 Extreme Processor QX9300s with mini-ITX motherboards are deployed in a custom chassis. Each motherboard is also equipped with a CUDA-compatible GPU, NVIDIA GT530, having 96 cores for accelerating computations. We also have a separate operator interface computing unit equipped with an Intel® Atom™Processor D525 to run a user application controlling the vehicle via an in-dash touch-screen interface. A tube extended from the car air conditioning system is directly attached to the custom chassis and pumps cold air into the computers.

(a) Normal

(b) With the cover is open

Figure 4.20: Computing hardware components are hidden in the spare tire compartment.

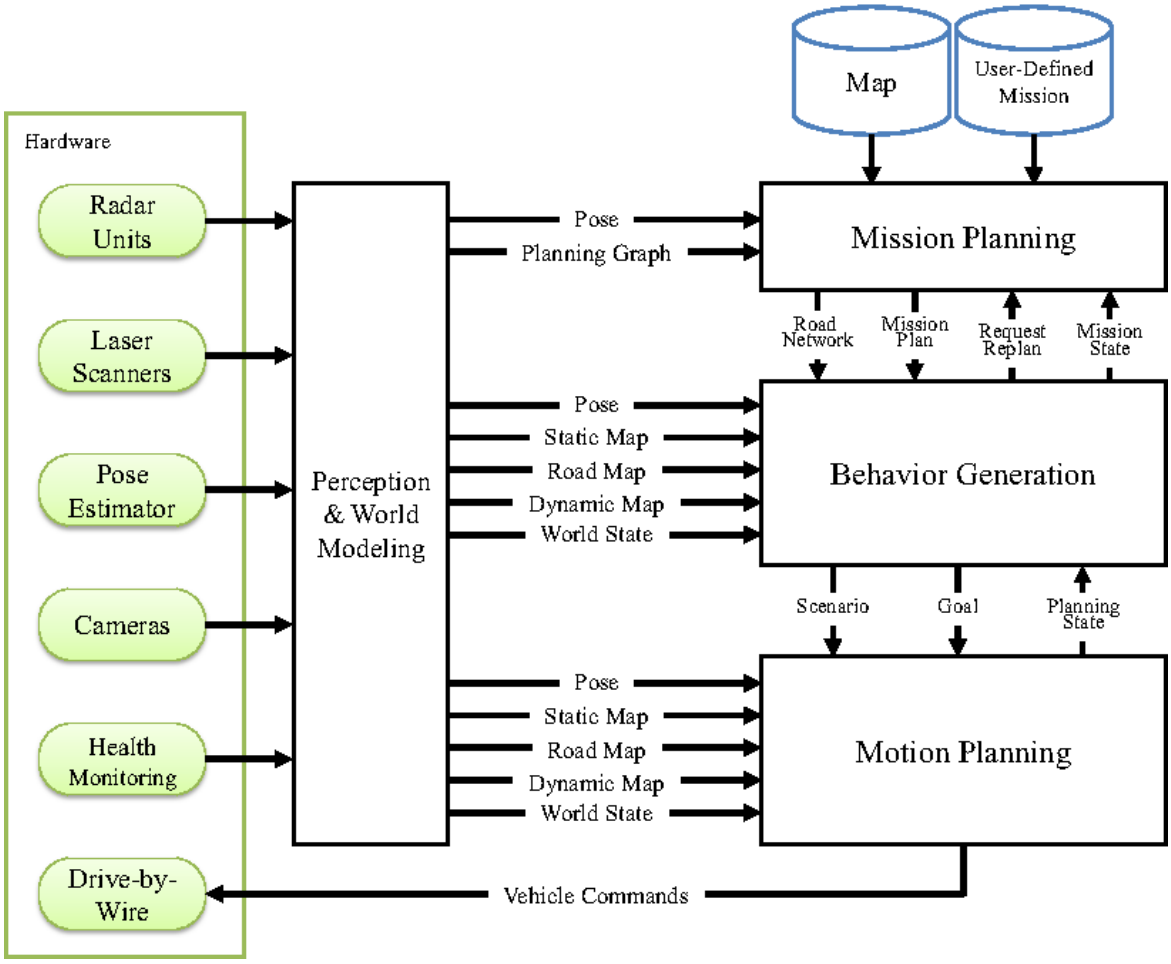


Figure 4.21: Software architecture of the autonomous vehicle

Software System

The software infrastructure for the SRX has been extended from the Tartan Racing Urban Challenge System (TRUCS) [48], which was designed to support the perception, behavior, planning, and other artificial intelligence components of an autonomous vehicle by providing an operator interface, inter-process communications system, process launching and management system, data logging system, configuration system, and task framework libraries.

On top of the software infrastructure described above, perception, behavior and planning algorithms will run. The perception subsystem fuses data from multiple sensors and provides a composite model of the world to the other subsystems. The behavior subsystem makes decisions to execute the output of the mission planner, which finds the optimal route to the next destination given the road network. The motion planning subsystem avoids static and dynamic obstacles. The abstracted software architecture is illustrated in Figure 4.21.

4.3.7 User Interface

The user interface has greater importance in an autonomous vehicle than in normal cars. In autonomous driving, both human and car have a certain level of intelligence and are able to make decisions based on their own understandings of the driving scenario. Therefore, without smooth communication between them, the human driver/passenger may easily feel uncomfortable or even panicked. The user interface should provide the human operator the ability to cooperate with the car at four levels.

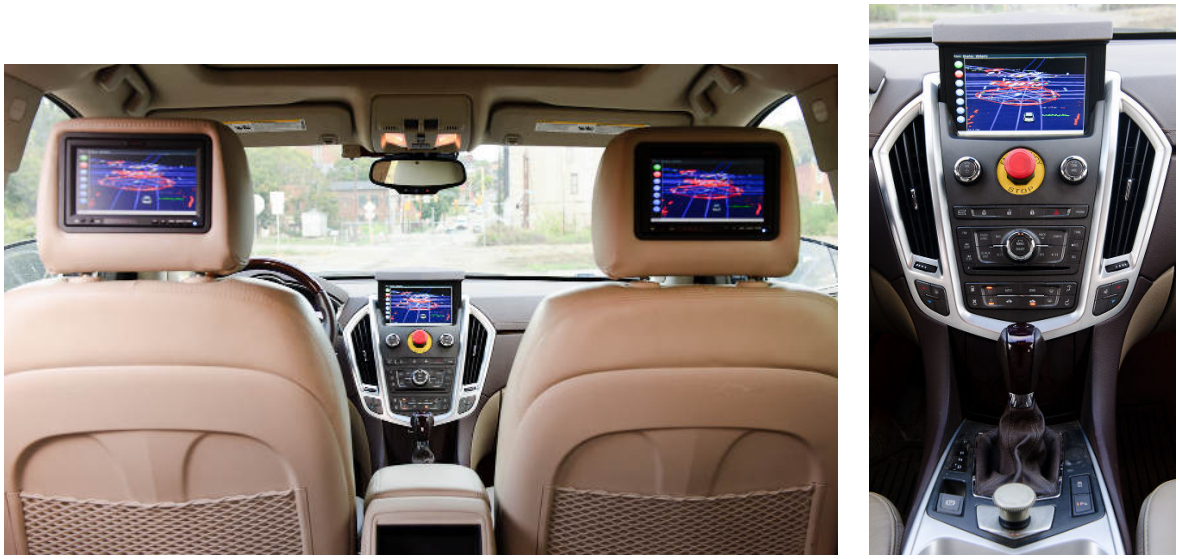
1. Mission level: Users should be able to input their destinations and select a preferred route;

2. Behavior level: Users should be able to adjust the autonomous driving preference (e.g. the preferred lane). The vehicle should generate notification or confirmation requests when it wants to perform certain behaviors like circumventions or U-turns. For the above two levels, a 7-inch touch screen integrated in the front dash of the vehicle (Figure 4.22b) is used as the main interface for the mission level, behavior-level communication and health status supervision. The vehicle's audio system is used for indication and notification.

3. Execution level: A safe switching mechanism allows the human driver to take over when desired. The auto/manual switching button described in Section 4.3.5 allows the user to switch between autonomous mode and manual mode by physically engaging or disengaging the auxiliary autonomous driving system.

4. Remote Access level: The vehicle should be accessible remotely for autonomous-parking and passenger pickup applications. A smart phone interface is implemented for such applications.

The vehicle is also equipped with a developer interface, including a dedicated diagnostic panel for the computing cluster, a special diagnostic widget on the main touch screen and two additional monitoring displays on the backs of the front-seat headrests (Figure 4.22a). These devices enable developers to perform debugging more efficiently.



(a) Car Interior (b) Center Console

Figure 4.22: User Interface

4.3.8 Autonomous Vehicle Intelligence

Based on the software framework developed by the Tartan Racing team of the DARPA Urban Challenge 2007, a variety of autonomous vehicle behaviors allow the car to deal with different kinds of driving scenarios.

Mission planning The vehicle is able to automatically generate a route from the user's current location to his desired destination. There can be penalties for low speed limit, number of traffic light / stop signs, number of lane changes, etc. to find the path with the soonest estimated time of arrival. After the destination is chosen, autonomous driving can be enabled by pushing the auto/manual switch button.

On-road driving On the road, the vehicle is able to perform lane-keeping and distance-keeping to slower traffic. It can perform a lane change if the vehicle needs to merge into another lane to exit the road or if the neighboring lane is faster [58]. The autonomous vehicle is also able to avoid on-road obstacles by either stopping if the road is blocked or circumventing if permitted, as shown in Figure 4.23.

Work zone behavior The autonomous vehicle can detect work zones. It notifies the human driver to be ready to take over and also slows down to the work zone speed limit. The vehicle

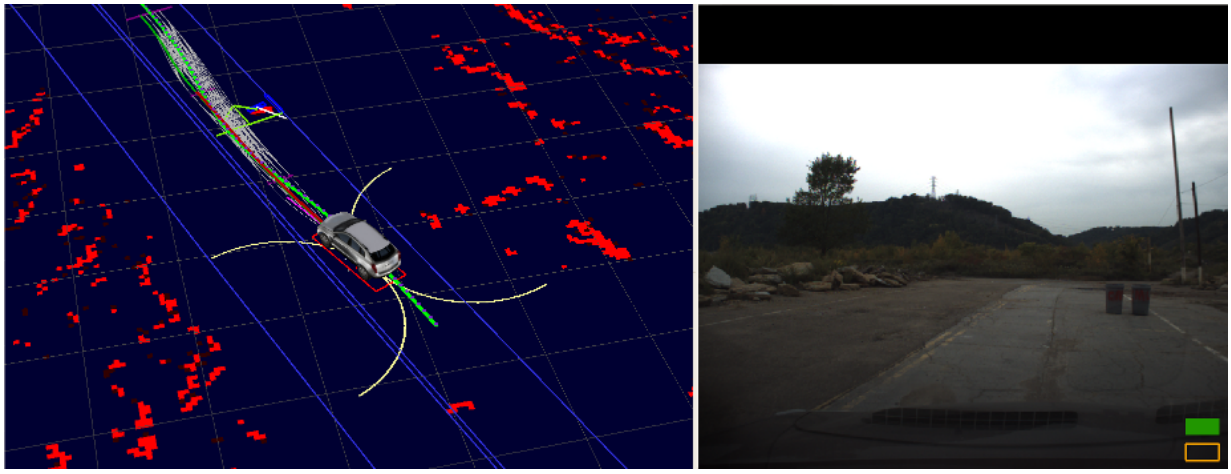


Figure 4.23: The autonomous vehicle slows down and drives around obstacles on the road

can also follow a shifted or alternate lane indicated by the work zone channelizers.

Pedestrian & Bicyclist Handling The vehicle is able to detect and stop for pedestrians crossing the road. If there is a bicyclist in front of the car, it will keep a safe distance and look for a safe opportunity to perform a lane change or circumvent.

Intersections Handling The vehicle is able to traverse intersections with both stop signs and traffic lights. For instance, for a four-way-stop intersection, it can estimate precedence and decide when it is its turn to proceed, as shown in Figure 4.24. For a traffic light-controlled intersection, the vehicle will follow the rules to stop on red and go on green.

4.3.9 Summary

A new, highly integrated autonomous vehicle has been developed and tested in a closed test field and public road for over five years. Experiments show that the platform design proposed in this paper is robust and easy for developers and end-users to operate. The vehicle is equipped with redundant sensors and has the ability to perform everyday driving while maintaining an appealing appearance. Most of the basic urban and highway driving features are developed on this vehicle. This system has been used for the autonomous vehicle social behavior algorithm and framework development.

Figure 4.24: The autonomous vehicle stops at a four-way-stop intersection, and waits for its turn to move

4.4 Baseline Performance of Rule-based Behavioral Planning Framework

To prove the ability of the proposed behavioral planning framework, a rule-based algorithm was developed in the behavioral planning layer to generate time headway th and lateral movement commands to handle entrance ramp and lane change cases. Tests have been conducted on public highways in autonomous driving mode.

The perception system of the vehicle is able to use LIDAR, radar and camera data to detect the vehicles ahead of the host vehicle in its current lane and neighboring lanes. The perception system is also able to detect vehicles behind and beside the host vehicle. However, the tracking and state estimation error in this case is larger than that of the front vehicles. Map information is also provided to the autonomous vehicle for better vehicle detection and association to the lanes.

4.4.1 Entrance Ramp

The most straightforward way to handle entering cars from an entrance ramp is to apply the same distance keeping or Adaptive Cruise Control (ACC) algorithm to the merging vehicles. There are also two special rules built in for the ACC algorithm. First, if the merging vehicle's estimated time of arrival to the merge point is greater than that of the host vehicle, it is ignored by the

distance keeping algorithm. This is because the host vehicle will be able to reach the merge point earlier, and the merging vehicle is assumed to yield. Second, the maximum deceleration the host vehicle can achieve for distance keeping is limited. This is because when the merging vehicle is still far from the merge point, there is no necessity for the host vehicle to slow down immediately to keep the reference distance behind it.

The commanded acceleration of the host vehicle is described in Equation 4.17, in which a_{cmd} is the output acceleration command, $a_{minMerge}$ is the allowed deceleration for a merging target in front of the host vehicle (set to $-0.7m/s^2$ in this test), $a_{cmd,merge}$ is the distance keeping acceleration command for the merging vehicle in front of us, and $a_{cmd,cur}$ is the acceleration command for keeping the desired distance to the host vehicle's leading target in its lane.

$$a_{cmd} = min(max(a_{minMerge}, a_{cmd,merge}), a_{cmd,cur}) \tag{4.17}$$

Figure 4.25 demonstrates how a car merging from a ramp is handled by the baseline algorithm. This process is shown by a sequence of screenshots taken from the implemented behavioral planner visualization interface. In each screenshot, the auto/manual mode of the vehicle is displayed at the top center of the screen, and the speed is displayed at the top right. The point cloud is displayed in yellow. The detected vehicles in the host lane and the merging ramp are drawn with solid and dotted arrows correspondingly.

As can be seen, the autonomous vehicle detects the merging car and applies the acceleration command computed using Equation 4.17 to keep distance to both the host lane and merging lane leading vehicles. This results in smooth yielding behavior to the merging car.

Another example in Figure 4.26 shows the results of the host vehicle overtaking the first merging vehicle because the later will arrive at the merge point later than the host vehicle. Then it starts to keep a distance to the second vehicle before it merges in.

4.4.2 Lane Change

Lane change handling also uses the th and lateral movement l interface between the behavioral planner and the lower-level planner/controller. There are three stages defined for the lane change

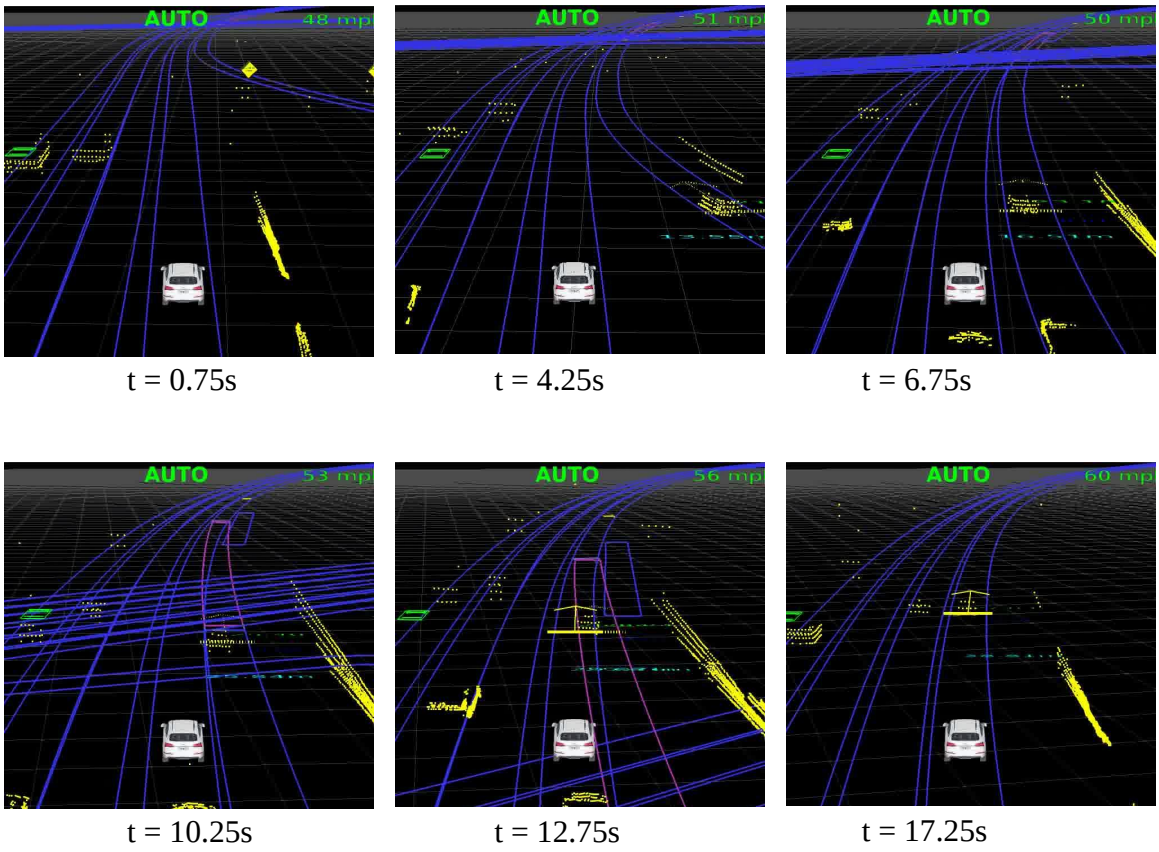


Figure 4.25: The host vehicle detects the merging car and decides to keep a safety distance to it while it is merging onto the main road. After the merging vehicle cuts in, the host vehicle follows it and speeds up.

process:

1. Speed Adjustment

   In the speed adjustment stage, the host vehicle not only keeps distance to the leading vehicle in the same lane, but also needs to consider the leading vehicle in the intended lane. This allows us to merge into the neighboring lane with slower traffic. In addition, it needs to merge into the spacing in between cars without interfering with the cars behind the host vehicle in the neighboring lane. Therefore the time headway for the leading vehicle in the intended lane is set to be shorter. The maximum deceleration is also constrained because the host vehicle does not necessarily need to brake hard for the leading vehicle in the intended lane, since it has the flexibility to select the next available gap.

2. Lane Changing

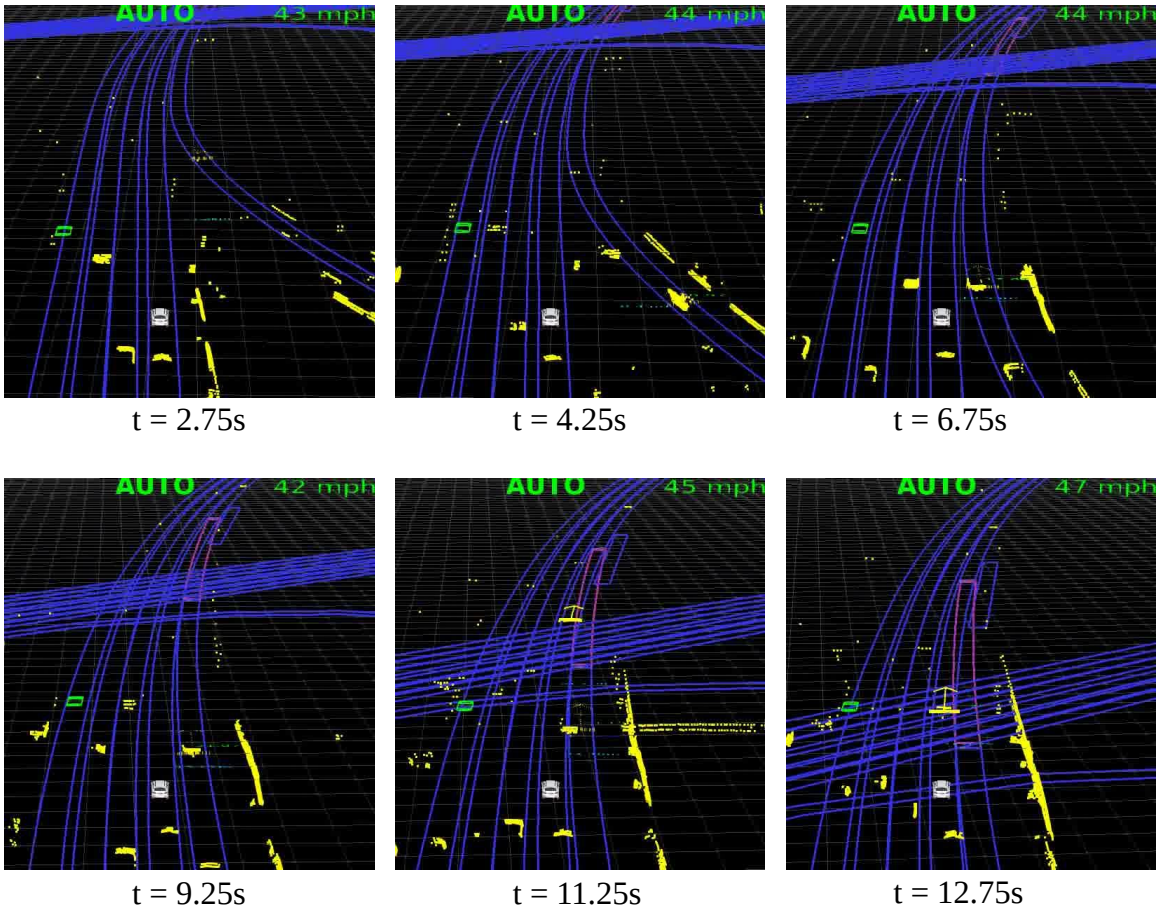


Figure 4.26: The host vehicle detected that the closest merging car at $t = 4.25$ will arrive at the merge point much later than it, therefore, decides to not slow down for it. At $t = 6.75$, it finds the next vehicle in the merging lane and slows down to merge behind the that vehicle.

After the speed and position of the vehicle in the intended lane indicate a safe merge, the vehicle will transition from speed adjustment into the lane change stage. During the lane changing process, the host vehicle is allowed to get closer to its leading vehicle while performing the lateral movement. Therefore time headway for the leading car in the current car is set to be shorter in this step.

3. Resuming distance

   After the host vehicle fully merges into the desired gap, it will resume the desired time headway smoothly and finish the whole lane change process.

Figure 4.27 visualizes the sequence of states using the implemented behavioral planner. The vehicle was commanded to initiate a lane change at the beginning of the scene. As shown in

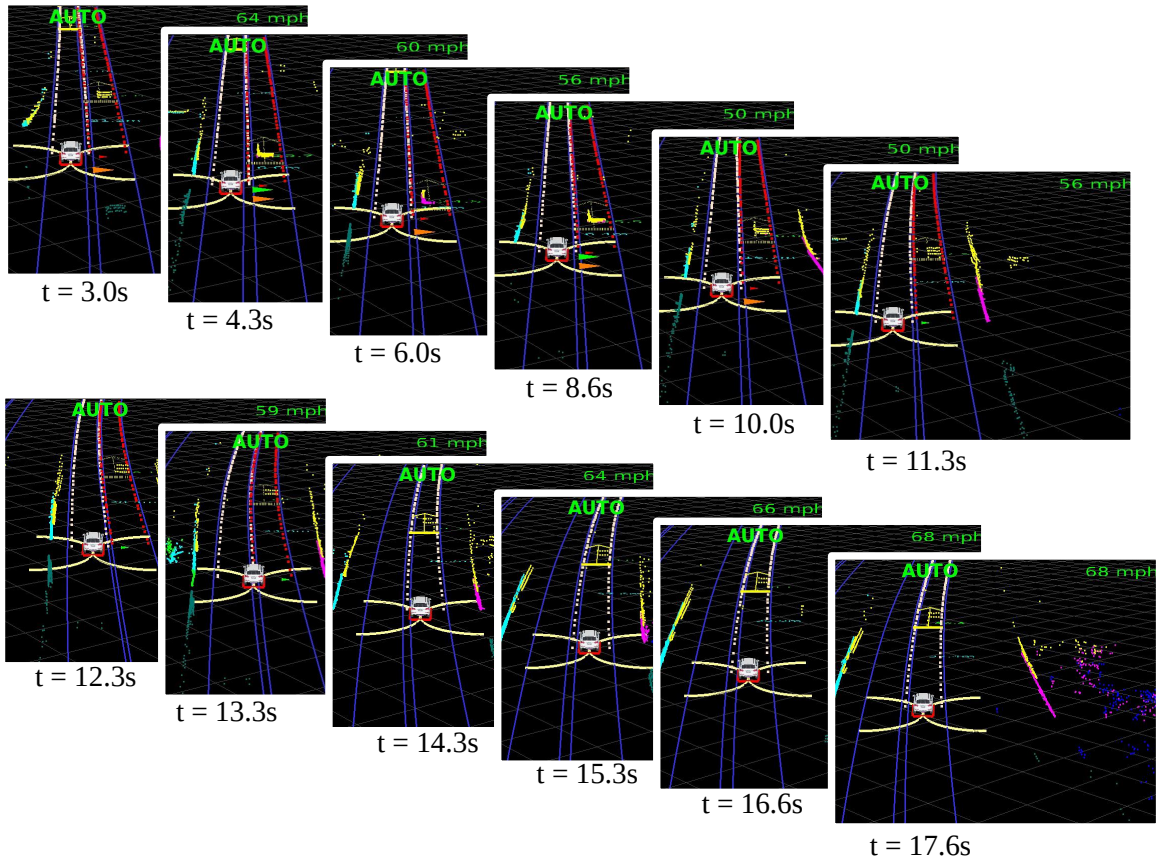


Figure 4.27: The host vehicle is performing distance keeping to the vehicle in its current lane when it receives the lane change command. It starts with speed adjustment from $t = 3.0$. It tracks cars in both its current lane and the desired lane and adapts speed to follow a slower-moving vehicle in the target lane. After the gap becomes sufficient around $t = 12.3$, the lateral movement starts to finish the lane change. Then the normal distance-keeping functionality resumes after the lane change is completed

Figure 4.27, it started to distance-keep to not only its immediate leader but also the car in the intended lane it wants to merge behind. Because the vehicle in the intended lane has lower speed, the host vehicle slows down from $64mph$ to $56mph$. After the host vehicle slows down and adjusts the speed and position to the car in the intended lane, it starts to move laterally and merge behind the car in the intended lane. Then it slowly resumes the normal distance-keeping distance.

Figure 4.28 shows another case using the rule-based scheme in lane changing. The host car first tries to adjust its speed to perform a lane change into the right lane. However, the immediate leading vehicle in the right lane is too slow. Therefore, the host car decelerates while passing the

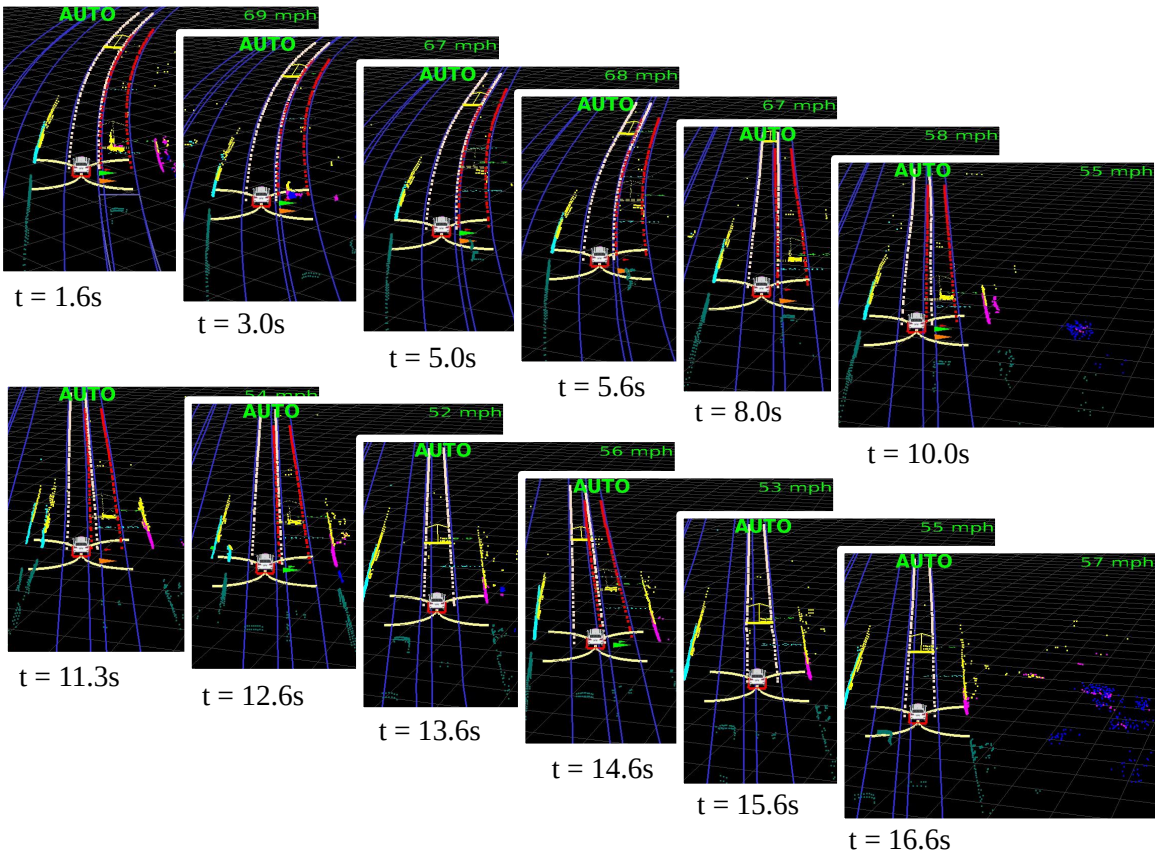


Figure 4.28: After the lane change command is received, the host vehicle first slows down to adjust its speed to match the vehicle in the intended lane from $69mph$ to around $55mph$. However, because the immediate leading vehicle in the intended lane is too slow, the deceleration command limits the host vehicle's braking and makes it pass the the first gap to search for the next opening. After finding the proper gap $t = 8.0s$, the host vehicle further slows down to adjust the position relative to the leading vehicle in the neighboring lane. Starting from $t = 11.3s$ the host vehicle finishes the lateral movement and resumes normal distance keeping to its leading vehicle in the new lane.

slowly moving car.

After the host vehicle passes the first car in the neighboring lane, it keeps slowing down for the second car it detects there. The host vehicle slows down further to 52mph in order to distance-keep to the vehicle in the neighboring lane, which also serves as an adjustment of its position to prepare for the merge. Then the host vehicle starts to move laterally.

4.5 Summary

160 lane changes and 313 entrance ramp merging cases were tested in light and average traffic conditions on a test route in Las Vegas. Through this on-road testing, we demonstrated the feasibility of integrating the behavioral planning framework into the vehicle. The longitudinal and lateral control work together, enabling the automated car to perform lane change and entrance ramp scenarios. Even though the rules are relatively simple, the underlying framework ensures the smoothness of driving in handling these cases without any computational expensive motion planning or speed profile optimization algorithm.

Though with the behavioral planning framework and developed logic, lane change and entrance ramp handling can be achieved in simple cases, the shortcomings of these algorithms are obvious. They do not have the capability to plan for a desired behavior. Therefore, if the surrounding vehicles' behavior deviates somewhat from the most standard situation, the rules will generate non-optimal behaviors, as shown in Figure 4.29 for lane change and Figure 4.30 and 4.31 for the entrance ramp case.

In Figure 4.29, the host vehicle performs speed adjustment, as shown in stage 1. However, after it believes it gets into the best position to perform the lane change, the car behind the host vehicle in the neighboring lane only slows down slightly for the host vehicle to merge in. As a human driver, we usually speed up to get closer to the intended lane leading vehicle and will leave a bigger gap and buffer for the trailing vehicle in the intended lane to react. However, using the rule-based approach, the host vehicle cannot plan for a different th. The host vehicle eventually finds a barely large enough gap for it to merge in between the two cars at a position that is not very human-like and uncomfortable for the trailing vehicle in the target lane.

Figure 4.30 shows a host vehicle accelerating before it detects a slowly moving merging car. The estimated time of arrival is just under the threshold for the host vehicle to ignore it. Therefore, the host vehicle still has to slow down for it slightly, which causes the speeding-up process to stop for a few seconds. Eventually, the merging vehicle slows down as expected and lets the host vehicle go first. A human driver would probably just keep accelerating in this situation, since the merging vehicle is not very fast and it has a yield sign in its lane. If the host vehicle accelerates, it will just get in front of the merging car, which is also very safe.

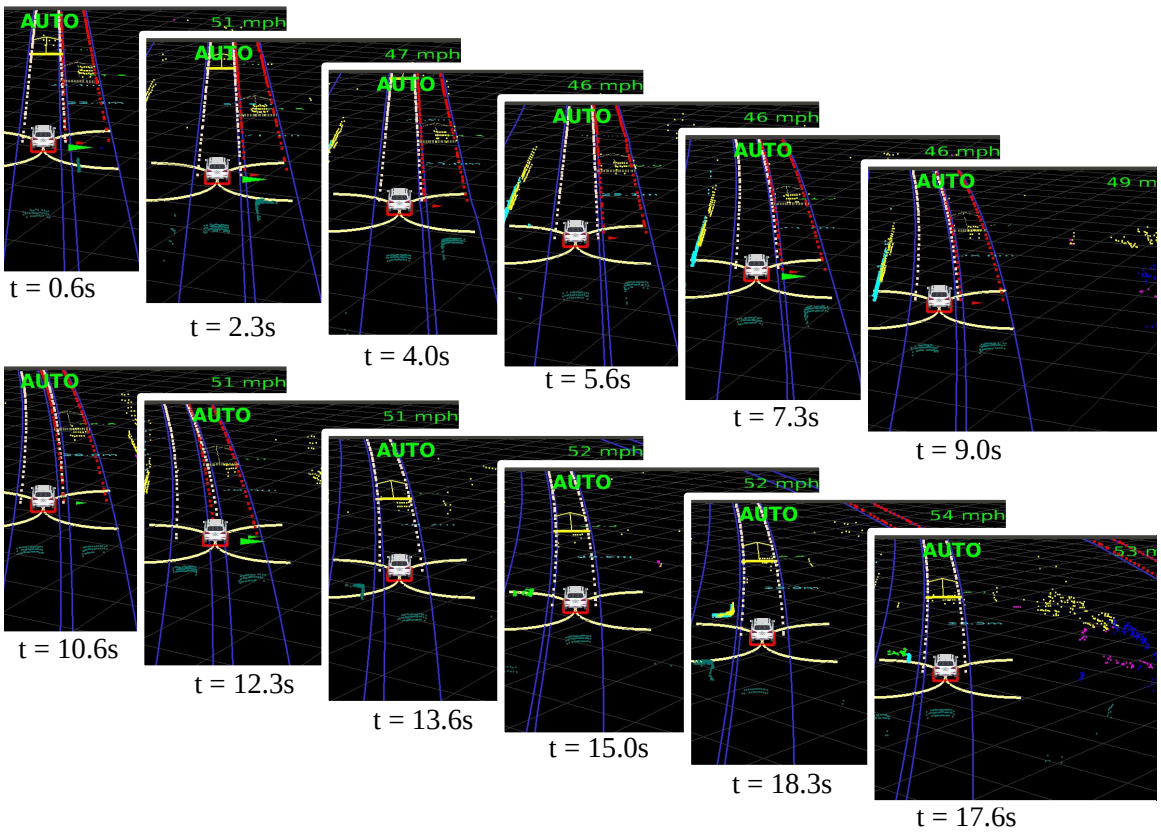


Figure 4.29: The host vehicle is adjusting its speed and position to prepare for a lane change. However, the vehicle behind the host vehicle in the intended lane did not give a big enough opening. The host vehicle has to wait at a spot that is further away from where the human driver would start the lane change, even though the trailing vehicle leaves enough gap for the host vehicle to merge into by accelerating closer to the leading vehicle in the intended lane.

In the second failure case example for the entrance ramp handling shown in Figure 4.31, the host vehicle fails to negotiate the cut-in space with the merging car. When the merging vehicle slows down, it also slows down. Therefore, the merging vehicle has to merge in very close to the end of the entrance ramp by applying large deceleration. A human driver might either slow down more and earlier to give it space to merge in front of the host car; or speed up to give the merging car a safe space to merge in behind us.

Even though the behavioral planning framework allows the host vehicle to handle the majority of the entrance ramp and lane change interaction cases with very minimum additional computing power, a few examples expose multiple shortcomings of the rule-based approach. A better algorithm to utilize the behavioral planning framework is therefore introduced and tested

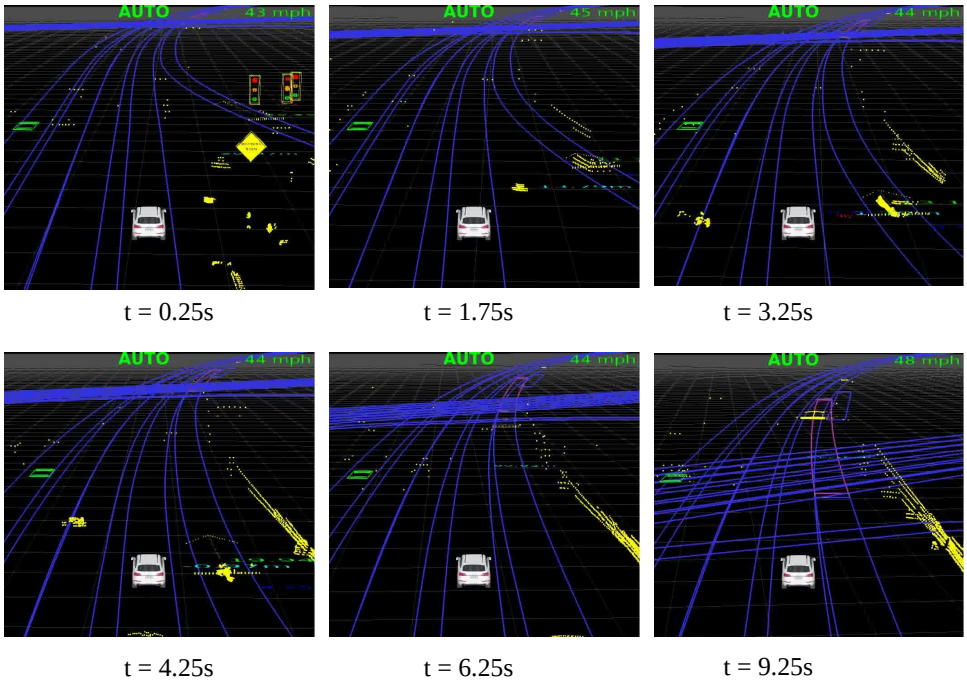


Figure 4.30: The host vehicle starts with no leading vehicle. It therefore tries to accelerate up to the speed limit, which is $70mph$. When it reaches the entrance ramp, it detects a merging vehicle. It computes the estimated time of arrival for both the merging vehicle and itself and decides to slow down for the merging car at $t = 1.75$ to $t = 4.25$. However, at the same time, the merging car slows down slightly too. After a few seconds around $t = 6.25s$, the host vehicle ignores the merging car based on the computed estimated time of arrival. However, the hesitation causes a pause in the host vehicle's speeding-up process, which is not optimal. A human driver might display aggressive behavior such that the host vehicle doesn't need to slow down at all.

in Chapter 5.

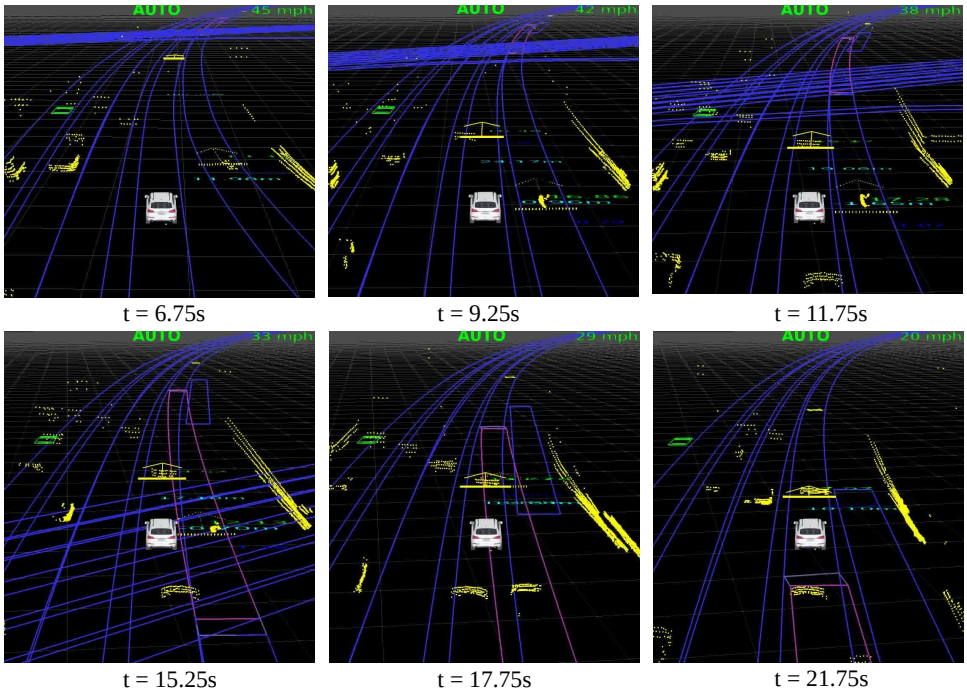


Figure 4.31: The host vehicle detects a merging car and decides to slow down for it. However, the merging vehicle slows down too. This causes the host vehicle and the merging vehicle to drive parallel to each other for a relatively long time from $t = 9.25$ to $t = 15.25$. Eventually, the merging car has to slow down more to merge behind the host vehicle. But it is already very close to the end of the merge. At $t = 17.75$, we can see that the vehicle behind the host vehicle has to move laterally to avoid collision with the merging vehicle. This potential critical situation is caused by the host vehicle not being able to indicate its intention earlier.

Chapter 5

Prediction- and Cost function-Based (PCB) Algorithm for Behavioral Planning

Some shortcomings of the rule-based approach to behavioral planning have been exposed in public road testing. To overcome the limitations of the rule-based approach, we develop a planning approach to generate the time headway for speed adjustment and find the best lateral velocity for entrance ramp and lane change, the two social behavior domains this thesis focuses on.

A Prediction- and Cost function-Based (PCB) algorithm architecture is proposed for the behavioral planning of the robot. There are three steps in the PCB algorithm: candidate strategy generation, prediction, and cost function-based evaluation, as shown in Figure 5.1.

In the candidate strategy generation step, a certain number of candidate strategies or potential output commands of the behavior planner are generated. Then the strategy set and the current moving obstacles set are sent to the prediction engine, which generates a series of simulated scenarios covering the next $10 - 20$ seconds. Exhaustive search is used to iterate through all candidate strategies and run prediction of all the traffic vehicles with the host vehicle executing each of the candidate strategies. The cost function-based evaluation module then computes the cost for each strategy, which is the sum of the cost of the series of predicted scenarios for the next $10 - 20$ seconds. The more desirable scenarios will have lower cost.

The benefit of using the PCB architecture is that it is able to consider surrounding vehicles' reactions to the host vehicle's movements. The more accurate prediction of the future traffic

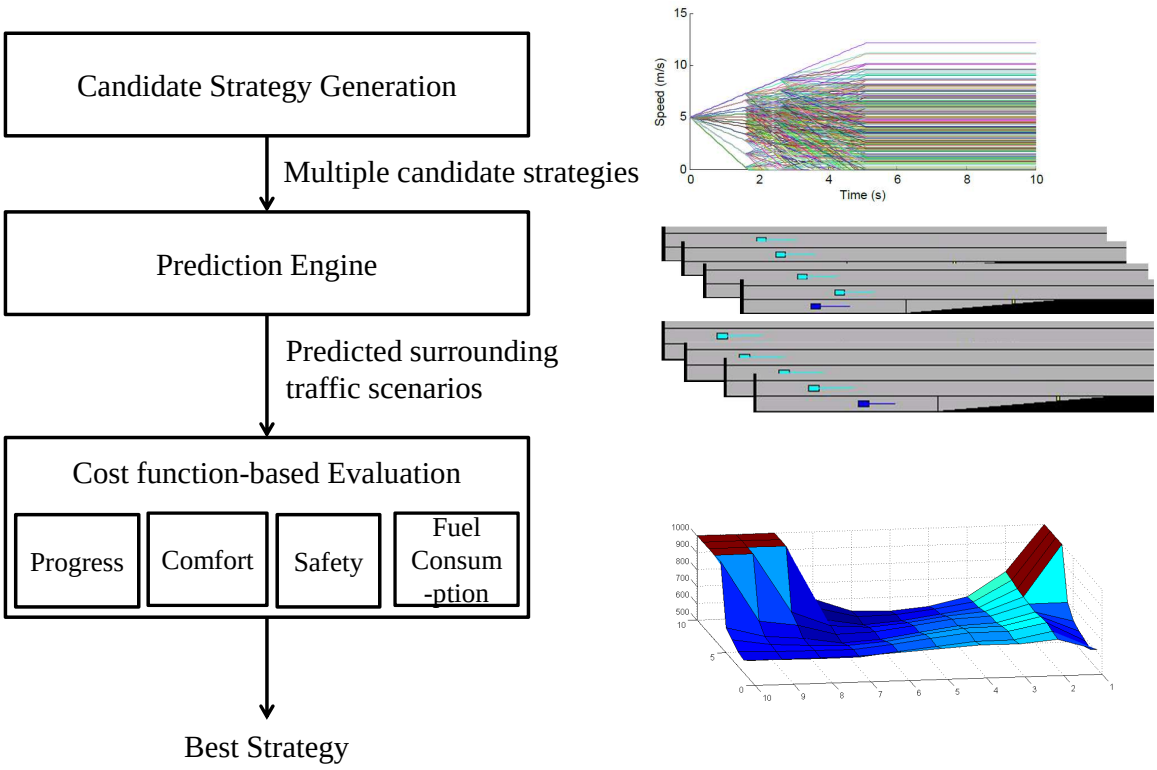


Figure 5.1: The Prediction- and Cost function-Based algorithm Framework

scenarios will improve the quality of the autonomous vehicle's decision making. This mechanism is also very adaptive to differing numbers of surrounding cars and different road conditions, such as number of lanes and speed limits.

5.1 Basic PCB Algorithm

5.1.1 Candidate Strategy Generation

As discussed in Section 4.2, the output of the behavior planner is the guidance for the lower-level planner/controllers. Therefore, the first step for the PCB algorithm is to generate a number of planner guidance candidates.

As described in Section 4.2, there are two outputs from the high-level behavior planner: the lateral velocity profile $vy_{0...t}$ and the distance keeping aggressiveness profile $th_{0...t}$. In the proposed algorithm a sampling mechanism is applied to the profiles to ensure the real-time per-

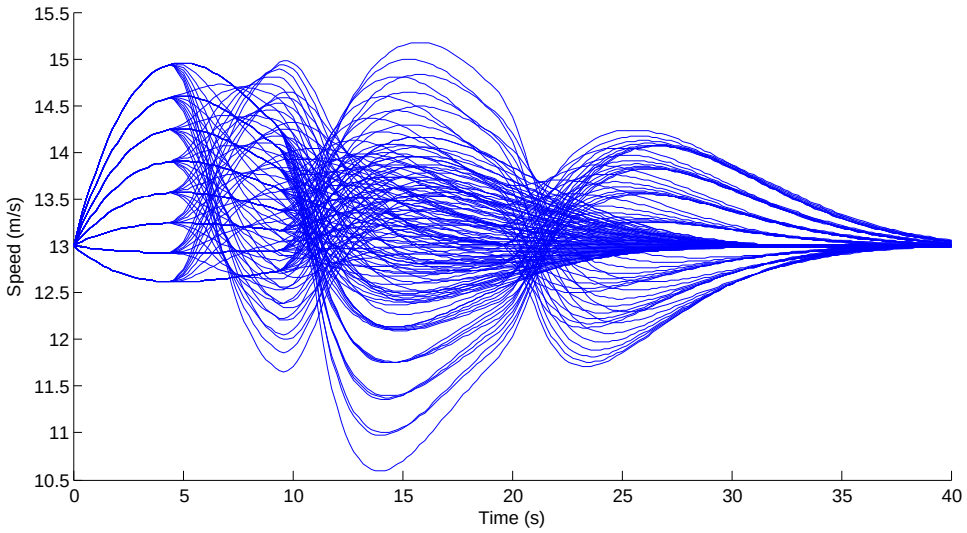


Figure 5.2: Time headway strategies change the speed profile of the host vehicle following a leading vehicle

formance of the decision making system.

For the lateral bias profile, sample points can be placed at $t = \frac{1}{3}t_{predict}$, $\frac{2}{3}t_{predict}$ and $t_{predict}$, where $t_{predict}$ is the time length of the lateral bias profile. For each of these times, we sample 5 different lateral offset values from $l = -w_{lane}$ to $l = w_{lane}$, which covers a potential left or right lane change and options in the middle. This gives us a total of $5^3 = 125$ path adjustment options. As the path needs to be continuous, the lateral offset cannot have discontinuity. Therefore, the sample points at different layers are connected using linear profiles.

For speed adjustment candidates, one example sample pattern is shown in Figure 5.2, the sample points are generated at $t = \frac{1}{3}t_{predict}$, $\frac{2}{3}t_{predict}$ and $t_{predict}$ with values of $h = 0.5, 1.5, 2.5, 3.5, 4.5$ seconds, which also gives us $5^3 = 125$ speed adjustment options. With the $125 \prod 125 = 15625$ profile candidates, the vehicle should have enough flexibility to perform normal lane driving for the next $t_{predict}$ time horizon including avoiding static obstacles, approaching and distance keeping to a slow vehicle, adjusting speed to perform a lane change and performing socially cooperative behaviors.

Figure 5.2 shows an example of 125 different host vehicle speed profiles following a leading vehicle. For a $t_{predict}$ of 20 seconds, the vehicle will use three different time headways. A wide variety of different speed profiles can be generated. For example, the vehicle could keep approx-

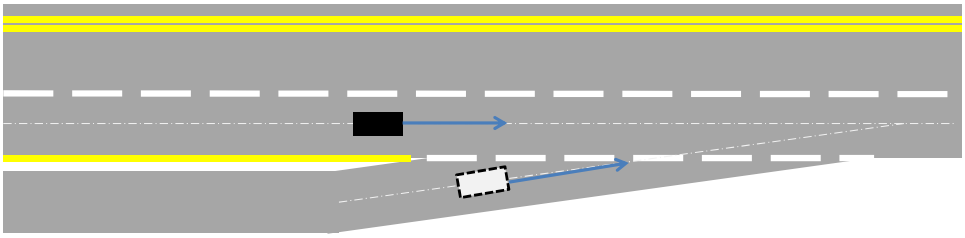


Figure 5.3: Freeway Entrance Ramp Domain

imately constant speed for $6.67s$, slow down for the next $6.67s$, and then speed up aggressively for the last $6.67s$. Because we only adjust the desired distance to the leading vehicle by changing the time headway, the smoothness of the velocity profile is still guaranteed. And the basic safety and functionality of following a leading vehicle is also satisfied. After the PCB time headway adjustments, the desired time headway for the host vehicle will revert to the default, which makes it catch up to its leading vehicle and then follow it with the same speed as the leader.

In the implementation of the PCB algorithm, trimming logic is implemented to reduce the computation time. If there is no static obstacle on the road and no lane change needed, the vehicle won't consider potential lateral offsets $l(t)$. When the autonomous vehicle is driving at the speed limit with no leading vehicles or upcoming entrance ramps that could require longitudinal speed change, no $t_h(t)$ will be sampled instead of iterating through all possible speed adjustment parameters.

Entrance Ramp In cases for which the host vehicle is able to identify what scenario it is in, some optimization could be applied towards generating more focused candidate strategies. Entrance ramp and lane changes are the two examples.

A road geometry model of the entrance ramp handling scenario is shown in Figure 5.3. In this scenario, we assume the host vehicle will always drive in the center of the main road. Therefore, the behavioral planning only needs to search for the longitudinal controller's time headway $th(t)$.

Following the sampling scheme discussed before, a modified version of the three-step approach is used. First, for the first $\frac{1}{3}t_{predict}$ the vehicle has 21 different time headway options $th1$ from 0 to 5 seconds. For the second $\frac{1}{3}t_{predict}$ the vehicle also has 21 of different time headway options $th2$ from 0 to 5 seconds. For the third $\frac{1}{3}t_{predict}$, we assume the vehicle should be close to

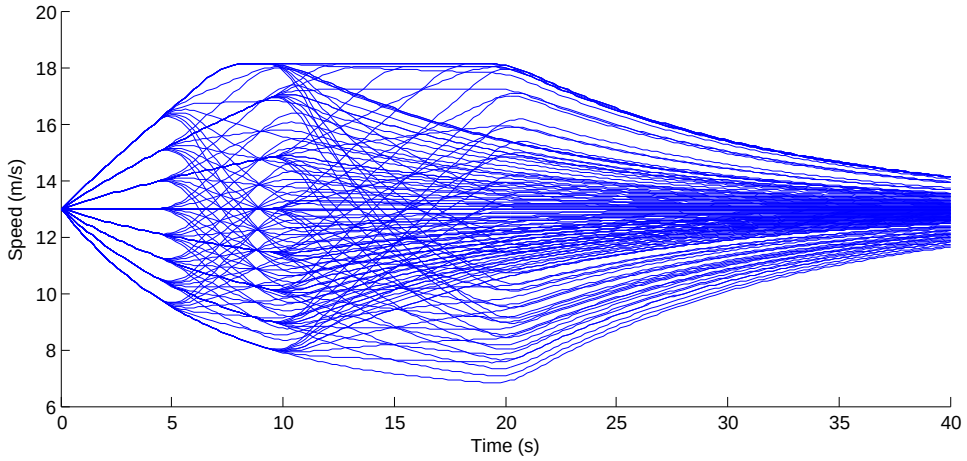


Figure 5.4: Time headway strategies change the speed profile of the host vehicle following a virtual leading vehicle

finishing the speed adjusment, and therefore revert to $th_{default}$.

In the general candidate strategy generation, when there is no leading vehicle, the host vehicle doesn't need to adjust its speed. However, for the entrance ramp scenario, the host vehicle sometimes has to adjust its speed to cooperate with the merging car. To allow the vehicle to adjust its speed even when there is no leading car in front of it, a virtual target for the ACC controller will be generated. The leading vehicle state is defined in Equation 5.1.

$$d_{lead} = d_{desired} \tag{5.1}$$

$$d_{desired} = d_{min} + th_{default} v_{lead} \tag{5.2}$$

$$v_{lead} = v_{host} \tag{5.3}$$

Here, d_{lead} and v_{lead} are the distance and velocity of the virtual vehicle in front of the host vehicle.

Figure 5.4 shows how th affects the speed of the host vehicle with a virtual vehicle in front. It shows that this mechanism allows us to generate a variety of smooth speed profiles.

Another dimension we need to search for is is the length of speed adjustment t_{adj}. Two different time headway options are generated so that the planning can select a speed adjustment that takes either a longer period of time or a shorter one. The first option is described above as $t_{adj} = \frac{2}{3} t_{predict}$. The second option is that $t_{adj} = \frac{1}{3} t_{predict}$, which means that after a shorter speed adjustment using two different time headways ($th1$ and $th2$), the vehicle will resume normal

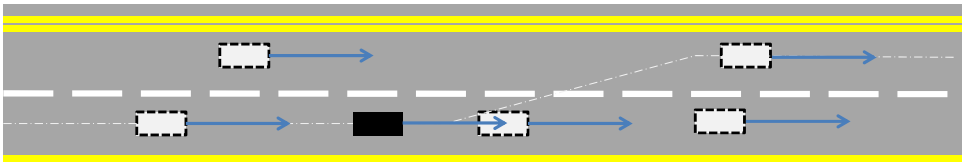


Figure 5.5: Lane Change Domain

driving mode with default parameters.

Lane Change Lane change is another special social behavior scenario where the candidate generation can be optimized. A normal lane change behavior usually involves two modules: lane selection and merge planning. Given our focus is on the social interaction between cars, we assume that the higher-level lane selection is given as an input to the PCB module. For example, the host vehicle's route planner is able to generate a right lane-change target if the host vehicle needs to take the next highway exit on the right side.

The PCB algorithm will be called whenever the target lane is different from our current lane and the turn signal of the vehicle will turn on accordingly for other vehicles to prepare for the host vehicle's movement. The model of this scenario is shown in Figure 5.5, where the black car is the host vehicle and the white cars are traffic vehicles. In the PCB lane change domain, we assume that other vehicles will not perform any lateral movement while the host vehicle is trying to do the lane change.

For the black car to find the best gap and perform a lane change, both speed adjustment and lateral movement commands need to be generated.

A similar scheme to that for entrance ramps is used to generate longitudinal speed profiles: for the first and second $\frac{1}{2}t_{adj}$ the vehicle has 21 different th options. There are also 2 options for t_{adj}. For the remainder of $t_{predict} - t_{adj}$, we assume the vehicle should be close to finishing the speed adjustment, and therefore revert to $th_{default}$.

Compared to entrance ramp management, one important difference for lane change strategy is the lateral movement start time. six different lane change start times are sampled, allowing the vehicle to perform either an immediate lane change or a lane change during or after speed adjustments. If no good lane change option can be found for the time horizon of $t_{predict}$, the vehicle will just keep driving with default distance keeping parameters.

After the lane change is triggered, the vehicle will start to perform a lane change. There are aggressive or conservative lane changes with different amplitudes of lateral speed of the host vehicle. In this thesis, instead of searching through all possible lane change lateral movement speeds, we assume the vehicle will perform a lane change lasting a constant t_{lc} seconds lane change whenever it is triggered.

Based on human driving road testing statistics, the length of lane change also varies between 4 to 8 seconds, based on traffic conditions. In our research, $t_{lc} = 5s$ lane changes provide reasonable smoothness for most lane changes cases. Therefore, a constant lane change lateral speed of $\frac{1}{5}w_{lane}$ per second is being used, where w_{lane} is the width of the lane. A total of 5292 different lane change strategies are generated in this step.

5.1.2 Prediction Engine

When human drivers operate vehicles with surrounding cars, they can interact efficiently with one another by recognizing/predicting other vehicles' behavior. The prediction capability and accuracy are major differentiators between new and experienced drivers. This mechanism provides experienced human drivers enough time to prepare and make proper maneuvers in advance. To implement this look-ahead ability in an autonomous vehicle, a prediction engine is built in the behavior planner.

As mentioned in Chapter 4, the behavioral planner only focuses on the interaction between host vehicle and surrounding cars. The road geometry for lane driving is not a factor at this level of planning. Therefore, the first step is to project all vehicles along the centerline of the road, as shown in Figure 5.6 and 5.7. It can be seen that the relative lateral position and longitudinal positions of cars remain constant after the projection.

This approach works well for highway and most urban roads that require the vehicle to perform a lane change. However, one practical issue is that for very curvy, narrow roads, this assumption may not apply. Therefore, the limitation of the planner is also generated in this step. If the curvature is too big or road topology or lane width does not fit into the normal lane driving or merging situation, PCB algorithm will be marked as unavailable to avoid its outputting the wrong command. Even though PCB algorithm cannot handle those extreme cases but given

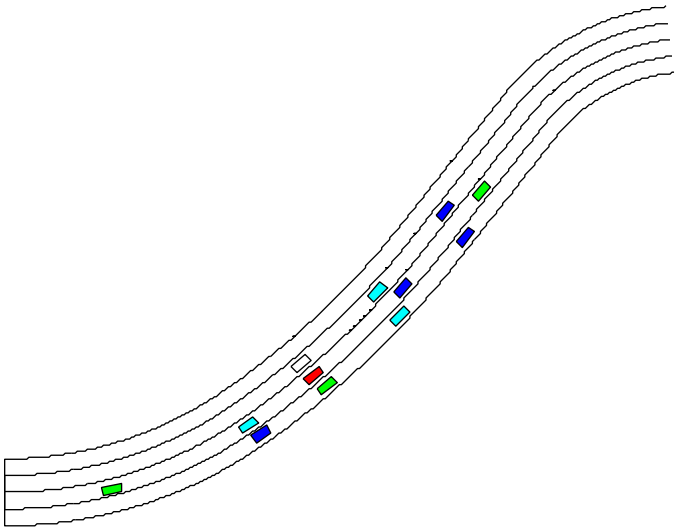


Figure 5.6: Traffic condition in world coordinates

Figure 5.7: Traffic condition projected into PCB algorithm coordinates

that they happen rarely, not being able to perform lane change does not really affect the overall autonomous driving capability.

To accurately predict surrounding vehicles' movements, we need to consider surrounding vehicles' reactions to their own microscopic traffic environment. For instance, if one vehicle runs faster than its leader, instead of maintaining constant velocity and crash into it, it should slow down as it gets close to it. We therefore introduce an interactive prediction kernel with basic distance-keeping capability into the prediction engine. There are two versions developed for the single-step prediction algorithm.

The first one is based on the logic that a given traffic vehicle will only slow down when it gets too close to its leader. The algorithm is described as follows:

1: Apply autonomous vehicle's strategy

2: $v_0(t + \Delta t) = v_0(t) + a_{cmd}\Delta t$

3: Update surrounding vehicles' states

4: **for** Each surrounding vehicle V_i **do**

5: $\quad d_i(t + \Delta t) = d_i(t) + (v_i(t) - v_0(t))\Delta t$

Enforce restrictions

$v_i(t + \Delta t) = min(v_i(t + \Delta t), v_{limit})$

if $D_{i,i_{lead}}(t + \Delta t) < D_{minSafe(i)}$ **then**

$v_i(t + \Delta t) = v_{i_{lead}}$

$d_i(t + \Delta t) = d_{i_{lead}} - D_{minSafe(i)}$

end if

end for

In which $D_{minSafe(i)}$ is the minimum safety gap threshold, $D_{i,i_{lead}}$ is the leading vehicle distance for vehicle i. It is assumed that the traffic vehicles will be extremely aggressive and will therefore not apply braking until the minimum safety distance is reached. Therefore, applying this algorithm makes the autonomous vehicle consider the worst-case scenario. For example, for a cut-in behavior in front of it, the host vehicle will be more conservative because in the PCB algorithm it believes that all the other cars will not pay attention until the very last moment before of a potential accident.

To better predict normal human-driven vehicles' behavior, a more advanced interactive prediction model is developed, where all the traffic cars perform distance keeping to each other. The following algorithm is used, which is similar to the baseline ACC controller introduced in Section 4.2.2.

Distance keeping algorithm for V_{hostID}

Identify leader vehicle

$d_{closest} = inf$, $v_{closest} = 0$

for Each vehicle **do**

Check lateral position

if $|lat_i - lat_{host}| < width_{dk}$ **then**

Identify the closest vehicle

if $lon_i - lon_{host} > 0$ and $lon_i - lon_{host} < d_{closest}$ **then**

$d_{closest} = lon_i - lon_{host}$

$v_{closest} = v_i$

end if

end if

end for

if $d_{closest} \neq inf$ **then**

Run LQR controller

k_v, k_d

$acc_{cmd} =$

else

$acc_{cmd} = k_{free}(v_{pref} - v_{host})$

end if

Apply maximum acceleration constraint

$acc_{cmd} = min(acc_{cmd}, f(v_{host}))$

Apply minimum acceleration constraint

$acc_{cmd} = max(acc_{cmd}, acc_{emergency})$

The prediction engine uses the single-step prediction mechanism described above for every Δt till $t_{predict}$. For each candidate strategy, the prediction engine will run to get a sequence of future scenarios with distances, speed, acceleration and relative lateral positions for each vehicle, for a horizon of $t_{predict}$. Figures 5.8 and 5.9 show two examples of different predicted scenario sequences with the host vehicle applying different candidate strategies. In these figure, the host vehicle is in the middle lane with red color, the time interval is $0.4s$.

In the case shown in Figure 5.8, the host vehicle speeds up quickly towards the leading car targeting a smaller gap to it and then back off from it to keep a further distance eventually. That's because the host vehicle is executing a strategy of $th1 = 0.5$, $th2 = 1.25$, $t_{adj} = 10$ with $t_{predict} = 20$, which means for the next 10 seconds, it will first keep $0.5s$ time headway to its leading vehicle for 3.3 seconds, and then keep $1.25s$ time headway to its leading vehicle. After the 10 seconds, it will resume to a normal time headway, which is $1.0s$.

In the other case shown in Figure 5.9, the host vehicle follows a command to keep $2.5s$ from the leading vehicle for $10s$. Therefore, for the first 5 steps shown in the figure, the host vehicle does not speed up to catch up with its leader. After $t_{adj} = 10$, it starts to converge to the default desired time headway and distance, which is $th = 1s$ from its leader.

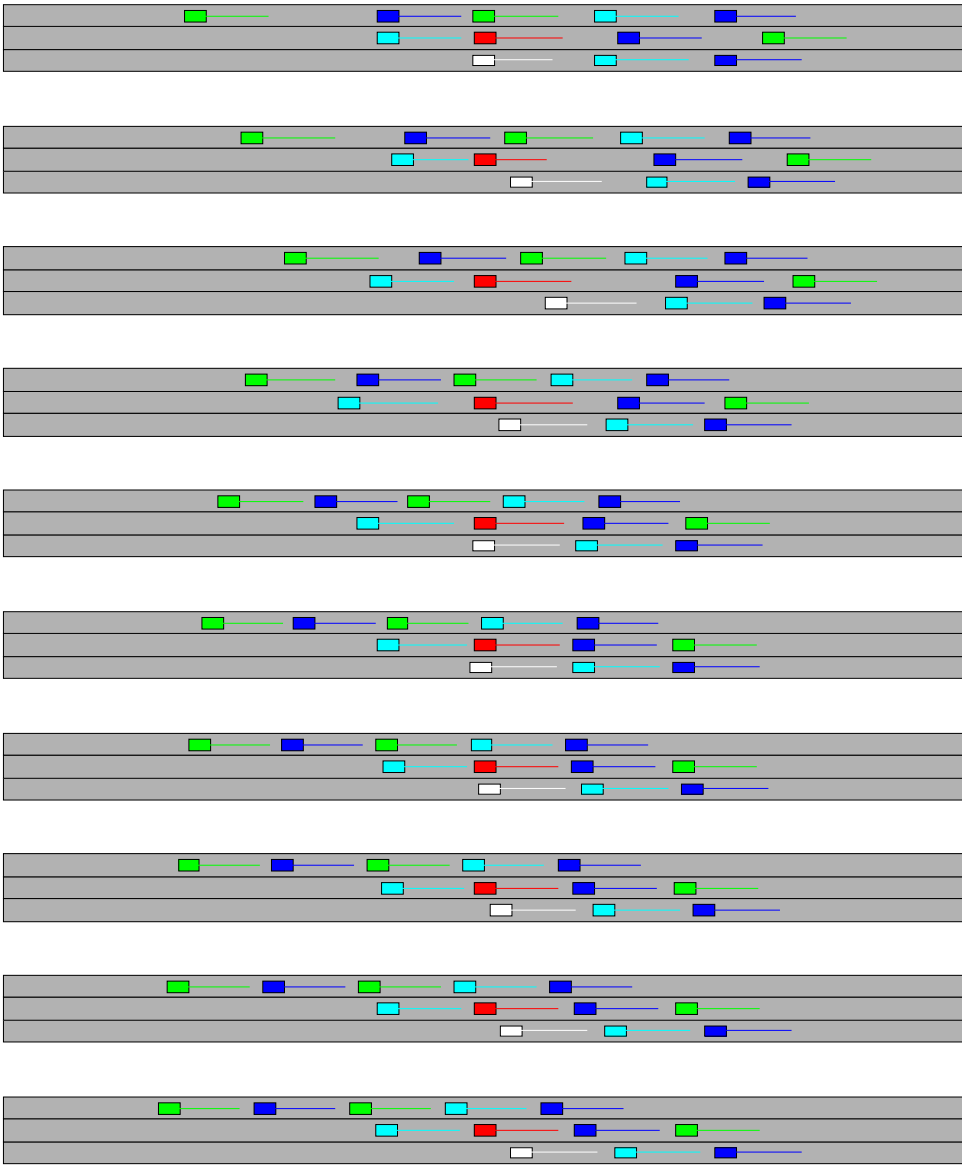


Figure 5.8: Prediction of all the traffic vehicle's reaction to a certain time headway strategy of the host vehicle: $0.5s$, $1.25s$ time headways with $10s$ speed adjustment time duration and $20s$ prediction time

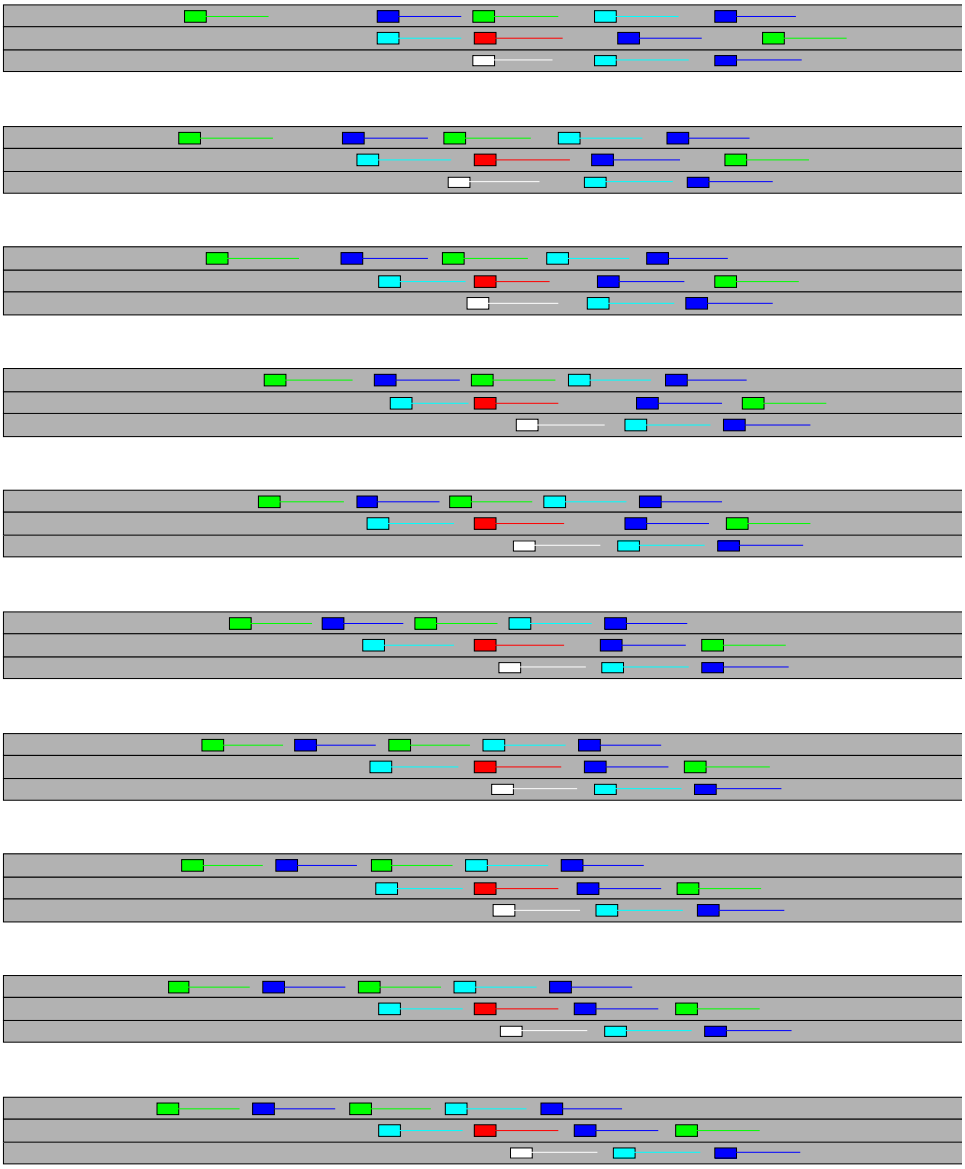


Figure 5.9: Prediction of all the traffic vehicle's reaction to a certain time headway strategy of the host vehicle: $th_1 = 2.5s$, $th_2 = 2.5s$ time headways with $10s$ speed adjustment time duration, meaning that for the first and second of $3.3s$, the host vehicle will be targeting keep $2.5s$ time headway to its leading vehicle. In the prediction result, the vehicle starts to slow down to keep further away than its leading vehicle than the default time headway. After $10s$, the vehicle speeds up to resume the default time headway, which is $1.0s$.

Different strategies not only affect the speed profile of the host vehicle, but by using the prediction engine, we can also see how they affect the following vehicle of the host vehicle. The cyan-colored vehicle behind the host vehicle in Figure 5.8 speeds up to catch up with the host vehicle. And after the host vehicle slows down slightly, it also slows down. In Figure 5.9, the host vehicle starts to slow down from the beginning. Therefore, the cyan-colored vehicle following it slows down first. It is demonstrated that with different candidate strategy, the prediction is able to generate different host vehicle and its surrounding traffic states.

Lane Change Scenario For prediction in lane change scenarios, all the traffic vehicles are assumed to be driving along their current lane. Only the longitudinal speed is be adjusted based on the above model.

In the example shown in Figure 5.10, the red host vehicle is commanded to make a lane change to its left into a tight space. The blue traffic vehicle in that lane is able to slow down for the host vehicle's lane change and close cut-in in front of it. If the blue vehicle doesn't slow down or follows constant speed, it will crash into the host vehicle. Therefore, motion planners using a constant-speed assumption will treat this host vehicle's lane change behavior as infeasible. This means that a state-of-the-art planner will prevent the host vehicle from performing the maneuver shown in Figure 5.10 eventhough it is a reasonable behavior in the real world.

Figure 5.11 shows the inter-vehicle distances and speed profiles of the two vehicles. At the beginning, the blue vehicle in the neighboring lane runs faster than the host vehicle. When the blue vehicle sees the host vehicle cut in front of it, it starts to apply the brake and eventually is able to converge to the desired distance keeping target distance.

Though this example behavior may not be preferable in some cases, it is necessary for the host vehicle to treat it as a candidate option. A penalty is applied to make the host vehicle not select that option if other better options are available. The strategy selection criteria will be discussed in Section 5.1.3.

Entrance Ramp Scenario For prediction in entrance ramp scenarios, the cars on the ramp are assumed to follow the entrance ramp center line to enter the main road. Merging vehicles are assumed to follow the same distance keeping logic as described in Algorithm 5.1.2. Cars on the

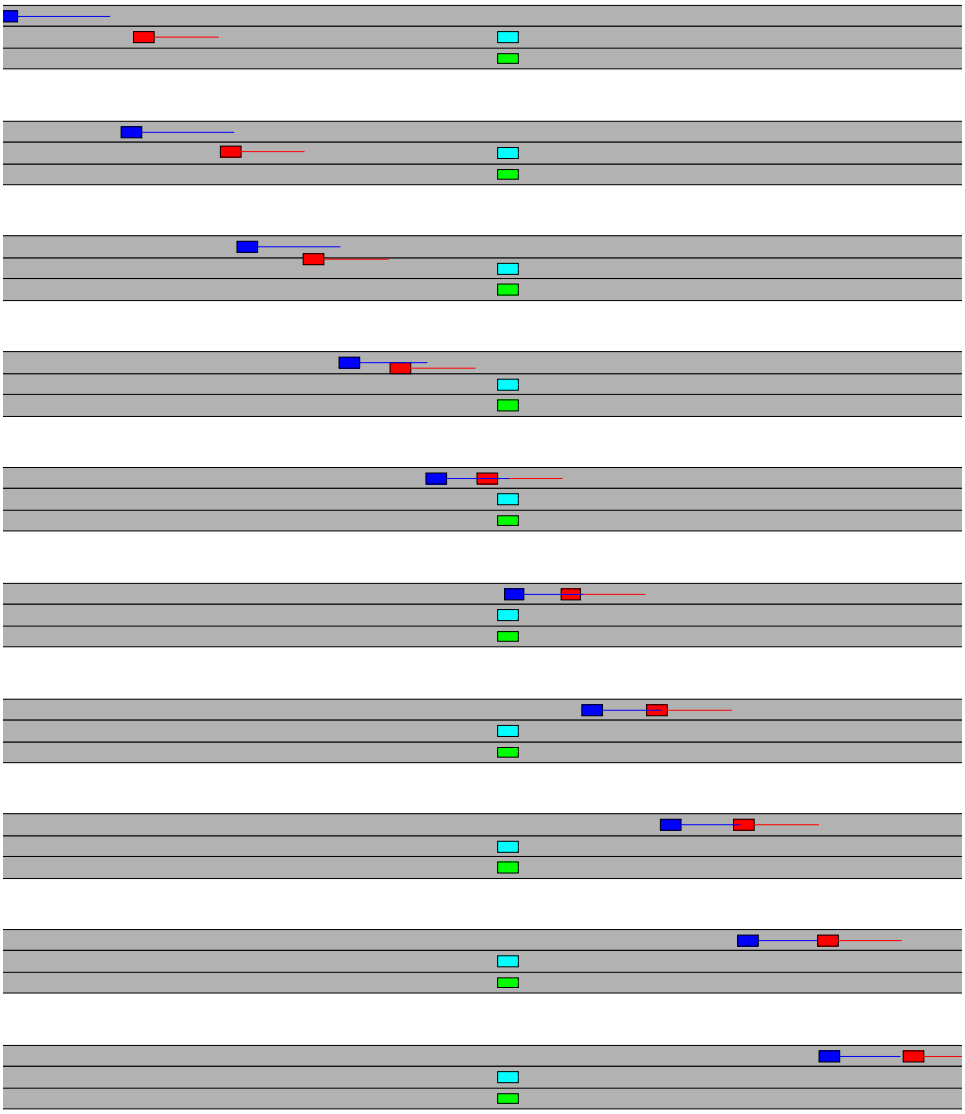


Figure 5.10: Time sequence of the host vehicle cut closely into the neighboring lane in front of a faster-moving vehicle. The faster-moving car is able to slow down for the host vehicle's behavior.

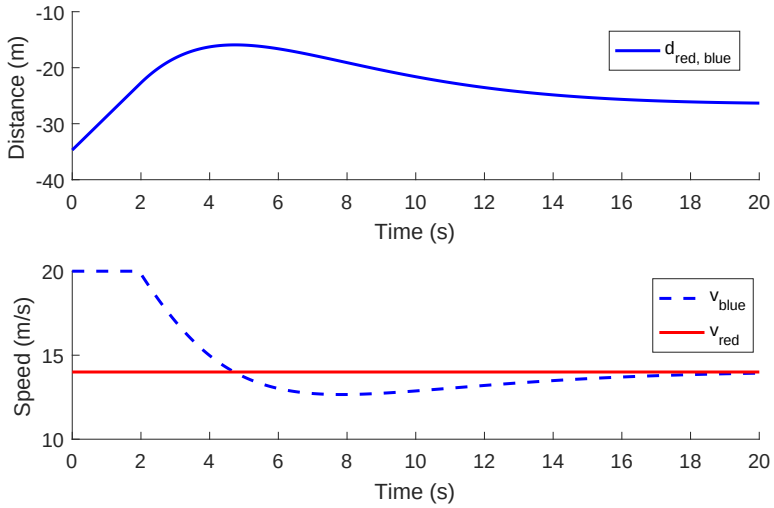


Figure 5.11: The distance between the two vehicle and speed profile of the fast-moving car in the neighboring lane and the host vehicle cut into that lane.

main road are predicted to distance-keep to their immediate leaders within the lane width.

Figure 5.12 shows that if the vehicle in the main lane does not slow down for a merging vehicle, the latter will reasonably slow down to keep a safety buffer between itself and the vehicle in the main lane.

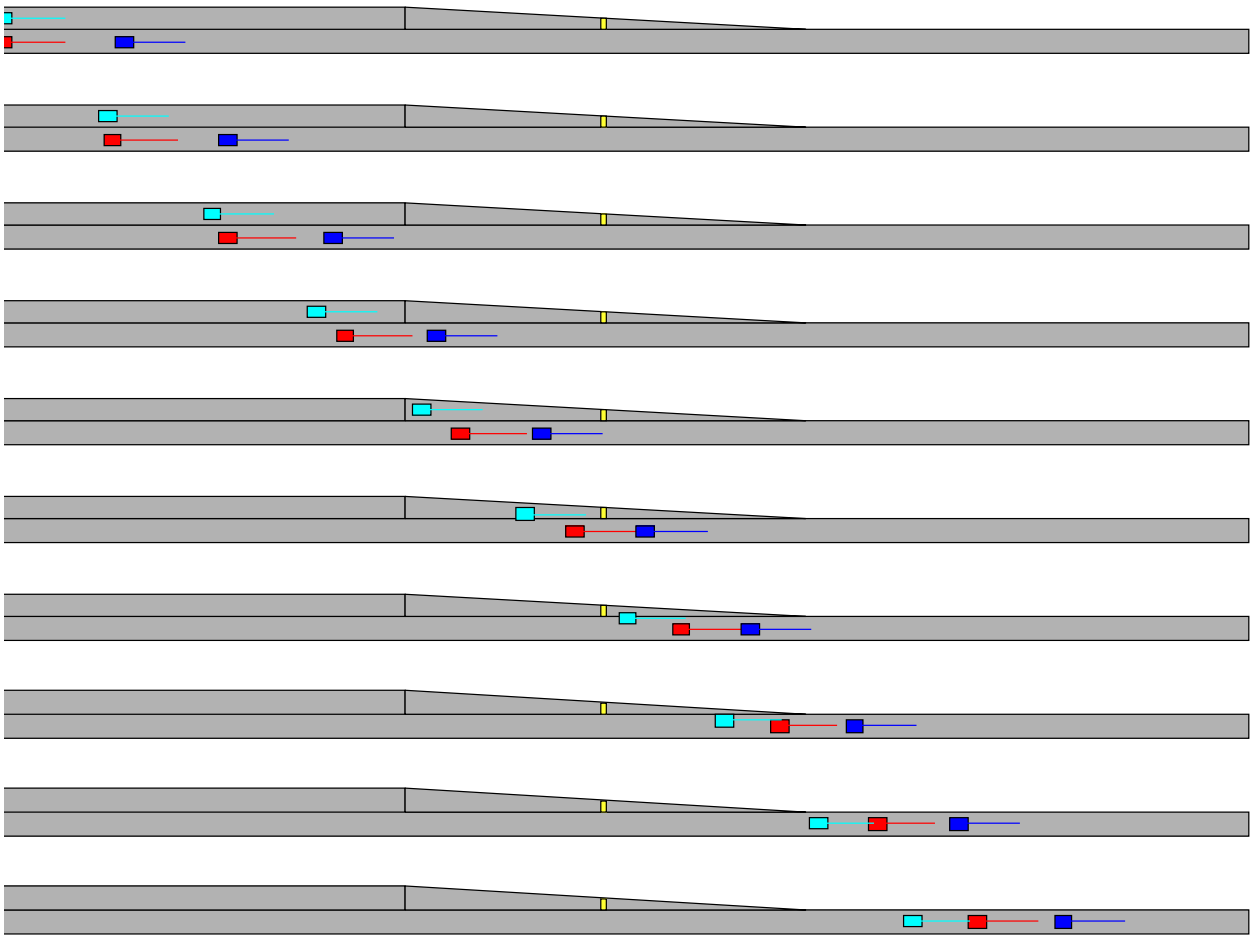


Figure 5.12: Applying the prediction engine, the merging vehicle (cyan) will slow down for the vehicle in the main lane (red).

5.1.3 Cost Function-Based Evaluation

After predicting a number of scenario sequences corresponding to each strategy for $t_{predict}$ with an interval of δt, a cost is computed for each of them. The cost is a mathematical way of describing the scenario's degree of preferability. The metrics in the cost representation are safety, comfort, progress and fuel efficiency.

In the computation, all the predicted scenarios from $t = 0$ to $t = t_{predict}$ are evaluated, as shown in Equation 5.4.

$$C_{strategy(i)} = \sum_{t=0}^{t_{predict}} C_{scenario(i,t)} \tag{5.4}$$

$$C_{scenario(i,t)} = C_{speed} + C_{comfort} + C_{safety} + C_{fuel} \tag{5.5}$$

Cost Function Parameterization There are multiple methods of cost function representation, including linear and polynomial models, etc. In PCB, we use a piece-wise linear model for the cost function's parameterization. An upper bound and lower bound for the inputs are also set to represent the safety criteria check. For example, if the distance between two cars is very small, the cost will be infinite to indicate the host vehicle should not select this strategy.

1. Progress Cost: Progress cost forces the autonomous vehicle to achieve its objective. For freeway driving, the driving goal is to keep reasonable speed in order to make steady progress towards reaching the vehicle's destination. Therefore, a progress cost is built to penalize low driving speed, which will push the vehicle to accelerate to the speed limit when there are no obstacles in the way. A linear cost based on the average speed of the vehicle is shown in Equation 5.6.

$$C_{speed} = v_{limit} - v_{host} \tag{5.6}$$

In addition, the vehicle should perform distance keeping to its leading vehicle if it exists. A second term penalizes distance-keeping error, which is the difference between the vehicle's desired distance to its leader and the current distance to it. The desired distance is

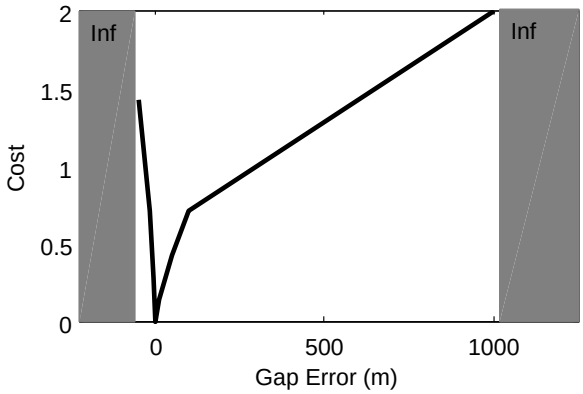


Figure 5.13: Distance keeper progress cost: $C_{DKpro}(\Delta d_{gap})$, with vertices (-25,1.5), (-15,0.9), (-5,0.14), (0,0), (10,0.14), (50,0.43), (100,0.7), (1000,2)

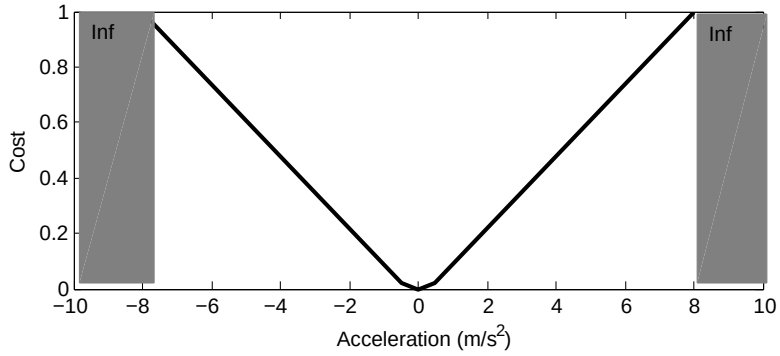


Figure 5.14: Comfort cost: $C_{comfort}(acc)$, with vertices (-8,1), (-0.5,0.02), (0,0), (0.5,0.02), (8,1)

computed in Equation 5.7.

$$d_{desired} = d_{min} + th_{default}v \tag{5.7}$$

The distance keeping progress cost is computed according to the difference between the current distance to the leading vehicle ($d_{current}$) and the desired distance to it ($d_{desired}$), which is Δd_{gap} ,as shown in Figure 5.13. Its unsymmetrical cost function pattern indicates that the vehicle is more tolerant of positive distance-keeping errors than negative ones, which means it is too close to its leader.

2. Comfort Cost: While driving a car, experienced human drivers will try to avoid large accelerations and decelerations for greater comfort. Therefore, a comfort cost $C_{comfort}$ is built to represent this logic, as shown in Figure 5.14. Minor accelerations from $-0.5m/s^2$

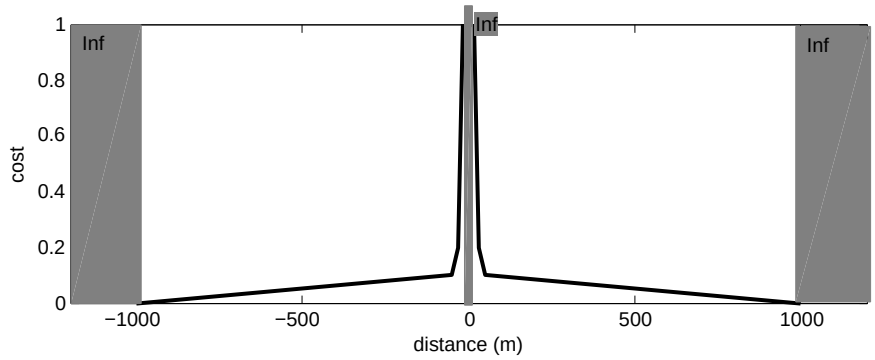


Figure 5.15: Clear distance cost: $C_{distance}(distance)$, with vertices (-1000,0), (-50,0.1), (-30,0.2), (-15,1), (15,1), (30,0.2), (50,0.1), (1000 , 0)

to $0.5m/s^2$ are not penalized as proportional to higher amplitude acceleration because that level of acceleration doesn't not affect the comfort. For example, instead of deceleration at $-1.5m/s^2$ for 1 seconds, the host vehicle will prefer to decelerate at $-0.5m/s^2$ for 3 seconds.

3. Safety Cost: The safety cost of a scenario consists of two terms: the clear distance cost and the braking distance cost. The clear distance cost $C_{distance}$ penalizes moving too close to surrounding vehicles within some lateral distance of the host vehicle. Compared to the distance keeping progress cost, it will penalize the host vehicle's performing a close cut-in in front of other cars. If at any time during a lane change, when the host vehicle is already almost merged almost into the target lane, the cars in that lane have not been fully cleared, the clear distance cost will penalize that strategy. This makes sure the host vehicle should always has a safety buffer $0.5(width_{dk} - width_{car})$ wide and $15m$ long based on the vertices selection. The cost function of $C_{distance}$ is shown in Figure 5.15, in which $distance$ is the distance to one surrounding vehicle.

   However, the clear distance cost $C_{distance}$ by itself doesn't provide enough heuristic information for us to avoid collision in situations wherein the autonomous vehicle approaches a slow car at high speed, because it does not consider the vehicles' velocities. Therefore, another safety cost based on the braking distance difference Δd_{brake} between two vehicles

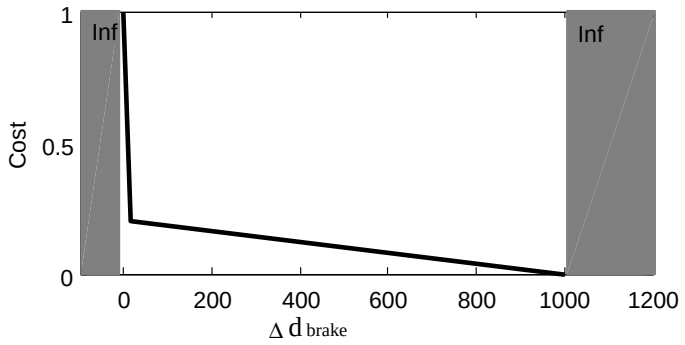


Figure 5.16: Braking distance cost: $C_{brake}(\Delta d_{brake})$, with vertices (0,1), (15,0.2), (1000,0)

is computed.

$$\Delta d_{brake} = d_{lead} + \tfrac{1}{2} v_{lead}^2 / a_{maxdec} - v_{follow} * t_{response} - \tfrac{1}{2} v_{follow}^2 / a_{maxdec} \tag{5.8}$$

In Equation 5.8, the remaining distance after braking is computed, in which d_{lead} is the distance to the leading vehicle, v_{lead} is the velocity of the leading vehicle, a_{maxdec} is the maximum deceleration the autonomous vehicle can issue, v_{follow} is the velocity of the following vehicle and $t_{response}$ is the system latency of velocity command execution. Δd_{brake} smaller than 0 means that if the leading vehicle begins to brake hard, we do not have enough space to avoid the collision, which is obviously not acceptable. The cost function C_{brake} based on Δd_{brake} is shown in Figure 5.16.

Then the overall safety cost is computed using Equation 5.9:

$$C_{safety} = \mu_3 C_{brake}(\Delta d_{brake}) + \mu_4 C_{distance}(d) \tag{5.9}$$

4. Fuel Consumption Cost: Besides freeing people from the driving task, one other major benefit of an autonomous driving system is to improve energy efficiency. To take that factor into account in the vehicle's decision making, the fuel consumption for each strategy is computed based on a statistical fuel consumption model [4]. Then the fuel consumption is converted to a cost term to make the vehicle prefer more fuel-economical behaviors. In this thesis, the weight of this cost is set to be very low because compared to other factors, fuel-consumption is not as important for passenger vehicles, especially given the social behavior interaction that needs to happen. But for a heavy semi-truck or for long-distance

ACC applications, this cost will be more dominant.

After the cost function is computed, the best strategy is selected as the one with the lowest accumulated scenario cost. If the best strategy's corresponding cost is infinite, meaning there is not an available strategy, then the host vehicle will perform emergency braking and also indicate to the human driver to take over.

Cost Function Optimization

After all the cost terms are defined, we can convert every scenario into a cost value. It is obvious that the value computed by the cost function-based evaluator will be affected by cost function shape and weights. To make sure the selection and tuning of cost functions will result in reasonable driving performance, a learning-based optimization is applied. Experienced human driving data are collected as a guideline for cost tuning. The vertices of the cost functions are tuned to make sure the output of autonomous vehicle fits human driver's behavior. In this thesis, the cost function shapes and weights are optimized using a combination of the learning-based approach [57] and manual tuning. After the learning process, the PCB algorithm of the autonomous vehicle evaluates preferable scenarios with undesirable driving scenarios very well.

5.1.4 Case Tests

The basic PCB algorithm is based on the assumption that each vehicle will follow its current lane and will keep the distance to its leading vehicle by slowing down. Case tests are implemented to demonstrate the capability of the basic PCB algorithm using the framework described above. The algorithm is implemented to output the desired time headway th and desired lateral speed v_l each cycle. The algorithm also replans at $5Hz$ to react to new perception inputs of the surrounding vehicles.

The simulation framework is shown in Figure 5.17. It uses a decentralized simulation architecture in which each car makes its own decision and reports to the simulator's step simulation module. In the simulation, all traffic cars are simulated using the model described in Algorithm 5.1.2. Our case testing in simulation shows that the autonomous vehicle can make very good selection, which will be shown in both lane change and entrance ramp scenarios.

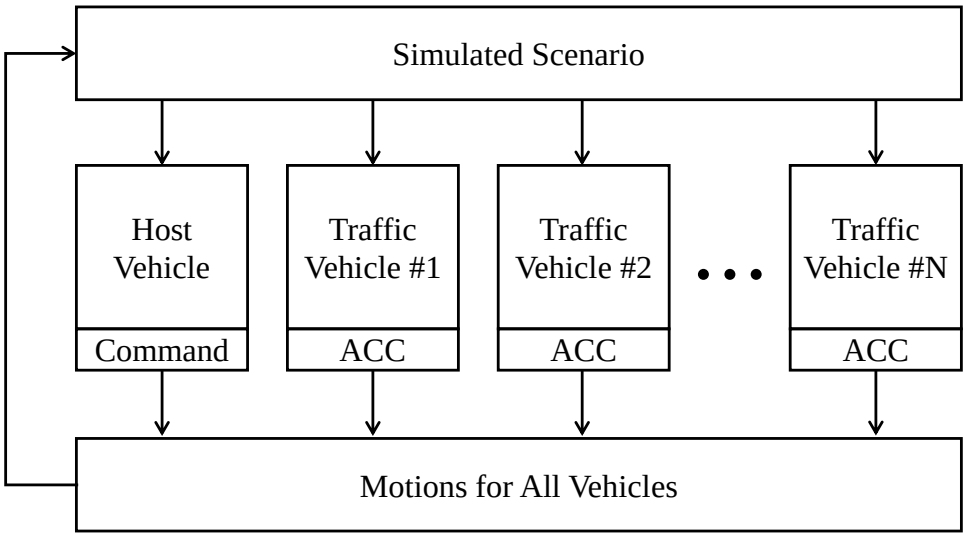


Figure 5.17: Simulation Framework

Entrance Ramp Case testing results are generated for the entrance ramp scenario. In the first case test as shown in Figure 5.18 and 5.19, the automated vehicle decides to speed up to approach its leading vehicle. This best strategy among all the candidate is $\{th1 = 1.0s, th2 = 1.0s, t_{adj} = 5s\}$. It makes sure the host vehicle arrives at the ramp earlier than the neighboring vehicle on the entrance ramp.

The plots in Figure 5.19 shows that the host vehicle starts with a faster speed than the merging vehicle, and the host vehicle's initial position is $10m$ behind the merging vehicle. Using the PCB algorithm it is able to identify that the best option is to drive normally approaching its leading car, which will naturally make enough space for the merging car to merge in behind the host vehicle. On the other hand, the rule-based logic for entrance ramps described in Section 4.4.1 will make the host vehicle slow down because it is behind the merging car, which will cause the host vehicle to sharply decelerate.

In Figure 5.20, the automated vehicle decides to slow down to arrive at the ramp later than the neighboring vehicle on the entrance ramp. Instead of speeding up to catch up with the leading vehicle, given the knowledge of the merging vehicle and its expected behaviors, the host vehicle decides to stay further from its leader to make space for the merging vehicle to cut in.

Corresponding to the scenario sequence shown in Figure 5.20, the speed profile and distance between cars are shown in Figure 5.21. The host vehicle slows down from the beginning of

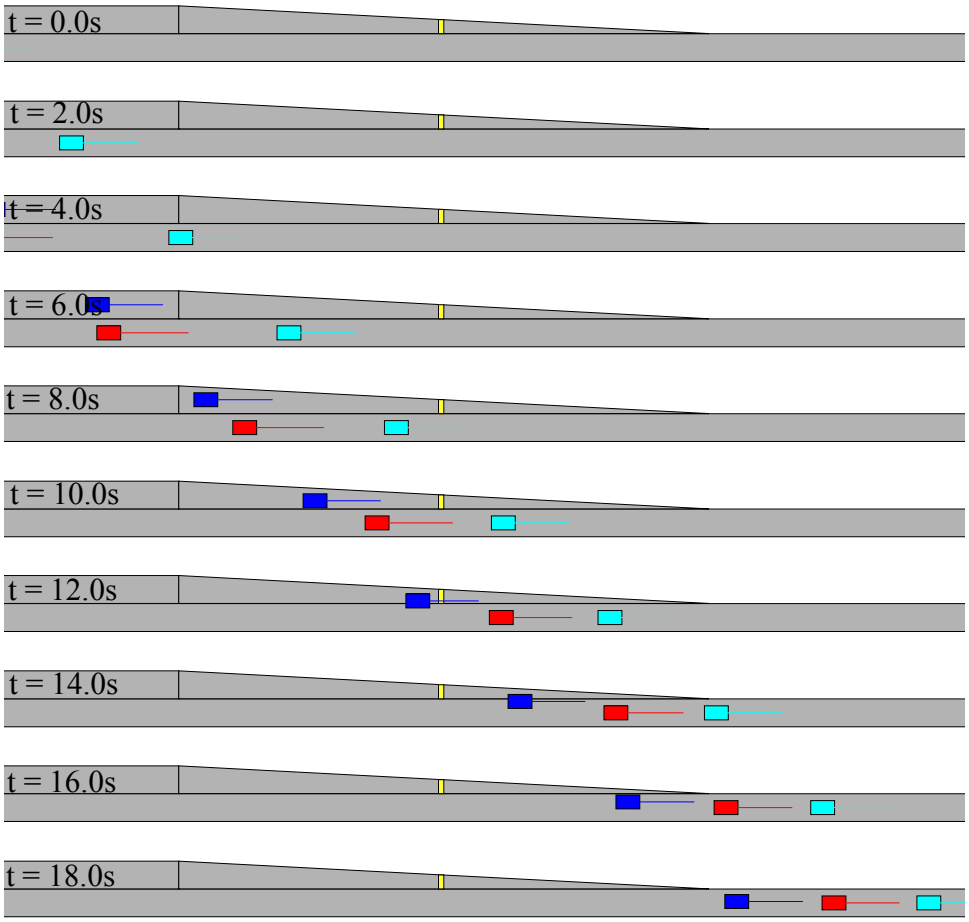


Figure 5.18: In the first case test, the best strategy generated by the PCB algorithm is $th1 = 1.0s$, $th2 = 1.0s$, $t_{adj} = 5.0s$. The simulation result shows that the host vehicle follows the preferred desired time headway and is eventually able to reach to the merge point earlier than the merging vehicle.

the interaction. After t_{adj}, the host vehicle resume to normal distance keeping time headway. However, at that time, the merging car hasn't entered the distance keeping zone of the host vehicle. Therefore, the host vehicle at $t = 8$ to $t = 10$ will accelerate first and then suddenly decelerate. This is caused by the discretization of the search space. In general, the jerk will be eliminated by applying replanning, which will be discussed in the implementation Section 5.4.2.

Lane Change Two examples are shown for the lane change cases. In the first case in Figure 5.22, the vehicle is commanded to perform a left lane change at the beginning of a randomly generated scenario. The speed profile is shown in Figure 5.23. There is a neighboring vehicle right next to the host vehicle; therefore, it needs to first select the best gap and perform speed

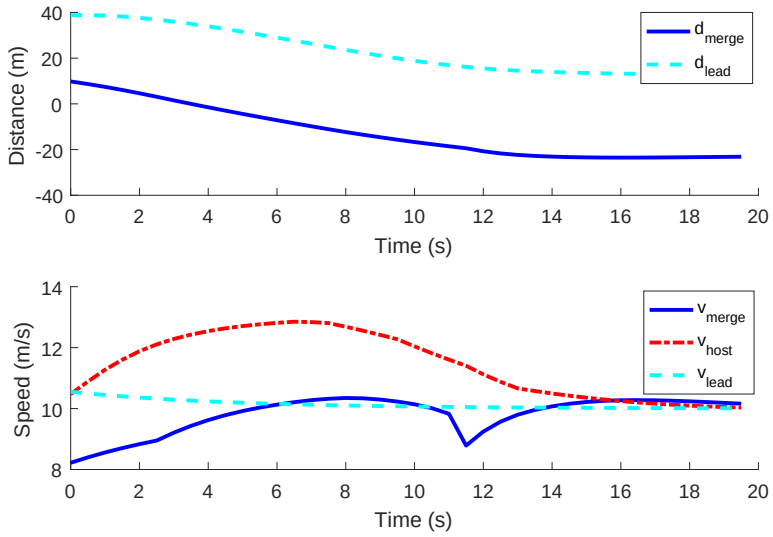


Figure 5.19: In the first case test, the distances and velocity of the vehicles are analyzed. It shows that how the host vehicle speed and to converge to its optimal distance to its leading car. It also shows the merging vehicle start move laterally and slows down to keep a safe gap to the host vehicle

adjustment to achieve it. The best candidate strategy it found is $\{th1 = 0.75s, th2 = 1.5s, t_{adj} = 8.0s\, t_{merge} = 5.0s\}$. It speeds up to catch up with its leading vehicle to gain more speed, and then slows down to merge into the neighboring lane which has a lower speed limit.

Figure 5.24 shows a case with similar initial conditions. The distances and speed profiles of vehicles are shown in Figure 5.25. But instead of speeding up for the best lane change gap, the host vehicle slows down to wait for the neighboring car to pass it first. The only difference between these two cases is that the leading vehicle of the host vehicle is slightly slower in the second test case. Therefore, the host vehicle is not able to gain enough gap by just speeding up slightly to shrink its distance to its leading car before performing the lane change.

Compared with the lane change logic-based algorithm, the PCB algorithm shows better robustness to handle different traffic situations. It is able to speed and slow down to actively achieve the best opening for lane change instead of keeping in the same relative position to the car in the target lane and waiting.

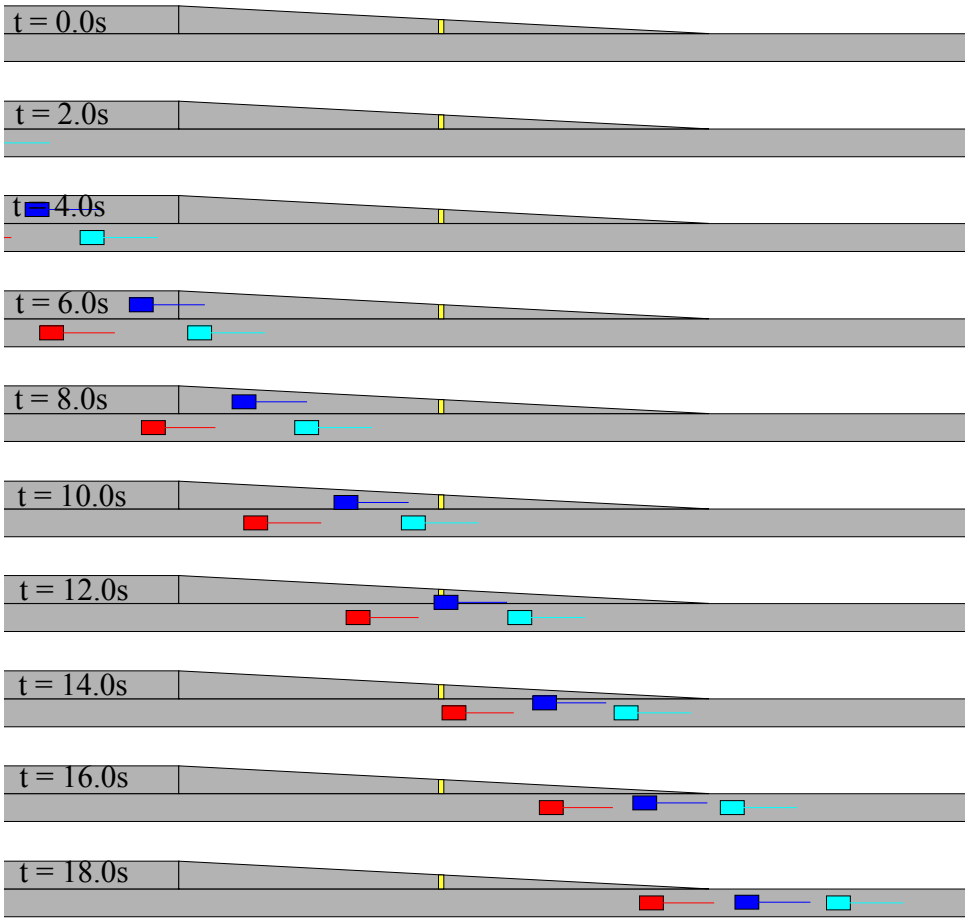


Figure 5.20: In the second case test, the best strategy of the host vehicle is to execute a larger time headway to slow down. This creates a larger gap for the merging vehicle to safely merge in. Then the host vehicle speeds up to the default distance for car following.

5.1.5 Summary

We have demonstrated the ability of the Prediction- and Cost-function Based algorithm to make behavior-level decisions for speed adjustment and lateral lane change in for entrance ramp and lane change scenarios. By assuming surrounding vehicles will react to their local surrounding cars and applying prediction for each candidate strategy of the host vehicle, it outperforms motion planners with a simplified assumption that all other cars keep constant speed. This allows the automated vehicle to better predict the consequence of its own candidate strategy options. It also extends the range of feasible strategies for the automated vehicle. For example, it could potentially select a strategy that cuts in front of other cars closely if that is the only option to avoid obstacles or stopped cars.

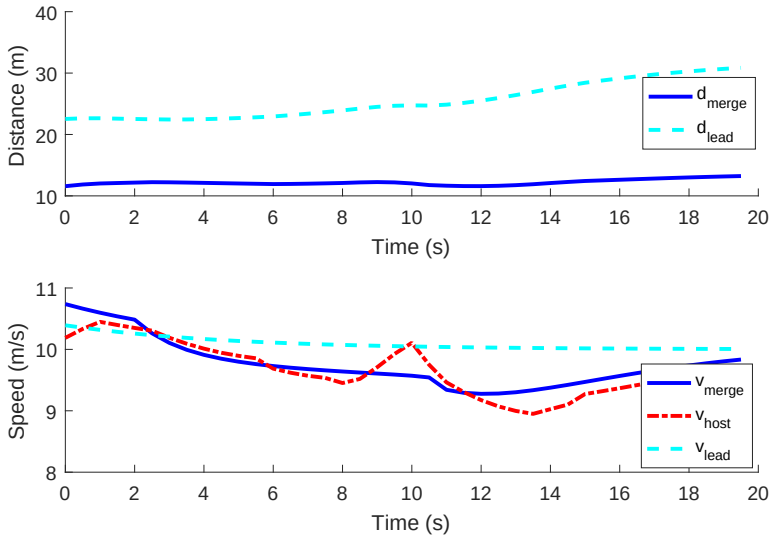


Figure 5.21: The distances between cars and velocities for the second case test in entrance ramp situation. It shows that the leading distance for the host vehicle increase and eventually the merging vehicle becomes the leading vehicle at around $t = 10.0s$. From then, both the merging vehicle and the host vehicle start to adjust their speeds to reach the default car following distance computed by $t = 1.0s$ time headway.

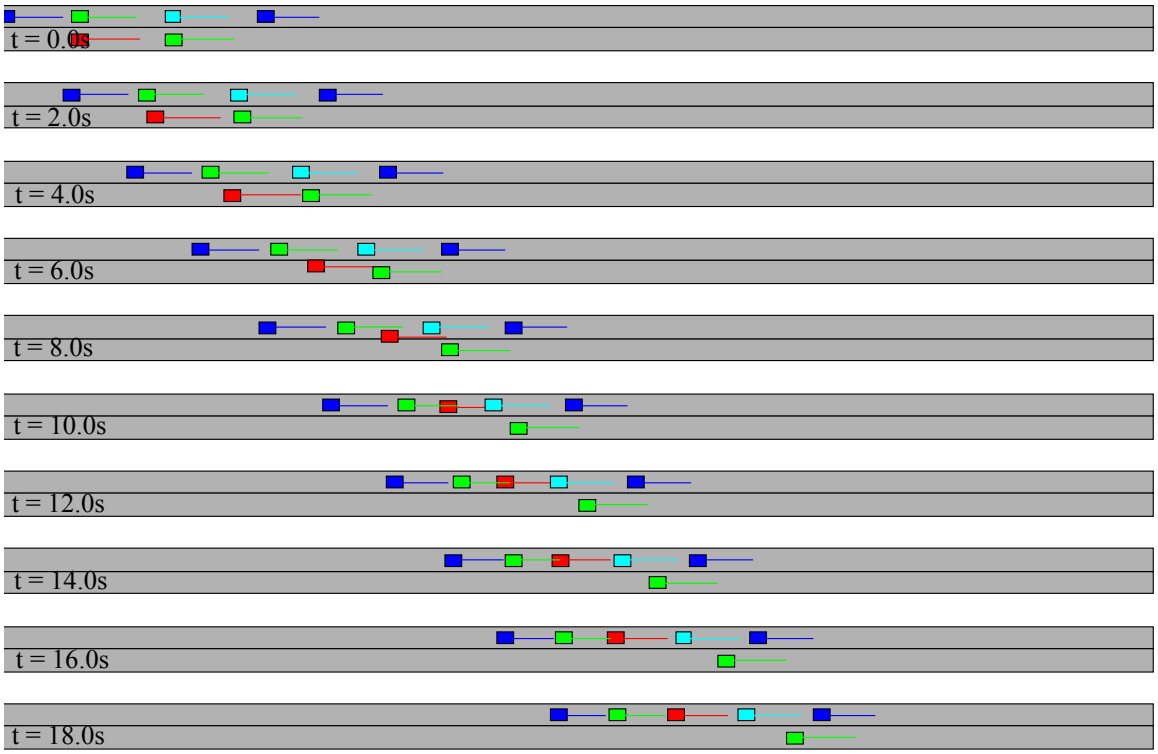


Figure 5.22: The first example case for lane change situation, the host vehicle finds the best strategy of $\{th1 = 0.75s, th2 = 1.5s, t_{adj} = 8.0s\}$. Therefore, it first speeds up to catch up with its leading vehicle to gain more speed, then slows down to merge into the neighboring lane where the cars are at slightly lower speed.

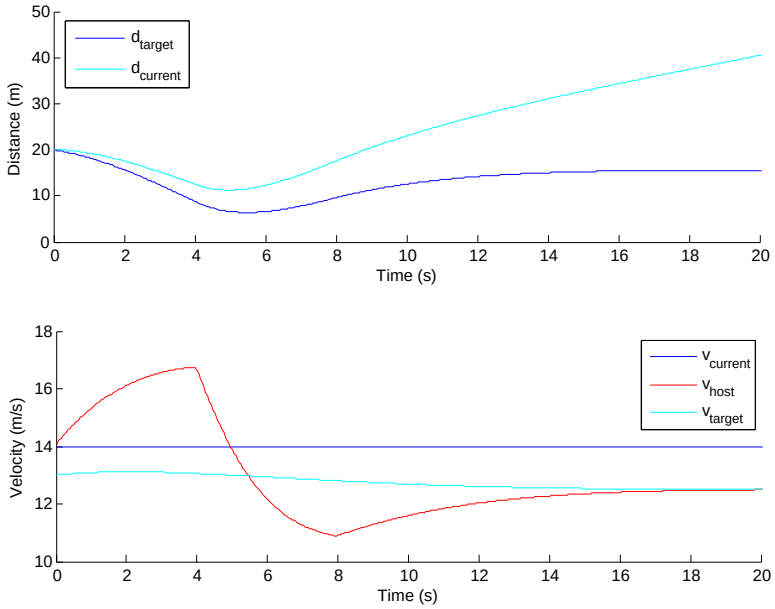


Figure 5.23: The distances and velocity of the vehicles in the first example case for lane change situation. It shows the host vehicle speeds up for $4.0s$ and slows down for $4.0s$.

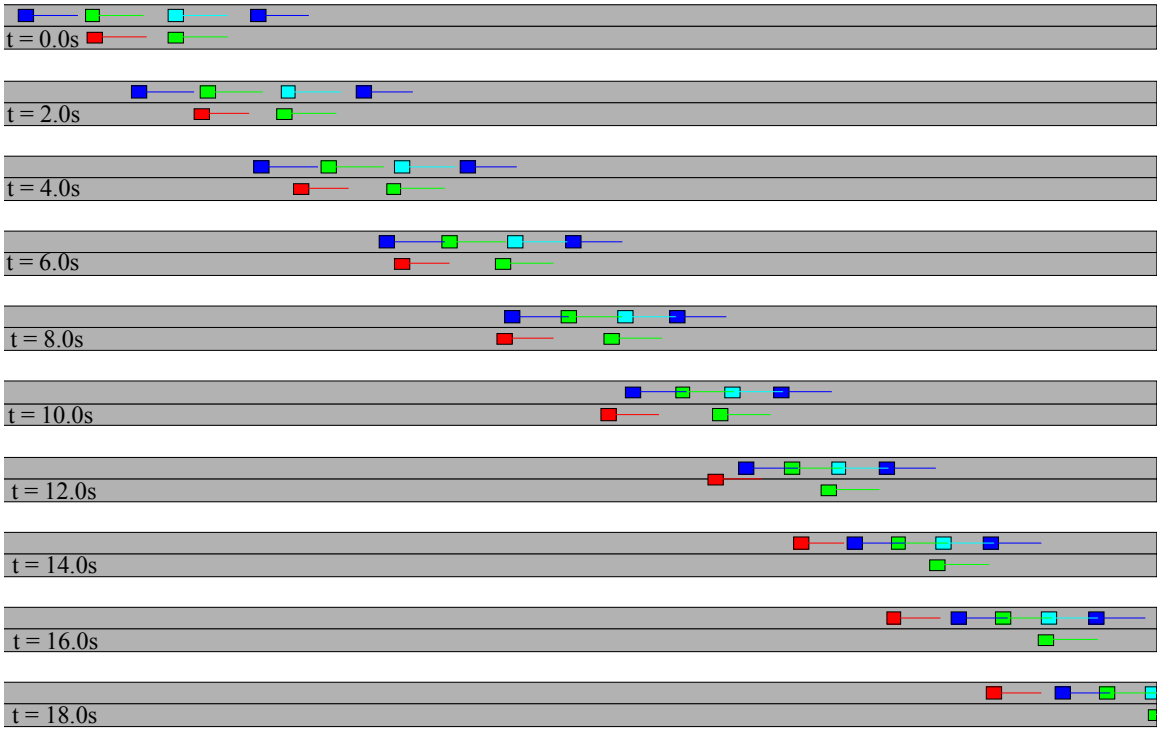


Figure 5.24: In the second example of lane change case test, the best strategy is for the host vehicle to slow down to merge behind the neighboring lane vehicles.

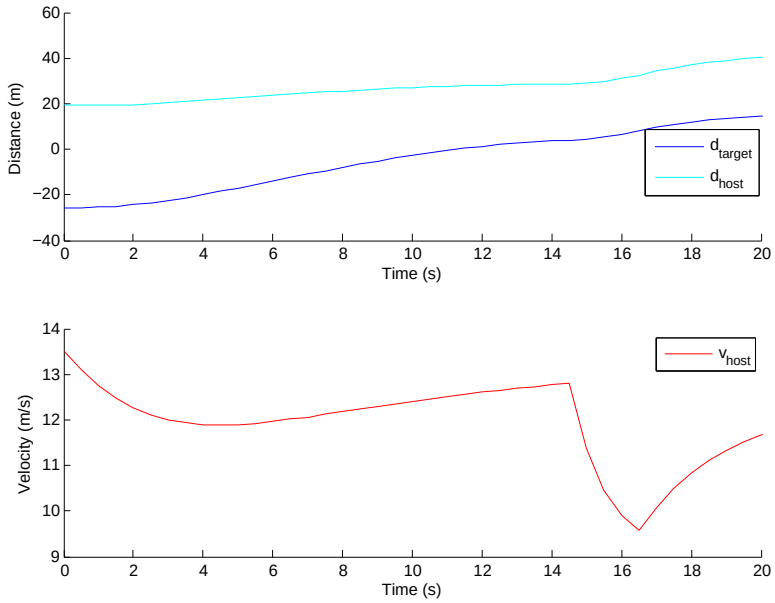


Figure 5.25: The distances and velocities of the vehicles in the second example case for lane change situation.

5.2 Context-based PCB

As shown in Section 5.1.2, prediction of the surrounding vehicles' behavior and reaction to the host vehicle is very important in the decision making process in the PCB algorithm. The prediction model introduced in Section 5.1.2 can represent the traffic in the future very well in normal lane driving scenarios. But in some special cases, the context information should be used to enable better prediction.

For example, in entrance ramp scenarios, the traffic vehicles entering the main road will not only react to the leading cars in front of them, they may consider the cars running on the main road and slow down in advance. In addition, for most entrance ramps, the merging cars from ramps are requested to yield to main road traffic due to traffic signs or other traffic rules. If these contexts information are not considered, it could cause inaccurate prediction of vehicles' behaviors, which may lead to suboptimal decisions made by automated vehicles.

5.2.1 Context-based Prediction

Figure 5.26 shows a block diagram of the host vehicle's decision process with the non-context-based prediction model. The best strategy selected by the PCB algorithm is $\{th1 = 2.25, th2 = 3.0, t_{adj} = 10\}$. As shown in Figure 5.27, if the merging vehicle behaves as expected, it should merge in front of the host vehicle. That is the reason why we select longer time headway to give it more space to merge in.

However, if the merging vehicle decides to slow down when it sees the host vehicle, this strategy will result in poor performance. The result is shown in Figure 5.26 with distances and speeds of cars shown in Figure 5.27.

It is seen that the merging vehicle slows down while the host vehicle slows down as well. This leads to much larger deceleration for both cars than the previous case and closer distance when the merging vehicle finally cuts in behind the host vehicle.

A human driver would be able to identify that there is no need to wait for the merging car because it is much slower than the traffic in the main lane and it needs to yield.

The prediction model is modified if special context knowledge could be used for better pre-

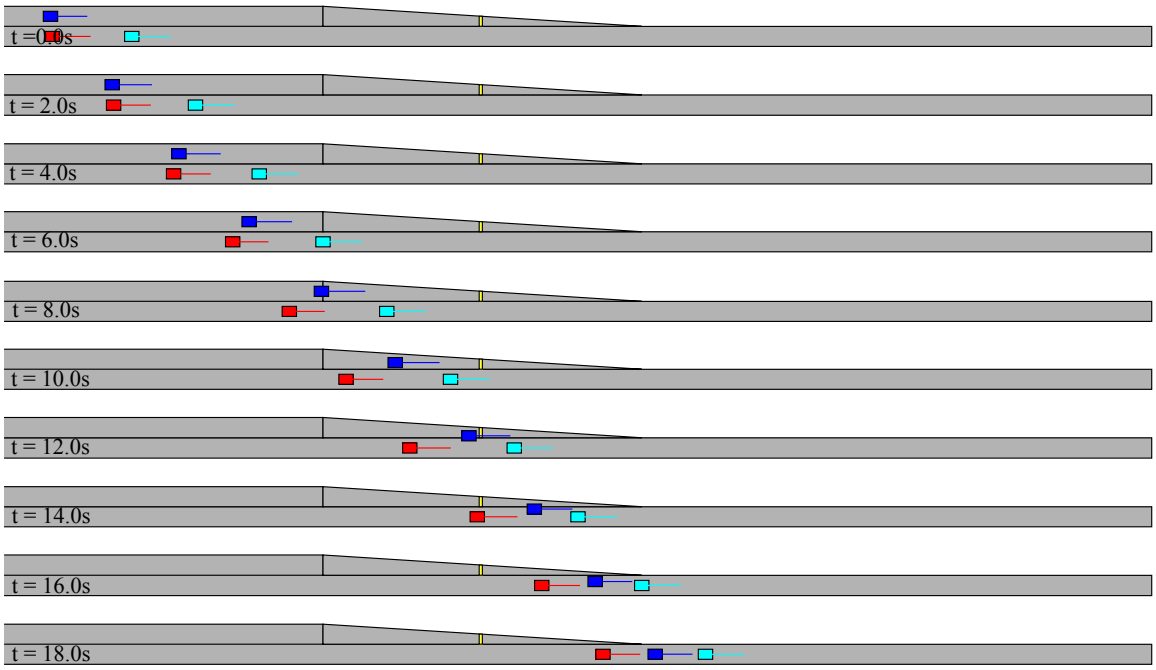


Figure 5.26: The host vehicle finds the best strategy of $\{th1 = 2.25, th2 = 3.0, t_{adj} = 10\}$ with the standard prediction model of the merging vehicle. It is assumed to only care about cars within a lane width in front of it.

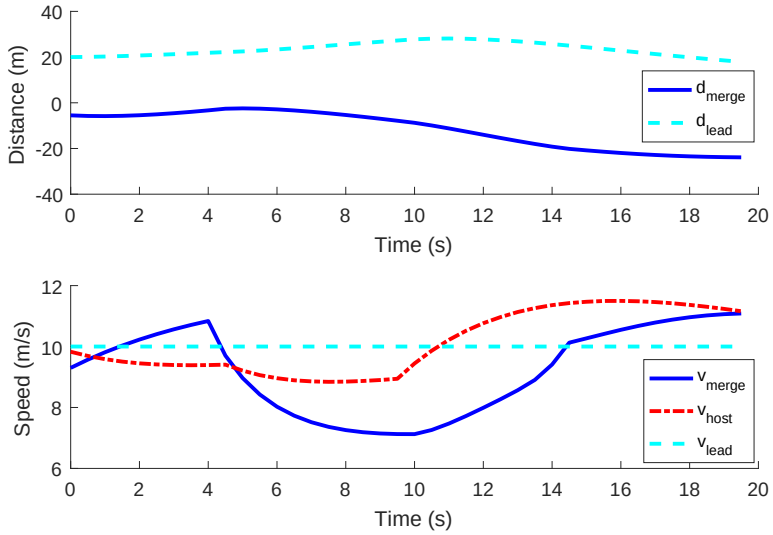


Figure 5.27: Distance and speed profile by applying the optimal strategy PCB algorithm finds assuming the prediction is accurate

diction of vehicles' behaviors:

1: Distance keeping algorithm for V_{hostID}

2: **if** V_{hostID} fits into $Context_j$ **then**

3: Run special context vehicle behavior model

4: $acc_{cmd} = f_{context_j}(V_{1toN})$

5: **else**

6: Run standard vehicle behavior model

7: **end if**

8: Apply maximum acceleration constraint

9: $acc_{cmd} = min(acc_{cmd}, f(v_{host}))$

10: Apply minimum acceleration constraint

11: $acc_{cmd} = max(acc_{cmd}, acc_{emergency})$

One implementation of the context-based PCB is for entrance ramps. We proposed the following algorithm in addition to the default algorithm discussed in Section 5.1.2 to handle context-specific behaviors.

1: Distance keeping algorithm for entrance ramp context

2: **if** $V_{mergeInd}$ is on entrance ramp **then**

3: Idenfity the closest car on main road

4: Generate $d_{closestInd}$ and $v_{closestInd}$

5: Compute t_{est} for both cars

6: $t_{est}[closestInd] = (d_{mergePoint} - lon_{closestInd})/v_{closestInd}$

7: $t_{est}[mergeInd] = (d_{mergeInd} - lon_{mergeInd})/v_{mergeInd}$

8: **if** $t_{est}[closestInd] - t_{est}[mergeInd] < t_{thresh}$ **then**

9: Distance-keep to $V_{closestInd}$

10: **else**

11: Approaching speed limit

12: **end if**

13: **end if**

In the algorithm, $V_{mergeInd}$ and $V_{closestInd}$ are the indexes of the merging vehicle and the car longitudinally closest to the merging vehicle, d and v are the distances and speed of cars, and $d_{mergePoint}$ is the position of the merge point in the PCB algorithm coordinate. In summary, if the closest car on main road will arrive at the merge point within t_{thresh} earlier than the merg-

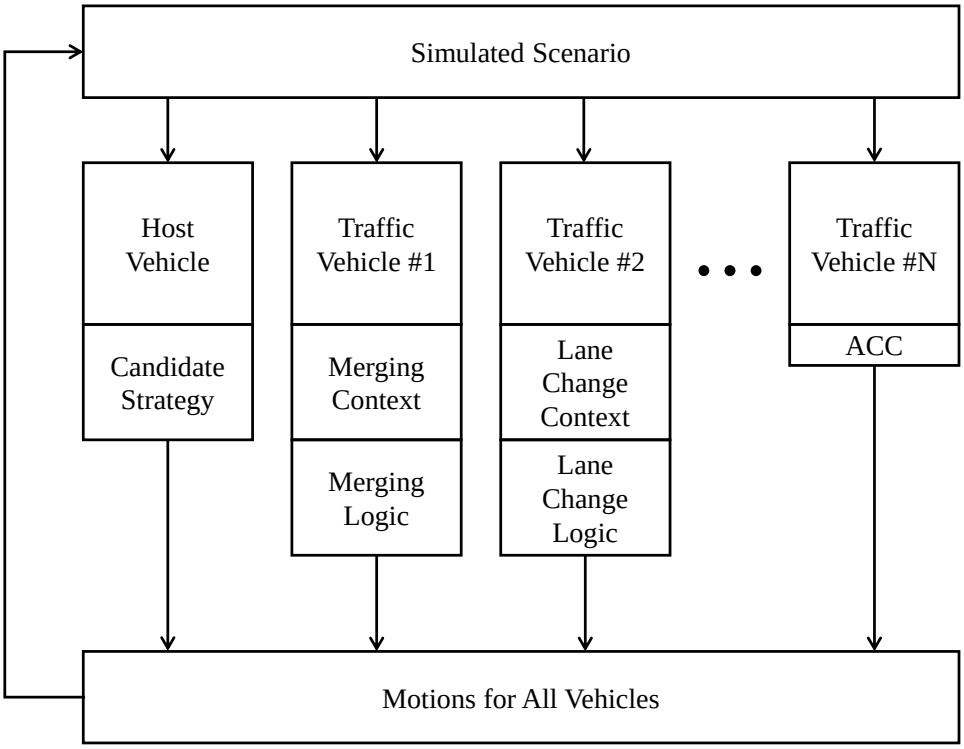


Figure 5.28: Context-based algorithm simulation

ing vehicle, the merging vehicle will consider the the main lane vehicle and perform distance keeping to it. In our experiment, t_{thresh} is set to $-2.0s$. This reflects the fact that merging lanes usually have lower priority and often need to yield to traffic on the main road. If the merging vehicle realizes that the main lane vehicle will arrive at the merge point much earlier, it will just accelerate without considering the host vehicle.

5.2.2 Case Test

A few case tests are implemented to show the result of applying the context-based prediction for PCB algorithm. In the simulation, we assume that the traffic vehicle will behave as predicted as the context-based driver model. The simulation framework is shown in Figure 5.29. Compared with the standard PCB simulation framework, if the traffic vehicle is identified as within some context, such as ”yield at entrance ramp”, the context-based model will apply in the simulation instead of a default distance-keeping model.

For the first case, the same initial condition as that in Figure 5.26 was generated. However, the

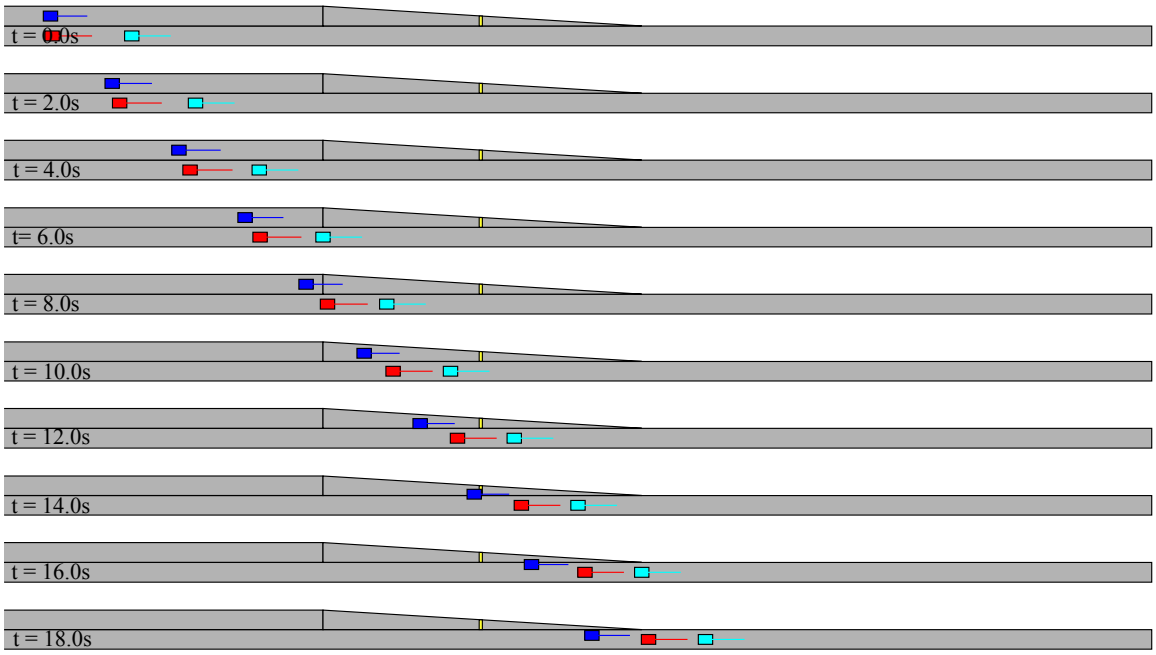


Figure 5.29: Result of applying context-based prediction and context-based PCB algorithm

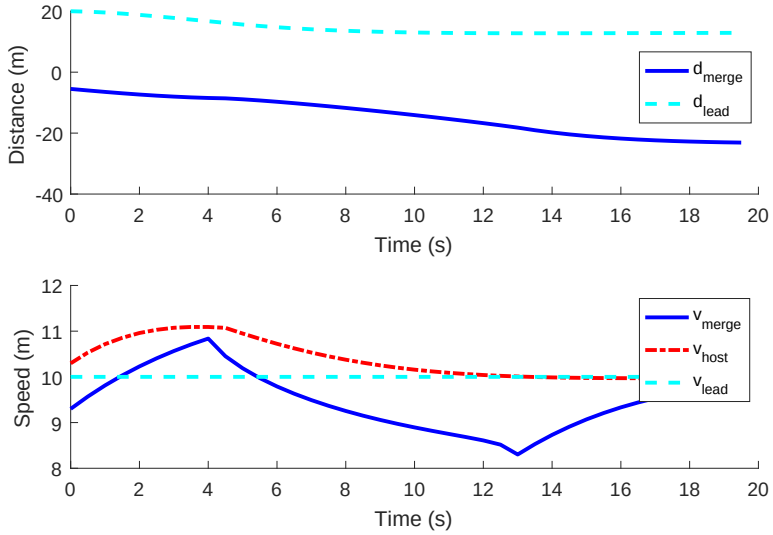


Figure 5.30: Distances and speeds of vehicles indicate smoother cooperation between cars with the context-based PCB algorithm

time context-based PCB algorithm was used to generate the best strategy. Due to its capability of foreseeing the yielding behavior of the merging vehicle, the final strategy is $\{th1 = 0.75, th2 = 1.00, t_{adj} = 10\}$. Therefore, after searching for the best strategy, the host vehicle decides to speed up towards its leading vehicles as shown in Figures 5.29 and 5.30.

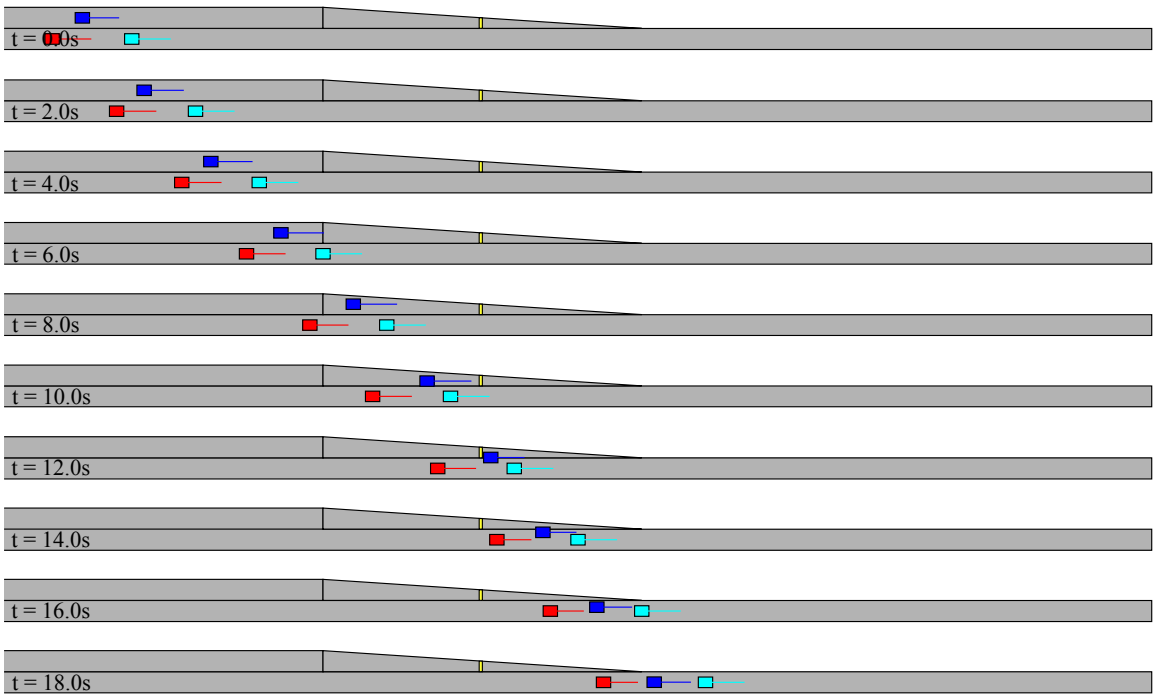


Figure 5.31: Another example of applying context-based PCB. The merging vehicle's starting position is a bit more forward than that of the previous example. Therefore, the condition for the yield-at-entrance-ramp context was not met. The merging car is predicted to perform normal distance keeping.

Figures 5.31 and 5.32 show another scenario using the context-based PCB. The initial condition is similar to the previous example, except that the initial position of the merging car is closer to the merge point. When the PCB algorithm starts to run the context-based prediction, it believes that the merging vehicle will not yield to the main lane cars. Therefore, a slow-down command is issued at the beginning to better provide the space needed by the merging car.

Compared with the entrance ramp handling logic described in the baseline performance Section 4.4.1, the context-based PCB algorithm is able to make the decision to either speed up to make the merging vehicle cut in behind it or slow down to give more space for the merging vehicle. All the decisions are made based on prediction of future scenarios, therefore, it is more adaptive to a wider range of scenarios compared to the algorithm using constant rules. Compared with the standard PCB algorithm, the accuracy of prediction of behaviors for traffic vehicles has been improved using the context-based algorithm. Therefore the overall quality of the strategy PCB the algorithm picks is improved.

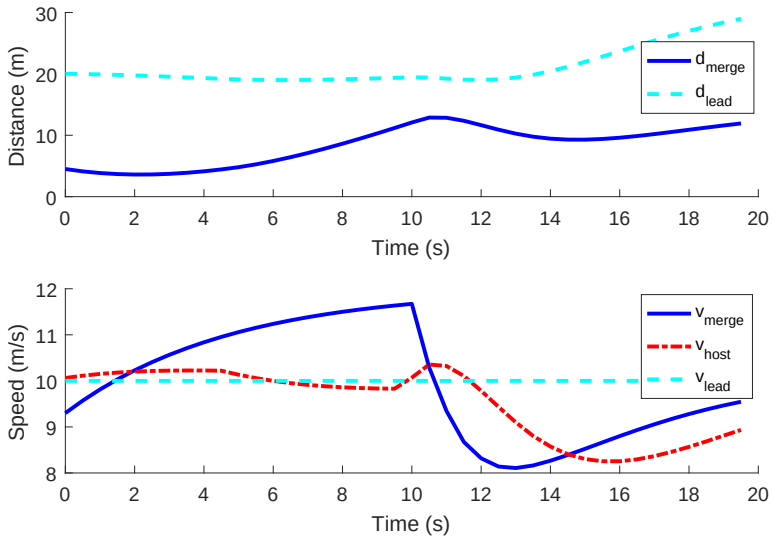


Figure 5.32: Distances between cars and speed profile for the case test in which the merging vehicle does not satisfy the yield-at-entrance-ramp context

5.3 Intention-integrated PCB

In Seciton 5.2, an improvement of the basic traffic scenario prediction is proposed to boost the performance of the basic PCB algorithm. In some special contexts such as entrance ramps, the behavior of the merging vehicle will be predicted differently based on the computing of its estimated time of arrival compared to the host vehicle. However, the prediction does not capture the uncertainties of all the potential behaviors of the surrounding cars. Taking the entrance ramps as an example, some cars may be more conservative, so that eventhough they could arrive at the merging point earlier, they may decide to yield. Or sometimes, even they are slower at the beginning, they may aggressively speed up to cut in front of the cars in the main lane.

For context-based prediction, if the prediction is accurate as shown in Figure 5.29, the autonomous vehicle will make a smart decision to cooperate with the merging car. However, if the context it identified is wrong, then the host vehicle will make bad decisions. One example is shown in Figure 5.33. In this example, the merging vehicle wants to yield to the main lane vehicles. However, the host vehicle believes it will arrive at the entrance ramp just a bit over the context-based PCB's threshold of context condition t_{thresh} because of a slightly higher speed at the beginning of the scenario.

Figure 5.34 shows the speed profiles throughout the whole simulation. It is seen that the host vehicle first slows down to let the merging vehicle go first. At $t = 6 - 8s$, it actually decelerates dramatically. However, the merging vehicle decelerates more. This causes a social conflict where both vehicles misinterpret each other's behavior and therefore make a non-optimal decision.

Figure 5.35 shows that from $t = 6-8s$, the cost becomes infinite because when both vehicles are confused about each other's behavior, they get closer to the ramp and start to move laterally without fully resolving the longitudinal spacing between them.

The example introduces an extreme case of the effect of wrong context-based prediction. Without capturing the uncertainty or possibility of wrong context identification, potentially dangerous situations can not be well evaluated in the PCB algorithm. If the host vehicle is very confident about how the surrounding vehicle will drive and react to its behaviors, it will perform differently compared to if the surrounding vehicle's behaviors have a lot of uncertainty.

To achieve this more human-like driving behavior, the autonomous vehicle needs to: first,

capture surrounding vehicles' intentions instead of depending on some simple rules; second, take the uncertainty of the prediction of surrounding vehicles' behavior into account.

Based on these motivations, an modified PCB algorithm is proposed. The candidate strategy generation steps are the same as for the basic PCB algorithm. The prediction engine is improved to first estimate the intention and corresponding probability for each possible intention. Then the prediction is applied for each possible intention of the traffic vehicles, which will be discussed in Section 5.3.2. The cost-function based evaluation is also modified to be able to capture the probability and compute the best strategy under this uncertainty, which will be discussed in Section 5.3.3.

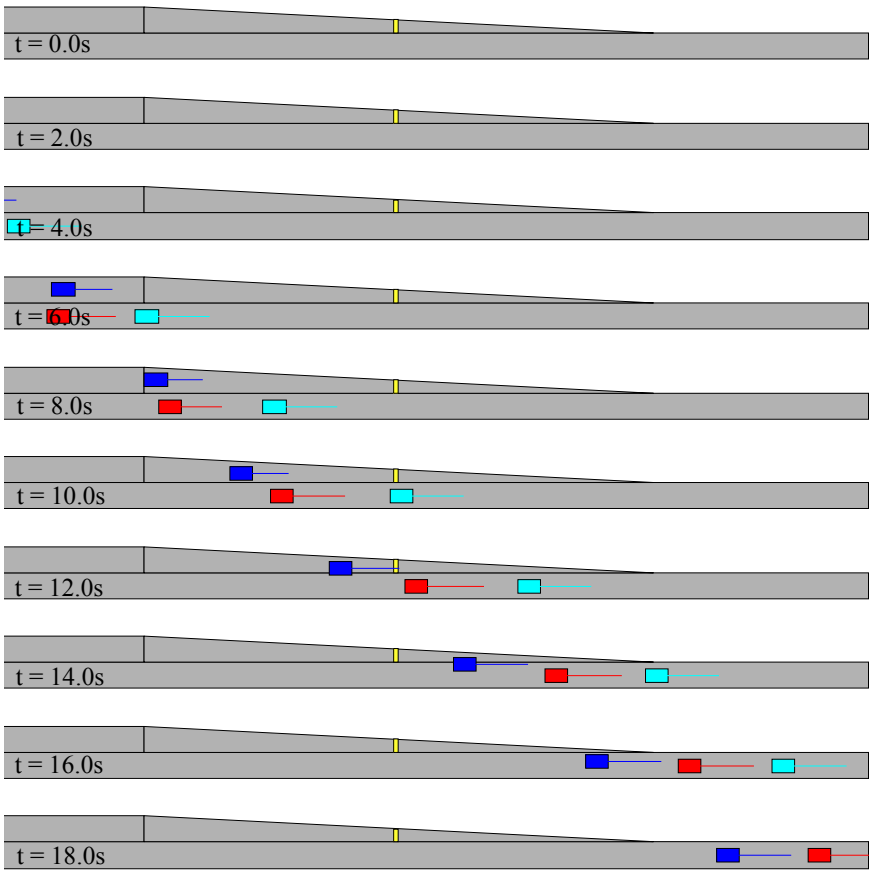


Figure 5.33: Context-based prediction could be wrong due to different driver preferences or styles. This causes the host vehicle to select a strategy that will cause a social conflict, where cars misunderstand each other's motion and make the wrong decision based on the wrong assumption.

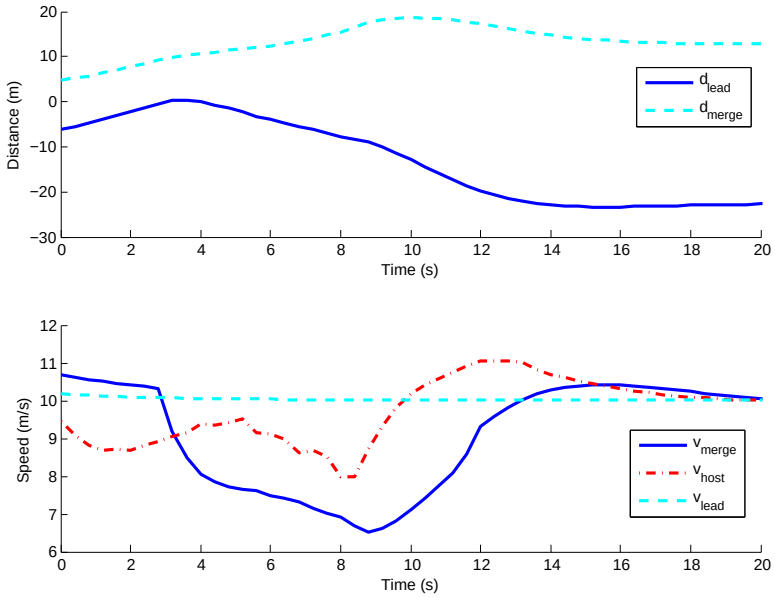


Figure 5.34: In the simulation of context-based PCB with a wrong prediction, the host vehicle's speed will have more jerks with larger amplitude of deceleration and acceleration.

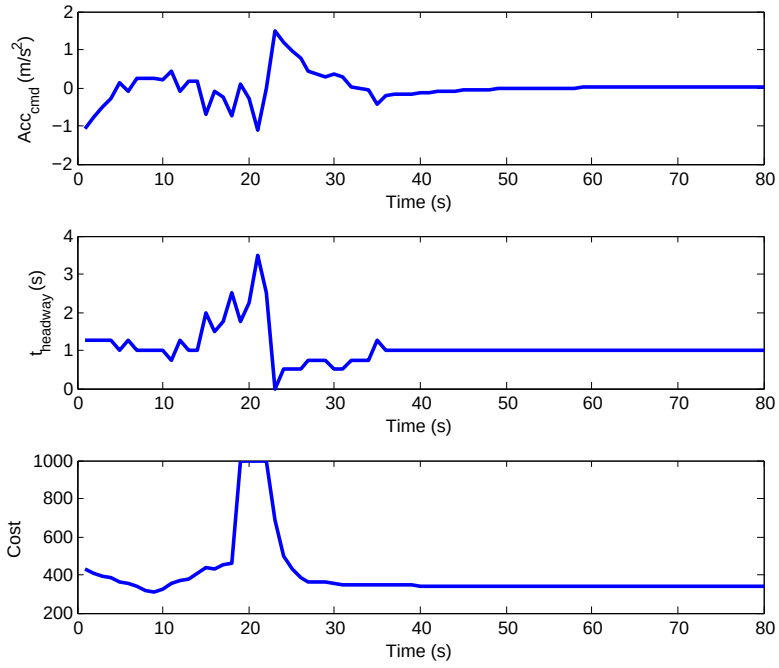


Figure 5.35: If the context identified by the context-based PCB is wrong, the algorithm will lead the vehicle into uncomfort or even unsafe situation with very high or infinite cost. And the algorithm failed to avoid the dangerous situations ahead of time because of the wrong prediction.

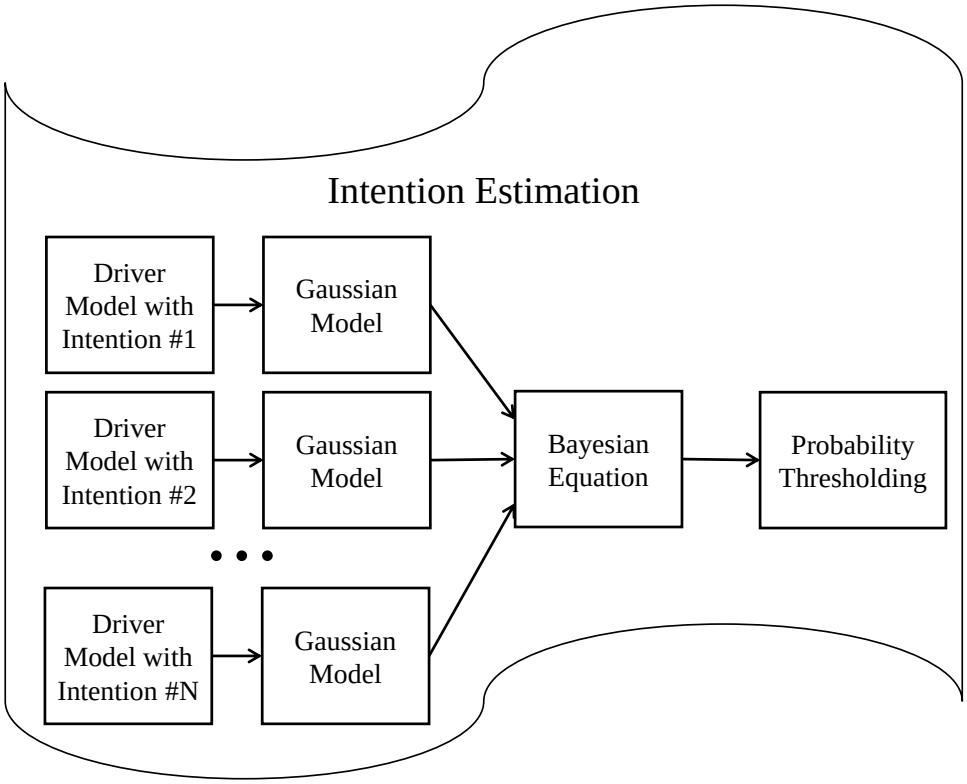


Figure 5.36: Intention-based PCB algorithm simulation

5.3.1 Intention-Estimation

As discussed in Section 2, there are multiple approaches to identify the human driver's intention or future behavior. In this thesis, a Bayesian model is used to classify the intention of the traffic vehicles and also generate the level of confidence of the estimation.

Equation 5.10 shows the computing of the probability of intention of the surrounding cars.

$$prob(i_n|Act) = prob(Act|I_n)/(sum_{i=1}^{N} prob(Act|I_i) \tag{5.10}$$

Here, I_n is the n^{th} intention of the vehicle and Act is the action of the vehicle. Figure 5.36 shows a Bayesian framework for the surrounding vehicle intention estimation and behavior prediction.

As shown in Figure 5.36 and Equation 5.10, to get $prob(i_n|Act)$, first a behavior model $Act|I_n$ is needed to compute if an intention is given to the human driver, how would she/he behave in the traffic environment. The second step is to convert the deterministic human behavior model $Act|I_n$ into a probabilistic behavior model $prob(Act|I_n)$. In this step, we made the assumption

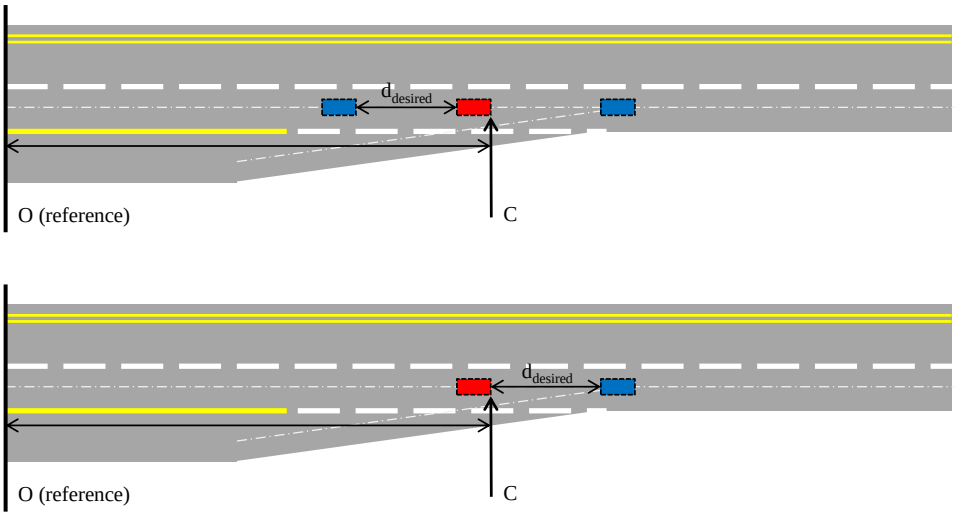


Figure 5.37: Merge driver model with intention

that the human driver's actual behavior will be distributed normally around the behavior model $Act|I_n$. Therefore, a Gaussian distribution is computed as shown in Equation 5.11.

$$prob(Act_{act}|I_n) = pdf_{gaussian}(\overline{Act_{mean}}, \lambda) \tag{5.11}$$

$$= e^{frac-(Act_{act}-Act_{mean})^2 2\lambda^2} \tag{5.12}$$

The general algorithm to compute the probability of surrounding vehicle intentions is introduced in this section. For different scenarios, different driver models need to be created to achieve the intention estimation.

Entrance Ramp As discussed before, the intention of yield or not yield of the merging vehicle on the entrance ramp is a dominant factor for their future behavior. In the example domain, the autonomous vehicle is driving on the main road while human-driven vehicles are merging in from entrance ramps. Human drivers can either cut in front of or yield to the host vehicle.

The entrance ramp driver model is shown in Figure 5.37. Given our assumption that the merging vehicle will merge in along a determined path, which is the centerline of the entrance ramp, the only intention-based behavior is the acceleration of cars.

The driver model of a merging vehicle is defined in Algorithm 5.3.1.

1: Intention-based merging vehicle driver model $Act_{merge}|I$

2: Identify the closest car on main road

3: Generate $d_{closest}$, $v_{closest}$

4: **if** $I = Yield$ **then**

5: Compute t_{est} for both cars

6: $t_{est}[closestInd] = (d_{mergePoint} - lon_{closestInd})/v_{closestInd}$

7: $t_{est}[hostInd] = (d_{mergePoint} - d_{desired} - lon_{hostInd})/v_{hostInd}$

8: Compute acc_{cmd}

9: $acc_{cmd} = k_a(t_{est}[closestInd] - t_{est}[hostInd])$

10: **else if** $I = NotYield$ **then**

11: Compute t_{est} for both cars

12: $t_{est}[closestInd] = (d_{mergePoint} - lon_{closestInd})/v_{closestInd}$

13: $t_{est}[hostInd] = (d_{mergePoint} + d_{desired} - lon_{hostInd})/v_{hostInd}$

14: Compute acc_{cmd}

15: $acc_{cmd} = k_a(t_{est}[closestInd] - t_{est}[hostInd])$

16: **end if**

17: Apply maximum acceleration constraint

18: $acc_{cmd} = min(acc_{cmd}, f(v_{host}))$

19: Apply minimum acceleration constraint

20: $acc_{cmd} = max(acc_{cmd}, acc_{emergency})$

This model is based on the logic that if the merging vehicle intends to yield to the host vehicle, it will adjust its speed by applying proper acceleration command. The target for the merging vehicle is to arrive exactly at the desired distance to the vehicle in the main road when it merges in, as shown in Figure 5.38. Instead of using the merge point in the entrance ramp geometry directly, we compute the first point that the merging and main lane car will likely to have a collision as shown in Equation 5.13 based the assumption that both cars follow the center of the main and merging lane.

$$d_C = d_A + \frac{w_{lane} - w_{car}}{w_{lane}}(d_B - d_A) \tag{5.13}$$

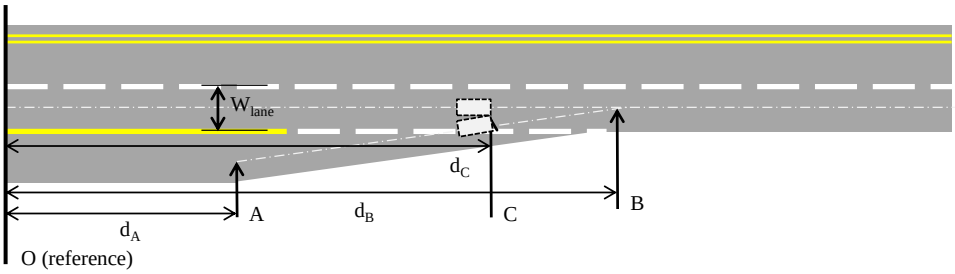


Figure 5.38: The location of the possible collision point, where the merging vehicle and main lane vehicle need to keep a minimum desired clearance.

If the intention is to yield, then the merging vehicle should arrive at $d_C - d_{desired}$ when the main lane vehicle reaches d_C. In contrast, if the intention is to not yield, then the merging vehicle is targeting $d_C + d_{desired}$ when the main lane vehicle reaches d_C. To achieve this goal, the merging vehicle will adjust its speed based on the difference in the estimated time of arrival. An acceleration is then computed for the merging vehicle. Max acceleration/deceleration and speed limit are also considered in the computation of the merging vehicle's acceleration.

An additional rule we applied to the driver model is that if the estimated time of arrival for the merging vehicle and host vehicle differ by a lot, the intention will not be executable. Therefore, the intention will be overridden in those cases. For example, if the host vehicle in the main lane is much faster and further forward than the merging vehicle, then it makes sense to assume that the merging vehicle will have to merge in behind the host vehicle. Even if an intention of not-yield is given, the merging vehicle is not able to accelerate fast enough to execute the intention. This constraint is applied as shown in the following algorithm:

1: Compute t_{est} for both cars
2: $t_{est}[closestInd] = (d_{mergePoint} - lon_{closestInd})/v_{closestInd}$
3: $t_{est}[hostInd] = (d_{mergePoint} + d_{desired} - lon_{hostInd})/v_{hostInd}$
4: **if** $t_{est}[closestInd] - t_{est}[hostInd] > t_{NotYieldThresh}$ **then**
5: $\quad prob(Yield) = 0$
6: **else if** $t_{est}[closestInd] - t_{est}[hostInd] < t_{YieldThresh}$ **then**
7: $\quad prob(Yield) = 1$
8: **end if**

Using this driver model, even with the same initial scenario, if different intention is given, the

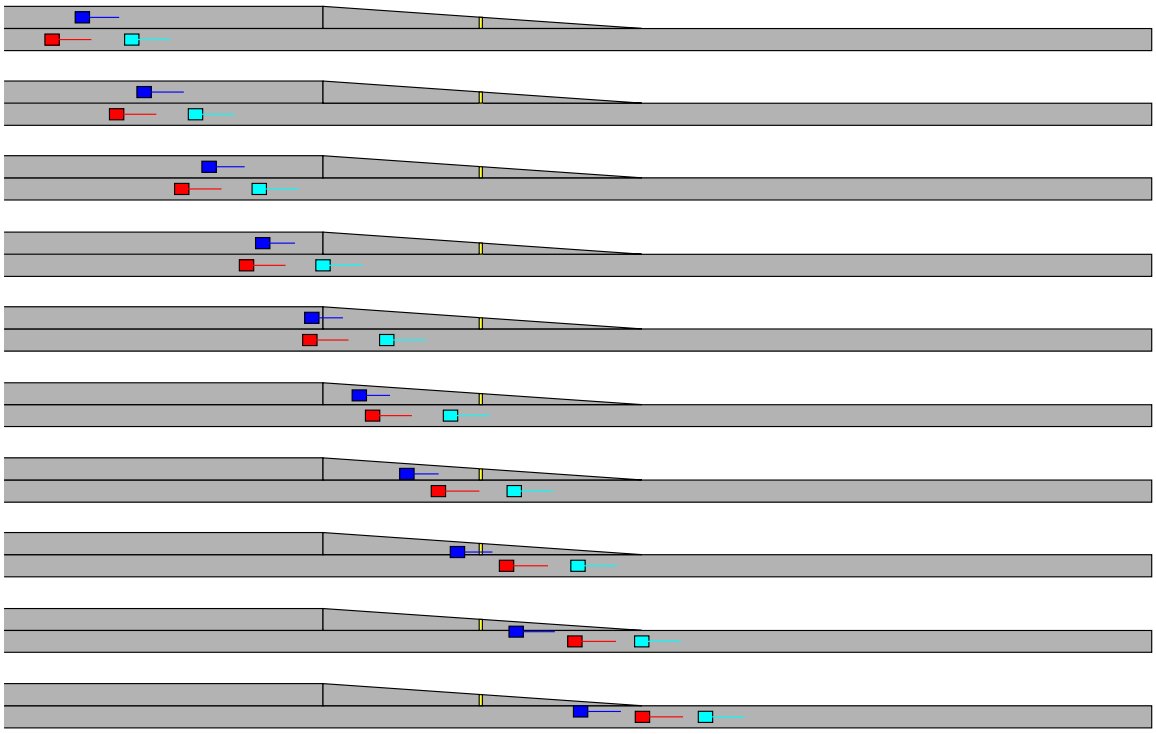


Figure 5.39: Simulation of merging vehicle driver model with yield intention

result of the human driver behavior will be very different. Figures 5.39 and 5.40 show that if the intention is to yield, the merging vehicle will slow down for the main lane traffic. The vehicles in the main lane are simulated using the default distance-keeping model. The speed profile of the merging car shows that before it arrives at the entrance ramp region at $t = 3s$, the merging car is accelerating. When it reaches the entrance ramp region, it starts to adjust its position to make sure it keeps proper distance to its leader after merging in. Eventually at around $t = 13s$, the merging car gets into the main lane with the desired distance to it.

In contrast, if the intention is to not yield, the merging vehicle will speed up to get in front of the traffic in the main lane, as shown in Figures 5.41 and 5.42. The speed profiles for the first few seconds are the same as for the yield scenario. Whenever the merging vehicle enters the entrance ramp region, instead of slowing down, its t_{est} drives it to keep the higher speed. Eventually, at $t = 10.5$, the vehicle in the main lane starts to distance-keep to it.

Lane Change In lane change cases, after realizing the host vehicle's lane change intention from the turn signal, there could be mainly two intentions for the neighboring vehicle. It could

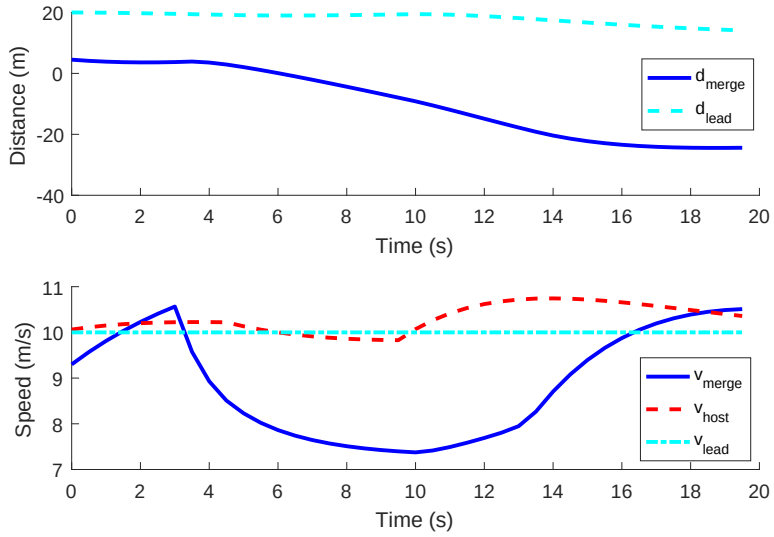


Figure 5.40: Distances and speeds of the simulated scenarios to test the driver model with yield intention

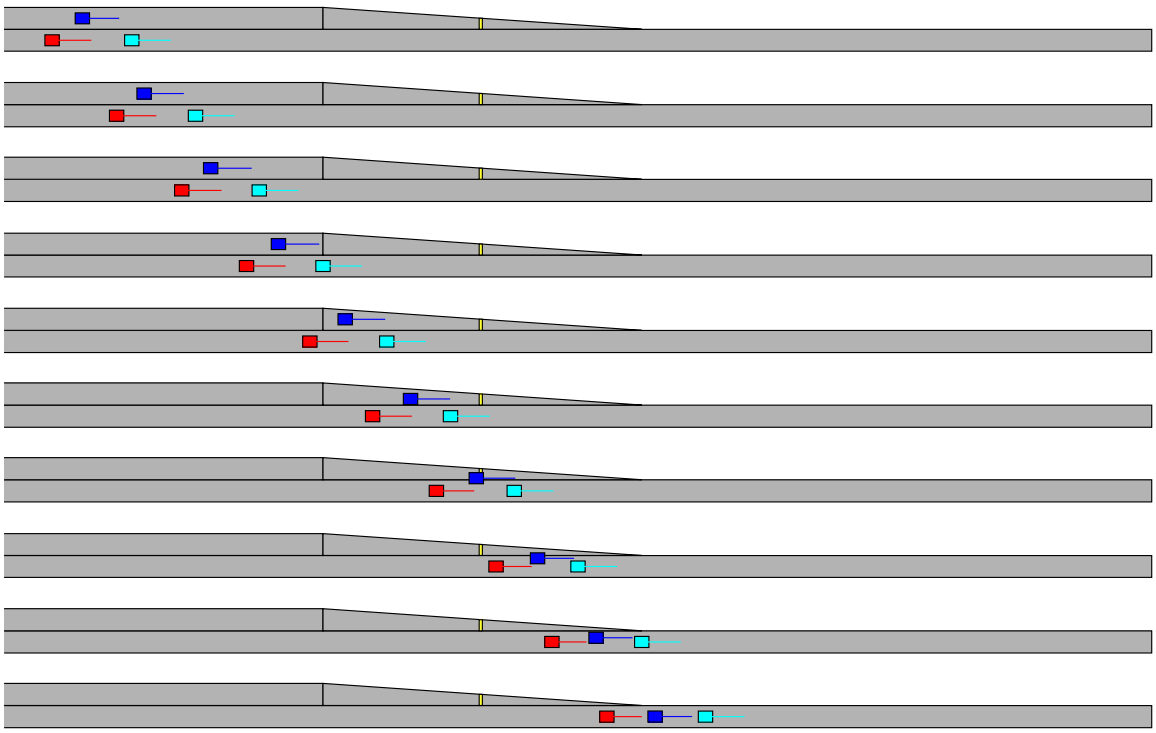


Figure 5.41: Simulation of merging vehicle driver model with not-yield intention

either actively yield to the host vehicle that intends perform lane change; or it could ignore the host vehicle's potential lane change behavior by either accelerating or keeping its current speed, as shown in Figure 5.43.

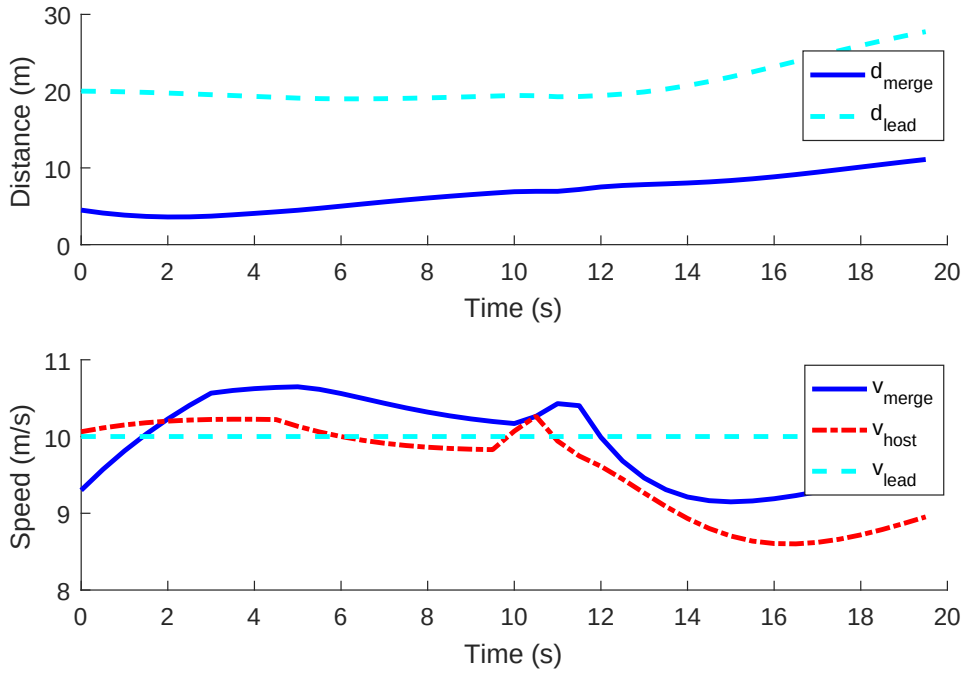


Figure 5.42: Distances and speeds of the simulated scenarios to test the driver model with not-yield intention

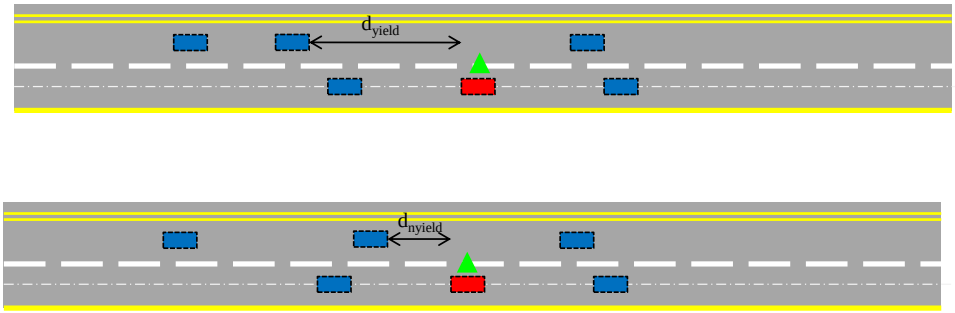


Figure 5.43: Lane change intention model

The driver model for the neighboring vehicle reacting to the host vehicle's lane change is described in the following algorithm.

1: Intention-based targeted lane vehicle driver model $Act_{intendLane}|I$

2: Identify cars plan to perform lane change

3: Generate $d_{laneChange}$, $v_{laneChange}$

4: Identify leading car in current lane

5: Generate d_{lead}, v_{lead}

6: **if** $I = Yield$ **then**

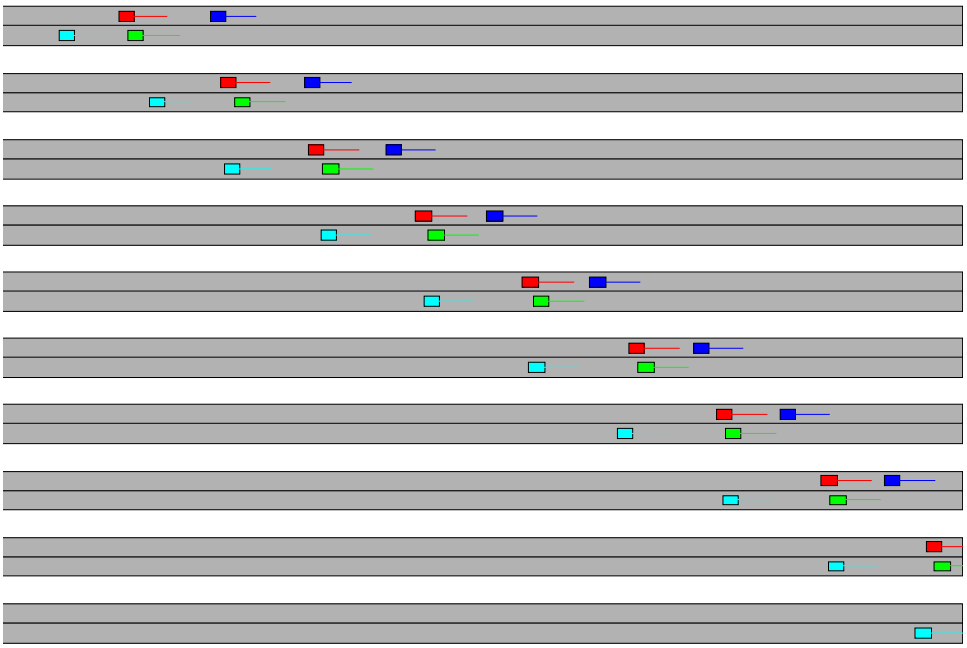


Figure 5.44: In lane change scenario, the driver model of the vehicle (cyan car) in the target lane with a given intention of yielding

$acc_{lanechange} = f(d_{lanechange}, v_{lanechange})$

$acc_{normal} = f(d_{lead}, v_{lead}, th_{desired} = 1.5th_{normal})$

$acc_{cmd} = min(acc_{lanechange}, acc_{normal})$

else if $I = NotYield$ **then**

$acc_{cmd} = f(d_{lead}, v_{lead}, th_{desired} = 0.5th_{normal})$

end if

Apply acceleration constraints

Figure 5.44 and Figure 5.45 show the result of the simulated scenario using the human driver model. The host vehicle wants to perform a right lane change. It starts with the turn indicator on to signal to surrounding cars. In the simulation, the host vehicle's PCB strategy is $\{th1 = 1.0, th2 = 1.0, t_{adj} = 5.0, t_{lanechange} = \infty\}$. Therefore, it will just perform default distance keeping without lateral movement.

The neighboring vehicle will observe the turn signal and start to react to it. The sequence of contexts shows that given different intention, the human-driven cars will either yield to the host vehicle by giving it a gap to merge into or intentionally speeding up towards its leader, leaving no

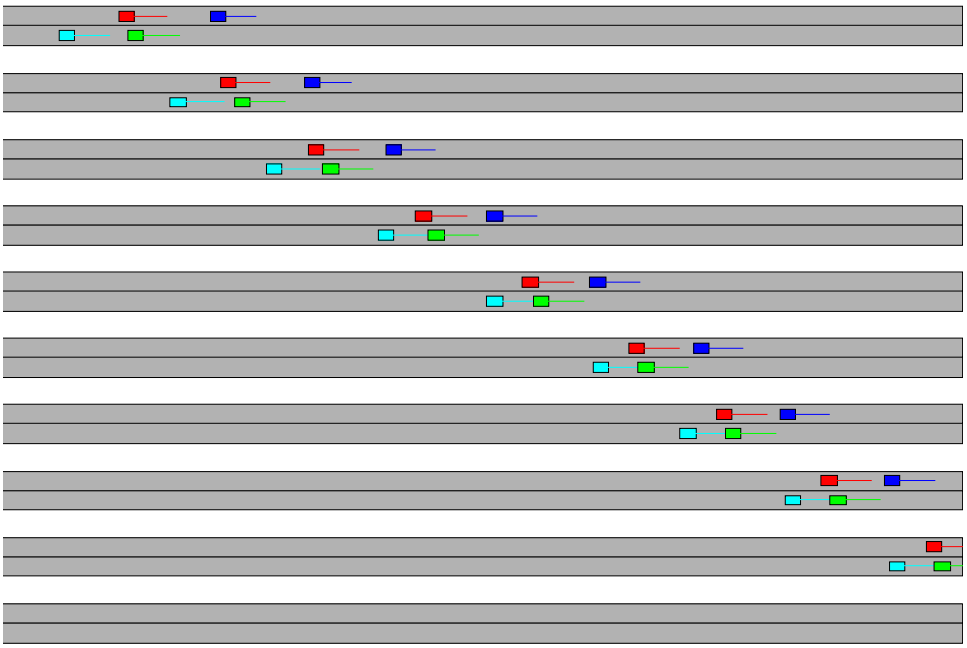


Figure 5.45: In lane change scenario, the driver model of the vehicle (cyan car) in the target lane with a given intention of not-yielding

gap for the host vehicle to merge into. The speed-up and slow-down are performed by adjusting the desired distance to the traffic vehicle's leading car.

After generating the human driver model of lane change, the probability of the neighboring vehicles' intentions can be generated using Equation 5.11.

5.3.2 Intention-based Prediction

In the prediction step, the probability of intention for each traffic vehicle is estimated. As discussed in Section 5.1.2, we want to predict the surrounding vehicles' reactions to each of the host vehicle's strategies. But given that there are multiple intentions that surrounding vehicle could have, instead of running the prediction engine only once for each strategy, we need to run it N times, where N is the number of different combinations of intentions surrounding vehicles could have as shown in Equation 5.14.

$$scenario(i, t + \delta t|I_n) = f_{predict}[scenario(i, t|I_n), i, I_n] \tag{5.14}$$

Here, i is the i^{th} strategy, I_n is the n_{th} the combination of intentions, $scenario(i, t + \delta t|I_n)$ is the scenario using strategy i with the intention I_n and $f_{predict}$ is the iPCB prediction model. In this case, if the intention can be identified for a vehicle, the intention-based driver model described in Section 5.3.1 is used. If no intention can be identified, the default driver model with and without context information is used, as described in Section 5.2 and 5.1.2

5.3.3 Cost function-based Evaluation under Uncertainties

The decision making algorithm needs to take into consideration the probability of the human driver's intentions and the prediction results corresponding to each possible human driver intention.

$$C_{strategy(i)} = C_{strategy(i)|I_n} p(I_n) \tag{5.15}$$

$$C_{strategy(i)|I_n} = \sum_{t=0}^{t_{predict}} C_{scenario(i,t)|I_n} \tag{5.16}$$

$$\begin{aligned} C_{scenario(i,t)|I_n} = C_{speed}(scenario(i,t)|I_n) + C_{comfort}(scenario(i,t)|I_n) + \\ C_{safety}(scenario(i,t)|I_n) + C_{fuel}(scenario(i,t)|I_n) \end{aligned} \tag{5.17}$$

The cost function computing Equation 5.4 can be modified to compute the expectation of the strategy cost given the combination of the probabilities of different intentions as shown in Equation 5.15.

5.3.4 Case Tests

The iPCB has been implemented in both stand-alone simulation and an autonomous driving vehicle. To demonstrate the performance of the iPCB algorithm, a few case tests are presented

For the iPCB testing, the simulation framework is changed to the configuration shown in Figure 5.46.

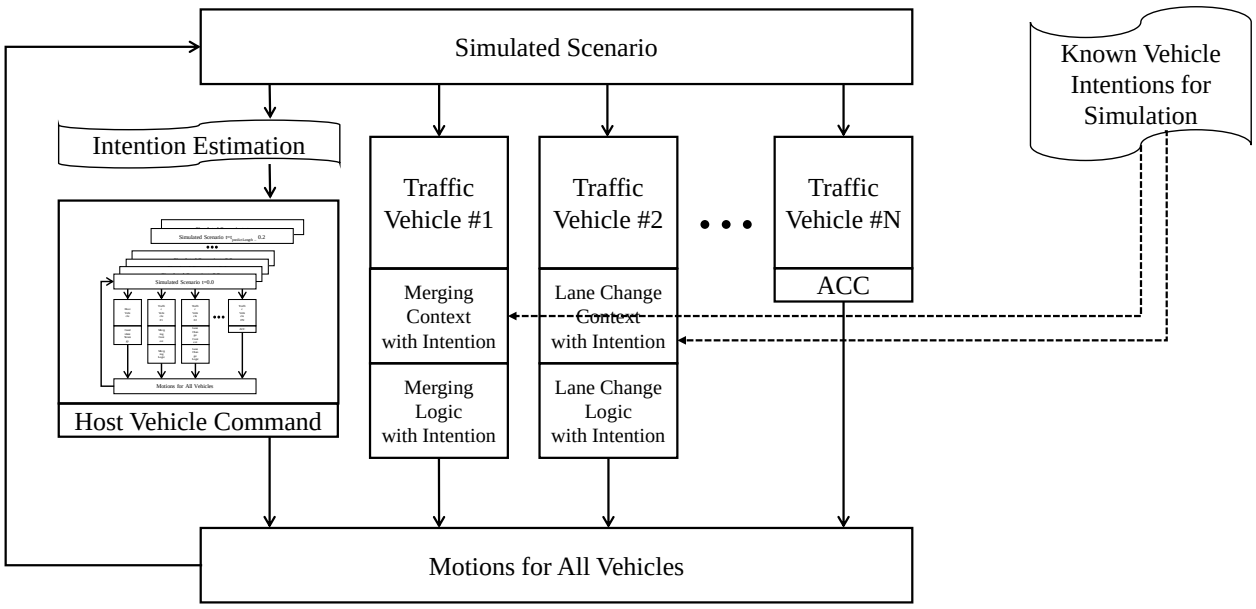


Figure 5.46: Intention-based PCB algorithm simulation

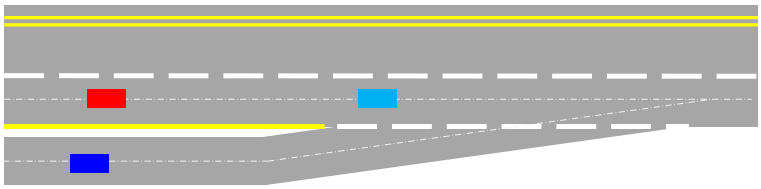


Figure 5.47: Entrance ramp scenario for case test

In the simulation steps, the intentions of vehicles are known as system input. Therefore, if the vehicle's context fits into the intention-based prediction model, the intention will be used to affect the simulated traffic vehicle's behavior. In the iPCB module, the intention information is not known. Therefore, the host vehicle has to estimate other vehicles' intentions based on surrounding vehicle's position, speed and acceleration inputs. The iPCB algorithm will generate a time headway command th_{cmd} for both the entrance ramp and lane change scenario and a relative lateral velocity command v_{lat} for lane changes. After running the iPCB algorithm, a suggested acceleration command will be generated to control the host vehicle.

Entrance Ramp In the entrance ramp scenario example, the initial condition is set to be as shown in Figure 5.47. There are three cars in this interaction. The host vehicle starts with a normal distance following its leading car, running at the preferred speed on the highway. Mean-

Table 5.1: Comparison of iPCB with context-based PCB

I_{act}	$iPCB$	$cPCB_{correct}$	$cPCB_{wrong}$
Yield	592.14	504.52	685.59
Not Yield	555.74	559.07	inf

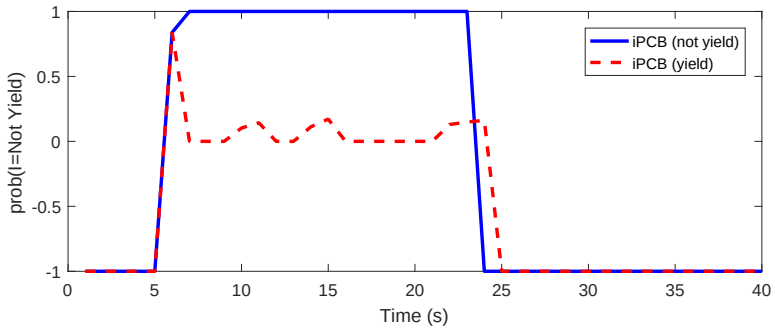


Figure 5.48: Intention estimation result using the iPCB algorithm with merging vehicle intention of yield and not-yield

while, there is a fast-moving merging car on the ramp. It is running at a similar speed to the host vehicle with the same longitudinal position and unknown intention.

Based on the context-based PCB algorithm, the host vehicle could assume the car will yield and make a corresponding decision. Alternatively, the host vehicle could assume the car will not yield. However, as discussed before, these assumptions may not be correct and could cause the car to make a non-optimal decision. Table 5.1 shows the total strategy cost for applying the best strategy under an estimated vs. the real intention of the merging vehicle, where I_{act} is the actual intention of the merging vehicle. As shown, if the merging vehicle actually will yield, then the cPCB algorithm with the yield intention achieves the lowest cost. However, if the merging vehicle actually would not yield, the cPCB algorithm can even generate an strategy that causes the host vehicle to make a unsafe decision with an infinite total scenario cost. Compared with the cPCB algorithm, the iPCB algorithm will select a strategy that works for both intentions of the merging vehicle. It will not achieve the optimal performance for a specific merging vehicle behavior, but it also will not cause the car to suffer from the wrong decision.

The intention estimation result for this case is shown in Figure 5.48. In the figure, -1 means the merging vehicle or host vehicle are not in the entrance ramp region, therefore intention estimation is not available. When both vehicles are in the entrance ramp region, as can be seen,

Table 5.2: Best strategy for one step using different variants of the PCB algorithm

	$iPCB$	$cPCB_{correct}$	$cPCB_{wrong}$
$th_1(s)$	2.75	0.5	3.50
$th_2(s)$	1.75	1.25	1.25
$t_{adj}(s)$	10	10	5

using the Bayesian rule, the host vehicle can quickly figure out the probability of the merging vehicle's intention. For some cases, such as the red line in Figure 5.48, $iPCB_{yield}$ shows that the iPCB algorithm needs to consider both whether the merging vehicle wants to yield (with higher probability) and not-yield (with lower probability).

Taking the iPCB decision making as an example, the best strategy chosen by the context-based PCB algorithm with different intention assumptions regarding the merging car and the best strategy of the iPCB algorithm are shown in Table 5.2. The optimal strategy for this situation is different from either of the best strategies with a determined intention assumption. It is more conservative than if the host vehicle knows for sure that the merging vehicle will yield to us, that the average time headway for the optimal strategy is $2.25s$ compared to $0.875s$. But it is a bit more aggressive than the strategy chosen by assuming the merging vehicle will not-yield to us of which the average time headway is $2.375s$. The balance between these two assumptions is based on the weight-sum of the cost using the identified probability of $prob_{yield} = 0.897$ in this step.

Figure 5.49 shows the result of executing the context-based iPCB algorithm in simulation for $20s$ with a replanning interval of $0.5s$. Figure 5.50 shows the speeds and distances between cars during the iPCB algorithm's case test in simulation. It can be seen that starting at $t = 5s$, the host vehicle begins to react to the merging vehicle by first slowing down a little bit because of the uncertainty of what its future behavior will be. Figure 5.48 shows that there is a 10% probability that the merging vehicle will not yield to us. Then at $t = 7s$ the host vehicle is more convinced that the merging vehicle will yield, and therefore starts to accelerate.

One of the main reasons why the iPCB algorithm works better is that it can consider the effect of each strategy with different reactions from the surrounding vehicles. The iPCB algorithm gives the vehicle the right level of conservativeness to avoid the risk of making a wrong decision based on a single optimal assumption of what others will do.

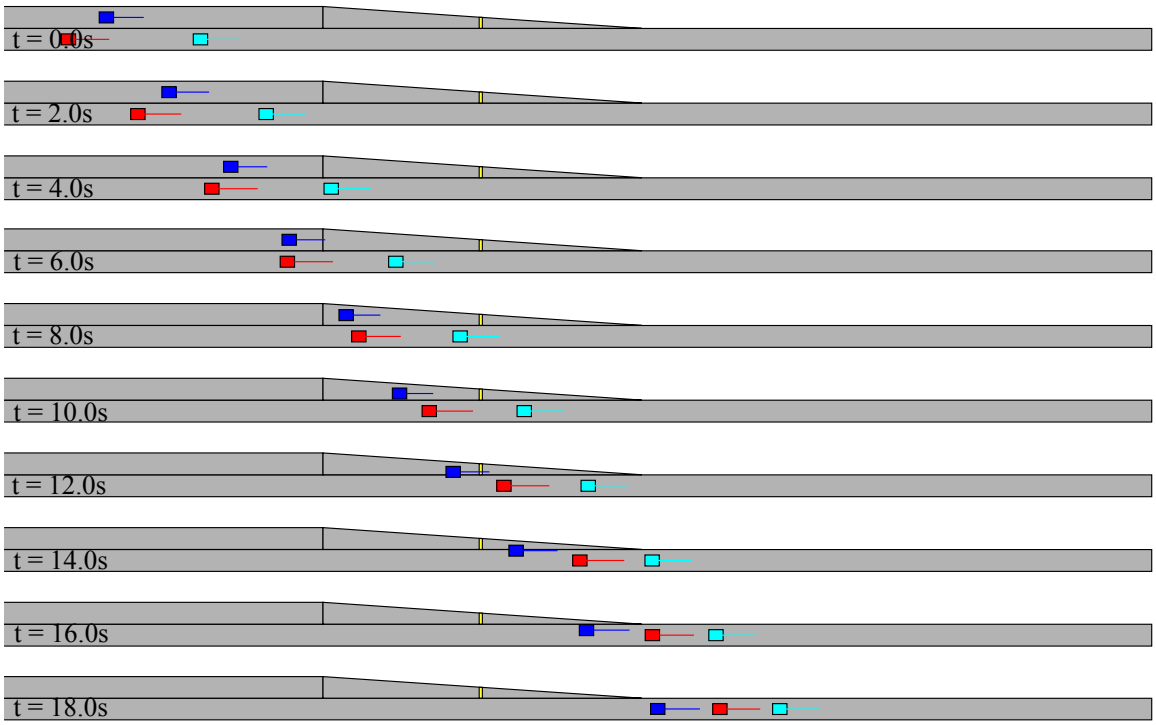


Figure 5.49: Running the iPCB algorithm in a case in which the merging vehicle wants to yield to the host vehicle

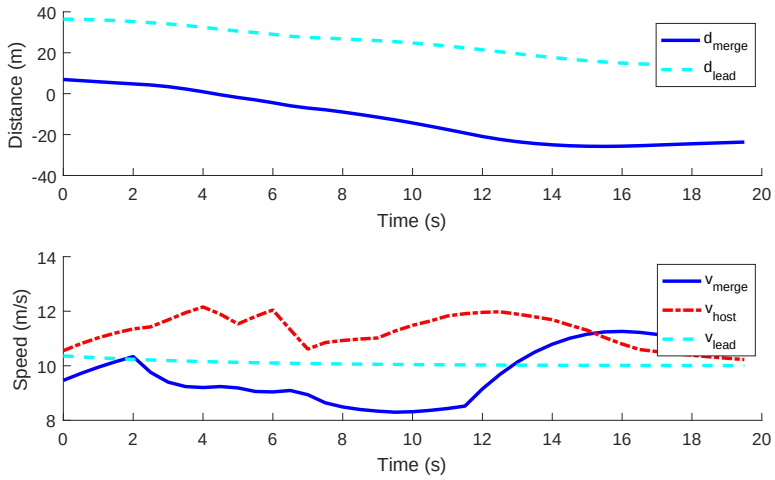


Figure 5.50: Speed and distances between cars in the iPCB algorithm case test in which the merging vehicle wants to yield to the host vehicle

Figure 5.51 shows another example of the iPCB algorithm with a slightly different initial condition where the merging vehicle is assigned with an intention of yield. Figure 5.54 shows the speed profile of the host vehicle in executing this strategy.

It can be seen that the host vehicle decides to speed up to get in front of the merging vehicle.

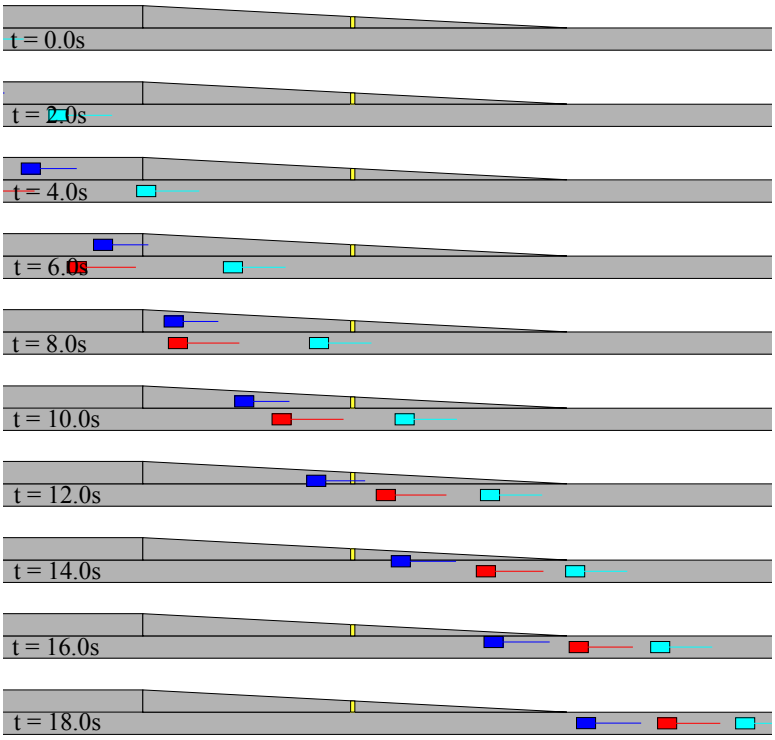


Figure 5.51: iPCB case test in a second scenario in which the merging car wants to yield

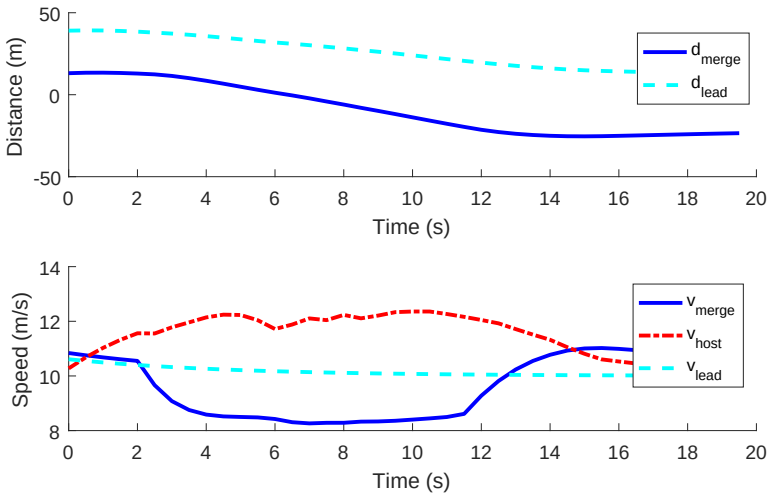


Figure 5.52: Speed profile and distances between cars for a second iPCB case test in another scenario in which the merging car wants to yield

This behavior is different from that in the previous scenario, where the host vehicle was intentionally more conservative due to the uncertainty of the merging vehicle. The main reason is that in the prediction step of the iPCB algorithm in the current scenario, the host vehicle realized that with an aggressive speed-up, the uncertainty about the merging vehicle will be eliminated due

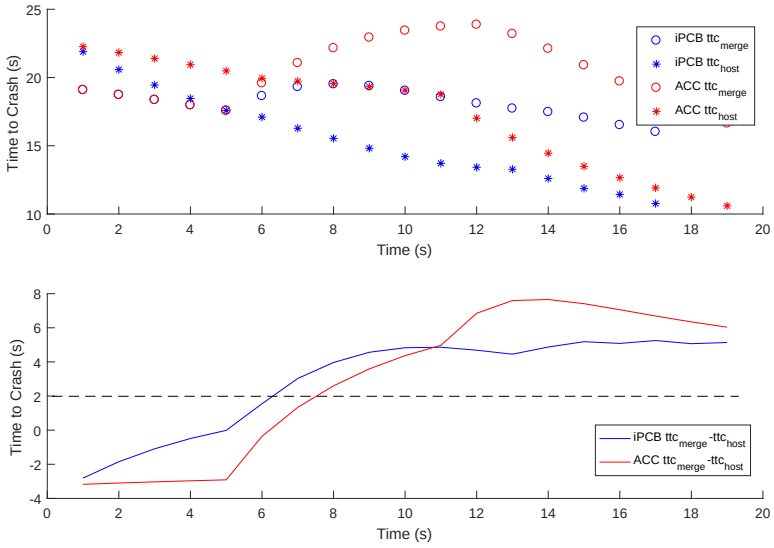


Figure 5.53: Estimated time of arrival for both the merging vehicle and host vehicle in two cases: 1) using default intention-overriding mechanism, 2) disabling the intention-overriding mechanism in iPCB's prediction engine by setting the threshold to 10% of the default value

to the prediction logic described in Algorithm 5.3.1: in cases in which the interaction between the vehicles becomes very clear, the intention will be determined. Figure 5.53 shows the estimated time of arrival for both vehicle at each time step running the algorithm with iPCB-based with and without the intention overriding mechanism. The host vehicle starts to use iPCB from $t = 4s$ when both the host and merging vehicle enter the entrance ramp zone. One can see that within 2 time steps, the estimated time of arrival difference between the merging and host vehicles is larger than $2s$, in which case, we can just assume the merging vehicle will definitely yield. Therefore, the possibility of the merging vehicle not yielding doesn't need to be considered by the iPCB algorithm.

Figure 5.54 shows that if we set the threshold for treating the intention of the merging car as determined to 10% of its current value, the host vehicle will choose a strategy that actually slows down first until it is very confident about the intention of the merging vehicle (which is yield).

Comparing the speed of both the host vehicle and merging vehicle, we can see that with the preference to remove the uncertainty, the host vehicle is able to not slow down as much as always keeping the uncertainty and perform very conservatively. This reduces the cost of the strategy, which means the decision's quality is higher.

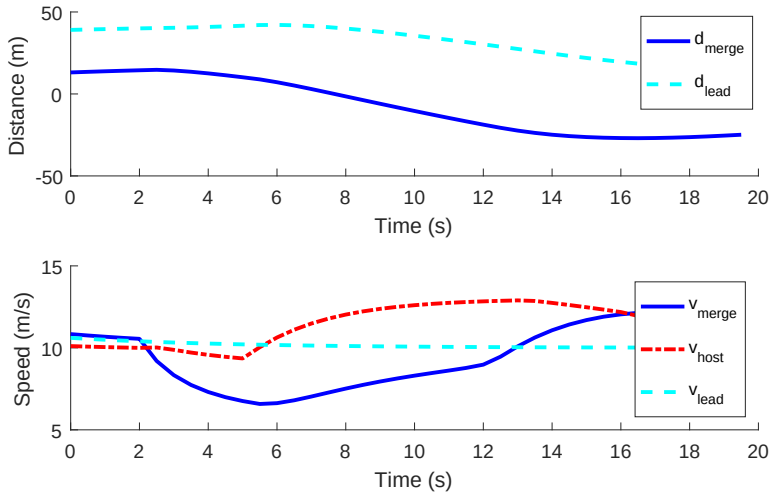


Figure 5.54: Speed profile and distances between cars for iPCB case test in another scenario that the merging car wants to yield. But the iPCB algorithm's elimination of uncertainty of the intention of merging vehicle is set to 10% of the default value

Figure 5.55: Case test scenario of iPCB algorithm for lane change

Similar behavior can be observed in human driving. In similar cases, a human driver could just speed up to show his own intention to the merging vehicle. At the beginning of making the decision to show the intention of the host vehicle, a human driver would also evaluate the safety of that behavior. The iPCB algorithm is able to replicate the same result with a similar reasoning process. It will only command the car to present its intention when even in the worst case, that behavior is safe. In the example, even if merging vehicle has the intention of not yield, it won't be able to speed up fast enough to be a big threat to the host vehicle.

Lane Change The iPCB's performance for lane change is demonstrated in the randomly generated case shown in Figure 5.55. The host vehicle is commanded to perform a lane change at the beginning of the simulation. It is following a leading vehicle running at the desired highway speed. There are multiple cars in the target lane beside the host vehicle.

Similar analysis to that in the entrance ramp scenario is applied here. We take one step of the PCB algorithm as an example. In the iPCB algorithm, there are three cars in the target lane whose intentions the host vehicles need: the cyan car (Vehicle #4), the blue car (Vehicle #5)

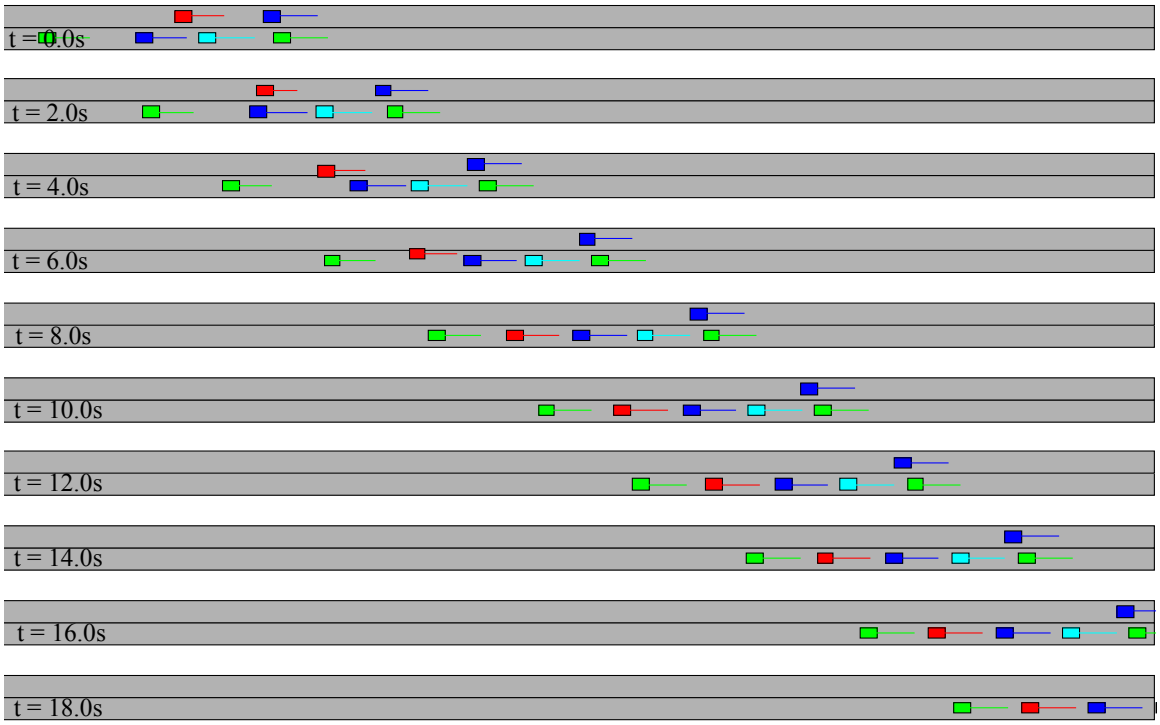


Figure 5.56: Case test result of applying the best strategy generated by iPCB algorithm for lane change

and the green car (Vehicle #6). We set the intention to be $I_{V4} = NotYield$, $I_{V5} = NotYield$, $I_{V6} = Yield$ for the case test simulation.

In the iPCB algorithm, based on vehicle accelerations, the host vehicles identified that $prob_{V4}(I = Y) = 0.0$, $prob_{V5}(I = Y) = 0.0$, $prob_{V6}(I = Y) = 0.71$. Therefore, the best strategy the iPCB algorithm identified is $\{th1 = 5.0, th2 = 1.25 t_{adj} = 10.0, t_{lanechagne} = 3.0\}$. The best strategy using iPCB is shown in Figure 5.56. The speed profiles and distances to vehicles are shown in Figure 5.57, which shows that the iPCB algorithm commands the vehicle to execute the lane change 3 seconds after a big adjustment of its speed by applying the brake.

Intuitively, the host vehicle could wait a little bit longer to do the lane change if we know that the green vehicle will be yielding to us. However, for the green car, there is a 29% possibility it will not yield to the host vehicle. And the initial distance between the green car and the blue car gives the host vehicle enough gap to execute a lane change before the green vehicle catches up with its leading car. If we wait for too long, there is a possibility that the opening will not be available.

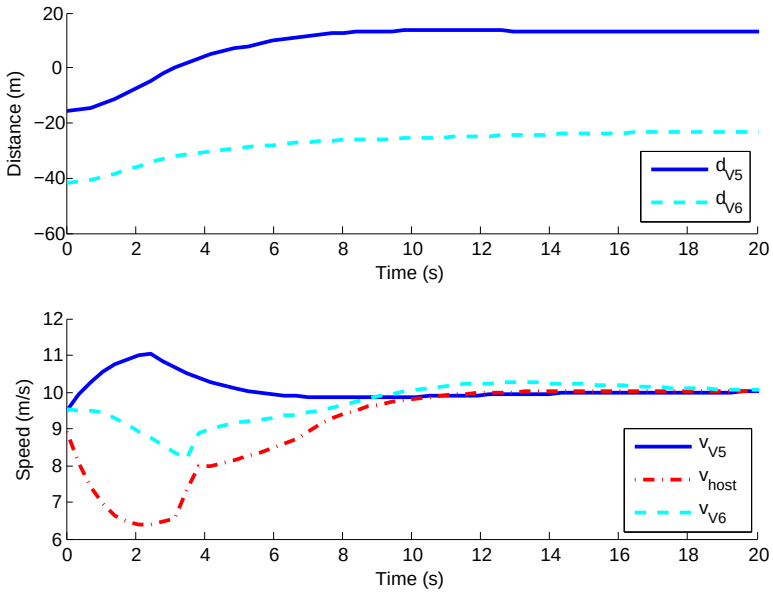


Figure 5.57: Speed profiles and distances between host vehicle and surrounding cars of applying the best strategy generated by iPCB algorithm for lane change

A comparison is done by applying the cPCB algorithm with the correct and wrong intention classification. Figure 5.58 shows that if the prediction is wrong, then the host vehicle will make the wrong decision about the best timing for performing lane change. If it just executes the initial strategy without replanning, it will end up in a precarious situation in which the distance between vehicles is very close driving merging into the target lane.

Table 5.3 shows the best strategy and corresponding cost for each of the three algorithms. It shows that even though the iPCB algorithm cannot achieve the same level of performance as $cPCB_{correct}$ if we know exactly the future behavior of surrounding cars, it will not face the problem of $cPCB_{wrong}$, where we misinterpret the scenario in a way that causes high or infinite scenario cost meaning it is not safe.

5.3.5 Statistical Test

In the case tests, the iPCB algorithm's general ability to perform social behavior is verified. A statistical test is then implemented in simulation to analyze its ability to deal with a wide variety of different entrance ramp and lane change scenarios. The simulation for each algorithm was run 2000 times to get a more accurate statistical result. In these tests, the merging vehicle is

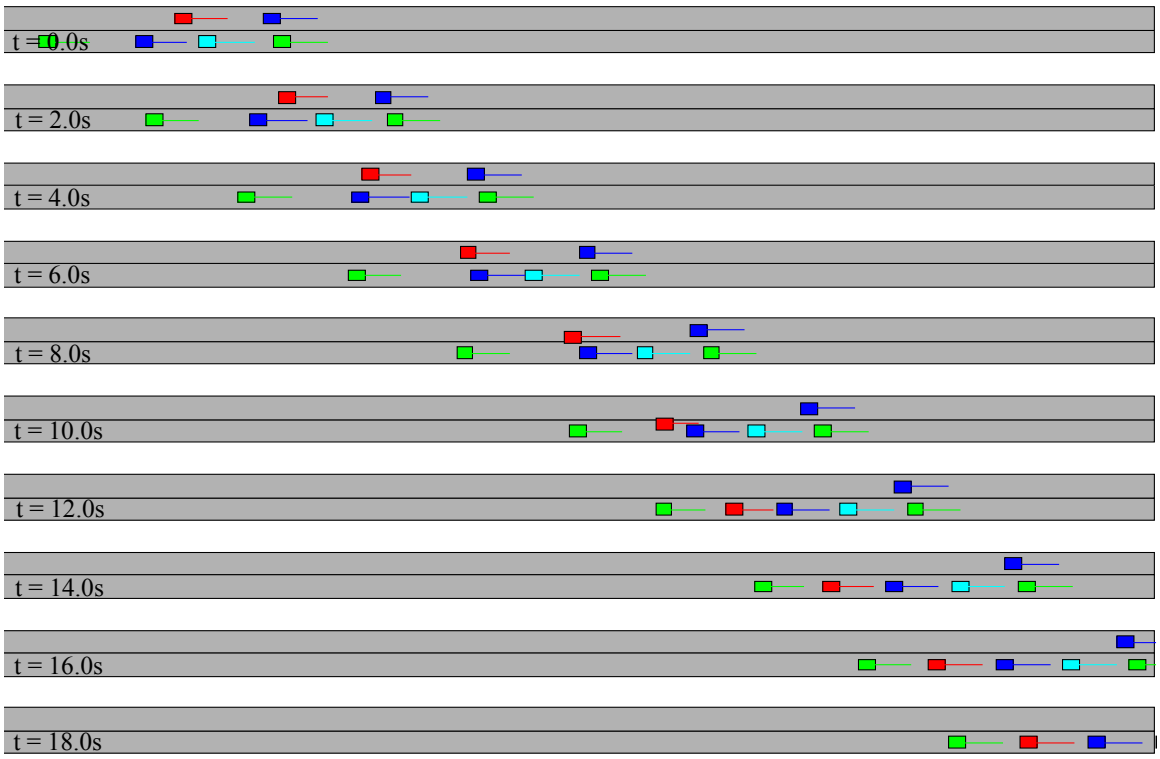


Figure 5.58: Case test result of applying the best strategy generated by the context-based PCB algorithm with the wrong context identification for lane change

Table 5.3: Best strategy and scenario cost for one step using different variants of the PCB algorithm

	$iPCB$	$cPCB_{correct}$	$cPCB_{wrong}$
$th_1(s)$	5.00	3.00	2.25
$th_2(s)$	1.25	4.75	3.00
$t_{adj}(s)$	10.0	10.0	10.0
$t_{lanechange}(s)$	3.0	9.0	9.0
$C_{strategy}$	1243.42	1064.46	inf

simulated using the model described in Section 5.3.1. One example of the testing scenario is shown in Figure 5.59.

As shown in Table 5.4, there are 5 variables with uniform distribution that represent the initial conditions of a given scenario. The road geometry and the host vehicle's initial condition do not change. The leading and merging vehicle's speeds and distances are randomly generated. For the same initial scenario, both yield and not-yield intentions of the merging vehicle are tested.

Table 5.4: Parameter ranges for statistical tests

Parameter	Min	Max
$d_{merge}(m)$	-40.0	-20.0
$v_{merge}(m/s)$	8.0	11.0
i_{merge}	Y or N	
$d_{host}(m)$	-50.0	-50.0
$v_{host}(m/s)$	10.0	10.0
$d_{lead}(m)$	-25.0	-5.0
$v_{lead}(m/s)$	8.0	11.0

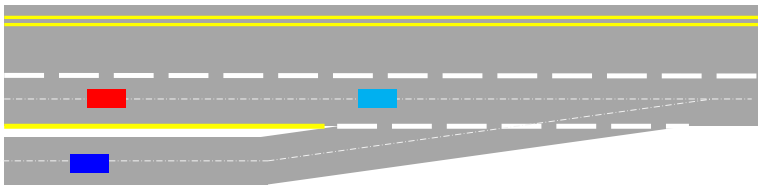


Figure 5.59: Entrance ramp scenario for statistical test

To analyze the iPCB algorithm's performance, we select a few algorithms with which to compare. The first one is the baseline entrance ramp handling algorithm described in Section 4.4.1. Basically, the host vehicle will distance-keep to any vehicles on the entrance ramp but with limited deceleration to give it a smoother adjustment of speed while approaching the ramp. The second uses the standard PCB algorithm, which treats the merging car the same as all other traffic vehicles. The third one is context-based PCB with the correct estimation of the intention of the merging vehicle, so that it is able to predict exactly what will happen in the planning horizon. The fourth one is context-based PCB with the wrong estimation of the intention of the merging vehicle. The last one is the iPCB algorithm introduced in this Section 5.3.

Table 5.5 shows the result of this test, where C_{ave} is the average strategy cost; C_{safety}, C_{acc}, C_{dk} are respectively the safety, comfort and distance keeping progress cost; and $N_{success}$ is the number of cases in which the vehicles perform well and do not need to apply hard braking (with deceleration larger than $3m/s^2$) to avoid potential accidents. Table 5.6 shows the test with the merging vehicle intention as not yield.

The PCB algorithm outperforms the logic-based algorithm in all aspects. It is able to reduce the average total scenario cost, which means the decision made using PCB algorithm is better. Especially for cases in which the merging vehicle wants to yield, the progress cost and comfort

Table 5.5: Statistical test result with $I = Y$

	Logic with ACC	$cPCB_{correct}$	$cPCB_{wrong}$	iPCB
C_{ave}	538.76	455.90	633.12	491.23
C_{dk}	203.28	136.83	232.63	155.66
$C_{progress}$	18.94	18.60	18.76	18.65
$C_{comfort}$	57.15	22.56	126.07	42.17
C_{safety}	259.39	278.91	255.65	274.75
$\%_{success}$	98.2	100.0	66.2	99.8

Table 5.6: Statistical test result with $I = NY$

	Logic with ACC	$cPCB_{correct}$	$cPCB_{wrong}$	iPCB
C_{ave}	506.57	453.05	673.80	458.97
C_{dk}	191.85	130.77	266.14	134.01
$C_{progress}$	19.56	19.22	19.92	19.20
$C_{comfort}$	40.68	25.14	159.25	28.58
C_{safety}	254.47	277.92	228.48	277.18
$\%_{sucess}$	98.5	100.0	91.7	100.0

cost are lower for the PCB algorithm, which indicates that it does not have to slow down a lot for any merging cars. It also achieves a similar success rate to that of the logic-based algorithm. The PCB algorithm has a prediction horizon, which helps it realize potential future dangers and react to them earlier.

The context-based PCB (cPCB) algorithm is also included in the statistical test. It can be seen from the tables that if the context is able to identify the yield or not-yield intention of the merging vehicle correctly, the cPCB algorithm out-performs the other algorithms in the statistical test. However, when cPCB uses the wrong intention to predict merging vehicle's behavior, the performance drops dramatically. For example, when the merging vehicle actually wants to yield but the host vehicle makes a decision based on the assumption that it wants to merge in front of the host vehicle, the success rate is only 66.2%. The average cost for the 66.2% successful scenarios is also much higher than that of other algorithms. Therefore, the cPCB algorithm cannot be used directly for entrance ramp handling.

Compared with the other algorithms, the iPCB algorithm performs much better from the average strategy cost perspective. In both merging vehicle yield and not-yield cases, it is able to reduce the distance keeping cost, progress cost, comfort cost and safety cost. The number

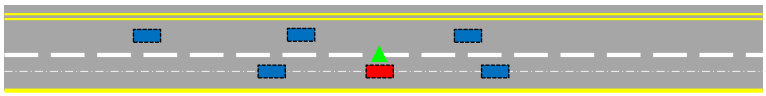


Figure 5.60: Lane change scenario for statistical test

Table 5.7: Statistical test result with random lane change scenarios

	logic with ACC	$cPCB_{correct}$	$cPCB_{wrong}$	iPCB
C_{ave}	1029.85	1005.09	1061.43	1137.36
C_{dk}	170.97	156.80	199.47	180.91
$C_{progress}$	30.57	30.69	30.57	31.36
$C_{comfort}$	45.25	47.91	78.80	81.42
C_{safety}	783.07	769.70	752.59	820.05
$\%_{sucess}$	70.15	100.0	60.30	100.0

of potentially unsafe scenarios is also greatly reduced by using the iPCB algorithm due to its ability to react earlier to merging vehicles based on their intention. This test verifies that the iPCB algorithm framework is beneficial across a wide range of entrance ramp scenarios.

The same test has been implemented for the lane change scenarios. The base scenario is shown in Figure 5.60.

Using the base scenario, the position and speed of all cars are generated with a uniform distribution with $\pm 5m$ distance and $\pm 1.5m/s$ speed variances. The intention of the vehicles is also randomly generated. Each car can either yield to the host vehicle's lane change behavior or not yield to it. Table 5.7 shows the result of the statistical test. Similar to the merging scenario, the average cost of applying iPCB is increased slightly compared to the context-based PCB algorithm with the correct intention, which is not realistic. The iPCB algorithm runs the online intention estimation of all cars in the neighboring lane and computes the best strategy based on the probability of each intention. Therefore, it is able to merge into tighter spots without sacrificing too much safety. From the table, it can be seen that the overall success rate for lane change is much higher than the logic-based lane change or the standard PCB algorithm. The iPCB algorithm may have higher safety cost due to its cut-in manner. Sometimes, it will cutin with smaller distance to the leading vehicle in the target lane even if it identified that there is a possibility that the traffic vehicle may not have decided to yield at the beginning. This allows it to reach much higher success rate but with slightly higher safety cost.

In summary, using the iPCB algorithm, the results indicate that the host vehicle is able to make almost perfect decisions for both entrance ramp and lane change scenarios. Based on the 4000 randomly generated lane change and entrance ramp tests, comparing with logic-based algorithm and the standard PCB algorithm, it reduces unsafe situations by over 99% and provides cost reduction which leads to better smoothness, less unnecessary braking and better tracking of the desired distance to traffic vehicles.

5.3.6 Robustness Analysis

In the previous test, the iPCB algorithm's adaptivity in a wide range of scenarios is tested. However, there is another factor that will affect iPCB's application for road testing. That is the difference between iPCB's driver model and the actual driver model. This is because the iPCB algorithm depends on the accuracy of the driver model for the estimation of the probability of intention. It also uses the driver model to predict future scenarios to make the proper decision.

This section therefore will focuses on analyzing the iPCB algorithm's robustness to model inaccuracy and noise. Simulation tests are deployed for the entrance ramp scenarios to analyze the effect of model mismatching on the algorithm. There are three setups of iPCB being tested here:

- Upper bound performance: In this simulation, it is assumed (unrealistically) that the autonomous vehicle knows all states of the merging vehicle, including its position, speed, acceleration and even the intention without any uncertainty. The costs of the upper bound performance can be used as normalized benchmarks for the other tests.
- Optimistic performance: This is the test implemented in Section 5.3.5. For the optimistic test, in the simulation, only the position and speed of the merging vehicle can be detected. Therefore, a filter for estimating its acceleration needs to be applied. Then the intention estimator is used to get the probability of the merging vehicle's intention. The test is biased to the optimistic side, because in this simulation, all surrounding vehicles will follow the model $B|I$ used by the iPCB algorithm.
- Practical performance: Based on the optimistic test, a practical test is implemented. It is assumed that the merging vehicle follows our prediction model $B|I$ with a reasonable

Table 5.8: Statistical test result

	UpperBound	Optimistic	Practical
C_{ave}	448.48	470.18	563.59
C_{dk}	171.32	141.08	180.96
C_{speed}	19.68	19.19	19.28
$C_{comfort}$	17.01	33.94	96.83
C_{safety}	264.40	275.97	266.52
$N_{success}$	100.0	100.0	99.8

level of deviation of a Gaussian distribution with $\sigma = 1.0m/s^2$. The sensor noise is also simulated for the observed velocity of the merging vehicle with Gaussian distribution with $\sigma = 1.0m/s$. The intention estimation algorithm parameter has to be adjusted accordingly due to larger model standard variation and it is set to $\sigma_{int} = 1.5m/s$. With the higher intention estimation covariance assumption, for the same acceleration reading, it is more likely that the iPCB algorithm will consider both yield and not-yield cases with closer to 50% weight for each. It makes the algorithm more conservative.

Table 5.8 shows the result of the robustness test. It is obvious that the iPCB algorithm cannot achieve the upper bound performance. But even compared with the upper bound performance and optimistic performance, the average cost for the practical situation cost is only a little bit higher. The number of unsafe scenarios and the safety are not degraded dramatically because of sensing and model uncertainties. When the perception system of the car is rather noisy and the merging vehicle model is very inaccurate, even though the cost is higher, the iPCB algorithm is still able to perform safely and smoothly.

5.4 Implementation of PCB Algorithm in Autonomous Vehicle Platform

The PCB algorithm has been implemented in an autonomous vehicle to test its real-time performance. The behavioral planning framework proposed in this thesis is used to make the PCB algorithm command the vehicle through the time headway and relative lateral velocity interfaces. Sensor fusion algorithms described in Section 4.3.4 provides the input needed for the iPCB algorithm. In this section, a few challenges of the real-time implementation of the PCB algorithm are discussed. First, a local vehicle map is generated to limit the number of vehicles the PCB algorithm needs to consider in different domains. Second, replanning needs to be applied at a certain time interval to react to driver model noise or biases and driving behavior changes in surrounding vehicle. Third, the computational complexity and real-time performance of PCB algorithm are analyzed. Finally, test results of iPCB algorithm on public roads are presented and discussed.

5.4.1 Problem Modeling

The first step is to model a more general highway decision making problem into the scope of the iPCB algorithm. In many cases, there could be as many as 10 cars within the host vehicle's perception range. However, the further the vehicle away from us, the noisier the inputs will be. To avoid noisy input of faraway vehicles causing the PCB algorithm to make the wrong decision, and also to reduce the complexity of the algorithm, a selection algorithm is implemented to only pass information about the most related cars to the iPCB algorithm.

Given that the vehicle is driving autonomously, it has a high-accuracy digital map and knows the upcoming context of either lane change or merge. Based on the context, for the lane change scenario, only the closest vehicle to the host vehicle and the closest 4 vehicles in the target lanes are considered. For the entrance ramp scenario, if there are multiple vehicles entering from the entrance ramp, the host vehicle will only use the information of the merging vehicle with the closest longitudinal distance to it and the leader of the host vehicle.

The projection of all the vehicles onto the centerline or center of the entrance ramp is also applied in this module, as described in Section 5.1.2 . After this preprocessing, the PCB algo-

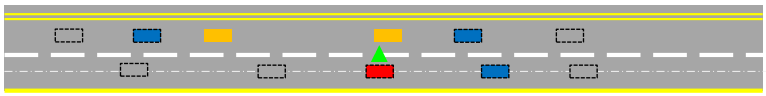


Figure 5.61: Vehicle selection module in the PCB algorithm for lane change scenario

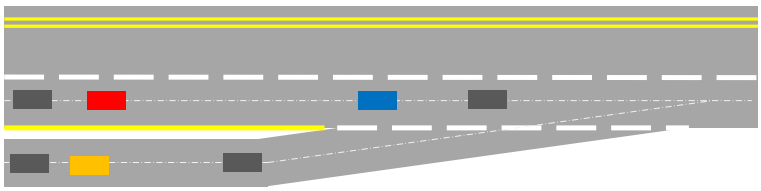


Figure 5.62: Vehicle selection module in the PCB algorithm for entrance ramp scenario

rithm will receive inputs of at most 5 cars for lane change scenarios and 3 cars for entrance ramp scenarios.

As shown in Figure 5.61 and 5.62, for the lane change scenario, we assume that only the vehicle closest to the host vehicle in the target lane $V_{closestInd}$ and the vehicle behind $V_{closestInd}$ will have intentions that affect their behavior of interacting with the host vehicle's desired lane change maneuver. In the figures, the red vehicle is the host vehicle, the blue and orange vehicles are the vehicles the PCB algorithm needs to consider, and the gray vehicles are the ones trimmed down by the vehicle selection modules. For the entrance ramp scenario, we assume that only the one merging vehicle (the closest one to the host vehicle) will interact with the host vehicle. The traffic vehicles whose intentions are considered by the host vehicle's PCB algorithm are marked in orange in Figure 5.61 and 5.62.

5.4.2 Replanning

Another important part for the real implementation of the algorithm is replanning. Even though with iPCB, we could achieve a higher level of performance and safety without as frequent replanning due to our better understanding and prediction of surrounding vehicles' movements, it is still necessary to plan for the changing traffic conditions.

Noise is added to the simulation model so that the surrounding cars will drive with slightly differernt speed or acceleration than the prediction model. The results of replanning for a merging scenario applying the intention-integrated PCB algorithm with and without the noise are

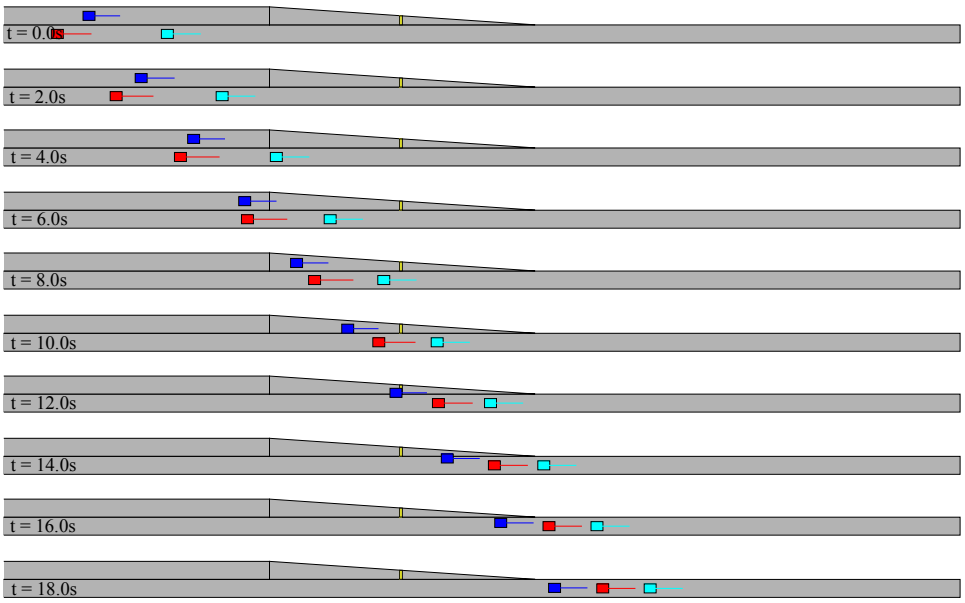


Figure 5.63: Simulation result when the host vehicle is running iPCB with a driver model but the actual merging vehicle's behavior is consistent with the iPCB driver model

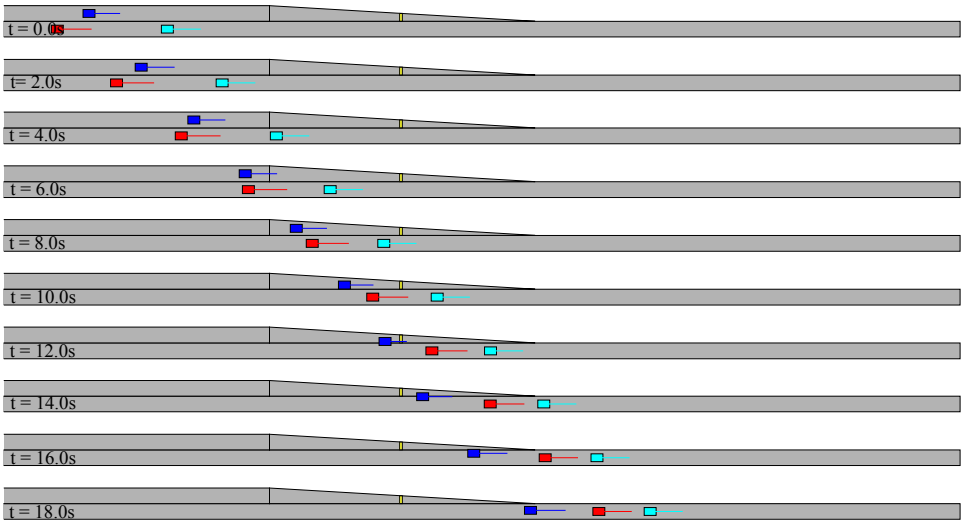


Figure 5.64: Simulation result when the host vehicle is running iPCB with a driver model but the actual merging vehicle's behavior diverges from the iPCB driver model

shown in Figure 5.63 and Figure 5.64. In both cases, the host vehicle is still able to handle the merging vehicle that wants to yield with the proper speed-up behavior. However, when the prediction model and the actual merging vehicle driver's behavior do not match, more oscillation in the host vehicle's acceleration command can be observed as shown in Figure 5.65. This is because the best strategy for the host vehicle changes more often with bigger amplitude due to

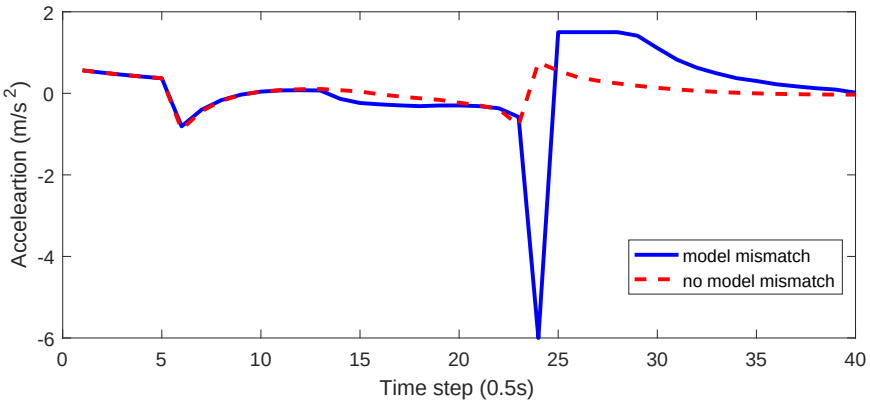


Figure 5.65: The comparison of accelerations of the host vehicle in cases that the surrounding vehicle behave like and unlike the iPCB driver model

the model mismatch. We decide to use a $5Hz$ update rate because that is able to handle most of the behavioral interaction cases.

Furthermore, a hysteresis cost has been added in the iPCB implementation based on the difference between the candidate strategy and the best strategy selected from the last cycle as shown in Equation 5.18.

$$C_{hyster} = \sum_{t=0}^{t_{predict}-\delta t} |th_{curr}(t) - th_{last}(t+\delta t)| \tag{5.18}$$

Figure 5.66 shows the comparison of the vehicle's selected best strategy and commanded acceleration with or without this hysteresis cost. It shows that with the hysteresis cost, the vehicle avoided some large jumps in commanded acceleration. It is important to have this mechanism in the implementation on a real autonomous driving platform. This is because in an autonomous vehicle, the input signals, especially the acceleration could be rather noisy or the estimation may not be available for a short period of time due to occlusion or other types of sensor noise. An example is provide in Figure 5.66. At time step 5.0, the vehicle lost track of the merging car and therefore needed to reset the acceleration to $0m/s^2$. At time 5.5, it sees the target again and starts to run the Kalman filter to estimate acceleration with $0m/s^2$ as the initial value, which may take some time for it to actually be able to converge to the real acceleration. This will cause undesired jumps of the estimated probability of intentions of the traffic vehicles and therefore affects the final decision. This hysteresis mechanism makes sure that if the host vehicle picks one decision,

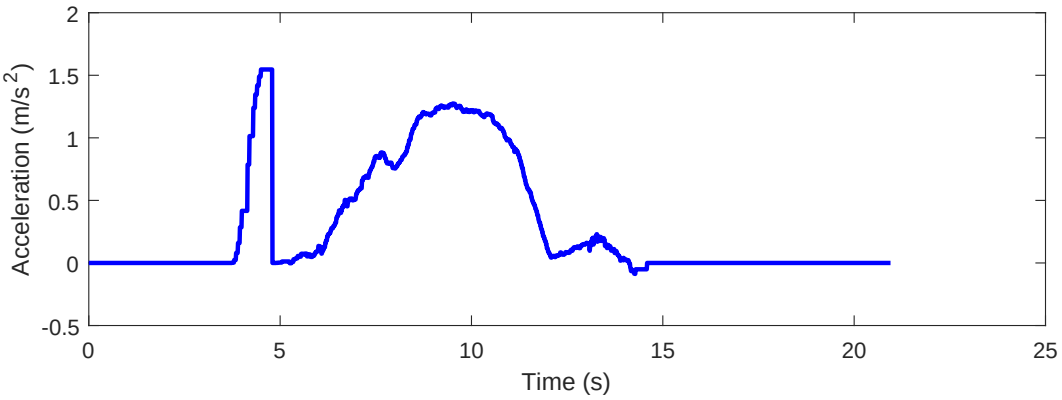


Figure 5.66: An example of the noisy acceleration estimated by the perception system of an autonomous vehicle for on-road testing

it is willing to stick with it unless the environment changes make the previous decisions much worse. A consistent decision for the iPCB algorithm also makes surrounding traffic react to the host vehicle's showing of its intention.

5.4.3 Computation Expense

The computation expense is also evaluated for iPCB algorithm. The computational complexity is shown in Equation 5.19, in which $OiPCB$ is the computational complexity of the iPCB algorithm, N_{veh} is the number of vehicles the iPCB algorithm need to predict, N_{intVeh} is the number of vehicles whose intentions need to be estimated, N_{th1} and N_{th2} are the two time headways used for speed adjustments, $t_{predict}$ is the prediction horizon of the iPCB algorithm and Δt is the time interval for the prediction.

$$OiPCB = ON_{veh} \cdot 2^{N_{intVeh}} \cdot N_{th1} \cdot N_{th2} \cdot N_{tadj} \cdot t_{predict} \cdot \frac{1}{\Delta t} \tag{5.19}$$

Table 5.9 shows the parameters used for the implementation of the iPCB algorithm for both lane changing the entrance ramp scenarios and the time-consumption running it in the autonomous driving vehicle's on-board embedded computers with an Intel i7 $2.4GHz$ processor. It is shown that the iPCB algorithm for entrance ramps is able to run in real-time with $5Hz$. The iPCB lane change algorithm takes around 36 times longer than the iPCB entrance ramp algorithm. For the lane change scenario, because the number of cars and number of cars requiring

Table 5.9: iPCB implementation parameter selection and computing time

	PCB_{LC}	$iPCB_{LC}$	PCB_{ER}	$iPCB_{ER}$
N_{veh}	6	6	3	3
N_{intVeh}	0	2	0	1
N_{th1}	21	21	21	21
N_{th2}	21	21	21	21
N_{tadj}	2	2	2	2
$t_{predict}$ (s)	15.0	15.0	15.0	15.0
Δt (s)	0.5	0.5	0.5	0.5
t_{ave} (ms)	189.0	724.1	8.6	21.7

intention estimation increases, the PCB algorithm is able to achieve a real-time update rate of $5Hz$ while iPCB algorithm can only be executed at around $1.25Hz$.

To improve the real-time performance of iPCB algorithm for it to run in real-time or even with lower-end embedded controllers, a multi-step search process is proposed. For the first step, a coarse search is applied with fewer number of $th1$, $th2$ selections to minimize the time consumption. Then another search step is applied with finer granularity to make sure the best strategy is found. Table 5.10 shows the performance comparison with a statistical test of 1000 randomly generated scenarios between using this multi-step approach and the original approach. After step1, which is a very coarse search through the candidate strategy space, the decision quality is not as good. But after step2 of the multi-step iPCB implementation, it achieves similar performance to the iPCB algorithm with only 22.41% of the time taken. This makes it feasible to run in real-time at $5Hz$. For the few cases in which the multi-step approach get a worse result than the original iPCB algorithm, it is because of the lack of granularity causing the first step to find a locally optimal solution. The second step cannot recover from the wrong decision made by the first step. But this happens only for fewer than 1% of all the testing cases and even in those cases, the decision cost is still acceptable based on the cost shown in Table 5.10.

5.4.4 Testing Result

Result of running the iPCB algorithm on public roads have been collected on four entrance ramps with different geometries in Pittsburgh on Route 28. The max allowed speed for the social interaction is set to be $68mph$ with the speed limit set to be $65mph$. Figure 5.67 shows the

Table 5.10: Comparison of decision quality and computing time of the original iPCB algorithm and multi-step iPCB algorithm

	$iPCB_{LC}$	$Step1\ iPCB_{LC}$	$Step2\ iPCB_{LC}$
N_{veh}	6	6	6
N_{intVeh}	2	2	2
N_{th1}	21	5	5
N_{th2}	21	5	5
N_{tadj}	2	2	1
$t_{predict}$ (s)	15.0	15.0	15.0
Δt (s)	0.5	1.0	0.5
t_{ave} (ms)	724.1	73.2	89.1
C_{ave}	1243.4	1355.2	1250.9

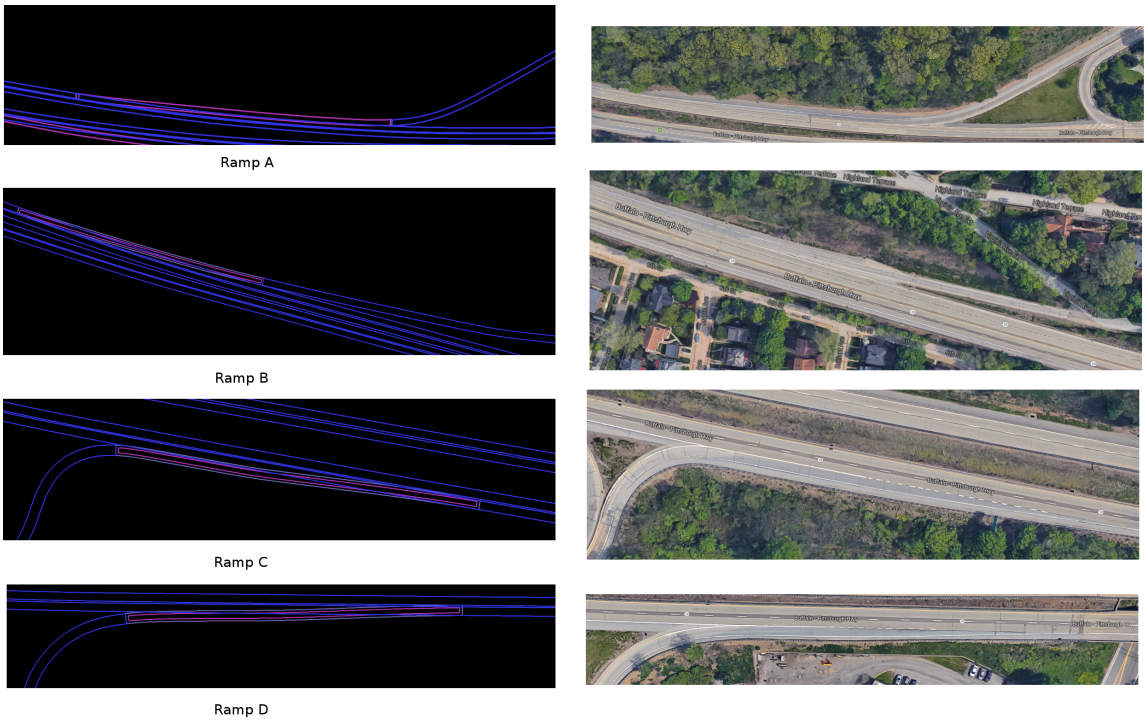


Figure 5.67: The map and corresponding aerial imagery of the four entrance ramps are used for the public road test of the iPCB algorithm

map used by the autonomous vehicle and the corresponding aerial imagery for the four entrance ramps.

For entrance ramp A and B, the length of the merging ramp is relatively long. There is a yield sign for the ramp, and people entering from the ramp need to speed up enough before entering into main lanes of the highway. For ramps C and D, the ramp is very short, causing

Table 5.11: Parameters for four entrance ramps cases in on-road testing

	Ramp A	Ramp B	Ramp C	Ramp D
l_{ramp} (m)	230.73	154.64	84.93	140.22
l_{pcb} (m)	97.85	201.74	73.14	28.50
w_{lane} (m)	4.33	4.18	4.45	4.24

most merging vehicles to have to stop and wait unless there are no vehicles driving in the lane they want to merge into.

Table 5.11 shows the geometry information used by the iPCB algorithm for these entrance ramps, in which l_{ramp} is the length of the entrance ramp, l_{pcb} is the distance to the start of the ramp where the PCB algorithm will start running and w_{lane} is the average width of the merging lane and the main road lanes. The less occlusion, the longer l_{pcb} that can be used to start the social interaction earlier to achieve better performance.

The result of one of the on road testing cases is shown in Figure 5.68. It happens for the entrance ramp A. The distances between vehicles and the speed profile for all cars are shown in Figure 5.69. The probabilistic intention estimation result is shown in Figure 5.70, in which $prob(I = NY)$ is the probability of the intention of the merging vehicle being not yield, when $prob(I = NY) = 0.5$, it means that the intention estimation is unavailable. It shows that the estimated merging vehicle intention keeps in between 0.6 to 0.9 for around 5 seconds. The uncertainty of the actual intention of the merging vehicle lead to a more conservative strategy for the host vehicle.

As shown in Figure 5.68, the host vehicle is following its leading vehicle before entering the entrance ramp scenario area. When the iPCB algorithm is executed, it realizes that the uncertainty for the merging vehicle's intention of yield or not-yield is high. Therefore, as discussed in Section 5.3.4, the iPCB algorithm selects a more conservative strategy to allow it to merge in first.

Another example of the case test is shown in Figure 5.71. This happens for entrance ramp B. The distances between vehicles and the speed profile for all cars are shown in Figure 5.72. The probabilistic intention estimation result is shown in Figure 5.73. In this case, the initial estimated intention of the merging car is that it is more likely to not yield to the host vehicle. But the host

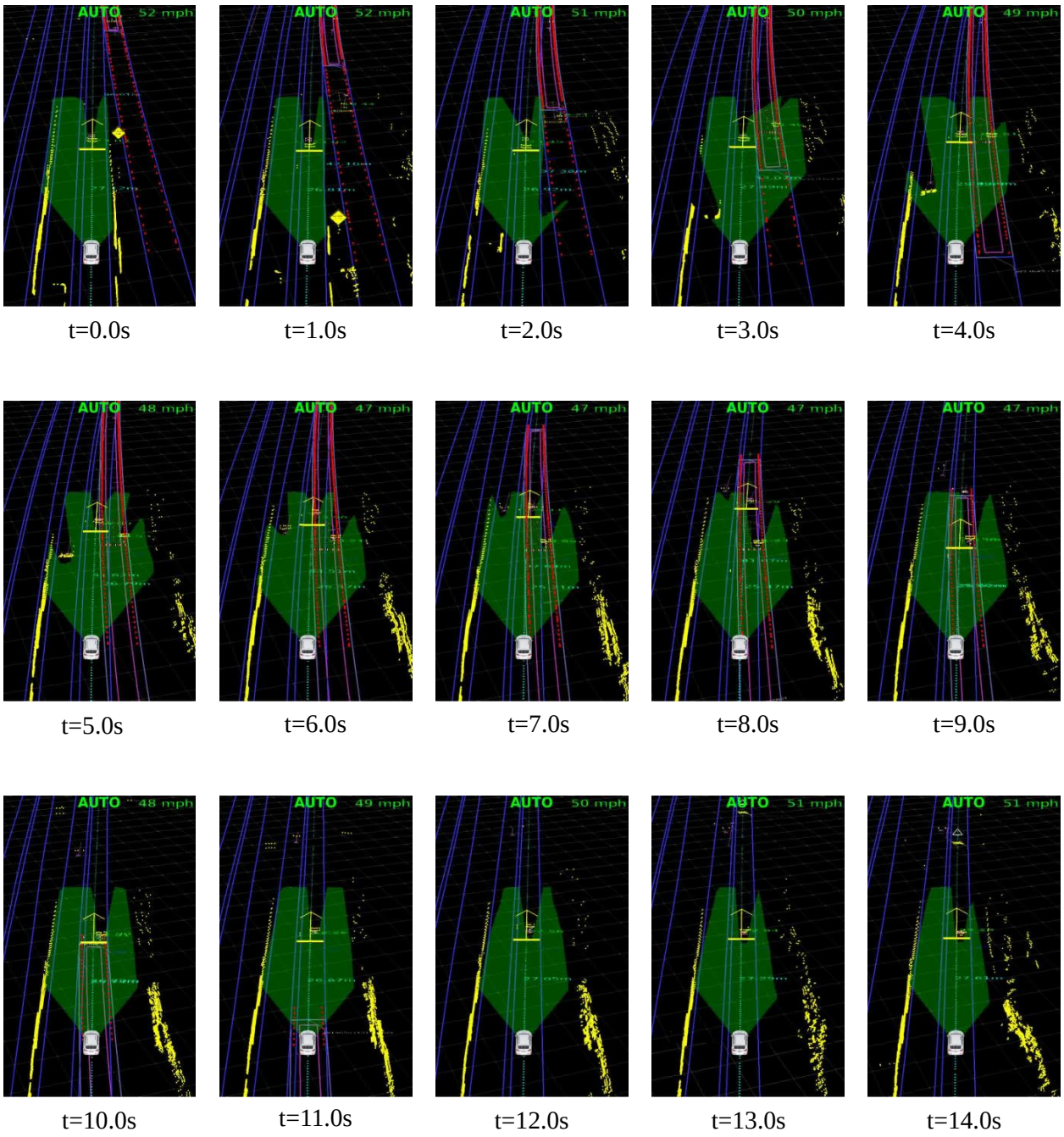


Figure 5.68: One case of the road testing result of the iPCB algorithm

vehicle decides to speed up based on evaluation different strategies. After less than 1 second, due to the speed change of the host vehicle, the probability of the merging car being yield is more dominant. At $t = 5s$, it is clear that the merging vehicle will yield to the host vehicle, which means the action chosen by the host vehicle successfully changed the behavior for the merging

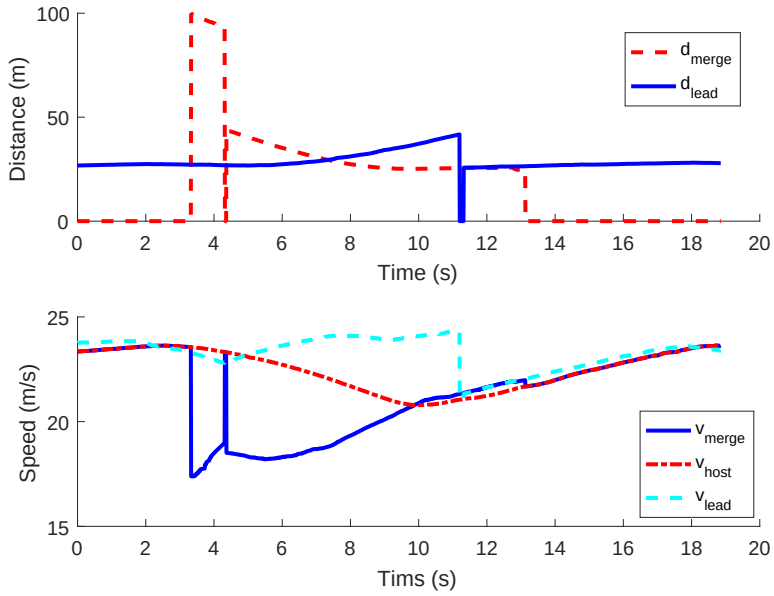


Figure 5.69: Distances between vehicles and speed profiles of all cars in the entrance ramp domain in on road testing using the iPCB algorithm

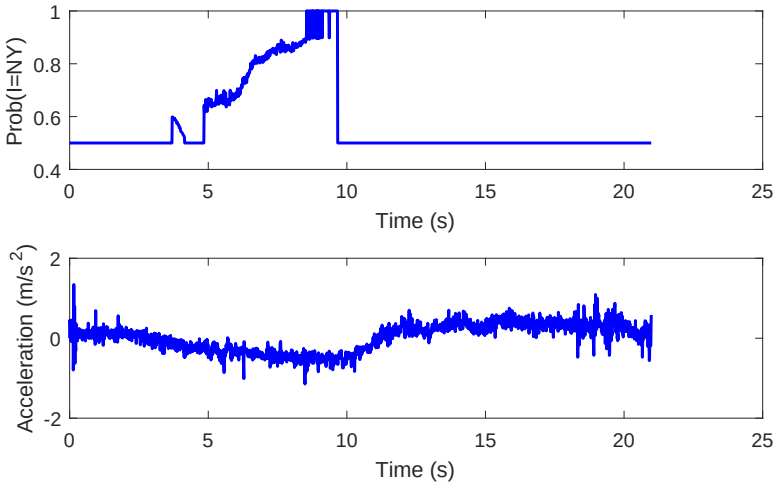


Figure 5.70: Intention-integrated PCB algorithm's intention estimation result in the first entrance ramp case

vehicle.

As shown in Figure 5.71, the host vehicle enters the entrance ramp area without a leading vehicle. When the iPCB algorithm is executed, it detects a vehicle next to it with similar speed. There is some uncertainty about the intention of the merging vehicle. The iPCB algorithm's decision is that given there are no leading vehicles to the host vehicle, the host vehicle can speed up to arrive at the merging point earlier than the merging vehicle

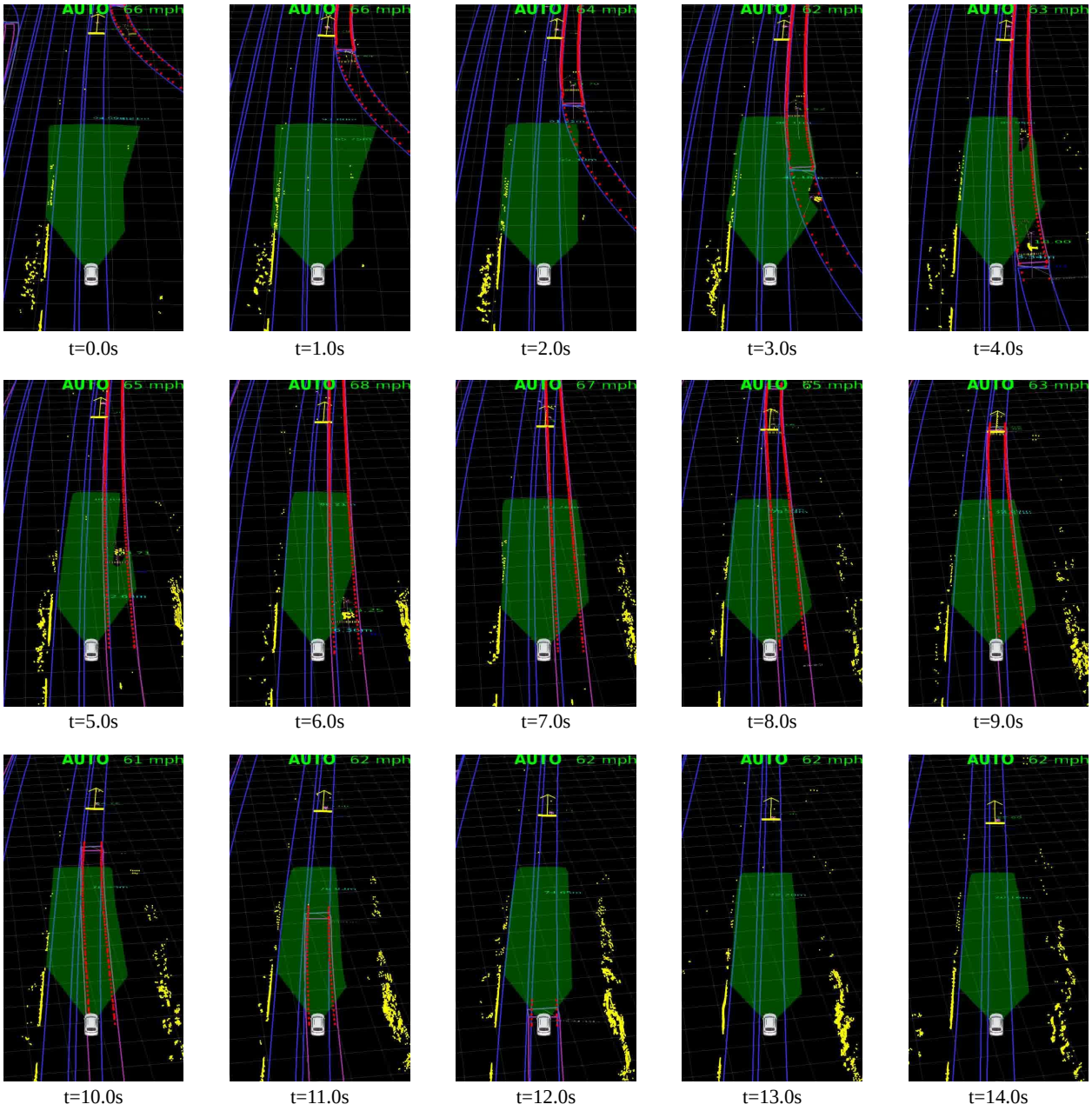


Figure 5.71: One case of the road testing result of the iPCB algorithm

Based on 35 case tests on public roads, the full implementation of the iPCB algorithm, the behavioral planning framework and the functionality of the autonomous vehicle with socially cooperative behavior capability is validated. It is able to make either an aggressive or conservative decision based on the uncertainty of the intention of the surrounding vehicles. It also out-performs the logic-based algorithm by allowing the vehicle to show its intention to better

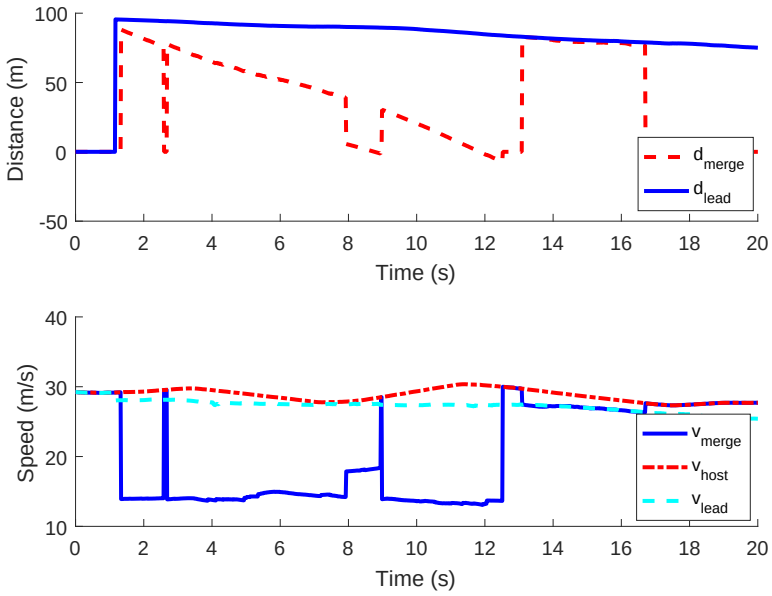


Figure 5.72: Distances between vehicles and speed profiles of all cars on the entrance ramp domain in on-road testing using the iPCB algorithm

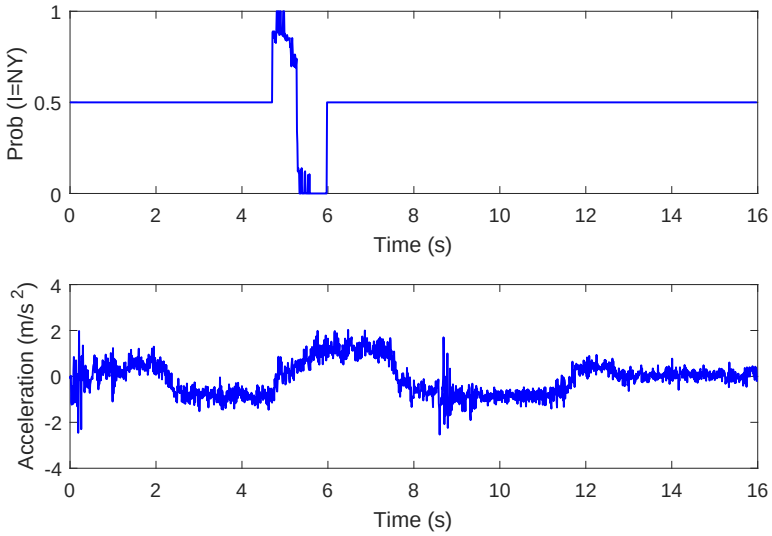


Figure 5.73: Intention-integrated PCB algorithm's intention estimation result in the second entrance ramp case

interact with surrounding cars. On-road testing shows similar results for iPCB algorithm to the simulation case tests.

Chapter 6

Conclusion

Autonomous vehicle has great potential to dramatically improve the safety and efficiency of the cargo and passenger logistics system. With the latest developments in sensor, computer and AI technologies, their commercial deployment on public roads can be expected in the very near future. However, based on previous implementation and testing of autonomous vehicles, we submit that making autonomous vehicles socially cooperative is an important topic. Autonomous vehicles have to share the road with human-driven vehicles. Among human-driven cars, people can often interpret each other's intentions well and also show their own intentions to surrounding cars. This capability is one of the major differentiators between an experienced human driver and a new driver. An experienced human driver is able to use this skill to perform smoother driving to achieve his goal without interrupting other people's driving. Therefore, it is important for autonomous vehicles to be able to perform socially cooperative behaviors with surrounding human-driven vehicles.

We introduced two example domains of autonomous vehicle social behavior: lane change and entrance ramp merging. For lane changes, the host vehicle needs to tell if the vehicles in the target lane are willing to give way to our indicated lane change. For entrance ramps, the host vehicle needs to figure out how to adjust its speed if it sees a merging car coming from an entrance ramp. The merging vehicle could either slow down to yield or speed up to not yield.

There are two main technical contributions in this thesis to enable autonomous vehicles able to handle socially cooperative behavior scenarios:

1. A novel behavior-planner integration framework that enables robust freeway autonomous driving and social cooperation with surrounding vehicles.
2. A Prediction-and Cost function-Based (PCB) algorithm for autonomous vehicles to perform social behaviors in a wide variety of scenarios. Quantitative analysis presented the importance of social behavior for autonomous vehicles based on experiments in simulation and on public roads.

6.1 Behavioral Planning Framework

Different from the state-of-the-art robot motion planning that focuses on obstacle avoidance or generating a trajectory from point A to point B, the social behavior planner needs to consider the interaction between vehicles on the road. To make the autonomous vehicle with limited computational power able to handle both behavioral-level and motion-level planning, we proposed a hybrid-hierarchical architecture for the planning modules in an autonomous vehicle's software system.

The behavioral planner utilizes the vehicle's production-ready and validated Adaptive Cruise Controller, lane centering controller and lane change controller to perform the desired social cooperation. The benefits of applying this framework include the reduction of search space dimensions and dependency on high-frequency replanning. By default, the vehicle is able to perform the basic autonomous driving behaviors of distance keeping and keeping in the center of the lane. The behavioral planner will command different time headways and relative lateral speeds for the vehicle to handle entrance ramps, lane changes, and other social behavior scenarios.

We proposed a reference design of sensor, computer, power and Drive-By-Wire hardware and software modifications to build an autonomous vehicle with ACC and lane centering algorithms. The behavioral planning framework and the iPCB algorithm we proposed in this thesis have been successfully implemented multiple autonomous vehicles built based on the proposed reference design. The effects of time headway and relative lateral speed to the centerline on the autonomous vehicle's trajectory and behavior are analyzed. Road testing shows that with this capability, a wide range of possible speed and lateral adjustments can be commanded by the social

behavioral planner. As a result, social behavior could be implemented as an addition module to state-of-the-art semi-autonomous vehicles on the highway.

Before introducing the PCB algorithm, we implemented a logic-based lane change and entrance ramp handling module using this framework. The host vehicle is able to perform a certain level of cooperation with merging vehicles on entrance ramps and vehicles in the target lane for lane changes. We have tested the logic-based algorithm for a month with over 450 lane changes and 450 entrance ramp occurrences with 97.4% success rate without human intervention. Even with this simple logic, the autonomous vehicle performs much better than state-of-the-art production systems such as the Tesla Auto Pilot system. By doing this test, we also demonstrated that more advanced decision making or planning algorithms are needed to advance the safety and smoothness of autonomous vehicle's driving on public roads in scenarios where social cooperative behaviors are needed.

6.2 PCB Algorithm

The PCB algorithm is what we proposed for the behavioral planner to find the best time headway and relative lateral velocity command for the host vehicle to perform social behavior. It has three steps: i) candidate strategy generating discretizes the potential combination of behavioral planner outputs for the length of the PCB algorithm's prediction horizon, which is usually 10 to 20 seconds, ii) the prediction engine generating the future scenarios based on the assumption that the host vehicle will execute a candidate strategy and surrounding vehicles will react to the host vehicle's behavior by slowing down if they are approaching the host vehicle in front of them or apply emergency braking when the host vehicle is within the critical distance, iii) cost function-based evaluation uses distance between the host vehicle and its leader, safety gaps around the host vehicle, speed, acceleration and fuel consumption of the host vehicle as the performance evaluation matrix of the candidate strategy. A best strategy will be picked based on the lowest cost computed for all candidate strategies. It is demonstrated that the PCB algorithm is able to handle some scenarios which the state-of-the-art planners cannot handle due to their assumption that all traffic vehicles will follow constant speed.

It is shown that an accurate prediction of the traffic conditions in the future is the key for a high-quality decision in the social behavior domain, such as for lane changes and entrance ramps. The basic prediction engine based on the assumption that traffic vehicles will slow down when they are approaching cars in front of them does not always apply, especially for entrance ramp cases in which the merging vehicle could slow down for the main lane traffic because it needs to yield to traffic on the main road. It would usually yield if the cars on the main road are behind it but approaching with faster speed. Therefore, a more accurate prediction model based on context is built. The host vehicle first determines whether the merging vehicle should be yielding or not and then applies the corresponding logic in prediction. The context-based PCB (cPCB) algorithm using this logic improves the performance of standard PCB algorithm by 29.85%.

The cPCB algorithm is able to classify obvious intentions of merging vehicles and apply more accurate predictions of their future behavior. However, the classification can be wrong due to the fact that different drivers have different driving styles, which could be more conservative or aggressive. By picking one determined context-classification and a corresponding prediction result, the cPCB algorithm performs poorly and can cause unsafe situations when it picks the wrong prediction model.

Therefore, the iPCB algorithm is introduced to capture the uncertainty of the potential behaviors caused by different intentions of the traffic vehicles. An intention probability estimation module is built to compute the probability of traffic vehicles' intentions to yield or not yield to a host vehicle in entrance ramp or lane change scenarios. Then, intention-based prediction is applied for each possible intention combination of all surrounding vehicles. The total strategy cost is the sum of the cost for the intention-based prediction result with the probabilities of the intentions as weights.

The iPCB algorithm has two advantages over the standard and context-based PCB algorithm. First, it is able to consider the different potential behaviors of traffic vehicles. When the uncertainty of the surrounding vehicles' behaviors are high, we demonstrate that the iPCB algorithm will generate more conservative strategies. This shows that using the iPCB algorithm, the host vehicle is making decisions based on its understanding of surrounding vehicles' intentions. Sec-

ond, the iPCB algorithm has a preference to select a strategy that would reduce the uncertainty of surrounding vehicles' behaviors in the future. This shows that the iPCB algorithm is able to present the host vehicle's preference through speed-up, slow-down or lateral movements with respect to surrounding vehicles. With these two capabilities, we conclude that the iPCB algorithm is able to make autonomous vehicles perform social behaviors.

4000 different entrance ramp and 2000 lane changing scenarios were simulated to get statistical testing results for different PCB algorithms. The context-based PCB (cPCB) algorithm with the correct classification of the context for traffic vehicle behavior prediction improved the performance by 13.04% and 2.4% for the entrance and lane change scenarios respectively. It also reduces the number of precarious cases by 100.0%. However, when the context classification result is wrong, the number of unsafe situations in the 6000 tests increased to 24.24%, which is even worse than the 11.05% failure rate of the logic-based algorithm. In contrast, the intention-integrated PCB (iPCB) algorithm shows a good balance between utilizing the intention information for better prediction and considering the uncertainty in the decision making. Compared with the logic-based algorithm, the number of unsafe situations is reduced by 99.4% in average, the performance represented by cost function is improved by 3.42%.

To evaluate the iPCB algorithm's robustness to deviation between the traffic vehicles' behaviors and the human driver model that the iPCB algorithm depends on, another statistical test in simulation was implemented. The performance was compared for: 1) the cPCB algorithm with known intention information of the traffic vehicles as upper bound performance; 2) the iPCB algorithm that has to estimate the intention of cars using the driver model while the same driver model is also used for simulation, which is an optimistic assumption; 3) the iPCB algorithm using a driver model that deviates from the driver model used for simulation. The result is that even with a large deviation between the iPCB's driver model and the actual model used by the simulation, as long as we increase the standard deviation assumption in the intention estimation module in the iPCB algorithm, it still achieves reasonable performance. Compared with the logic-based algorithm, it reduces 99.4% of the potential unsafe scenarios.

6.3 Road testing

The logic-based entrance ramp and lane change handling algorithm as well as the behavioral planning framework are implemented in an autonomous vehicle with the reference design proposed in this thesis as a baseline performance reference. The logic-based algorithm was tested on over 450 entrance ramp cases and 450 lane changes on public roads, with a success rate 97.4%. The success is defined as that both safety and comfort are satisfactory based on the testing reports from test engineers sitting in the automated car. It is demonstrated that even with the simple logic, though the host vehicle will perform very conservatively and without social interaction with surrounding cars, it is able to handle the majority of the use cases.

To handle more complicated scenarios in which social conflict occurs, and to achieve better performance for the entrance ramp and lane change scenarios, the iPCB algorithm was implemented in the same behavioral planning framework on the same vehicle. The computational complexity of the iPCB algorithm was analyzed. An optimization of the exhaustive search algorithm was proposed for better real-time performance. Challenges in the real-time implementation such as replanning and perception noise handling were addressed to achieve the best performance of iPCB. Public road case tests of 40 entrance ramp handlings using iPCB algorithm show that it is able to work as demonstrated in simulation to make the autonomous vehicle react actively based on its estimation of a merging vehicle's intention and its ability to show its own intention.

6.4 Future Work

Even though the iPCB algorithm with the behavioral planning framework shows great potential to improve the quality of interaction between autonomous and human-driven vehicles, there are still areas that can be further explored.

First, the iPCB algorithm has considerable dependency on the accuracy of prediction of surrounding vehicles' behaviors. Therefore, it is important to validate the proposed prediction model using real data. There are multiple parameters in the proposed human driver model that could be learned from a large amount of data from human-driven vehicles.

Second, the intention estimation algorithm also affects the iPCB algorithm's decision making.

It directly decides how conservative the host vehicle needs to be. By collecting more data, similar to the human driver model, we can achieve a higher-performance intention estimator, which will greatly boost the performance of the iPCB algorithm.

For better prediction and intention estimation, more inputs or hints from human-driven vehicles could be used, such as the lateral position and velocity of the vehicles, turn signals, horn and high-beam lights. In addition to that, the host vehicle could use these actions or interfaces to indicate its own behavior to surrounding human-driven cars to affect their behaviors or intentions.

In this thesis, we have only explored the entrance ramp and lane change social cooperative behavior domain. For autonomous vehicles to drive better with surrounding cars, the iPCB algorithm should be tested in handling cut-in vehicles, host vehicle merging onto the highway and other scenarios requiring the host vehicle to socially interact with surrounding cars. Furthermore, in the entrance ramp and lane change domain, we simplify the model by assuming the vehicles will drive in the center of the lane or entrance ramp. But in the real world, the lateral position and speed of the traffic vehicle and host vehicle could also be a strong hint of their intentions. Therefore a more complete intention-based driver model could be built to fully explore the lateral movements' effect on the social behavior.

In addition to the statistical and robustness testing about human driver model accuracy, further analysis needs to be done on the algorithms sensitivity to the geometry of the road including entrance ramp length, width and road curvature and environmental occlusions. Different cost function tuning will lead to different driving style and aggressiveness, which is another topic worth exploring. The algorithm's robustness to perception and execution system's latency still need to be explored.

The road geometry, environmental occlusion and traffic signal parameters also affect how a human driver handles the interactions. For example, for a very short entrance ramp with a yield sign, a human driver entering from the ramp will very likely stop before merging onto the highway if there is any traffic. Being able to understand how human drivers utilize the context information will improve the autonomous vehicles prediction capability and therefore generate higher quality decisions.

The iPCB algorithm has only been tested on public roads for limited entrance ramp scenarios.

To demonstrate its capability, more testing needs to be done on a wider variety of entrance ramp geometries and locations as well as on lane changes. We have also identified that the perception system of the autonomous vehicle needs to be improved for better estimating of the acceleration of the surrounding vehicles. The occlusion of entrance ramps and other practical issues will also affect the performance of the social behavior of autonomous vehicles, which could be added as part of the iPCB algorithm.

Even though the focus of this thesis is mainly on the interaction between automated vehicles and human-driven cars, it is beneficial to analyze the performance of this algorithm when multiple automated vehicles interact with each other. Scenario tests could be done in applying the algorithm with a combination of semi-automated vehicles, and fully automated cars with or without intention estimation capability. It is necessary to make sure the interaction is still smooth even under conditions in which the automated vehicles involved in the social interaction are running the same intention estimation and decision making algorithm.

6.5 Conclusion

In this thesis, we proposed that autonomous vehicles need to perform socially cooperative behavior on public roads for better interaction with human-driven vehicles. We implemented a behavioral planning framework to enable socially cooperative behavior without adding exponentially high computing complexity. A reference design of a fully autonomous vehicle is proposed with sensor configuration and software/hardware architectures to support the highway socially cooperative behaviors. A Prediction- and Cost-function Based algorithm is implemented for the host vehicle to perform social behavior in two example domains of entrance ramps and lane change. A more advanced intention-integrated PCB algorithm is proposed to perform socially cooperative behaviors by understanding surrounding vehicles' intentions and showing other vehicles the host vehicle's intention. It is demonstrated in case tests, simulation tests and road tests that the iPCB algorithm is a promising solution that can greatly improve driving smoothness and safety where the host vehicle needs to socially interact with surrounding cars.

Bibliography

[1] Traffic safety facts. *National Highway Traffic Safety Administration (NHTSA)*, 2008. 1

[2] Kazi Iftekhar Ahmed. *Modeling Drivers' Acceleration and Lane Changing Behavior*. PhD thesis, Department of Civil and Environmental Engineering, MIT, Cambridge, MA, 1996. 2.6

[3] John Alcock. *Animal behavior*. Sinauer Associates Sunderland, 1989. 3.1

[4] M. Barth, F. An, T. Younglove, G. Scora, C. Levine, M. Ross, and T. Wenzel. Comprehensive modal emission model (cmem), version 2.0 users guide. *University of California, Riverside*, 2000. 4

[5] M. Bertozzi, L. Bombini, A. Broggi, M. Buzzoni, E. Cardarelli, S. Cattani, P. Cerri, A. Coati, S. Debattisti, A. Falzoni, et al. Viac: An out of ordinary experiment. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 175–180. IEEE, 2011. 1, 2.1

[6] Ryan Bradley. Tesla autopilot, the electric-vehicle maker sent its cars a software update that suddenly made autonomous driving a reality, 2016. 2.4, 3.2

[7] Frank Broz. *Planning for Human-Robot Interaction:Representing Time And Human Intention*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, December 2008. 2.8

[8] M. Buehler, K. Iagnemma, and S. Singh. *The DARPA urban challenge: autonomous vehicles in city traffic*, volume 56. Springer, 2009. 4.1

[9] A.R. Cassandra. A survey of pomdp applications. In *Working Notes of AAAI 1998 Fall Symposium on Planning with Partially Observable Markov Decision Processes*, pages 17–24, 1998. 2.8

[10] Ernst D. Dickmanns. Vehicles capable of dynamic vision: a new breed of technical beings? *Artificial Intelligence*, 103(1-2):49–76, August 1998. 1, 2.1

[11] Kurt Dresner and Peter Stone. Multiagent traffic management: A reservation-based intersection control mechanism. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 530–537. IEEE Computer Society, 2004. 2.8

[12] Kurt Dresner and Peter Stone. A multiagent approach to autonomous intersection management. *Journal of artificial intelligence research*, 31:591–656, 2008. 2.8

[13] Demeester E., A. Huntemann, D. Vanhooydonck, G. Vanacker, A. Degeest, H. Van Brussel, and M. Nuttin. Bayesian estimation of wheelchair driver intents: Modeling intents as

geometric paths tracked by the driver. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 5775–5780. IEEE, 2006. 2.7

[14] Darrell Etherington. Delphis self-driving car takes us on a stress-free tour of las vegas. *TechCrunch*, 2017. 2.1

[15] Alon Farchy, Samuel Barrett, Patrick MacAlpine, and Peter Stone. Humanoid robots learning to walk faster: From the real world to simulation and back. In *Proceedings of the 2013 international conference on autonomous agents and multi-agent systems*, pages 39–46. International Foundation for Autonomous Agents and Multiagent Systems, 2013. 2.8

[16] Denos C. Gazis, Robert Herman, and Richard W. Rothery. Nonlinear follow-the-leader models of traffic flow. *OPERATIONS RESEARCH*, 9(4):545–567, 1961. 2.6

[17] GM Press. Self-Driving Car in Cadillac's Future. `http://media.gm.com/media/us/en/cadillac/news.detail.html/content/Pages/news/us/en/2012/Apr/0420_cadillac.html` as of Oct 5, 2012. 2.2, 3.2, 4.2

[18] R. Gregor et al. Ems-vision: a perceptual system for autonomous vehicles. *IEEE Journal of ITS*, 3(1):48–59, March 2002. doi: 10.1109/6979.994795. 1, 2.1

[19] P. Grisleri and I. Fedriga. The braive autonomous ground vehicle platform. In *Intelligent Autonomous Vehicles*, volume 7, pages 497–502, 2010. 2.1

[20] Tianyu Gu and John Dolan. On-road motion planning for autonomous vehicles. *Intelligent Robotics and Applications*, pages 588–597, 2012. 2.3

[21] Randolph W Hall, Ali Nowroozi, and Jacob Tsao. Entrance capacity of an automated highway system. *Transportation Science*, 35(1):19–36, 2001. 2.5

[22] Arne Kesting, Martin Treiber, and Dirk Helbing. *General lane-changing model MOBIL for car-following models*, volume 1999 of *Transportation research record*. National Academy Press of the National Academies, Washington, DC, 2007. 2.4

[23] Rachel Kirby. *Social Robot Navigation*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 2010. 2.8

[24] Kris M. Kitani, Brian Ziebart, Drew Bagnell, and Martial Hebert. Activity forecasting. In *European Conference on Computer Vision (ECCV 2012)*, 2012. 2.7

[25] Jin-Woo Lee and Bakhtiar Litkouhi. A unified framework of the automated lane centering/changing control for motion smoothness adaptation. In *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, pages 282–287. IEEE, 2012. 2.4

[26] Anthony Levandowski and Travis Kalanick. Pittsburgh, your self-driving uber is arriving now. *UBER Newsroom*. 2.1

[27] Patrick MacAlpine, Mike Depinet, and Peter Stone. Ut austin villa 2014: Robocup 3d simulation league champion via overlapping layered learning. In *AAAI*, pages 2842–2848, 2015. 2.8

[28] J. Markoff. Google cars drive themselves, in traffic. *The New York Times*, 10:A1, 2010. 1, 2.1

[29] M. McNaughton, C. Urmson, J.M. Dolan, and J.W. Lee. Motion planning for autonomous driving with a conformal spatiotemporal lattice. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4889–4895. IEEE, 2011. 2.3, 2.4

[30] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, et al. Junior: The stanford entry in the urban challenge. *Journal of Field Robotics*, 25(9):569–597, 2008. 2.1, 4.1

[31] B. Morris, A. Doshi, and M. Trivedi. Lane change intent prediction for driver assistance: On-road design and evaluation. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 895–901. IEEE, 2011. 2.7

[32] Jose E Naranjo, Carlos Gonzalez, Ricardo Garcia, and Teresa De Pedro. Lane-change fuzzy control in autonomous vehicles for the overtaking maneuver. *IEEE Transactions on Intelligent Transportation Systems*, 9(3):438–450, 2008. 2.4

[33] Julia Nilsson and Jonas Sjöberg. Strategic decision making for automated driving on two-lane, one way roads using model predictive control. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pages 1253–1258. IEEE, 2013. 2.3

[34] US Bureau of Transit Statistics. National household travel survey. 2001. 1

[35] MM Peden, Richard Scurfield, David Sleet, Dinesh Mohan, Adnan A Hyder, Eva Jarawan, Colin D Mathers, et al. World report on road traffic injury prevention, 2004. 1

[36] Dean Pomerleau. Alvinn: An autonomous land vehicle in a neural network. 1989. 1, 2.1, 2.6

[37] N.D. Ratliff, D. Silver, and J.A. Bagnell. Learning to search: Functional gradient techniques for imitation learning. *Autonomous Robots*, 27(1):25–53, 2009. 2.6

[38] Dorsa Sadigh, Shankar Sastry, Sanjit A Seshia, and Anca D Dragan. Planning for autonomous cars that leverages effects on human actions. In *Proceedings of the Robotics: Science and Systems Conference (RSS)*, 2016. 2.8

[39] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016. 2.4

[40] D. Silver. Learning preference models for autonomous mobile robots in complex domains. Technical report, DTIC Document, 2010. 2.6

[41] R. Simmons, R. Goodwin, K.Z. Haigh, S. Koenig, and J. O'Sullivan. A layered architecture for office delivery robots. In *Proceedings of the first international conference on Autonomous agents*, pages 245–252. ACM, 1997. 4.1

[42] Jarrod M Snider et al. Automatic steering methods for autonomous automobile path tracking. *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RITR-09-08*, 2009. 4.2.1

[43] K.J. Sullivan, W.G. Griswold, Y. Cai, and B. Hallen. The structure and value of modularity in software design. *ACM SIGSOFT Software Engineering Notes*, 26(5):99–108, 2001. 4.1

[44] D. Swaroop. String stability of interconnected systems: An application to platooning in automated highway systems. *UC Berkeley: California Partners for Advanced Transit and Highways (PATH). Retrieved from: http://www. escholarship. org/uc/item/86z6h1b1*, 1997.

1, 2.1

[45] C. Thorpe, M. Herbert, T. Kanade, and S. Shafer. Toward autonomous driving: the cmu navlab. i. perception. *IEEE Expert*, 6(4):31–42, Aug. 1991. doi: 10.1109/64.85919. 1, 2.1

[46] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, et al. Stanley: The robot that won the darpa grand challenge. *The 2005 DARPA Grand Challenge*, pages 1–43, 2007. 2.1, 4.1

[47] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005. 2.3

[48] C. Urmson and W. Whittaker. Self-driving cars and the urban challenge. 23(2):66–68, March–April 2008. doi: 10.1109/MIS.2008.34. 4.3.6

[49] C. Urmson, J. Anhalt, M. Clark, T. Galatali, J.P. Gonzalez, J. Gowdy, A. Gutierrez, S. Harbaugh, M. Johnson-Roberson, H. Kato, et al. High speed navigation of unrehearsed terrain: Red team technology for grand challenge 2004. *Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-04-37*, 2004. 2.1

[50] C. Urmson, C. Ragusa, D. Ray, J. Anhalt, D. Bartz, T. Galatali, A. Gutierrez, J. Johnston, S. Harbaugh, W. Messner, et al. A robust approach to high-speed navigation for unrehearsed desert terrain. *Journal of Field Robotics*, 23(8):467–508, 2006. 4.1

[51] Chris Urmson et al. Autonomous driving in urban environments: Boss and the Urban Challenge. *Journal of Field Robotics*, 25(8):425–466, 2008. 1, 2.1, 4.1, 4.3.4

[52] A. Vahidi and A. Eskandarian. Research advances in intelligent collision avoidance and adaptive cruise control. *IEEE Transactions on Intelligent Transportation Systems*, 4(3): 143–153, 2003. 2.2

[53] P. Venhovens, K. Naab, and B. Adiprasito. Stop and go cruise control. *International Journal of Automotive Technology*, 1(2):61–69, 2000. 2.2

[54] Viknesh Vijayenthiran. Volkswagen Shows Off Self-Driving Auto Pilot Technology For Cars. MotorAuthority, 2011. 2.2, 4.2

[55] C Visvikis, TL Smith, M Pitcher, R Smith, et al. Study on lane departure warning and lane change assistant systems. *Study on lane departure warning and lane change assistant systems*, 1(1):1–124, 2013. 2.2

[56] Yunpeng Wang, E Wenjuan, Wenzhong Tang, Daxin Tian, Guangquan Lu, and Guizhen Yu. Automated on-ramp merging control algorithm based on internet-connected vehicles. *IET Intelligent Transport Systems*, 7(4):371–379, 2013. 2.5

[57] J. Wei and JM Dolan. A learning-based autonomous driver: Emulate human drivers intelligence in low-speed car following. In *SPIE Defense, Security and Sensing (submitted)*, 2010. 5.1.3

[58] J. Wei and JM Dolan. A prediction- cost function-based algorithm for robust autonomous freeway driving. In *2010 IEEE Intelligent Vehicles Symposium (submitted)*, 2010. 4.3.8

[59] Junqing Wei, John M Dolan, and Bakhtiar Litkouhi. Autonomous vehicle social behavior for highway entrance ramp management. In *Intelligent Vehicles Symposium (IV), 2013*

IEEE, pages 201–207. IEEE, 2013. 2.5

[60] Junqing Wei, Jarrod M Snider, Junsung Kim, John M Dolan, Raj Rajkumar, and Bakhtiar Litkouhi. Towards a viable autonomous driving research platform. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pages 763–770. IEEE, 2013. 1

[61] Wikipedia. Autonomous cruise control system, 2010. 2.2

[62] Q. Xu, K. Hedrick, R. Sengupta, and J. VanderWerf. Effects of vehicle-vehicle/roadside-vehicle communication on adaptive cruise controlled highway systems. pages 1249–1253, 2002. 1, 2.2

[63] Wenda Xu, Junqing Wei, John M Dolan, Huijing Zhao, and Hongbin Zha. A real-time motion planner with trajectory optimization for autonomous vehicles. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2061–2067. IEEE, 2012. 2.3