Planning under Uncertainties for Autonomous Driving on Urban Road

Liu Wei

(M.Sc., NUS)

A THESIS SUBMITTED
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
DEPARTMENT OF MECHANICAL ENGINEERING
NATIONAL UNIVERSITY OF SINGAPORE
2016

"As far as the laws of mathematics refer to reality, they are not certain; and as far as they are certain, they do not refer to reality."

— Albert Einstein, 1922

Declaration

I hereby declare that this thesis is my original work and it has been written by me in its entirety.

I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Liu Wei

July 2016

Summary

Autonomous vehicles have attracted dramatic attention during recent decades, which can not only enhance operational safety and efficiency of the transportation system, but also provide convenience to the vehicle users and improve their productivity. As an essential component for autonomous driving on the urban road, the autonomous vehicle planning system is in charge of planning appropriate maneuvers to interact with the environment and searching dynamically feasible trajectories. Acknowledging the inevitable uncertainties in reality, this thesis focuses on the problem of planning under uncertainties for autonomous driving on the urban road.

We start the discussion by reviewing the history and current status of autonomous driving technologies, where the typical autonomous vehicle planning framework that consists of a route planner, a decision maker (or behavior planner) and a motion planner is identified. We further study the uncertainties that can be involved in the planning process, which include the *internal motion uncertainty* introduced by the autonomous vehicle control system and the *external situation uncertainty* arising from the autonomous vehicle perception system.

To address the internal motion uncertainty, a risk-aware motion planning algorithm CC-RRT*-D is proposed. As motion planning algorithms continue to mature and become more sophisticated, one of the key research focuses is to guarantee that the algorithms are applicable to real-world scenarios, where the control model error and actuator noise are inevitable. The resulting motion uncertainty makes it difficult to ensure that the autonomous vehicle can always precisely locate at the desired position along the trajectory, and the collision checking of the trajectory thereby loses its reliability. Given a stochastic vehicle

dynamics model, the proposed CC-RRT*-D algorithm can derive in advance the probabilistic distributions of the vehicle states along the given trajectory. After that, the derived vehicle state distributions are checked against the obstacles to evaluate the probability of collision. As such, the motion planning can be conducted in a risk-aware manner with the planned trajectories' collision risks being bounded.

After the motion uncertainty is resolved, the road context learning and its applications for vehicle behavior analysis are discussed to facilitate the situation awareness. The awareness of the driving situation is essential for the autonomous vehicles to maneuver with safety-critical driving behaviors. The situation awareness usually follows the general principle of *Learning then Prediction*; in a sense, the reasoning of the situation evolution is based on the knowledge or experience learned beforehand. Our study follows a similar principle, where the road context is learned first by extracting the consistencies within the observed vehicle behaviors. Thereafter, the knowledge of the road context is applied to analyze the vehicle behaviors and facilitate the situation awareness.

Given the road context and vehicle behavior analysis, a situation-aware decision making algorithm is proposed to address the external situation uncertainty. Unlike human drivers, who have the ability of extracting essential information to make driving decisions, a well-designed environment model is usually required by the autonomous vehicles. As such, the concept of an *urban road situation*, which is defined as an integration of the road context and the vehicle's motion intention, is proposed first to model the urban road environment. After that, the situation-aware decision making problem is solved as a Partially Observable Markov Decision Process (POMDP), which can equip the autonomous vehicle with the ability to estimate and evaluate the outcome of driving decisions, even when the driving situation cannot be fully observed.

Last but not least, a comprehensive planning framework for autonomous driving on the urban road is introduced, which is built by integrating the algorithms discussed above. Compared to the existing planning frameworks, the proposed framework explicitly addresses the uncertainties from different aspects, i.e. internal motion uncertainty and ex-

ternal situation uncertainty, at different planning levels, i.e. motion planning and decision making, in a consistent and integrated manner.

Keywords: Autonomous Vehicle, Urban Road Environment, Motion Planning, Decision Making, Road Context Inference, Vehicle Behavior Analysis, Planning Framework Design.

Acknowledgements

Firstly, I would like to express my sincere gratitude to Prof. Marcelo H. Ang Jr., my inspirational supervisor, for his enthusiastic and continuous support and guidance. I am grateful for his constant encouragement, suggestions and critical comments for the progress of my PhD study. With his valuable supervision and personal concerns, I had a meaningful and fruitful academic journey.

My sincere thanks also goes to Prof. Emilio Frazzoli and Prof. Daniela Rus from MIT, who advise our project, keep us motivated all the time, and give priceless guidances.

I would like to thank all of our current and previous team members in SMART: Dr. Chong Zhuangjie Demian, Dr. Qin Baoxing, Shen Xiaotong, Weng Zhiyong, Scott Pendleton, Dr. James Fu, Dr. Tirthankar Bandyopadhyay, Tawit Uthaicharoenpong, Mark Ang, Cody, Hans Andersen, Feng Mengdan, and Dr. Eng Youhong. We had joy, we had fun, we had seasons in the sun. In particular, a special thanks to Dr. Kim Seong-Woo for enlightening me the first glance of research and providing constructive advices on my works.

Last but definitely not the least, I would like to thank my dear parents, for their unwavering support and love that have provided me the strengths to move forward throughout my whole life. I also owe my deep gratitude to my wife, Youyou, for accompanying me, sharing my pains and bringing me happiness everyday.

Table of contents

List of tables

List of figures

List of abbreviations

DARPA	Defense Advanced Research Projects Agency
MoD	Mobility on Demand
RRT	Rapidly-exploring Random Tree
PRM	Probabilistic Road Map
POMDP	Partially Observable Markov Decision Process
GNG	Growing Neural Gas
GP(R)	Gaussian Process (Regression)
ML	Maximum Likelihood
ROI	Region of Interest
OGM	Occupancy Grid Map
DESPOT	Determinized Sparse Partially Observable Tree

List of important symbols

x	State [Metric]
$\mathcal{X}$	State space [Metric]
$\mathcal{X}_F$	Obstacle free space
u	Control
$\mathcal{U}$	Control space
ξ	Trajectory
$C(\cdot)$	Cost function
$r(\cdot)$	Collision risk
δ	Collision risk bound
$\mathcal{N}$	Normal distribution
$\Pr(\cdot)$	Probability
$\mathbb{E}(\cdot)$	Expected value
α	Driving activity
ρ	Traffic rules
$\mathbf{E}$	Mutually exclusive region set
$\mathbf{A}$	Activity driven transition probability matrix
$\mathbf{R}$	Traffic rule driven transition probability matrix
$\mathcal{V}$	Transition probability vector
$\mathbf{MP}(\cdot)$	Motion pattern
$rc(\cdot)$	Road context
$\mathcal{RC}$	Probabilistic road context
$\mathcal{GNG}$	GNG topology graph

$\mathbf{K}$	GNG node set
$\mathbf{C}$	GNG egde set
w	Feature (or reference) vector
$\varpi(\cdot)$	Weight function
π	Control policy
$\mathcal{I}$	Motion intention set
s	State [Situation]
$\mathcal{S}$	State space [Situation]
a	Action
$\mathcal{A}$	Action space
z	Observation
$\mathcal{Z}$	Observation space
$T(\cdot)$	State transition function
$O(\cdot)$	Observation function
$R(\cdot)$	Reward function
γ	Discount factor
GR	Global route
$\mathcal{RNG}$	Road network graph

Chapter 1

Introduction

1.1 Background

The autonomous vehicle, with a goal of realizing an autonomous system that independently and robustly performs all intelligent and safety-critical driving functions, has attracted dramatic attention during recent decades. The autonomous vehicle can not only enhance operational safety and efficiency of the transportation system, but also provide convenience to the vehicle users and improve their productivity.

The early presentation of the autonomous vehicle can be traced back to the World's Fair sponsored by General Motors in 1939 [1]. The autonomous vehicle was depicted as a vehicle equipped with sensors to detect the electro-magnetic fields that control both the driving speed and the path of travel. Moreover, the Vehicle-to-Vehicle (V2V) communication was also featured as each vehicle is equipped with radios such that neighboring vehicles can keep a safe gap between each other when traveling.

Recent advances in sensing, real-time control, and computation technologies have further spurred interest in autonomous driving, and many achievements have been made in the past decades [2, 3]. Two of the largest and most recent field demonstrations of autonomous vehicles are the DARPA Grand Challenge [4] and the DARPA Urban Challenge [5], which had led an elegant setup for autonomous vehicle development. These events were organized by DARPA to stimulate the research on autonomous driving and investigate the applicability of autonomous vehicles for military service. Given the success of DARPA Grand

Figure 1.1 Autonomous vehicle fleet at Singapore-MIT Alliance for Research and Technology.

Challenge in 2005, where the autonomous vehicles were managed to navigate through a vast desert area, the DARPA Urban Challenge held in 2007 took a step further. In DARPA Urban Challenge, the autonomous vehicles, in a fully autonomous mode, had to navigate through a typical urban environment, involving off-road parking, traffic negotiation at intersections, and obstacle avoidance while obeying traffic rules. As the emphasis of these challenges was geared more towards military applications, the autonomous vehicles had to be fully self-contained in every aspect including perception, planning, and control.

In 2010, the *Autonomy for Mobility-on-Demand* project was initialized as an interdisciplinary research program at Singapore-MIT Alliance for Research and Technology [6, 7]. The Mobility-on-Demand (MoD) system features the one-way sharing with light electric vehicles, where any driver can get the vehicle at any vehicle collecting station and drive the vehicle privately toward the destination. After that, the other drivers can share the same vehicle for their own purpose. The MoD system has emerged as a viable alternative to the conventional use of private transportation. However, one of the key challenges of the MoD system is how to keep a balanced distribution of vehicles based on where they are picked up and dropped off. Thanks to the autonomous vehicles, the load rebalancing process can be handled in an efficient way, where a fleet of autonomous vehicles can be rebalanced au-

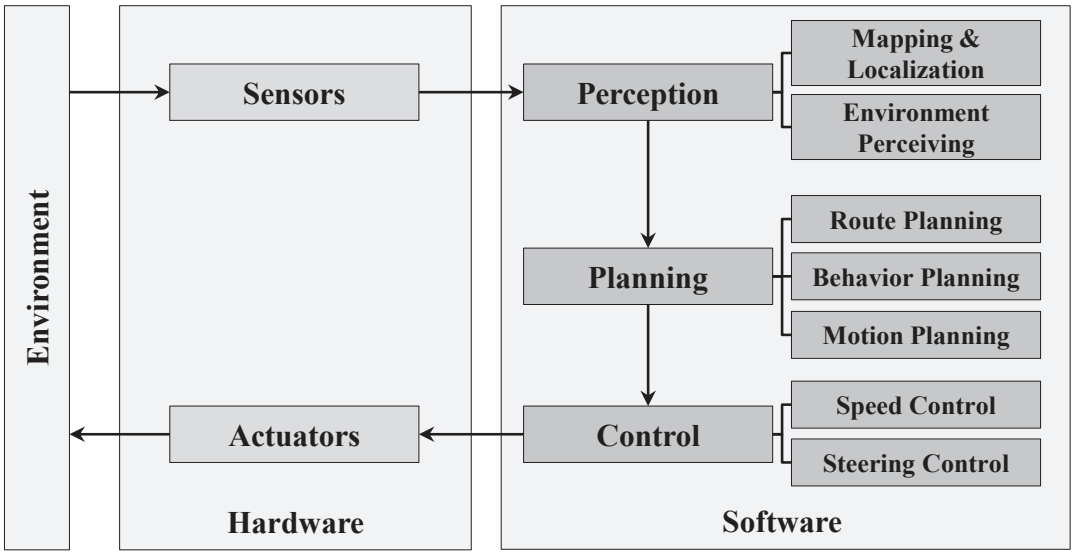


Figure 1.2 Autonomous vehicle system overview.

tomatically. As one of our research focuses, we are aiming at developing the autonomous vehicles that are equipped with full self-driving ability in an uncontrolled urban environment. To date, an autonomous vehicle fleet, including three autonomous golf-carts and one autonomous Mitsubishi iMiEV (see Figure 1.1), has been built and thoroughly tested inside and outside the campus of the National University of Singapore. The public trials in Singapore's Chinese Garden and One-North area have been conducted for the purpose of proving the concept of the MoD service and raising the public awareness of autonomous vehicles [8, 9].

1.2 Planning for Autonomous Driving on Urban Road

Generally speaking, the ability of autonomous driving safely and efficiently in a complicated environment is accomplished by integrating three major systems: Perception, Planning, and Control (see Figure 1.2). The autonomous vehicle perception system needs to be capable of estimating the autonomous vehicle's own state and perceiving the surrounding environment. The autonomous vehicle planning system is in charge of planning appropriate maneuvers to interact with the environment and searching dynamically feasible trajectories. Given the planning results, the control system, thereafter, comes into effect to compute proper throttle, brake and steering commands to drive the vehicle towards the destination.

The autonomous vehicle planning system, as suggested in DARPA Urban Challenge, is usually designed into three levels: Route Planning, Behavior Planning, and Motion Planning [10]. The route planning is defined at the mission level, which aims at searching an optimal route through the road network to reach the destination that is specified by the driving mission. The action-level or decision-level behavior planning is in charge of parsing the driving task into a sequence of actions and making proper driving decisions to interact with the environment. The motion planning, moreover, is responsible for searching the dynamically feasible trajectories to accomplish the desired driving actions, such as obstacle avoidance, three-point turn, and so forth.

Being an essential component for autonomous driving on the urban road, the autonomous vehicle planning system bridges the perception system and control system, which is fed with the perception results and outputs the driving decisions and trajectories to the control system. A properly designed autonomous vehicle planning system, therefore, needs to explicitly acknowledge the characteristics of the *urban road environment* and establish seamless connections with both the *perception* and *control* systems.

A. Characteristics of Urban Road Environment

The urban road is a unique environment with distinctive characteristics. On the one hand, the urban road is much denser and more dynamic in motion than the off-road, desert environment, because it is full of moving agents like pedestrians and vehicles. The proper perception, planning and control strategies therefore are in great need to efficiently account for the crowdedness and robustly guarantee the driving safety.

On the other hand, the urban road is more structured, such that the agents usually behave in a more organized manner by following the traffic rules or the related regulations. This favorably provides the possibility and even convenience of learning driving skills and environment contextual knowledge from the agents' behaviors, and the difficulty of situation evolution reasoning, to some extent, can be alleviated.

B. Connections between Planning and Perception

Inevitably, the autonomous vehicle perception in the dense urban road environment has to face the challenges introduced by the sensor noise and sensing occlusion. The resulting sensing limitation can dramatically weaken the autonomous vehicle's ability to perceive the surrounding environment. The reliability of the perception therefore has to be well acknowledged by the planning system, because a small perception error can propagate into vehicle planning and place the autonomous vehicle into a dangerous situation [5].

Additionally, the autonomous vehicle needs to be cognitive enough to reason about the situation evolution for safe and efficient driving. The autonomous vehicle needs to track the moving agents, together with the available contextual information, to infer the situation evolution in a probabilistic manner. The resulting uncertainty of situation evolution reasoning generally points to the difficulties in connecting the probabilistic perception with the conventional deterministic planning [11].

C. Connections between Planning and Control

The relationship between planning and control is quite obvious, where the planning results have to be executable by the control system. As such, the vehicle planning needs to be carefully conducted to explicitly address the vehicle's innate dynamic and kinematic constraints. This generally raises the demand for planning algorithms that can plan efficiently in the high-dimension control space.

Moreover, the planning results can never be perfectly executed due to the control model error and the actuator noise. Without accounting for this control limitation, the autonomous driving safety, which has been guaranteed within the planning phase, may easily break down when the planning results are committed for execution. As such, the autonomous vehicle planning system needs to recognize the imperfectness of the control system, and make sure that the planning results can stay robust to the control model error and the actuator noise.

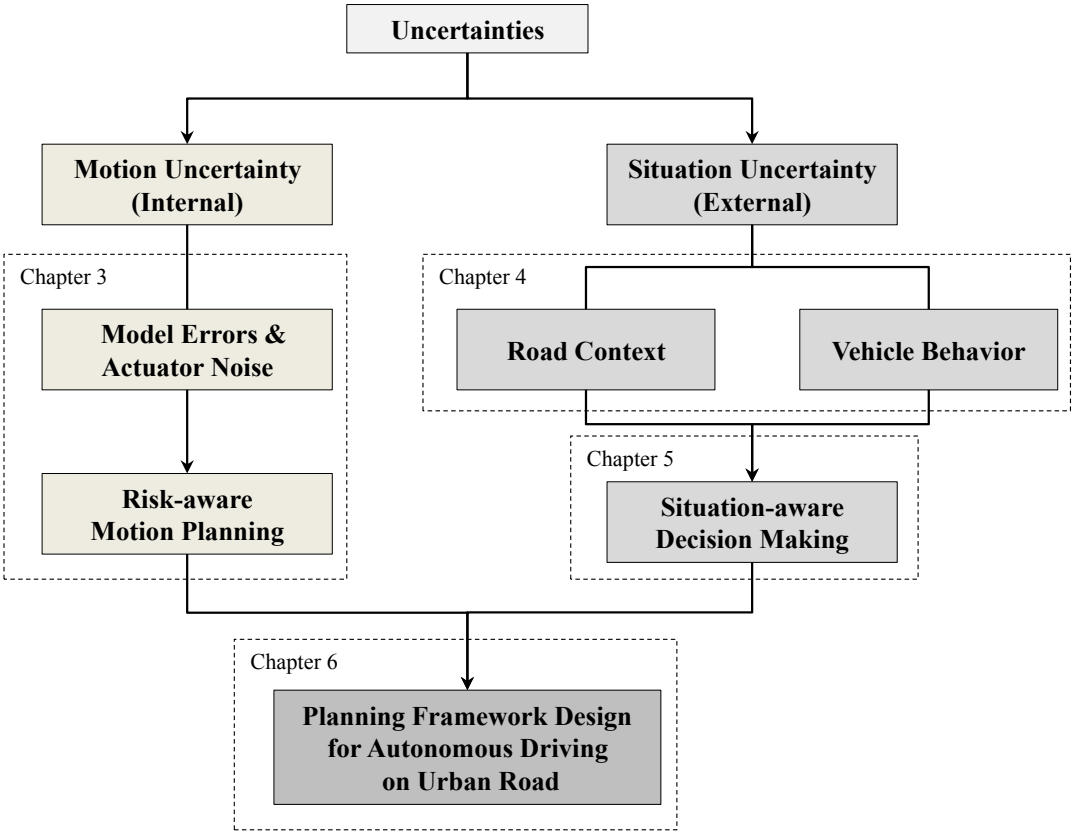


Figure 1.3 Diagram of thesis scope and outline.

1.3 Thesis Scope

The autonomous vehicle planning is a broad research area covering a large variety of topics. In this thesis, we focus on the problem of planning under uncertainties for autonomous driving on the urban road. Specifically, we aim at designing the autonomous vehicle planning system in a probabilistic manner, such that the uncertainties arising from the control and perception systems can be properly acknowledged and addressed. Along this way, the gap between the planning and perception can be filled, and the connection between planning and control will be tightened as well.

Broadly speaking, the uncertainties involved in the autonomous vehicle planning process are summarized into two major types in this thesis: the *internal motion uncertainty* and the *external situation uncertainty* (see Figure 1.3). The internal motion uncertainty, which arises internally from the control system, stands for the fact that vehicle motion unpredictably deviates from what a dynamics model predicts due to the control model error and

actuator noise. Arising from the perception system, the external situation uncertainty represents the probabilistic nature of the environment modeling and the corresponding situation evolution reasoning. This thesis addresses the internal motion uncertainty by proposing a *risk-aware motion planning algorithm* for the motion planner. The situation uncertainty is handled by the decision maker with a *situation-aware decision making algorithm* being developed. To facilitate the *situation awareness* for situation-aware decision making, the road context learning and its applications for vehicle behavior analysis are developed as well.

A. Risk-aware Motion Planning

The motion planning problem seeks to search a sequence of control inputs so as to drive the vehicle from its initial state to the goal state while obeying the kinematic or dynamic constraints and not colliding with any obstacles. As motion planning algorithms continue to mature and become more sophisticated, one of the key research focuses is to ensure the algorithms are applicable to real-world scenarios, where the control model error and actuator noise are inevitable. The resulting motion uncertainty makes it difficult to ensure that the autonomous vehicle can always precisely locate at the desired position along the trajectory, and the collision checking of the trajectory thereby loses its reliability.

In view of this limitation, a risk-aware motion planning algorithm is proposed to improve the robustness of the vehicle motion planner [12]. Given a stochastic model of the vehicle motion dynamics, we can derive in advance (i.e., before execution) the probabilistic distributions of the vehicle states along the given trajectory. Thereafter, the derived vehicle state distributions are checked against the obstacles to evaluate the probability of collision (i.e., collision risk). As such, the conventional collision checking is replaced by the risk evaluation, and the motion planning can be conducted in a risk-aware manner with the planned trajectories' collision risks being bounded.

B. Environment Modeling and Situation Awareness

Unlike human drivers, who have the ability to perceive the environment and extract essential information to make driving decisions, a well-designed environment model is always

required by most of the state-of-the-art autonomous vehicles [13–15]. As such, the concept of *urban road situation* is proposed in this thesis, which is specifically defined as the integration of the *road context* and the *vehicle motion intention* [16]. The road context can provide a compact representation of the road topology, feature the typical vehicle motion patterns and encapsulate the traffic rules that regulate the vehicle behaviors. The vehicle motion intention, moreover, can provide a semantic meaning of the obstacle vehicle's behavior, such that a reliable prediction of its future motion can be achieved.

Given the concept of urban road situation, the challenge now shifts to the situation awareness, which is also known as the reasoning of the situation evolution. Most of the existing work on situation awareness follows the general principle of *Learning then Prediction*; in a sense, the reasoning of the situation evolution is based on the knowledge or experience learned beforehand. Our research follows a similar principle, where the road context is learned first by analyzing the vehicle behavior data collected randomly during our autonomous driving trials. Thereafter, the knowledge of the road context is applied to facilitate the vehicle behavior analysis and improve the reliability of the situation awareness [17]. The rationale of this proposal can be evidenced by the fact that the vehicle behaviors are always constrained by the driving context, especially in the structured urban road environment.

C. Situation-aware Decision Making

The autonomous driving decision making module is in charge of properly maneuvering the autonomous vehicle's own behaviors and negotiating with other traffic participants. Toward this end, the full awareness of the surrounding environment is greatly needed, but difficult to achieve, as it is always aggravated by the partial observability of the situation evolution.

Given the defined urban road situation, a situation-aware decision making algorithm for autonomous driving on the urban road is developed [16]. The situation-aware decision making problem is solved as a Partially Observable Markov Decision Process (POMDP) to sophisticatedly address the uncertainties arising from the situation evolution reasoning. The proposed POMDP model can equip the autonomous vehicle with the ability to estimate and evaluate the outcome of driving decisions, even when the driving situation cannot be

exactly observed. For the purpose of the real-time application, the online POMDP solver DESPOT [18] is employed for its computational efficiency.

D. Autonomous Vehicle Planning Framework Design

Up to now, the risk-aware motion planning algorithm and the situation-aware decision making algorithm have been introduced, where the internal motion uncertainty and external situation uncertainty are addressed separately. Safe and efficient autonomous driving, however, needs the motion planning and decision making to be conducted in an integrated manner, such that the uncertainties can be handled more comprehensively. As such, a hierarchical autonomous vehicle planning framework is designed to firmly integrate the proposed decision making and motion planning algorithms.

1.4 Thesis Contributions

This thesis aims at designing the autonomous vehicle planning system in a probabilistic manner, such that the autonomous vehicle planning can be more robust to the internal motion uncertainty and more conscious of the external situation evolution. The contributions of this thesis can be summarized as:

- A risk-aware motion planning algorithm is proposed to address the motion uncertainty arising from the vehicle control system. The proposed risk-aware motion planning algorithm is robust to the control model error and the actuator noise, such that the driving safety can be improved. Compared to existing work, the proposed motion planning algorithm is less conservative, and can be easily extended for asymptotically optimal planning in high-dimension space. (Chapter 3)

- The gap between perception and planning has been properly bridged by the situation-aware decision making algorithm, while most state-of-the-art autonomous vehicle systems use separate approaches to perception and planning, with relatively little work on connecting them. The proposed algorithm is general enough for autonomous driving on the urban road, which can account for various urban road scenarios, such

as leader following on a single-lane road, traffic negotiation at the T-junction or roundabout, and so forth. (Chapter 5)

- In the course of developing the situation-aware decision making algorithm, the concept of an urban road situation is proposed, which has been demonstrated to be abstractive and informative enough for safe driving on the urban road. The correlation between the vehicle behavior and the road context has also been carefully analyzed for rational road context learning. The proposed road context inference approach is robust to the vehicle behavior data noise and can be conducted incrementally. (Chapter 4)

- System-wise, by firmly integrating the risk-aware motion planning and the situation-aware decision making, an autonomous vehicle planning framework has been properly developed for autonomous driving on the urban road. Compared to the existing autonomous vehicle planning frameworks, the proposed one explicitly addresses the uncertainties from different aspects, i.e., internal motion uncertainty and external situation uncertainty, at different planning levels, i.e., motion planning and decision making, in a consistent and comprehensive manner. (Chapter 6)

1.5 Thesis Outline

In summary, this thesis focuses on the problem of planning under uncertainties for autonomous driving on the urban road. After this introductory chapter, this thesis is organized as follows. Chapter 2 provides a general literature review about the autonomous vehicle planning techniques in the urban road environment. Chapter 3 details the risk-aware motion planning algorithm. The road context learning and vehicle behavior analysis are discussed in Chapter 4. Thereafter, the situation-aware decision making algorithm is presented in Chapter 5. The autonomous vehicle planning framework design is discussed in Chapter 6, and Chapter 7 concludes the thesis and discusses the future work. The supplementary materials can be found in the appendices.

Chapter 2

Literature Review

Being an essential component of the autonomous vehicle system, the planning function plays a vital role in autonomous driving, which enables the autonomous vehicle to safely approach the desired destination and efficiently negotiate with other traffic participants. In this chapter, the review of autonomous vehicle planning systems will be discussed first. Afterward three important autonomous vehicle planning functions, i.e., *motion planning*, *situation awareness*, and *decision making*, will be reviewed in detail.

2.1 Autonomous Vehicle Planning System

Early-stage autonomous vehicles worked in a lane-following manner, where they were usually designed to perform lane-keeping, cruise control, and some other basic operations. These systems are called semi-autonomous, in the sense that they had dedicated lanes and needed human intervention from time to time. The planning systems in these vehicles therefore are generally immature [1].

In order to spur the innovation of autonomous driving technology, DARPA launched the DARPA Grand Challenge [4] and DARPA Urban Challenge [5] in 2005 and 2007 respectively. In these two events, the autonomous vehicles were equipped with the ability to navigate through both the desert course and the urban environment without any human intervention. Especially, the success of the DARPA Urban Challenge has witnessed the impressive advances made in autonomous driving technology, which not only revealed the

challenges residing in autonomous driving [19], but also demonstrated the feasibility of self-driving in the urban environment [20]. The research carried out in DARPA Urban Challenge stood as the state-of-the-art at that time, and is still shedding light on the development of autonomous vehicles nowadays. Therefore, the planning systems suggested in the DARPA Urban Challenge will be discussed to provide the first look at autonomous vehicle planning.

2.1.1 Planning Systems in DARPA Urban Challenge

In the DARPA Urban Challenge, a Route Network Definition File (RNDF), which includes locations for intersections and selected way-points, plus details of the number of lanes and directions on road segments, is provided beforehand [21]. A description of the driving tasks that needs to be carried out is offered as well. During the competition, the driving tasks are given as individual missions consisting of reaching a set of checkpoints in a certain order. The autonomous vehicles therefore had to properly plan their maneuvers to safely reach the checkpoints and efficiently interact with the environment.

Boss, the winning entry of the DARPA Urban Challenge, employed mainly two separate strategies to navigate through the urban environment: on-road navigation (lane following, intersection negotiation, etc.) and zone navigation (parking, U-turn, etc.) [10]. For both of these driving strategies, a global mission plan is always generated as the first step by searching the road network with road blockage detection being enabled. A behavioral planner is then designed to execute the global mission plan and negotiate with the other traffic participants. The behavioral planner, more specifically, is structured into three subcomponents: lane-driving, intersection handling, and goal selection for motion planning, each consisting of a number of sub-behaviors. During the on-road driving, the autonomous vehicle is provided with a desired lane and takes the end of the lane as the sub-goal for motion planning using a lattice planning algorithm. For the zone driving, the motion planning goal is represented as the autonomous vehicle's desired pose, and the Anytime D* planning algorithm is employed for its replanning efficiency [22]. A similar framework can also be found in *Junior*, Stanford's entry of DARPA Urban Challenge, where the behavior plan-

ning is managed by a Finite State Machine (FSM) and a Hybrid A* planning algorithm is employed for off-road motion planning purposes [15].

Odin, the vehicle which placed third in the DARPA Urban Challenge, also shows a similar three-level autonomous vehicle planning framework, but with a unique structure for the behavior planner [23]. Specifically, the behavior planning is performed by considering several driving behavior options at once, where each behavior is modeled by a nested finite state machine. The final executed behavior is then chosen by a voting strategy. Placed 4th in the DARPA Urban Challenge, the *Tarlos* by the MIT team employed a slightly different approach for general navigation using a sampling-based motion planner, called closed-loop Rapidly-exploring Random Trees [24]. Instead of partitioning their planning strategy into a three-tier hierarchy, a two-tier framework, with a navigator and a motion Planner, is employed, where the navigator is an implicit combination of the mission planner and the behavior planner.

Compared to *Tarlos*, the planning framework of *Little Ben*, the Ben Franklin Racing Team's Entry in the DARPA Urban Challenge, is more dedicated [25]. Similar to the other teams, the first-stage mission planning is to search the global route consisting of a set of desired way-points. The behavior planner, thereafter, comes into effect to choose one of the four purposely designed motion planners for each specific driving scenario, which includes the lane-following planner, the U-Turn planner, the intersection planner and the zone-parking planner. Lastly, the autonomous vehicle planning system developed by Team Cornell can also be split into three primary layers, including the behavioral layer, the tactical layer and the operational layer [26]. The behavioral layer functions as a mission planner, which is to combine the mission information with sensing information to decide which driving route and behavior state should be executed. The tactical layer then plans a contextually appropriate set of actions to accomplish the planned behavior state. With the road constraints and nearby obstacles being accounted for, the operational layer interleaves motion planning and vehicle control to translate these abstract actions into actuator commands.

2.1.2 Discussion

After reviewing the planning systems of the six finishers in the DARPA Urban Challenge, it is observed that the autonomous vehicle planning system is typically consisted of three sub-systems: route planner, decision maker (or behavior planner), and motion planner. The route planner is defined at the mission level, which is responsible for searching an optimal global route through the road network to reach the destination. The action-level or decision-level behavior planner is in charge of parsing the driving mission into a set of driving actions and efficiently making driving decisions to interact with the environment. Lastly, the motion planner is to search the dynamically feasible trajectories and accomplish the desired driving behaviors. Inheriting from the experience of the DARPA Urban Challenge, most of the state-of-art autonomous vehicle planning systems follow a similar framework [27, 28].

Admittedly, the autonomous vehicle planning strategies proposed in the DARPA Urban Challenge have proven their proper functionality, but they are still not qualified enough for autonomous driving in a real and uncontrolled urban environment. The driving tasks encountered in the DARPA Urban Challenge were provided beforehand; the planning strategies therefore can be carefully designed. As such, the ability of the designed planning strategies to account for various unexpected driving situations is doubtful. Moreover, the various uncertainties arising from the control and perception systems are occasionally overlooked, where the autonomous vehicle is assumed to operate in a fully observed environment, which, however, is rarely true in the real world. Therefore, the autonomous driving planning strategies are designed in a probabilistic manner in our study, and we aim at equipping the autonomous vehicle with the ability of learning, such that the autonomous vehicle can safely and efficiently handle unexpected driving situation.

2.2 Motion Planning

The motion planning problem seeks to search a sequence of control inputs so as to drive the vehicle from its initial state to the goal state while obeying the dynamic constraints and not colliding with any obstacles. The algorithm to address this problem is said to

be complete if it terminates in finite time, returning a valid solution if one exists, and failure otherwise [29]. A central theme throughout motion planning is to transform the continuous model into a discrete one [30]. There exist two main alternatives to achieve this transformation: combinatorial motion planning and sampling-based motion planning. The combinatorial motion planning algorithms build a discrete representation that *exactly* represents the original problem, then utilize searching algorithms for discrete planning. On the other hand, the sampling-based motion planning algorithms use a collision checking module to model the configuration space and conduct discrete searching on the samples [30].

2.2.1 Combinatorial Motion Planning Algorithms

Combinatorial motion planners came around with the development of cell decomposition methods [31]. These approaches aim at relaxing the completeness requirement to resolution completeness, namely the ability to return a valid solution, if one exists, when the resolution parameter of the algorithm is set fine enough. The combinatorial motion planning algorithms heavily rely on an explicit representation of the obstacles in the workspace, which is utilized to construct a solution directly. This may result in an excessive computational burden in high dimensions, and in environments described by a large number of obstacles. Avoiding such a representation is the main underlying idea leading to the development of the sampling-based algorithms.

2.2.2 Sampling-based Motion Planning Algorithms

Instead of using an explicit representation of the environment, the sampling-based motion planning algorithms rely on random samples and a collision checking module to model the configuration space, where a set of points sampled from the obstacle-free space are connected to build a graph of feasible trajectories [32–35]. This graph is then used to construct the solution to the original motion planning problem. The sampling-based motion planning algorithms have demonstrated their probabilistic completeness, namely with enough samples, the probability that it finds an existing solution converges to one. The most influential state-of-art sampling-based motion planning algorithms are Probabilistic

RoadMaps (PRM) [32], Rapidly-exploring Random Trees (RRT) [33, 34] and Optimal Rapidly-exploring Random Trees (RRT*) [35]. The philosophy of generating the obstacle-free random samples is essentially identical in these algorithms, but the way in which they construct a graph connecting these samples is different.

A. Multi-query Method: Probabilistic Roadmaps

The PRM algorithm and its variants consist of two phases: construction and query [32]. The algorithms first construct a graph (or roadmap), which represents a rich set of collision-free trajectories, and then answer queries by searching a path that connects the initial state with the goal state through the roadmap. The PRM algorithm has been reported to perform well in high-dimensional state spaces. Furthermore, the PRM algorithm is probabilistically complete, such that the probability of failure decays to zero exponentially with the number of samples used in the construction of the roadmap [36].

B. Single-query Method: Rapadily-exploring Random Trees

While multiple-query methods had been a focus of robotics research and lots of improvements were suggested in the last two decades [37, 38], many online motion planning problems do not require multiple queries, because the environment can be dynamic and the environment may not be available a priori. Therefore, computing a roadmap could be computationally challenging or even infeasible in these applications. Tailored mainly for these applications, the incremental sampling-based planning algorithms, such as RRT and RRT*, have emerged as an online, single-query counterpart to PRMs [33–35]. The incremental nature of these algorithms avoids the necessity to set the number of samples beforehand, which enables online implementation with the solution being found once the set of trajectories built by the algorithm is rich enough. The RRT algorithm has also been shown to be probabilistically complete, with an exponential rate of decay for the probability of failure. The basic version of the RRT algorithm has been extended in several directions and found many applications in the field of robotics domain and elsewhere [39–43].

2.2.3 Optimal Motion Planning Algorithms

In addition to the completeness, the quality of the solution returned by a motion planning algorithm is also important. The problem of computing optimal motion plans has been proven to be very challenging even for basic cases [44].

In the context of combinatorial motion planning algorithms, the guarantee of optimality is based on graph searching algorithms, such as Dijkstra [45], and A* [46], applied over a finite discretization generated beforehand. These algorithms have received a large amount of attention and have been extended to operate in an anytime fashion [47] and deal with dynamic environments on various robotic platforms [48–50].

For sampling-based motion planning algorithms, it is often the case that the feasible trajectory is found quickly, so additional available computation time can then be devoted to improving the solution with heuristics [24]. Some heuristics are proposed in [51] to bias the RRT tree growth towards the regions that result in low-cost solutions. Ferguson *et al.* in [52] considered running the RRT algorithm multiple times in order to progressively improve the quality of the solution, which showed that each run of the algorithm results in a solution with smaller cost. This procedure, however, cannot be guaranteed to converge to an optimal solution. Criteria for restarting multiple RRT runs were also studied in [53]. A more recent approach is the transition-based RRT (T-RRT) in [54], which is designed to combine the RRT's rapid exploration properties with the stochastic global optimization methods.

The most promising optimal sampling-based planning algorithms are proposed and detailed in [35]. The broadly used sampling-based motion planning algorithms (e.g. PRM and RRT) have been proved to converge almost surely to a non-optimal value. The optimal versions of these algorithms, e.g. PRM* and RRT*, are then introduced, which are able to provably guarantee the asymptotic optimality of the returned solutions. Thanks to the asymptotically optimal property, an anytime version of RRT*, namely Anytime RRT*, is proposed in [55] and has demonstrated its applicability for real-time robotic planning.

Recently, there is also an increasing number of works with an objective of speeding up the optimality convergence of RRT* [56, 57].

2.2.4 Motion Planning under Uncertainties

As motion planning algorithms continue to mature and become more sophisticated, one of the key research focuses is to ensure that the algorithms are applicable to real-world scenarios. In light of this, the problem of motion planning under control disturbance and sensing noise is attracting increasing research interest recently. Some existing approaches tend to approximate the vehicle dynamic model to maintain an explicit estimate of the vehicle state and conduct probabilistic planning on the state estimate to guarantee the success [58–60]. Some other approaches tend to blend planning and control by returning a global control policy over the entire space. Markov Decision Processes (MDPs), for instance, are able to take account of motion uncertainty and optimize the probability of success [61].

For systems with Linear dynamics, Quadratic cost and Gaussian noise (LQG), the control policy can be obtained by employing a Kalman filter [62] to maintain a Gaussian state estimate and evaluate the planning performance over the mean value of the estimate [58]. Blackmore $et\ al.$ presented a chance constraint approximation approach in [59] to compute probabilistically robust trajectories under Gaussian disturbance. To get rid of the LQG limitation, this work was further improved in [63, 64] to handle non-linear systems with non-Gaussian disturbance. In order to compute trajectories with smaller sub-optimality, Ono $et\ al.$ in [60] presented an Iterative Risk Allocation (IRA) approach by intelligently allocating risk limit for each chance constraint.

Recognizing the research improvements obtained in sampling-based planning algorithms for deterministic systems, many research works have been proposed to extend sampling-based planning algorithms to handle stochastic systems. The employment of the sampling-based planning algorithms can help to leverage their benefits of both the easier scalability to the high-dimensional space and the reduced planning horizon due to the piecewise path property. Following the same principle as the earlier chance constraint approximation approach, Luders $et\ al.$ proposed a Chance Constraint RRT (CC-RRT) al-

gorithm in [65] to build the RRT tree with a bounded collision probability. Similarly, the Particle-RRT proposed in [66] utilizes particles to represent the state distribution along the RRT tree for space exploring and trajectory evaluation. Moreover, the Belief Roadmap in [67] performs a belief space planning using factorized covariance for linear systems, where the computational efficiency can be dramatically improved. While great improvements have been achieved, these approaches can only return sub-optimal trajectories due to the non-optimality of RRT and PRM [35].

With the emergence of the asymptotically optimal sampling-based motion planning algorithms (e.g. RRG, RRT*, PRM*), the Rapidly-exploring Random Belief Tree (RRBT) algorithm proposed in [68] aims at constructing and refining a belief tree by using the RRT* algorithm to address the motion and sensing uncertainty. Recent work in [69] improves the CC-RRT by using the RRT* framework instead, which accomplished a robust planning algorithm with asymptotic optimality being guaranteed.

2.2.5 Discussion

As reviewed above, the motion planning problem has been researched for decades with an lot of promising algorithms being proposed. The combinatorial motion planning algorithms have demonstrated their proper applicability and efficiency for planning in the low-dimensional space. The dimension constraint is then released by the sampling-based motion planning algorithms, which use random samples and a collision checking module to model the configuration space. The requirements of completeness and optimality have also been properly considered and addressed.

As motion planning algorithms become widely applied to solve real world planning problems, the various uncertainties involved in the planning process become inevitable. As a trend, the extension of sampling-based motion planning algorithms for motion planning under uncertainties has made impressive achievements, but most of existing algorithms lack a proper state propagation and neglect the dependency for state distribution modeling. In sight of this limitation, our research on a risk-aware motion planning algorithm will be discussed in Chapter 3, where the RRT* algorithm is extended to solve the problem of

planning under motion uncertainty, and the state dependency is explicitly considered for state distribution modeling.

2.3 Situation Awareness

Situation awareness, which is also known as the reasoning of the situation evolution, is a prerequisite for autonomous vehicle planning. Without a comprehensive understanding of the situation evolution, it is hardly possible to evaluate the potential outcomes of the planned driving actions. As such, the autonomous vehicle planning can only be conducted in a greedy and reactive manner, and the driving safety and efficiency cannot be guaranteed.

In order to be aware of the driving situation, the autonomous vehicle needs to properly acknowledge the *road context* and analyze the other *vehicles' driving behaviors* [16].

2.3.1 Vehicle Behaviors Analysis

Analyzing the vehicle behavior has emerged as an active and challenging research topic nowadays. The studies in this area take a variety of approaches to characterize the vehicle behaviors [70]. Early studies aim at labeling vehicle behaviors as either normal or abnormal for the purpose of driving assistance [71, 72]. Thereafter, there came some works of identifying the specific driving maneuvers, such as overtaking [73], turning [74] and lane changing [75]. Most recently, more and more research weight is placed on the inference over the semantic meaning of the vehicle behavior, i.e., motion intention. As such, a more robust behavior classification and a more accurate motion prediction can be achieved [76–78].

Generally speaking, most of the studies in the literature follow the rule of *Learning then Reasoning*, in the sense that the analysis of the vehicle behavior is driven by the learned vehicle dynamics or behavior model. The trajectory modeling approaches, for instance, target predicting a vehicle's future motion for up to several seconds using the learned prototype trajectories [79, 80]. Depending on the expected learning output, these studies can be broadly categorized into three major categories: motion pattern or physics model learning [81], behavior pattern learning [82, 83] and contextual information extraction [84, 85].

By observing the vehicle's dynamics over time, the typical motion patterns can be generalized. The advantage of these techniques is that they are generally applicable and do not require manual re-training for new scenes. A non-parametric Gaussian Process learning approach is proposed in [81] to capture the moving obstacles' motion pattern, which is then demonstrated in [86] for the safe avoidance of dynamic obstacles. As a limitation, the motion prediction, however, can only be achieved in a short-term manner due to the lack of comprehensive vehicle behavior understanding.

In view of the limitation of vehicle motion pattern learning, behavior pattern learning aims at extracting the semantically meaningful patterns from the observed trajectories. The Growing Hidden Markov Model (GHMM) is utilized in [82] for vehicle behavior learning. A topological map of spatial states is built first. It is then connected through the GHMM to represent the vehicle behaviors. Similarly, a three-level hierarchical learning framework using HMM is developed in [83] to robustly cluster and model vehicle behavior patterns. The first level accounts for reasoning about the goals of the scene; the second level performs spatial clustering of the trajectories; and the final level leverages the resulting clusters for vehicle behavior pattern modeling. Moreover, the vehicle behaviors are always correlated, where each vehicle's behavior is affected by the others. To model this correlation, the Coupled Hidden Markov Model (CHMM) is employed in [78] to model each vehicle's motion intention as a dependency on its neighbors.

Last but not least, the contextual information can help to narrow down the reasoning scope of the vehicle behaviors, while increasing robustness [87]. A machine learning approach to infer the road network is studied in [84], which targeted improving the road network accuracy for better driving assistance. As an extension of contextual information for autonomous driving, [85] proposed a generic way of employing contextual knowledge, such as the lane trajectory ahead, to anticipate yaw rate and acceleration of other traffic participants.

2.3.2 Road Context Learning

In sight of the obvious benefits of contextual information for autonomous driving, the road context is usually manually abstracted and represented as a prior Road Network Graph (RNG) in most of the state-of-art autonomous vehicle systems [13]. While the manually specified urban road rules have demonstrated their proper functionality for autonomous driving in the DARPA Urban Challenge, exhaustively listing all possible rule specifications is time-consuming and carries the risk of introducing unwanted bias when the environment is changed. Therefore, an automatic approach for road context learning is greatly needed.

While great improvements in road context learning have been achieved, most existing approaches focus on geometrical and topological road network learning only. The aerial and Synthetic Aperture Radar (SAR) image-based approaches, for instance, are proposed to extract road networks by employing image processing techniques. Although these approaches have been practically utilized for driving assistance [88, 89], the further extension for autonomous driving is obstructed by their poor accuracy.

With the emergence of the mass-manufactured Global Positioning System (GPS) device, there have been some research efforts on utilizing the abundant GPS data to infer the road context [90, 91]. While the accuracy shortcoming can be well compensated, its response to the road context change, however, can be tardy.

Another intuitive approach to road context learning is traffic signs recognition [92, 93], which is able to provide a number of valuable traffic information, like the speed limit, driving directions, and so forth. These approaches, however, might become tricky when the traffic signs become not so obvious to follow, for instance, the traffic signs are damaged, or the traffic sign styles are not consistent in different regions.

Thanks to recent research advances in vehicle detection and tracking, the trajectory modeling approach has been widely adopted for road context analysis [83]. The motivation of these approaches is that the vehicle behavior and the road context are always correlated. In these works, the complete vehicle trajectories are usually required as a preliminary for the prototype trajectory extraction. The collection of complete trajectories, however, can

be troublesome in cluttered urban environment, where missing observations are quite common due to sensing occlusion. Although infrastructural sensors, e.g., surveillance cameras, can favorably alleviate this limitation [94], their extension to a larger operating area is impeded by their static property. Recognizing the difficulty of performing trajectory collection, Agamennoni *et al.* proposed a two-stage approach, i.e., sampling and linking, for road map inference in [84, 95], but the learning process is in a deterministic manner, and the sensing uncertainty is overlooked.

2.3.3 Discussion

Proper situation awareness requires both the knowledge of driving context and the analysis of other vehicles' behaviors. The review of the existing works reveals the correlation between the road context and vehicle behavior, where the road context can be inferred from the vehicle behaviors and the learned road context is able to facilitate the vehicle behavior analysis.

While significant achievements have been made in both road context learning and vehicle behavior analysis, a general approach to interleave them is still lacking. Motivated by this fact, our research work into road context inference and its application to vehicle behavior analysis will be presented in Chapter 4.

2.4 Decision Making

Decision making is the process of identifying and choosing alternatives based on the values and preferences of the decision maker, and has been widely employed in various fields. As an essential component of the autonomous vehicle planning systems, the autonomous driving decision making has been actively researched in the past decade. In the literature, the autonomous vehicle decision maker is usually designed in either a reactive or a deliberative manner. The reactive decision maker relies on local or temporal information that is collected online, and links the response or reaction with the sensing by combining the decisions made by both the human and the machine [96]. On the other hand, given the

proper reasoning of the situation evolution, the decision maker can think and act deliberately [97, 98].

2.4.1 Reactive Decision Making

The most common approach for autonomous driving decision making is to manually tailor the specific action sets for different driving scenarios using a Finite State Machine or similar frameworks like behavior trees. For instance, the vehicle *Junior* in the DARPA Urban Challenge designed a state machine for behavior control, where the behavior transitions were triggered by events characterized by the temporal sensing information [15]. As a milestone of autonomous vehicle development, this strategy has been widely adopted by many other vehicle platforms [99].

While great achievements have been made, these approaches are functioning in a reactive and greedy manner, which lacks the comprehensive understanding of the environment. The absence of the full situation awareness can make the driving decisions danger-prone, which was evidenced by an incident during the DARPA Urban Challenge [19].

2.4.2 Deliberative Decision Making

Recognized the shortcomings of reactive decision making, the research interest in extending road context learning and vehicle behavior analysis for deliberative decision making is increasing [78, 97, 100]. Road context learning and vehicle behavior analysis can favorably improve the robustness and the accuracy of the situation evolution reasoning, such that the autonomous driving decision making can be conducted safely and intelligently by evaluating the potential outcomes. The inevitable uncertainties arising from road context learning and vehicle behavior analysis, however, need to be carefully addressed within the decision making phase, which has been recognized as a challenge for deliberative decision making [98].

As a principled mathematical framework for planning under uncertainty, POMDP and its variants, which can properly integrate the situation awareness and decision making, have emerged as a novel solution for autonomous driving decision making. An intention-aware decision making algorithm using MOMDP [101] is presented in [76] for autonomous vehi-

cle speed control, where the vehicle motion is assumed to be driven by the corresponding motion intentions that are defined as their hidden destinations. A similar work was then applied to a clustered pedestrian environment in [102].

Besides intention-aware decision making, Brechtel *et al.* presented a probabilistic decision making module using continuous POMDP in [103], whose interest is in balancing the exploration and exploitation to account for the incomplete perception issue. The vehicle behavior uncertainty, however, is not explicitly considered. Similarly, Wei *et al.* presented a QMDP-based approach to account for the perception limitation in [104]. Moreover, the work proposed in [105] aims to employ POMDP to solve the decision making problem for lane change behavior, which, however, is too specific for other driving scenarios.

2.4.3 Partially Observable Markov Decision Process

While the research interest in applying POMDP for autonomous vehicle decision making is increasing, there still arise many concerns about the POMDP's computational tractability issue. Generally speaking, POMDP provides a general sequential decision making framework to yield globally optimal policies subject to various uncertainties, but it is also computation-intensive and scales poorly with increasing number of state (curse of dimensionality) and planning horizon (curse of history) [106].

Thanks to the recent advances in POMDP research, a variety of general and domain-specific POMDP solvers, which seek to approximate the solution to improve the computational tractability [107, 108], have become available. Depending on the working principle, most of the existing POMDP solvers can be classified into two major types: offline policy computation and online searching.

For offline policy computation, the agent computes beforehand a policy contingent upon all possible future scenarios and executes the computed policy based on the observations received. Although offline policy computation algorithms have achieved dramatic progress in computing near-optimal policies [108–110], they are difficult to scale up to very large POMDP, because of the exponential number of future scenarios that must be

considered. Moreover, small changes in the environment's dynamics require recomputing the full policy.

On the other hand, online searching algorithms interleave planning and execution, and try to circumvent the complexity of computing a policy by planning online only for the current belief state. In other words, the agent searches for a single best action for the current belief, executes the action, and updates the belief [111]. This process repeats at the new belief until the task is accomplished. As such, online POMDP planning is more applicable to the dynamic environment compared to offline policy computation approaches, because the environment changes can be easily handled without much more computation.

2.4.4 Discussion

While the applicability of reactive decision making has been widely recognized, the lack of comprehensive environment understanding makes the driving decisions sub-optimal in most cases. In contrast, POMDP has emerged as a general framework to solve the environment evolution reasoning and decision making in a deliberative manner. The recent research advance in online POMDP also fosters its applications in various realistic problems. In light of these achievements, a situation-aware decision making algorithm using online POMDP will be discussed in Chapter 5.

2.5 Summary

This chapter studied the history and current status of autonomous vehicle planning system and discussed the ongoing research trends. Three autonomous vehicle planning functions, including motion planning, situation awareness, and decision making, were highlighted and reviewed in detail. The studies on these three topics are the main body of this thesis and will be presented in the following chapters.

Chapter 3

Risk-aware Motion Planning under Motion Uncertainty using CC-RRT*-D

3.1 Introduction

As motion planning algorithms continue to become more sophisticated, a key research focus is ensuring the algorithms are applicable to real-world scenarios, in which motion uncertainty is inevitable. The motion uncertainty stands for the fact that vehicle motion unpredictably deviates from what a dynamics model predicts due to the model error or actuator noise. The motion uncertainty makes it difficult to ensure that the autonomous vehicle can always precisely locate at the desired position along the trajectory, and the collision checking thereby loses its reliability.

In view of this limitation, this chapter considers the problem of motion planning for linear systems subject to Gaussian motion disturbance, and a risk-aware planning algorithm, CC-RRT*-D, is proposed. CC-RRT*-D employs the chance constraint approximation and leverages the asymptotically optimal property of the RRT* algorithm to search risk-aware and asymptotically optimal trajectories. More specifically, the RRT* algorithm is employed for space exploration and trajectory searching, and the state propagation is conducted to derive in advance (i.e., before execution) the state distributions along the RRT* tree. The chance constraint approximation, thereafter, will come into effect to evaluate the collision risk of the RRT* tree. Consequently, a probabilistically feasible tree

will be expanded incrementally over the entire space. By explicitly considering the state dependency within the state propagation process, the over-conservative problem of chance constraint approximation can be alleviated. Extensive computational experiments show that CC-RRT*-D is more efficient and less conservative compared with the existing algorithms. The real experiment of obstacle avoidance on an autonomous vehicle suggests that the proposed algorithm is applicable to real-time motion planning.

The remainder of this chapter is organized as follows. Section 3.2 formulates the motion planning problem to be solved. The details of the proposed risk-aware motion planning algorithm are discussed in Section 3.3. The experiment results are presented in Section 3.4, and this chapter is concluded in Section 3.5

3.2 Problem Statement

Let $\mathcal{X} \subset \mathbb{R}^{n_x}$ be the state space and let $\mathcal{U} \subset \mathbb{R}^{n_u}$ be the control space, where n_x and n_u denote the dimension of state space and control space, respectively. Let the obstacle-free space be represented as $\mathcal{X}_F \subset \mathcal{X}$, which thereby can take the form as,

$$\mathcal{X}_F = \mathcal{X} - \mathcal{X}_1 - \ldots - \mathcal{X}_K, \tag{3.1}$$

where $\mathcal{X}_k, k \in \{1, ..., K\}$ are convex polyhedral that models the obstacle-occupied region, and operator "$-$" denotes set subtraction.

We assume that applying a control input $u_t \in \mathcal{U}$ at stage t brings the vehicle from state $x_t \in \mathcal{X}$ to state $x_{t+1} \in \mathcal{X}$ according to the following stochastic model,

$$\begin{aligned} x_{t+1} &= f(x_t, u_t, m_t), \\ m_t \sim \mathcal{N}(0, M_t),\ & x_0 \sim \mathcal{N}(\bar{x}_0, \Sigma_{x_0}), \end{aligned} \tag{3.2}$$

where $m_t \in \mathbb{R}^{n_u}$ is the Gaussian motion disturbance with covariance M_t that is imposed on the control input and x_0 is the initial state modeled as a Gaussian distribution with mean $\bar{x}_0$ and covariance Σ_{x_0}. The state transition function f is assumed to be either linear or locally well approximated by its linearization.

Let $\mathcal{X}_{goal}$ denote the goal region, and let ξ denote a trajectory defined as a series of states and control inputs $\xi = \{x_0, u_0, ..., x_T, u_T\}$ that starts from x_0 and ends at the goal region as $x_T \in \mathcal{X}_{goal}$. The risk-aware motion planning under motion uncertainty seeks to solve a chance-constrained optimization problem as,

$$\xi^* = \arg\min_{\xi} \mathbb{E}\{\sum_{t=0}^{T} C(x_t, u_t)\Pr(x_t)\},$$
$$\text{subject to Equation } (3.2),$$
$$\text{and } \Pr(x_t \notin \mathcal{X}_F) \leq \delta, \forall t \in \{0, 1, \ldots, T\}, \tag{3.3}$$

where $C(x, u) : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^+$ is the cost function to be minimized. Due to the motion uncertainty, the state x_t at stage $t \in \{0, 1, \ldots, T\}$ cannot be perfectly measured, so it is therefore modeled as a probabilistic distribution $\Pr(x_t)$. The collision checking over x_t is thereby replaced by the collision risk evaluation $\Pr(x_t \notin \mathcal{X}_F) \leq \delta, \forall t \in \{0, 1, \ldots, T\}$, where δ is a bound for the collision risk. The optimal trajectory, therefore, is considered as the one providing the minimum expected cost while maintaining a bounded collision risk.

3.3 Risk-aware Planning under Motion Uncertainty

This section details the risk-aware planning algorithm for a system under motion uncertainty, where the chance constraint approximation, the conditional state propagation and the proposed motion planning algorithm will be discussed step by step.

3.3.1 Chance Constraints

Given a convex polyhedral obstacle $\mathcal{X}_k \subset \mathcal{X}, k \in \{1, \ldots, K\}$ defined by L linear segments, the event that the vehicle collides with obstacle $\mathcal{X}_k$ at stage t can be modeled as a conjunction of linear constraints on the state x_t,

$$\bigwedge_{i=1,\ldots,L} \mathbf{a}_i^{\mathrm{T}} x_t < \mathbf{b}_i, \tag{3.4}$$

where $\mathbf{a}_i$ and $\mathbf{b}_i$ are the vectors that model the ith linear constraint.

If x_t is modeled as a probabilistic distribution at stage t, the collision with the obstacle then can be evaluated as the probability of Equation (3.4) being satisfied. Recognizing that the obstacle is assumed to be convex, the probability of any of the linear constraints in Equation (3.4) being satisfied is an upper bound on the obstacle-colliding probability,

$$\Pr(x_t \in \mathcal{X}_k) = \Pr(\bigwedge_{i=1,\dots,L} \mathbf{a}_i^{\mathrm{T}} x_t < \mathbf{b}_i) \leq \Pr(\mathbf{a}_i^{\mathrm{T}} x_t < \mathbf{b}_i). \tag{3.5}$$

Assume that the state $x_t \sim \mathcal{N}(\bar{x}_t, \Sigma_{x_t})$ follows a Gaussian distribution with mean $\bar{x}_t$ and covariance Σ_{x_t}, and define an affine transform as $AT_{it} = \mathbf{a}_i^{\mathrm{T}} x_t - \mathbf{b}_i$. The probability of the ith linear constraint being satisfied then can be calculated as,

$$\Pr(AT_{it} < 0) = [1 - \mathbf{erf}((\mathbf{a}_i^{\mathrm{T}} \bar{x}_t - \mathbf{b}_i)/\sqrt{2\mathbf{a}_i^{\mathrm{T}} \Sigma_{x_t} \mathbf{a}_i})]/2, \tag{3.6}$$

where $\mathbf{erf}$ denotes the standard error function [112]. This result can be inserted into the usage of Boole's inequality in Equation (3.5) to give an approximated evaluation of the obstacle colliding probability as,

$$\Pr(x_t \in \mathcal{X}_k) = \min_{i \in \{1,\dots,L\}} \Pr(AT_{it} < 0). \tag{3.7}$$

Extend the collision risk evaluation to multiple obstacles, and let $\Pr_t^{[k]} = \Pr(x_t \in \mathcal{X}_k), k \in \{1, \dots, K\}$ denote the probability of x_t colliding with obstacle k, which has been addressed in Equation (3.7). The overall collision risk is then loosely defined as,

$$\Pr(x_t \notin \mathcal{X}_F) = \max_{k \in \{1,\dots,K\}} \Pr_t^{[k]}, \tag{3.8}$$

which might not reflect the true risk, but provides a conservative approximation by considering the worst case.

3.3.2 Conditional State Propagation

In principle, the proposed approach applies to linear dynamics, but considering the fact that the vehicle will always be controlled to stay close to the nominal trajectory during

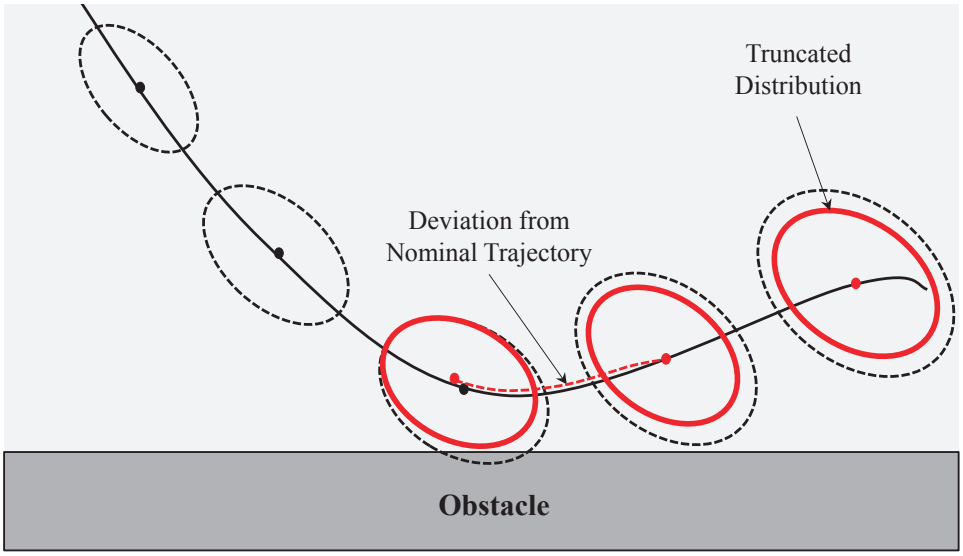


Figure 3.1 Illustration of conditional state propagation: conditional state propagation based state distribution is represented as red ellipses, and the black dashed ellipses denote the state distribution without using conditional state propagation.

execution, we assume that the non-linear vehicle model can be locally linearized around the nominal trajectory as,

$$\hat{x}_t = A_t \hat{x}_{t-1} + B_t \hat{u}_t + V_t m_t, \tag{3.9}$$

where

$$\begin{aligned} \hat{x}_t &= x_t - x_t^*, \\ \hat{u}_t &= u_t - u_t^* \end{aligned} \tag{3.10}$$

is the deviation from nominal state x_t^* and control u_t^* respectively, and

$$\begin{aligned} A_t &= \frac{\partial f}{\partial x}(x_{t-1}^*, u_{t-1}^*, 0), \\ B_t &= \frac{\partial f}{\partial u}(x_{t-1}^*, u_{t-1}^*, 0), \\ V_t &= \frac{\partial f}{\partial m}(x_{t-1}^*, u_{t-1}^*, 0) \end{aligned} \tag{3.11}$$

are the Jacobian matrices of f along the nominal trajectory.

Due to the motion uncertainty, the true state x_t as well as the state deviation $\hat{x}_t$ cannot be perfectly measured, so a Kalman Filter is employed for state estimation by approximating

the distribution of $\hat{x}_t$ as Gaussian with mean $\bar{\hat{x}}_t$ and covariance $\Sigma_{\hat{x}_t}$ ($\Sigma_{\hat{x}_t}$ will be represented as Σ_t in what follows for notational clarity). Many previous works use the maximum likelihood measurement model to update the estimate, which, however, cannot reflect the true state distributions. Instead, the maximum likelihood model only measures how the state distribution can be inferred in the best case. Hence the estimate of $\hat{x}_t$ is given as following without measurement update,

$$\begin{aligned} \bar{\hat{x}}_t &= A_t \bar{\hat{x}}_{t-1} + B_t \hat{u}_t, \\ \Sigma_t &= A_t \Sigma_{t-1} A_t^T + V_t M_t V_t^T. \end{aligned} \tag{3.12}$$

Thereafter, the probabilistic distribution of x_t can be given as, $x_t \sim \mathcal{N}(\bar{\hat{x}}_t + x_t^*, \Sigma_t)$, where $\bar{x}_t = \bar{\hat{x}}_t + x_t^*$ denotes the mean of x_t.

The state propagation in Equation (3.12) has shown its functionality in many existing works; the state dependency, however, is not properly considered. Recalling the state transition function in Equation (3.2), the state x_{t-1} needs to be collision-free in order to propagate the next state x_t, otherwise the feasibility of x_t is questionable. In other words, the distribution of x_t should be given as $\Pr(x_t \mid \bigwedge_{\tau=0:t-1} x_\tau \in \mathcal{X}_F)$, which means the distribution of x_t should be strictly conditional on its previous states being collision-free.

To properly address the state dependency within the state propagation process, two notations are introduced first, i.e., $x_{t|k}$ and $\tilde{x}_{t|k}$. The state $x_{t|k}$ represents the estimate of x_t conditional on the state x_τ being collision-free for all stages $\tau = 0 : k$ and $\tilde{x}_{t|k}$ denotes the distribution of $x_{t|k}$ as,

$$x_{t|k} \sim \tilde{x}_{t|k} = \Pr(x_t \mid \bigwedge_{\tau=0:k} x_\tau \in \mathcal{X}_F). \tag{3.13}$$

Given the definition of $x_{t|k}$, the distribution of x_t thereby can be reformulated as $\tilde{x}_{t|t-1} = \Pr(x_t \mid \bigwedge_{\tau=0:t-1} x_\tau \in \mathcal{X}_F)$, and $x_{t|t-1}$ is not ensured to be collision-free given this distribution. As a consequence, the state dependency will break down if $x_{t|t-1}$ is applied directly to propagate the state x_{t+1}. In this study, we seek to extract the collision-free part of $\tilde{x}_{t|t-1}$

and model it as a new distribution $\tilde{x}_{t|t}$, such that state x_{t+1} can be propagated from $\tilde{x}_{t|t}$ instead.

Let's consider the case in which $\tilde{x}_{t|t-1}$ is a Gaussian $\mathcal{N}(\bar{x}_{t|t-1}, \Sigma_{t|t-1})$, then its collision-free part w.r.t. a convex polyhedral obstacle can be approximated as a Gaussian as well by following the method proposed in [112]. As illustrated in Figure 3.1, this can be achieved by truncating the distribution $\mathcal{N}(\bar{x}_{t|t-1}, \Sigma_{t|t-1})$ against the linear constraints defined by the obstacle, resulting in a shift of mean and covariance by Δx_t and $\Delta\Sigma_t$,

$$\begin{aligned} \bar{x}_{t|t} &= \bar{x}_{t|t-1} - \Delta x_t, \\ \Sigma_{t|t} &= \Sigma_{t|t-1} - \Delta\Sigma_t, \end{aligned} \tag{3.14}$$

where $\bar{x}_{t|t}$ and $\Sigma_{t|t}$ represent the mean and covariance of $\tilde{x}_{t|t}$.

Given a linear constraint $\mathbf{a}^{\mathrm{T}} x_{t|t-1} \leq \mathbf{b}$, the shift Δx_t and Σ_t can be given as,

$$\begin{aligned} \Delta x_t &= -\frac{\lambda}{\sqrt{\mathbf{a}^{\mathrm{T}}\Sigma_{t|t-1}\mathbf{a}}}(\Sigma_{t|t-1}\mathbf{a}), \\ \Delta\Sigma_t &= \frac{\lambda^2 - \beta\lambda}{\mathbf{a}^{\mathrm{T}}\Sigma_{\mathbf{t|t-1}}\mathbf{a}}(\Sigma_{t|t-1}\mathbf{a})(\Sigma_{\mathbf{t|t-1}}\mathbf{a})^{\mathrm{T}}, \end{aligned} \tag{3.15}$$

where $\beta = \frac{\mathbf{b}^{\mathrm{T}}\hat{x}_{t|t-1}}{\sqrt{\mathbf{a}^{\mathrm{T}}\Sigma_{\mathbf{t|t-1}}\mathbf{a}}}$ denotes the degree of truncation, and $\lambda = \frac{\mathbf{pdf}(\beta)}{1-\mathbf{cdf}(\beta)}$ is the ratio of the standard Gaussian probability distribution function **pdf** and the cumulative distribution function **cdf** evaluated at β. Because of the approximation made in Equation (3.7), extending this strategy to the obstacle defined by L linear constraints is straightforward, where the truncation is only applied to the linear constraint $[\mathbf{a}^*, \mathbf{b}^*]$ that returns the minimum risk,

$$[\mathbf{a}^*, \mathbf{b}^*] = \arg\min_{[\mathbf{a}_i, \mathbf{b}_i]} \Pr(\mathbf{a}_i x_{t|t-1} < \mathbf{b}_i), i \in \{1, \ldots, L\}. \tag{3.16}$$

Furthermore, let's consider the multiple obstacle case, and let $[\Delta x_t^{[k]}, \Sigma_t^{[k]}]$ be the shift pair for obstacle $\mathcal{X}_k, k \in \{1, ..., K\}$, then the truncation w.r.t. all the obstacles is given as the cumulative shift,

$$\Delta x_t = \Sigma_{k=0}^{[K]} \Delta x_t^{[k]}, \; \Delta\Sigma_t = \Sigma_{j=0}^{[K]} \Delta\Sigma_t^{[k]}. \tag{3.17}$$

Algorithm 1: Conditional State Propagation

input : $\tilde{x}_{t-1|t-1} = \mathcal{N}(\bar{x}_{t-1|t-1}, \Sigma_{t-1|t-1})$
output : $\tilde{x}_{t|t}$

$\bar{\tilde{x}}_{t-1|t-1} = \bar{x}_{t-1|t-1} - x^*_{t-1}$
$\bar{\tilde{x}}_{t|t-1} = A_t \bar{\tilde{x}}_{t-1|t-1} + B_t(\hat{u}_t)$
$\Sigma_{t|t-1} = A_t \Sigma_{t-1|t-1} A_t^T + V_t M_t V_t^T$
$\bar{x}_{t|t-1} = \bar{\tilde{x}}_{t|t-1} + x^*_t$
$\tilde{x}_{t|t-1} = \mathcal{N}(\bar{x}_{t|t-1}, \Sigma_{t|t-1})$
if `DegreeTrunc`$(\tilde{x}_{t|t-1}) > \omega$ **then**
    $[\Delta x_t, \Delta\Sigma_t] \leftarrow$ `Truncation`$(\tilde{x}_{t|t-1})$
    $\bar{x}_{t|t} = \bar{x}_{t|t-1} - \Delta x_t$
    $\Sigma_{t|t} = \Sigma_{t|t-1} - \Delta\Sigma_t$
    return $\tilde{x}_{t|t} = \mathcal{N}(\bar{x}_{t|t}, \Sigma_{t|t})$
else
    return $\tilde{x}_{t|t} = \mathcal{N}(\bar{x}_{t|t-1}, \Sigma_{t|t-1})$
end

Given the above analysis, the conditional state propagation is summarized as Algorithm 1. At each stage t of conditional state propagation, the distribution $\tilde{x}_{t|t-1}$ is truncated if the degree of truncation returned by function `DegreeTrunc` is over the desired threshold ω. After that, the truncated distribution $\mathcal{N}(\hat{x}_{t|t}, \Sigma_{t|t})$ is utilized for the propagation of the next stage $t+1$. Because of the mean shift Δx_t, using $\tilde{x}_{t|t}$ to propagate the next stage will introduce a certain deviation from the nominal trajectory (red dashed curve in Figure 3.1), which is assumed to be negligible when the interval between stage t and stage $t+1$ is small enough. Consequently, the overall collision risk of a trajectory ξ consisting of states $\{x_0, x_1, ..., x_T\}$ can be given as,

$$\Pr(\xi \in \mathcal{X}_F) = \prod_{t=0}^{T} \Pr(x_{t|t-1} \in \mathcal{X}_F), \tag{3.18}$$

which is less conservative than the approximation $\Pr(\xi \in \mathcal{X}_F) \approx \prod_{t=0}^{T} \Pr(x_t \in \mathcal{X}_F)$ made by most of the existing approaches.

3.3.3 CC-RRT*-D Algorithm

After the discussion of the chance constraint approximation and the conditional state propagation, the proposed CC-RRT*-D algorithm will be detailed in this section. In short, the

Algorithm 2: CC-RRT*-D Tree Expansion

input : x_{init}
output: $T = (V, E)$

$T \leftarrow$ `InitializeTree()`
$T \leftarrow$ `InsertNode`$(\emptyset, x_{init}, T)$
for $i \leftarrow 1$ **to** N **do**
  $x_{rand} \leftarrow$ `Sample()`
  $x_{nearest} \leftarrow$ `Nearest`(T, x_{rand})
  $(x_{new}, u_{new}, \pi_{new}) \leftarrow$ `Steer`$(x_{nearest}, x_{rand})$
  if `RiskFeasible`$(x_{nearest}, u_{new}, x_{new})$ **then**
    $X_{near} \leftarrow$ `Near`(T, x_{new}, η)
    $x_{min} \leftarrow$ `Parent`(X_{near}, x_{new})
    $T \leftarrow$ `InsertNode`(x_{min}, x_{new}, π)
    $T \leftarrow$ `Rewire`$(T, X_{near}, x_{min}, x_{new})$
  end
end
return $T = (V, E)$

RRT* algorithm is employed for space exploration and trajectory searching, and the conditional state propagation is used to derive in advance the state distributions along the RRT* tree. The chance constraint approximation, thereafter, will be used to evaluate the collision risk of the RRT* tree, and a probabilistically feasible tree can be expanded incrementally over the entire space. The problem stated in Equation (3.3) can thereby be properly addressed.

The CC-RRT*-D tree expansion algorithm is detailed in Algorithm 2. In each iteration, the `Sample` function generates independent, identically distributed samples for space exploration. Then the nearest node in terms of a certain distance metric is identified, and the `Steer` function is applied to compute a trajectory segment ξ_{new} connecting the nearest node with an intermediate state x_{new}. The function `RiskFeasible` in Algorithm 3 acts as an alternative to the collision checker, which employs Algorithm 1 to propagate the states' distribution along the nominal path, and evaluates the risk of collision using Equation (3.8), which will return true if the given trajectory is probabilistically safe. If x_{new} and π_{new} are probabilistically feasible, a near vertex set X_{new} will be returned by the function

$Near(T, x, \eta)$ as,

$$X_{near} = \{x' \mid ||x' - x|| \leq \min\{\varsigma(log(n)/n)^{1/d}, \varrho\}, \tag{3.19}$$

where d is the dimension of the state, n is the number of vertices in the tree, ϱ is a pre-defined maximum ball radius and ς is a constant [35].

For all the near vertices $x \in X_{near}$ that have a probabilistically feasible connection with x_{new}, the best one w.r.t. the objective function, i.e., the sum of current cost $J(x)$ and cost-to-go $C(x, x_{new})$, will be set as the parent of x_{new} (see Algorithm 4) and inserted into the tree. Then the `Rewire` function in Algorithm 5 recalculates the best parents for all vertices within X_{near}. Once the rewiring procedure is necessary, the tree will be reconnected. Then the rewired vertex together with its children's distribution are re-propagated in order to maintain the explicit dependency between the vertices.

Finally, Algorithm 2 builds the probabilistically feasible tree and the solution trajectory can be asymptomatically optimized by the RRT* algorithm.

3.3.4 Optimality Analysis

The proposed CC-RRT*-D algorithm is built upon the RRT* algorithm, so asymptotic optimality can be naturally guaranteed by following the principles of the RRT* algorithm. For the sake of proving the optimality of Algorithm 2, a necessary assumption has been made as,

Assumption 3.3.1. *There exists a ball $\mathcal{X}^{\varepsilon}$ of radius ε at every point $x \in \mathcal{X}$ such that for points $x' \in \mathcal{X}^{\varepsilon}$, $\int_{\mathcal{X}_F} \Pr(x')dx' > 1 - \delta$.*

This assumption states that it is possible to move the mean of the distribution within some ball but not violate the chance constraints, which thereby gives the graph a finite sample volume to converge in. Based on this assumption, the optimality proof can follow Theorem 38 in [35].

In addition to the assumption made, the cost function C defined in Algorithm 2 also needs to stay monotonic and bounded in order to guarantee the optimality [35]. A risk-

Algorithm 3: $RiskFeasible(x_{start}, u, x_{end})$

input : x_{start}, u, x_{end}
output : $True/False$

$\mathcal{N}(\bar{x}_{0|0}, \Sigma_{0|0}) \leftarrow x_{start}$
for $t \leftarrow 1$ **to** k **do**
  $\tilde{x}_{t|t-1} \leftarrow \texttt{Propagate}(\mathcal{N}(\bar{x}_{t-1|t-1}, \Sigma_{t-1|t-1}), u)$
  if $\Pr(x_{t|t-1} \notin \mathcal{X}_F) > \delta$ **then**
    return $False$
  end
  $\mathcal{N}(\bar{x}_{t|t}, \Sigma_{t|t}) \leftarrow \texttt{Truncate}(\tilde{x}_{t|t-1})$
end
$x_{end} \leftarrow \mathcal{N}(\bar{x}_{k|k}, \Sigma_{k|k})$
return $True$

Algorithm 4: $Parent(X_{near}, x_{new})$

input : X_{near}, x_{new}
output : x_{min}, J_{min}

for $x \in X_{near}$ **do**
  $(x', u', \xi') \leftarrow \texttt{Steer}(x, x_{new})$
  if $\texttt{RiskFeasible}(x', u', x_{new})$ **then**
    $X_{steer} \leftarrow \texttt{InsertVertex}(x)$
  end
end
$J_{min} = \min_{x \in X_{steer}}(J(x) + C(x, x_{new}))$
$x_{min} = \arg\min_{x \in X_{steer}}(J(x) + \mathcal{C}(x, x_{new}))$

Algorithm 5: $Rewire(T, X_{near}, x_{min}, x_{new})$

input : $T, X_{near}, x_{min}, x_{new}$
output : T

for $x \in X_{near} \backslash x_{min}$ **do**
  $(x', u', \pi') \leftarrow \texttt{Steer}(x_{new}, x)$
  if $\texttt{RiskFeasible}(x_{new}, u', x')$ **and** $J(x_{new}) + C(x_{new}, x) < J(x)$ **then**
    $T \leftarrow \texttt{Reconnect}(x_{new}, x, \pi)$
    $\texttt{RePropagate}(x)$
  end
end

based cost function for trajectory ξ thereby is defined as,

$$C(\xi) = Length(\xi) + \kappa \max_{t\in\{0,...,T\}} r(\xi(t)), \tag{3.20}$$

where $r(\xi(t)), t \in \{0, ..., T\}$ denotes the collision risk of trajectory state $\xi(t)$, and κ is a scaling factor to balance the trajectory length $Length(\xi)$ and the maximum risk over the trajectory $\max_{t\in\{0,...,T\}} r(\xi(t))$. The trajectory length factor aims at improving the driving efficiency, and the inclusion of maximum risk over the trajectory is to evaluate the trajectory's risky level. Because both $Length(\xi)$ and $\max_{t\in\{0,...,T\}} r(\xi(t))$ are monotonic and bounded, this cost function can be easily verified to be able to meet the optimality requirement.

3.4 Experiment

In this section, we will present the computational experiments to evaluate the performance of the CC-RRT*-D algorithm, and demonstrate a real experiment on an autonomous vehicle for real-time obstacle avoidance.

3.4.1 Computational Experiment

A. Settings

The computational experiments were implemented on a quad-core 1.6 GHz processor with 4 GB of RAM, and the algorithms were programmed in C++.

The simulation was conducted in a constrained, two-dimensional 10 m × 15 m environment, which contains two obstacles as in Figure 3.2. The starting location is set as the bottom-left corner and goal region is located at the top-right corner. Regarding the stochastic system model, we started from a 2D single integrator dynamics as,

$$x_t = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x_{t-1} + \begin{bmatrix} \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} [v_x + \Delta v_x, v_y + \Delta v_y]^{\mathrm{T}}, \tag{3.21}$$

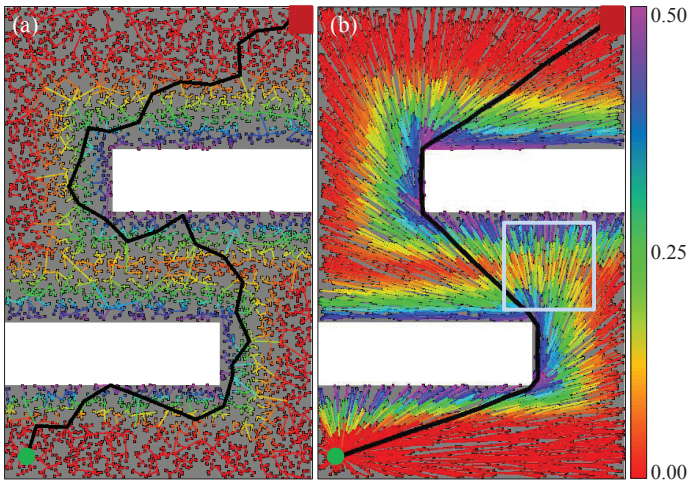


Figure 3.2 Planning results without taking account motion disturbance after 5,000 vertices trials: (a) shows the planning result of RRT and (b) depicts the planning result of the RRT*, where the solution trajectory is given as the black line. The trajectories' collision risks, ranging from 0.0 to 0.5, are demonstrated by the color scales, and the region highlighted by the light blue rectangle in (b) is associated with relatively higher collision risk.

where $\Delta t = 0.1\ s$ is the duration of each time step and the speed limit is set as $|v_x| \leq 10\ m/s, |v_y| \leq 10\ m/s$. The initial state x_0 is subjected to a Gaussian localization error: $x_0 \sim \mathcal{N}(\bar{x}_0, \Sigma_0)$, $\Sigma_0 = \left[\begin{smallmatrix} 0.5 & 0 \\ 0 & 0.5 \end{smallmatrix}\right]$, and the motion disturbance is imposed on the input velocity as $[\Delta v_x, \Delta v_y]^{\mathrm{T}} \sim \mathcal{N}\left(0, \left[\begin{smallmatrix} 0.2|v_x| & 0 \\ 0 & 0.2|v_y| \end{smallmatrix}\right]\right)$. The collision risk bound δ is set as 0.05 and the degree of truncation threshold is set as 0.005.

B. Results

Figure 3.2 represents the planning results using both RRT and RRT* after 5,000 vertices trials, where the motion uncertainty is not considered. The trajectory length is defined as the objective function, and the solution trajectory is represented by the black line, from which we can find that RRT, unlike RRT*, lacks the ability to ensure the trajectory optimality. The trajectories' collision risks, ranging from 0.0 to 0.5, are demonstrated by the color scales. Obviously, the collision risk is increasing when the obstacles are approached, and the region highlighted by the light blue rectangle in Figure 3.2(b) is associated with relatively higher collision risk, which will result in a bottleneck when motion disturbance is imposed.

Thereafter, we compared the planning results of our proposed algorithm (CC-RRT*-D) with a previous related approach CC-RRT* [69], in which state dependency is not properly

considered. To further investigate the impact of state dependency on planning performance, another algorithm called CC-RRT*-R was included for comparison as well. CC-RRT*-R shares the same framework with Algorithm 2 but uses Equation (3.12) for state propagation instead. More specifically, the comparison between CC-RRT* and CC-RRT*-R is to show the necessity of state re-propagation when the rewiring procedure occurs. By comparing CC-RRT*-R and CC-RRT*-D, we can observe the improvements obtained in solving the over-conservative issue of chance constraint approximation. Without the loss of generality, trajectory length is defined as the cost function. 2,000 vertices were computed each time, and each algorithm was tested 20 times. The probabilistically feasible tree for each algorithm is then demonstrated in Figure 3.3.

Firstly, we want to explore the necessity of state re-propagation when the RRT* rewiring procedure occurs, and we start from analyzing how the vertices are expanded in the region between the two white obstacles (shown in Figure 3.3). Compared to CC-RRT*-R, we found that CC-RRT* needs much more effort to pass through this constrained region, and a large number of vertices end up being expanded in the lower part of the environment by CC-RRT*. As such, the solution trajectory found by CC-RRT* is longer and more conservative. This can be further verified by the planning results in Table 3.1, where CC-RRT* needs around 465 vertices to find a feasible solution but CC-RRT*-R only takes about 372 vertices. The underlying reason for this difference is that each state's distribution is a function of the motion disturbance that is imposed on the state propagation. For this particular case, the shorter trajectory requires less control effort, therefore less motion disturbance is imposed on the state propagation. When the state's distance to the initial state is converging, its distribution variance should also converge. However, CC-RRT* lacks the functionality to update the state distribution when the rewiring procedure is performed, and the convergence of the state distribution therefore cannot be guaranteed. On the other hand, the re-propagation process in CC-RRT*-R can ensure this convergence, such that CC-RRT*-R can handle the bottleneck more efficiently.

Now let us move to the comparison between CC-RRT*-R (see Figure 3.3(b)) and CC-RRT*-D (see Figure 3.3(c)). Obviously, the CC-RRT*-R tree is more conservative than

Table 3.1 Number of Vertices Needed to Find Feasible Trajectory (20 Trials)

	CC-RRT*	CC-RRT*-R	CC-RRT*-D
Average	465.5	372.8	388.0
Stdev.	229.0	174.1	166.65
Min.	262	186	190
Max.	958	693	690

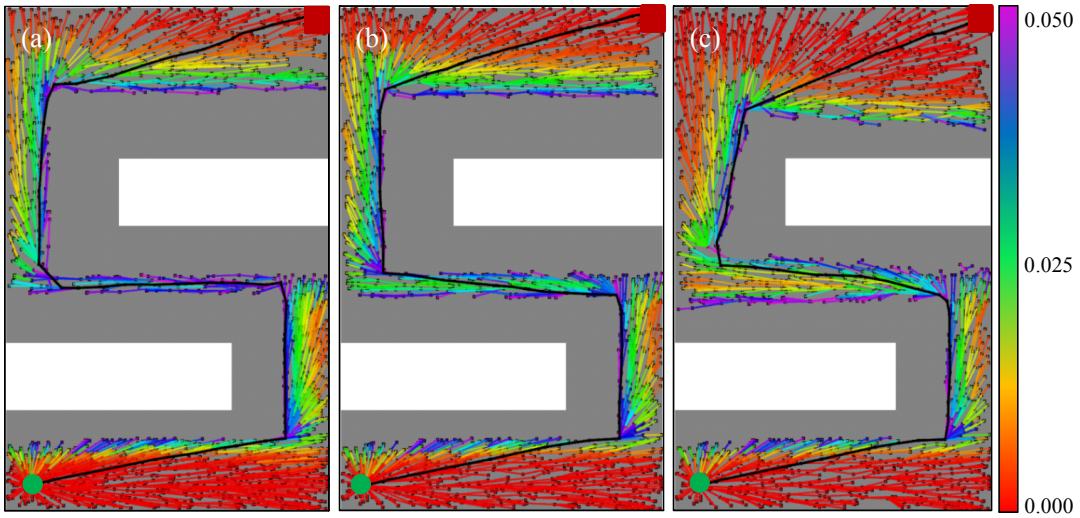


Figure 3.3 Probabilistically feasible trees of three algorithms using length-based cost function over 2,000 vertices trials: (a) CC-RRT*, (b) CC-RRT*-R and (c) CC-RRT*-D.

that of CC-RRT*-D, and the solution trajectory returned by CC-RRT*-D is shorter. This is consistent with our previous discussion in Section 3.3.2, where the conditional state propagation is able to alleviate the over-conservative issue of chance constraint approximation.

In order to provide a more intuitive comparison of the planning performance, Figure 3.4 charts the solution trajectory's length versus the number of vertices. The median over 20 trials for each algorithm is plotted, and one can easily see that the solution trajectories are asymptotically optimized. As expected, CC-RRT*-D returned the shortest trajectory, and the solution trajectory returned by CC-RRT*-R is relatively shorter than that of CC-RRT*. Moreover, the computation time versus the number of vertices is charted in Figure 3.5, which is to evaluate the planning efficiency. In the early stage, CC-RRT* spent much more time on finding feasible vertices to travel through the bottleneck region. After that, the CC-RRT* tree can be quickly expanded after a feasible trajectory is found. Overall, CC-RRT* is more computationally efficient than CC-RRT*-R, which is reasonable because CC-RRT*-R requires additional computation for state re-propagation after the CC-RRT*-R tree is rewired. While additional computation is also required by CC-RRT*-D for state re-propagation and distribution truncation, the less conservative property makes CC-RRT*-D take the least time to expand the tree.

To validate the risk-based cost function discussed in Section 3.3.4, Figure 3.6 demonstrates the planning results using the risk-based cost function with the scaling factor $\kappa = 10$. The resulting probabilistically feasible trees present significant qualitative difference from those in Figure 3.3. The trajectories in Figure 3.6 tend to travel toward the obstacle surfaces, such that the duration spent by traveling in the high-risk regions can be minimized. The solution trajectory can thereby properly trade off between trajectory length and collision risk. In Figure 3.3, however, the trajectories passing around obstacles tend to be parallel with the obstacle surfaces, which aims at minimizing the trajectory length instead.

Thereafter, we extended the previous evaluations to a system with high motion disturbance, where the motion disturbance is increased to $[\Delta v_x, \Delta v_y]^T \sim \mathcal{N}(0, \begin{bmatrix} 0.4|v_x| & 0 \\ 0 & 0.4|v_y| \end{bmatrix}$. Sharing the same settings, CC-RRT*-R and CC-RRT*-D were evaluated over 5,000 vertices and the resulting probabilistically feasible trees are demonstrated in Figure 3.7. Due

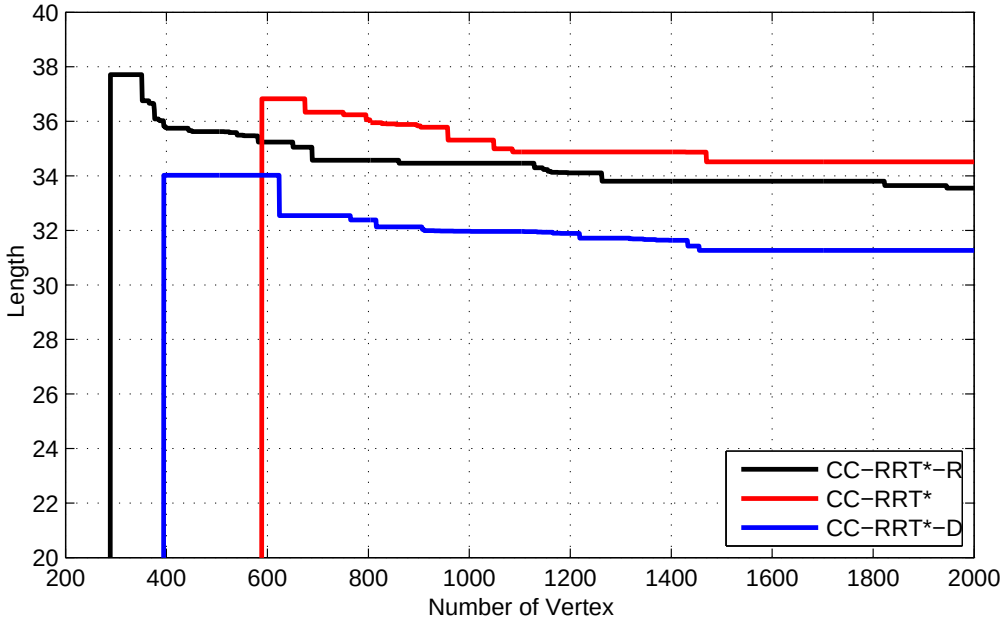


Figure 3.4 Evolution of solution trajectory's length over 2,000 vertices trials.

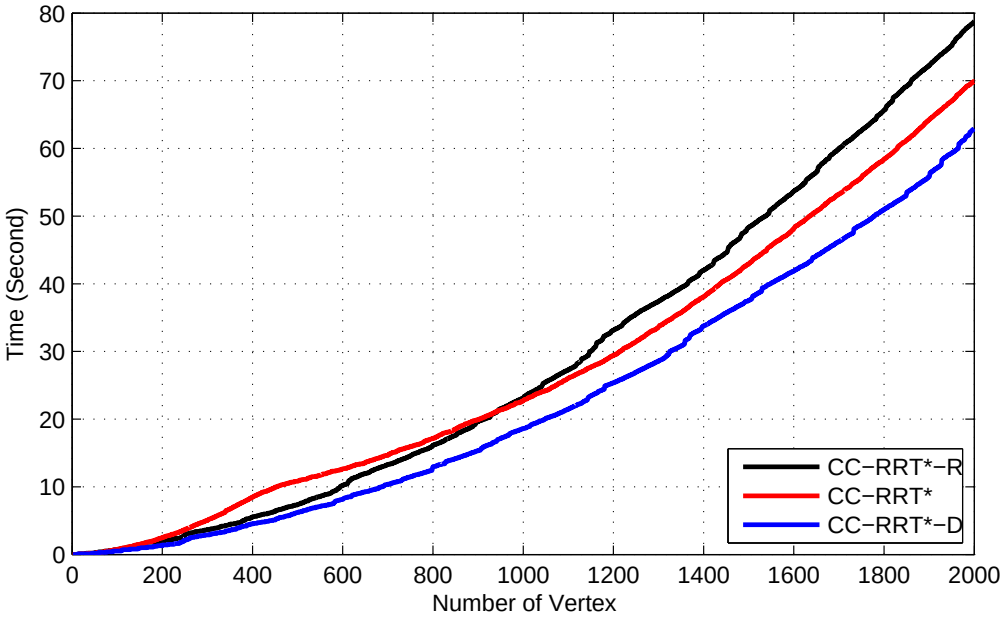


Figure 3.5 Computation time versus the number of vertex over 2,000 vertices trials.

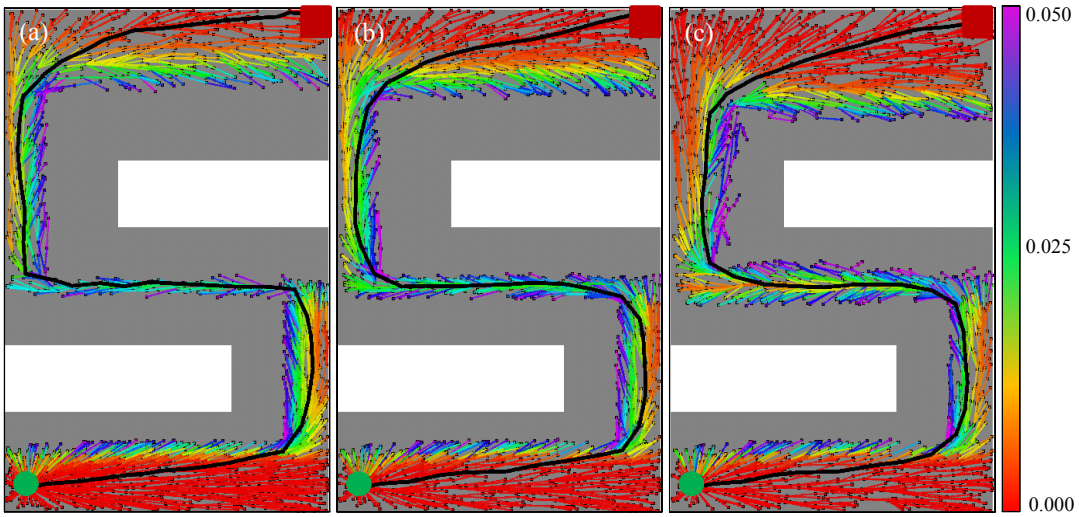


Figure 3.6 Probabilistically feasible trees of three algorithms using risk-based cost function over 2,000 vertex trials: (a) CC-RRT*, (b) CC-RRT*-R and (c) CC-RRT*-D.

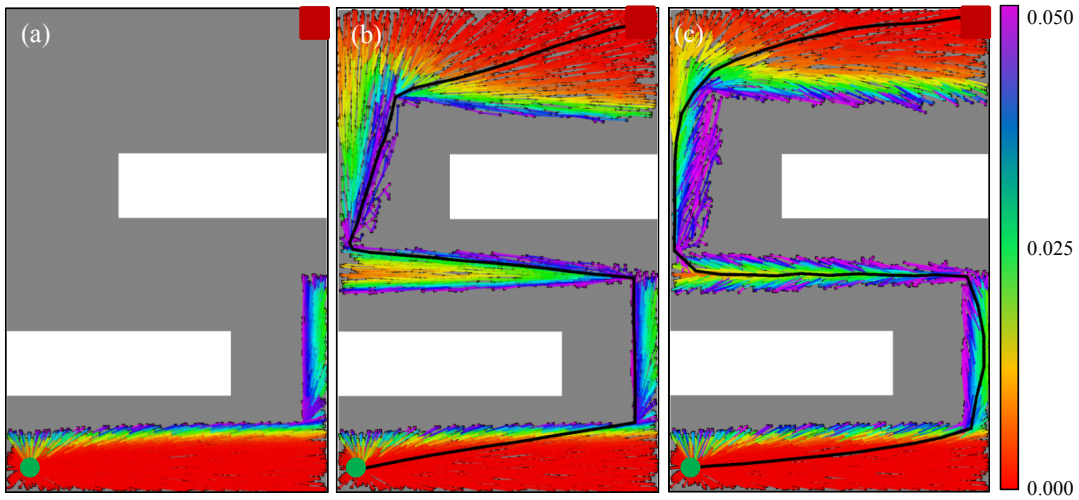


Figure 3.7 Probabilistically feasible trees of CC-RRT*-R and CC-RRT*-D under high motion disturbance: a) CC-RRT*-R failed to find a solution trajectory, b) CC-RRT*-D using trajectory length as objective function, and c) CC-RRT*-D using risk-based objective function.

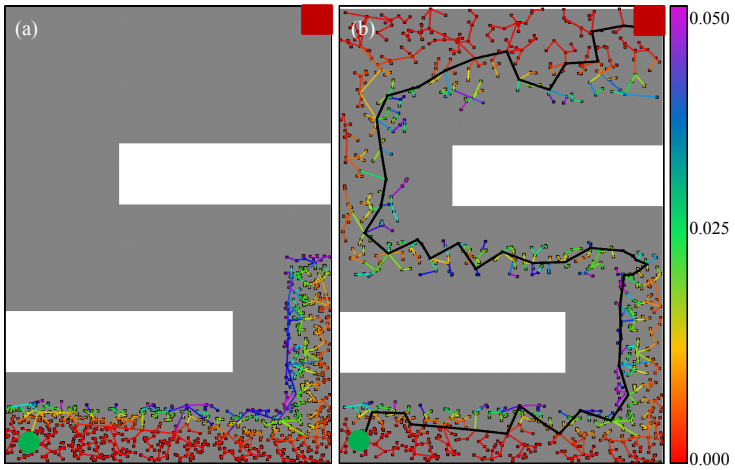


Figure 3.8 Conditional state propagation extended to CC-RRT: a) CC-RRT without using conditional state propagation and b) CC-RRT using conditional state propagation.

to the over-conservativeness of chance constraint approximation, CC-RRT*-R failed to return a solution trajectory, as shown in Figure 3.7(a). On the other hand, solution trajectories can be efficiently found by CC-RRT*-D after about 1,500 vertices trials using two different objective functions, as shown Figure 3.7(b) and Figure 3.7(c) respectively.

Moreover, the extension of the conditional state propagation for CC-RRT [65] was also implemented. Figure 3.8 represents the comparison between the CC-RRT tree with and without using conditional state propagation after 1,000 vertices trials. In contrast to CC-RRT in Figure 3.8(a) that fails to find a solution, CC-RRT using conditional state propagation can efficiently handle the constrained region in Figure 3.8(b).

Last but not least, some addition statistical tests on a broader range of obstacle fields were performed. Given the same environment size and planning settings (e.g., stochastic system model, staring and goal positions) as in Section 3.4.1, we randomly generated 500 polyhedral obstacle fields using 5 different obstacle densities[1], where 100 obstacle fields were generated for each obstacle density. Thereafter, three algorithms, i.e., CC-RRT*, CC-RRT*-R and CC-RRT*-D, were evaluated for each obstacle field. To evaluate the achievements we have made to resolve the over-conservative issue, two metrics are computed and presented in Figure 3.9 and Figure 3.10. As shown in Figure 3.9, the first metric we measured is the success rate of finding a feasible trajectory after the 2000-vertices trial. As the

[1] Obstacle_Density = Area_of_Obstacle_Field / Area_of_Environment

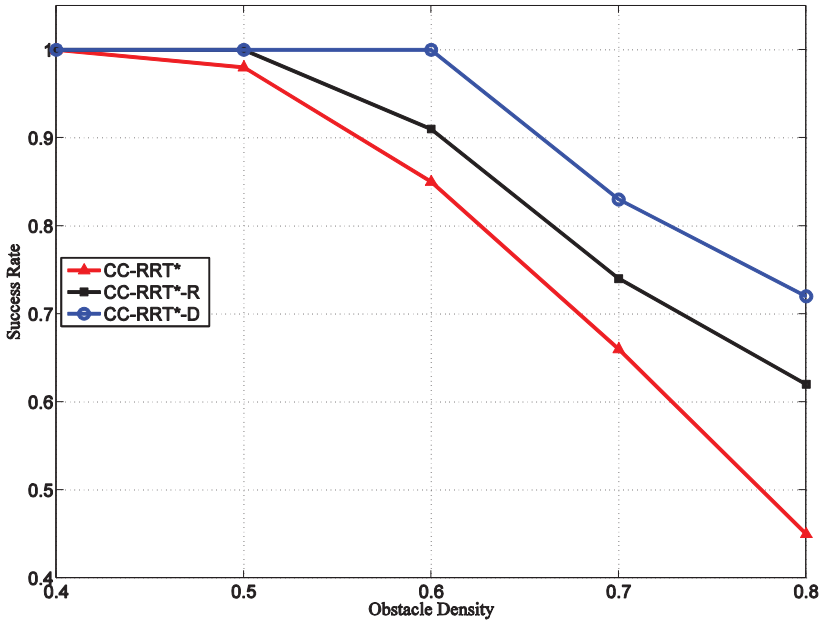


Figure 3.9 Success rate of finding a feasible trajectory when planning on the randomly generated polyhedral obstacle fields.

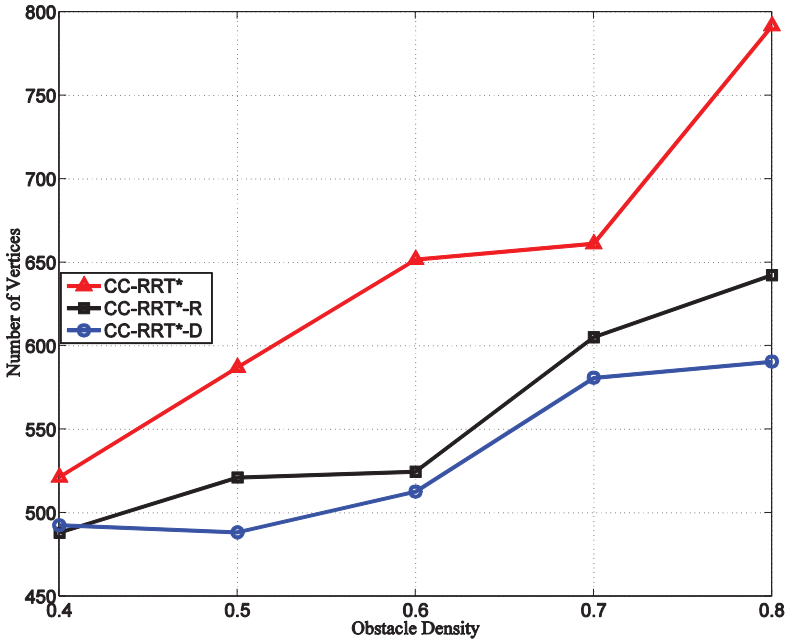


Figure 3.10 Average number of vertices needed to find a feasible trajectory when planning on the randomly generated polyhedral obstacle fields.

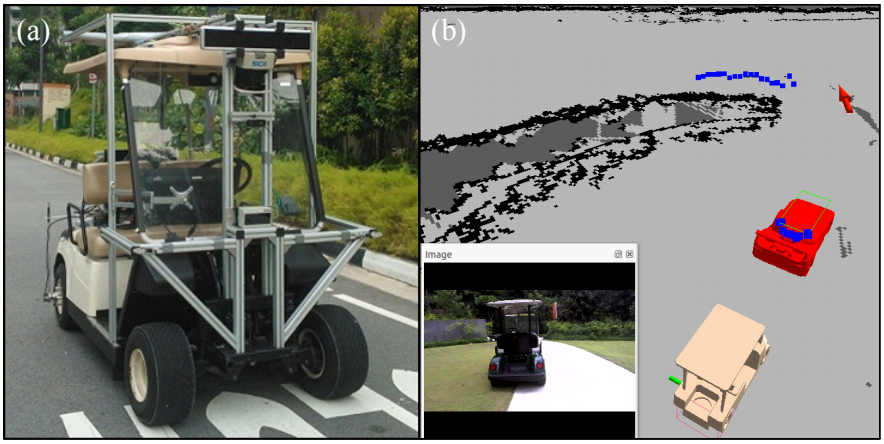


Figure 3.11 Real-time experiment setup for CC-RRT*-D evaluation: a) Autonomous vehicle mounted with various sensors; b) Experiment scenario: the autonomous vehicle needs to avoid the front vehicle and navigate to the destination that marked with a red arrow.

environment getting denser and more cluttered, the success rate is gradually dropping for all these three algorithms. As we expected, CC-RRT*-D has a higher success rate than that of CC-RRT* and CC-RRT*-R. Besides the success rate, the second metric, i.e., the average number of vertices needed to find a feasible trajectory, is shown in Figure 3.10. In line with our earlier discussion, more vertices is required by CC-RRT* to search a probabilistically feasible trajectory, but CC-RRT*-D is able to efficiently handle those constrained regions.

To summarize, the presented computational experiments have verified the necessity of re-propagation when the tree rewiring occurs, and the introduction of conditional state propagation can improve the over-conservative result in chance constraint approximation.

3.4.2 Real-time Experiment for Obstacle Avoidance

Besides the extensive computational experiments, we also implemented real-time obstacle avoidance using the proposed CC-RRT*-D on an autonomous vehicle (see Figure 3.11(a)).

A. Settings

Regarding the system model, the vehicle state $x = [\mathrm{x}, \mathrm{y}, \theta]^{\mathrm{T}}$ is composed of its position $[\mathrm{x}, \mathrm{y}]^{\mathrm{T}}$, and orientation θ. The control input $u = [\mathrm{v}, \phi]^{\mathrm{T}}$, consisting of its speed v and steering angle ϕ, is corrupted by the motion disturbance $m = [\Delta\mathrm{v}, \Delta\phi]^{\mathrm{T}} \sim \mathcal{N}(0, \mathrm{M})$. Due

to the innate vehicle kinematic constraint, the stochastic system model is given as,

$$f(x,u,m) = \begin{bmatrix} \mathrm{x} + (\mathrm{v} + \Delta\mathrm{v})cos\theta\Delta t \\ \mathrm{y} + (\mathrm{v} + \Delta\mathrm{v})sin\theta\Delta t \\ \theta + (\mathrm{v} + \Delta\mathrm{v})tan(\phi + \Delta\phi)\Delta t/l \end{bmatrix}, \tag{3.22}$$

where Δt is the duration of each time step and l is the distance between front and rear axle. The speed limit is set to $|\mathrm{v}| \leq 1.0\ \mathrm{m/s}$ and the motion disturbance is modeled as $m \sim \mathcal{N}\left(0, \left[\begin{smallmatrix} 0.1m/s & 0 \\ 0 & 0.1rad \end{smallmatrix}\right]\right)$. Also, the collision risk bound δ is set to 0.1, and the objective function is defined as the trajectory length. The experimental scenario is depicted in Figure 3.11(b), where the autonomous vehicle received a task of navigating to the parking yard exit (red arrow in the top-right of Figure 3.11(b)), meanwhile gracefully avoiding the front vehicle occupying the road.

B. Result

The vehicle was localized first using the Monte-Carlo Localization algorithm [113], and the initial state distribution was approximated as a Gaussian by clustering the particles. Then the obstacle detection and classification module was called to perceive the front vehicle. Based on the obstacle classification result, the front vehicle was modeled as a rectangle consisting of four linear constraints, which can be visualized in Figure 3.13(a).

After the planning trial consisting of 86 vertices, the CC-RRT*-D tree and the solution trajectory is given in Figure 3.13(a). As expected, the tree is bounded within a probabilistically feasible region imposed by the chance constraint. The solution trajectory was then executed as in Figure 3.13 (b)-(d), where the red curve indicates the autonomous vehicle's driving trace. Although the solution trajectory cannot be tracked closely due to the motion disturbance and localization error, the autonomous vehicle can still safely and smoothly avoid the front obstacle.

To further evaluate CC-RRT*-D's efficiency for real-time motion planning, this obstacle avoiding experiment was then conducted another 35 times using seven different settings

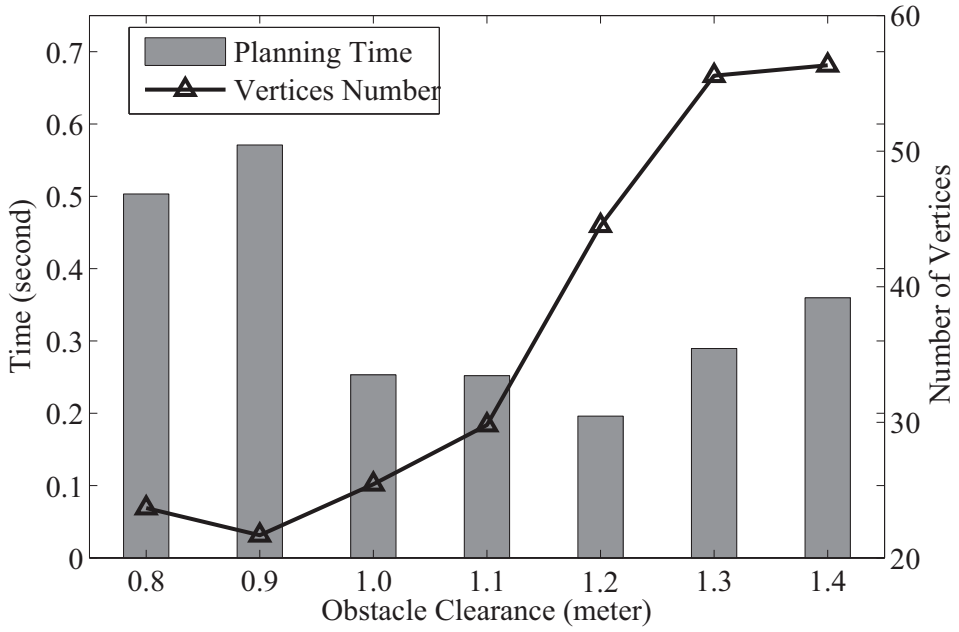


Figure 3.12 Evaluation of CC-RRT*-D's efficiency for real-time motion planning: the averaged planning time and the number of vertices used to find a feasible trajectory are evaluated w.r.t. the obstacle clearance.

of obstacle clearance (i.e., the distance to the front vehicle), where five experimental trials were conducted for each setting of obstacle clearance.

The averaged planning time (i.e., time consumed to plan a feasible trajectory) w.r.t. the obstacle clearance is depicted in Figure 3.12. As the obstacle clearance was increased, the planning time decreased first and grew a bit afterward. This is because the CC-RRT*-D algorithm experiences less difficulty in searching a feasible trajectory when the planning space is less constrained. However, when the planning space turned out to be quite free, more planning time are needed to explore the entire space and optimize the solution trajectory. This is evidenced by the number of vertices used to find a feasible trajectory, where the number of CC-RRT*-D vertices increased gradually when the obstacle clearance is increased.

Given these experimental results, we can find that the proposed CC-RRT*-D algorithm is safe and efficient enough for real-time motion planning.

3.5 Summary

This chapter introduced the CC-RRT*-D algorithm for risk-aware planning under motion uncertainty. The algorithm leveraged the asymptotic optimal property of the RRT* framework and employed the chance constraint approximation to compute risk-aware and asymptotically optimal trajectories. State dependency was explicitly considered by the proposed algorithm, whose necessity had been extensively verified by the experiment results. The employment of conditional state propagation can help to resolve the over-conservative issue of many existing works. The consistent experiment results showed that the CC-RRT*-D algorithm can address the internal motion uncertainty in a safe and efficient manner.

The internal motion uncertainty has been addressed in this chapter, and the external situation uncertainty will be discussed in the following two chapters.

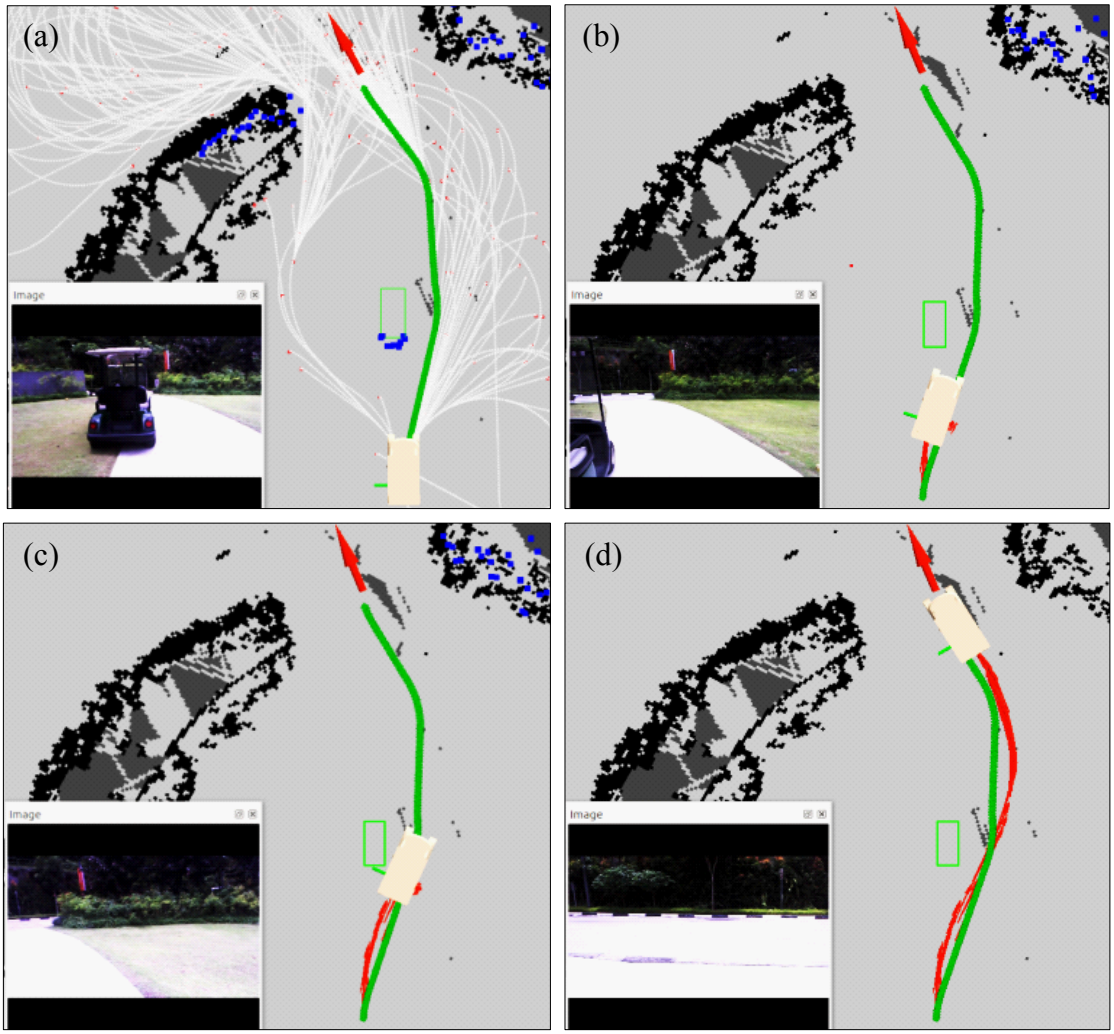


Figure 3.13 Real-time experiment on an autonomous vehicle for obstacle avoidance. The planning result is shown in (a) after 86 vertices trials, where the solution trajectory is indicated as the green curve. The execution process is in (b)-(d), where the red curve is the autonomous vehicle's driving trace.

Chapter 4

Road Context Inference and Applications for Vehicle Behavior Analysis

4.1 Introduction

As the autonomous vehicles venture out onto the urban roads, the recognition of the road context is always essential, because safe driving requires a comprehensive understanding of the environment. The road context can be employed to not only regularize the autonomous vehicles' own behavior, but also narrow down the reasoning scope of other vehicles' driving behaviors [87].

Acknowledging the correlation between the vehicle behavior and the road context, a probabilistic approach for road context inference is proposed in this chapter. Instead of purely focusing on spatial analysis [84] or motion pattern learning [81], the vehicle behaviors are systematically analyzed from different aspects in the proposed approach, and the road context is inferred in a consistent and integrated manner. More specifically, the proposed approach aims at extracting the consistencies within the observed vehicle behaviors to model the road context, which consists of three stages: *Topology Learning*, *Motion Learning* and *Rule Learning*. Given the randomly-collected vehicle behavior data, the

topology learning is to analyze the spatial distributions of the vehicle behavior data, and aims at extracting the topological connectivity of the traversable space, i.e., road topology. Thereafter, the motion learning is to generalize the typical vehicle motion pattern by analyzing the vehicle dynamics. Lastly, the road topology and the generalized motion pattern are integrated to achieve a probabilistic representation of the traffic rules. After the road context inference, the resulting road context is then extended for the applications of vehicle behavior analysis, including the abnormal driving behavior detection and the long-term motion prediction. The proposed approach has been validated through a case study of an urban road that includes round-about and T-junction, and some promising results have been achieved.

This chapter is organized as follows. Section 4.2 provides the analysis of the correlation between vehicle behavior and road context. The overview of the road context learning system is presented in Section 4.3, after which the proposed approach for probabilistic road context inference is discussed in Section 4.4. The applications of the inferred road context for vehicle behavior analysis are demonstrated in Section 4.5. The case study is discussed in Section 4.6 and this chapter is summarized in Section 4.7.

4.2 Problem Statement

As vehicles are driven on the urban roads, their behaviors are believed to follow a hierarchical process that can be divided into three levels: *Activity*, *Action*, and *Trajectory* [83]. Each *activity* is accomplished by a sequence of *actions* that comply with the *traffic rules*, and the *trajectories* are the realizations of these actions with desired vehicle *pose* and *velocities* being generated.

Taking the lane merging activity in Figure 4.1 for instance, the red vehicle tends to merge into the opposite lane, which usually follows a three-stage action as: slowing down to approach the merging point (or stop line), stopping for traffic negotiation, and speeding up to merge into the desired lane. With the desired velocity profile, these actions can be explicitly represented by the transitions between three regions, including current vehicle location ($e_{current}$), merging point (e_{merge}), and the desired lane ($e_{desired}$). Meanwhile, the

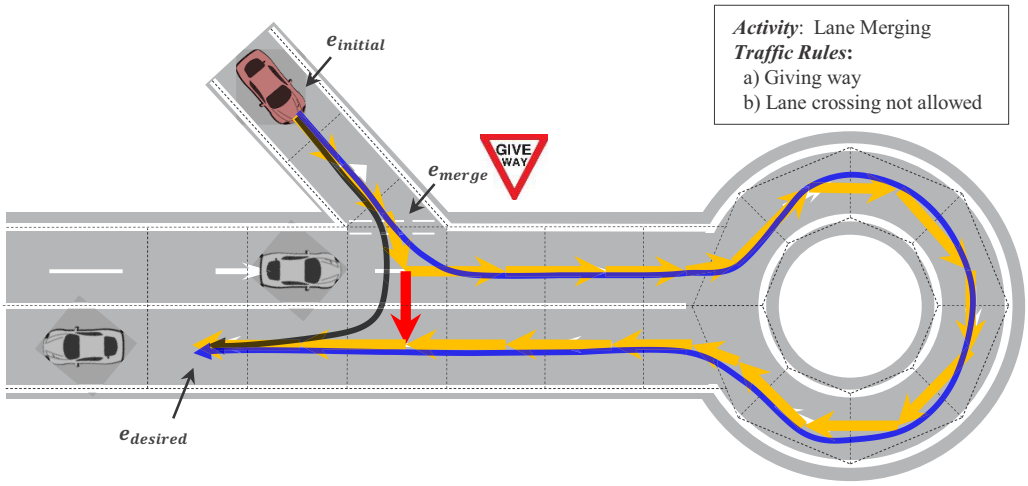


Figure 4.1 Illustration of the process of merging into opposite lane. The environment is evenly discretized by the black dashed grid. The traffic rules allowed region transition is represented as the orange arrows, and the red arrow shows the prohibited region transition. The reasonable vehicle trajectory is shown as the blue curve, while the black curve denotes the trajectory breaking the traffic rules.

transitions between these regions have to comply with the traffic rules. While the vehicle can cross the lane and approach the desired location directly by following the black trajectory in Figure 4.1, the corresponding region transition represented by the red arrow is prohibited by the traffic rules, thus only the blue trajectory is acceptable.

Formally speaking, let α and ρ be the driving activity and traffic rules respectively, and let

$$\mathbf{E} = \{e^{[1]}, e^{[2]}, ..., e^{[N]}\} \tag{4.1}$$

denote a set of mutually exclusive regions decomposing the free space $\mathcal{X}_F \in R^2$, where

$$\bigcup_{i=1}^{N} e^{[i]} = \mathcal{X}_F \text{ and } e^{[i]} \cap e^{[j]} = \emptyset, \forall i \neq j. \tag{4.2}$$

Then the activity α can be modeled as a probabilistic model specified as a tuple $\mathcal{H}^{[\alpha]} = \{\mathbf{A}, \mathbf{E}\}$, where $\mathbf{A}_{N\times N} : \mathbf{E} \times \mathbf{E}$ denotes a N by N transition probability matrix that models the vehicle actions as,

$$\mathbf{A}_{i,j} = \Pr(e^{[j]}|e^{[i]}, \alpha, \rho), \forall i, j \in \{1, 2, ..., N\}. \tag{4.3}$$

The inclusion of ρ into $\mathbf{A}$ indicates that the driving actions should comply strictly with the traffic rules. Exhaustively modeling all the distinct activities, however, is difficult due to the diversity of vehicle behaviors. Also, the model training for each specific activity is time-consuming and lacks generality. Recognized that there always exists a common factor, i.e., the traffic rules ρ, that dominates the actions in each activity model $\mathcal{H}^{[\alpha]}$, by summing all the possible activities α over $\mathcal{H}^{[\alpha]}$, we believe that there exists a probabilistic model $\mathcal{H}^{[\rho]} = \{\mathbf{R}, \mathbf{E}\}$ that is driven by the traffic rules ρ only, where

$$\mathbf{R}_{i,j} = \Pr(e^{[j]}|e^{[i]}, \rho) = \sum_{\alpha} \mathbf{A}_{i,j} \Pr(\alpha), \forall i, j \in \{1, ..., N\} \tag{4.4}$$

As such, the transition probability matrix $\mathbf{R}_{N\times N} : \mathbf{E} \times \mathbf{E}$ models the region transitions that are dominated by the traffic rules.

Although the traffic rules can be properly captured by $\mathcal{H}^{[\rho]}$, the desired speed profile of each activity is not modeled. Though the vehicle velocity profiles can vary significantly given different driving activities and traffic situations, it's believed that the vehicles do follow certain motion patterns on the urban road, such as slowing down for a stop line, accelerating after passing the pedestrian crossing, etc. Therefore, we want to further extend $\mathcal{H}^{[\rho]}$ to include a vehicle motion pattern factor $\mathbf{MP}(x) : v \leftarrow x \in \mathcal{X}_F$ that can provide a generalized velocity profile v for each vehicle position $x \in \mathcal{X}_F$ over the entire free space. The definition of probabilistic road context $\mathcal{RC}$ is therefore proposed as,

Definition 4.2.1. $\mathcal{RC} = \{\mathbf{E}, \mathbf{R}, \mathbf{MP}\}$, $\mathbf{E} = \{e^{[1]}, e^{[2]}, ..., e^{[N]}\}$, $\mathbf{R}_{ij} = \Pr(e^{[j]}|e^{[i]}, \rho), \forall i, j \in \{1, 2, ..., N\}$ *and* $\mathbf{MP}(x) : v \leftarrow x \in \mathcal{X}_F$,

In this study, we seek to infer the road context $\mathcal{RC}$ in a probabilistic manner using a set of randomly observed vehicle behavior data.

4.3 System Overview

Given the analysis of the correlation between the road context and the vehicle behavior, this research aims at learning the road context using the vehicle behavior data that are collected randomly. The overview of the road context learning system is illustrated in Figure 4.2.

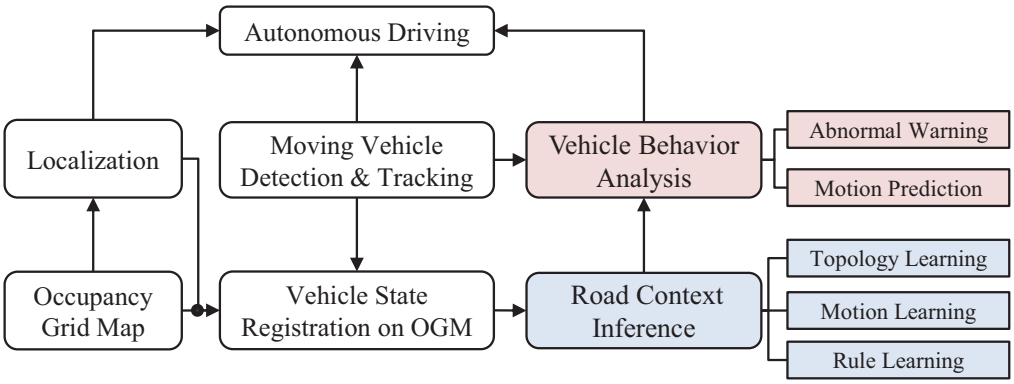


Figure 4.2 Overall framework of road context inference and applications for vehicle behavior analysis.

To start with, the moving obstacle detection and tracking function is needed for vehicle behavior data collection as in Figure 4.3(a), which is accomplished by proposing a spatial-temporal (ST) feature for moving object recognition using only a 2D LIDAR [114]. The vehicle behavior data are represented by the corresponding vehicle state $s = [x, v]$ that includes the vehicle position $x = [\mathrm{x}, \mathrm{y}]$ and the velocity $v = [v_{\mathrm{x}}, v_{\mathrm{y}}]$. Since our autonomous vehicles are usually operated on an Occupancy Grid Map (OGM) generated by SLAM algorithm [61], these observed vehicle states are then transformed into the OGM frame using the vehicle localization function. It is worth mentioning that this study does not assume the existence of a complete trajectory for road context learning, where the learning input is given as a set of randomly-collected and mutually-independent vehicle behavior data as shown in Figure 4.3(b).

After registering the detected vehicle states in the global OGM frame, the road context learning function will come into effect to infer the road context in a probabilistic manner. The fundamental principle of the proposed road context inference approach is to extract the consistencies within the observed vehicle states, where the consistencies can arise from both the spatial distribution of their positions and their driving velocities. The proposed road context inference thereby is divided into three major stages: *topology learning*, *motion learning*, and *rule learning*. The topology learning aims at extracting the vehicle position consistency to model the road topology, and the motion learning is to generalize the vehicle motion patterns using the vehicle velocity information. By coupling the road topology with

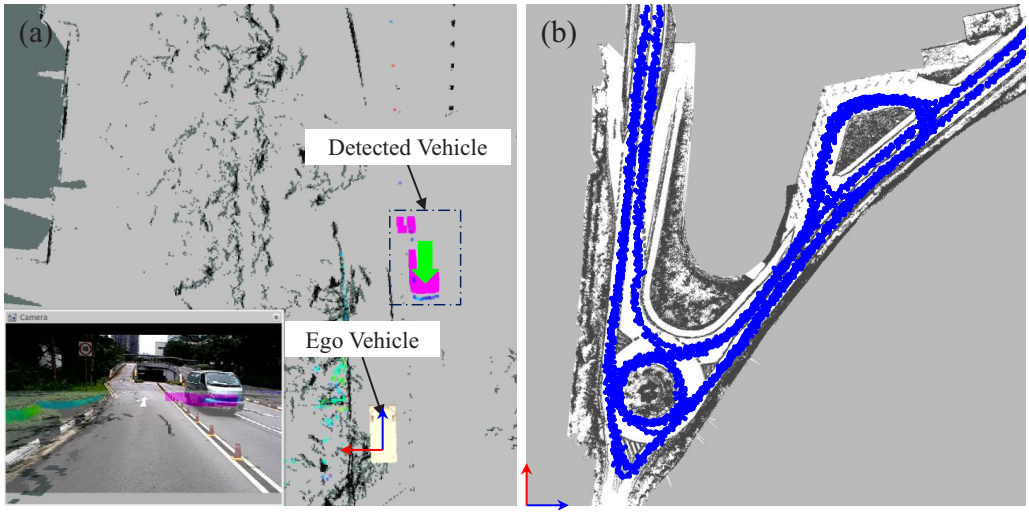


Figure 4.3 Vehicle detection and tracking for data collection: (a) shows the snapshot of vehicle detection and tracking; (b) illustrates the collected data and the Occupancy Grid Map on which the autonomous vehicle are operating.

the generalized motion patterns, a probabilistic representation of the traffic rules can be inferred.

Given the inferred road context, its applications for vehicle behavior analysis can be extended. Recognizing that the vehicle behaviors may or may not comply with the inferred road context, two potential applications are investigated in this study, i.e., *abnormal driving behavior warning* and *long-term motion prediction*. The abnormal driving behavior warning is to trigger alarms when the vehicles are not following the inferred road context. On the other hand, if the vehicles are complying strictly with the road context, the motion prediction can be applied to provide a long-term prediction (up to 10 seconds) of the vehicles' future motion. The autonomous vehicle planning module, thereafter, can utilize this predicted information for either risk evaluation or collision checking.

4.4 Road Context Inference

The details of road context inference will be discussed in this section, which is divided into three major stages: *topology learning*, *motion learning*, and *rule learning*.

4.4.1 Topology Learning

In the urban environment, the vehicles are usually operated on the road, thus the road surface is the area with a high density of vehicle states. The randomly-collected vehicle states thereby can be spatially clustered to extract a compact representation of the road surface. In order to capture the topological connectivity information that can be lost due to the dimension reduction of the statistical clustering, the Growing Neural Gas (GNG) algorithm (see Appendix A) is employed for topology learning in this study.

The GNG algorithm presented in [115] is an unsupervised incremental topology learning algorithm. Given the input sample space $\mathcal{X}_{input} \subset \mathbb{R}^n$ that is distributed as an unknown distribution $\Pr(\mathcal{X}_{input})$, the GNG algorithm can create a topology graph $\mathcal{GNG} = \{\mathbf{K}, \mathbf{C}\}$ that includes the GNG nodes $\mathbf{K}$ and the GNG edges $\mathbf{C}$. Each GNG node $k^{[i]} \in \mathbf{K}, i \in \{1, ..., N\}$ is associated with a feature vector $w^{[i]} \in \mathbb{R}^n$ that encapsulate a number of features to represent the sample space. The connections among the GNG nodes are defined by the GNG edges $\mathbf{C}$ to model the topological structure of the sample space, where the GNG edge $c = [k^{[i]}, k^{[j]}] \in \mathbf{C}, i, j \in \{1, ..., N\}$ indicates that GNG node $k^{[i]}$ and $k^{[j]}$ are topologically connected. Implementation-wise, GNG only uses parameters that are constant in time. Further, it is not necessary to decide on the number of GNG nodes beforehand, because the GNG nodes can be added incrementally during execution. Insertion of new GNG nodes ceases when a defined performance criterion is met or alternatively if a maximum number of GNG nodes has been reached.

In our application, the observed vehicle position $x \in \mathbb{R}^2$ is deemed as the random sample drawn from an unknown distribution $\Pr(x)$, which is then provided as the input sample space of the GNG algorithm. The resulting road topology graph is given as $\mathcal{GNG}_r = \{\mathbf{K}_r, \mathbf{C}_r\}$. The GNG nodes $\mathbf{K}_r = \{k_r^{[1]}, ..., k_r^{[N]}\}$ are learned by minimizing the expected representation error as,

$$\mathbf{K}_r^* = \arg\min_{\mathbf{K}_r} \sum_{i=1}^{N} \int D_e(x, w_r^{[i]}) \Pr(x) dx, \tag{4.5}$$

where $w_r^{[i]} = [k_r.\mathrm{x}, k_r.\mathrm{y}]^{\mathrm{T}} \in \mathbb{R}^2, \forall i \in \{1, ..., N\}$ is the feature vector that corresponds to GNG node $k_r^{[i]}, i \in \{1, ..., N\}$ and is regarded as the GNG node's position in the input sample space. The representation error function is therefore defined as the Euclidean distance measurement $D_e(x, w_r^{[i]}) = \parallel x, w_r^{[i]} \parallel_2$. As such, the GNG nodes $\mathbf{K}_r$ will have high chances to reside within the road surface, and the road structure can be correctly captured. Additionally, the GNG edges $\mathbf{C}_r$ only exist in the regions with $\Pr(x) > 0$ [115], so the road connectivity can be well represented as well. Thanks to the incremental property of the GNG algorithm, the road topology can always be efficiently updated when new vehicle states are observed.

Given the road topology graph $\mathcal{GNG}_r$, the driving space $\mathcal{X}_F$ can be evenly discretized by generating a Voronoi diagram w.r.t. the feature vectors $\mathbf{W}_r = \bigcup_{\forall k_r \in \mathbf{K}_r} (w_r \leftarrow k_r)$. The spatial region $e^{[i]} \in \mathbf{E}, \forall i \in \{1, ..., N\}$ thereafter can be represented by its corresponding Voronoi cell as,

$$e^{[i]} = \{x \in \mathcal{X}_F | D_e(x, w_r^{[i]}) \leq D_e(x, w_r^{[j]}), \forall j \neq i\}. \tag{4.6}$$

Moreover, the one-on-one representation of the GNG edge $c_r = [k_r^{[i]}, k_r^{[j]}]^{\mathrm{T}} \in \mathbf{C}_r, i, j \in \{1, ..., N\}$ in the input sample space is given as $l_r = [w_r^{[i]}, w_r^{[j]}]^{\mathrm{T}} \in \mathbf{L}_r$, where l_r denotes the *topological path segment* that connects the feature vectors $w_r^{[i]}$ and $w_r^{[j]}$ via a straight line, and $\mathbf{L}_r = \bigcup_{\forall c_r \in \mathbf{C}_r} (l_r \leftarrow e_r)$.

4.4.2 Motion Learning

Similarly, the vehicles' velocities also follow certain patterns on the urban road, like slowing down for pedestrian crossing, accelerating after passing the stop-line, and so forth. These motion consistencies can be directly employed for vehicle behavior representation and road context modeling. In order to generalize these motion consistencies, the Gaussian Process algorithm is adopted, which allows a time-independent representation of the vehicle motion pattern over the entire driving space $\mathcal{X}_F$.

Generally speaking, the Gaussian Process is a collection of random variables, any finite number of which have a joint Gaussian distribution [116], which has been widely adopted

for regression and classification. Letting $\mathbf{X}$ and y denote the input vector and output variable respectively, the Gaussian Process Regression (GPR) model can be presented as,

$$y = f(\mathbf{X}) + n, f \sim \mathbf{GP}(\mathrm{m}; \mathrm{K}), n \sim \mathcal{N}(0, \sigma_n^2), \tag{4.7}$$

where $f(\mathbf{X})$ is the function distributed as a Gaussian Process with mean function m and covariance function K. Given the training data $\{\mathbf{X}, y\}$, the posterior distribution of the output variable y^* mapped from the testing data $\mathbf{X}^*$ is believed to follow the normal distribution as $y^* \sim \mathcal{N}(\bar{y^*}, \Sigma_{y^*})$, where

$$\bar{y^*} = \mathrm{m}(\mathbf{X}^*) + \mathrm{K}^T(\mathbf{X}, \mathbf{X}^*)\mathrm{K}^{-1}(\mathbf{X}, \mathbf{X})(y - \mathrm{m}(\mathbf{X})),$$
$$\Sigma_{y^*} = \mathrm{K}(\mathbf{X}^*, \mathbf{X}^*) - \mathrm{K}^T(\mathbf{X}, \mathbf{X}^*)\mathrm{K}^{-1}(\mathbf{X}, \mathbf{X})\mathrm{K}(\mathbf{X}, \mathbf{X}^*).$$

For our motion learning purpose, we seek to generalize a motion pattern $\mathbf{MP} : v \leftarrow x$ that maps the posterior distribution of the velocity $v = [v_\mathrm{x}, v_\mathrm{y}]^\mathrm{T}$ from the position $x = [\mathrm{x}, \mathrm{y}]^\mathrm{T}$ over the entire space, i.e. $v_{mp} \sim \mathbf{MP}(x) = \mathcal{N}(\bar{v}_{mp}, \Sigma_{v_{mp}}), x \in \mathcal{X}_F$, where the time-dependent aspect of the model is avoided. Implementation-wise, the training inputs are provided by the observed vehicle positions and their corresponding velocities are provided as the training output. As a popular strategy, the Zero mean function $\mathrm{m}(\mathbf{X}) = 0$ and the Squared Exponential covariance function $\mathrm{K}(\mathbf{X}, \mathbf{X}') = \sigma_y \exp \frac{-(\mathbf{X}-\mathbf{X}')^2}{2ll^2}$ are utilized. The corresponding hyper-parameters $\{\sigma_n, \sigma_y, ll\}$ are learned by maximizing the log-likelihood of the observations in the training data. The v_x and v_y are assumed to be independent, thereby two separate GPR models are trained for v_x and v_y respectively. This assumption might not hold exactly, but provides a reasonable trade-off between theory completeness and computational complexity. It is worth mentioning that each training datum is independent of the others within the training set, which means that the GPR model can also be incrementally built if the hyper-parameters are properly updated.

4.4.3 Rule Learning

Given the consistencies abstracted from the vehicle positions and velocities, the problem now lies in the coupling of these consistencies to infer the transition probability matrix $\mathbf{R}$ that is dominated by the traffic rules.

A. Analysis

The vehicle motion is usually consistent with the traffic rules, meanwhile the motion pattern $\mathbf{MP}$ is generalized from these vehicle motion consistencies. Therefore, the traffic rules-dominated region transitions can be considered as the compact representation of the generalized motion pattern $\mathbf{MP}$. The inference over $\mathbf{R}$ thereby can be accomplished by utilizing the generalized motion pattern to simulate the region transitions. Moreover, the time-independent property of the Gaussian Process suggests that the inference over $\mathbf{R}$ can be performed for each region $e \in \mathbf{E}$ independently.

To begin with, a N-dimension multinomial variable $\Omega_t = [\psi_t^{[1]}, ..., \psi_t^{[N]}]^{\mathrm{T}}$ is introduced first to represent the N mutually exclusive states of the region e at time t, namely $\psi_t^{[i]} = 1$ iff $e_t = e^{[i]}$, otherwise $\psi_t^{[i]} = 0$. Moreover, let $\mathcal{V}^{[i]} = [\mu_1^{[i]}, ..., \mu_N^{[i]}]^{\mathrm{T}}, \sum_{j=1}^{N} \mu_j^{[i]} = 1$ denote the transition probability vector that corresponds to $e^{[i]}$, where $\mu_j^{[i]}$ denotes the possibility of $\psi_{t+\Delta t}^{[j]} = 1$. Essentially, the transition probability vector $\mathcal{V}^{[i]}$ is identical to the ith row of the transition probability matrix $\mathbf{R}$.

Without losing generality, let $e_t = e^{[i]}, i \in \{1, ..., N\}$ be a generic region to represent where the vehicle locates at time t and let $e_{t+\Delta t}$ be the generic region to be transited at time $t + \Delta t$, then the distribution of $e_{t+\Delta t}$ or equivalently $\Omega_{t+\Delta t}$ can be formulated as,

$$\Pr(\Omega_{t+\Delta t}|\mathcal{V}^{[i]}) = \prod_{j=1}^{N} (\mu_j^{[i]})^{\psi_{t+\Delta t}^{[j]}}, \tag{4.8}$$

which is a standard multinomial distribution. Then the inference over $\mathcal{V}^{[i]}$ can be made via maximizing the log-likelihood of the observations as,

$$(\mathcal{V}^{[i]})^{ML} = \arg\max_{\mathcal{V}^{[i]}} \ln \Pr(\mathcal{O}_{\Omega}|\mathcal{V}^{[i]}), \tag{4.9}$$

where $\mathcal{O}_\Omega = \{o_1, ..., o_M\}$ denotes the observation set consisting of M independent observations. Each observation $o_i \in \mathcal{O}_\Omega, i \in \{1, ..., M\}$ is defined w.r.t. the multinomial variable Ω as $o_i = [\psi_{t+\Delta t}^{[1]}, ..., \psi_{t+\Delta t}^{[N]}]^{\mathrm{T}}$, where $\psi_{t+\Delta t}^{[j]} = 1$ iff $\Omega_{t+\Delta t} = e^{[j]}, j \in \{1, ..., N\}$.

The real observations, however, can be missing in some regions due to the sensing limitation, therefore this study seeks to approximate the observation set $\mathcal{O}_\Omega$ by simulating the region transitions using the generalized motion pattern $\mathbf{MP}$. Specifically, if the vehicles are located in region $e_t = e^{[i]}$ at time t, the probabilistic distribution of the region $e_{t+\Delta t}$ where the vehicle locates at time $t + \Delta t$ can be simulated as,

$$\Pr(e_{t+\Delta t}|e_t = e^{[i]}) = \sum_{x_{t+\Delta t}} \Pr(e_{t+\Delta t}|x_{t+\Delta t})\Pr(x_{t+\Delta t}|e^{[i]}), \tag{4.10}$$

where $\Pr(e_{t+\Delta t}|x_{t+\Delta t}) = 1$ iff $x_{t+\Delta t} \in e_{t+\Delta t}$, otherwise $\Pr(e_{t+\Delta t}|x_{t+\Delta t}) = 0$. The distribution of vehicle position $x_{t+\Delta t}$ given $e^{[i]}$ can be further analyzed as,

$$\begin{aligned}
&\Pr(x_{t+\Delta t}|e^{[i]}) = \sum_{x_t} \Pr(x_{t+\Delta t}|x_t, e^{[i]})\Pr(x_t|e^{[i]}) \\
&= \sum_{x_t}\sum_{v_t} \Pr(x_{t+\Delta t}|x_t, v_t, e^{[i]})\Pr(v_t|x_t, e^{[i]})\Pr(x_t|e^{[i]}) \\
&= \sum_{x_t}\sum_{v_t} \underbrace{\Pr(x_{t+\Delta t}|x_t, v_t)}_{\textbf{Propagation}} \underbrace{\Pr(v_t|x_t)}_{\textbf{GPR}} \underbrace{\Pr(x_t|e^{[i]})}_{\textbf{Sampling}},
\end{aligned} \tag{4.11}$$

where the generalized vehicle velocity is distributed as $v_t \sim \Pr(v_t|x_t) = \mathbf{MP}(x_t)$ and the position transition $\Pr(x_{t+\Delta t}|x_t, v_t)$ can be modeled as the linear propagation: $[\mathrm{x}_{t+\Delta t}, \mathrm{y}_{t+\Delta t}]^{\mathrm{T}} = [\mathrm{x}_t, \mathrm{y}_t]^{\mathrm{T}} + [v_{\mathrm{x}}, v_{\mathrm{y}}]^{\mathrm{T}}\Delta t$. The vehicle position sampling function $\Pr(x_t|e^{[i]})$ follows a uniform distribution within the region represented by $e^{[i]}$.

Given the distribution $\Pr(e_{t+\Delta t}|e_t = e^{[i]})$ in Equation (4.10), we can always simulate the vehicle region transition and draw the propagated vehicle position $x_{t+\Delta t}$ according to $x_{t+\Delta t} \sim \Pr(x_{t+\Delta t}|e^{[i]})$. The approximated observation set $\widetilde{\mathcal{O}}_\Omega$, thereafter, can be generated by checking the region where each propagated vehicle position $x_{t+\Delta t}$ locates. The maximum likelihood solution of $\mathcal{V}^{[i]} = [\mu_1^{[i]}, ..., \mu_N^{[i]}]^{\mathrm{T}}$ is then given as $\mu_j^{[i]} = M_j/M, \forall j \in \{1, ..., N\}$, where M_j is the number of observations of $\psi_{t+\Delta t}^{[j]} = 1$.

Algorithm 6: Transition Probability Matrix Inference

input : $\mathbf{E}$, $\mathbf{C}_r$
output : $\mathbf{R}$

begin
  $\mathbf{R} \leftarrow \mathbf{C}_r$
  for $i \leftarrow 1$ **to** N **do**
    $\mathcal{V}^{[i]} \leftarrow \mathbf{R}_{i,\{1,\ldots,N\}}$
    while $\neg$`Converging`$(L^{[i]})$ **do**
      $\bar{\Phi}^{[i]} = \Phi^{[i]} = \emptyset$
      $\mathbf{P}^{[i]} \leftarrow$ `SampleArea`$(e^{[i]})$
      for $x_t \in \mathbf{P}^{[i]}$ **do**
        $x_{t+\Delta t} \leftarrow$ `Propagate`$(x_t,$ `GPR`$(x_t))$
        $e_{t+\Delta t} \leftarrow$ `SampleRule`$(e^{[i]}, \mathcal{V}^{[i]})$
        $\varpi \leftarrow$ `Weight`$(x_{t+\Delta t}, e_{t+\Delta t})$
        $\bar{\Phi}^{[i]} \leftarrow \bar{\Phi}^{[i]} \bigcup \langle x_{t+\Delta t}, e_{t+\Delta t}, \varpi \rangle$
      end
      $\Phi^{[i]} \leftarrow$ `ImportanceSampling`$(\bar{\Phi}^{[i]})$
      for $j \leftarrow 1$ **to** N **do**
        $\mathcal{V}^{[i]} \leftarrow$ `TransitionAnalyze`$(\Phi^{[i]})$
      end
      $L^{[i]} \leftarrow$ `Evaluate`$(\Phi^{[i]})$
    end
    $\mathbf{R}_{i,\{1,\ldots,N\}} \leftarrow \mathcal{V}^{[i]}$
  end
  return $\mathbf{R}$
end

B. Algorithm

According to the above analysis, the inference over the traffic rules-dominated transition probability matrix $\mathbf{R}$ is demonstrated in Algorithm 6, where the road topology edges $\mathbf{C}_r$ are considered as the initial hypothesis.

For each region $e^{[i]} \in \mathbf{E}$, we first conduct rejection-sampling to draw M vehicle position samples $\mathbf{P}^{[i]}$ from the distribution $\Pr(x_t|e_t = e^{[i]})$ (Line 7 in Algorithm 6). For each sample $x_t \in \mathbf{P}^{[i]}$, its corresponding velocity distribution can be generated as $v_t \sim \mathbf{MP}(x_t)$. Thereafter, the simulated vehicle position $x_{t+\Delta t}$ is propagated by drawing samples from $\Pr(x_{t+\Delta t}|x_t, v_t)$. Given the transition probability vector $\mathcal{V}^{[i]}$ of the current iteration, $e_{t+\Delta t}$ can be similarly sampled according to Equation (4.8). After this, the simulated vehicle position $x_{t+\Delta t}$ and the sampled region $e_{t+\Delta t}$ are combined as a joint particle and weighed as $\varpi = 1$ iff $x_{t+\Delta t} \in e_{t+\Delta t}$, otherwise ϖ is assigned a small value.

After conducting importance sampling w.r.t. the proposal particle set $\bar{\Phi}^{[i]}$, the transition probability vector $\mathcal{V}^{[i]}$ can be updated as $\mu_j^{[i]} = M_j/M, \forall j \in \{1, ..., N\}$, where M_j is the number of survival particles with $e_{t+\Delta t} = e^{[j]}$. This process is repeated until the evaluation metric

$$L^{[i]} = \sum_{m=1}^{M} \varpi_m/M, \forall \langle x_m, e_m, \varpi_m \rangle \in \Phi^{[i]} \tag{4.12}$$

converges. Rather than only sampling M observations for maximum likelihood inference as Equation (4.9), the designed algorithm is implemented in an iterative manner, which is to ensure that enough observations are generated to account for the noise of the collected vehicle data. The convergence of $L^{[i]}$ can be guaranteed given our analysis in Equation (4.8)-(4.11).

4.5 Applications for Vehicle Behavior Analysis

This section will present the applications of the inferred road context. Depending on whether the vehicle behavior complies with the road context or not, the vehicle behavior analysis is divided into two types, including abnormal driving behavior detection and the long-term motion prediction.

4.5.1 Abnormal Driving Behavior Warning

The extension of the inferred road context for abnormal driving behavior detection is explored first, which is spurred by the fact that the traffic rules may not be strictly followed by all the vehicles. This study focuses on two abnormal driving behaviors, including wrong-way driving and reckless driving.

A. Wrong-way driving warning

The wrong-way driving warning is to identify whether the vehicles are driven in the wrong lane or direction, which can be accomplished by checking the vehicle's trajectory against the region transitions that are dominated by the traffic rules.

To start with, we construct a traffic rules graph $\mathcal{G}_{rule} = \{\mathbf{K}_{rule}, \mathbf{C}_{rule}\}$ to map the regions and the corresponding region transitions dominated by the traffic rules. In other words, each region $e^{[i]} \in \mathbf{E}$ will be considered as a graph node $k_{rule}^{[i]} \in \mathbf{K}_{rule}$ and the directed edge $c_{rule} = [k_{rule}^{[i]}, k_{rule}^{[j]}] \in \mathbf{C}_{rule}$ will start from $k_{rule}^{[i]}$ and be directed to $k_{rule}^{[j]}$ iff $\mathbf{R}_{i,j} > 0$.

For the wrong-way driving detection, we can check the regions $e_t = e^{[i]}, i \in \{1, ..., N\}$ and $e_{t+\Delta t} = e^{[j]}, j \in \{1, ..., N\}$ where the vehicle locates at two consecutive times with small interval Δt. Then the graph nodes $k_{rule}^{[i]}$ and $k_{rule}^{[j]}$ that correspond to $e^{[i]}$ and $e^{[j]}$ can be mapped out from the traffic rules graph $\mathcal{G}_{rule}$. By associating each graph edge $c_{rule} \in \mathbf{C}_{rule}$ with the length $length(c_{rule}) = 1$, the shortest path between the graph nodes $k_{rule}^{[i]}$ and $k_{rule}^{[j]}$ can be searched using the Dijkstra algorithm [45]. If the shortest path's length is over a desired threshold ΔD_{wrong}, which suggests that the region transition is abnormal with a big jump within the short duration Δt, then the vehicle behavior is classified as wrong-way driving.

B. Reckless driving warning

As discussed earlier, the vehicle velocities always follow some typical patterns that can be properly generalized by the Gaussian Process Regression, such as decelerating at pedestrian crossings or the stop lines. The detection of reckless driving therefore can be partially

solved by comparing the observed vehicle velocity against the reference velocity represented by the generalized motion pattern $\mathbf{MP}$.

Formally speaking, let the observed vehicle state be $[x_o, v_o]$, its corresponding reference velocity v_{ref} thereby can be generalized as $v_{ref} \sim \mathbf{MP}(x_o) = \mathcal{N}(\bar{v}_{ref}, \Sigma_{v_{ref}})$. Thereafter, we can check the deviation of the observed vehicle velocity v_o from the reference velocity v_{ref} to evaluate the vehicle's current driving status. For the sake of reckless driving warning in this study, the vehicle speed deviation will be employed. Specifically, the reckless driving warning will be triggered if the observed vehicle speed's deviation from the reference speed is larger than a designed speed threshold $\Delta v_{reckless}$, i.e. $|v_o| - |\bar{v}_{ref}| > \Delta v_{reckless}$. In the next chapter, we will further explore the utility of this velocity deviation and propose a reaction-based vehicle motion intention modeling.

4.5.2 Long-term Motion Prediction

Assuming that the vehicle will strictly follow the inferred road context, another important application of the inferred road context is the long-term motion prediction.

A. Analysis

Since the vehicles are always regularized by the traffic rules in the urban road environment, the areas where the vehicles are allowed to move can be reasonably predicted by the inferred road context. As such, a road context-based area prediction algorithm is proposed in Algorithm 7. Specifically, if the region where the vehicle is located as $e_{current}$, then the regions $e^{[i]}, i \in \{1, ..., N\}$ satisfying the non-zero transition probability from $e_{current}$ can be easily identified by looking up the transition probability matrix $\mathbf{R}$. The same rule is then applied to $e^{[i]}$, and this process is repeated until the prediction horizon H is satisfied. Afterward, the topological path segments $l_r \in \mathbf{L}_r$ connecting the predicted areas $\mathbf{E}_{predict}$ are combined to represent the *predicted topological path* $\mathcal{T}$, which is to provide a more intuitive representation of the vehicle's future behavior.

While the area prediction can provide a reasonable guess at the regions where the vehicle can move, the exact prediction of the vehicle position cannot be accomplished. Therefore, the generalized motion pattern, which can properly capture the vehicle's dynamics

Algorithm 7: Road Context based Area Prediction

input : $e_{current}$, $\mathbf{E}$, $\mathbf{R}$
output : $\mathbf{E}_{predict}$

begin
  $\mathbf{E}_{predict} = \mathbf{E}_{searched} = \emptyset$
  $\mathbf{E}_{predict} \leftarrow \mathbf{E}_{predict} \bigcup e_{current}$
  for $h \leftarrow 1$ **to** H **do**
    for $e_{source} \in \mathbf{E}_{predict} \setminus \mathbf{E}_{searched}$ **do**
      for $i \leftarrow 1$ **to** N **do**
        if $\Pr(e^{[j]}|e_{source}, \rho) > 0$ **then**
          $\mathbf{E}_{predict} \leftarrow \mathbf{E}_{predict} \bigcup e^{[i]}$
        end
      end
      $\mathbf{E}_{searched} \leftarrow \mathbf{E}_{searched} \bigcup e_{source}$
    end
  end
  return $\mathbf{E}_{predict}$
end

and provide the corresponding reachability information, will be employed for the motion prediction purpose.

Formally speaking, let $\varphi = [x, e]$ be the augmented vehicle position that includes the vehicle position and the region it occupies. Then the distribution of $\varphi_{t+\Delta t}$ given φ_t and the vehicle position observation $o_{t+\Delta t}$ can be formulated as,

$$\Pr(\varphi_{t+\Delta t}|\varphi_t, o_{t+\Delta t}) = \eta \Pr(o_{t+\Delta t}|\varphi_{t+\Delta t}) \Pr(\varphi_{t+\Delta t}|\varphi_t) \tag{4.13}$$

where η is the normalization factor. For the motion function $\Pr(\varphi_{t+\Delta t}|\varphi_t)$, it can be further factorized as,

$$\begin{aligned} \Pr(\varphi_{t+\Delta t}|\varphi_t) &= \Pr(x_{t+\Delta t}|x_t, e_t) \Pr(e_{t+\Delta t}|x_t, e_t) \\ &= \sum_{v_t} \Pr(x_{t+\Delta t}|x_t, v_t) \Pr(v_t|x_t) \Pr(e_{t+\Delta t}|x_t, e_t), \\ &= \sum_{v_t} \sum_{\rho} \underbrace{\Pr(x_{t+\Delta t}|x_t, v_t)}_{\textbf{Propagation}} \underbrace{\Pr(v_t|x_t)}_{\textbf{GPR}} \underbrace{\Pr(e_{t+\Delta t}|e_t, \rho)}_{\textbf{Traffic Rule}} \Pr(\rho), \end{aligned} \tag{4.14}$$

where $x_{t+\Delta t}$ and $e_{t+\Delta t}$ is assumed to be conditional independent given φ_t. As such, the inferred traffic rules act as an extra constraint on the motion transition. Regarding the observation function $\Pr(o_{t+\Delta t}|\varphi_{t+\Delta t})$, since the regions where the vehicle tends to move can be roughly predicted by the area prediction algorithm in Algorithm 7, the likelihood field w.r.t. the predicted topological path $\mathcal{T}$ can be generated. The corresponding observation function is then formulated as,

$$\Pr(o_{t+\Delta t}|\varphi_{t+\Delta t}) = \frac{1}{\sqrt{2\pi}D}\exp(-\frac{D_p(x_{t+\Delta t},\mathcal{T})^2}{2D^2}) \tag{4.15}$$

where $D_p(x_{t+\Delta t},\mathcal{T})$ measures the distance between the observed vehicle position $x_{t+\Delta t}$ and the predicted topological path $\mathcal{T}$, and the parameter D is to adjust how closely the predicted vehicle position should adhere to the topological path $\mathcal{T}$. The objective of this observation function is to remove the outliers that are off the road surface.

B. RuleGPPF

The exact prediction following Equation (4.13)-(4.15) is complicated, so the particle filter is therefore employed as an approximated prediction approach in this study. The proposed motion prediction algorithm ***RuleGPPF*** is illustrated in Algorithm 8, where the input vehicle position x_0 follows a distribution $\Pr(x_0)$.

To start with, the source particle set Φ_{source} is generated by drawing N samples $\mathbf{P}_{init}$ from $\Pr(x_0)$ and checking the region where each sample locates. For each particle in Φ_{source}, we use the generalized motion pattern $\mathbf{MP}$ to simulate the vehicle position transitions and check the corresponding region transitions. If the region transition satisfies the inferred traffic rules, the propagated vehicle position x' and the corresponding region e' is inserted into the proposal particle set $\bar{\Phi}$, whose weight is then given by Equation (4.15). Afterward, the proposal particles are re-sampled to remove the outliers. Given the survival particles Φ, the predicted vehicle position $\bar{x}_t$ at time t is then approximated as $\bar{x}_t = \sum_{m=1}^{M}(\varpi_m \times x_m)$, where x_m denotes the particle's position and ϖ_m is the corresponding weight. The prediction error is then computed as the Euclidean distance between the ground-truth vehicle position and the predicted vehicle position. The survival particles

Algorithm 8: RuleGPPF

input : $\Pr(x_0)$, $\mathbf{E}$, $\mathbf{R}$
output : $\bar{x}_t, \forall t \in \{1, ..., H\}$

begin
  $\Phi_{source} = \emptyset$
  $\mathbf{P}_{init} \leftarrow \texttt{Sample}(\Pr(x_0))$
  for $x \in \mathbf{P}_{init}$ **do**
    $e \leftarrow \texttt{RegionCheck}(x)$
    $\Phi_{source} \leftarrow \Phi_{source} \bigcup \langle x, z, 1/M \rangle$
  end
  for $t \leftarrow 1$ **to** H **do**
    $\bar{\Phi} = \Phi = \emptyset$
    for $\langle x, e, \varpi \rangle \in \Phi_{source}$ **do**
      $x' \leftarrow \texttt{Propagate}(x, \texttt{GPR}(x))$
      $e' \leftarrow \texttt{RegionCheck}(x')$
      if $\Pr(e'|e, \rho) > 0$ **then**
        $\varpi' \leftarrow \texttt{Weight}(x', e')$
        $\bar{\Phi} \leftarrow \bar{\Phi} \bigcup \langle x', e', \varpi' \rangle$
      else
        $\bar{\Phi} \leftarrow \bar{\Phi} \bigcup \langle x, e, \varpi \rangle$
      end
    end
    $\Phi \leftarrow \texttt{ImportanceSampling}(\bar{\Phi})$
    $\bar{x}_t \leftarrow \texttt{ParticleAnalyze}(\Phi)$
    $\Phi_{source} \leftarrow \Phi$
  end
  return $\bar{x}_t, \forall t \in \{1, .., H\}$
end

Φ are then utilized as the source particles for the next prediction interval. This process will be repeated until the prediction horizon is reached.

4.6 Case Study

This section presents the case study of a single-lane road that includes a T-Junction (TJ) and a Round-About (RA) within the campus of National University of Singapore (see Figure 4.4(a)). The corresponding Occupancy Grid Map (OGM) on which the autonomous vehicles are operating is shown in Figure 4.4(b). A fleet of autonomous vehicles operating autonomously within the campus was employed for the data collection purpose, and 4,000 random vehicle states were collected over two weeks, as shown in Figure 4.3(b). As

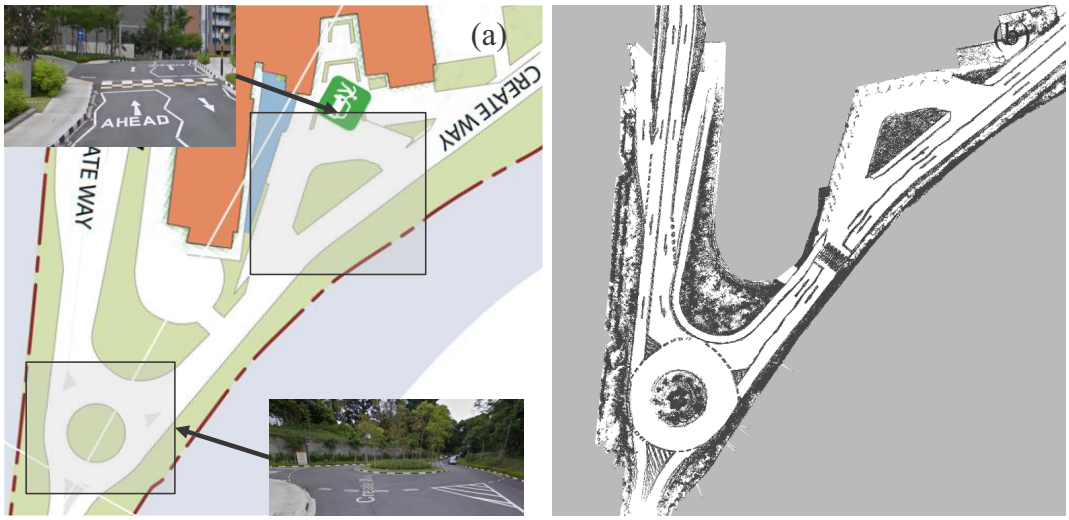


Figure 4.4 Case study environment for road context inference: (a) shows the environment structure, and the corresponding Occupancy Grid Map is demonstrated in (b).

demonstrated, the collected vehicle states are quite noisy and distributed randomly on the road surface, which, however, is well acknowledged by our proposed algorithm and can be properly handled in a probabilistic manner.

4.6.1 Road Context Inference

Given the collected vehicle states, the road topology information can be learned by the GNG algorithm. The GNG parameters are essential to guarantee the training performance, but a general rule for GNG parameters tuning is still lacking. Therefore, extensive trials-and-error testing has been carried out and the desired parameters are given in Table 4.1, where the meaning of these parameters can be referred to Appendix A.

The topology learning results are demonstrated in Figure 4.5(a). The red dots depict the feature vectors associated with the GNG nodes, and the undirected blue edges denote the topological path segments representing the topological connectivities within these GNG nodes. As demonstrated, the road structure and connectivity can be properly captured except that some nodes are mis-connected as lane crossing. This is reasonable because only the vehicle position information is fed into the GNG algorithm, and the data collected from different lanes will share high similarity if they are close enough. This mis-connection will be addressed when the vehicle motion patterns are coupled. Figure 4.5(b) shows the

Table 4.1 GNG Parameters for Topology Learning

Parameters	Value	Parameters	Value
$age_{increase}$	1	age_{max}	13
ϵ_b	0.25	ϵ_n	$1e-6$
Γ	80	ϑ	0.5
ν	0.99	n_{max}	180

corresponding Voronoi diagram w.r.t. the GNG nodes, where the entire space can be evenly discretized and each Voronoi cell corresponds to a region $e \in \mathbf{E}$.

In order to validate the incremental property of the topology learning, another experiment was conducted. In the beginning, the 4,000 training samples were divided into two groups depending on whether the samples are located in the left side or the right side of the map, and 500 samples were used for training in each epoch. For the purpose of evaluating the incremental property, only the samples locating the left side of the map were used in the first four training epochs. Afterward, the remaining 2000 samples were used for the next four training epochs. One snapshot of the topology learning result in the fourth epoch is shown in Figure 4.6(a), where the topological connectivity of the lefthand-side road can be properly captured. Figure 4.6(b) depicts the topology learning result using 4000 samples over the entire map. From Figure 4.6(a) to Figure 4.6(b), it can be easily observed that the road topology can be incrementally learned given the increasing number of samples, which is further evidenced by the training error analysis in Figure 4.6(c). The training error converges quickly in the first four training epochs and increases a bit in the fifth training epoch due to the "unfamiliar" training samples. Finally, the training error converges to an acceptable value, and an incrementally constructed road topology graph can be achieved.

Thereafter, the motion learning came into effect to generalize the vehicle motion pattern using the Gaussian Process Regression. Two separate GPR models are trained for v_{x} and v_{y} respectively, where the hyper-parameters are learned by maximizing the log-likelihood functions. The mean values of the generalized velocity distribution in the x and y direction are illustrated in Figure 4.7 (a) and Figure 4.7(b). Given these velocity distributions, the velocity flow over the whole map can be easily determined, which can properly represent

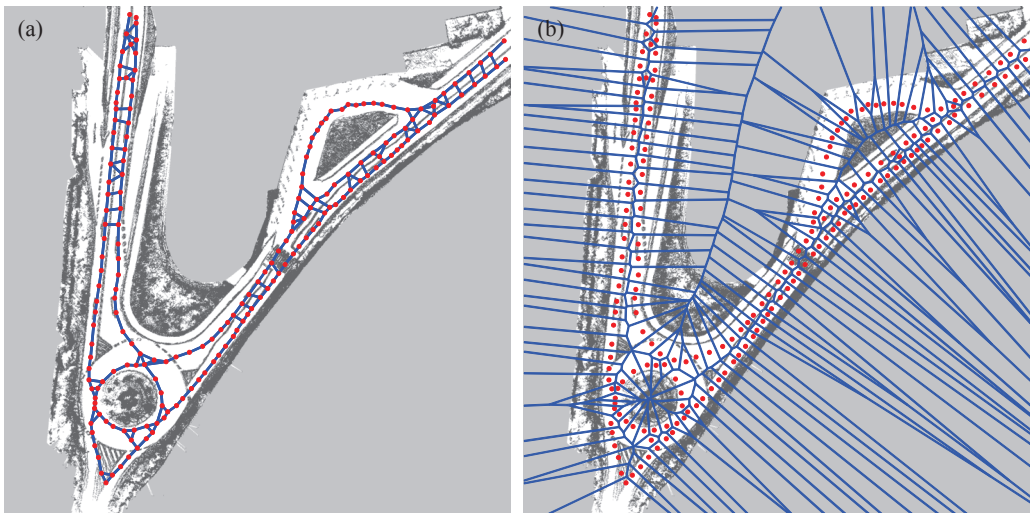


Figure 4.5 Topology learning result: The GNG learning result is demonstrated in (a) and the corresponding Voronoi diagram is shown in (b).

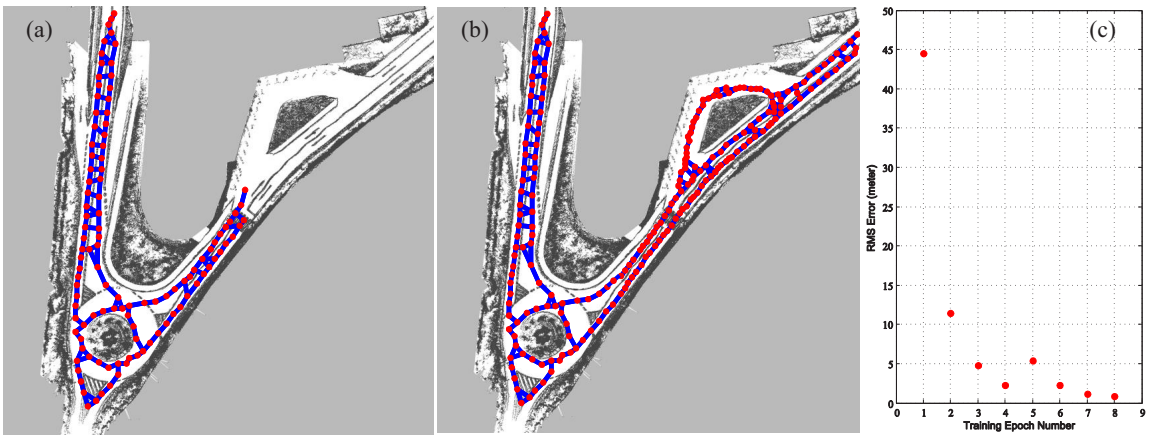


Figure 4.6 Analysis of the incremental property of topology learning: The topology learning result using 2000 samples located on the lefthand side of the map is shown in (a). Thereafter, another 2000 samples located on the righthand side of the map are used, and the topology learning over the entire map is demonstrated in (b). (c) depicts the convergence of training error, where 400 samples are injected into GNG in each epoch.

the typical vehicle motion patterns. For instance, the velocity flows of the RA and TJ areas are highlighted in Figure 4.7(c) and Figure 4.7(d), from which we can find the vehicle moving direction and speed can be properly captured.

Given the road topology and motion pattern learning results, the traffic rules can be inferred by following Algorithm 6. By removing the traffic rule edges associated with $\mathbf{R}_{i,j} < 0.001, i, j \in \{1, ..., N\}$, the graph representation of the probabilistic traffic rules is shown in Figure 4.8 (a), where the region transition probability is visualized with different colors. The highlights of the traffic rules inference results at TJ and RA areas are shown in Figure 4.8(b) and Figure 4.8(c). Compared to the GNG training result in Figure 4.5(a), the undirected edges are replaced by the directed arrows to model the probabilistic dependency. The GNG edges that cross the lane are removed accordingly, as they are not consistent with the traffic rules.

4.6.2 Applications for Vehicle Behavior Analysis

Given the inferred road context, the vehicle behavior analysis can be achieved. For the sake of evaluating the wrong-way driving detection, we manually drove the vehicle illegally to cross the lane, and two snapshots of the detecting results are presented in Figure 4.9, where the warning triggering threshold is set as $\Delta D_{wrong} = 2$. The vehicle's trajectory is marked as either red or green to indicate whether the warning is triggered, from which we can find the wrong-way driving can be detected in an efficient manner.

Similarly, the detecting result of reckless driving is demonstrated in Figure 4.10. As shown in Figure 4.10(a), the vehicle was approaching a stop line of the round-about, where the vehicle was supposed to decelerate and maintain a slow speed according to the reference speed plotted as the red curve in Figure 4.10(b). The vehicle, however, was driven in a reckless manner, and the deviation of the observed vehicle speed from the reference speed is significant enough to trigger the warning.

Before moving to the motion prediction, Figure 4.11 demonstrates the area prediction results. The predicted areas are represented by the corresponding predicted topological path $\mathcal{T}$, from which we find that the area prediction can reasonably estimate the vehicle's

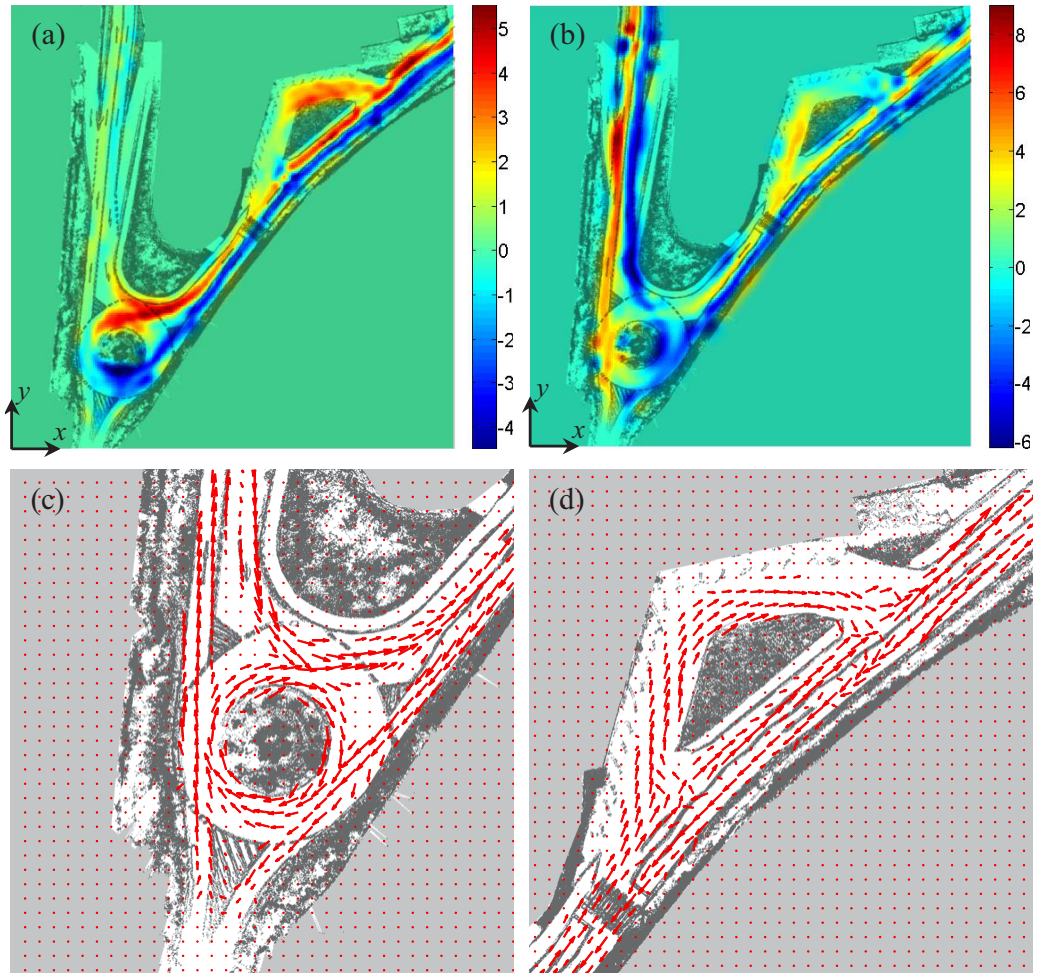


Figure 4.7 Motion learning results: (a) and (b) show the mean value of the generalized velocity distribution in the x and y direction respectively. (c) and (d) illustrate the velocity flow at the T-junction and roundabout.

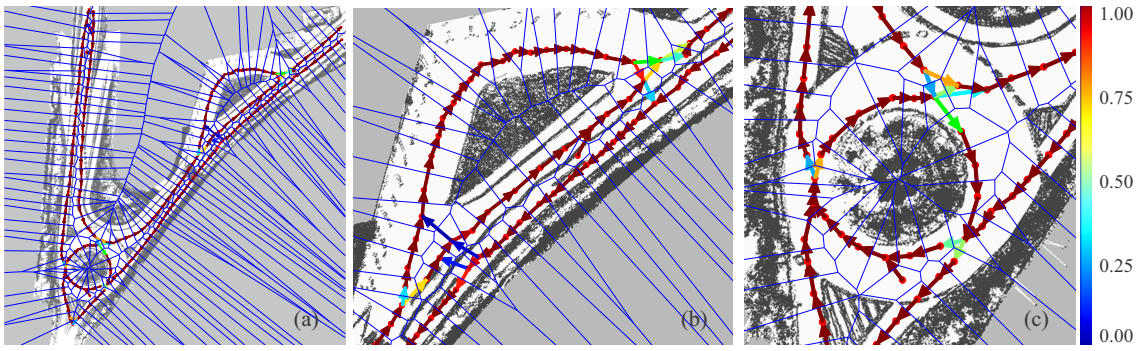


Figure 4.8 Probabilistic traffic rules inference result: (a) shows the graph representation of the probabilistic traffic rules; (b) and (c) are the highlights of the traffic rules learning results in the TJ and RA areas.

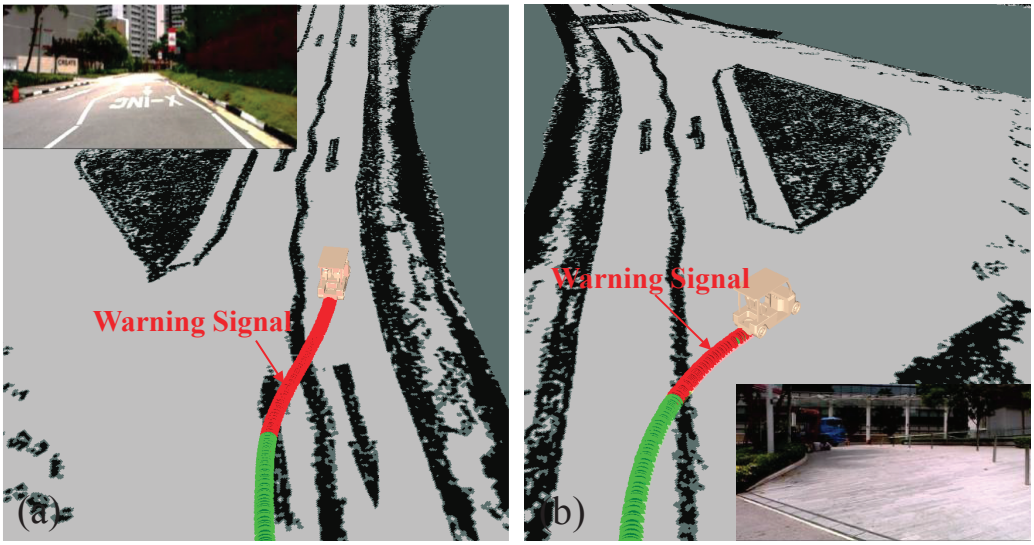


Figure 4.9 Wrong-way driving detection. The red markers indicates the positions where the wrong-way driving warning is triggered.

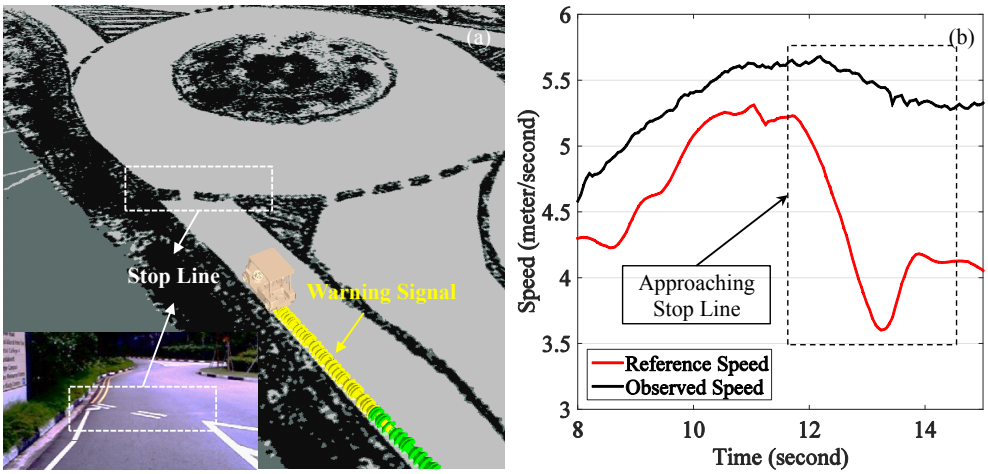


Figure 4.10 Reckless driving detection at round-about. The yellow markers indicate the position where the warning is triggered when the vehicle fails to slow down before the stop line.

future behavior that complies with the road context. Let us now move our focuses to the motion prediction evaluations. In order to examine the ability of our proposed motion prediction algorithm to handle different and complex road scenarios, such as *Enter RA*, *Exit RA*, *Enter TJ* and *Exit TJ*, we recollected a data set consisting of twelve trajectories performed by two drivers, which were utilized as the ground-truth for the prediction evaluation.

For the purpose of evaluating the improvements that can be achieved by introducing the road context for motion prediction, we simultaneously applied the proposed RuleGPPF and the state-of-the-art GPPF in [116] for each single scenario. Because the particle filtering is a probabilistic approach, and each run in the same setting will give slightly different results, the two algorithms with 500 particles were evaluated over ten trials for each collected trajectory.

Four snapshots of the prediction results are demonstrated in Figure 4.12, where the particles denote the predicted vehicle positions up to ten seconds. The particles are clustered into different groups and labeled as either blue or red to represent the vehicle's driving direction hypotheses. The orange marker is the position where the motion prediction starts, and the green marker indicates the vehicle's ground-truth position. As demonstrated, the RuleGPPF can maintain a less divergent particle set compared to that of GPPF. The quantitative comparison of the motion prediction accuracy is demonstrated in Figure 4.13 and Table 4.2. Obviously, the prediction accuracy of the RuleGPPF is significantly higher than that of the GPPF when the prediction is up to five seconds. This is because the GPPF particles can be easily diverged by the infeasible motion dynamics generated by the Gaussian Process Regression. Thanks to the inferred road context, the RuleGPPF, on the other hand, can stay very robust to those outliers. As such, the RuleGPPF can maintain a reasonable prediction error around two meters for different scenarios.

Looking deep into the motion prediction error plotted in Figure 4.13, the prediction error of the RuleGPPF will always grow a bit when the intersection is approaching. This is within our expectation, because the predicted vehicle position will follow a multi-modal distribution at the intersections. The exact direction toward which the vehicle wants to

Table 4.2 RMSE of the Motion Prediction (meters)

Prediction Horizon	Algorithm	Enter RA	Exit RA	Enter TJ	Exit TJ
5 seconds	GPPF	2.812	7.456	3.202	2.713
	RuleGPPF	1.743	3.782	1.506	2.779
10 seconds	GPPF	14.303	11.449	6.104	3.815
	RuleGPPF	1.153	2.314	0.479	0.857

move becomes ambiguous, thus the prediction error will increase accordingly. Once the predicted vehicle position passes the intersection, the particles will be divided into separate clusters to model the hypothesis of different possible moving directions. For each moving direction, the distribution of the predicted vehicle position becomes unimodal, and the prediction error can be reduced accordingly.

4.7 Summary

This chapter has discussed a probabilistic approach for the road context inference. The learning process was divided into multiple stages to extract the road topology and vehicle motion pattern, which were then coupled to infer the traffic rules that dominate the region transitions. The case study of an urban road that includes a round-about and a T-junction has shown the proper functionality of the proposed algorithm. The applicability of the inferred road context has been properly explored as well, and some promising results have been achieved.

The inferred road context, together with the vehicle behavior analysis results, will be employed for urban road situation modeling in the next chapter, and a situation-aware decision making algorithm will be discussed in detail.

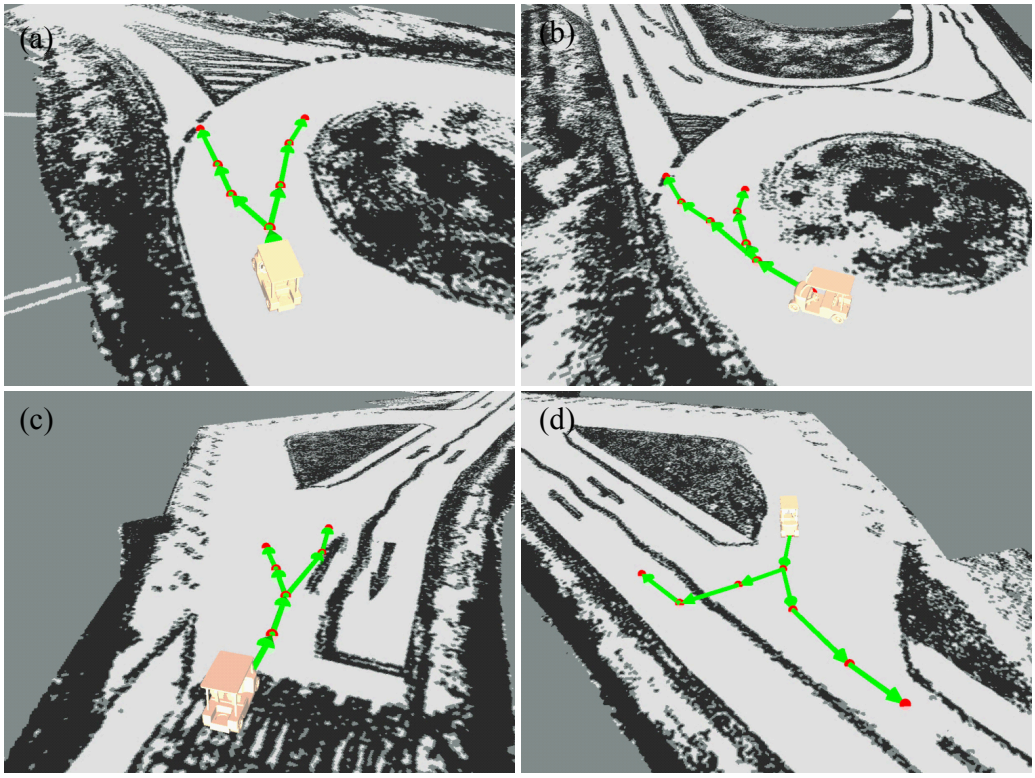


Figure 4.11 Area prediction at four regions: the green edges are the predicted topological path.

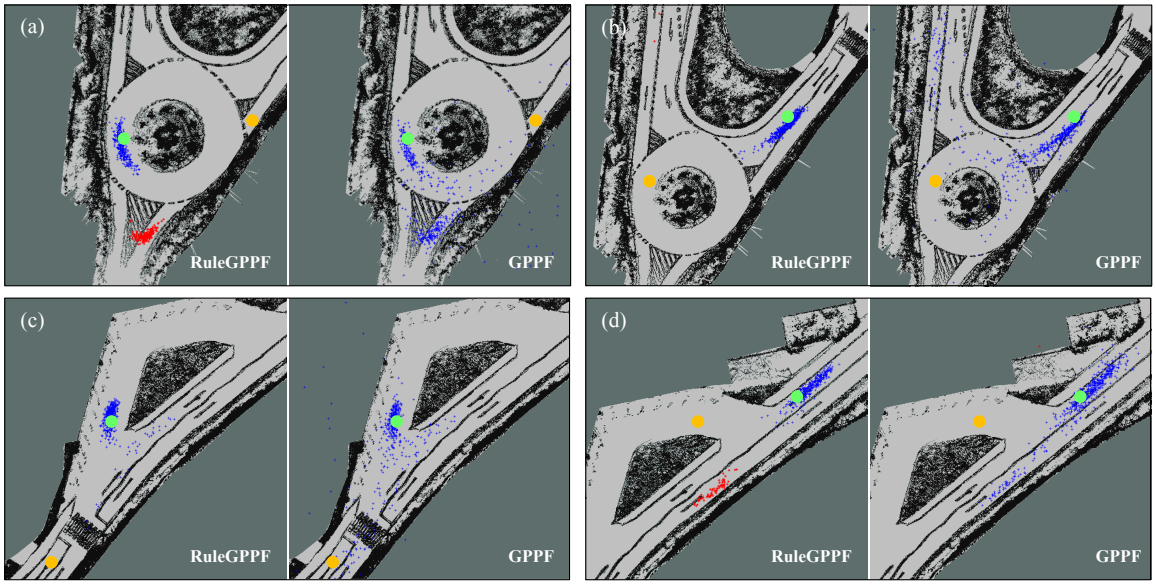
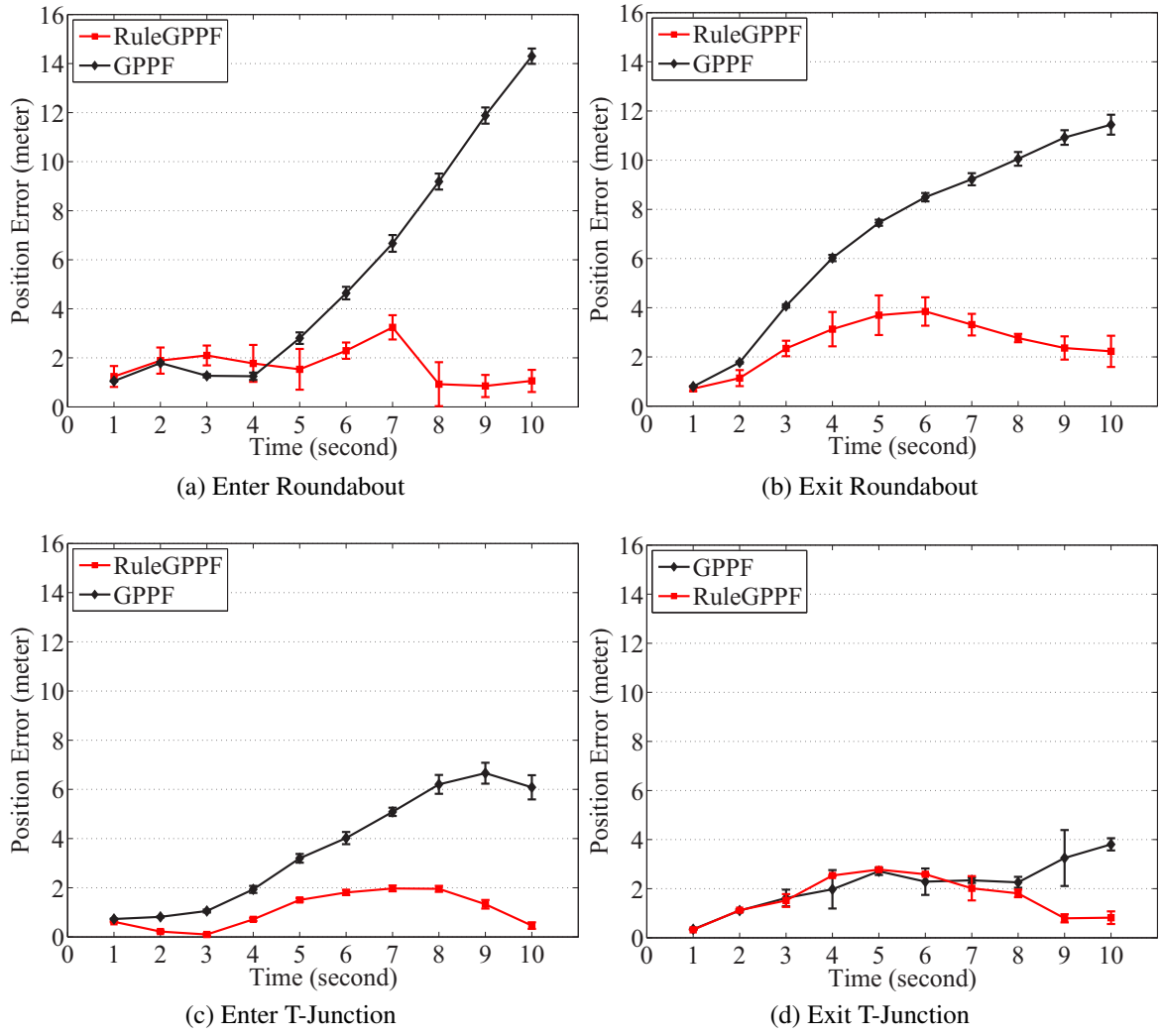


Figure 4.12 Motion prediction results up to 10 seconds: The particles represent the predicted vehicle positions, which are labeled as either blue or red to represent the vehicle's driving direction hypotheses. The green marker denotes the vehicle's ground-truth position and the orange marker is the position where the motion prediction starts.



(a) Enter Roundabout

(b) Exit Roundabout

(c) Enter T-Junction

(d) Exit T-Junction

Figure 4.13 Motion prediction error of 10 trials for each scenario.

Chapter 5

Situation-aware Decision Making under Situation Uncertainty using POMDP

5.1 Introduction

Given the road context inference and vehicle behavior analysis results presented in the last chapter, this chapter proposes a situation-aware decision making algorithm for autonomous driving in the urban road environment. The autonomous vehicle decision making module is in charge of properly maneuvering the autonomous vehicle's own behavior and negotiating with the other traffic participants. Toward this end, the full awareness of the surrounding environment is necessary, but difficult to achieve, where the difficulty is always aggravated by both the lack of proper environment representation and the imperfect vehicle perception system [103].

To begin with, an *urban road situation* model, which is defined as the integration of the road context and the obstacle vehicle's motion intention, is proposed for proper environment representation. Thereafter, the situation-aware decision making problem is modeled as a POMDP to handle the uncertainties introduced by the situation evolution reasoning. For the sake of real-time application, the online POMDP solver DESPOT [18] is employed for its efficiency to account for the large observation space. The proposed algorithm has been extensively validated, and is general enough to account for various urban road driv-

ing scenarios, such as leader following on single-lane road and traffic negotiation at the T-junction or roundabout.

This chapter is organized as follows. The situation-aware decision making problem is stated in Section 5.2, and the environment representation is discussed in Section 5.3. In Section 5.4, the POMDP model for situation-aware decision making is presented. The evaluation results are shown in Section 5.5, and this chapter is concluded in Section 5.6.

5.2 Problem Statement

Let $\mathcal{A}$ denote a discrete vehicle action set, and each $a \in \mathcal{A}$ is referred to as a vehicle action, which can include acceleration, deceleration, and so forth. In order to address the situation awareness, let $s \in \mathcal{S}$ be the situation state, where $\mathcal{S}$ denotes the situation state space. The definition of the situation state is driven by the environment modeling, which has to encapsulate enough information for the proper situation evolution reasoning.

However, the situation state usually cannot be fully observed due to the lack of a "situation sensor" and the uncertainties arising from the vehicle perception module. A situation reasoning function $\Theta(s, z) = \Pr(s|z) : \mathcal{S} \times \mathcal{Z}$ thereby is introduced to infer the situation state in a probabilistic manner, where $z \in \mathcal{Z}$ represents the observation provided by the vehicle perception module.

Given the observation and the situation reasoning function, this study seeks to address the situation-aware decision making as an optimization problem,

$$\pi^*(z) = \arg\max_{\pi} \mathbb{E}\{\sum_{t=0}^{\infty} R(a_t, s_t)\Pr(s_t|z_t)\}, \tag{5.1}$$

where $R(a, s) : \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ represents the reward function that we aim to optimize. The optimal policy $\pi^* : \mathcal{A} \leftarrow \mathcal{S}$ specifies an optimal action $a \in \mathcal{A}$ for any situation state $s \in \mathcal{S}$ in each time step, such that a maximum expected reward $\mathbb{E}$ can be returned.

5.3 Urban Road Situation Modeling

In the urban environment, an autonomous vehicle's driving decisions are affected by both the road context and the other vehicles' driving behaviors. (In the sequel, we refer to the other vehicles as obstacle vehicles.) The knowledge of the road context can help to provide a more comprehensive understanding of the environment, such as the road topology, typical vehicle motion pattens and so forth. Similarly, understanding the semantic meaning of the obstacle vehicles' behaviors, i.e., motion intention, can contribute to a more robust prediction of their future motions, which in turn can be employed for a more accurate risk evaluation. Therefore, a loose definition of the *Urban Road Situation* $\mathcal{URS}$ is proposed as *the integration of the road context* $\mathcal{RC}$ *and the obstacle vehicle's motion intention* $\mathcal{I}$, namely $\mathcal{URS} = \{\mathcal{RC}, \mathcal{I}, \mathcal{RC} \bigoplus \mathcal{I}\}$, where the operator $\bigoplus$ indicates the correlation between the road context and the obstacle vehicle's motion intention. As such, the urban road situation not only features the motion intention and the road context, but also considers implicit correlations between them.

5.3.1 Road Context

In the last chapter, the road context is inferred in a probabilistic manner using an amount of randomly observed vehicle behavior data, where the resulting road context $\mathcal{RC}$ is given in a tuple $\mathcal{RC} = \{\mathbf{E}, \mathbf{R}, \mathbf{MP}\}$. Given an observed vehicle state s_o, the corresponding road context information $rc(s_o) = \{e(s_o), \mathbf{R}(s_o), \mathrm{v}_{\mathrm{ref}}(s_o)\}$ can be mapped out as,

$$\begin{aligned} e(s_o) &= \{e \in \mathbf{E} | s_o.p \in e\}, \\ \mathbf{R}(s_o) &= \mathbf{R}_{e(s_o), \forall e' \in \mathbf{E}}, \\ \mathrm{v}_{\mathrm{ref}}(s_o) &= |\bar{v}_{\mathrm{ref}}(s_o)|, v_{\mathrm{ref}}(s_o) \sim \mathbf{MP}(s_o.p), \end{aligned} \tag{5.2}$$

where $s_o.p \in \mathbb{R}^2$ denotes the vehicle position and $e(s_o)$ is the region where the vehicle is located. The $\mathbf{R}(s_o) = \mathbf{R}_{e(s_o), \forall e' \in \mathbf{E}}$ denotes the transition probability vector that corresponds to the region $e(s_o)$, and the speed $\mathrm{v}_{\mathrm{ref}}(s_o)$ is generated from the vehicle motion pattern $\mathbf{MP}$.

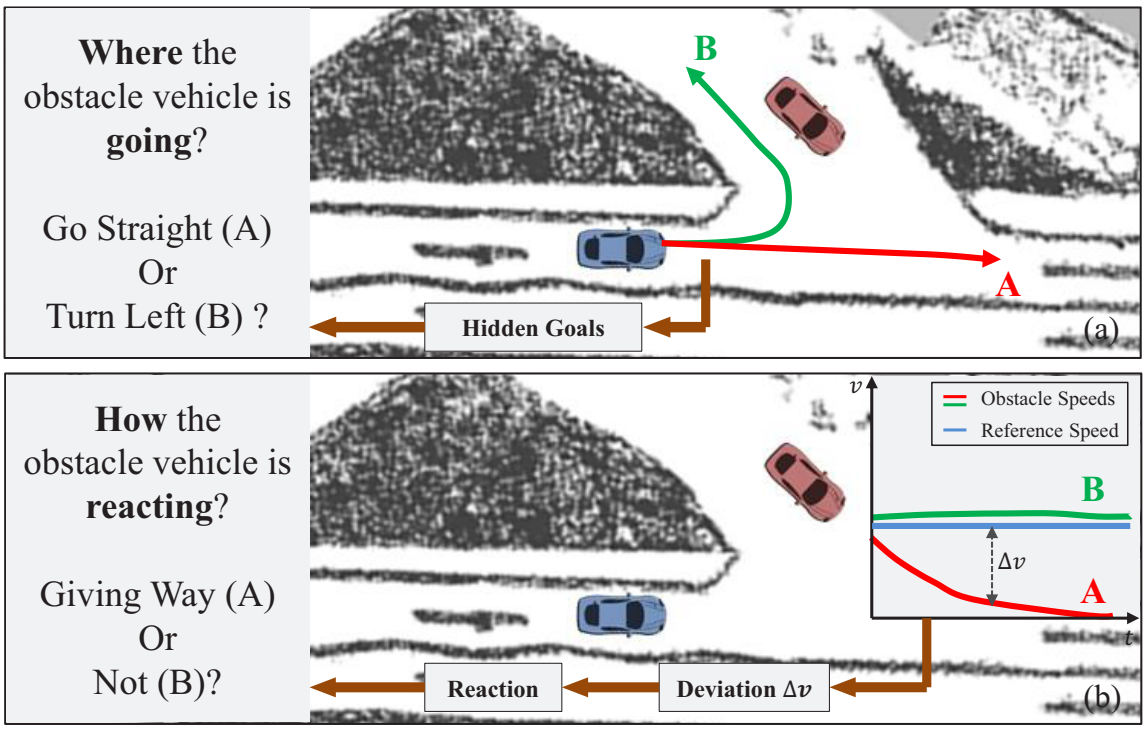


Figure 5.1 Motion intention explanation: (a) Hidden goal-based motion intention, where A and B denote two optional routes for the obstacle vehicle (blue). (b) Reaction-based motion intention (using speed deviation), where A and B denote two possible obstacle vehicle speed profiles, which are checked against the reference speed for motion intention inference.

The road context $\mathcal{RC}$ essentially is a generalization of the vehicles' behaviors derived by abstracting the vehicle behavior consistencies, thus it can properly represent the normal vehicle behavior that should be followed. Specifically, the road context information $rc(s_o)$ not only regulates the directions toward which the observed vehicle is allowed to move, but also provides the reference speed $\mathrm{v_{ref}}$ that the observed vehicle should follow. Therefore, the $rc(s_o)$ can be defined as the *reference vehicle state* that corresponds to the observed vehicle state s_o.

5.3.2 Motion Intention

The most popular strategy of vehicle motion intention modeling is to define the motion intention as their hidden destinations in Figure 5.1(a). Namely, the obstacle vehicle's final destination is not fully observable and there always exists a motion model $m \in \mathbf{M}$ that dominates the vehicle's future motion for each hidden goal $g \in \mathbf{G}$, where $\mathbf{M}$ and

$\mathbf{G}$ denote the motion model set and hidden goal set respectively. More specifically, let $\mathcal{ROI}s$ be the *Regions Of Interest* where the autonomous vehicle is primarily concerned about the obstacle vehicles' future motion. Without losing generality, assume that the autonomous vehicle is operating in an environment that includes M $\mathcal{ROI}s$, which can include T-junctions, roundabouts, and so on. For each $\mathcal{ROI}^{[i]}, i \in \{1, ..., M\}$, there exists a goal set $\mathbf{G}^{[i]} \leftarrow \mathcal{ROI}^{[i]}, i \in \{1, ..., M\}$ that defines the possible directions where the vehicles can move. Therefore, the size of the required motion models can be given as $|\mathbf{M}| = \sum_{i=1}^{M} |\mathbf{G}^{[i]}|$, where $|\mathbf{G}^{[i]}|$ denote the size of the goal set $\mathbf{G}^{[i]}, i \in \{1, ..., M\}$. Given $M \rightarrow \infty$, the size of $\mathbf{M}$ becomes unbounded, which implies that using the hidden goal to model the motion intention may suffer from the scalability problem as the number of $\mathcal{ROI}s$ increases.

Instead of relying on the hidden goals, this study aims at employing the obstacle vehicle's *reaction* to model the motion intention as in Figure 5.1(b). The most intuitive method of reaction modeling might be measuring the temporal evolutions of the vehicle state, such as the route changing, speed variance, etc. To properly quantify these changes, however, is complicated due to the diversity of vehicle behaviors. In our study, the *reaction* φ is proposed as the *deviation* of the observed vehicle state s_o from the corresponding reference vehicle state $rc(s_o)$, i.e., $\varphi \leftarrow s_o \bigotimes rc(s_o)$, where the operator $\bigotimes$ measures the deviations. This proposal is inspired by the fact that the reactive driving behaviors, in most cases, are different from the normal driving behaviors that are captured by the road context.

Given the deviation measurement, the reaction-based motion intention is proposed as $\iota \sim \Pr(\iota | s_o \bigotimes rc(s_o)), \iota \in \mathcal{I}$. As such, the motion intention is abstracted as the conditional distribution over the reactions, where the implicit correlations between the road context and motion intention are acknowledged. Compared to the hidden goal method, employing the reactions to model the motion intention is more general and scalable, because the measurement of deviation is not restricted to any specific $\mathcal{ROI}$. Therefore, we can always find a bounded-size reaction-based motion intention set $\mathcal{I}$ to model the obstacle vehicle behaviors.

In this study, the vehicle speed deviation is utilized, where the observed vehicle speed is checked against the reference speed (featured by the road context) for motion intention inference. The resulting motion intention set with four hypotheses is designed as,

$$\mathcal{I} = \{Stopping, Hesitating, Normal, Aggressive\},$$

where their meanings and the corresponding speed deviations are defined as follows:

- The *Stopping* intention indicates that the obstacle vehicle plans to perform a stop for temporal parking or giving way to approaching vehicles. As a feature, the obstacle vehicle's speed is close to zero, while the reference speed is much higher.
- The *Hesitating* intention indicates that the obstacle vehicle is hesitating in making driving decisions. The speed deviation is given as the obstacle vehicle speed being slower than the reference speed but not close to zero. This is inspired by human driving activity, where slower speeds are usually preferable when the current driving decisions are not confident.
- The *Normal* intention indicates that the obstacle vehicle will maintain its current normal behavior. As an evidence, the obstacle vehicle's speed matches well with the reference speed, namely the speed deviation is small.
- The *Aggressive* intention indicates that the obstacle vehicle may aggressively accelerate and has no sense for negotiation. As an indicator, the obstacle vehicle's current speed is much higher than the reference speed.

Admittedly, these motion intention definitions may be questionable at first sight, but they are abstracted from human driving behaviors and their functionalities have been validated by our extensive evaluations.

5.4 Situation-aware Decision Making using POMDP

As a principled general framework for acting and planning in a partially observable environment, the POMDP is adopted for situation-aware decision making in this study. This

equips the autonomous vehicle with the ability to estimate and evaluate the outcome of driving decisions, even when the situation cannot be exactly observed.

5.4.1 POMDP Model for Situation-aware Decision Making

A POMDP model is formally a tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{Z}, T, O, R, \gamma\}$, where a detailed introduction can be found in Appendix B. This section details how the situation-aware decision making problem can be fitted and solved as a POMDP by elaborating on each component of the contextualized POMDP tuple.

A. State Space $\mathcal{S}$

Due to the Markov property, the state space $\mathcal{S}$ must hold sufficient information for either decision making or belief update [103], which in this case comprises the vehicle pose $[\mathrm{x}, \mathrm{y}, \theta] \in \mathbb{R}^2 \times \mathbb{S}$ and vehicle speed $\mathrm{v} \in \mathbb{R}$ for all the vehicles involved. For the obstacle vehicles, the motion intention $\iota \in \mathcal{I}$ also needs to be covered by the obstacle vehicle state for proper state transition modeling. The road context $\mathcal{RC}$, on the other hand, can be employed as a reference knowledge, and is consequently excluded from the vehicle state.

Formally speaking, the joint state $s \in \mathcal{S}$ can be given as,

$$s = [s_e, s_1, s_2, ..., s_K]^{\mathrm{T}}, \tag{5.3}$$

where s is composed by the ego vehicle state s_e and the obstacle vehicles' states $s_i, i \in \{1, 2, ..., K\}$, and K is the number of the obstacle vehicles involved. Let the vehicle metric state $x \in \mathbb{R}^3 \times \mathbb{S}$ be defined as $x = [\mathrm{x}, \mathrm{y}, \theta, \mathrm{v}]^{\mathrm{T}}$, which includes the vehicle pose and speed. The ego vehicle state can then be equivalently defined as $s_e = x_e$. Similarly, the obstacle vehicle state s_i is given as $s_i = [x_i, \iota_i]^{\mathrm{T}}$, where the vehicle motion intention ι_i is explicitly modeled. Given the recent research advances in vehicle detection and tracking [114], we assume that the vehicle metric state x can be fully observed, however the motion intention is only partially observable.

B. Action Space $\mathcal{A}$

In our autonomous vehicle navigation system, the decision making module is responsible for planning some tactical maneuvers, such as traffic negotiation at a T-junction or roundabout, by maintaining a safe distance with other obstacle vehicles while following a reference route. Therefore the action space $\mathcal{A}$ can be covered by a discrete action set consisting of acceleration, deceleration or maintaining current speed as,

$$\mathcal{A} = [Acc., Dec., Cur.]^{\mathrm{T}}. \tag{5.4}$$

As such, the defined actions are in charge of the vehicle speed control. The steering control of the ego vehicle, on the other hand, is accomplished by tracking the reference routes closely.

C. Observation Space $\mathcal{Z}$

Similar to the joint state $s \in \mathcal{S}$, the joint observation $z \in \mathcal{Z}$ consists of the following elements,

$$z = [z_e, z_1, z_2, ..., z_K]^{\mathrm{T}}, \tag{5.5}$$

where z_e is the ego vehicle's observation and z_i denote the observation of obstacle vehicle i. To properly update the obstacle vehicle's motion intention belief, each vehicle's observation consists of its pose and speed. Since the vehicle metric state x can be properly observed, we can simply generate the observations with a one-to-one mapping directly from the corresponding metric states.

D. Transition Model $T(s, a, s')$

The transition function describes the stochastic system dynamics driven by both the action applied to the ego vehicle and the obstacle vehicles' motion intention. The overall structure of the transition model is illustrated by the Bayesian Network in Figure 5.2.

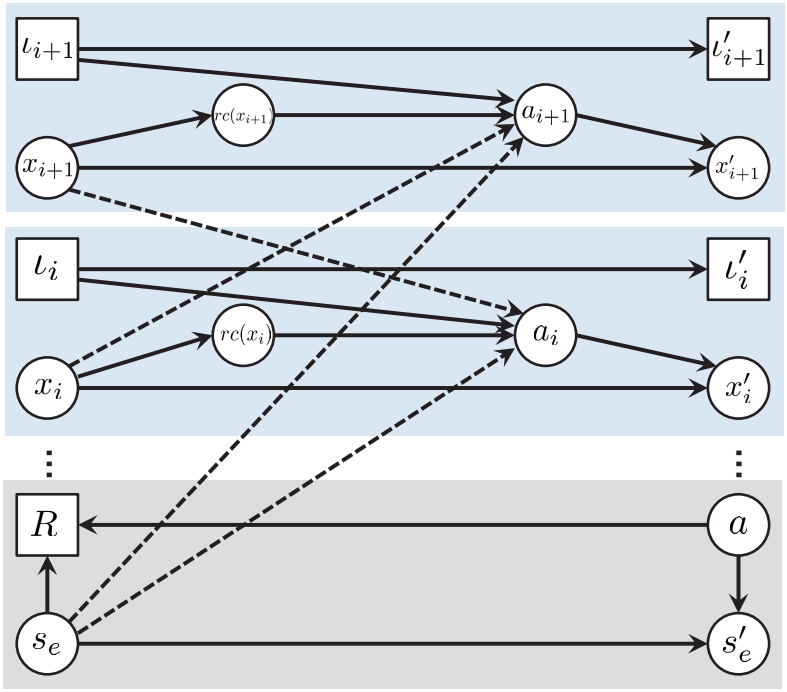


Figure 5.2 Transition model for situation-aware decision making. The dashed lines are to model the vehicle correlations.

Formally speaking, the transition model can be represented by the probabilistic transition as,

$$\Pr(s'|s,a) = \Pr(s'_e|s_e,a)\prod_{i=1}^{K}\Pr(s'_i|s), \tag{5.6}$$

where $a \in \mathcal{A}$ is the action applied to the ego vehicle and s' is the new joint state. For the ego vehicle, the state transition $\Pr(s'_e|s_e,a)$ is dominated by the applied action a and the turning angle $\Delta\theta$ only, which can be given in detail as,

$$\begin{bmatrix} \mathrm{x}'_e \\ \mathrm{y}'_e \\ \theta'_e \\ \mathrm{v}'_e \end{bmatrix} = \begin{bmatrix} \mathrm{x}_e \\ \mathrm{y}_e \\ \theta_e \\ \mathrm{v}_e \end{bmatrix} + \begin{bmatrix} (\mathrm{v}_e + a\Delta t)\Delta t\cos(\theta+\Delta\theta) \\ (\mathrm{v}_e + a\Delta t)\Delta t\sin(\theta+\Delta\theta) \\ \Delta\theta \\ a\Delta t \end{bmatrix}, \tag{5.7}$$

where the turning angle $\Delta\theta$ is achieved by tracking the reference route.

For the obstacle vehicles, the state transition model $\Pr(s'_i|s)$ is more complicated, as the obstacle vehicle's motion intention is not observable and the dependencies within the vehicles need to be carefully accounted for. Recalling our discussion in Section 5.3, the

vehicles' behavior is driven by both the road context and their motion intentions, therefore an obstacle vehicle's state transition can be factorized as,

$$\begin{aligned}\Pr(s_i'|s) &= \Pr(x_i'|s)\Pr(\iota_i'|\iota_i) \\ &= \sum_{a_i} \Pr(x_i'|x_i, a_i)\Pr(a_i|s)\Pr(\iota_i'|\iota_i),\end{aligned} \tag{5.8}$$

where a_i is the action applied to obstacle vehicle i. Unlike the ego vehicle action $a \in \mathcal{A}$, the obstacle vehicle action a_i is defined as $a_i = [a_{vi}, \Delta\theta_i]^{\mathrm{T}} \in [Acc., Dec., Cur.] \times \mathbb{S}$, which includes both the speed action a_{vi} and turning angle $\Delta\theta_i$. Given the obstacle vehicle action a_i, the obstacle vehicle's metric state transition $\Pr(x_i'|x_i, a_i)$ is identical to that of the ego vehicle in Equation (5.7). Regarding the motion intention transition $\Pr(\iota_i'|\iota_i)$, the motion intention remains unchanged during the state transition process, which will be updated accordingly when a new observation is available.

The difficulty then lies in the inference of the obstacle vehicle action a_i. Let $\bar{X}_i = \bigcup_{j \neq i} x_j, \forall j \in \{e, 1, ..., K\}$ represent all the vehicles' metric states excluding the one of vehicle i. The obstacle vehicle action distribution $\Pr(a_i|s)$ in Equation (5.8), thereafter, can be reformulated as,

$$\begin{aligned}\Pr(a_i|s) &= \Pr(a_{vi}, \Delta\theta_i|\bar{X}_i, x_i, \iota_i) \\ &= \sum_{rc(x_i)} \Pr(a_{vi}, \Delta\theta_i|\bar{X}_i, x_i, \iota_i, rc(x_i))\Pr(rc(x_i)|x_i) \\ &= \sum_{rc(x_i)} \Pr(\Delta\theta_i|rc(x_i))\Pr(a_{vi}|\bar{X}_i, x_i, \iota_i)\Pr(rc(x_i)|x_i),\end{aligned} \tag{5.9}$$

where $rc(x_i) = \{e(x_i), \mathbf{R}(x_i), \mathrm{v}_{\mathrm{ref}}(x_i)\}$ is the reference vehicle state sampled from Equation (5.2). Since the obstacle vehicle's moving direction is not explicitly modeled by the motion intention, its turning angle $\Delta\theta_i$ is therefore controlled by the road context $rc(x_i)$ only as $\Pr(\Delta\theta_i|rc(x_i))$. More specifically, given the current region $e(x_i)$ inside which the obstacle vehicle i is located, we can sample the next possible region $e(x_i)' \in \mathbf{E}$ where the obstacle vehicle might locate according to $\mathbf{R}(e(x_i)) = \Pr(e(x_i)'|e(x_i), \rho)$. There-

Table 5.1 Obstacle Vehicle Speed Action Corresponding to Motion Intention

$\iota \in \mathcal{I}$	$Stopping$	$Hesitating$	$Normal$	$Aggressive$
a_v	$Dec.$	$Dec./Cur./Acc.$[a]	$Cur.$	$Acc./Cur.$[a]

a: Each action has equal probability of being selected.

after, the turning angle is calculated as $\Delta\theta_i = \mathcal{Q}(e(x_i), e(x_i)') - \theta_i$, where the function $\mathcal{Q}(e(x_i), e(x_i)')$ calculates the orientation angle from region $e(x_i)$ to region $e(x_i)'$.

As formulated in Equation (5.9), the speed action a_{vi} depends on both the motion intention and the correlations with the other vehicles as $\Pr(a_{vi}|\bar{X}_i, x_i, \iota_i)$. Toward this end, the dependency within the vehicles is simply modeled as *each vehicle has to avoid a potential collision with any other vehicles*. Therefore, if the obstacle vehicle i has an extremely low collision risk with other vehicles, the speed action is mapped from the motion intention only as $\Pr(a_{vi}|\iota_i)$. In accordance with the motion intention definitions in Section 5.3.2, the speed action corresponding to each motion intention is given in Table 5.1. Once the collision risk is high enough, the obstacle vehicle would either follow the inferred motion intention or choose the $Dec.$ action. The probability of selecting $Dec.$ action is given as $\Pr(a_{vi} = Dec.|\bar{X}_i, x_i) = \varepsilon \mathrm{v}_{nearest}/\mathrm{v}_{max}$, where $\varepsilon \in (0, 1]$ is a scaling factor. The $\mathrm{v}_{nearest}$ denotes the nearest vehicle's speed and v_{max} is the maximum speed allowed. That is to say, the higher the speed of the nearest vehicle, the higher the probability of vehicle i would decide to decelerate.

E. Observation Model $O(z, s', a)$

The observation model aims at simulating the measurement process, which can be employed to update the motion intention belief. For each vehicle, the observation is conditionally independent of other variables given the corresponding vehicle state, so the observation model can be factorized as,

$$\Pr(z|s', a) = \Pr(z_e|s'_e) \prod_{i=1}^{K} \Pr(z_i|s'_i). \tag{5.10}$$

Since the vehicle metric states are considered as fully observable, the observation function of the ego vehicle is given as $\Pr(z_e|s'_e) = 1$ iff $z_e = s'_e$, otherwise $\Pr(z_e|s'_e) = 0$. The

Table 5.2 Mean Function Corresponding to Motion Intention

$\iota \in \mathcal{I}$	$Stopping$	$Hesitating$	$Normal$	$Aggressive$
$\mathbf{m}(\mathrm{v}_{\mathrm{ref}}, \iota)$	0.0	$0.5\mathrm{v}_{\mathrm{ref}}$	$\mathrm{v}_{\mathrm{ref}}$	$1.5\mathrm{v}_{\mathrm{ref}}$

obstacle vehicles' observation functions are more complicated due to the inclusion of their motion intentions, which are reformulated as,

$$\Pr(z_i|s'_i) = \Pr(z_i|x'_i, \iota'_i) = \Pr(z_i|\iota'_i)\Pr(z_i|x'_i)/\Pr(z_i), \tag{5.11}$$

where the metric state observation function $\Pr(z_i|x'_i)$ follows the same distribution as that of the ego vehicle. The distribution $\Pr(z_i|\iota'_i)$ models the probability of observing z_i given the motion intention is ι'_i, and can be given as,

$$\Pr(z_i|\iota'_i) = \mathcal{N}(z_{vi}|\mathbf{m}(\mathrm{v}_{\mathrm{ref}}(z_i), \iota'_i), \mathrm{v}_{\mathrm{ref}}(z_i)/\sigma)), \tag{5.12}$$

where z_{vi} is the observed speed of vehicle i and $\mathrm{v}_{\mathrm{ref}}(z_i)$ denotes the reference speed that corresponds to observation z_i. In other words, given the reference speed $\mathrm{v}_{\mathrm{ref}}(z_i)$, the observation function for each motion intention $\iota \in \mathcal{I}$ is modeled as a Gaussian with mean $\mathbf{m}(\mathrm{v}_{\mathrm{ref}}(z_i), \iota)$ and covariance $\mathrm{v}_{\mathrm{ref}}(z_i)/\sigma$. The mean value $\mathbf{m}(\mathrm{v}_{\mathrm{ref}}(z_i), \iota)$ is a function of the reference speed and the motion intention type as shown in Table 5.2, which explicitly models the relationship between the motion intention and the speed deviation. The scaling factor σ in the covariance function is used to adjust the motion intention confidence. Together with the transition function in Equation (5.8), the update of the obstacle vehicle's motion intention belief thereby becomes straightforward.

F. Reward Function R

The main objective of the ego vehicle is to arrive at the target destination as quickly as possible, while avoiding collision with other obstacle vehicles. The reward function $R(s, a)$ is thus defined in a multi-objective manner to properly balance the driving efficiency and safety,

$$R(s, a) = R_{goal}(s, a) + R_{crash}(s, a) + R_{action}(a) + R_{vel}(s), \tag{5.13}$$

where $R_{goal}(s, a)$ denotes the reward when the ego vehicle reaches the final destination and $R_{crash}(s, a)$ assigns a high penalty if the ego vehicle is in collision. $R_{action}(a)$ provides a small penalty if the $a \in [Acc., Dec.]^{\mathrm{T}}$, which is to avoid the frequent speed change and to improve driving comfort. The speed reward $R_{vel}(s) = k\mathrm{v}/\mathrm{v}_{max}$ is to encourage high speed travel and improve the driving efficiency, where v is the ego vehicle's speed and k is a scaling factor.

5.4.2 POMDP Solver

Given the POMDP model for situation-aware decision making as described in the previous sections, an efficient POMDP solver must then be employed for real-time applications.

The general POMDP is usually solved in an offline manner, where the optimal policy is computed offline and the agent conducts the policy look-up for online execution. Since the algorithm aims to return a policy that prescribes an action for every possible belief state, this strategy has the risk of becoming intractable when the POMDP model is scaled up. In contrast, the online POMDP algorithm searches for a good policy given the agent's current belief state $b_0 \in \mathcal{B}$, then considers only a small set of belief states that are achievable from that current belief state, thereby greatly improving the tractability [117].

In our application, the online POMDP solver DESPOT is employed for its efficiency in handling a large observation space. Rather than searching the whole belief tree, DESPOT only samples a *scenario* set with constant size Q. As a consequence, the belief tree of height H contains only $O(|\mathcal{A}|^H Q)$ nodes, which is not correlated to the size of the observation space. This can greatly alleviate our observation space constraints. The belief state is represented by the random particles within DESPOT, and for each particle, the obstacle vehicles' motion intentions are randomly sampled for a belief representation. Implementation-wise, since the belief state is represented by the particles, it is straightforward to implement both the state transition in Equation (5.6)-(5.8) and the observation update in Equation (5.11).

5.5 Evaluation

In this section, the proposed algorithm will be evaluated to validate its functionality and generality. These evaluations are carried out on the urban roads within the National University of Singapore as shown in Figure 4.4. Given the environment and the corresponding road context, the *Stage* simulator [118] is employed for simulation evaluation, where Gaussian noise is purposely imposed on the vehicle pose and speed measurement. Thereafter, real-time experiments using two vehicles, including one autonomous vehicle and one manually driven vehicle, are conducted to validate the applicability of the proposed algorithm on the real urban road.

5.5.1 Simulations

The proposed situation-aware decision making algorithm has been successfully applied to various urban road driving scenarios within the simulation stage, such as leader following and negotiation with obstacle vehicles at a T-junction or roundabout. For the sake of generality evaluation, the designed POMDP model, especially the reward function, remained unchanged for all the evaluations. The demonstration of the complete simulation results together with the parameters is available in the video at http://youtu.be/W37haHhfU34. In this context, we want to highlight the ability of our algorithm to handle the decision making problem at the T-junction and roundabout, which is widely recognized as challenging.

As the autonomous vehicle was conducting the lane merging behavior at the T-junction shown in Figure 5.3(a), it was essential to properly detect the road context and reason about the obstacle vehicle's motion intention to guarantee driving safety. As illustrated in Figure 5.3(b), the *Dec.* action was triggered first to avoid the potential collision with the approaching vehicle that was associated with a high belief in maintaining its current $Normal$ behavior. After some time, the obstacle vehicle, however, changed its mind and decided to give way to the ego vehicle as in Figure 5.3(c), where its motion intention belief was updated accordingly. Given the increasing confidence that the obstacle vehicle may want to wait as Figure 5.3(d), the ego vehicle decided to accelerate and cautiously merged into the desired lane as Figure 5.3(e). The purpose of this scenario is to evaluate

the ability of our proposed algorithm to handle complicated interactions with the other vehicles. Rather than waiting forever, the ego vehicle is able to properly reason the obstacle vehicle's intention and react in an efficient manner.

The extension of the proposed algorithm to handle multiple obstacle vehicles is shown in Figure 5.4. As expected, the ego vehicle decided to decelerate and wait when the leading obstacle vehicle chose not to give way in Figure 5.4(b). After the leading obstacle vehicle passed by, the ego vehicle decided to slowly move forward in Figure 5.4(c), although the following obstacle vehicle was still holding a high $Normal$ belief. Eventually, the following obstacle vehicle performed a stop to avoid the collision with the ego vehicle, where its motion intention belief is updated as Figure 5.4(d). Thereafter, the ego vehicle successfully merged into the gap between the two obstacle vehicles as Figure 5.4(e). While actively cutting into the following obstacle vehicle's route seems dangerous, the ego vehicle's decision is within our expectation. This is because the dependency within the vehicles is well acknowledged by the ego vehicle, namely the ego vehicle knows that there would be a higher chance to make the obstacle vehicles give way if it actively cuts into their route and moves at a relatively higher speed. This is the same way that the human drivers negotiate with each other.

In sight of the improvements achieved, the same model was applied to the roundabout navigation. The evaluation snapshots are shown in Figure 5.5 and Figure 5.6, where the settings are identical to that of the T-junction evaluation. Similar results are achieved, which indicates that the proposed approach can function properly at the roundabout as well.

Acknowledging that the proposed approach is probabilistic in nature, we extensively conducted 100 evaluation trials for each scenario discussed above. For the single obstacle vehicle case, one additional evaluation scenario in which the obstacle vehicle will not purposely give way is implemented. Moreover, we extended the multiple obstacle vehicle case by trying different gaps between the obstacle vehicles. Two evaluation metrics including the *average traveling time for goal reaching* and the *overall failure rate* were measured to validate the efficiency and safety respectively. The failure is defined as either a col-

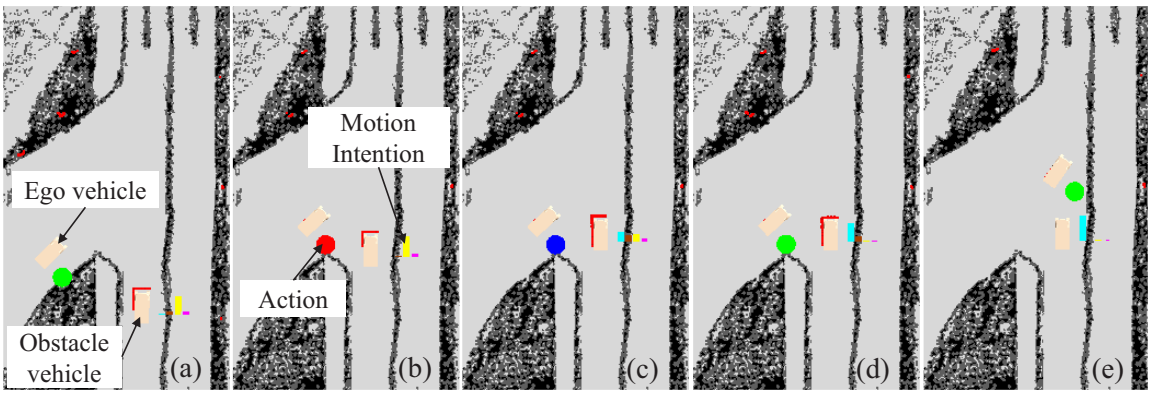


Figure 5.3 Negotiation with a single obstacle vehicle which purposely gives way at the T-junction. The action is represented as the circle marker: [Red:$Dec.$, Green:$Acc.$, Blue:$Cur.$]. The motion intention is represented as the histogram to the right of the obstacle vehicle:[Sky Blue:$Stopping$, Brown:$Hesitating$, Yellow:$Normal$, Purple:$Aggressive$], where the histogram height is proportional to the corresponding belief value.

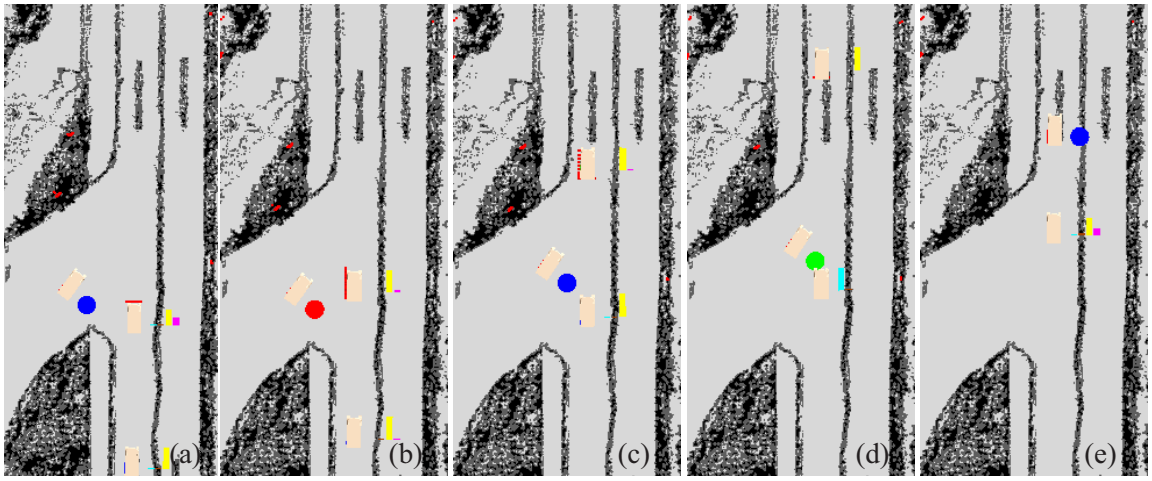


Figure 5.4 Negotiation with multiple obstacle vehicles at the T-junction. The action is represented as the circle marker: [Red:$Dec.$, Green:$Acc.$, Blue:$Cur.$]. The motion intention is represented as the histogram to the right of the obstacle vehicle:[Sky Blue:$Stopping$, Brown:$Hesitating$, Yellow:$Normal$, Purple:$Aggressive$], where the histogram height is proportional to the corresponding belief value.

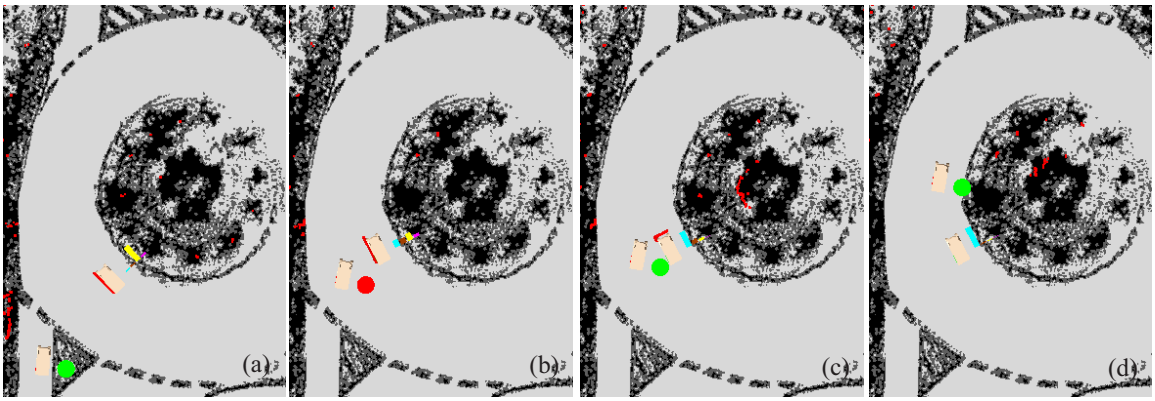


Figure 5.5 Negotiation with a single obstacle vehicle which purposely gives way at the roundabout. The action is represented as the circle marker: [Red:$Dec.$, Green:$Acc.$, Blue:$Cur.$]. The motion intention is represented as the histogram to the right of the obstacle vehicle marker:[Sky Blue:$Stopping$, Brown:$Hesitating$, Yellow:$Normal$, Purple:$Aggressive$], where the histogram height is proportional to the corresponding belief value.

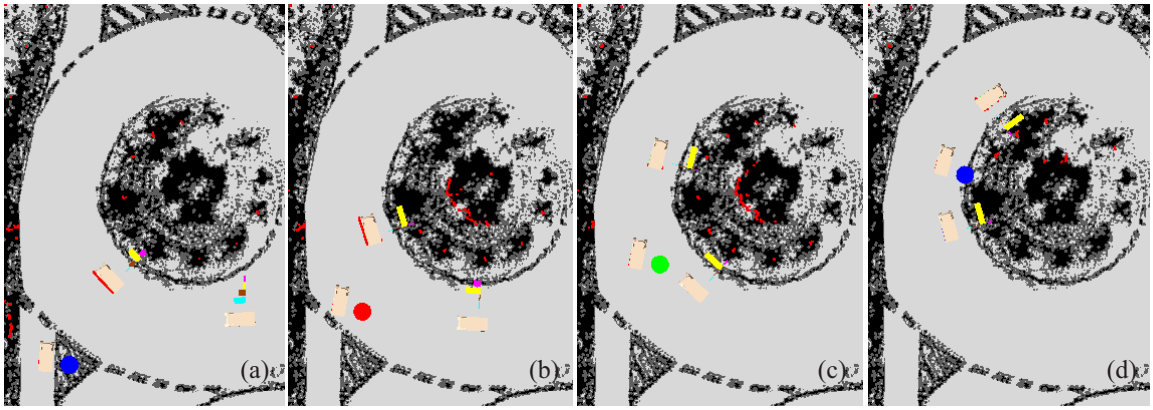


Figure 5.6 Negotiation with multiple obstacle vehicles at the roundabout. The action is represented as the circle marker: [Red:$Dec.$, Green:$Acc.$, Blue:$Cur.$]. The motion intention is represented as the histogram to the right of the obstacle vehicle marker:[Sky Blue:$Stopping$, Brown:$Hesitating$, Yellow:$Normal$, Purple:$Aggressive$], where the histogram height is proportional to the corresponding belief value.

lision happened or the autonomous vehicle failed to complete the navigation task within the permitted time. The corresponding failure rate is given by statistically measuring the frequency with which the failure happens. The traveling time was only measured for the successful trials. For the sake of performance comparison, we also implemented the reactive decision making method in [7], where the ego vehicle iteratively checks the safe region for obstacle clearance measurement and reactively decides whether to go or not.

As depicted in Table 5.3, the reactive approach has a relatively higher failure rate in handling the scenario in which the obstacle vehicle purposely gives way. This is because when the obstacle vehicle decided to give way, the distance between itself and the ego vehicle was smaller than the threshold designed for safe driving, so the ego vehicle just waited forever to bypass the stopped obstacle vehicle. On the other hand, our POMDP-based approach can maintain an impressive success rate for all the scenarios, especially the one in which the obstacle vehicles purposely give way. Rather than getting stuck, the task failures of the POMDP-based approach are because of the collisions contributed by the proactive actions, where the autonomous vehicle actively cut into the obstacle vehicle's route for lane merging. As discussed earlier, these proactive actions are rational and can be well balanced by tuning the model parameters.

By comparing the average traveling time, we can find the ego vehicle always passively decided to wait, even when the gap between itself and the obstacle vehicles is large enough for safe merging. Therefore, the reactive approach takes much more time to navigate through the designed scenarios, especially the one with multiple obstacle vehicles and a large separation distance. On the other hand, the POMDP-based approach can efficiently merge into the gaps between the multiple obstacle vehicles by adopting some proactive actions.

Last but not least, a Monte-Carlo simulation [119] using the proposed situation-aware decision making algorithm was conducted. Compared to the earlier analysis, the Monte-Carlo simulation can offer a better evaluation of how the proposed algorithm can robustly handle the randomly-generated driving situations.

Table 5.3 Navigation Performance of Situation-aware Decision Making

Metric	Algorithm	T-Junction				Round-About			
		Single Obstacle Vehicle		Multiple Obstacle Vehicles		Single Obstacle Vehicle		Multiple Obstacle Vehicles	
		Give-way	Not Give-way	Small Gap[b]	Large Gap[b]	Give-way	Not Give-way	Small Gap[b]	Large Gap[b]
F.R.[a]	Reactive	0.18	0.00	0.02	0.00	0.15	0.00	0.00	0.01
	POMDP	0.00	0.02	0.01	0.03	0.00	0.03	0.00	0.02
T.T.[a] (sec)	Reactive	25.2	26.9	32.1	34.9	36.5	38.8	42.0	44.5
	POMDP	23.5	25.2	30.6	28.4	35.1	38.0	40.8	38.7

a: F.R.: Failure Rate; T.T.: Traveling Time. b: Small Gap: $2m \sim 3m$; Large Gap: $5m \sim 6m$.

Table 5.4 Failure analysis using Monte-Carlo simulation

Driving Scenario	Leader Following	Round-about Merging	T-junction Merging
Scenario Count	363	168	102
Failure Count	1	4	3
Failure Rate	0.0027	0.0238	0.0291

In order to randomly generate the driving situations, we defined five stations for the environment shown in Figure 4.4, such that the vehicles, including one autonomous vehicle and twenty obstacle vehicles, can easily travel between different stations by following the pre-learned routes. Moreover, each vehicle's driving speed can also be randomly assigned by uniformly sampling it from the range $[0, \mathrm{v}_{aggressive}]$, where $\mathrm{v}_{aggressive}$ denotes an aggressive speed generalized from vehicle behavior data. As such, the obstacle vehicles were expected maintain different traveling routes and motion intentions in each test.

Given these simulation settings, 200 simulation tests were conducted and all the failures were captured and recorded. Similarly, the failure is defined as either a collision happens or the autonomous vehicle cannot finish the driving task in time. Based on the location where the failure is happening, these failures are categorized into three different driving scenarios: leader following, round-about merging, and T-junction merging. As shown in Table 5.4, there were 8 failures being captured within this Monte-Carlo simulation. Compared to the leader following scenario, the proposed algorithm introduced slightly more failures for the scenarios of round-about mering and T-junction merging. This is reasonable because the vehicle behavior analysis is quite challenging for these two scenarios, and the decisions we made can be easily biased towards the wrong side. Moreover, we also recorded how ofter each driving scenario was handled by the autonomous vehicle, and a failure rate was computed for each driving scenario accordingly. As we expected, the proposed algorithm can maintain a fairly low failure rate for all these three driving scenarios.

5.5.2 Experiments

After the extensive evaluations were conducted in the simulation stage, the proposed situation-aware decision making algorithm was then applied to a real-time autonomous driving decision making at the T-junction. As demonstrated in Figure 5.7, two vehicles, including one

autonomous vehicle (or ego vehicle) and one manually driven vehicle (or obstacle vehicle), were employed for this real experiment, where the ego vehicle was negotiating with the coming obstacle vehicle to conduct a lane merging.

Instead of repeating the scenarios that have been extensively evaluated in the simulation stage, two new driving scenarios were tested in the experiment, including negotiating with an obstacle vehicle that approached the intersection slowly, and negotiating with an obstacle vehicle that approached the intersection aggressively. The maximum speed of the ego vehicle was set to $2m/s$, the acceleration and deceleration commands were given as $\pm 0.25m/s^2$ and the situation-aware decisions making algorithm was running at $2Hz$.

In Figure 5.8(a), the obstacle vehicle was driving normally to approach the intersection, and its motion intention was updated accordingly. Recognizing that the obstacle vehicle was still far away from the intersection, the ego vehicle made a reasonable decision to accelerate and enter the intersection immediately. After a while, the ego vehicle approached the lane merging point much earlier than the obstacle vehicle, as shown in Figure 5.8(b). The ego vehicle therefore decided to maintain its current speed and gracefully merged into the desired lane as in Figure 5.8(c). This experiment shows that the driving efficiency can be ensured by the proposed decision making algorithm. Instead of waiting for the coming vehicle that approached the intersection slowly, the ego vehicle can efficiently merge into the desired lane.

In another experiment that is demonstrated in Figure 5.9(a), the obstacle vehicle was driven aggressively to approach the intersection. The ego vehicle therefore cautiously decelerated and stopped until the obstacle vehicle moved ahead (see Figure 5.9(b)). Thereafter, the ego vehicle started to accelerate for lane merging in Figure 5.9(c). Compared to the earlier experiment, the driving decisions were made more cautiously in this experiment. This suggests that the proposed algorithm can properly recognize the risk of the current driving situation and choose appropriate actions to guarantee the driving safety.

Given these consistent results, we can find that our proposed algorithm is able to properly trade-off between safety and efficiency without any manually specified rules.

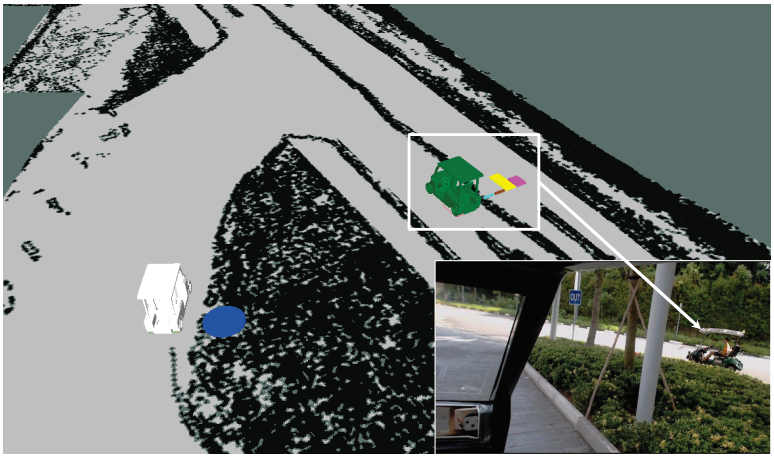


Figure 5.7 Real experiment settings for autonomous driving decision making with POMDP. The green vehicle is driven manually and functions as an obstacle vehicle, and the white vehicle represents the ego vehicle that is driven autonomously. The image is captured by the on-board camera of the ego vehicle.

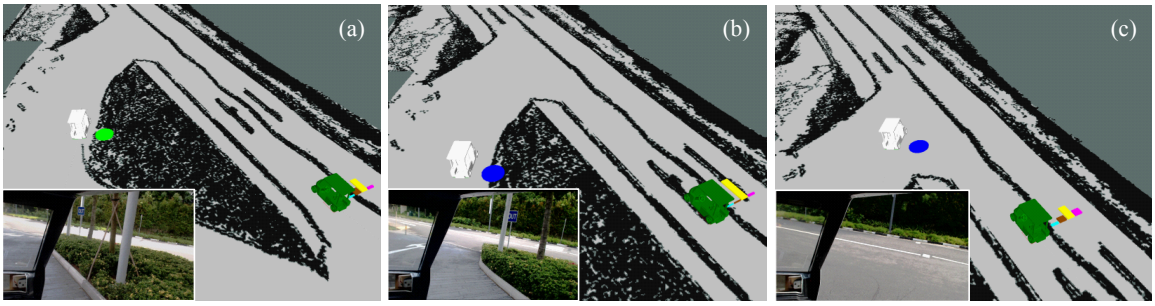


Figure 5.8 Negotiation with an obstacle vehicle which approached the intersection slowly. The action is represented as the circle marker: [Red:$Dec.$, Green:$Acc.$, Blue:$Cur.$]. The motion intention is represented as the histogram to the right of the obstacle vehicle:[Sky Blue:$Stopping$, Brown:$Hesitating$, Yellow:$Normal$, Purple:$Aggressive$], where the histogram height is proportional to the corresponding belief value.

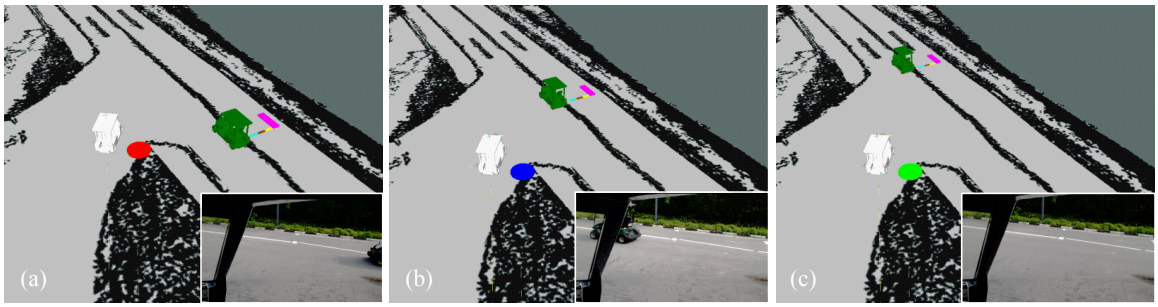


Figure 5.9 Negotiation with an obstacle vehicle which is driven aggressively. The action is represented as the circle marker: [Red:$Dec.$, Green:$Acc.$, Blue:$Cur.$]. The motion intention is represented as the histogram to the right of the obstacle vehicle:[Sky Blue:$Stopping$, Brown:$Hesitating$, Yellow:$Normal$, Purple:$Aggressive$], where the histogram height is proportional to the corresponding belief value.

5.6 Summary

In this chapter, a situation-aware decision making algorithm was designed for autonomous driving on the urban road. The concept of the urban road situation was discussed first, thereafter the situation-aware decision making problem was solved as a POMDP in an online manner. The proposed approach has been extensively evaluated, and its functionality and generality have been properly validated.

In the following chapter, a planning framework for autonomous driving on the urban road will be designed, where the situation-aware decision making algorithm is going to be extended to enable the steering control.

Chapter 6

Planning Framework Design for Autonomous Driving on Urban Roads

6.1 Introduction

Up to now, a risk-aware motion planning algorithm and a situation-aware decision making algorithm have been introduced, where the internal motion uncertainty and external situation uncertainty are addressed separately. However, safe and efficient autonomous driving in the urban road environment needs the motion planning and decision making to be conducted in a more integrated manner. As such, this chapter will introduce an autonomous vehicle planning framework that firmly integrates the planning algorithms proposed in the previous chapters.

To start with, a route planner, together with an autonomous vehicle booking system, will be introduced to provide the global guidance for autonomous driving. After that, the situation-aware decision making algorithm discussed in Chapter 5 will be extended to enable the vehicle steering control, i.e., deciding whether or not the trajectory planned by the motion planner should be committed. Moreover, some implementation issues of the motion planning algorithm will be discussed to facilitate real-time autonomous driving. The experimental results suggest that the designed autonomous vehicle planning framework can function safely and efficiently in the urban road environment.

The remainder of this chapter is organized as follows. The overview of the designed autonomous vehicle system is given in Section 6.2. The autonomous vehicle planning framework is then discussed in Section 6.3. The evaluations of the designed autonomous vehicle planning framework for autonomous overtaking are presented in Section 6.4, and Section 6.5 concludes this chapter.

6.2 Autonomous Vehicle System Overview

Before proceeding to the detailed discussion of the autonomous vehicle planning framework design, the autonomous vehicle system overview will be introduced in this section.

The autonomous vehicles used to carry out the field tests are a fleet of autonomous golf-carts developed for the Mobility-on-Demand (MoD) purpose. As demonstrated in Figure 6.1, the primary sensor input is provided by the LIDARs, webcam, IMU, and encoders. Regarding the environment sensing, one SICK LMS151 facing horizontally in the front and two SICK TIM551 mounted in the rear are properly configured to achieve 360-degree sensing coverage. The vehicle actuation is accomplished by installing two electric motors for the steering wheel and brake pedal, and the throttle and gear shift are managed by a customized microcontroller that communicates with the golf-cart's main ECU.

Software-wise, the ability of our vehicles to navigate autonomously is accomplished by integrating five major software modules: *Booking*, *Perception*, *Learning*, *Planning* and *Control* (see Figure 6.1), which are extended from the experience of the DARPA Urban Challenge. As discussed earlier, the perception module is to monitor both the vehicle's internal state and the surrounding environment. Given the perception results, the learning module is responsible for the road context and vehicle behavior learning. The planning module, moreover, is in charge of planning some proper maneuvers to drive the vehicle towards the destination. Thereafter, the controllers will come into effect to compute desired speed and steering commands. The corresponding autonomous driving missions are given by the passengers through an autonomous vehicle booking system.

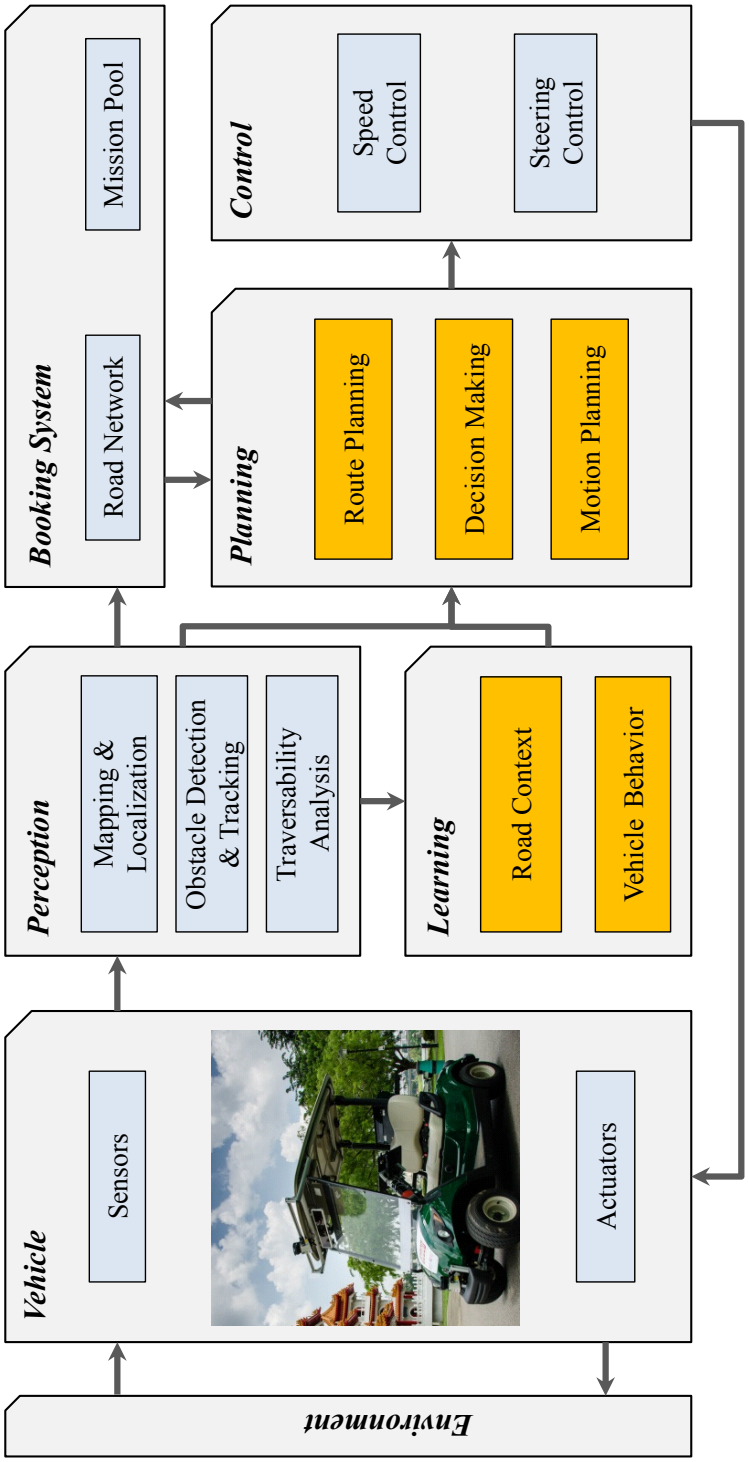


Figure 6.1 Overview of autonomous vehicle system. The autonomous driving functions highlighted by the orange blocks are the focuses of this thesis.

6.3 Autonomous Vehicle Planning Framework Design

In this section, we will detail the autonomous vehicle planning framework and the relevant components that are necessary for proper autonomous vehicle planning. In general, the designed autonomous vehicle planning framework consists of three planners: route planner, decision maker, and motion planner (see Figure 6.1). The route planner aims at planning the global traveling routes. The decision maker is responsible for the complicated driving decision making. Lastly, the motion planner searches the dynamically feasible trajectories to accomplish the desired driving behaviors. The details of these three planners will be discussed sequentially in this section.

6.3.1 Route Planning

A. Autonomous Vehicle Booking System

The objective of our autonomous vehicles is to provide a Mobility-on-Demand service, through which the autonomous driving missions are given by the public passengers, who can use our booking system (available at: http://booking.smartnusav.com) to request the autonomous shuttle service. Similar to the Uber booking service [120], the autonomous driving mission is given in the form of [*Pick-up Station*, *Drop-off Station*], where the *Pick-up Station* indicates the location where the passenger wants to be picked up, and the *Drop-off Station* denotes the passenger's destination. These requests are then queued into the database that functions as a mission pool. Given a large number of requests in the queue, a mission scheduling optimization can be conducted to balance the autonomous vehicle's travel distance and the passengers' waiting time [121].

From the autonomous vehicle side, the new mission will be fetched from the mission pool once the current mission has been accomplished. After receiving the newly assigned mission, the route searching module will come into effect to search two global routes. One route $GR_{veh,pick}$ will link the vehicle's current location to the pick-up station and another one $GR_{pick,drop}$ will start from the pick-up station and end at the drop-off station. A snapshot of the on-board GUI showing the route searching results is seen in Figure 6.2,

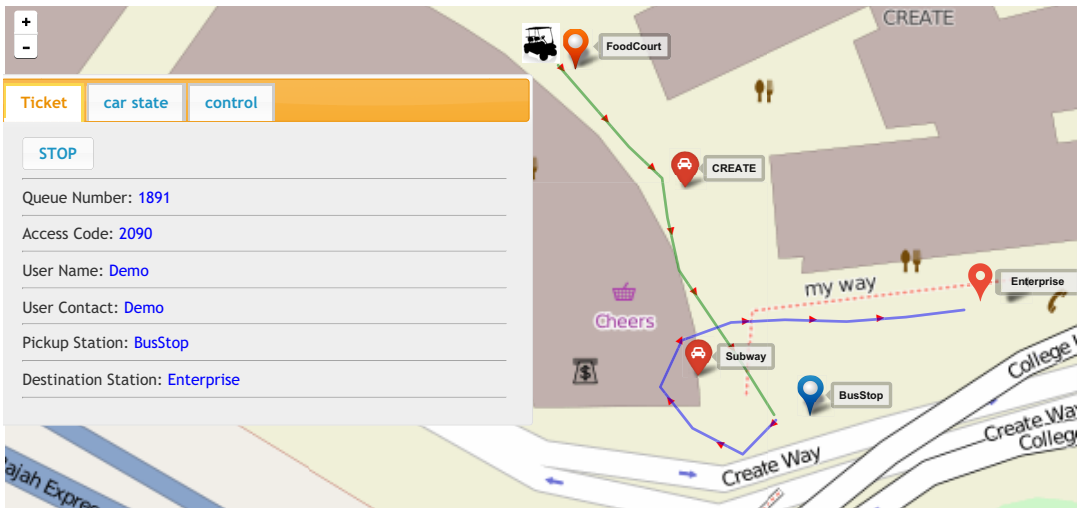


Figure 6.2 Snapshot of the on-board GUI. The green curve denotes the route $GR_{veh,pick}$, while the blue curve represents the second route $GR_{pick,drop}$.

where the autonomous vehicle is located at the *FoodCourt* station. The green curve denotes $GR_{veh,pick}$ and the blue curve represents $GR_{pick,drop}$.

B. Route Searching

Our autonomous vehicles are usually operated within a known environment, thus the road network information can always be extracted before the actual autonomous driving. The extraction of the road network can be accomplished by either adopting the road context inference approach detailed in Chapter 4 or performing a manual driving for road network information collection. While the road context inference approach is more promising than the manual driving, the vehicle behavior data collection needs to be conducted over a long duration. Comparatively, the manual road network information collection is more straightforward as the first step, and the manually driven routes can be guaranteed to be feasible for the time duration that the vehicle was driven. Therefore, we can always memorize these manually driven routes for road network processing and consider them as reference paths for autonomous driving. As an instance, the road network of Singapore's Chinese Garden is shown in Figure 6.3, where the manually driven routes (or reference paths) are divided into multiple segments to represent the road topology and facilitate the further route searching.

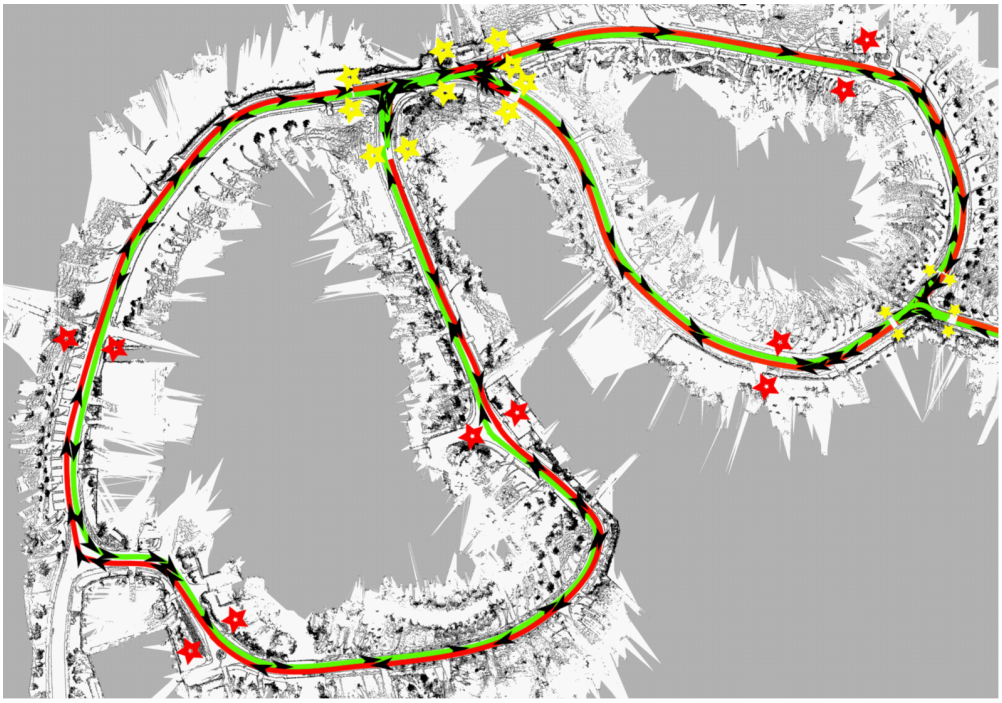


Figure 6.3 Road network of Singapore's Chinese Garden. The road segments are represented as the green and red curves, and their connectives are depicted by the red and yellow stars.

Given the road network information, the route searching can be conducted to provide global guidance for autonomous driving. To start with, we implicitly construct a directed road network graph $\mathcal{RNG} = [\mathcal{RV}, \mathcal{RE}]$ by mapping the road segments and their connectivities into directed edges $\mathcal{RE}$ and vertices $\mathcal{RV}$ respectively, where $\mathcal{RV} = \{RV_1, RV_2, ..., RV_N\}$ and $RE_{i,j} \in \mathcal{RE}, i, j \in [1, N]$ denotes the edge connecting RV_i and RV_j. Afterward, each edge is given a certain weight $\varpi(RE_{i,j}) : \mathcal{RE} \to \mathbb{R}^+$, such as the road segment's length, road congestion level, and so forth. Thereafter, we need to find out the autonomous vehicle's nearest station (or vertex) and treat it as the initial vertex for route searching. This is solved by searching the nearest road segment instead as,

$$RE_{veh} = \arg\min_{RE \in \mathcal{E}} \Gamma(P_{veh}, \Upsilon_{RE}), \tag{6.1}$$

where P_{veh} is the autonomous vehicle's current pose and Υ_{RE} denotes the road segment that corresponds to edge RE. The function $\Gamma(P_{veh}, \Upsilon_{RE})$ computes the sum of *perpendicular distance* and *heading angle difference* between autonomous vehicle pose P_{veh} and road

segment Υ_{RE}. The vehicle's nearest vertex RV_{veh} is then given as the starting vertex of edge RE_{veh}.

Given the road network graph $\mathcal{RNG}$ and the autonomous vehicle's nearest vertex RV_{veh}, the route searching can be solved as a graph search instead. Let $RT_{start,goal}$ be a graph path from RV_{start} to RV_{goal} consisting of a sequence of vertices,

$$RT_{start,goal} = \{RV_{start}..., RV_i, RV_{i+1}, ..., RV_{goal}\} \tag{6.2}$$

where any two adjacent vertices RV_i and RV_{i+1} are linked by the edge $RE_{i,i+1}$. The Dijkstra algorithm is then employed to solve the following shortest path problem and project out the optimal traveling route $GR^*_{veh,pick}$ as,

$$GR^*_{veh,pick} \leftarrow RT^*_{veh,pick} = \arg \min_{RT_{veh,pick}} \sum_{i=veh}^{pick-1} w(RE_{i,i+1}). \tag{6.3}$$

A similar process can be applied to find $GR^*_{pick,drop}$, and the autonomous vehicle will follow these two routes as autonomous driving guidance.

6.3.2 Decision Making

The situation-aware decision making algorithm discussed in Chapter 5 only considers the speed control, so that some basic but essential driving maneuvers, such as overtaking and lane changing, cannot be achieved. Therefore, extending the action space of the proposed POMDP model to incorporate steering control is greatly needed.

One straightforward approach to incorporating steering control into the decision maker is conducting steering command searching directly, where the action space becomes continuous, and its size becomes infinite. As is widely recognized, the computational tractability would be the main concern if the action space of POMDP is extended too much. Admittedly, some great achievements have been made in the research of continuous POMDP, but to efficiently solve a POMDP model with a continuous action space is still a challenge [122]. Therefore, an improperly extended action space can break down the real-time applicability of our situation-aware decision making algorithm.

As an alternative, the *trajectory selection* strategy is employed in this study. Given a set of trajectory candidates generated by the motion planner, a trajectory selection is responsible for selecting the best one according to the specific driving situation. The action space size, therefore, can still stay at an acceptable level. In this study, the extension is conducted by constructing a trajectory set with two candidates, including the previously committed trajectory and the newly planned trajectory. The previously committed trajectory denotes the trajectory that is committed in the previous planning loop, and the newly planned trajectory is the one planned by the motion planner in the current planning loop. As such, the decision maker's additional job is to decide whether or not the trajectory planned by the motion planner should be committed in each planning loop.

In view of the action space extension, three components of the POMDP model proposed in Chapter 5 need to be modified accordingly: the action space modeling, the ego vehicle's state transition model, and the reward function. The other components related to the obstacle vehicle modeling will remain unchanged. Unless otherwise stated, the mathematical symbols used below share the same meaning with that in Chapter 5.

Action Space Modeling. The modification of the action space is straightforward, where the extended action $a \in \mathcal{A}$ consists of two sub-actions: the speed control action $a_{speed} \in \mathcal{A}_{speed}$ and steering control action $a_{steer} \in \mathcal{A}_{steer}$. The extended action space is therefore modeled as,

$$\begin{aligned} \mathcal{A} &= \mathcal{A}_{speed} \times \mathcal{A}_{steer}, \\ \mathcal{A}_{speed} &= [Acc., Dec., Const.]^{\mathrm{T}}, \\ \mathcal{A}_{steer} &= [Commit, NewPlan]^{\mathrm{T}}, \end{aligned} \tag{6.4}$$

where the speed action space $\mathcal{A}_{speed}$ is composed of three acceleration commands, and the steering action space $\mathcal{A}_{speed}$ includes two trajectory candidates to be selected. The $Commit$ steering action means the vehicle should choose the previously committed trajectory, while the $NewPlan$ steering action suggests that the newly planned trajectory will be committed.

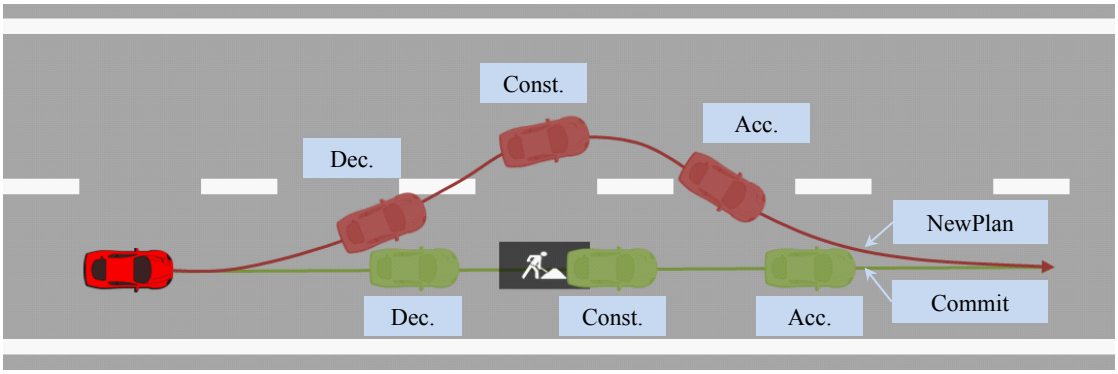


Figure 6.4 Illustration of the ego vehicle state transition. Two possible trajectories are available, and three speed actions can be decided for each trajectory.

Ego Vehicle State Transition Model. Due to the extended action space, the ego vehicle state transition model needs to be updated as well. To provide a better understanding of the ego vehicle state transition, a good instance is illustrated in Figure 6.4. The autonomous vehicle is driven on an urban road by following a committed trajectory (green) that is blocked by the construction field in front, and a new trajectory (red) is planned to bypass the construction field. The autonomous vehicle therefore has to choose a proper action defined in the action space, and its state transition can be conducted accordingly. Given the trajectory selected by the steering action a_{steer}, either committed or planned, three possible state transitions along each trajectory can be achieved by adjusting the ego vehicle speed according to the speed action a_{speed}.

Formally speaking, letting ξ denote the trajectory candidate, the ego vehicle state transition can be reformulated as,

$$\begin{aligned} \Pr(s_e'|s_e, a) &= \Pr(s_e'|s_e, a_{speed}, a_{steer}) \\ &= \sum_{\xi} \Pr(s_e'|s_e, a_{speed}, \xi) \Pr(\xi|a_{steer}), \end{aligned} \tag{6.5}$$

where $\Pr(\xi|a_{steer})$ models the one-to-one mapping between the steering action and the trajectory to be selected. The function $\Pr(s_e'|s_e, a_{speed}, \xi)$, thereafter, models the ego vehicle state transition along the trajectory that corresponds to the steering action a_{steer}. Implementation-wise, this can be achieved by conducting a forward simulation along the

selected trajectory (see Figure 6.4). More specifically, letting x_{veh} be the nearest trajectory state to the ego vehicle state s_e, the forward simulation then will start from x_{veh} and end at another trajectory state $x_{simulate}$, where the traveling distance between x_{veh} and $x_{simulate}$ along the trajectory is decided by the speed action as $(\mathrm{v}_e + a_{speed}\Delta t)\Delta t$. By assuming that the selected trajectory can be properly tracked, the updated state s'_e can be approximated as $s'_e = x_{simulate}$.

Reward Function. The objective of the autonomous vehicle is to arrive at the destination as efficiently as possible, while maintaining a safe distance to all the obstacles. Let $R(s, a_{speed})$ and $R(s, a_{steer})$ be the reward function for speed action and steering action respectively, the modified reward function is updated as $R(s, a) = R(s, a_{speed}) + R(s, a_{steer})$. The speed action-driven reward function $R(s, a_{speed})$ is the same as the one defined in Equation (5.13), and the steering action-driven reward function $R(s, a_{steer})$ is proposed as,

$$R(s, a_{steer}) = R_{action}(s, a_{steer}) + R_{scenario}(s, a_{steer}, \mathcal{RC}), \tag{6.6}$$

where $R_{action}(s, a_{steer}) \in \mathbb{R}^-$ provides a small penalty when $a_{steer} = NewPlan$. This is to discourage the frequent commitment of the newly planned trajectories in order to alleviate the trajectory oscillation. The scenario-driven reward function $R_{scenario}(s, a_{steer}, \mathcal{RC})$ assigns a reward or penalty to the steering action according to the specific driving scenarios. Since the road context $\mathcal{RC}$ has been learned, the autonomous vehicle can easily recognize the current driving scenario and decide whether or not the planned trajectory should be committed or not. For instance, if the autonomous vehicle is following a slow-moving obstacle vehicle, committing a new trajectory to overtake the front vehicle would be more desirable. On the other hand, if the autonomous vehicle is operated on a single-lane road with overtaking being prohibited, then the new trajectory to overtake the front vehicle should never be committed. Therefore, the design of $R_{scenario}(s, a_{steer}, \mathcal{RC})$ is context-based, where a tedious but straightforward lookup table can be constructed to model the scenario driven reward.

Given these three modifications, the POMDP model is able to take account of both the steering and speed control. Admittedly, the steering control cannot be completely addressed by the trajectory selection, but there is always a trade-off between the algorithm completeness and the real-time applicability, and we seek to find a balance between them.

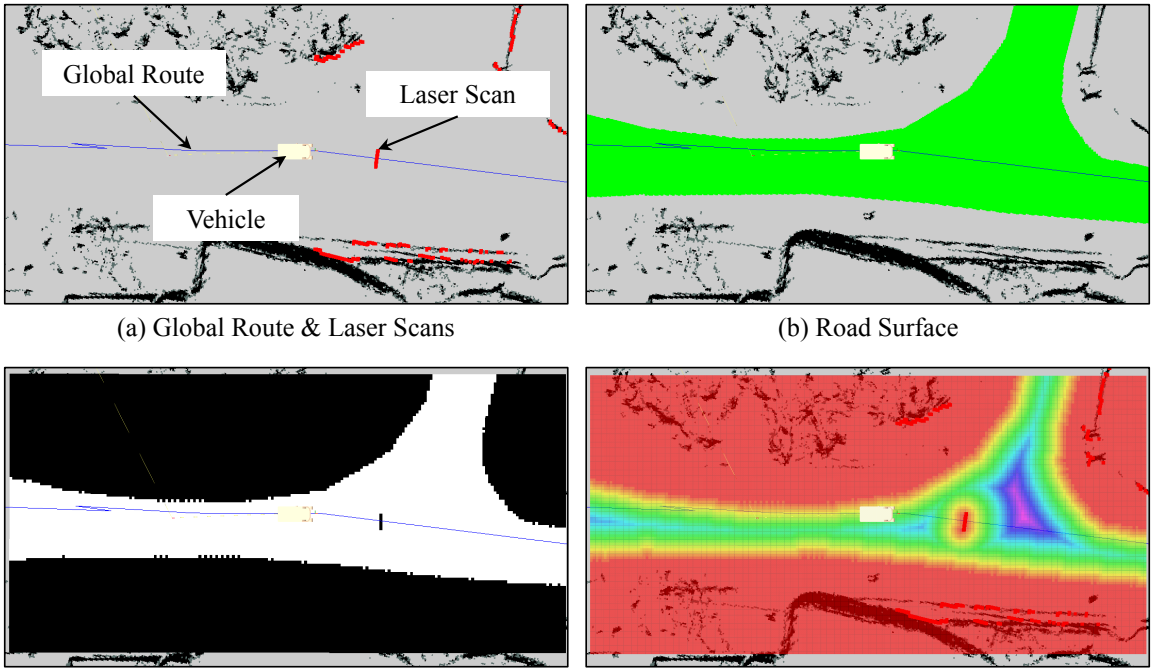
6.3.3 Motion Planning

In order to prepare the trajectory candidates requested by the decision maker, the motion planner will be employed to explore the continuous space and search the feasible trajectories.

To start with, the workspace for motion planning needs to be properly considered. As the autonomous vehicle is being operated in a large-scale urban road environment, to define the workspace as the entire driving field is inefficient and unnecessary, and would waste computation sources. Since a global route generated by the route planner is always available for our applications, the motion planning only needs to be performed locally. As such, the workspace can be dramatically narrowed by introducing a small-scale local map that is attached to the autonomous vehicle's own frame.

Thereafter, a careful analysis of the traversability is necessary to guarantee the trajectory safety. To appropriately represent the road surface (or drivable region) and the obstacles, a local metric map $\mathbb{M}_{metric}$ is tailored for our application. Specifically, the road surface detection can always be implemented beforehand as in Figure 6.5(b), then a local traversability map with height H and width W can be generated by labeling the road surface as collision-free. Afterward, the real-time obstacle detection results are projected onto this OGM with the corresponding grids being labeled as occupied (see Figure 6.5(c)). By calculating certain metrics $M([h, w]) : [H, W] \rightarrow \mathbb{R}^+$, such as the obstacle clearance and collision risk, for each grid $[h, w], h \in [0, H], w \in [0, W]$, the final metric map can be achieved as in Figure 6.5(d).

Given the workspace and the corresponding traversability information, the motion planning can then be carried out to search a trajectory that drives the autonomous vehicle towards the goal region. The initial state x_{init} for motion planning thereby is configured as



(a) Global Route & Laser Scans

(b) Road Surface

(c) Traversability Map

(d) Metric Map

Figure 6.5 Process of local metric map generation.

the vehicle's current pose and the motion planning goal $\mathcal{X}_{goal}$ is provided by the decision maker. In this study, the motion planning goal is simply selected as a state that is located on the global route. As such, the autonomous vehicle will always adhere to the global route and follow its guidance towards the destination.

Algorithm-wise, the risk-aware motion planning algorithm CC-RRT*-D is adopted to account for the internal motion uncertainty, and the cost function to be optimized is given as,

$$C(\xi) = Length(\xi) + \kappa \max_{t\in[0,T]} r(\xi(t)) + \varrho \sum_{t\in[0,T]} M(\xi(t)), \tag{6.7}$$

where ξ denotes the trajectory and $M(\xi(t))$ is the metric value of the trajectory state $\xi(t), t \in [0, T]$. The scaling factors κ and ϱ are to balance the impact of three factors: the trajectory length $Length(\xi)$, the maximum collision risk along the trajectory $\max_{t\in[0,T]} r(\xi(t))$, and the linearly integrated metric value $\sum_{t\in[0,T]} M(\xi(t))$. Acknowledged the constrained workspace within the urban road environment, the preferable sampling strategy is applied to bias the samples to the region with larger obstacle clearance.

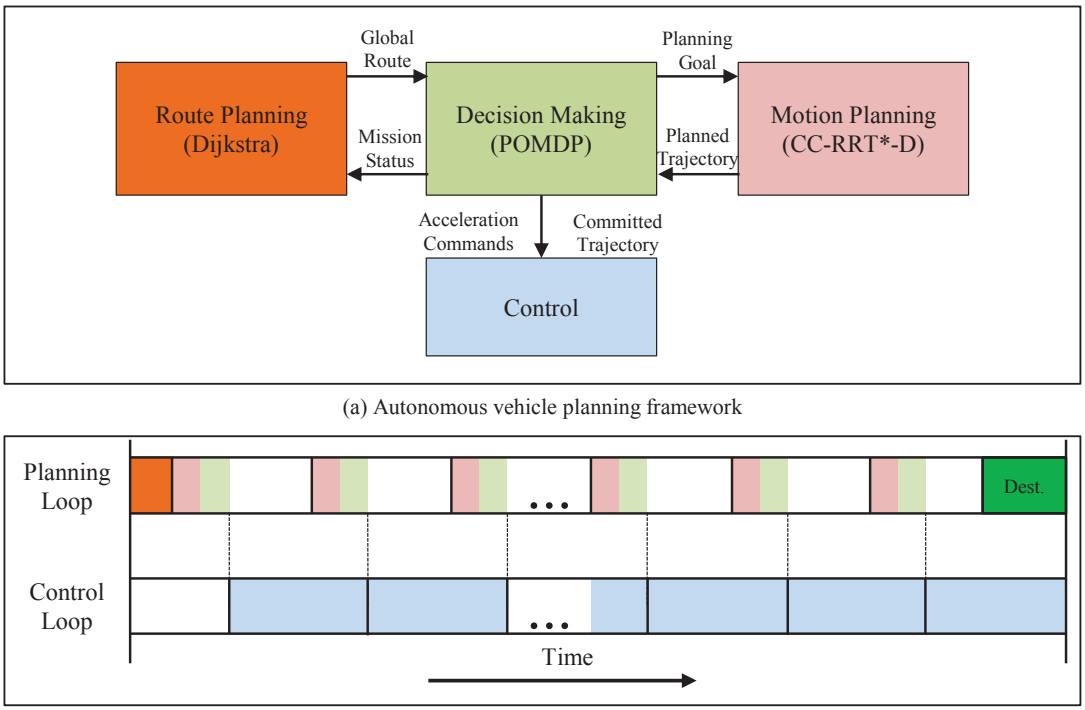
Algorithm 9: Implementation of Motion Planning

input : $\mathcal{X}_{goal}$, $\mathbb{M}_{metric}$
output : ξ

```
1  begin
2      while ¬TaskTimeOut do
3          x_init ← GetVehiclePose()
4          while ¬PathFound & ¬PlanTimeOut do
5              ξ ← Planning(x_init, X_goal, M_metric)
6          end
7          if PathFound then
8              return ξ
9          end
10         if PlanTimeOut then
11             x_init ← GetVehiclePose()
12             ResetPlanTimer()
13         end
14     end
15     return TimeoutFailure
16 end
```

Implementation-wise, the motion planner may spend quite a while or even fail to find a feasible trajectory when the environment becomes rather crowded. Also, we cannot expect the autonomous vehicle to stay steady within the motion planning process. Therefore, two timers, including the planning trial timer T_{plan} and the planning task timer T_{task}, are proposed to guarantee the proper functionality of the motion planner. The planning trial timer T_{plan} denotes a short period within which a solution trajectory should be found. It is introduced to account for the fact that the vehicle will not stay steady during the trajectory searching process, such that the motion planner needs to shift the initial state x_{init} to the updated vehicle pose if necessary. The duration of planning task timer T_{task} is relatively longer, which denotes the overall motion planning time permitted by the decision maker. If no feasible trajectories can be found within T_{task}, then the motion planning will be terminated, and a failure will be returned.

The detailed motion planning implementation strategy is illustrated in Algorithm 9. Given the assigned motion planning goal $\mathcal{X}_{goal}$ and the local metric map $\mathbb{M}_{metric}$, the motion planner will try to search for a feasible trajectory until the motion planning trial time T_{plan} runs out. If a solution trajectory is available, it will be inserted into the tra-



(a) Autonomous vehicle planning framework

(b) Planning and control loop for autonomous driving

Figure 6.6 Autonomous vehicle planning framework and autonomous driving control loop: (a) autonomous vehicle planning framework, (b) planning and control loop for autonomous driving.

jectory candidates for decision making, and the motion planning process will be stopped accordingly. For the $PlanTimeOut$ case (i.e., T_{plan} runs out), the motion planner will be re-initialized with the initial state being updated to the vehicle's current pose. Regarding the $TaskTimeOut$ case, since there is no solution trajectory that can be found within the permitted time, the $TimeoutFailure$ message will be sent to the decision maker.

6.3.4 Integration for Real-time Autonomous Driving

Till now, the details of the route planner, decision maker, and motion planner have been introduced. The integration of these three planners for real-time autonomous driving will be briefly discussed hereafter.

The overview of the designed autonomous vehicle planning framework and the autonomous driving control loop is depicted in Figure 6.6. The autonomous vehicle planning starts from the route planning when a driving mission is fetched from the autonomous vehicle booking system. The planned global route is then fed into the decision maker, where

Table 6.1 Evaluation Parameters for Autonomous Overtaking

Parameters	Value	Parameters	Value
Max Speed (Ego)	$3.0m/s$	Max Speed (Obst.)	$3.0m/s$
Acceleration	$0.5m/s^2$	Deceleration	$-0.5m/s^2$
Position Noise Variance	$0.3m$	Velocity Noise Variance	$0.3m/s$
Heading Angle Noise	$0.05rad$	Control Frequency	$1Hz$
Planning Trial Time	$0.02s$	Planning Task Time	$0.1s$

the decision maker utilizes it to check the current mission status and generate the local goals for motion planning.

After the global route searching, the autonomous vehicle planning is conducted in an iterative manner. In each planning loop, the decision maker will call the motion planner first with a goal being provided. The motion planner then starts to search a new trajectory given the real-time traversability information and sends back the solution trajectory to the decision maker. If no solution trajectory can be found by the motion planner within the permitted time, the decision maker will treat the previously committed trajectory as a "pseudo" newly planned trajectory. Afterward, the decision making will proceed to decide a proper acceleration command and select which trajectory to commit. Given the acceleration command and the selected trajectory, the autonomous vehicle control system will come into effect to execute the plan and drive the autonomous vehicle forward until the planning result in the next planning loop is available. This process keeps repeating until the autonomous vehicle reaches the final destination.

6.4 Evaluation of Autonomous Overtaking

In this section, we will describe the evaluations in which we validated the autonomous vehicle planning framework described in this chapter. We want to highlight the ability of the designed planning framework to handle the autonomous overtaking problem, which is not achievable by the decision making algorithm introduced in Chapter 5. Evaluations are carried out on an urban road demonstrated in Figure 4.4, and the *Stage* simulator [118] is employed for this evaluation. The Gaussian noise is purposely imposed on the vehicle pose and speed measurement and some essential evaluation parameters are given in Table 6.1.

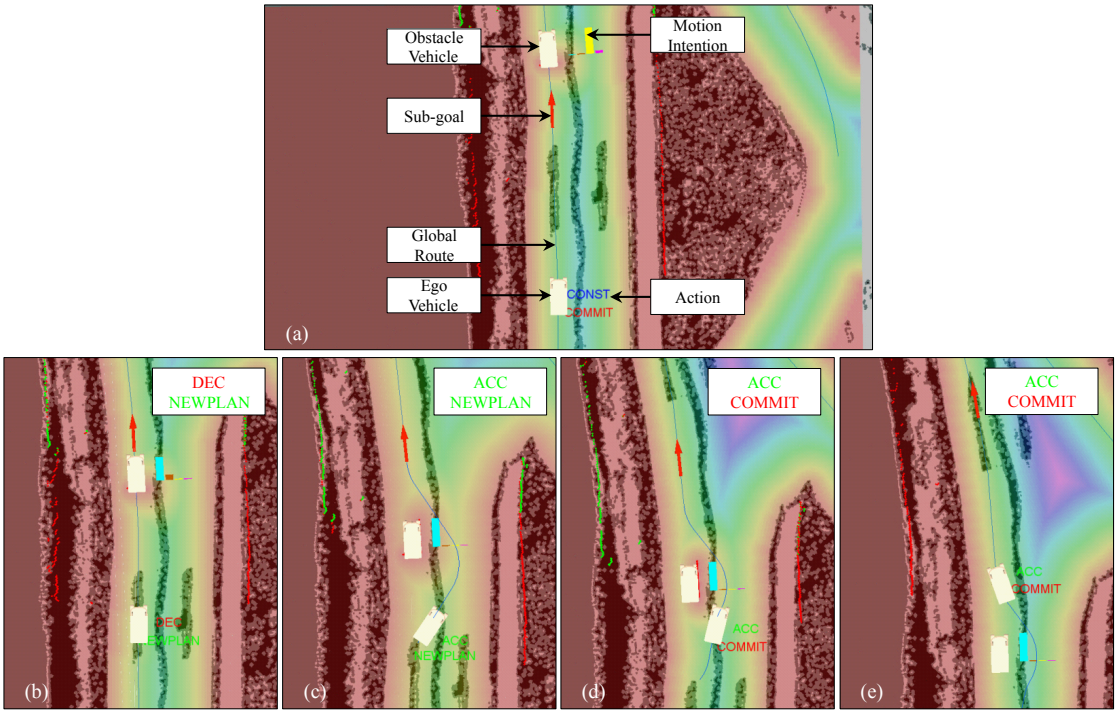


Figure 6.7 Autonomous overtaking with single obstacle vehicle: (a) shows the evaluation scenario, and (b)-(e) demonstrate the overtaking process. The global route is represented by the blue curve and the sub-goal is shown as the red arrow. The ego vehicle's action is presented by the corresponding text. The motion intention is represented as the histogram to the right of the obstacle vehicle:[Sky Blue:$Stopping$, Brown:$Hesitating$, Yellow:$Normal$, Purple:$Aggressive$], where the histogram height is proportional to the corresponding belief value.

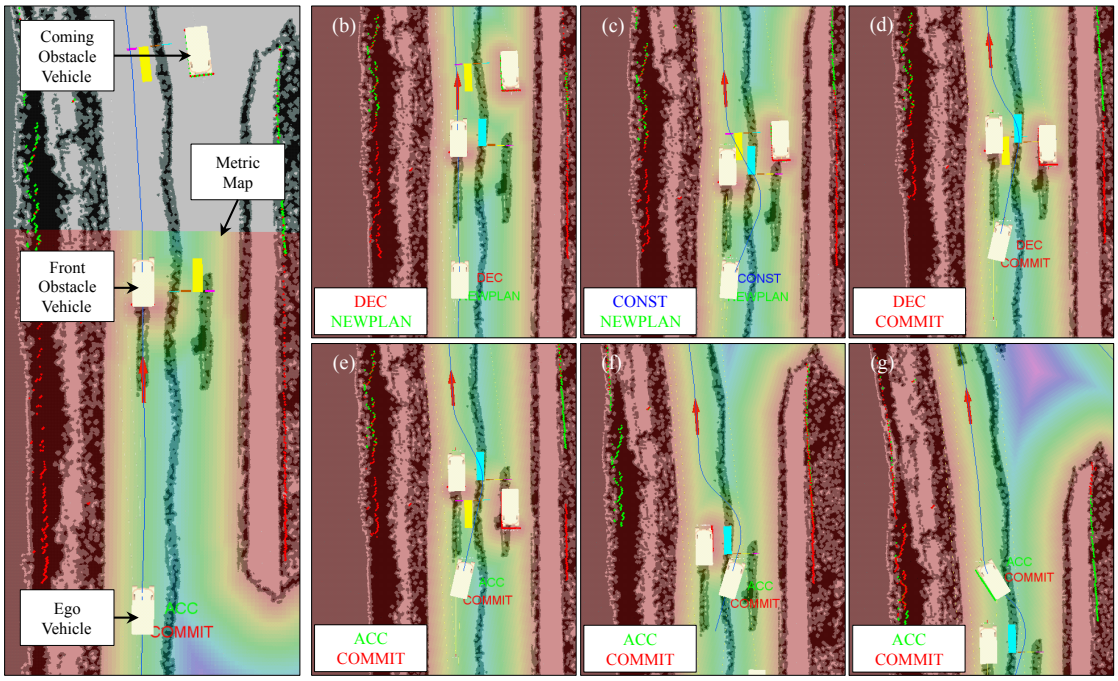


Figure 6.8 Autonomous overtaking with multiple obstacle vehicles: (a) shows the evaluation scenario, and (b)-(g) demonstrate the overtaking process. The global route is represented by the blue curve and the sub-goal is shown as the red arrow. The ego vehicle's action is presented by the corresponding text. The motion intention is represented as the histogram to the right of the obstacle vehicle:[Sky Blue:$Stopping$, Brown:$Hesitating$, Yellow:$Normal$, Purple:$Aggressive$], where the histogram height is proportional to the corresponding belief value.

To start with, the overtaking for a single obstacle vehicle was conducted, where the detailed scenario is illustrated in Figure 6.7(a). The autonomous vehicle was driven toward the final destination by following a global route, while maintaining a safe distance to the obstacle vehicle in front. Accordingly, the speed action *Const.* and steering action *Commit* were selected. Meanwhile, the motion planner kept trying to search for a trajectory towards the sub-goal represented by the red arrow. After a while, the front obstacle vehicle started to slow down, and its motion intention was updated as in Figure 6.7(b). Thereafter, the ego vehicle decided to decelerate and tried to select a new trajectory to overtake the front vehicle. In Figure 6.7(c), a feasible trajectory was found, and the ego vehicle started to accelerate and follow the new trajectory. The overtaking process then went through as in Figure 6.7(d) and Figure 6.7(e) until the obstacle vehicle was overtaken, and the ego vehicle drove back to the global route. From this simple case, we can find that the designed autonomous vehicle planning framework can properly coordinate the speed and steering control, and the autonomous driving ability is greatly improved.

Given the promising result achieved, the evaluation scenario was then extended by introducing another obstacle vehicle that was coming from the opposite direction as in Figure 6.8 (a). As such, the autonomous vehicle needs to safely and efficiently negotiate with these two obstacle vehicles. Similarly, the autonomous vehicle maintained a constant speed until the front obstacle vehicle stopped. In accordance to the situation evolution, the ego vehicle decided to decelerate before the overtaking trajectory was available (see Figure 6.8 (b)). After the new trajectory was found in Figure 6.8 (c), the ego vehicle started to slowly follow the new trajectory and then decided to decelerate again when the oncoming obstacle vehicle was in the way as shown in Figure 6.8 (d). After the oncoming obstacle vehicle moved passed in Figure 6.8 (e), the ego vehicle decided to accelerate in order to overtake the front parking obstacle vehicle. Compared to the simple case discussed earlier, negotiating with multiple obstacle vehicles is more complicated, where the autonomous vehicle needs to reason about the situation evolution more comprehensively, and the decision making becomes more safety-critical. As demonstrated by the evaluation results, our autonomous vehicle planning framework can still properly handle this complicated scenario.

6.5 Summary

In this chapter, the planning framework for autonomous driving on the urban road was designed. Building upon the algorithms proposed earlier, including the risk-aware motion planning algorithm and the situation-aware decision making algorithm, the route planner, decision maker and the motion planner were integrated seamlessly to improve the autonomous driving ability. The evaluations of the designed planning framework for autonomous overtaking have been conducted, and some promising results have been achieved.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

Planning is a key enabler for autonomous driving in the urban road environment. This thesis focuses on developing autonomous vehicle planning strategies in a probabilistic manner, such that the inevitable uncertainties arising from the internal actuator noise and the external situation evolution can be properly addressed.

We started the discussion showing the history and current status of autonomous driving technologies. The typical planning framework for autonomous navigation in the urban road environment was discussed. It consists of the route planner, decision maker (or behavior planner) and motion planner. We further studied the uncertainties that can be involved in the planning process, which include the internal motion uncertainty introduced by the autonomous vehicle control system and the external situation uncertainty arising from the autonomous vehicle perception system.

To address the internal motion uncertainty, a risk-aware motion planning algorithm CC-RRT*-D was proposed. The CC-RRT*-D algorithm leveraged the RRT* algorithm for space exploring and utilized the chance-constrained approach to evaluate the trajectories' collision risks. Within this process, the internal motion uncertainty was assumed to follow a Gaussian distribution and the vehicle dynamics model was locally linearized to achieve a closed-form solution for risk evaluation. By employing the conditional state propagation, the proposed algorithm is able to out-perform most existing methods. Extensive experimen-

tal results showed that the CC-RRT*-D is applicable to real-time motion planning given its efficiency.

After the motion uncertainty was addressed, the road context learning and its applications for vehicle behavior analysis were discussed to facilitate the situation awareness. Given the correlation between the road context and the vehicle behavior, the road context was learned by extracting consistencies within the observed vehicle behaviors. More specifically, the inference over the road context was divided into three stages, including the topology learning to model the road network, the motion learning to generalize the typical vehicle motion pattern, and the rule learning for traffic rules reasoning. After that, the road context was employed to analyze the vehicle behaviors. Depending on whether or not the vehicle behavior complies with the road context, the vehicle behavior analysis was classified into two types, including the abnormal driving behavior detection and the long-term motion prediction. The functionality and applicability of the proposed approach have been properly demonstrated by a case study that was conducted on a typical urban road.

Given the road context and the vehicle behavior analysis, a situation-aware decision making algorithm was introduced to address the situation uncertainty. The concept of *urban road situation*, which was defined as an integration of the road context and the vehicle's motion intention, was proposed first for urban road environment modeling. Thereafter, the situation-aware decision making problem was solved as a Partially Observable Decision Making Process (POMDP) to address the uncertainties arising from the situation evolution reasoning. Given the proposed POMDP model, the online POMDP solver DESPOT was employed for its efficiency. Thanks to DESPOT's online property, the proposed situation-aware decision making algorithm can handle various urban road driving scenarios in a real-time manner, which includes the leader following, traffic negotiation at T-junction or roundabout, and so forth.

While both the internal motion uncertainty and the external situation uncertainty had been properly resolved, safe and efficient autonomous driving needs motion planning and decision making to be conducted in an integrated manner. Therefore, a planning framework for autonomous driving on the urban road was designed, which was built upon the

algorithms discussed above. More specifically, the designed vehicle planning starts from the route planning, which searches the global route to guide the autonomous vehicle towards the destination. Then in each planning loop, the decision maker will call the motion planner first to search for a new trajectory given the real-time traversability information. Once the new trajectory becomes available, the situation-aware decision making will proceed to decide a proper acceleration command and select the best trajectory to follow. The evaluations of the autonomous overtaking behavior showed that the proposed motion planning and decision making algorithms can be seamlessly integrated, and the uncertainties can be comprehensively addressed.

7.2 Future Work

Although this thesis has successfully reported the proposal of planning under uncertainties for autonomous driving on the urban road, there are still some limitations, and more works need to be done to improve the current framework. The following four aspects are recommended for further study:

- **Extending the risk-aware motion planning algorithm to the non-linear systems with non-Gaussian noise.** While the CC-RRT*-D algorithm can properly handle linear systems with Gaussian noise, the extension of CC-RRT*-D to the non-linear systems with non-Gaussian noise is necessary to improve the algorithm's applicability. In the future, we plan to employ the Monte-Carlo approach for state distribution modeling, such that the constraints of linear systems with Gaussian noise can be alleviated. The increased computation cost, however, needs to be carefully taken into account.

- **Extending the road context inference and vehicle behavior analysis to handle the vehicle interactions.** In our learning process, the vehicle behavior training data are considered as independent, thus the interactions between the vehicles cannot be properly learned, and some unwanted bias can be introduced into the road context learning results. Therefore, future should explore the interactions between the vehicles and generalize the behavior models for vehicle interaction. The vehicle

interaction learning results can then be applied to improve the accuracy of the road context inference and the vehicle behavior analysis.

- **Performing some broader statistical tests on the situation-aware decision making algorithm.** In Chapter 5, the proposed situation-aware decision making algorithm has been extensively evaluated for various urban driving scenarios that are considered as representative. However, it would be more desirable to perform some broader statistical tests in order to advertise the proposed algorithm's generality. As a prerequisite, a well-learned road context and vehicle behavior model are necessary for the proposed decision making algorithm to function properly, so future work should also investigate how the learning results would affect the performance of the proposed decision making algorithm.

- **Extending POMDP for efficient planning in continuous space.** Recalling the discussion in Chapter 6, POMDP has provided a general planning framework to account for various uncertainties, whose applicability, however, is still limited when planning in continuous space. Autonomous vehicles naturally operate in a continuous space, so future work should extend POMDP for efficient planning in continuous space as well.

- **Exploring cooperative autonomous driving.** In this thesis, the planning strategies are designed for a single autonomous vehicle only. Thanks to the recent research advances in Vehicle-to-Vehicle and Vehicle-to-Infrastructure communication, cooperation among the vehicles has garnered significant research interests in the past decade. As demonstrated in the author's publication list (see Appendix C), some preliminary results on cooperative perception have been achieved, where the perception range of the autonomous vehicle can be extended as far as the connected vehicles can reach. The future work thereby will explore the cooperative driving, such that a fleet of autonomous vehicles can drive cooperatively by sharing their driving intention directly.

References

[1] U. Ozguner, T. Acarman, and K. A. Redmill, *Autonomous ground vehicles*. Artech House, 2011.

[2] "University of California Berkely", “Partners for Advanced Transportation Technologies.” http://www.path.berkeley.edu/about, 1986. [Online].

[3] "Carnegie Mellon Universiy", “From 0-70 in 30.” http://www.cmu.edu/homepage/environment/2014/fall/from-0-70-in-30.shtml, 2014. [Online].

[4] M. Buehler, K. Iagnemma, and S. Singh, *The 2005 DARPA grand challenge: the great robot race*, vol. 36. Springer Science & Business Media, 2007.

[5] M. Buehler, K. Iagnemma, and S. Singh, *The DARPA Urban Challenge: Autonomous vehicles in city traffic*, vol. 56. springer, 2009.

[6] Z. Chong, B. Qin, T. Bandyopadhyay, T. Wongpiromsarn, E. Rankin, M. Ang Jr, E. Frazzoli, D. Rus, D. Hsu, and K. Low, “Autonomous personal vehicle for the first-and last-mile transportation services,” in *Cybernetics and Intelligent Systems (CIS), 2011 IEEE 5th International Conference on*, pp. 253–260, IEEE, 2011.

[7] Z. Chong, B. Qin, T. Bandyopadhyay, T. Wongpiromsarn, B. Rebsamen, P. Dai, E. Rankin, and M. H. Ang Jr, “Autonomy for mobility on demand,” in *Intelligent Autonomous Systems 12*, pp. 671–682, Springer, 2013.

[8] S. Pendleton, T. Uthaicharoenpong, Z. J. Chong, G. M. J. Fu, B. Qin, W. Liu, X. Shen, Z. Weng, C. Kamin, M. A. Ang, *et al.*, “Autonomous golf cars for public trial of mobility-on-demand service,” in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, IEEE, 2015.

[9] "Singapore-MIT Alliance for Research and Technology", “Launched! smart-nus driverless car trials at one-north.” http://smart.mit.edu/news-a-events/press-room/article/54-launched-smart-nus-driverless-car-trials-at-one-north-.html, 2015. [Online].

[10] D. Ferguson, T. M. Howard, and M. Likhachev, "Motion planning in urban environments," *Journal of Field Robotics*, vol. 25, no. 11-12, pp. 939–960, 2008.

[11] M. Campbell, M. Egerstedt, J. P. How, and R. M. Murray, "Autonomous driving in urban environments: approaches, lessons and challenges," *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 368, no. 1928, pp. 4649–4672, 2010.

[12] W. Liu and M. Ang, "Incremental sampling-based algorithm for risk-aware planning under motion uncertainty," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 2051–2058, IEEE, 2014.

[13] L. Gomes", "Hidden obstacles for google's self-driving cars." http://www.technologyreview.com/news/530276/hidden-obstacles-for-googles-self-driving-cars, 2014. [Online].

[14] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[15] 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*, vol. 25, no. 9, pp. 569–597, 2008.

[16] W. Liu, S.-W. Kim, S. Pendleton, and M. H. Ang Jr, "Situation-aware decision making for autonomous driving on urban road using online pomdp," in *Intelligent Vehicles Symposium (IV), 2015 IEEE*, pp. 1126–1133, IEEE, 2015.

[17] W. Liu, S.-W. Kim, and M. H. Ang Jr, "Probabilistic road context inference for autonomous vehicles," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 1640–1647, IEEE, 2015.

[18] A. Somani, N. Ye, D. Hsu, and W. S. Lee, "Despot: Online pomdp planning with regularization," in *Advances in Neural Information Processing Systems*, pp. 1772–1780, 2013.

[19] L. Fletcher, S. Teller, E. Olson, D. Moore, Y. Kuwata, J. How, J. Leonard, I. Miller, M. Campbell, D. Huttenlocher, *et al.*, "The mit–cornell collision and why it happened," *Journal of Field Robotics*, vol. 25, no. 10, pp. 775–807, 2008.

[20] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, “Practical search techniques in path planning for autonomous driving,” in *Proceedings of the First International Symposium on Search Techniques in Artificial Intelligence and Robotics (STAIR-08)*, AAAI, 2008.

[21] "DARPA", “Road network definition file.” http://archive.darpa.mil/grandchallenge/docs/RNDF_MDF_Formats_031407.pdf, 2007. [Online].

[22] M. Likhachev, D. I. Ferguson, G. J. Gordon, A. Stentz, and S. Thrun, “Anytime dynamic a*: An anytime, replanning algorithm.,” in *ICAPS*, pp. 262–271, 2005.

[23] A. Bacha, C. Bauman, R. Faruque, M. Fleming, C. Terwelp, C. Reinholtz, D. Hong, A. Wicks, T. Alberi, D. Anderson, *et al.*, “Odin: Team victortango’s entry in the darpa urban challenge,” *Journal of Field Robotics*, vol. 25, no. 8, pp. 467–492, 2008.

[24] Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. How, and G. Fiore, “Real-time motion planning with applications to autonomous urban driving,” *Control Systems Technology, IEEE Transactions on*, vol. 17, no. 5, pp. 1105–1118, 2009.

[25] J. Bohren, T. Foote, J. Keller, A. Kushleyev, D. Lee, A. Stewart, P. Vernaza, J. Derenick, J. Spletzer, and B. Satterfield, “Little ben: the ben franklin racing team’s entry in the 2007 darpa urban challenge,” *Journal of Field Robotics*, vol. 25, no. 9, pp. 598–614, 2008.

[26] I. Miller, M. Campbell, D. Huttenlocher, A. Nathan, F.-R. Kline, P. Moran, N. Zych, B. Schimpf, S. Lupashin, E. Garcia, *et al.*, “Team cornell’s skynet: Robust perception and planning in an urban environment,” in *The DARPA Urban Challenge*, pp. 257–304, Springer, 2009.

[27] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, *et al.*, “Towards fully autonomous driving: Systems and algorithms,” in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pp. 163–168, IEEE, 2011.

[28] A. Broggi, M. Buzzoni, S. Debattisti, P. Grisleri, M. C. Laghi, P. Medici, and P. Versari, “Extensive tests of autonomous driving technologies,” *Intelligent Transportation Systems, IEEE Transactions on*, vol. 14, no. 3, pp. 1403–1415, 2013.

[29] J.-C. Latombe, *Robot motion planning*, vol. 124. Springer Science & Business Media, 1996.

[30] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[31] R. A. Brooks and T. Lozano-Perez, "A subdivision algorithm in configuration space for findpath with rotation.," tech. rep., DTIC Document, 1982.

[32] L. Kavraki and J.-C. Latombe, "Randomized preprocessing of configuration for fast path planning," in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pp. 2138–2145, IEEE, 1994.

[33] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[34] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 2, pp. 995–1001, IEEE, 2000.

[35] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[36] E. Kavraki, M. N. Kolountzakis, and J.-C. Latombe, "Analysis of probabilistic roadmaps for path planning," *Robotics and Automation, IEEE Transactions on*, vol. 14, no. 1, pp. 166–171, 1998.

[37] D. Hsu, J.-C. Latombe, and H. Kurniawati, "On the probabilistic foundations of probabilistic roadmap planning," *The International Journal of Robotics Research*, vol. 25, no. 7, pp. 627–643, 2006.

[38] M. S. Branicky, S. M. LaValle, K. Olson, and L. Yang, "Quasi-randomized path planning," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 2, pp. 1481–1487, IEEE, 2001.

[39] J. Cortés, L. Jaillet, and T. Siméon, "Molecular disassembly with rrt-like algorithms," in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 3301–3306, IEEE, 2007.

[40] M. S. Branicky, M. M. Curtiss, J. Levine, and S. Morgan, "Sampling-based planning, control and verification of hybrid systems," *IEE Proceedings-Control Theory and Applications*, vol. 153, no. 5, pp. 575–590, 2006.

[41] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 1, pp. 116–129, 2002.

[42] A. Shkolnik, M. Levashov, I. R. Manchester, and R. Tedrake, "Bounding on rough terrain with the littledog robot," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 192–215, 2011.

[43] S. Teller, M. R. Walter, M. Antone, A. Correa, R. Davis, L. Fletcher, E. Frazzoli, J. Glass, J. P. How, A. S. Huang, *et al.*, "A voice-commandable robotic forklift working alongside humans in minimally-prepared outdoor environments," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 526–533, IEEE, 2010.

[44] J. Canny and J. Reif, "New lower bound techniques for robot motion planning problems," in *Foundations of Computer Science, 1987., 28th Annual Symposium on*, pp. 49–60, IEEE, 1987.

[45] S. Skiena, "Dijkstra's algorithm," *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, Reading, MA: Addison-Wesley*, pp. 225–227, 1990.

[46] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, no. 2, pp. 100–107, 1968.

[47] M. Likhachev, G. J. Gordon, and S. Thrun, "Ara*: Anytime a* with provable bounds on sub-optimality.," in *NIPS*, 2003.

[48] A. Stentz, "The focussed dˆ* algorithm for real-time replanning," in *IJCAI*, vol. 95, pp. 1652–1659, 1995.

[49] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, "Anytime search in dynamic graphs," *Artificial Intelligence*, vol. 172, no. 14, pp. 1613–1643, 2008.

[50] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *The International Journal of Robotics Research*, vol. 28, no. 8, pp. 933–945, 2009.

[51] C. Urmson and R. G. Simmons, "Approaches for heuristically biasing rrt growth.," in *IROS*, pp. 1178–1183, 2003.

[52] D. Ferguson and A. Stentz, "Anytime rrts," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pp. 5369–5375, IEEE, 2006.

[53] N. A. Wedge and M. S. Branicky, "On heavy-tailed runtimes and restarts in rapidly-exploring random trees," in *Twenty-Third AAAI Conference on Artificial Intelligence*, pp. 127–133, 2008.

[54] L. Jaillet, J. Cortés, and T. Siméon, "Transition-based rrt for path planning in continuous cost spaces," in *Intelligent Robots and Systems (IROS), 2008 IEEE/RSJ International Conference on*, pp. 2145–2150, IEEE, 2008.

[55] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt*," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 1478–1483, IEEE, 2011.

[56] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed rrt*: Optimal incremental path planning focused through an admissible ellipsoidal heuristic," *arXiv preprint arXiv:1404.2334*, 2014.

[57] F. Islam, J. Nasir, U. Malik, Y. Ayaz, and O. Hasan, "Rrt*-smart: Rapid convergence implementation of rrt* towards optimal solution," in *Mechatronics and Automation (ICMA), 2012 International Conference on*, pp. 1651–1656, IEEE, 2012.

[58] J. Van Den Berg, P. Abbeel, and K. Goldberg, "Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 895–913, 2011.

[59] L. Blackmore, H. Li, and B. Williams, "A probabilistic approach to optimal robust path planning with obstacles," in *American Control Conference, 2006*, pp. 7–pp, IEEE, 2006.

[60] M. Ono and B. C. Williams, "Iterative risk allocation: A new approach to robust model predictive control with a joint chance constraint," in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pp. 3427–3432, IEEE, 2008.

[61] S. Thrun, W. Burgard, D. Fox, *et al.*, *Probabilistic robotics*, vol. 1. MIT press Cambridge, 2005.

[62] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[63] L. Blackmore and B. C. Williams, "Optimal, robust predictive control of nonlinear systems under probabilistic uncertainty using particles," in *American Control Conference, 2007. ACC'07*, pp. 1759–1761, IEEE, 2007.

[64] L. Blackmore, M. Ono, A. Bektassov, and B. C. Williams, "A probabilistic particle-control approximation of chance-constrained stochastic predictive control," *Robotics, IEEE Transactions on*, vol. 26, no. 3, pp. 502–517, 2010.

[65] B. Luders, M. Kothari, and J. P. How, "Chance constrained rrt for probabilistic robustness to environmental uncertainty," in *Proceeding of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2010.

[66] N. A. Melchior and R. Simmons, "Particle rrt for path planning with uncertainty," in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 1617–1624, IEEE, 2007.

[67] S. Prentice and N. Roy, "The belief roadmap: Efficient planning in belief space by factoring the covariance," *The International Journal of Robotics Research*, vol. 28, no. 11-12, pp. 1448–1465, 2009.

[68] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 723–730, IEEE, 2011.

[69] B. D. Luders, S. Karaman, and J. P. How, "Robust sampling-based motion planning with asymptotic optimality guarantees," in *AIAA Guidance, Navigation, and Control Conference (GNC), Boston, MA*, 2013.

[70] S. Sivaraman and M. M. Trivedi, "Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis," 2013.

[71] S. Cherng, C.-Y. Fang, C.-P. Chen, and S.-W. Chen, "Critical motion detection of nearby moving vehicles in a vision-based driver-assistance system," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 10, no. 1, pp. 70–82, 2009.

[72] G. S. Aoude, B. D. Luders, D. S. Levine, and J. P. How, "Threat-aware path planning in uncertain urban environments," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 6058–6063, IEEE, 2010.

[73] F. Garcia, P. Cerri, A. Broggi, A. de la Escalera, and J. M. Armingol, "Data fusion for overtaking vehicle detection based on radar and optical flow," in *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pp. 494–499, IEEE, 2012.

[74] A. Barth and U. Franke, "Tracking oncoming and turning vehicles at intersections," in *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pp. 861–868, IEEE, 2010.

[75] D. Kasper, G. Weidl, T. Dang, G. Breuel, A. Tamke, A. Wedel, and W. Rosenstiel, "Object-oriented bayesian networks for detection of lane change maneuvers," *Intelligent Transportation Systems Magazine, IEEE*, vol. 4, no. 3, pp. 19–31, 2012.

[76] T. Bandyopadhyay, K. S. Won, E. Frazzoli, D. Hsu, W. S. Lee, and D. Rus, "Intention-aware motion planning," in *Algorithmic Foundations of Robotics X*, pp. 475–491, Springer, 2013.

[77] T. Bandyopadhyay, C. Z. Jie, D. Hsu, M. H. Ang Jr, D. Rus, and E. Frazzoli, "Intention-aware pedestrian avoidance," in *International Symposium on Experimental Robotics*, pp. 963–977, Springer International Publishing, 2013.

[78] W. Liu, S.-W. Kim, K. Marczuk, and M. H. Ang, "Vehicle motion intention reasoning using cooperative perception on urban road," in *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pp. 424–430, IEEE, 2014.

[79] C. Hermes, C. Wohler, K. Schenk, and F. Kummert, "Long-term vehicle motion prediction," in *Intelligent Vehicles Symposium, 2009 IEEE*, pp. 652–657, IEEE, 2009.

[80] A. D. V. Govea, "Growing hidden markov models," in *Incremental Learning for Motion Prediction of Pedestrians and Vehicles*, pp. 71–82, Springer, 2010.

[81] G. S. Aoude, B. D. Luders, J. M. Joseph, N. Roy, and J. P. How, "Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns," *Autonomous Robots*, vol. 35, no. 1, pp. 51–76, 2013.

[82] D. Vasquez, T. Fraichard, and C. Laugier, "Incremental learning of statistical motion patterns with growing hidden markov models," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 10, no. 3, pp. 403–416, 2009.

[83] B. T. Morris and M. M. Trivedi, "Trajectory learning for activity understanding: Unsupervised, multilevel, and long-term adaptive approach," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 11, pp. 2287–2301, 2011.

[84] G. Agamennoni, J. I. Nieto, and E. M. Nebot, "Robust and accurate road map inference," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 3946–3953, IEEE, 2010.

[85] A. Alin, M. V. Butz, and J. Fritsch, "Incorporating environmental knowledge into bayesian filtering using attractor functions," in *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pp. 476–481, IEEE, 2012.

[86] B. D. Luders, G. S. Aoude, J. M. Joseph, N. Roy, and J. P. How, "Probabilistically safe avoidance of dynamic obstacles with uncertain motion patterns," 2011.

[87] S. Sivaraman, B. Morris, and M. Trivedi, "Observing on-road vehicle behavior: Issues, approaches, and perspectives," in *Intelligent Transportation Systems - (ITSC), 2013 16th International IEEE Conference on*, pp. 1772–1777, IEEE, Oct 2013.

[88] F. Tupin, H. Maitre, J.-F. Mangin, J.-M. Nicolas, and E. Pechersky, "Detection of linear features in sar images: application to road network extraction," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 36, no. 2, pp. 434–453, 1998.

[89] G. Lisini, C. Tison, F. Tupin, and P. Gamba, "Feature fusion to improve road network extraction in high-resolution sar images," *Geoscience and Remote Sensing Letters, IEEE*, vol. 3, no. 2, pp. 217–221, 2006.

[90] S. Edelkamp and S. Schrödl, "Route planning and map inference with global positioning traces," in *Computer Science in Perspective*, pp. 128–151, Springer, 2003.

[91] R. Brüntrup, S. Edelkamp, S. Jabbar, and B. Scholz, "Incremental map generation with gps traces," in *Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE*, pp. 574–579, IEEE, 2005.

[92] B. Qin, Z. Chong, T. Bandyopadhyay, M. H. Ang Jr, E. Frazzoli, and D. Rus, "Road detection and mapping using 3d rolling window," in *Intelligent Vehicles Symposium*, IEEE, 2013.

[93] B. Qin, W. Liu, X. Shen, Z. Chong, T. Bandyopadhyay, M. Ang, E. Frazzoli, and D. Rus, "A general framework for road marking detection and analysis," in *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*, pp. 619–625, IEEE, 2013.

[94] D. Makris and T. Ellis, "Learning semantic scene models from observing activity in visual surveillance," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 2005.

[95] G. Agamennoni, J. Nieto, E. M. Nebot, *et al.*, "Estimation of multivehicle dynamics by considering contextual information," *Robotics, IEEE Transactions on*, vol. 28, no. 4, pp. 855–870, 2012.

[96] T. Taha, J. V. Miró, and G. Dissanayake, "Pomdp-based long-term user intention prediction for wheelchair navigation," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pp. 3920–3925, IEEE, 2008.

[97] H. Bai, D. Hsu, and W. S. Lee, "Integrated perception and planning in the continuous space: A pomdp approach," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1288–1302, 2014.

[98] E. Galceran, A. G. Cunningham, R. M. Eustice, and E. Olson, "Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction," in *Proc. Robot.: Sci. & Syst. Conf*, no. 1, p. 2, 2015.

[99] C. Berger and B. Rumpe, "Autonomous driving-5 years after the urban challenge: The anticipatory vehicle as a cyber-physical system.," *GI-Jahrestagung*, pp. 789–798, 2012.

[100] S. Lefèvre, D. Vasquez, and C. Laugier, "A survey on motion prediction and risk assessment for intelligent vehicles," *Robomech Journal*, vol. 1, no. 1, pp. 1–14, 2014.

[101] S. C. Ong, S. W. Png, D. Hsu, and W. S. Lee, "Planning under uncertainty for robotic tasks with mixed observability," *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 1053–1068, 2010.

[102] C. Shaojun, "Online POMDP Planning for Vehicle Navigation in Densely Populated Area," Master's thesis, National University of Singapore, 2014.

[103] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic decision-making under uncertainty for autonomous driving using continuous POMDPs," in *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pp. 392–399, IEEE, 2014.

[104] J. Wei, J. M. Dolan, J. M. Snider, and B. Litkouhi, "A point-based mdp for robust single-lane autonomous driving behavior under uncertainties," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 2586–2592, IEEE, 2011.

[105] S. Ulbrich and M. Maurer, "Probabilistic online pomdp decision making for lane changes in fully automated driving," in *Intelligent Transportation Systems*, pp. 2063–2070, 2013.

[106] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1, pp. 99–134, 1998.

[107] G. Shani, J. Pineau, and R. Kaplow, "A survey of point-based POMDP solvers," *Autonomous Agents and Multi-Agent Systems*, vol. 27, no. 1, pp. 1–51, 2013.

[108] H. Kurniawati, D. Hsu, and W. S. Lee, "SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces," in *Robotics: Science and Systems*, vol. 2008, Zurich, Switzerland, 2008.

[109] J. Pineau, G. Gordon, S. Thrun, *et al.*, "Point-based value iteration: An anytime algorithm for POMDPs," in *IJCAI*, vol. 3, pp. 1025–1032, 2003.

[110] M. T. Spaan and N. A. Vlassis, "Perseus: Randomized point-based value iteration for POMDPs," *J. Artif. Intell. Res.(JAIR)*, vol. 24, pp. 195–220, 2005.

[111] D. Silver and J. Veness, "Monte-carlo planning in large POMDPs.," in *NIPS*, vol. 23, pp. 2164–2172, 2010.

[112] S. Patil, J. van den Berg, and R. Alterovitz, "Estimating probability of collision for safe motion planning under gaussian motion and sensing uncertainty," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 3238–3244, IEEE, 2012.

[113] Z. Chong, B. Qin, T. Bandyopadhyay, Wongpiromsarn, and M. Ang Jr, "Synthetic 2d lidar for precise vehicle localization in 3d urban environment," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, IEEE, 2013.

[114] B. Qin, Z. Chong, S. H. Soh, T. Bandyopadhyay, M. H. Ang Jr., E. Frazzoli, and D. Rus, "A spatial-temporal approach for moving object recognition with 2d lidar," in *International Symposium on Experimental Robotics (ISER)*, Springer, 2014.

[115] B. Fritzke *et al.*, "A growing neural gas network learns topologies," *Advances in neural information processing systems*, vol. 7, pp. 625–632, 1995.

[116] C. E. Rasmussen, "Gaussian processes for machine learning," 2006.

[117] S. Ross, J. Pineau, S. Paquet, and B. Chaib-Draa, "Online planning algorithms for POMDPs.," *J. Artif. Intell. Res.(JAIR)*, vol. 32, pp. 663–704, 2008.

[118] R. Vaughan, "Massively multi-robot simulation in stage," *Swarm Intelligence*, vol. 2, no. 2-4, pp. 189–208, 2008.

[119] C. Z. Mooney, *Monte carlo simulation*, vol. 116. Sage Publications, 1997.

[120] "Uber Technologies Inc.", "Uber ride service." https://www.uber.com/ride/. [Online].

[121] K. A. Marczuk, H. S. S. Hong, C. M. L. Azevedo, M. Adnan, S. D. Pendleton, E. Frazzoli, and D. H. Lee, "Autonomous mobility on demand in simmobility: Case study of the central business district in singapore," in *Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM), 2015 IEEE 7th International Conference on*, pp. 167–172, IEEE, 2015.

[122] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, "Intention-aware online POMDP planning for autonomous driving in a crowd," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 454–460, IEEE, 2015.

Appendix A

Growing Neural Gas

To start with, the concept of a network is specified as:

- A set of nodes (neurons). Each neuron has its associated reference vector $w \in \mathbb{R}^n$. The reference vectors can be regarded as positions in the input space of their corresponding neurons.
- A set of edges (connections) between pairs of neurons. These connections are not weighted and their purpose is to define the topological structure. An *edge aging scheme* is used to remove connections that are invalid due to the motion of the neuron during the adaptation process.

The Growing Neural Gas learning algorithm to map the network to the input manifold that follows distribution $\Pr(\zeta)$ is as follows:

1. Start with two units j and k at random positions w_j and w_k in $\mathbb{R}^n$.
2. Generate an input signal ζ according to $\Pr(\zeta)$.
3. Find the nearest unit s_1 (winner neuron) and the second-nearest unit s_2.
4. Increment the age of all edges emanating from s_1 by $age_{increase}$.
5. Add the squared distance between the input signal and the winner neuron to a counter error of s_1 as: $\Delta error(s_1) = \|w_{s_1} - \zeta\|^2$.

6. Move s_1 and its topological neighbors n (neurons connected to s_1) towards ζ by fractions ϵ_b and ϵ_n of the total distance: $\Delta w_{s_1} = \epsilon_b(\zeta - w_{s_1})$, $\Delta w_{s_n} = \epsilon_n(\zeta - w_{s_n})$.

7. If s_1 and s_2 are connected by an edge, set the age of this edge to zero. If such an edge does not exist, create it.

8. Remove edges with an age larger than age_{max}. If this results in points having no emanating edges, remove them as well.

9. If the number of input signals generated so far is an integer multiple of a parameter Γ, insert a new unit as follows:

   - Determine the unit q with the maximum accumulated error.
   - Insert a new unit r halfway between q and its furthest neighbor f: $w_r = 0.5(w_q + w_f)$.
   - Insert edges connecting the new unit r with units q and f, and remove the original edge between q and f.
   - Decrease the error variables of q and f by multiplying them by a constant ϑ. Initialize the error variable of r with the new value of the error variable of q.

10. Decrease all error variables by multiplying them with a constant ν.

11. If a stopping criterion (e.g., net size n_{max} or some performance measure) is not yet fulfilled go to Step 2.

Appendix B

Partially Observable Markov Decision Process

The POMDP models an agent taking sequential actions under uncertainties to maximize a certain reward. Formally it is specified as a tuple $\{S, A, O, T, Z, R, \gamma\}$, where:

- S is the set of all possible states.
- A is the set of all possible actions.
- $T : S \times A \times S \rightarrow [0, 1]$ is the transition function, where $T(s, a, s') = \Pr(s'|s, a)$ represents the probability of ending in state s' if the agent performs action a in the state s.
- $R : S \times A \rightarrow \mathbb{R}$ is the reward function, where $R(s, a)$ is the reward obtained by executing action a in state s.
- Z is the set of all possible observations.
- $O : S \times A \times Z \rightarrow [0, 1]$ is the observation function, where $O(s', a, z) = \Pr(z|a, s')$ gives the probability of observing z if action a is performed and the resulting state is s'.
- $\gamma \rightarrow [0, 1)$ is the discount factor.

In each time step, the agent lies in some state $s \in S$; it takes some action $a \in A$ and moves from s to a new state s'. Due to the uncertainty in the action, the end state s' is modeled as a conditional probability function $T(s, a, s') = \Pr(s'|s, a)$, which gives the probability that the agent lies in s' after taking action a in state s. The agent then makes an observation to gather information on its state. Due to the uncertainty in observation, the observation result $z \in Z$ is again modeled as a conditional probability function $O(s', a, z) = \Pr(z|a, s')$.

Moreover, the agent will receive a real-valued reward $R(s, a)$, if it takes action a in state s, and the goal of the agent is to maximize its expected total reward by choosing a suitable sequence of actions. For infinite-horizon POMDP, the sequence of actions has infinite length, and a discount factor $\gamma \in [0, 1)$ is specified to ensure that the total reward is finite. In this case, the expected total reward is given by $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right]$, where s_t and a_t denote the agent's state and action at time t.

The solution to a POMDP is an optimal policy that maximizes the expected total reward. Normally, a policy is a mapping from the agent's state to a prescribed action. However, the agent's state is partially observable and not known exactly in a POMDP, so the concept of beliefs is defined. A belief is a probability distribution over S. A POMDP policy $\pi : B \to A$ thereby maps a belief $b \in B$ to a prescribed action $a \in A$.

Appendix C

Author's Publications

Journals and Book Chapter

1. Seong-Woo Kim, and **Wei Liu**, "Cooperative Autonomous Driving: A Mirror Neuron Inspired Intention Awareness and Cooperative Perception Approach", IEEE Intelligent Transportation Systems Magazine, 2016 (To Appear)
2. Seong-Woo Kim, **Wei Liu**, Marcelo H. Ang Jr., Emilio Frazzoli, and Daniela Rus, "The Impact of Cooperative Perception on Decision Making and Planning of Autonomous Vehicles", IEEE Intelligent Transportation Systems Magazine, Vol. 7, No. 3, pp. 39 - 50, July 2015
3. Seong-Woo Kim, Baoxing Qin, Zhuang Jie Chong, Xiaotong Shen, **Wei Liu**, Marcelo H. Ang Jr., Emilio Frazzoli, and Daniela Rus, "Multivehicle Cooperative Driving using Cooperative Perception: Design and Experimental Validation," IEEE Transactions on Intelligent Transportation Systems, Vol. 16, No. 2, pp. 663 - 680, April 2015
4. Seong-Woo Kim, Tirtha Bandyopadhyay, Baoxing Qin, Zhuang Jie Chong, **Wei Liu**, Xiaotong Shen, Scott Pendleton, James Guo Ming Fu, Marcelo H. Ang Jr., Emilio Frazzoli, and Daniela Rus "Vehicle Autonomy and Cooperative Perception for Mobility-on-Demand Systems," Book chapter in *Motion and Operation Planning of Robotics System: Background and Practical Approaches*, Springer Verlag, 2015

Conference Papers

1. Scott Pendleton, Tawit Uthaicharoengpong, Zhuang Jie Chong, Guo Ming James Fu, Baoxing Qin, **Wei Liu**, etc., "Autonomous golf cars for public trial of mobility-on-demand service," *IEEE/RSJ International Conference on IntelligentRobots and Systems (IROS)*, October 2015
2. **Wei Liu**, Zhiyong Weng, Zhuang Jie Chong, Xiaotong Shen, Baoxing Qin, Guo Ming James Fu and Marcelo H. Ang Jr., "Autonomous Vehicle Planning System Design under Perception Limitation in Pedestrian Environment", *IEEE International conferences on Cybernetics and Intelligent Systems (CIS)*, July 2015
3. **Wei Liu**, Seong-Woo Kim, Scott Pendleton and Marcelo H. Ang Jr., "Situation-aware Decision Making for Autonomous Driving on Urban Road using Online POMDP", *IEEE Intelligent Vehicles Symposium (IV)*, June 2015
4. Xiaotong Shen, Zhuang Jie Chong, Scott Pendleton, **Wei Liu**, Baoxing Qin, Guo Ming James Fu, and Marcelo H. Ang Jr., "Multi-vehicle Motion Coordination Using V2V Communication", *IEEE Intelligent Vehicles Symposium (IV)*, June 2015
5. **Wei Liu**, Seong-Woo Kim, and Marcelo H. Ang Jr., "Probabilistic Road Context Inference for Autonomous Vehicles", *IEEE International Conference on Robotics and Automation (ICRA)*, May 2015
6. **Wei Liu**, Seong-Woo Kim, and Marcelo H. Ang Jr., "Obstacle Avoidance for Autonomous Driving in Clustered Pedestrian Environment: Framework and Experimental Evaluation," *International Conference on Electronics, Information and Communication (ICEIC)*, January 2015
7. Seong-Woo Kim, **Wei Liu**, Marcelo H. Ang Jr., Seung-Woo Seo, and Daniela Rus, "Cooperative Autonomous Driving using Cooperative Perception and Mirror Neuron Inspired Intention Awareness", *IEEE International Conference on Connected Vehicles and Expo (ICCVE)*, November 2014

8. **Wei Liu**, Seong-Woo Kim, Katarzyna Marczuk, and Marcelo H. Ang Jr., "Vehicle Motion Intention Reasoning using Cooperative Perception on Urban Road", *IEEE International conference on Intelligent Transportation System (ITSC)*, October 2014
9. Seong-Woo Kim, **Wei Liu**, and Katarzyna Marczuk, "Autonomous Parking from a Random Drop Point", *IEEE Intelligent Vehicles Symposium (IV)*, June 2014
10. **Wei Liu**, and Marcelo H. Ang Jr., "Incremental Sampling-based Algorithm for Risk-aware Planning under Motion Uncertainty", *IEEE International Conference on Robotics and Automation (ICRA)*, May 2014
11. **Wei Liu**, Seong-Woo Kim, Zhuang Jie Chong, Xiaotong Shen and Marcelo H. Ang Jr., "Motion Planning using Cooperative Perception on Urban Road", *IEEE International conferences on Robotics, Automation and Mechantronics (RAM)*, November 2013
12. Seong-Woo Kim, Zhuang Jie Chong, Baoxing Qin, Xiaotong Shen, Zhuoqi Cheng, **Wei Liu**, and Marcelo H. Ang Jr., "Cooperative Perception for Autonomous Vehicle Control on the Road: Motivation and Experimental Results", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, November 2013
13. Baoxing Qin, **Wei Liu**, Xiaotong Shen, Zhuang Jie Chong, T. Bandyopadhyay, Marcelo H. Ang Jr., Emilio Frazzoli, and Daniela Rus, “A General Framework for Road Marking Dection and Analysis”, *IEEE International Conference on Intelligent Transportation Systems (ITSC)* , October 2013

Appendix D

Video Links

1. **Incremental Sampling-based Algorithm for Risk-aware Planning under Motion Uncertainty**

   https://www.youtube.com/watch?v=FJ7EhBYp7vw

2. **Probabilistic Road Context Inference for Autonomous Vehicles**

   https://www.youtube.com/watch?v=YC8fR7SYGnE

3. **Situation-aware Decision Making for Autonomous Driving on Urban Road using Online POMDP**

   https://www.youtube.com/watch?v=W37haHhfU34

4. **Driverless Buggies at the Singapore Chinese Gardens**

   https://www.youtube.com/watch?v=aSm027Rzj9E

5. **Launched! SMART-NUS driverless car trials at One-North**

   https://www.youtube.com/watch?v=bIGcG4K2ckc