Motion Planning and Decision-Making for Autonomous Systems: From Quadrotors To Autonomous Vehicles

by

Wenchao Ding

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy
in the Department of Electronic and Computer Engineering

August 2020, Hong Kong

Authorization

I hereby declare that I am the sole author of the thesis.

I authorize the Hong Kong University of Science and Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the Hong Kong University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Wenchao Ding

August 2020

Motion Planning and Decision-Making for Autonomous Systems: From Quadrotors To Autonomous Vehicles

by

Wenchao Ding

This is to certify that I have examined the above Ph.D. thesis
and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by
the thesis examination committee have been made.

Prof. Shaojie Shen, Thesis Supervisor
Department of Electronic and Computer Engineering

Prof. Bertram Emil Shi, Head of Department
Department of Electronic and Computer Engineering

Thesis Examination Committee:

1.	Prof. Shaojie SHEN (Supervisor)	Department of Electronic and Computer Engineering
2.	Prof. Bertram Emil SHI	Department of Electronic and Computer Engineering
3.	Prof. Wei ZHANG	Department of Electronic and Computer Engineering
4.	Prof. Xiaojuan MA	Department of Computer Science and Engineering
5.	Prof. Jianru XUE (External Examiner)	Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong Univeristy

Department of Electronic and Computer Engineering
The Hong Kong University of Science and Technology
August 2020

ACKNOWLEDGMENTS

This thesis would never be completed without the help of many people. First of all, I would like to express my sincere thanks to my supervisor, Professor Shaojie Shen. I did not major in automation/cv nor any domain related to robotics before my Ph.D. Instead, I majored in Wireless Communication in my Bachelor and the first year of my Ph.D. study. However, when I started my second year, I became confused, because I found out that although I love math, I do not like pure mathematical derivations that are far away from being applied to the real world. Keeping working on wireless communication problems seemed not to steer my Ph.D. study effectively. Therefore, I decided to switch to another area which makes me excited. I had a hard time checking out the research of different professors and finally, I found robotics. I was fascinated by real-world robots that have "math" really running on it. However, as you can imagine, I was a freshman in robotics with an almost mutually uncorrelated background. However, Prof. Shen accepted me and guided me through the hardship. Without him, there wouldn't be this man with a communication background defending his robotics PhD.

I thank the other members of my thesis committee, Professor Bert Shi, Professor Wei Zhang, Professor Xiaojuan Ma, Professor Jianru Xue. Scheduling the examination should have been one of the challenges of the graduation, but with their help and kind consideration, scheduling is far simpler than I expected.

I also would like to thank my colleagues in the HKUST Aerial Robotics Group. I thank Lu Zhang who has been working closely with me. We encountered many challenges but we never gave up. I thank Peiliang Li and Kaixuan Wang. We work, play, and graduate together. I enjoyed doing experiments with Wenliang Gao. And I also enjoyed playing badminton with Siqi Liu, Shaozu Cao, Luqi Wang. My daily life would be plain without communicating with Fei Gao, Tong Qin, Kejie Qiu, William Wu, Jieqi Shi, Chuhao Liu, Pan Jie, Boyu Zhou, Yi Lin, and Ke Wang.

Last but not least, I thank my family and my girlfriend Jieru Zhao. They have been with me, through all the ups and downs.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Motion Planning and Decision-Making for Autonomous Systems: From Quadrotors To Autonomous Vehicles

by

Wenchao Ding

Department of Electronic and Computer Engineering

The Hong Kong University of Science and Technology

Abstract

Autonomous systems have been widely deployed in various field applications, such as inspection, exploration, and aerial videography using micro aerial vehicles (MAVs), and highway pilot (HWP), rush-hour pilot (RHP), robo-taxi in field of intelligent vehicles. For accomplishing such missions in challenging environments, navigation functionality is essential.

In this thesis, we study practical and efficient planning methods for autonomous systems, which can achieve full autonomy in real-world complex environments. Our study covers both low-level motion planning problems and high-level decision-making (i.e., behavior planning) problems, targeting for safe and smooth autonomous navigation with minimum user intervention. We investigate the navigation problem for two different platforms, namely, quadrotors for MAVs, and autonomous vehicles (AVs), which have significant practical impacts while illustrating different focuses and functionalities.

We start with a novel kinodynamic motion planning method for quadrotors, which achieves safe and efficient replanning in unknown cluttered 3-D environments. Leveraging the knowledge of motion planning for quadrotors, we further study the planning problem for autonomous vehicles in highly dynamic environments. Different from the motion planning for quadrotors, planning for AVs requires reasoning about other dynamic traffic participants and social compliance. To this end, a novel behavior prediction method is proposed to understand the behaviors of

other traffic participants in the real world. The behavior prediction is then tightly incorporated into our proposed decision-making framework which generates robust decisions efficiently. The output decision is subsequently utilized by our novel motion planning module to generate a smooth and safe trajectory for closed-loop execution. We assemble all the individual modules into a complete and robust planning system which is validated in real-world dense city traffic.

CHAPTER 1

INTRODUCTION

The development of mechanization and autonomy has always been the source of the industrial revolution. New machine tools have boosted production efficiency in a wide range of field applications, such as industrial machines, automobiles, etc. Unlike robots that work in simple and dedicated environments (e.g., manipulators in the industry), mobile robots need to work in fast-changing environments. There are numerous kinds of mobile robots emerging nowadays, such as micro aerial vehicles (MAVs), micro-ground vehicles (e.g., sweeping robots), autonomous vehicles (AVs), etc. Onboard perception and planning capabilities are essential for mobile robots to ensure their general applicability.

In this thesis, we focus on planning for *fully* autonomous navigation in complex environments. Specifically, we focus on two typical autonomous systems, namely, MAV and AV. The problem which planning needs to solve can be concluded in one sentence: how to accomplish the user-input task *safely*, *smoothly* and *optimally* with *minimum intervention*? For example, we may require MAVs to inspect some narrow, confined environments for us. Or we input our destination and require AVs to take us to the destination without any human intervention. The first term "safely" is straightforward: the robot should not collide with any obstacle even faced with unknown, cluttered, or changing environments. The second term "smoothly" can be interpreted differently depending on the application. For example, in autonomous driving, this term poses comfort constraints, while for micro aerial robots this term represents that the planned trajectory should be smooth enough so that the controller can follow. The third term "optimally" requires an elegant plan. For instance, the trajectory for MAVs should be energy-efficient, while the decisions of AVs should illustrate enough intelligence (e.g, automatically overtakes when appropriate). The fourth term "minimum intervention" means that the planning system should handle all the circumstances which the robot will potentially encounter in the operational design domain (ODD). This is the ultimate goal of using an autonomous system, namely, "click, go and done", and this is exactly where the working efficiency and user-experience come from.

To this end, we target at building complete planning systems for MAV and AV to approach the goal of fully autonomous navigation. To build such systems, several research problems are raised and solved in this thesis. Specifically, navigating quadrotors smoothly in unknown

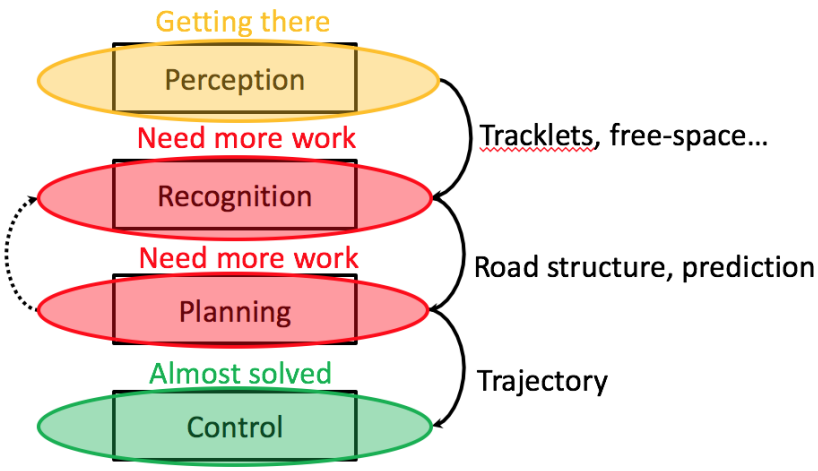


Figure 1.1: Illustration of the typical hierarchical structure of an autonomous system. For quadrotors, the perception layer includes depth sensing (i.e., static obstacles) and 3-D map reconstruction, while the recognition layer may include trajectory prediction for tracked objects. For autonomous vehicles, the perception layer may further include free-space detection, traffic signal detection, etc, while the recognition layer includes behavior and trajectory prediction for traffic participants. The control layer for both quadrotor and autonomous vehicle has the same meaning, namely, tracking the output trajectory of the planning layer as accurate as possible.

cluttered 3-D environments requires an efficient motion planning system. While for AVs, the planning system is more complex and consists of prediction, decision-making, and motion planning modules to adapt to dynamic multi-agent environments. In the following, we first discuss the related research problems and our solutions are elaborated in the following chapters.

1.1 Research Problems

We start with the motion planning system for MAVs, which has a straightforward goal. Specifically, the planning system is required to constantly generate smooth and dynamically feasible trajectories for quadrotors in unknown 3-D environments. When encountering unforeseen obstacles, the planning systems should replan timely and automatically to avoid the obstacles in a safe and optimal manner. The abovementioned system is a typical motion planning system, which can serve as a good starting point for this thesis.

As far as autonomous vehicles are concerned, other traffic participants are involved. These traffic participants can be stochastic and nosily rational while perceiving these participants also suffers from noises and errors. To understand the behaviors of other participants, prediction is required. Moreover, apart from motion planning, decision-making is more critical for AVs, since competing and cooperating with other participants are required. To this end, we divide the

whole planning system into several components, namely, behavior prediction, decision-making, and motion planning. We specify the research problems for these aspects one by one and finally we integrate these modules into a complete planning system for autonomous vehicles.

At a high level, although quadrotors and autonomous vehicles may seem quite different, they share a similar hierarchical structure as shown in Fig. 1.1. For quadrotors, the perception layer mainly focuses on sensing static obstacles and reconstructs accurate 3-D map for the environments, while the planning layer basically focuses on the low-level motion planning. On the other hand, for autonomous vehicles, the perception layer may includes broader categories including perceiving the traffic signals, road structures, dynamic objects, etc. At the same time, since autonomous vehicles need to drive in social environments, they require a recognition layer to identify and understand complex social behaviors. For example, the recognition layer may include behavior and trajectory prediction for other traffic participants. Moreover, since autonomous vehicles need to accomplish complex user-input tasks (e.g., highway pilot, rush-hour pilot, shared control), their planning system requires high-level behavior planning to facilitate accomplishing the high-level goals specified by users.

Generally speaking, the perception layer is "getting there". The reason is that from algorithmic perspective, the perception algorithms are becoming mature in terms of accuracy, latency and robustness. However, how to guarantee the perception quality with low-cost sensor suites remains a challenging problem. For example, with high-end lidars, depth sensing and object tracking can be extremely accurate. However, high-end lidars are too expensive to be deployed widely in industrial products. Perception with low-cost sensors such as cameras and noisy inertial measurement units (IMUs) is the current research direction. Honestly speaking, the recognition and planning layer are quite preliminary in the current stage. The algorithms are far from becoming mature and no consensus is reached. As a result, autonomous vehicles nowadays still seem "stupid". The control layer is mature since the control algorithms have been mature for decays. The remaining challenges mainly involve controlling complex systems such as a truck with payload.

To summarize, in this thesis we mainly focus on the two layers we mark as "need more work", as in Fig. 1.1, namely, the recognition layer and planning layer. For the recognition layer, we study the behavior prediction problem for autonomous vehicles in Chap. 3. For the planning layer, we study the high-level decision-making problem in Chap. 4 while the low-level motion planning problems in Chap. 2 and Chap. 5.

1.1.1 Motion Planning for Quadrotors in Unknown 3-D Environments

Consider a typical application that we need the quadrotors to complete an inspection task with full autonomy. We can derive two basic requirements for this problem. First, the environment is probably unknown beforehand. Second, the quadrotor may need to deal with narrow and confined 3-D environments such as inspection during disaster response. Considering these two problems together, the quadrotor is required to *replan* constantly when encountering unforeseen obstacles. To this end, we focus on trajectory replanning for quadrotors. However, replanning is a challenging task. First of all, the replanning should be highly efficient especially during fast autonomy. As such, heavy-weight motion planning methods can not be adopted. Second, replanning typically starts from nonstatic initial states, i.e., the quadrotor is not static for replanning. This poses additional feasibility constraints for planning. Many previous works [5–9] may be problematic when faced with nonstatic initial states. To this end, kinodynamic replanning is proposed in this thesis to accomplish efficient dynamically feasible replanning.

1.1.2 Behavior Prediction for Autonomous Vehicles

When coming to the problem of autonomous driving, we have to consider other traffic participants. However, traffic participants are known to be stochastic, noisily rational, and hard to predict. Therefore, it is essential to build behavior models for traffic participants. During online execution, through the inference process based on past observations, we can interpret the future motions of other traffic participants. However, building such behavior models is complex due to the multi-modal nature of the problem (e.g., thinking about predicting a vehicle in a uncontrolled intersection). Moreover, the uncertainty of the model increases dramatically as the prediction horizon increases. The problem is that, for planning purpose we need long-term prediction up to 5 to 10 s, while prediction is only accurate for short term (less than 3 s) in many existing works [10–13]. To this end, we require accurate long-term prediction while accounting for the multi-modal nature of the behaviors. And in this thesis, we work towards extending the prediction horizon by modeling additional multi-agent interaction information and build a learning-based behavior prediction pipeline.

1.1.3 Decision-making for Autonomous Vehicles

After building up the understanding of other traffic participants, the remaining problem is how to make intelligent decisions to interact, cooperate, or even compete with them. This problem

can be characterized as decision-making in a multi-agent setting. When considering onboard autonomous driving, the problem becomes more complex. Since for onboard driving, all the observations come from onboard sensors, while sensing is hardly perfect. Various restrictions (e.g., occlusion, shading) impose errors in perception, which will be then propagated to prediction. As a result, from the decision-making perspective, the prediction received may be uncertain. The core problem is how to efficiently conduct safe decision-making in uncertain and multi-agent environments. In this thesis, we propose an efficient uncertainty-aware decision-making framework to tackle the above challenges.

1.1.4 Motion Planning for Autonomous Vehicles

The difference between decision-making and motion planning is that decision-making cares about a rough course of behavior plan which can be typically represented by a preliminary sequence of states, while the role of motion planning is to generate a smooth and safe trajectory in the local feasible solution space surrounding the initial guess provided by the decision-making. Unlike quadrotors which typically operate in static or semi-static environments, the challenge for motion planning of autonomous vehicles is to deal with dynamical obstacles. To this end, it is essential to take temporal information into account. Moreover, constraints or cost engineering has been a headache for motion planning, since there are various constraints imposed by the semantics of the environments such as lane geometry and traffic rules. We require a unified trajectory generation process which is easy to tune and seamlessly adapt to diverse environments. In this thesis, we propose a spatio-temporal semantic corridor structure to achieve this goal.

1.1.5 System Integration for Autonomous Vehicles

Based on our studies on individual components, it is time to assemble these modules into a complete planning system. In academia, a lot of planning systems have been proposed claiming various kinds of novelties. However, few of them are validated on a real autonomous vehicle. In industry, various demos are presented showing promising autonomy, but few technical details are published. For autonomous driving, the applicability of a planning system cannot be verified without onboard experiments. The reason is that real-world driving is constantly different from simulation and dataset, no matter how multi-modal and accurate the simulation or dataset is. To this end, we require verifying the whole planning system on a real autonomous vehicle with purely onboard sensors, and achieve closed-loop driving in dense city traffic. In this thesis, we achieve this goal leveraging all the advances made in this thesis.

1.2 Thesis Overview

The whole thesis is devoted to answer the following question:

> How to accomplish the user-input task *safely*, *smoothly* and *optimally* with *minimum intervention*?

This question is repeatedly raised throughout this thesis and will be answered in detail in the following chapters.

We start this thesis by introducing our motion planning framework for quadrotors in unknown complex 3-D environments (Chap. 2). The planning system is implemented on a quadrotor with only one monocular camera and one IMU. The system is extensively validated in both simulation and real-world experiments.

Based on the understanding of motion planning for quadrotors, we turn to the planning system for autonomous vehicles. Unlike planning for quadrotors, motion planning module is only one particular component for the planning system of autonomous vehicles. Before stepping into the motion planning for autonomous vehicles, we first introduce behavior prediction for autonomous vehicles (Chap. 3) which is employed to model traffic participants for driving. This behavior prediction module is further incorporated into the decision-making module (Chap. 4) which provides high-level decisions for the ego (controlled) vehicle in interactive driving environments. The interesting part is that our decision-making module (Chap. 4) can account for uncertain predictions provided by (Chap. 3), which is critical in real-world driving. Since in real-world, perfect prediction never exists. A robust decision-making system has to take the responsibility of tolerating uncertainties of upstreaming modules while still being able to ensure safety.

Given the decision, the remaining problem is to generate a smooth and safe trajectory which faithfully follows the decision. A motion planning module for autonomous vehicles is then introduced in Chap. 5. Having all the individual components, in Chap. 6 we assemble them into a complete planning system, and conduct field tests in real world to verify its effectiveness. The story is then finished and conclusions are drawn in Chap. 7, where we also point out promising future directions for the planning community.

1.3 Thesis Contributions

This thesis contributes to several aspects of motion planning and decision-making for two autonomous systems, i.e., MAV and AV. The contributions are not only algorithms but also complete system designs and extensive real-world validations.

In Chap. 2, we propose a complete and efficient kinodynamic replanning for quadrotors. This framework is designed to overcome the problem that traditional methods often generate non-smooth or even infeasible trajectory under non-static initial states. The kinodynamic search we propose provides superior performance in terms of the optimality and efficiency tradeoff compared to several state-of-the-art methods. Moreover, we implement the whole framework on two different vision-based platforms: one quadrotor with one camera and one IMU (monocular) and the other with dual-fisheye vision. Although the perception quality and field of view are quite different for two platforms, our planning framework consistently provides safe and smooth performance in real-world unknown 3-D environments.

In Chap. 3, we illustrate our first step into the field of autonomous driving. Before considering planning for autonomous vehicles, we have to understand the driving environments at the first hand. In this chapter, we propose a learning-based behavior prediction method for autonomous vehicles. This method is dedicated to solving the problem that the prediction horizon of existing methods is insufficient. However, planning requires long-term prediction. To this end, we propose utilizing interaction information among agents to introduce an additional clue for prediction and achieve extending the prediction horizon. Our method has superior accuracy and a longer prediction horizon compared to several state-of-the-art methods. We validate the method using a real-world highway driving dataset.

In Chap. 4 and Chap. 5, we finally approach the problem of planning for autonomous vehicles after obtaining the understanding of other traffic participants. In Chap. 4, we propose an efficient uncertainty-aware decision making (EUDM) framework. EUDM is designed to be robust to uncertain predictions while being highly efficient, which addresses two key concerns of decision making (i.e., efficiency and uncertainty). In Chap. 5, we introduce how to realize a particular decision to an executable trajectory. We propose a unified abstraction which wraps various constraints and obstacles in the driving scenario, i.e., the spatio-temporal semantic corridor (SSC). With the help of SSC, the trajectory generation problem boils down to a straightforward optimization formulation. Moreover, the SSC makes the formulation free of tuning, which is the key challenge for motion planning for autonomous vehicles.

In Chap. 6, we assemble a complete planning system for autonomous vehicles, based on our accumulated contributions. The key feature of our system that we not only have a systematic and consistent design but also validate our system in real-world city traffic. Although many existing works have been published on the planning for autonomous vehicles, few of them are validated in the real world. In this Chap. 6 we validate the real-world impact of our planning methods.

CHAPTER 2

KINODYNAMIC MOTION PLANNING FOR QUADROTORS

2.1 Scientific Background and Literature Review

2.1.1 Background

Autonomous navigation for quadrotors in unknown environments has gained significant interest for its practical usage in various inspection and exploration tasks. To fulfill the need of fully autonomous exploration in unknown environments, trajectory replanning is of great significance. Replanning requires a real-time response to unexpected obstacles to guarantee safety while satisfying the low-level feasibility constraints induced by the non-trivial dynamics.

Many existing methods [5–9] tackle this challenging problem using a hierarchical framework, which first finds a geometric path and then locally optimizes the path to a dynamically feasible trajectory with respect to a given time allocation. Although this framework is efficient, inadequacy exists between the path finding and the local path parameterization. Specifically, the parameterization may be restricted by the geometric path to a homotopy class that does not contain a globally optimal (or even feasible) solution, especially when faced with non-static initial states (non-zero velocities or any other higher-order derivatives), as shown in Fig. 2.1.

To address the limitations of the hierarchical methods, it is essential to use kinodynamic motion planners, which directly find time-parameterized trajectories that are globally optimal with respect to control efforts and dynamical limits. The incorporation of kinodynamic planners into replanning facilitates dealing with non-static initial states and enhances replanning consistency.

Sampling-based motion planning algorithms, such as rapidly-exploring random trees (RRTs) [14] and their variants [15–19], are popular in the kinematic/kinodynamic planning literature. Asymptotical optimality has been proved for some of them [15–18]. However, when applied to complex kinodynamic systems, they typically require solving a computationally expensive nonlinear two-point boundary value problem (BVP) [20, 21] and cannot run in real-time. Liu *et al.* [22] explored a search-based counterpart and proposed a heuristic-guided resolution-

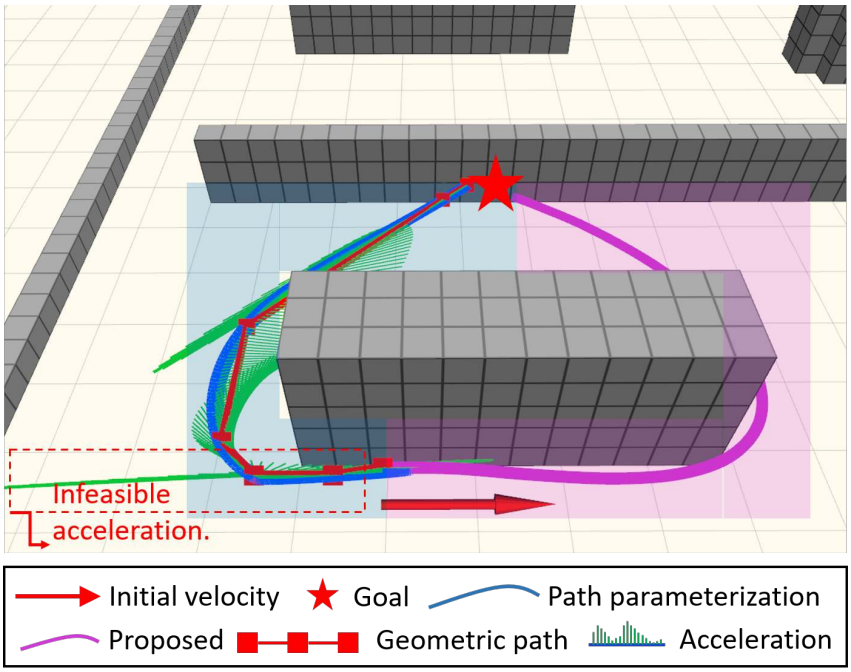


Figure 2.1: Illustration of the motivating example. The initial state has non-zero velocity (*red arrow*). The traditional geometric planner finds the shortest path (red squares) and then parameterizes it using a piecewise polynomial (*blue*). However, the local path parameterization is restricted to a homotopy class (*blue* area), and the resultant trajectory is jerky (even infeasible) with respect to the given time allocation. In contrast, our framework produces a dynamically feasible trajectory (*purple line*) using kinodynamic planning.

complete (optimal with respect to discretization) search method using linear quadratic minimum time control. However, for high-order kinodynamic systems or a large dynamic range, the run-time efficiency is inadequate.

In this chapter, we present a kinodynamic replanning framework which addresses the efficiency bottleneck. Our framework starts with an efficient B-spline-based kinodynamic (EBK) search algorithm, which finds B-spline control points on a spatial grid. We introduce the novel concept of vertex tuple to keep the search problem simple and analyzable, which enables a thorough theoretical characterization of the problem. Built on top of the structure of the optimal solution, a graph aggregation technique is proposed to minimize the computation time through a controllable discretization of the search space. An offline-computable minimum inflation is adopted to avoid unnecessary collision checking and further accelerate the online search. Compared to state-of-the-art methods (Sect. 2.6.2), our kinodynamic search finds the lowest-cost dynamically feasible trajectories in real time.

To compensate for the discretization in the search, an elastic optimization (EO) approach is proposed to refine the control point placement to the optimal location, by solving a convex quadratically constrained quadratic programming (QCQP) problem. The two components are

integrated into a receding horizon replanner based on the local control property of the B-spline.

Our replanning framework is not only theoretically analyzed, but also validated in simulated and onboard experiments. Superior performance is shown through comprehensive comparisons against the state-of-the-art kinodynamic planning and trajectory optimization methods. Moreover, the practical impact of our framework is verified through onboard experiments using a monocular vision-based quadrotor and a dual-fisheye vision-based quadrotor in unknown indoor and outdoor environments. We summarize our contributions as follows:

- An EBK search algorithm which provides an initial-state-aware dynamically feasible time-parameterized trajectory.
- An EO approach that refines the control point placement to the optimal location while preserving the safety and dynamical feasibility.
- Systematic comparisons against the state-of-the-art showing the superior performance of the proposed framework.
- Integration of our framework into a real monocular vision-based quadrotor and a dual-fisheye vision-based quadrotor as well as extensive experiments demonstrating fully autonomous navigation in unknown, complex indoor and outdoor environments.

A basic version of our framework was originally presented in [23], where we introduced the real-time B-spline-based kinodynamic (RBK) search. Although the RBK search achieves high computational efficiency, the absence of theoretical optimality analysis in [23] limits confidence in its solution quality and further limits its theoretical impact. In this chapter, instead of directly developing efficient methods, we tackle the kinodynamic search problem in a systematic way: 1) We first characterize the complexity and optimal solution of the search problem using a novel vertex tuple structure. 2) We then establish the quality-efficiency tradeoff using a novel graph aggregation technique, which provides a user-specified parameter to control algorithm efficiency and solution quality. The above two theoretical additions render the more flexible and theoretically reliable EBK search. It also turns out that the preliminary version in [23] is perfectly contained in the EBK search. Apart from the theoretical additions, more comprehensive experimental analyses in a wide variety of environments are presented to support the new characteristics.

The relevant literature is discussed in Sect. 2.1.2. An overview of the proposed replanning framework is provided in Sect. 2.2. Mathematical background and advantageous properties of

B-spline are introduced in Sect. 2.3. The problem formulation and algorithm detail of the EBK search are elaborated in Sect. 2.4. The EO approach is presented in Sect. 2.5. Implementation details are given in Sect. 2.6.1. Systematic comparisons against the state-of-the-art methods are provided in Sect. 2.6.2, and onboard experimental results are illustrated in Sect. 2.6.3. Finally, a conclusion and further possible research directions are provided in Sect. 3.5.

2.1.2 Related Works

There is extensive literature on motion planning techniques for quadrotors from various perspectives, such as control-based methods [24–26], search-based methods [22, 27–29], sampling-based methods [15, 17, 30–32] and optimization-based methods [6, 9, 33, 34]. It is difficult to give a full literature review of all these techniques, so in this section, we choose the most relevant and organize them into two categories, namely, hierarchical motion planning techniques and kinodynamic motion planning techniques.

Hierarchical motion planning refers to a high-level geometric path planner coupled with a low-level time parameterization scheme. The high-level geometric planner is concerned with finding an obstacle-free path, while the low-level parameterization scheme takes care of the vehicle dynamical constraints and generates a time-parameterized trajectory for execution. For quadrotors which have non-trivial dynamics, directly generating a trajectory in the high-dimensional state space is time consuming, while smoothing a given geometric path is computationally efficient (with a suitable relaxation) [35]. As such, the hierarchical framework is popular for quadrotors, and it enables a number of online methods [5–9].

Two pioneering works [5, 7] extract waypoints from the geometric path and formulate the trajectory generation problem as quadratic programming (QP) on polynomial coefficients. These methods are based on the differential flatness of the quadrotor [5]. Due to the deviation of the polynomial trajectory from the straight-line collision-free path, an iterative waypoint insertion scheme is adopted [7]. However, how many additional waypoints are needed is not quantified. Chen *et al.* [9] propose a corridor-based geometric planner based on the octree-based map structure [36]. The control effort can be reduced by generating the trajectory in a series of connected cubes. Apart from that, they propose an iterative process of adding constraints on polynomial extremas to cope with the deviation from the corridor, and prove that a finite number of iterations is needed to guarantee safety. Liu *et al.* [8] further generalize the corridor representation to a series of connected convex polygons.

Although the hierarchical methods have made significant achievements, they suffer from the

common problem that the geometric planner is unaware of the vehicle dynamics, resulting in inadequacy between the path planning and path parameterization, especially when faced with non-static initial states. The example in Fig. 2.1 motivates us to explore the problem from the kinodynamic planning perspective.

The kinodynamic motion planner directly explores the high-dimensional state space, and outputs a time-parameterized trajectory, which fundamentally avoids the inadequacy between path planning and parameterization. RRTs [14] and their variants [15–19, 30] were originally designed for kinematic systems and can be easily extended to kinodynamic systems. These methods provide an efficient way of exploring the high-dimensional state space, and some of them possess asymptotical optimality [15–18]. However, for robots with non-trivial dynamics, the tree expansion typically involves solving the BVP, which is non-linear and challenging. Webb *et al.* [15] propose a fixed-final-state-free-final-time optimal controller which solves the BVP for linear (or linearized) controllable systems in closed form. Xie *et al.* [20] propose an efficient BVP solver for general kinodynamic systems using sequential quadratic programming (SQP). Li *et al.* [21] work in another direction, namely, expanding the tree using random control propagation, for cases where system models are complex and BVP solvers are not available.

Despite the fact that the efficiency of the kinodynamic planning techniques keeps improving [20, 21], it is still prohibitively expensive for replanning. Allen *et al.* [31] work towards a real-time kinodynamic planning framework by combining FMT* [18] with a support vector machine (SVM) for the classification of the reachable set. This framework [31] reduces the calling of the BVP solver to gain efficiency. However, the solution quality largely depends on the number of states pre-sampled. On the other hand, Liu *et al.* [22] explore the search-based kinodynamic planning counterpart and develop efficient heuristics by solving a linear quadratic minimum time problem. Their solution is resolution-complete with respect to the discretization on the control input, and achieves near real-time performance. Note that both [22] and [31] use a simplified system model, i.e., a double or triple integrator, to reduce the computation complexity. However, the resultant trajectory only has limited continuity. To improve the smoothness, both [22] and [31] adopt trajectory reparameterization using the unconstrained QP formulation [7], which may break the dynamical feasibility and safety.

In contrast, the proposed EBK search adopts a high-order B-spline parameterization with continuity up to snap, which can be directly used to control the quadrotor. Moreover, the advantageous properties of the B-spline facilitate the kinodynamic replanning as follows:

- Local control property for incrementally constructing the B-spline trajectory in the kino-

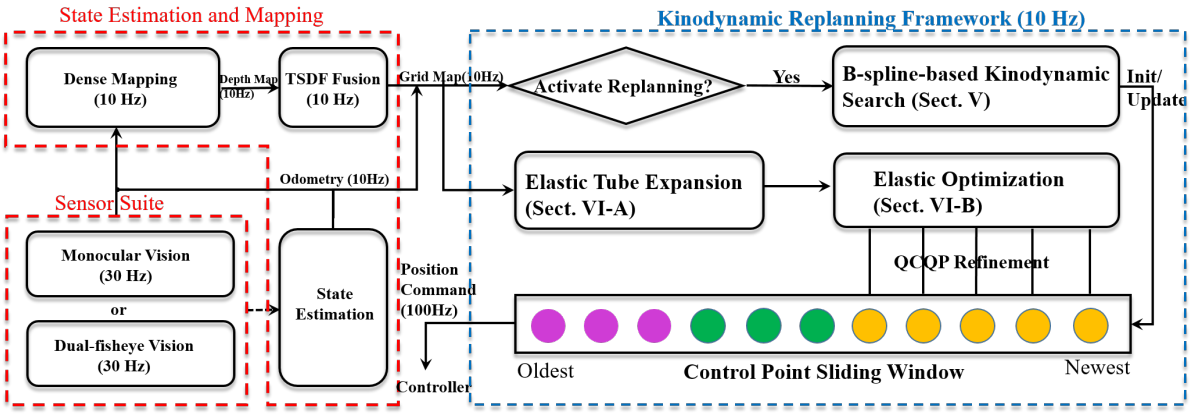


Figure 2.2: A diagram of our kinodynamic replanning framework together with state estimation and mapping modules.

dynamic search and local refinement of the trajectory during replanning.

- Convex hull property for enforcing collision-free constraints and providing a dynamical feasibility guarantee for the entire trajectory.

Given the same run-time budget according to the real-time requirement, the EBK search finds a lower-cost trajectory, as validated by the comprehensive comparisons against the state-of-the-art in Sect. 2.6.2. Apart from this, the proposed refinement module, namely, the EO approach, can preserve the safety and dynamical feasibility by taking advantage of the convex hull property of the B-spline.

2.2 System Overview

The structure of the proposed framework is shown in Fig. 2.2. The replanning framework is built on top of the state estimation and dense mapping module, which are discussed in Sec. 2.6.1. The frequency of the grid map update is 10 Hz. As such, the real-time requirement in this chapter refers to a run-time of less than 100 ms. The updated map and the initial state of the quadrotor are fed to the EBK search module, and the replanning strategy is elaborated in Sect. 2.6.1. The control points are constantly refined by the proposed EO approach (Sect. 2.5), which consists of an elastic tube expansion module (Sec. 2.5.1) for free space characterization and a convex optimization formulation (Sec. 2.5.2) for trajectory refinement. The confirmed control points are evaluated, and position commands are generated accordingly.

2.3 B-spline Curve and Replanning

For the proposed kinodynamic planning framework, we adopt a B-spline parameterization for its advantageous properties, namely, local control and convex hull property, and we further adopt the uniform B-spline for its convenient closed-form evaluations. In this section, we elaborate these properties and explain how they can be applied to the replanning system.

Given $n+1$ control points $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n$ and knot vector $\{t_0, t_1, \ldots, t_m\}$, the B-spline curve $\mathbf{s}(t)$ of degree k is defined as follows:

$$\mathbf{s}(t) = \sum_{i=0}^{n} \mathbf{p}_i N_{i,k}(t), \tag{2.1}$$

where $N_{i,k}(t)$ is the B-spline blending function of degree k, which can be evaluated recursively as follows:

$$\begin{aligned} N_{i,0}(t) &= \begin{cases} 1 & \text{if } t_i \le t < t_{i+1} \\ 0 & \text{otherwise} \end{cases} \\ N_{i,k}(t) &= \frac{t - t_i}{t_{i+k} - t_i} N_{i,k-1}(t) + \frac{t_{i+k+1} - t}{t_{i+k+1} - t_{i+1}} N_{i+1,k-1}(t). \end{aligned} \tag{2.2}$$

The total number of knots should satisfy $m + 1 = n + k + 2$. The uniform B-spline is a special type of B-spline whose knot vector is uniformly distributed. Suppose the knot vector is separated with equidistance Δ_t. The half-open interval $[t_i, t_{i+1})$ is called the $i-$th knot span. We normalize each knot span using $u = (t - t_i)/\Delta_t$, and for the i-th knot span, only $k + 1$ blending functions are non-zero, corresponding to $k + 1$ control points $\mathbf{p}_{i-k}, \ldots, \mathbf{p}_i$. We stack the $k + 1$ control points and call the stacked coordinate matrix a control point span $\mathbf{P}_{i-k} \coloneqq [\mathbf{p}_{i-k}\, \mathbf{p}_{i-k+1} \cdots \mathbf{p}_i]^\intercal \in \mathbb{R}^{(k+1)\times 3}$. Since the blending functions $N_{i,k}(t)$ are shifted versions of each other for the uniform B-spline, we have closed-form matrix representations [37] for parametric evaluation. Let $j = i - k$, and the position and the derivatives of the B-spline curve corresponding to the j-th control point span can be evaluated as follows:

$$\frac{d\mathbf{s}_j(u)}{d^l u} = \frac{1}{(\Delta t)^l} \frac{d\mathbf{b}^\intercal}{d^l u} \mathbf{M}_k \mathbf{P}_j, \tag{2.3}$$

where l denotes the order of the derivative ($l = 0$ means the position), $\mathbf{b} = \begin{bmatrix} 1\, u\, u^2 \cdots u^k \end{bmatrix}^\intercal \in \mathbb{R}^{k+1}$ denotes the basis vector, and $\mathbf{M}_k = (m_{i,j}) \in \mathbb{R}^{(k+1)\times(k+1)}$ denotes the blending matrix, where $m_{i,j} = \frac{1}{k!}\binom{k}{k-i} \sum_{s=j}^{k} (-1)^{s-j} \binom{k+1}{s-j} (k-s)^{k-i}$. According to Eq. 2.3, the evaluation of the derivatives of the B-spline curve can be expressed by a linear matrix multiplication in terms

of the control point span $\mathbf{P}_j$. This chapter uses a quintic uniform B-spline ($k = 5$) to ensure the continuity up to snap for controlling quadrotors.

As described by Mellinger [5], the control cost of a quadrotor is closely related to the integral over squared derivatives of the planned trajectory, which can also be evaluated in closed form in the case of the uniform B-spline. The total control cost E_j^l of the j-th control point span can be expressed by the integral over the squared derivatives of degree l (e.g., for the min-snap trajectory, $l = 4$) as follows:

$$E_j^l = \int_0^1 \left(\frac{d\mathbf{s}_j(u)}{d^l u} \right)^2 du = \mathbf{P}_j^\intercal \mathbf{M}_k^\intercal \mathbf{Q}_l \mathbf{M}_k \mathbf{P}_j, \tag{2.4}$$

where $\mathbf{Q}_l = \int_0^1 \left(\frac{d\mathbf{b}}{d^l u}\right) \left(\frac{d\mathbf{b}}{d^l u}\right)^\intercal du/(\Delta_t)^{2l-1}$ is the Hessian matrix of the l-th squared derivative, which is constant for the uniform B-spline. The control cost E_j^l is quadratic with respect to the control point span $\mathbf{P}_j$. Note that the cost evaluation of a span only depends on the stacked control point coordinates of this span.

2.3.1 Local Control Property and Replanning

The *local control* is one of the important properties of B-spline, making it suitable for replanning. Specifically, the evaluation of any point of the B-spline curve is controlled by a single control point span containing $k + 1$ control points, and any control point only affects $k + 1$ control point spans. We incorporate the local control property into a receding horizon (re-)planner, and we divide the planned trajectory into three types, namely, executed trajectory, executing trajectory and optimizing trajectory. The executed trajectory means the part of the trajectory which has already been executed, the executing trajectory means the part of the trajectory corresponding to the control point span being executed, and the optimizing trajectory means the part of the trajectory whose supporting control points are potentially under optimization.

Thanks to the local control property, modification of the supporting control points of the optimizing trajectory will not affect the evaluation of the executing trajectory, as shown in Fig. 2.3. Unlike [38] and [39] where local reshaping may cause the violation of dynamical constraints, the dynamical feasibility of the executing trajectory can be preserved by leveraging the local control property. Moreover, the locality also makes it possible to optimize any subset of control points without re-generation of the whole trajectory, which is computationally efficient. The locality also helps to preserve a smooth trajectory since the next executing control point

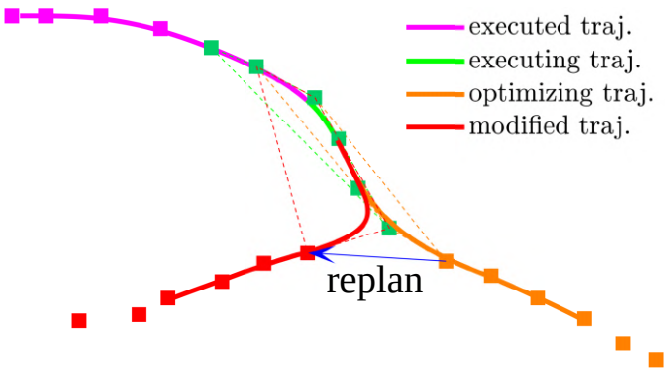


Figure 2.3: Illustration of the B-spline local control property and its application to the replanning system. The control points are shown by squares. The control points corresponding to the executing trajectory are shown in *green*. The original locations of the control points ahead of the executing control point are in *orange*, and their subsequently modified positions are marked in *red*. Due to the locality of the B-spline, the replan causes no perturbation to the executing trajectory.

span always shares k control points with the current executing control point span, yielding k-th-order continuity and a consistent trajectory.

2.3.2 Convex Hull Property and Dynamical Feasibility

Another important property of the B-spline is the *convex hull* property. A B-spline (or Bèzier [40]) trajectory is strictly bounded inside the convex hull supported by the corresponding control point span. Strictly speaking, dynamical feasibility should be induced by the robot's kinematic and dynamical constraints. Given polynomial/spline parameterization, a common practice to enforce dynamical feasibility is to use maximum velocity and maximum acceleration bounds [8, 41, 42]. We follow this practice in this chapter. Note that for piecewise polynomial parameterization methods [6, 8], the dynamical feasibility constraints are enforced on a finite number of checkpoints. Denser checkpoints will enhance the robustness but yield higher computation complexity. In contrast, by using the convex hull property, the *entire* velocity and acceleration profile can be strictly bounded.

We utilize the fact that the derivative of the B-spline of degree k is a B-spline of degree $k-1$, which also enjoys the convex hull property. Therefore, if the supporting control points are bounded inside the convex hull expanded by the allowed maximum derivative, the derivative spline is subsequently bounded, as elaborated in Prop. 1. Note that Prop. 1 is a sufficient but not necessary condition. A toy example illustrating the relation between the convex hull property and dynamical feasibility is shown in Fig. 2.4.

Proposition 1. *Given a uniform B-spline of degree k and knot separation Δ_t, there exists a constant linear combination $\mathbf{S}_l = \mathbf{M}_k^{-1}\mathbf{C}_l\mathbf{M}_k/(\Delta_t)^l \in \mathbb{R}^{(k+1)\times(k+1)}$ such that $u_{l,D}^{min}\mathbf{1}_{(k+1)\times 1} \leq$*

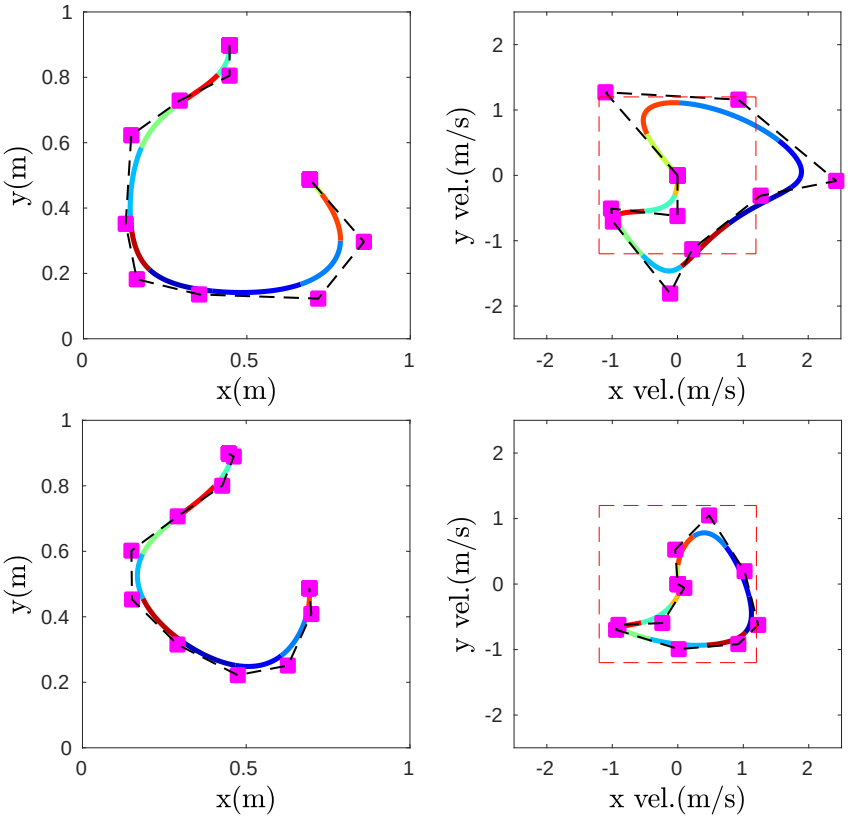


Figure 2.4: Illustration of the convex hull property. The dashed red box shows the feasible velocity hull (1.2 m/s for each axis). Applying Prop. 1 under the objective of minimum change to control point positions (bottom left figure), the resulting velocity profile is shown at the bottom right, where the velocity profile is strictly bounded.

$\mathbf{S}_l\mathbf{P}^D \leq u_{l,D}^{max}\mathbf{1}_{(k+1)\times 1}$ *is a sufficient condition for the derivative along coordinate* D *to be thoroughly bounded; i.e.,* $u_{l,D}^{min} \leq \frac{d\mathbf{s}^D(u)}{d^l u} \leq u_{l,D}^{max}$, $\forall u \in [0,1]$, *where* $\mathbf{P}^D \in \mathbb{R}^{k+1}$ *is coordinate* $D \in \{x, y, z\}$ *of the control point span* $\mathbf{P}$, *and* $\mathbf{C}_l \in \mathbb{R}^{(k+1)\times(k+1)}$ *is a constant mapping matrix of the* l*-th derivative satisfying* $\frac{d\mathbf{b}}{d^l u} = \mathbf{C}_l\mathbf{b}$. [1]

Proof. Please refer to Appendix A.1 for the detailed proof. □

2.4 B-spline-Based Kinodynamic Search

2.4.1 Motivating Example

As shown in the motivating example in Fig. 2.1, hierarchical motion planners may produce sub-optimal or even dynamically infeasible trajectories given a non-static initial state of the quadrotor. The reason is that the geometric planner has no knowledge about the vehicle dynamics

[1] $\mathbf{S}_l$ and $\mathbf{C}_l$ are fixed given the degree k and the derivative order l.

and restricts the solution space of path parameterization to a *homotopy class* of the geometric shortest path. The inadequacy motivates us to propose an efficient kinodynamic planning algorithm which can work in real-time. However, kinodynamic planning is typically time consuming [22, 31]. The major computation of the traditional kinodynamic planning lies in three tasks, namely, covering the large state space, solving the BVP and collision checking.

Given the advantageous properties of the B-spline introduced in Sect. 2.3, we propose using uniform B-spline parameterization in kinodynamic planning, which facilitates reducing the computation time for the above three tasks. Specifically, we propose a spatial-grid-based deterministic graph search to place B-spline control points. The proposed search algorithm has three major features as follows:

- Controllable discretization of the state space: Due to the locality of the B-spline, it is possible to incrementally sample the B-spline control points during the search. A vertex tuple structure is proposed to recover the Markovian assumption and make the problem analyzable. A novel graph aggregation technique is proposed to control the discretization of the state space, which achieves a speed-quality tradeoff.
- Closed-form evaluations of control cost and dynamical feasibility: The control cost and feasibility of the B-spline can be evaluated in closed forms efficiently.
- Offline-computable inflation to avoid collision checking: The maximum deviation of uniform B-spline from the free-cells can be characterized offline and compensated for by workspace inflation.

The kinodynamic search accounts for the total control efforts and dynamical limits, which is a systematic way to deal with the non-static initial states in replanning.

2.4.2 Problem Formulation

In this section, we formally present the problem of the B-spline-based kinodynamic search on a spatial grid. The problem is formalized as a deterministic graph search where the action is the placement of the control points. Suppose the topological graph associated with the grid map is denoted as $\mathcal{G} \coloneqq (V, E)$, where V is the set of vertices denoting the collection of free cells and E denotes the set of edges $(i, j) \subset V \times V$ between all adjacent vertices i and j. The adjacency of the vertices depends on the grid connectivity adopted. In this chapter, every cell in the 3-D

grid has 26 neighbors which are cells connected to this cell at a Chebyshev distance of 1.[2]

Given uniform B-spline parameterization of degree k and knot separation Δ_t, the proposed search method finds a finite sequence $\pi = (v_0, v_1, \ldots, v_T)$ of vertices representing an *admissible* control point placement which satisfies $v_i \in V, (v_{i-1}, v_i) \in E$ for each $i = 1, \ldots, T$ and connects the given initial state $\pi_s = (v_s^0, v_s^1, \ldots, v_s^k)$ and goal state $\pi_g = (v_g^0, v_g^1, \ldots, v_g^k)$, i.e., $(v_s^k, v_0) \in E$ and $(v_T, v_g^0) \in E$. The sequence π possibly contains repetition since we allow placing the control points at the same cell. π_s and π_g are two tuples, both containing $k+1$ vertices which form the control point span according to the definition of the B-spline in Sect. 2.3. Therefore, π_s and π_g actually represent two short trajectories, different from the initial and goal positions used in geometric planners and the position-velocity-acceleration state vector used in these kinodynamic planners [15, 22, 31].

Since the B-spline is evaluated in terms of the control point span, we re-organize π by combining neighboring $k+1$ vertices as one vertex tuple. Combining the sequence π with π_s and π_g, the overall sequence can be formalized as $\tilde{\pi} = (v_s^0, \ldots, v_s^k, v_0, \ldots, v_T, v_g^0, \ldots, v_g^k)$. We define the *ordered* sub-sequence containing consecutive $k+1$ vertices of $\tilde{\pi}$ as a *k-degree vertex tuple*, which is denoted by $[\tilde{\pi}]^k$. We provide a toy example in Fig. 2.5 showing how 3-degree vertex tuples can be constructed.

We denote by $[\tilde{\pi}]_j^k$ the j-th k-degree tuple in $\tilde{\pi}$, and two neighboring tuples, $[\tilde{\pi}]_{j-1}^k$ and $[\tilde{\pi}]_j^k$, overlap for k vertices. According to this convention, $[\tilde{\pi}]_0^k = \pi_s$ and $[\tilde{\pi}]_J^k = \pi_g$, where $J+1 = k+T+3$. Each vertex tuple represents a short trajectory, and a sequence of neighboring vertex tuples represents a continuous trajectory. We associate with each k-degree tuple $[\tilde{\pi}]$ a *strictly positive* cost function $f_{k,\Delta_t} : [\tilde{\pi}]^k \to \mathbb{R}_+$. Note that the cost function has to be strictly positive to cope with the possible repetition in π. We state the problem as follows:

Problem 1. Given a uniform B-spline of degree k and knot separation Δ_t, initial state π_s and goal state π_g, find admissible control point placement $\pi = (v_0, v_1, \ldots, v_T)$ on $\mathcal{G}$ such that the following cost function is minimized:

$$\mathcal{J}_{k,\Delta_t}(\tilde{\pi}) = \sum_{j=0}^{J} f_{k,\Delta_t}([\tilde{\pi}]_j^k),$$

where $[\tilde{\pi}]_j^k$ is a k-degree vertex tuple, whose corresponding trajectory should satisfy collision-free and dynamical feasibility requirements.

[2]Note that the connectivity can also be defined based on a Chebyshev distance larger than one, which will result in a non-uniform control point placement and a higher computational complexity.

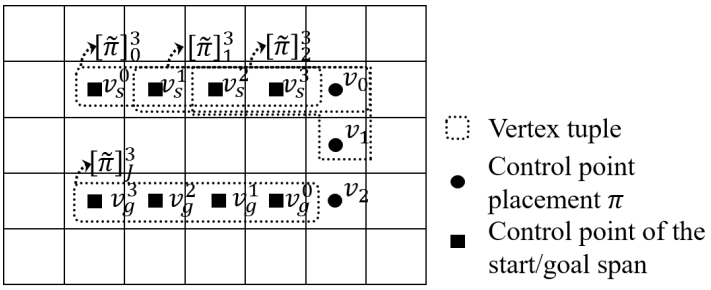


Figure 2.5: Illustration of the construction process of 3-degree vertex tuples from an admissible path. Each vertex tuple is formed by combining four consecutive control points. $[\tilde{\pi}]_0^3$ and $[\tilde{\pi}]_1^3$ are two neighboring vertex tuples since they overlap for three vertices.

We adopt a cost function following the idea of the linear quadratic minimum time control problem [22, 43], where the control cost is represented by Eq. 2.4 with a tradeoff penalty on the execution time Δ_t. Mathematically, the cost function f_{k,Δ_t} is represented as follows:

$$f_{k,\Delta_t}([\tilde{\pi}]_j^k) = \lambda\Delta_t + \int_0^1 \left(\frac{d\mathbf{s}_{[\tilde{\pi}]_j^k}(u)}{d^l u}\right)^2 du, \tag{2.5}$$

where $[\tilde{\pi}]_j^k$ can be rewritten into matrix form, as in Sect. 2.3; the integral can be evaluated according to Eq. 2.4; and λ is the weight for the trajectory execution time. The criterion to check the collision-free and dynamical feasibility of the vertex tuple is introduced in Sect. 2.4.4.

2.4.3 Optimal B-spline-Based Kinodynamic Search

The difficulty of solving Prob. 1 is that the placement of any control point depends on the placed k control points (not only the predecessor) within the vertex tuple. Regarding the placement of a single control point as an action, the Markovian assumption does not hold, which makes traditional graph search algorithms inapplicable. However, we find that Problem 1 can be transformed into an equivalent standard shortest path problem on a higher dimensional directed graph $\mathcal{G}_H = (V_H, E_H)$ induced by $\mathcal{G}$ by regarding the whole vertex tuple as a "vertex". Since the vertex tuple is the basic evaluation unit of the B-spline, the placement of the next vertex tuple will only depend on its predecessor, which follows the Markovian assumption. The transformation enables the usage of well-characterized shortest path search algorithms, such as Dijkstra's [44] and A* [45]. In the following, we elaborate the transformation.

Similar to the construction process of the vertex tuple in Sect. 2.4.2, we construct $\mathcal{G}_H$ as follows: 1) each vertex tuple $\hat{v}_i = (v_i, v_{i+1}, \ldots, v_{i+k}) \in V_H$ is created by combining a sequence of adjacent vertices of $\mathcal{G}$ satisfying $(v_{j-1}, v_j) \in E$ for $j = i+1, \ldots, i+k$; and 2) two vertices

Algorithm 1: Optimal B-spline-Based Kinodynamic Search

$\mathcal{O}, \mathcal{L} \leftarrow \texttt{Initialize}(\pi_s, \pi_g, k, \Delta_t)$;
while $\mathcal{O} \neq \varnothing$ **do**
  $(m, \hat{v}_i) \leftarrow \texttt{Pop}(\mathcal{O})$;
  if $m = \texttt{Index}(\pi_g)$ **then**
    return **success**;
  end
  for $\hat{v}_j \in \underline{\texttt{FeasibleSuccs}}(\hat{v}_i, \mathcal{G}, k, \Delta_t)$ **do**
    $n \leftarrow \underline{\texttt{Index}}(\hat{v}_j)$;
    if *not* $Visited(n, \mathcal{L})$ **then**
      $g(\hat{v}_j) \leftarrow \infty$;
    end
    if $g(\hat{v}_j) > g(\hat{v}_i) + \underline{f_{k,\Delta_t}(\hat{v}_j)}$ **then**
      $g(\hat{v}_j) \leftarrow g(\hat{v}_i) + f_{k,\Delta_t}(\hat{v}_j)$;
      $h(\hat{v}_j) \leftarrow \texttt{Heuristic}(\hat{v}_j, \pi_g)$;
      $\mathcal{O} \leftarrow \texttt{Insert}(\mathcal{O}, g(\hat{v}_j) + h(\hat{v}_j), \hat{v}_j)$;
    end
  end
end

$\hat{v}_i$ and $\hat{v}_j$ on V_H are adjacent if and only if the last k vertices of $\hat{v}_i$ overlap with the first k vertices of $\hat{v}_j$. The construction of $\mathcal{G}_H$ groups the associated control point coordinates into a high-dimensional state. Note that each dimension of the combined state has the same physical meaning, i.e., the spatial coordinates of the control point. This observation motivates us to come up with a low-dispersion search algorithm, as presented in Sect. 2.4.5.

In Fig. 2.6a, we show an example of how the trace of control points on $\mathcal{G}$ can be mapped to the path on $\mathcal{G}_H$. The initial vertex tuple $\hat{v}_s$ is shown by squares. Starting from $\hat{v}_s$, we consider two directions of the next-step placement, which form $\hat{v}_0$ and $\hat{v}_1$, respectively. In a similar way, starting from $\hat{v}_0$, two expansion directions are considered, forming $\hat{v}_3$ and $\hat{v}_4$, while for $\hat{v}_1$, one direction ($\hat{v}_5$) is considered. Note that although the last vertex of $\hat{v}_4$ and $\hat{v}_5$ are in the same spatial cell, in the induced high-dimensional graph $\mathcal{G}_H$, they are two distinct vertex tuples. Following the expansion of control points, we form a tree of vertex tuples on $\mathcal{G}_H$. From the expansion process, we observe that, given the initial and goal vertex tuple, the problem of finding the optimal control point placement is equivalent to finding the shortest path on the graph $\mathcal{G}_H$. We refer interested readers to Appendix A.2 for further details.

Recall that in Sect. 2.4.2, we allow the repetition of vertices since some necessary vertex tuples rely on repetition; for instance, the same vertex being repeated $k + 1$ times actually represents a static state (for Δ_t). According to the definition of $f_{k,\Delta_t}([\tilde{\pi}]^k)$, the cost of $\mathcal{G}_H$ is

defined on vertices V_H instead of the edges. Although repetition is allowed, each vertex $\hat{v} \in V_H$ of $\mathcal{G}_H$ is associated with a *strictly positive* cost so that repetition is properly penalized.

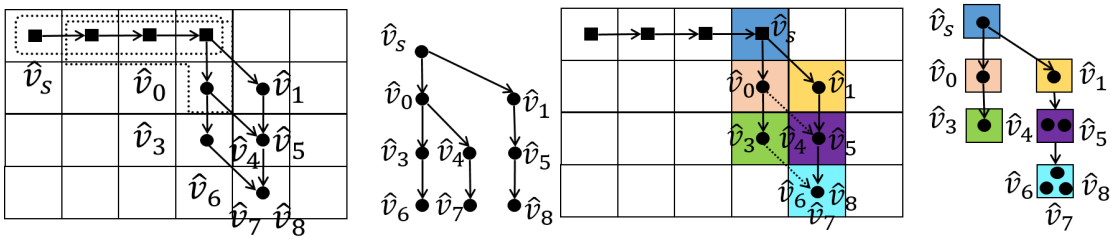
Note that any path on $\mathcal{G}_H$ is a time-parameterized B-spline trajectory instead of a geometric path. The dynamical feasibility and control cost are taken into account by evaluating the corresponding short trajectories of the nodes of $\mathcal{G}_H$. Problem 1 can be solved optimally using traditional label-correcting algorithms such as Dijkstra's [44] and A* [45] on the induced graph $\mathcal{G}_H$. The optimal control point placement π^* can then be re-constructed. The optimal B-spline-based kinodynamic (OBK) search algorithm is outlined in Algo. 1.

Typically, label-correcting algorithms maintain one or multiple sets of vertices as so-called *fringes* [46]. For example, A* [45] maintains two fringes (known as the *OPEN* set and the *CLOSED* set) to reduce the expansion of nodes and save computation. Similarly, Algo. 1 maintains two fringes, namely, the *OPEN* set (as denoted by $\mathcal{O}$) and the *VISITED* set (as denoted by $\mathcal{L}$). The visited set $\mathcal{L}$ provides query and retrieving functions for vertices. We use $\texttt{Index}(\hat{v})$ (Algo. 2) to assign a unique integer index to each distinct vertex tuple. Specifically, Algo. 2 collects the extracted coordinates $\mathcal{I}(\hat{v})$ (using the $\texttt{Coord}(v)$ function) for each vertex v in the tuple $\hat{v}$, and uses the $\texttt{UniqueEncode}(\cdot)$ function to generate a unique hash encoding for a series of integer coordinates $\mathcal{I}(\hat{v})$.

We construct the nodes of $\mathcal{G}_H$ only on demand during the search process as the graph size may be prohibitively large. At the beginning of the search, the whole $\mathcal{G}_H$ is not explicitly constructed, and the visited set $\mathcal{L}$ and the open set $\mathcal{O}$ are both initialized to empty. After the insertion of the initial state π_s, a tree of nodes is gradually expanded based on the $\texttt{FeasibleSuccs}(\cdot)$ function and the priority queue structure maintained by $\mathcal{O}$. Every time a new node is found (whether or not a successor node is a new node is identified by $\mathcal{L}$, which is implemented using a hash map structure), the node is added to $\mathcal{L}$ to trace its open/closed status. In practice, only a small proportion of the nodes of $\mathcal{G}_H$ are constructed before the search succeeds. A toy example[3] which compares the number of actual expanded nodes with the estimated graph size is shown in Tab. 2.1. Note that the EBK method used in Tab. 2.1 is an efficient version of Algo. 1, which is discussed in detail in Sect. 2.4.5. The estimated graph size is computed based on the graph aggregation technique introduced in Sect. 2.4.6.

Note that there are three functions making Algo. 1 different from the traditional geometric

[3]The setup of this example is the same as the qualitative experiment in Fig. 2.10, where the grid size is $51 \times 51 \times 5$ and the B-spline degree is five.



(a) Mapping from the traces on $\mathcal{G}$ to $\mathcal{G}_H$ (b) Graph aggregation

Figure 2.6: Illustration of the mapping process to $\mathcal{G}_H$ and the graph aggregation process used by the EBK search.

Algorithm 2: Given a k-degree vertex tuple $\hat{v} \in V_H$, assign a unique index to each distinct tuple. $\texttt{Index}(\hat{v})$

$\mathcal{I}(\hat{v}) = \varnothing$;
for *all* $v \in \hat{v}$ **do**
  $\mathcal{I}(\hat{v}) \leftarrow \mathcal{I}(\hat{v}) \cup \texttt{Coord}(v)$;
end
return $\texttt{UniqueEncode}(\mathcal{I}(\hat{v}))$;

path search: 1) the cost function $f_{k,\Delta_t}(\cdot)$ evaluates the control effort of the k-degree vertex tuple, instead of a simple path length measure; 2) the function $\texttt{Index}(\cdot)$ regards the k-degree vertex tuple as the "state", which expands the high-dimensional state space supported by the control point coordinates, while the geometric path search typically regards positions as states; and 3) the function $\texttt{FeasibleSuccs}(\cdot)$ will expand to the neighboring k-degree tuples with dynamical feasibility checking, while the geometric path search cannot check feasibility without parameterization. As for the heuristic function $\texttt{Heuristic}(\cdot)$, we adopt the admissible minimum time heuristic in [22].

It is worth noting that the design of Algo. 1 heavily relies on the properties of the B-spline. Thanks to the local control property, the cost evaluation and feasibility checking can be done locally based on the k-degree vertex tuple. Instead of solving the BVP, the proposed search method expands to new states on the high-dimensional graph $\mathcal{G}_H$ by expanding low-dimensional control point coordinates, which are associated with closed forms for cost evaluation. More-

Table 2.1: Illustration of the on-demand graph construction.

Method	**Run Time (s)**	**Actual # of Exp. Nodes**	**Total # of Nodes (Esti.)**	**Percentage (%)**
EBK-D1	0.034	2,334	13,005	17.9
EBK-D2	0.383	36,818	351,135	10.5
EBK-D3	7.422	460,469	9,480,645	4.9
EBK-D4	173.710	9,096,338	255,977,415	3.6

over, checking for collision is time consuming in traditional kinodynamic planners [15, 22, 31]. By using B-spline parameterization, the process can be avoided by characterizing the B-spline deviation, as introduced in Sect. 2.4.4. The limitation of our method is that the expansion of control points is restricted by the resolution and connection of the grid, which results in limited representations of B-spline trajectories.

2.4.4 Feasibility Condition

The dynamical feasibility of the k-degree vertex tuple can be validated via checking the extrema of the derivative of the B-spline. Since the derivative of the B-spline is another B-spline with decreasing degree, the velocity spline is of degree $k-1$ and the acceleration spline is of degree $k-2$. Considering that the convex hull property in Prop. 1 is a sufficient but not necessary condition, directly using Prop. 1 for feasibility checking may be conservative. Actually, there is a non-conservative approach for feasibility checking by using the closed-form solutions of the extremas of the uniform B-spline. Take the fifth-degree uniform B-spline as an example. The B-spline can be rewritten in a monomial basis according to Eq. 2.3. The velocity profile is a degree-4 polynomial whose extremas can be checked by finding the roots of its derivative (degree-3) in closed form.

For traditional kinodynamic planners [15, 22, 31], collision checking is an expensive process and may become the computation bottleneck of the algorithm [47]. Position-only shortest path search on the graph of the cell decompositions does not require collision checking since the piecewise linear connection between cell centers is restricted to collision-free cells. For the proposed method, given the B-spline parameterization of degree k and cell size of the decomposed environment, the B-spline may deviate from the piecewise linear connection due to the fact that it does not exactly pass through the control points. However, the maximum distance that the B-spline curve deviates from the piecewise linear collection can be characterized offline, which is compensable by moderate obstacle inflation. The inflation needed is characterized in Appendix A.3. In practice, since the degree of the B-spline is fixed and the cell size is not tuned frequently, the inflation can be calculated once and then used for many experiments.

2.4.5 Efficient Low Dispersion Search

Before proposing the efficient methods, we present a complexity analysis of the OBK search. According to the definition of the k-degree vertex tuple, the total number of vertices V_H of the graph $\mathcal{G}_H$ grows exponentially w.r.t. the degree k of B-spline parameterization, i.e., $|V_H| =$

Algorithm 3: Given a k-degree vertex tuple $\hat{v} \in V_H$, assign an integer index based on the selected coordinates of the tuple. $\texttt{Index}(\hat{v}, d)$

$\mathcal{I}(\hat{v}) = \varnothing$;
for *all* $i \in \{k-d+1, \ldots, k\}$ **do**
  $\mathcal{I}(\hat{v}) \leftarrow \mathcal{I}(\hat{v}) \cup \texttt{Coord}(\hat{v}[i])$;
end
return $\texttt{UniqueEncode}(\mathcal{I}(\hat{v}))$;

$O(|V|^{k+1})$. Note that according to Prop. 2, Algo. 1 shares a similar complexity with the execution of Dijkstra's algorithm on the graph $\mathcal{G}_H$ if the heuristic is set to zero. According to the known result that each vertex is expanded at most once for Dijkstra's algorithm (see [48, 49]), the maximum number of iterations of Algo. 1 is upper bounded by $|V_H| = O(|V|^{k+1})$, which characterizes the worst-case execution time of the OBK search. Actually, since $|V_H|$ and $|E_H|$ grow exponentially with k, the worst-case execution time of *any* algorithm that optimally solves Problem 1 scales exponentially with k.

Given the observation that the state in the OBK search is homogeneous (i.e., all the dimensions are control point coordinates), we can aggregate the nodes of $\mathcal{G}_H$ based on the proximity of the coordinates to gain efficiency. Specifically, according to Algo. 2, each vertex $\hat{v} \in V_H$ is marked with a unique integer index. In the modified $\textsc{Index}(\cdot)$ function in Algo. 3, we encode the vertex $\hat{v}$ only based on the coordinates for the last d vertices such that the vertices which share the same partial coordinates will be regarded as the same node. By aggregating the nodes of $\mathcal{G}_H$, the dimension of the search space is directly controlled by the user-specified parameter d. The modification essentially conducts a low dispersion search on the high-dimensional graph $\mathcal{G}_H$ with a control on the number of expanded nodes.

Different d values will determine how the vertex tuples are aggregated, which in turn affects the solution quality and algorithm efficiency. Note that although the search space is reduced, the continuity and smoothness of the resultant trajectory is maintained since the modification preserves the sharing of the B-spline coordinates. The resultant search method is called EBK search and a formal analysis of the EBK search is provided in Sect. 2.4.6.

2.4.6 Analysis of the EBK Search

As introduced in Sect. 2.4.5, the user-specified parameter d determines how vertex tuples in the graph $\mathcal{G}_H$ are aggregated. We provide a toy example of $d = 1$ in Fig. 2.6b to understand the graph aggregation. When choosing $d = 1$, as in Fig. 2.6b, vertex tuples are aggregated based on

the last vertex of the tuple. For instance, $\hat{v}_4$ and $\hat{v}_5$ share the same last vertex and are aggregated into the same node, as marked in *purple*. In the same way, the three distinct nodes $\hat{v}_6$, $\hat{v}_7$ and $\hat{v}_8$ are aggregated into the same *cyan* node.

The edges of $\mathcal{G}_H$ are also reduced accordingly. For example, the edges $(\hat{v}_4, \hat{v}_7)$ and $(\hat{v}_5, \hat{v}_8)$ are aggregated since they are connecting the same two aggregated nodes. Note that once a path to the aggregated node is determined, such as the path *blue-yellow-purple-cyan*, the vertex tuple associated with each aggregated node is determined. In this example, the *purple* node is associated with $\hat{v}_5$ and the *cyan* node is associated with $\hat{v}_8$. Therefore, given the initial vertex tuple, any path on the aggregated graph, can be uniquely transformed to a path on the graph $\mathcal{G}_H$. And, apparently, a path on the graph $\mathcal{G}_H$ can be transformed to a path of aggregated nodes. The equivalence states that we are actually conducting a graph search on the aggregated graph with a controllable number of vertices, i.e., a low-dispersion search on the original high dimensional graph $\mathcal{G}_H$. The resultant path on the aggregated graph can be reconstructed as an admissible path on $\mathcal{G}_H$.

For the simple case of $d = 1$, as illustrated in Fig. 2.6b, the size of the aggregated graph is the same as the original graph $\mathcal{G}$. Therefore, the EBK search can be as efficient as a shortest path search on the spatial grid by choosing a small d. And the advantage of the EBK search is that it directly outputs a time-parameterized dynamically feasible trajectory. It turns out that the EBK search is resolution complete with respect to the aggregated graph, and we refer interested readers to a detailed analysis of the EBK search in Appendix A.4.

By choosing a small $d < k+1$, a large number of vertex tuples are aggregated into one group, and the "resolution" of the graph becomes large. Due to the aggregation, the representation of the trajectory is limited. An intuitive example is that, when choosing $d = 1$, the search process will never choose to place the same control point in the same grid cell due to the strictly positive cost. As a result, the trajectory obtained may fail to reach the exact end state, such as a static state. However, the issue can be addressed by choosing a larger d and sacrificing efficiency.

Compared to the preliminary version, i.e., the RBK search in [23], the EBK search is more flexible and allows for control of the algorithm efficiency and solution quality. The connection is that the RBK search is essentially the EBK search using $d = 1$.

2.5 Elastic Optimization

To compensate for the discretization introduced by the EBK search and further improve the trajectory quality, we present the EO approach, which refines the control point placement to the optimal location w.r.t. the free space. Our approach is motivated by the seminal work [50], in which a collision-free "tube" around the initial path is identified and the path is "stretched" within the tube so that the shape is optimized. Mathematically, the tube is defined as a series of balls, with the ball centers denoted as $\mathcal{P} \coloneqq \{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_T\}$ and corresponding radiuses denoted as $\mathcal{R} \coloneqq \{r_0, r_1, \ldots, r_T\}$, where $\mathbf{p}_i$ denotes the ball center and r_i denotes the radius. The tube is defined to be "well-connected" if and only if $\|\mathbf{p}_i - \mathbf{p}_{i+i}\|_2 \leq r_i + r_{i+1}, \forall i \in \{0, \ldots, T-1\}$. Compared to [50] which cannot handle dynamical feasibility constraints for complex kinodynamic systems such as quadrotors, we propose a convex optimization formulation based on B-spline parameterization, which uses the convex hull property to enforce feasibility.

Note that Zhu *et al.* [51] also propose a convex elastic smoothing formulation for car-like robots. There are two major differences: 1) The formulation in [51] is based on the dynamics of car-like robots and cannot be applied to complex dynamic systems, while our formulation uses high-order B-spline parameterization, which can be directly used to control quadrotors. 2) In [51], the smoothed trajectory may collide with obstacles due to the geometric incompleteness of the tube constraint (as shown in Fig. 2.8a) and only a heuristic waypoint insertion/obstacle inflation scheme is provided, while the EO approach has a theoretical safety guarantee, which is achieved by a two-level inflation scheme to ensure the connectivity of the tube and a finite iterative control point insertion process.

2.5.1 Elastic Tube Expansion

In [50] and [51], the elastic tube is a series of connected balls which are centered at the waypoints of the reference path. Intuitively, the tube generated in this way cannot fully utilize the free space around it, as shown in Fig. 2.7a. We therefore propose a lightweight tube expansion algorithm so that the tube can roughly represent the locally largest free space. Given the initial control point placement $\pi = (v_0, \ldots, v_T)$ provided by Algo. 1, we first extract the coordinates of π, and denote the collection of coordinates as $\mathcal{P} \coloneqq \{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_T\}$ following the notation used in Sect. 2.3.

The elastic tube expansion algorithm (Algo. 4) can be divided into two steps: First, we

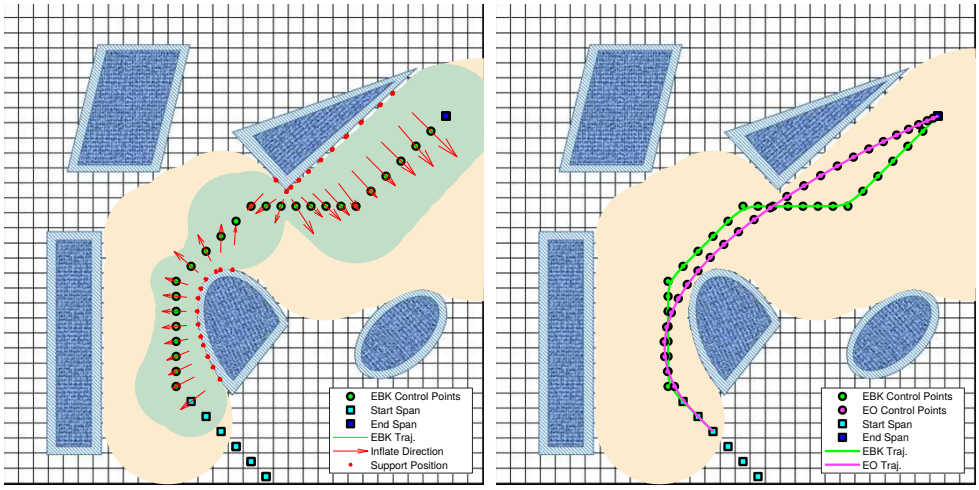
Algorithm 4: Expand elastic tube in configuration space $\mathcal{C}^{\text{ELAS}}$. $\texttt{TubeExpansion}(\mathcal{P}, \mathcal{C}^{\text{ELAS}})$

$\mathcal{I}(\hat{v}) = \varnothing$;
Initializes: $d_{\text{infl}}^{\min}, d_{\text{infl}}^{\max}, d_{\text{thres}}$. $\mathcal{R} = \mathcal{R}' = \mathcal{Q} = \varnothing$;
for *all* $\mathbf{p}_i \in \mathcal{P}$ **do**
  $(\mathbf{n}_i, r_i) \leftarrow \text{NNSEARCH}(\mathbf{p}_i, \mathcal{C}^{\text{ELAS}})$;
  $\overrightarrow{n} = (\mathbf{p}_i - \mathbf{n}_i)/\text{NORM}(\mathbf{p}_i - \mathbf{n}_i)$;
  while $d_{\textit{infl}}^{\max} - d_{\textit{infl}}^{\min} > d_{\textit{infl}}^{\textit{tol}}$ **do**
    $d \leftarrow \left(d_{\text{infl}}^{\max} + d_{\text{infl}}^{\min}\right)/2$, $\mathbf{p}_{i,\text{infl}} \leftarrow \mathbf{p}_i + d \cdot \overrightarrow{n}$;
    $(\mathbf{n}_i', r_i') \leftarrow \texttt{NNSearch}(\mathbf{p}_{i,\text{infl}}, \mathcal{C}^{\text{ELAS}})$;
    if $\texttt{Abs}(r_i' - d - r_i) > d_{\textit{thres}}$ **then**
      $d_{\text{infl}}^{\max} \leftarrow d$;
    else
      $d_{\text{infl}}^{\min} \leftarrow d$;
    end
  end
  $\mathcal{Q} \leftarrow \mathcal{Q} \cup \mathbf{p}_{i,\text{infl}}$, $\mathcal{R}' \leftarrow \mathcal{R}' \cup r_i'$;
end
return $\mathcal{Q}, \mathcal{R}'$;

construct the initial tube, by conducting a radius search for the initial placement $\mathcal{P}$, and obtain the nearest obstacle position $\mathbf{n}_i$. Second, we push the center of the bubbles in the direction $\overrightarrow{n}$ (away from the nearest obstacle) while satisfying the criterion that the new bubble contains the original bubble, as required by condition $\text{ABS}(r_i' - d - r_i) \leq d_{\text{thres}}$, as shown in Fig. 2.7a. The inflation process is implemented in a binary search manner. Algo. 4 will finally find a series of local maximum volume bubble centers $\mathcal{Q} \coloneqq \{\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_T\}$ based on the initial tube $\mathcal{P}$. For the parameter settings, $d_{\text{infl}}^{\max}$ and $d_{\text{infl}}^{\min}$ are the maximum and minimum inflation distance, respectively; d_{thres} is the threshold for checking whether the new bubble contains the original one, and should be set to a small value, e.g., less than the map resolution; and $d_{\text{infl}}^{\text{tol}}$ is the binary search end condition, which can be set to the resolution of the map. The function NNSEARCH is the nearest neighborhood search, which can be done efficiently if a KD-tree is maintained. The efficiency of Algo. 4 is verified in Section. 2.6.2.

2.5.2 Elastic Optimization Formulation

Given the inflated ball centers $\mathcal{Q} = \{\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_T\}$ and corresponding radiuses $\mathcal{R}' = \{r_0', r_1', \ldots r_T'\}$, the EO formulation minimizes the total control effort by finding the optimal placement $\mathcal{P}^* = \{\mathbf{p}_0^*, \mathbf{p}_1^*, \ldots, \mathbf{p}_T^*\}$ while satisfying the safety and dynamical feasibility constraints. The safety constraints are enforced by constraining the control point position inside the 3-D balls, and in



(a) (b)

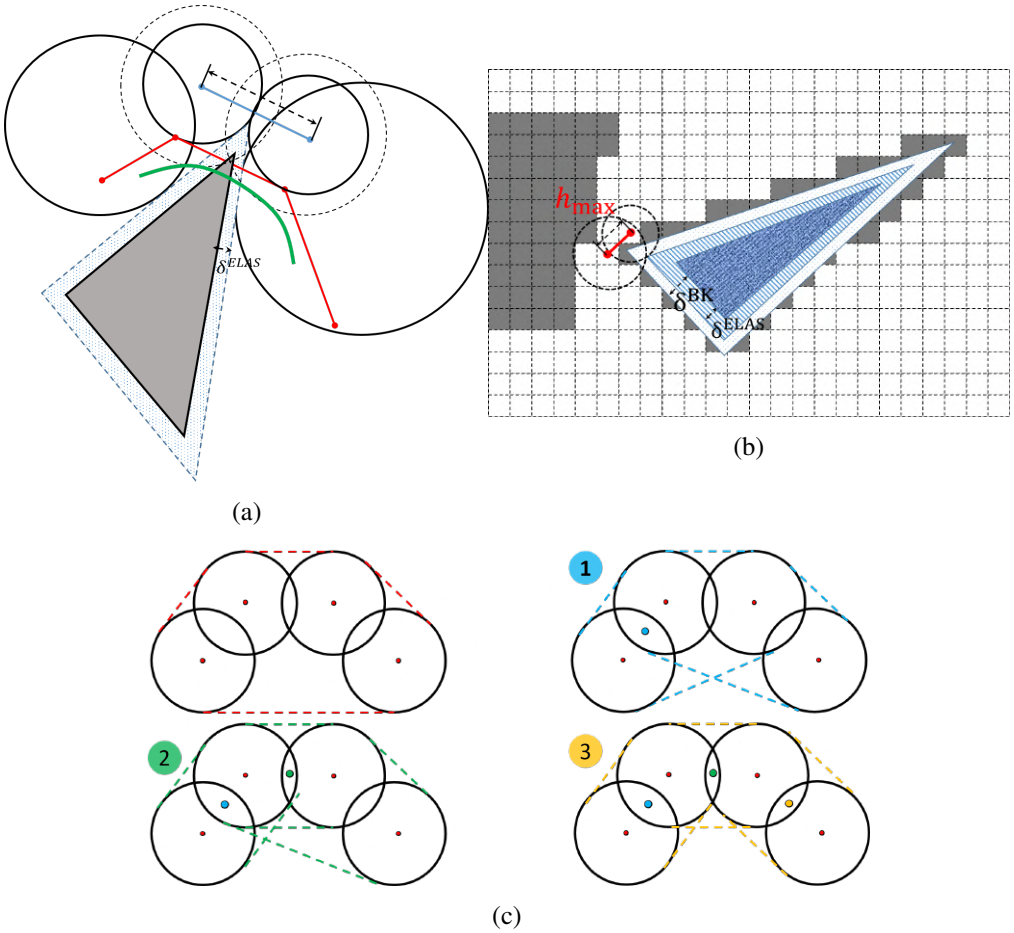
Figure 2.7: Illustration of the EO approach: (a) shows the elastic tube expansion process (Sect. 2.5.1); (b) shows the optimization process (Sect. 2.5.2). In (a), the original tube is marked in *green*, while the inflated tube is marked in *yellow*.

Sect. 2.5.3, we will discuss how to theoretically guarantee the safety of the resultant trajectory. The dynamical feasibility constraints are enforced using Prop. 1.

Note that like the EBK search, we consider the initial and goal state of the quadrotor. Denote the coordinates of π_s and π_g as $\mathcal{P}^s = \{\mathbf{p}_s^0, \mathbf{p}_s^1, \ldots, \mathbf{p}_s^k\}$ and $\mathcal{P}^g = \{\mathbf{p}_g^0, \mathbf{p}_g^1, \ldots, \mathbf{p}_g^k\}$, respectively. These coordinates are fixed during optimization. Similar to the construction of the k-degree vertex tuple in Sect. 2.4.2, we concatenate $\mathcal{P}^s$, $\mathcal{P}$ and $\mathcal{P}^g$ and formalize the concatenation in terms of the control point spans. Following the similar notation in Sect. 2.4.2, we denote by $[\tilde{\mathcal{P}}]_j^k$ the j-th control point span of the concatenated coordinates $\tilde{\mathcal{P}}$. It follows that $[\tilde{\mathcal{P}}]_0^k = \mathcal{P}^s$ and $[\tilde{\mathcal{P}}]_J^k = \mathcal{P}^g$, where $J = k + T + 2$. We denote by $\mathbf{P}_j$ the stacked coordinates matrix of $[\tilde{\mathcal{P}}]_j^k$, with $\mathbf{P}_j^D$ denoting the $D \in \{x, y, z\}$ axis. The optimization problem can be expressed as follows:

$$
\begin{aligned}
\min \quad & \sum_{j=0}^{J} f_{k,\Delta_t}([\tilde{\mathcal{P}}]_j^k) \\
\text{s.t} \quad & [\tilde{\mathcal{P}}]_0^k = \mathcal{P}^s, [\tilde{\mathcal{P}}]_J^k = \mathcal{P}^g \\
& \|\mathbf{p}_i - \mathbf{q}_i\|_2 \leq r_i', \qquad \forall i \in \{0, \ldots, T\} \\
& |\mathbf{SP}_j^D| \leq u_{l,D}^{\max} \mathbf{1}_{(k+1)\times 1}, \quad \forall j, D \in \{x, y, z\},
\end{aligned}
\tag{2.6}
$$

where the first constraint expresses that the initial and goal states need to be fixed. And the second constraint (quadratic) restricts the control points inside the expanded tube. The third constraint (linear) is to ensure the dynamical feasibility using the sufficient condition in Prop. 1.



(a) (b) (c)

Figure 2.8: Illustration of (a) the geometric incompleteness of the ball constraint, (b) the two-level inflation scheme, and (c) the iterative convex hull shrinking process for enforcing the safety guarantee. It can be observed in (a) that even the straight-line segments between the balls may collide with the obstacle.

Since the control points can be adjusted in continuous free space, the potential conservativeness brought by Prop. 1 is minor in practice. The overall formulation is a QCQP which can be solved efficiently using off-the-shelf convex solvers. The time cost is fixed given the initial placement $\mathcal{P}$, so it is not included in the objective.

2.5.3 Enforcing Safety Guarantee

Restricting control points inside the balls is not a sufficient condition for safety, for the following two reasons: 1) the balls are placed with a finite density and constraining the control points in the balls cannot preserve the collision-free characteristic even for straight-line segments between control points (Fig. 2.8a), and 2) the B-spline does not exactly pass the control points and deviates from the straight-line segments. The first issue is also observed in [51], which proposes using waypoint insertion/obstacle inflation to handle the problem. However, there is

no quantification of how much inflation or how many insertions are needed and only a heuristic is given. The second issue is inherently similar to one common issue faced by piecewise polynomial parameterization [6–9]: the polynomial may deviate from the collision-free straight-line segments between waypoints or exceed the safe flight corridor. Chen *et al.* [9] propose a finite iterative process by adding constraints on polynomial extremas based on a cube corridor (linear constraints), but this is not directly applicable to B-spline parameterization with quadratic constraints.

In our case, the issues can be resolved using a *two-level inflation scheme* and an iterative *convex hull shrinking process*. The two-level inflation scheme is to ensure the connectivity of the elastic tube, and furthermore, the inflation needed is quantified in the scheme. For simplicity, we first consider the original tube before applying the tube expansion algorithm (Algo. 4). The two-level obstacle inflation scheme is as follows: the configuration space in which we conduct kinodynamic search with larger obstacle inflation δ^{BK} is $\mathcal{C}^{\text{BK}}$, while we generate the elastic tube and optimize the control points in the configuration space $\mathcal{C}^{\text{ELAS}}$ with smaller obstacle inflation δ^{ELAS}, as shown in Fig. 2.8b. The difference adds additional clearance to any point in the configuration space $\mathcal{C}^{\text{BK}}$. Without the difference, the minimum radius of the ball at the grid center is $\min(c_x, c_y, c_z)$, where c_x, c_y and c_z are the grid size. Denote the maximum separation distance of two neighboring control points in the 3-D grid as $h_{\max}$. It follows that, if the difference $\delta^{\text{BK}} - \delta^{\text{ELAS}} > h_{\max}/2 - \min(c_x, c_y, c_z)$ holds, the additional clearance will ensure that the two neighboring balls overlap, thus ensuring the connectivity of the elastic tube. Note that Algo. 4 maintains the connectivity of the tube by ensuring that the inflated ball contains the original ball while keeping the same support point on the obstacle. For the low-level inflation, given $h_{\max}$, $\delta^{\text{ELAS}} \geq \frac{\sqrt{2}-1}{2} h_{\max}$ is sufficient for the straight-line segments to be contained in the free space [23].

Based on the two-level inflation scheme, we propose an iterative *convex hull shrinking* process, which pulls the B-spline trajectory back to the free space in the case of collision. The idea of the process is to iteratively add control points to the original B-spline control point sequence. The newly added control points are constrained in the intersection of two consecutive balls. [4] The process is built upon the convex hull property of the B-spline and constrains the B-spline trajectory by shrinking the convex hull. We highlight that only a *finite* number of control points are needed to enforce the safety of the B-spline trajectory. This could save a significant amount

[4] According to the introduction in Sect. 2.3, the total number of knots should satisfy $m + 1 = (n + 1) + k + 1$. Since uniform B-spline is used, when a new control point is inserted, the number of knots is increased by one, while the knot separation Δ_t remains the same. The new control point sequence still matches the new knot vector.

of computation power compared to the methods which apply dense constraint points based on a conservative heuristic [5, 51]. We conclude this feature in Theorem 1.

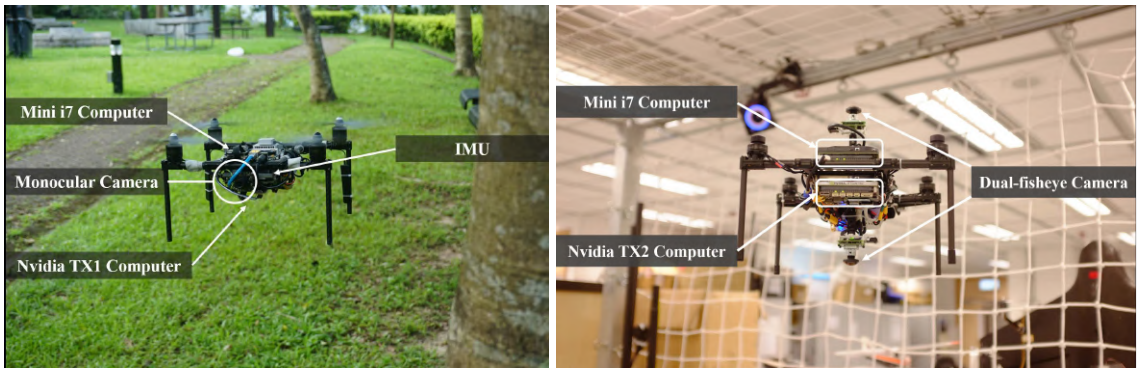
Theorem 1. *Given a well-connected elastic tube, a B-spline trajectory that fits within the tube can be generated by iteratively adding constrained control points to the original B-spline control point sequence. The newly added control point is constrained inside the intersection of neighboring balls. The iterative process succeeds in a finite number of iterations, or infeasibility is reported when the dynamical feasibility cannot be satisfied, given the current tube and control point sequence.*

Proof. Please refer to Appendix A.5 for the detailed proof. □

In Fig. 2.8c, we provide a toy example of the convex hull shrinking process. The initial placement is constrained in four balls, and the convex hull envelope exceeds the free space. If collision is detected, we add one additional control point (*blue dot*) which is constrained to be inside the intersection of the first and second ball. The extended EO is expected to be executed again, and the convex hull shrinks. Similarly, if the collision is still not resolved, we iteratively add control points to the intersection space. It can be shown that at most nine iterations are needed to resolve the collision in this case.

2.6 Validation

2.6.1 Implementation Details



(a) Monocular Vision-Based Quadrotor Testbed (b) Dual-fisheye Vision-Based Quadrotor Testbed

Figure 2.9: Illustration of our (a) monocular vision-based quadrotor testbed and (b) dual-fisheye vision-based quadrotor testbed.

Receding Horizon Replanner Using Local Control

As shown in Fig. 2.3, B-spline has the local control property which facilitates the receding horizon (re)planning. Specifically, all the control points are organized in a sliding window. The control points corresponding to the executed and executing trajectory are committed and fixed. The disturbance caused by the optimization will not affect the feasibility of the executing trajectory due to the local control. A stopping policy will be activated if no feasible solution is found before the end of the executing trajectory.

When replanning is activated, the EBK search is called to update the placement for the control points under optimization. Note that the initial and goal state of the EBK search can be determined by the control points inside the window according to the local planning range. Note that the control points from the sliding window are in the continuous space after reshaping. However, the EBK search should use the discretized control points as the reference initial/ goal state to preserve optimality. The strategy of getting the reference states for the EBK search are to find the closest span pattern in terms of position and velocity error while matching the last control point to the grid cell. We constantly gather a fixed number of control points (e.g., twelve) for EO as the window moves forward.

There are two modes for the activation of replanning, namely, active mode and passive mode. For the passive mode, the EBK search is only called when collision is detected, while for the active mode, the EBK search is constantly activated as the sliding window moves forwards. Since the active mode can constantly improve the trajectory quality, it is more robust when the mapping quality is limited, but it is more computationally expensive.

Monocular Vision-Based Testbed

The monocular quadrotor testbed is equipped with a monocular camera (30 Hz), one IMU (100 Hz), an Intel i7 processor and an NVIDIA Jetson TX1 (Fig. 2.9a). The localization, mapping and planning modules are all running onboard. The localization module is based on our Monocular Visual Inertial Navigation System (VINS-Mono) [52], and the mapping module is based on our monocular dense mapping method [53] and truncated signed distance field (TSDF) fusion. No prior knowledge of the environment is required.

Dual-Fisheye Vision-Based Testbed

The dual-fisheye quadrotor testbed is equipped with two fisheye cameras (30 Hz), one IMU (100 Hz), an Intel i7 processor and an NVIDIA Jetson TX2 (Fig. 2.9b). All modules are running onboard. It is worth noting that by using the two fisheye cameras, the system can provide omnidirectional perception and the quadrotor is able to fly a round-trip with a fixed yaw angle. The mapping module is based on our dual-fisheye omnidirectional stereo system [54]. Also no prior knowledge of the environment is required.

2.6.2 Analysis

In this section, we present an analysis of the proposed kinodynamic planning framework. We begin with two individual analyses for the EBK search and the EO approach, respectively. To analyze the EBK search, we compare the proposed method with two kinodynamic planning algorithms, namely, search-based motion primitive (SMP) [22] and kinodynamic RRT* (kRRT*) [15], representing both the search-based method and sampling-based method, respectively. For the EO, we compare the EO approach with two state-of-the-art trajectory optimization techniques, namely, continuous trajectory (CT) optimization [33] and gradient-based safe (GS) trajectory optimization [41], which are popular non-linear optimization techniques for trajectory refinement. After the individual tests, we analyze the run-time efficiency of the proposed replanning framework, and compare the whole replanning system with the SMP [22] method. We run all the simulations on a desktop computer equipped with an Intel I7-8700K CPU.

Analysis of the B-spline-Based Kinodynamic Search

Recall that the motivation for introducing kinodynamic search to replanning is to facilitate dealing with the non-static initial states of the quadrotors. To this end, we analyze the performance of the kinodynamic search by planning from a given non-static initial state to varying goal states in a $10 \times 10 \times 2$ m test field. We compare our results with SMP [22] and kinodynamic RRT* [15] in terms of the trajectory quality and time efficiency, under the same planning setup. At the same time, we also illustrate the results of a hierarchical geometric planner as a common baseline. Specifically, the geometric planner first uses A* under the Euclidean distance measure to find the shortest path, and then parameterizes the path using the unconstrained QP formulation introduced in [7].

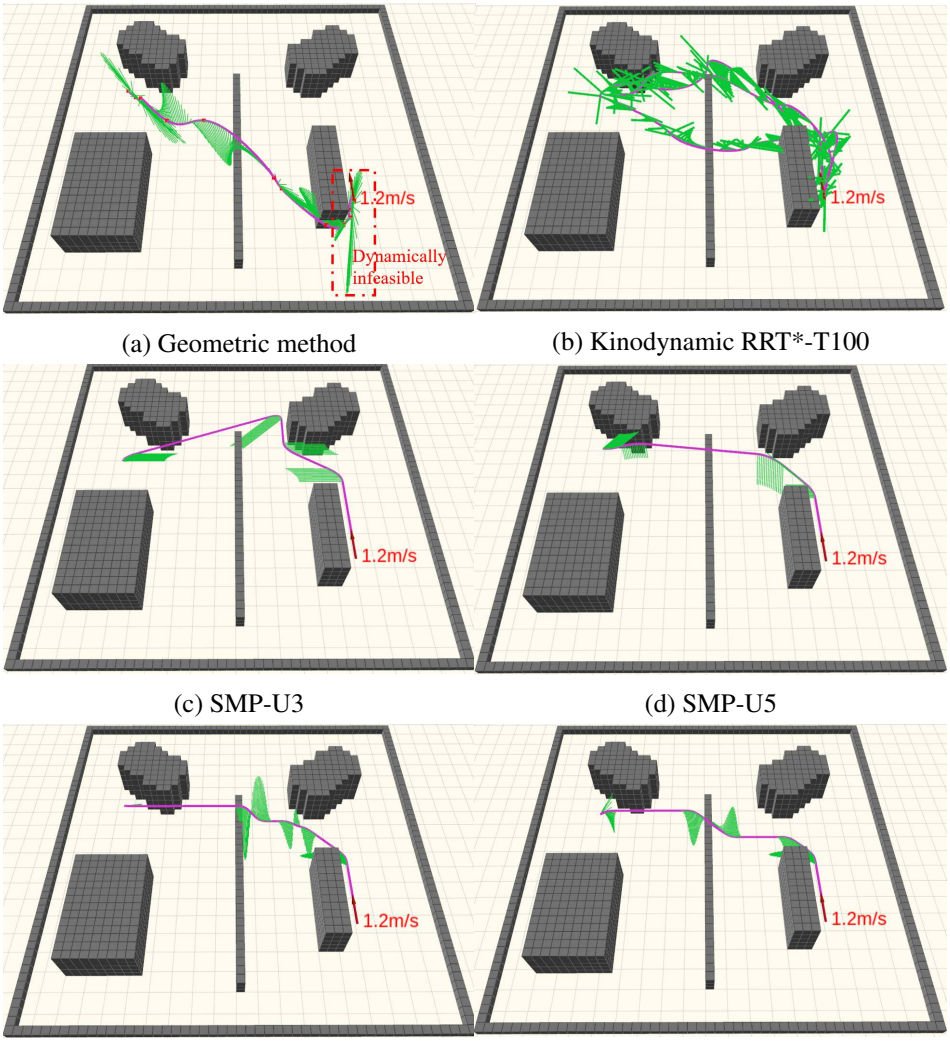
The kinodynamic planners from [15], [22] and our method all have a tuning parameter to

control the algorithm complexity and solution quality. For instance, SMP [22] can control the discretization resolution for the control input. To further investigate the optimality-efficiency tradeoff achieved by each algorithm, we also demonstrate the results for these algorithms under different parameter setups. Note that we focus on the real-time replanning scenario, so we are concerned with the operation region where the run-time efficiency is close to real-time. Specifically, we compare the following methods:

- *Geometric*†: A path finder using A* and a fifth-degree polynomial parameterization using the unconstrained QP formulation [7] by minimizing the average integral of acceleration.
- *kRRT***-T100*†: The kinodynamic RRT* planner [15] using a 3-D acceleration-controlled double integrator system under a 100 ms termination condition for the sampling.
- *kRRT***-T600*: The kinodynamic RRT* planner under a 600 ms termination condition.
- *SMP-U3*†: The SMP planner [22] using a 3-D acceleration-controlled double integrator system with three discrete control inputs for each axis.
- *SMP-U5*: The SMP planner with five discrete control inputs for each axis.
- *EBK-D1*†: Our EBK planner using fifth-degree B-spline parameterization and $d = 1$.
- *EBK-D2*: The EBK planner using $d = 2$.

The reason for using the acceleration-controlled double integrator system for kinodynamic RRT* and SMP is that if a higher-order system were used, the run-time would be too large, making it inapplicable to replanning. For a similar reason, a termination time larger than 600 ms for kRRT* and number of control inputs larger than five for SMP are not considered. The methods marked with † are amenable to real-time, and the other methods serve as showcases illustrating how the trajectory quality can be improved given a larger computation time budget. All the kinodynamic planners have a similar form of the cost function according to Eq. 2.4. Since both kRRT* and SMP adopt the acceleration-controlled double integrator system, the order of the derivative l we can take is 2 for all the experiments. The weight λ of the trajectory time is set to 20. We consider two additional metrics, namely, average acceleration and maximum acceleration, which represent the smoothness and feasibility of the resultant trajectory.

For the methods where cell decomposition is needed, such as A* for the geometric method and EBK, the environment is decomposed into cells with a fixed size of 0.2 m for each dimension. The geometric planner requires a *time allocation* module since the path does not contain



(a) Geometric method (b) Kinodynamic RRT*-T100

(c) SMP-U3 (d) SMP-U5

(e) EBK-D1 (f) EBK-D2

Figure 2.10: Comparisons of different kinodynamic planning approaches.

any time information. Generating the optimal time profile for a path is actually not a trivial problem [55], and the common practice is based on a heuristic allocation method, such as a trapezoid velocity profile [8, 33]. We follow this practice and set the average speed to achieve a similar trajectory duration to the kinodynamic planners. Time allocation is not needed for kinodynamic planners since they directly produce time-profiled trajectories. The initial state is set to a fixed position with non-zero velocity 1.2 m/s and zero acceleration, as shown in Fig. 2.10. The goal state is static, and its position is regularly sampled with a distance separation of 0.7 m. In total, there are 136 collision-free goals available. The velocity and acceleration limit are set to 2 m/s and 4.7 m/s^2 for each axis. The Δ_t for the EBK method is set to 0.17 s. The statistics averaged over the 136 rounds of planning are shown in Tab. 2.2, and the qualitative results for the planning to the same goal state are shown in Fig. 2.10.

Table 2.2: Comparison of different kinodynamic planning approaches.

Method	**Run Time (s)**	**Traj. Dur. (s)**	**Acc. Cost** (m^2/s^3)	**Ave. Acc.** (m/s^2)	**Max. Acc.** (m/s^2)
kRRT*-T600	0.661	4.04	30.6	2.42	4.18
SMP-U5	2.878	**3.86**	**12.8**	1.36	2.97
EBK-D2	**0.383**	4.52	14.4	0.95	4.60
Geometric†	0.018	4.34	108.8	4.60	9.48
kRRT*-T100†	0.110	4.60	38.8	2.54	4.28
SMP-U3†	0.111	4.47	32.3	1.70	4.70
EBK-D1†	**0.034**	**4.31**	**15.2**	1.10	4.56

Table 2.3: Comparison of different trajectory optimization approaches on random map.

Method	**Dens.**	**Success Frac.** (%)	**Run-Time (s)**	**Traj. Dur. (s)**	**Jerk Cost** (m^2/s^5)	**Ave. Vel.** (m/s)	**Ave. Acc.** (m/s^2)	**Max. Acc.** (m/s^2)	**Traj. Len.** (m)
	0.1	95	0.030	8.1	70.2	1.73	0.93	2.63	14.3
CT [33]	0.2	68	0.082	8.0	91.6	1.72	1.01	2.90	13.9
	0.4	46	0.073	7.9	105.6	1.69	1.05	3.10	13.7
GS [41]	0.1	**100**	0.062	11.1	186.8	1.39	0.87	2.99	15.6
Ours	0.1	**100**	**0.012**	**10.6**	**174.0**	1.43	0.38	1.35	14.1
GS [41]	0.2	**100**	0.075	12.8	205.9	1.28	0.85	2.89	15.9
Ours	0.2	**100**	**0.012**	**11.3**	**181.2**	1.38	0.49	1.85	14.5
GS [41]	0.4	**100**	0.206	12.9	190.8	1.33	0.89	3.10	17.4
Ours	0.4	**100**	**0.013**	**12.4**	**132.4**	1.36	0.55	1.34	16.2

According to the qualitative results in Fig. 2.10, there is a significant difference in the performance. For the geometric method (Fig. 2.10a), since the shortest path diverges from the initial velocity direction, the parameterization is jerky at the beginning. Moreover, as the unconstrained QP cannot enforce the dynamical feasibility, the generated trajectory is infeasible due to the non-static initial state. As for the kRRT* with a 100 ms time budget (Fig. 2.10b), it can respect the initial state of the quadrotor since the control effort is directly considered in the sampling process. It also guarantees the dynamical feasibility of the resultant trajectory by constraining the control input and states along the tree edges. However, the trajectory quality is unsatisfactory due to the limited sampling. We also observe some unpredictable randomized behavior as shown by the three rounds of planning with exactly the same initial state and goal state in Fig. 2.10b. Meanwhile, for the SMP method, the shape of the trajectory of SMP-U3 is not natural since the resolution of the control input is large.[5] Some maneuvers such as flying over the little step in the middle are not included in the solution space of SMP-U3. SMP-U5 performs better than SMP-U3 due to finer discretization. Finally, EBK-D1 and EBK-D2 both

[5]The acceleration bound we use is larger than that in [22].

generate an initial-state-aware smooth trajectory. EBK-D2 finds a slightly better trajectory than EBK-D1 according to the acceleration profile.

It is notable that the trajectory provided by the EBK method has continuity up to snap, which is good for controlling quadrotors. We can observe from Fig. 2.10 that the kRRT* trajectory only has continuity up to acceleration and that of the SMP has continuity up to velocity, due to the restriction of computation complexity and order of the system model.

From the quantitative results in Tab. 2.2, among the real-time methods (with †), EBK-D1 finds the lowest cost trajectory, achieving a $15.2\ m^2/s^3$ total cost within 0.034 s. Our method shows superior performance given the real-time requirement. It is of interest to examine the situation where we have a time budget in the range of seconds. In that case, SMP-U5 achieves a lower cost than EBK-D1 and EBK-D2. The reason is that the trajectory duration of EBK-D2 cannot be efficiently reduced since the control points have to be expanded step by step on the discrete grid. This illustrates the limitation induced by the discretization for the EBK method. However, the difference between the SMP method and EBK method is minor, meaning the EBK method can provide competitive solutions given a run-time budget of seconds.

Analysis of the Elastic Optimization

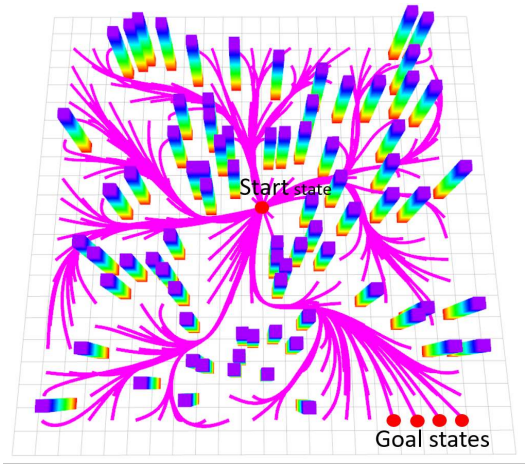
To evaluate the performance of EO, we compare our method with two state-of-the-art trajectory optimization methods: the CT method [33] and GS method [41]. Two test environments are provided, namely, a random map (shown in Fig. 2.11c) and a Perlin noise map[6] (shown in Fig. 2.11d). The map size is fixed to $20\,m \times 20\,m \times 4\,m$ for both maps. The start location is fixed to the center of the test field for the qualitative experiments in Fig. 2.11a and Fig. 2.11b, and is fixed to the left bottom corner for all the experiments in Tab. 2.3 to test the performance for long trajectories. The goal location is regularly sampled in the test field with a distance separation of 1 m. We vary the obstacle density of the random map from 0.1 pillars/m^2 to 0.4 pillars/m^2, where the pillar size is set to $0.5\,m \times 0.5\,m$. The qualitative results are provided in Fig. 2.11 and the quantitative statistics of the optimization performance are organized in Tab. 2.3. Note that we omit the statistics for the Perlin noise map in Tab. 2.3 since the trend is similar to that of the random map. All the optimization methods can handle high-order parameterization and they are set to minimizing the integral of the squared jerk. According to Eq. 2.4, the jerk cost listed in Tab. 2.3 has unit m^2/s^5 and represents the accumulated integral of the squared change rate of the acceleration, which represents the total control efforts.

[6]https://github.com/HKUST-Aerial-Robotics/mockamap

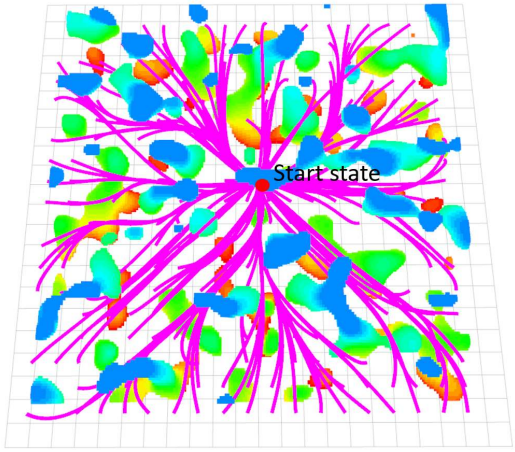
The CT method uses a gradient-based non-linear optimization process to optimize the polynomial coefficients such that the resultant trajectory is collision free. It requires a Euclidean signed distance field (ESDF) to evaluate the collision cost. Following the practice in [33], we provide an initial polynomial trajectory as an initial guess, which is parameterized by a straight-line guiding path. The number of segments is determined by a 3 m distance separation. The GS method shares a similar formulation to the CT method, with the difference being that the GS method starts from a collision-free initial guess, which is provided by A* search in the experiment. Both the CT method and GS method require *time allocation*, and in the experiment, we use the trapezoid velocity profile and scale the total allocated time such that different optimization methods achieve a similar average velocity, as shown in Tab. 2.3. The CT method has a larger average velocity due to the cases where it fails to resolve collision and the trajectory does not contain necessary deceleration. The success fraction is calculated by counting the collision-free and dynamically feasible trajectories among the total number of rounds.

Firstly, we examine the optimization reliability, i.e., the success fraction for the different methods. As shown in Tab. 2.3, the CT method is sensitive to the obstacle density. For a density of 0.1 pillars/m^2, the success fraction of the CT method is 95%, which means that in obstacle-sparse environments, the CT method can resolve collision efficiently. However, when the density increases to 0.4 pillars/m^2, we observe that the success fraction drops significantly to 46%. The reason is that the CT method easily gets stuck in the infeasible local minimum when the trajectory is inside the cluttered obstacle. As shown in Fig. 2.11c, the trajectory of the CT method gets stuck between two pillars where there is not enough clearance considering the quadrotor size. On the other hand, we do not observe a drop in the success fraction for either the GS method or EO method. The reason is that the GS method starts from a collision-free initial guess and has a dominant collision penalty compared to its smoothness cost. The EO method has a high success fraction according to its theoretical guarantee.

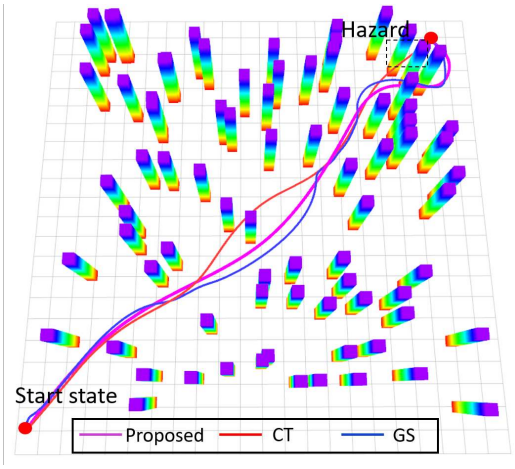
Secondly, we focus on the two reliable methods, namely, the GS method and EO method, and further investigate the trajectory statistics. As shown in Tab. 2.3, the average trajectory durations of the two methods are close, so the comparison of the jerk cost is fair. For an obstacle density of 0.2 pillars/m^2, averaged over the 135 rounds, the jerk cost of the EO method is 181.2 m^2/s^5, which is smaller than that of the GS method. Similar results are also observed for different densities. The reason is that the GS method is sensitive to the time allocation, since it also starts with an unparamterized path. However, for unknown environments, the shape of the initial path may vary and there is no systematic way to allocate the time. The heuristic trapezoid



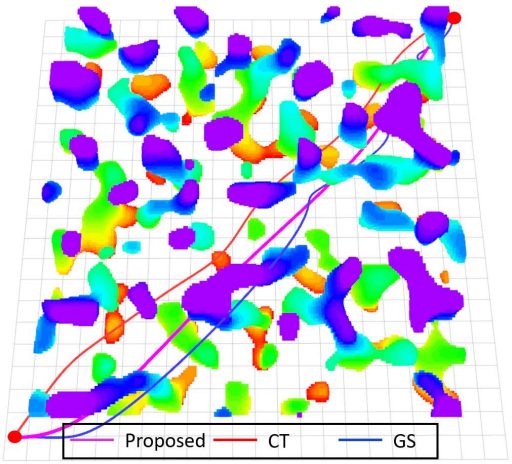
(a) Random map (0.2 pillars/m^2)



(b) Perlin noise map



(c) Random map (0.2 pillars/m^2



(d) Perlin noise map

Figure 2.11: Comparisons of different trajectory optimization methods on two different maps. The EO method is shown in *purple*, the CT method is shown in *red* and the GS method is shown in *blue*.

velocity profile may fail to provide a good initial guess when the initial path is not in a regular shape. For the EO method, since it starts from a time-parameterized B-spline trajectory, there is no need for time allocation.

From the efficiency perspective, for the GS method to achieve a similar jerk cost to the EO method, it needs run-times of 0.062 s, 0.075 s and 0.206 s on average for the three densities, which are significantly longer than the EO method. Note that the GS method is based on a non-linear optimization formulation. In the experiments, the non-linear optimization of the GS method is terminated when the non-linear solver (NLOPT [56]) reports convergence. For the GS method to converge to a compatible solution to that of our EO method, it requires a significantly longer run-time. Moreover, the run-time of the GS method is sensitive to the density, since for cluttered environments, more segments of the polynomial are needed, which may affect the convergence rate of the GS method. However, the EO method has lower run-times for

the three different densities. The efficiency of the EO method is affected by the total number of control points, but we observe that the efficiency difference is minor for the different densities.

Analysis of the Run-Time Efficiency

In this section, we test our replanning system, combining the EBK search with EO refinement. For the EBK search, we adopt EBK-D1 since it is the most efficient and the trajectory quality is satisfactory as verified in Sect. 2.6.2. We provide two different maps, namely, the random map and Perlin noise map. We evaluate the run-time efficiency of our replanning system, and list the statistics of all components in Tab. 2.4. The overall trajectory illustrating the whole round trip is shown in Fig. 2.12. As shown in Tab. 2.4, on the random map, the EBK-D1 method consumes an average computing time of $0.017\,s$ with a standard deviation of $0.01\,s$. The elastic tube expansion method can be finished in $0.002\,s$, and the optimization can be done in $0.021\,s$. On the Perlin noise map, which contains unstructured 3D obstacles, our method has a similar performance, showing that our method works well in complex 3-D environments.

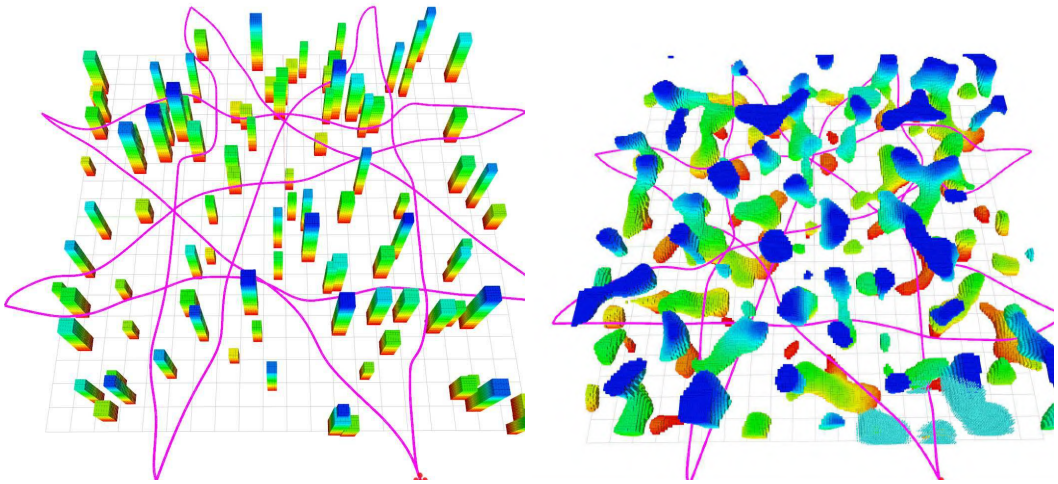
Comparison of the Replanning Framework

In this section, we conduct a system comparison with the SMP method [22]. We use SMP-U3 since SMP-U5 cannot work in real-time. Moreover, we conduct an ablation test, which excludes the initial-state aware EBK search and adopts the naive position-only A* search combined with EO local reshaping. We call the method for the ablation test A*-EO. The ablation test validates the critical role of the kinodynamic search in replanning.

We set up a challenging obstacle-cluttered 3-D complex simulation environment containing walls, 3-D steps (free space below) and pillars, as shown in Fig. 2.13. The replanning strategy is choosing a local goal state on a given straight-line guiding path with a local replanning range

Table 2.4: Run-time analysis on different maps

Maps	# **Replans**	**Time (s)**	**EBK Search**	# **Opt.**	**Time (s)**	**Tube Expan.**	**Traj. Opt.**	**Total Opt.**
Random map (0.25 pillars/m^2)	76	Avg	**0.017**	993	Avg	**0.002**	**0.021**	**0.023**
		Max	0.049		Max	0.009	0.043	0.044
		Std	0.010		Std	0.001	0.010	0.010
Perlin Map	19	Avg	**0.014**	1044	Avg	**0.002**	**0.028**	**0.030**
		Max	0.026		Max	0.008	0.058	0.061
		Std	0.006		Std	0.001	0.010	0.011



(a) Replan on the random map (b) Replan on the Perlin noise map

Figure 2.12: Illustration of our replanning system on different maps.

of 5 m. The simulated quadrotor is equipped with a depth camera which has a sensing range of 4 m. The maximum velocity and acceleration bound are set to 2 m/s and 3.2 m/s^2 for each axis. For SMP-U3, we use a conservative bound with maximum acceleration 1.0 m/s^2 for each axis since SMP-U3 can only work with a narrow velocity and acceleration range. Using a large dynamic range for SMP-U3 will result in a very sparse primitive graph, which does not even contain one feasible solution. For the EBK search of our method, we use EBK-D1 with a $60 \times 60 \times 20$ uniform grid. For the EO approach, 12 control points are refined as the window moves forward. We evaluate the replanning system from the trajectory statistics and time efficiency perspectives, as shown in Tab. 2.5.

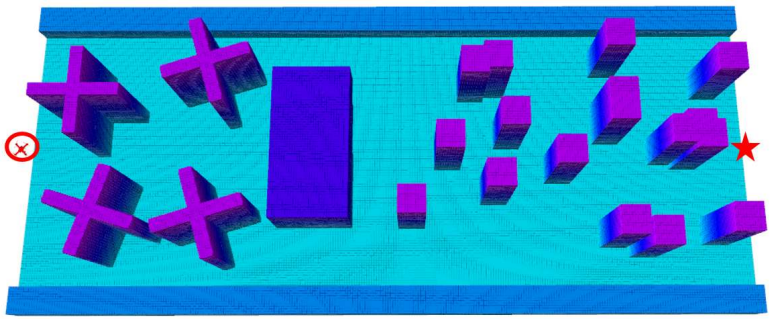
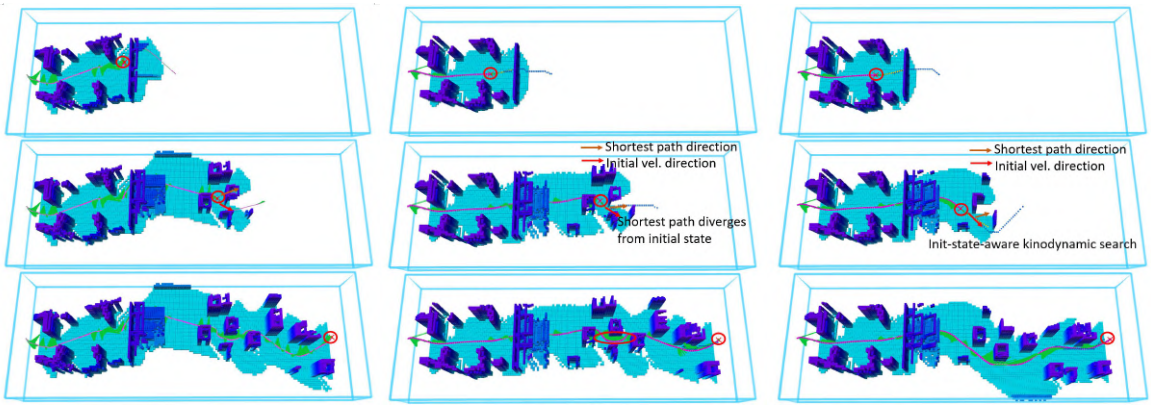


Figure 2.13: Illustration of the simulated environment for benchmarking.

Critical snapshots of the three methods are shown in Fig. 2.14. For SMP-U3 (Conservative) in Fig. 2.14a, when the quadrotor observes the 3-D step, it chooses to circle around instead of directly flying through the space below since the latter action requires a large acceleration range. This phenomenon demonstrates that SMP-U3 (Conservative) sacrifices maneuverability due to



(a) SMP-U3 (Conservative) (b) A*-EO (c) Our method

Figure 2.14: Illustration of different replanning methods in the same simulated environment. The trajectory is shown in *purple*, and the acceleration profile is marked in *green*. For the replanning cases where the shortest path direction is inconsistent with the initial state, we mark the initial velocity with a *red arrow* and the shortest path direction in *brown*.

the restriction of the dynamic range. SMP-U3 takes the initial state into account, as shown in the middle snapshot in Fig. 2.14a. The necessity of using the kinodynamic search instead of the position-only A* search is identified by the totally different maneuvers in Fig. 2.14b and Fig. 2.14c. When the quadrotor enters the region of the pillars, it has two distinct choices, namely, "pass on the left" or "pass on the right". For the A*-EO method, as shown in the middle snapshot in Fig. 2.14b, the quadrotor tends to choose the direction purely based on the shortest path. So there are cases where the quadrotor is passing in one direction and suddenly switches to the opposite direction due to finding a new shortest path, which results in an inconsistent and non-smooth replanning trajectory. Compared to A*-EO, the kinodynamic EBK search provides an initial-state-aware trajectory for the local reshaping, as shown in Fig. 2.14c. With the EBK search, the overall trajectory is clearly more natural and the replanning is more consistent.

As shown in Tab. 2.5, SMP-U3 (Conservative) is efficient and has an average computation time of 0.010 s. However, since the acceleration bound is conservative, the maneuverability is sacrificed and the trajectory duration is 33.3 s, longer than the other two methods. Note that we use acceleration-controlled SMP [22] with unconstrained QP [7] reparameterization. Since the unconstrained QP has no dynamical feasibility guarantee, the maximum acceleration is 1.3 m/s^2, which exceeds its designed maximum acceleration (1 m/s^2). Compared to SMP-U3 and A*-EO, our method has the lowest total jerk cost and the improvement is achieved by incorporating the kinodynamic search. The average velocity of our method is 1.37 m/s, slightly higher than the A*-EO, due to the fact that the kinodynamic search can reduce sharp decelerations. The maximum acceleration of our method is 3.18 m/s^2, which obeys the dynamical feasibility constraint. Compared to the position-only A*-EO method, the jerk cost is reduced by $10\,\%$ by

Table 2.5: Performance of different replanning systems

Method	**Trajectory Statistics**				**Time Efficiency**(s)		
	Traj. Dura.(s)	Jerk Cost (m^2/s^5)	Mean Vel. (m/s)	Max Acc. (m/s^2)	Ave	Max	Std
SMP-U3 (Conservative)	33.3	956.3	1.14	**1.3**	**0.010**	0.027	0.004
A*-EO	25.2	723.0	1.31	3.18	0.014	0.062	0.011
Our method	**24.4**	**648.1**	**1.37**	3.18	0.019	0.080	0.012

using the kinodynamic search. Considering that the advantages of the EBK search are outstanding for part of the trajectory where potential inconsistency exists, this quantitative improvement still faithfully identifies the gain of using the kinodynamic search. Note that navigating through this challenging environment with enough agility already requires considerable control efforts, and the $10\,\%$ cost reduction represents a reasonable overall improvement.

2.6.3 Experimental Results

We conduct onboard experiments[7] with the two vision-based testbeds to show the general applicability of the proposed framework. For onboard testing, the parameters are as follows: the time step Δ_t is set to $0.35\,s$; the maximum velocity and maximum acceleration are set to 1.2 m/s and 2.0 m/s^2, respectively; [8] and the local planning range is set to $10m \times 6m \times 1.1m$. (The corresponding grid size is $55 \times 35 \times 6$.)

Monocular-vision-based indoor navigation

As shown in Fig. 2.15, our replanning system works in complex 3-D environments with only a local map. The quadrotor is commanded to navigate to a 3-D position where the environment is previously unknown. The whole trajectory and the final accumulated map is shown in Fig. 2.16. The trajectory length of the final trajectory is $18.6\,m$ and total trajectory execution time is 43.4 s. The average velocity of the quadrotor is 0.45 m/s with a maximum velocity of 0.79 m/s. The maximum acceleration of the trajectory is 0.58 m/s^2, the whole trajectory is dynamically feasible, and there are a total 125 calls of the EBK search (active mode), with an average computation time of 0.010 s. There are 105 calls of EO. The average computation times of the elastic tube expansion and the optimization are 0.001 s and 0.031 s, respectively.

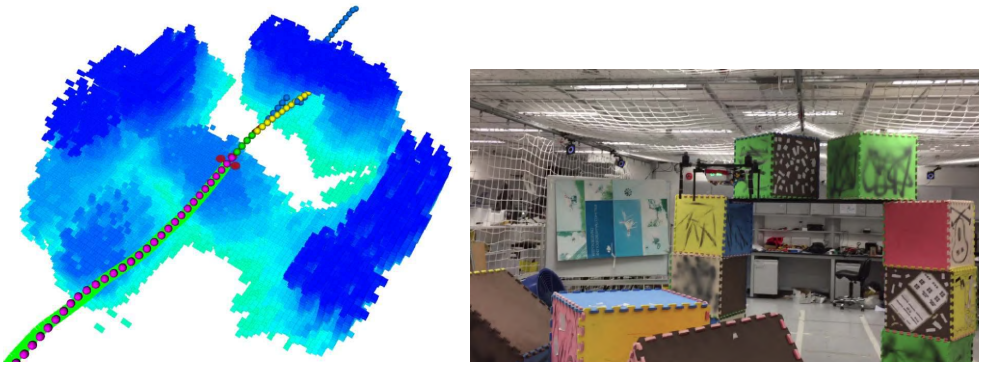


Figure 2.15: Illustration of the snapshot of the indoor replanning with the monocular vision. In (a), the control points found by EBK (*blue*), executed control points (*pink*), committed control points (*green*), and control points under optimization (*yellow*) are marked. The corresponding indoor environment is shown in (b).

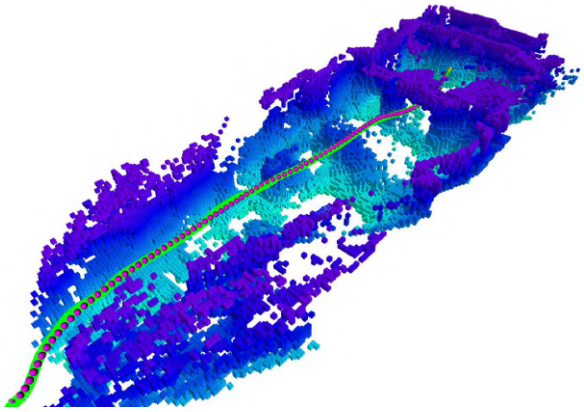


Figure 2.16: Illustration of the whole trajectory and final accumulated map for the indoor replanning using the monocular perception system.

Dual-fisheye-based indoor round-trip navigation

As shown in Fig. 2.17 and Fig. 2.18, with omnidirectional perception, our quadrotor testbed is able to fly a round-trip without controlling the yaw angle. Online mapping using the dual-fisheye cameras is challenging due to the high distortion of the images acquired from the fisheye cameras. Although the uncertainty of the map is larger than the monocular case, our replanning system is still able to robustly avoid the unexpected obstacles and navigate in the unstructured cluttered environment.

[7] `https://www.youtube.com/watch?v=sg46XT9-o1k`

[8] The speed limit and acceleration limit are set to be slightly conservative considering the perception delay in onboard experiments.

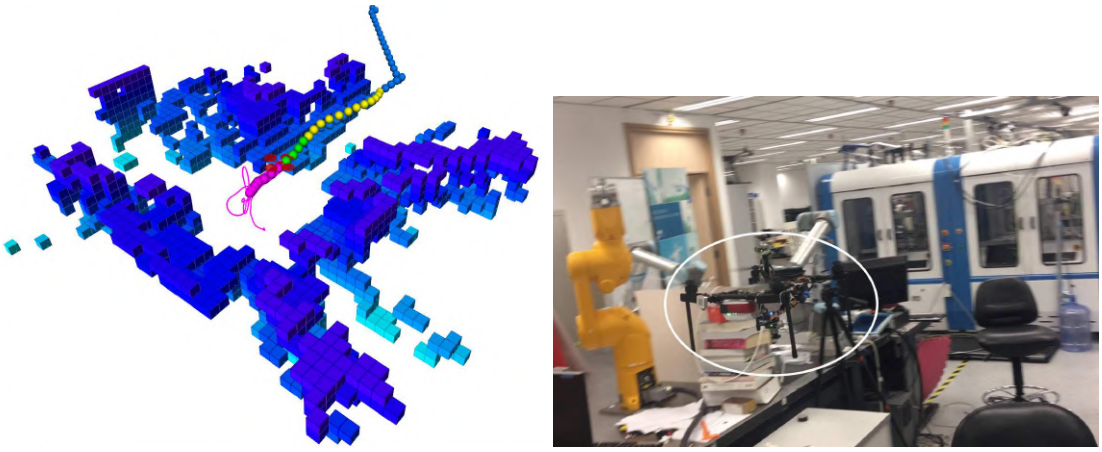


Figure 2.17: Illustration of the snapshot of the indoor replanning with the omnidirectional vision. With a fixed yaw angle, the obstacles around the quadrotor can be mapped.

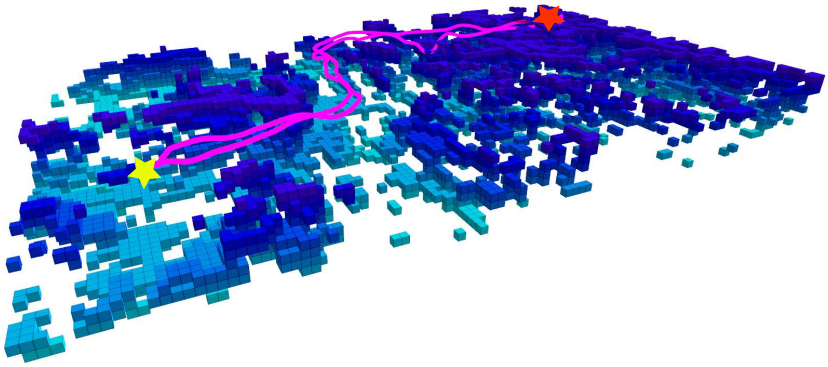
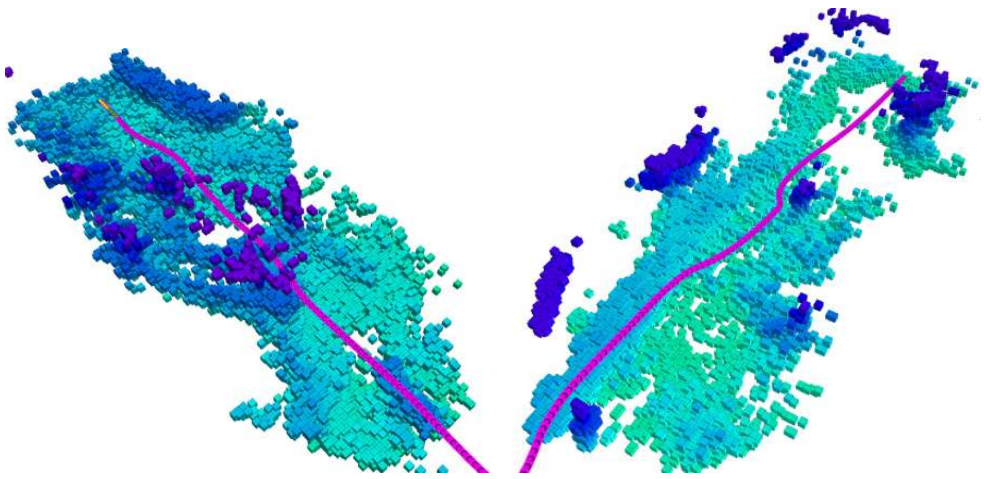


Figure 2.18: Illustration of the whole trajectory for the replanning using the dual-fisheye omnidirectional perception system. Unlike the experiments using the monocular testbed, we can achieve round-trip flight in an unknown complex indoor environment.

Outdoor Replanning Performance

As shown in Fig. 2.19, we demonstrate outdoor experiments using the monocular vision testbed. For Fig. 2.19a, the trajectory length is $19.6\,m$ and total execution time is $41.3\,s$. The average velocity of the quadrotor is $0.49\,m/s$. The maximum acceleration of the trajectory is $1.06\,m/s^2$, and the whole trajectory is dynamically feasible. There are a total 35 calls of EBK search (passive mode) with an average computation time of $0.025\,s$. The average computation times of the elastic tube expansion and optimization are $0.001\,s$ and $0.030\,s$, respectively. For the experiment shown in Fig. 2.19b, the performance and trajectory statistics are similar.



(a) (b)

Figure 2.19: Illustration of the outdoor experiments using the monocular vision: A flight through a pavilion is shown in (a) (part of the map on the top is cut for visualization purposes); a flight avoiding trees is shown in (b).

2.7 Conclusion

In this chapter, we present an efficient kinodynamic replanning framework by exploiting the advantageous properties of the B-spline. The proposed EBK search algorithm is flexible and provides a user-specified parameter which can be used to control the algorithm efficiency and solution quality. The problem of the B-spline-based kinodynamic search on a spatial grid is characterized in detail, and the theoretical performance of the EBK search is analyzed. To compensate for the discretization, we propose an elastic optimization process. We combine the two components into a receding horizon framework. Detailed analysis and comprehensive experiments are carried out to validate the performance. Systematic comparisons against the state-of-the-art are provided to verify the claims. The replanning framework is efficient and complete, and can be used in various kinds of exploration tasks and different kinds of quadrotor testbeds. The current limitation of the framework lies in the fact that for EBK search we are using the uniform B-spline on the spatial grid, which will result in limited B-spline patterns. In the future, we expect to explore NURBS for kinodynamic search, which will allow for various motion patterns in the kinodynamic search.

CHAPTER 3

BEHAVIOR PREDICTION FOR AUTONOMOUS VEHICLES

3.1 Scientific Background and Motivation

In recent years, there has been growing interest in building autonomous vehicles which can navigate naturally in complex environments. Despite the fact that perception techniques are maturing, autonomous vehicles are still criticized for being over conservative or socially incompliant. To make wise decisions, the planning module of autonomous vehicles needs to reason about long-term future outcomes, which requires predicting future behaviors in three to five seconds.

There is an extensive literature on the prediction of vehicles, which can be divided into two categories [57]: **intention/maneuver/behavior** prediction and **motion/trajectory** prediction. The two categories have different outputs: the behavior prediction outputs high-level behaviors such as lane keeping (LK), lane change (LC), etc., while the motion prediction produces a time-profiled predicted trajectory. Recent works suggest that the motion prediction can be augmented by the behavior prediction to model the multimodal nature of future motion [57–60]. In this chapter, we focus on the behavior prediction problem, and our method can be easily incorporated into motion prediction algorithms to generate time-profiled prediction.

An overview of prediction methods is provided in Sect. 3.2, where we find that many proposed methods are *detection-based* and only have a limited prediction horizon. For example, a lane change is detected only 1.0 s to 1.7 s before the prediction target crosses the dividing line [10–13]. To avoid generating false alarms in the face of noisy maneuvers such as zigzagging, these methods tend to ***detect*** *the behavior only when it is clearly happening*. Detection-based methods can achieve high accuracy and are suitable for a system like ADAS. However, the short prediction horizon is problematic for the planning module, as shown in Fig. 3.1.

To meet the needs of the planning module, the prediction is required to *anticipate the likelihood of the behavior even before it is clearly happening*, and we call prediction of this kind prediction over an **extended horizon**. Such prediction typically requires reasoning about the

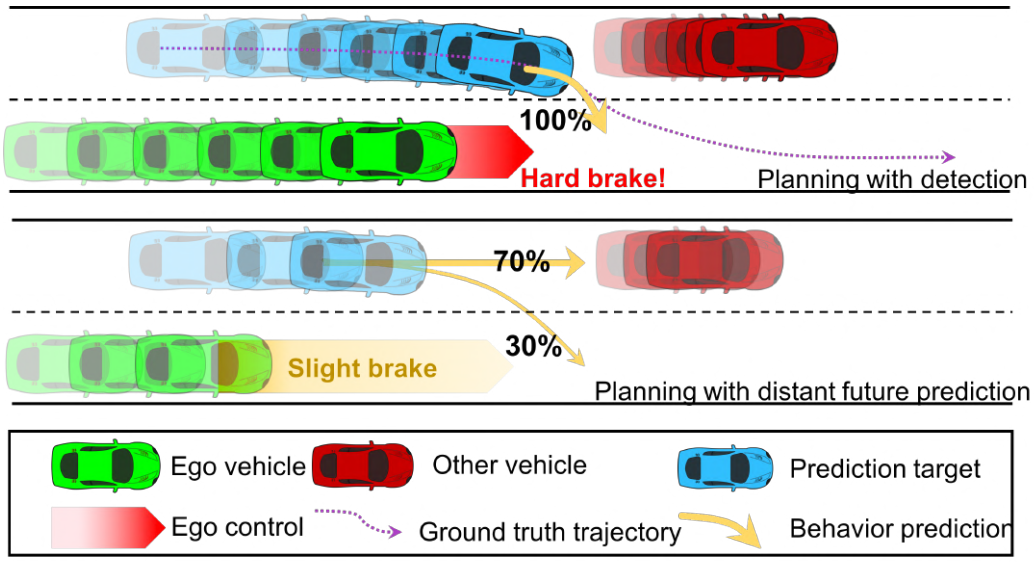


Figure 3.1: Illustration of the benefit of extending the prediction horizon. Assume that the **green** vehicle is the ego vehicle with the planning module, while the transparency of the vehicles represents the time elapsed. For a detection-based method, as shown on the top, the LC prediction is given when the **blue** vehicle has a clear LC pattern, which may result in a sudden braking of the ego vehicle due to the late discovery. However, from the interaction point of view, the **blue** vehicle is moving at a high speed and is blocked by the slowly moving **red** car. The **blue** vehicle has two interaction choices: brake to avoid collision or merge into the other lane. By learning from a large number of interaction patterns, the likelihood of LC can be estimated, even before the **blue** vehicle has a clear LC maneuver. More examples can be found in the video `https://www.youtube.com/watch?v=SuQzxAusU_0`.

behavior in three to five seconds. However, it is a challenging problem due to the increasing uncertainty with respect to the distance of the future. For example, at four seconds before a LC, the observed maneuver lacks features to make the LC distinguishable.

Obviously, additional information is required to reduce the uncertainty of the future. We uncover that social interaction is highly informative for prediction over the extended horizon. An illustrative example is shown in Fig. 3.1. In [1], Alahi *et al.* proposed a learning-based social pooling strategy for modeling pedestrian interactions, which became very popular and was later applied to vehicles [58]. However, using their methods agents appearing in the same spatial location will be weighted equally in spite of their different dynamics, which is problematic for a highly dynamic driving scenario, as shown in Fig. 3.2.

In this chapter, we propose a novel vehicle behavior interaction network (VBIN) for modeling vehicle interactions. The novelty is that the VBIN is capable of *learning to weight* the social effect of another agent on the prediction target based on their maneuver features and also relative dynamics (e.g., relative positions and velocities). Each interaction pair is dynamically

weighted and then used to evaluate the total social effect on the prediction target. Contrary to the social pooling strategy, which is effective in modeling low-speed collision avoidance behaviors of pedestrians, our proposed method is suitable for highly dynamic driving scenarios where the dynamics of agents affect their importance in social interactions. Our method is end-to-end trainable.

We compare the proposed method with state-of-the-art interaction-aware models through systematic experiments on a publicly available real-world highway dataset, and show in Sect. 3.4 that our novel VBIN structure can achieve significant improvements for behavior prediction over the extended horizon in terms of various critical metrics.

3.2 Related Works

Detection and prediction. Detection-based methods, such as maneuver recognition approaches, have been studied extensively. Houenou *et al.* [10] identify maneuver patterns such as LK and LC based on a handcrafted model which evaluates the similarity between the vehicle trajectory and lane center line. However, user-specified parameters and thresholds are required. Mandalia *et al.* [11] classify the LC maneuver using support vector machine (SVM), while Schlechitriemen *et al.* [12] use Gaussian mixture models to estimate driving intentions. Woo *et al.* [13] also adopt an SVM, and suggest using trajectory prediction to reject false alarms from the intention estimations. These methods detect LCs based on the target's own maneuver pattern and only have a limited prediction horizon. In contrast, our method captures the driver intention, even in the absence of any clear maneuver pattern, by making use of the vehicle behavior interactions.

Behavior prediction and trajectory prediction. Many methods focus on trajectory prediction and generate a time-profiled trajectory directly. Regression methods, such as linear regressions [61] and non-linear Gaussian process regression models [62–64], are extensively studied. Recurrent neural networks (RNNs) have recently been widely applied to various trajectory prediction tasks including trajectory prediction for pedestrians and vehicles. For vehicles, apart from predicting trajectories, many methods also aim to predict high-level intentions [12, 13, 65, 66]. Moreover, some recent works suggest that trajectory prediction can be augmented by the prediction of high-level semantic behaviors, e.g., LCs and insertions, due to the multimodal nature of the problem [57–59]. In this chapter, we focus on the behavior prediction problem.

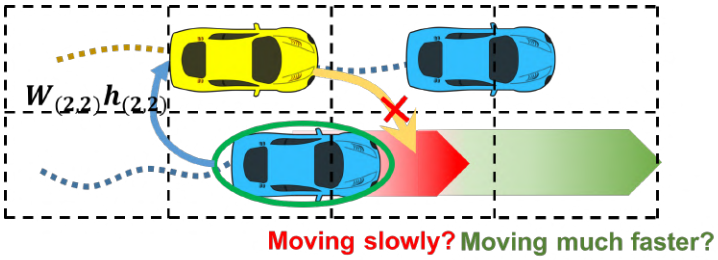


Figure 3.2: Illustration of the popular social pooling strategy [1]. The RNN hidden states in the same spatial cell are pooled and passed to a fully connected layer to generate the total social effect on the target vehicle (**yellow**). Vehicles which have totally different dynamics in the same cell will share the same weight. However, as shown in the Illustration, if the vehicle that is highlighted with a circle is moving slowly, it is supposed to has a large weight since it blocks the LC route of the target vehicle, but if it is moving much faster than the target vehicle, it should has little impact on the target vehicle's LC.

Learning-based methods for various prediction tasks. Many learning-based prediction methods follow an RNN encoder-decoder structure [2] and use it for predicting the trajectory of pedestrians [67–69] and vehicles [58, 68]. RNNs are an alternative to the traditional methods (e.g., SVM, Gaussian process) [70, 71] for capturing maneuver patterns. Some works move one step further and consider the multi-agent interactions among pedestrians [1, 68, 72, 73] and vehicles [58, 68] in the RNN structure. The interaction-aware models for vehicles mainly adopt a social pooling strategy using RNN hidden states, which may be problematic for a highly dynamic scenario, as elaborated in the following.

Modeling interactions. Modeling social interaction in RNNs started with the seminal work by Alahi *et al.* [1], which proposes a social occupancy grid pooling mechanism, as shown in Fig. 3.2. This design is very effective for pedestrian trajectory prediction in a crowd [68, 72, 74], and is suitable for unstructured environments and modeling collision avoidance behaviors. It is directly applied to vehicle prediction in [68] and [58]. However, for highly dynamic on-road driving, spatial locations should not be the only reference for weighting social effects, as shown in Fig. 3.2. Going beyond these methods, our proposed VBIN can learn to weight the interactions automatically based on the relative dynamics of both agents.

3.3 Methodology

3.3.1 Problem Formulation

The motivation behind our proposed behavior prediction method is to facilitate the development of socially compliant planning algorithms. To this end, we describe the problem from the planning perspective to make clear what requirements the planning module for the prediction.

We follow multi-policy decision making (MPDM) [75, 76] for a theoretically-grounded multi-agent planning formulation. Note that our proposed method is interaction-aware and can serve as a further developed prediction module for this planning framework.

At time t, a vehicle v can take an action $a_t^v \in \mathcal{A}^v$ to transition from a state $x_t^v \in \mathcal{X}^v$ to x_{t+1}^v and make an observation $z_t^v \in \mathcal{Z}^v$. For example, a state x_t^v can be a tuple of the pose, velocity and acceleration, an action a_t^v can be a tuple of the controls for steering, throttle, braking, etc., and an observation z_t^v can be a tuple of estimated poses and velocities of other vehicles. Let e denote the ego (i.e., controlled) vehicle and V_e denote the set of vehicles in a local neighborhood of the ego vehicle e. Let $z_t^{e,v}$ denote the observation made by the ego vehicle about its nearby vehicle $v \in V_e$, let x_t include all state variables x_t^v for all vehicles at time t, let $a_t \in \mathcal{A}$ be the actions of all vehicles, and let $z_t \in \mathcal{Z}$ be the observations made by all vehicles. From the planning perspective, the goal is to find an optimal policy π' that maximizes the expected reward over a given decision horizon H, where a policy is a mapping $\pi : \mathcal{X} \times \mathcal{Z}^v \to \mathcal{A}^v$. π' can be obtained by solving a partially observable Markov decision process (POMDP) with the following probability transition:

$$p(x_{t+1}) = \prod_{v \in V} \iiint_{\mathcal{X}^v \mathcal{Z}^v \mathcal{A}^v} p^v(x_t^v, x_{t+1}^v, z_t^v, a_t^v) da_t^v dz_t^v dx_t^v, \tag{3.1}$$

where $p^v(x_t^v, x_{t+1}^v, z_t^v, a_t^v)$ denotes the joint density for a single vehicle. The MPDM make the assumption on the driver model $p(a_t^v|x_t^v, z_t^v)$ that the agents are executing a policy/behavior from a discrete set of behaviors, such as LC and LK. Mathematically, the assumption is

$$p(a_t^v|x_t^v, z_t^v) = p(a_t^v|x_t, z_t^v, \pi_t^v) \underbrace{p(\pi_t^v|x_t^v, z_t^v)}_{\text{behavior decision}}, \tag{3.2}$$

where π_t^v belongs to the discrete set of behaviors Π. The ego vehicle needs to infer $p(\pi_t^v|x_t^v, z_t^v)$ for all $v \in V_e$ via $p(\pi_t^v|\mathbf{z}_{0:t}^e)$, where $\mathbf{z}_{0:t}^e$ is a time series of observations of the nearby vehicles.

Many prediction methods actually simplify the problem by further approximating $p(\pi_t^v|\mathbf{z}_{0:t}^e)$ using $p(\pi_t^v|\mathbf{z}_{0:t}^{e,v})$, where $p(\pi_t^v|\mathbf{z}_{0:t}^{e,v})$ denotes the observations of vehicle v only. Specifically, the predicted behavior is only based on past observations of each vehicle itself. The approximation ignores the potential interactions and results in a short prediction horizon, as shown in Fig. 3.1. It also becomes clear that learning-based methods try to find a parametric representation of the policy likelihood $p(\pi_t^v|\mathbf{z}_{0:t}^e, \boldsymbol{\theta})$ using the neural network parameters $\boldsymbol{\theta}$. The idea of the proposed method is to uncover the potential interactions lying inside $\mathbf{z}_{0:t}^e$ through the specially designed VBIN structure and yield a better parametric estimation of $p(\pi_t^v|\mathbf{z}_{0:t}^e)$ which holds for a long decision horizon H.

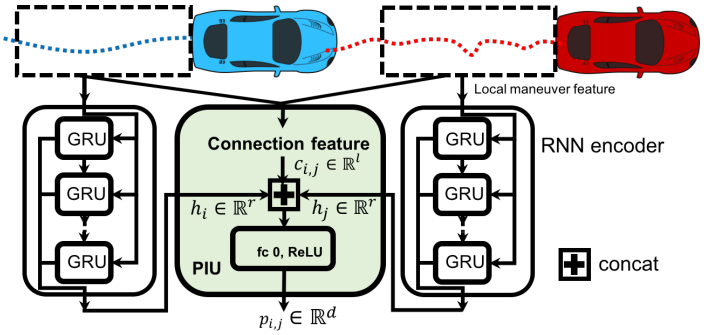


Figure 3.3: VBIN building block: pairwise interaction unit (PIU). RNNs are implemented using gated recurrent units (GRUs) [2], and the hidden states are used as the input of the PIU.

3.3.2 RNN Encoder Network

As introduced in Sect. 3.3.1, the network is supposed to be amenable to encoding the observed states of all vehicles. To accomplish this task, we start with obtaining the encoding for each individual agents, which provides a compact representation of the vehicle's maneuver. We adopt the popular RNN encoder network [2] to obtain the maneuver encoding for each individual vehicle. We first extract features from a time series of the observed states of a given vehicle via a feature selection function $f_m(\mathbf{z}_{0:t}^{e,v})$. The selection of features depends on the design of the behavior class, and in Sect. 3.4.3, we provide a practical example of the feature selection for LC behavior.

For brevity, we do not include details of the RNN encoder structure, and refer interested readers to [1] and [68] for the details. It is worth noting that, during the training phase, all the RNNs share the same set of weights. After the RNN encoding, the maneuver history of each vehicle $v \in V_e$ is encoded in a vector $h_v \triangleq \text{RNN}(f_m(\mathbf{z}_{0:t}^{e,v})) \in \mathbb{R}^r$, where r denotes the size of the RNN encoding. All the encodings can be computed and stored before the inference of the VBIN.

3.3.3 Vehicle Behavior Interaction Network (VBIN)

Pairwise interaction unit (PIU). The basic element of the VBIN is the pairwise interaction unit (PIU). The PIU *learns to weight* the social effect of an interaction pair based on their maneuver histories and relative dynamics. As shown in Fig. 3.3, the PIU takes three inputs: two RNN encodings h_i and h_j of a pair of vehicles, with $i \in V_e$, $j \in V_e$ and $i \neq j$, and the connection feature $c_{i,j}$. The connection feature $c_{i,j}$ is extracted via another feature extraction function, $c_{i,j} = f_c(\mathbf{z}_{0:t}^{e,i}, \mathbf{z}_{0:t}^{e,j}) \in \mathbb{R}^l$, which is based on the history of both vehicles and represents the relative states of both vehicles. For example, $c_{i,j}$ can include the relative positions and relative

velocities of the two vehicles.

In other words, the hidden states h_i and h_j are responsible for describing the maneuver in each vehicle's *local reference frame*, and are invariant no matter which prediction target it is. The connection feature $c_{i,j} \in \mathbb{R}^l$ describes the "translation" of two reference frames and the dynamics of both vehicles. The PIU is dedicated to "weighting" the two hidden states given the additional information of translation and vehicle dynamics. Unlike the conventional social pooling [1], which weights the hidden states purely based on spatial location, the PIU learns different weighting strategies for different relative states of each interaction pair. The output of the PIU unit is denoted as $p_{i,j} \triangleq \text{PIU}(h_i, h_j, c_{i,j}) \in \mathbb{R}^d$, with d denoting the size of the PIU embedding.

Neighborhood interaction unit (NIU). Before stepping into the details of the NIU, we give a formal definition of "neighborhood". In the series of works on pedestrian trajectory prediction [1], [58] and [72], the "neighborhood" is defined by a grid centered at the prediction target and each vehicle is associated with one cell. However, vehicles travel in a semi-structured environment where there are semantic elements, such as lanes, which makes the occupancy grid not the best choice. [57] suggests selecting a fixed number of reference vehicles around the prediction target based on the environment semantics such as lanes. Taking the highway scenario for example, vehicles tend to interact with the nearest vehicles in the current and neighboring lanes, and these neighboring vehicles will be informative for interaction modeling [57].

In this section, we provide a selection strategy for a highway scenario as an example, as shown in Fig. 3.4. The selection process is as follows: 1) Select the two vehicles at the front and rear in the current lane. 2) Select the two vehicles with the closest longitudinal coordinates to the target vehicle in the neighboring lanes. 3) Select the vehicles immediately at the front and the vehicle immediately at the rear w.r.t. the neighboring vehicle. In the case of the non-existence of such neighboring vehicles, we create a virtual vehicle with the same longitudinal velocity as the target vehicle and a far longitudinal relative distance (e.g., 100 m) in the same direction. Note that the neighboring vehicles are determined online. For the consecutive rounds of predictions, the neighboring vehicles may vary but the prediction results are still consistent, as shown in the video.

Let $\mathcal{N}_i$ be the set of selected neighboring vehicles around the target vehicle $i \in V_e$. We denote the number of selected neighboring vehicles as $N_{\text{nbr}} = |\mathcal{N}_i|$. Like [57], we use a fixed N_{nbr} for each prediction target. As shown in Fig. 3.4, we establish the PIU link between the

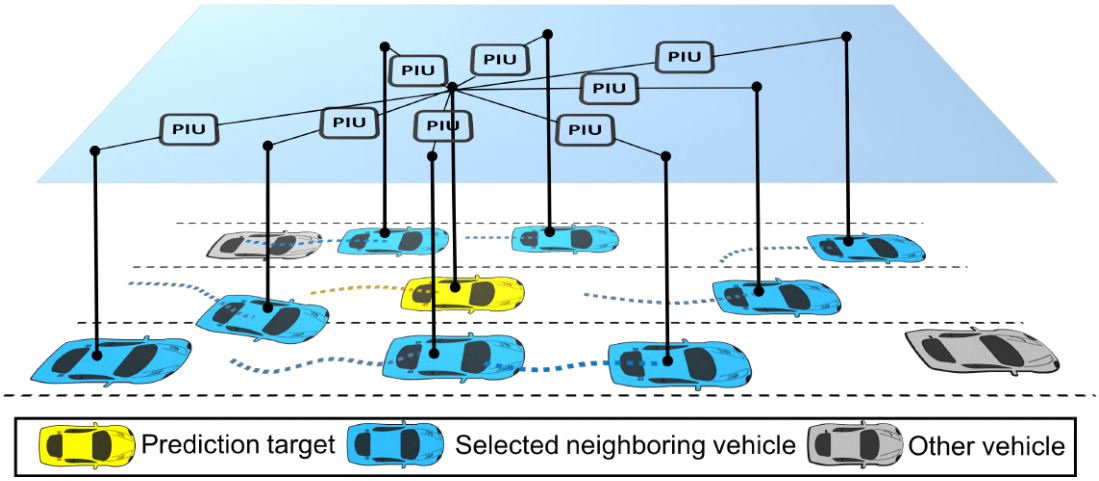


Figure 3.4: Illustration of the PIU connections for each selected neighboring vehicle.

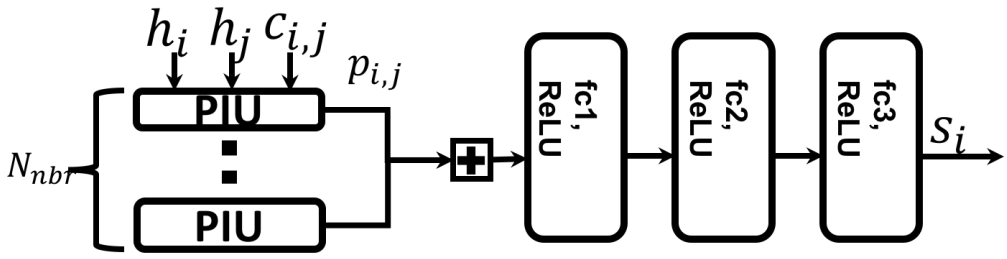


Figure 3.5: VBIN building block: neighborhood interaction unit (NIU). All the PIUs are identical.

prediction target i and each selected vehicle $j \in \mathcal{N}_i$. The N_{nbr} PIU embeddings are concatenated and passed to three cascaded fully connected layers with ReLU activation. The output of the NIU unit is denoted as $s_i \in \mathbb{R}^n$, with n denoting the NIU embedding size. At a high level, s_i represents the total social effect of the neighboring vehicles on the prediction target i.

Behavior decoding. As mentioned above, s_i represents the social effect applied to vehicle i. We then concatenate s_i with its original local maneuver encoding h_i. The concatenated vector now contains both the features extracted from its own maneuver history and the features extracted from the neighborhood interaction. Applying the above process to a total number of N_a prediction targets, we obtain a social batch, with each row representing the combined encoding for each individual vehicle, as shown in Fig. 3.6. The social batch is passed to cascaded fully connected layers with ReLU activation and another softmax layer to decode the output, which is the likelihood $b \in \mathbb{R}^{N_a \times C}$ of C behavior classes for each predicted vehicle.

Scalable deployment. In the following, we show that the inference time of the VBIN does not scale with N_a and is amenable to scalable deployment. Specifically, since the PIUs and NIUs are all identical, we can precompute and reuse the following tensors: 1) precompute the connection features for the neighborhood of each agent, which forms a tensor of size $N_a \times N_{\text{nbr}} \times l$, and store

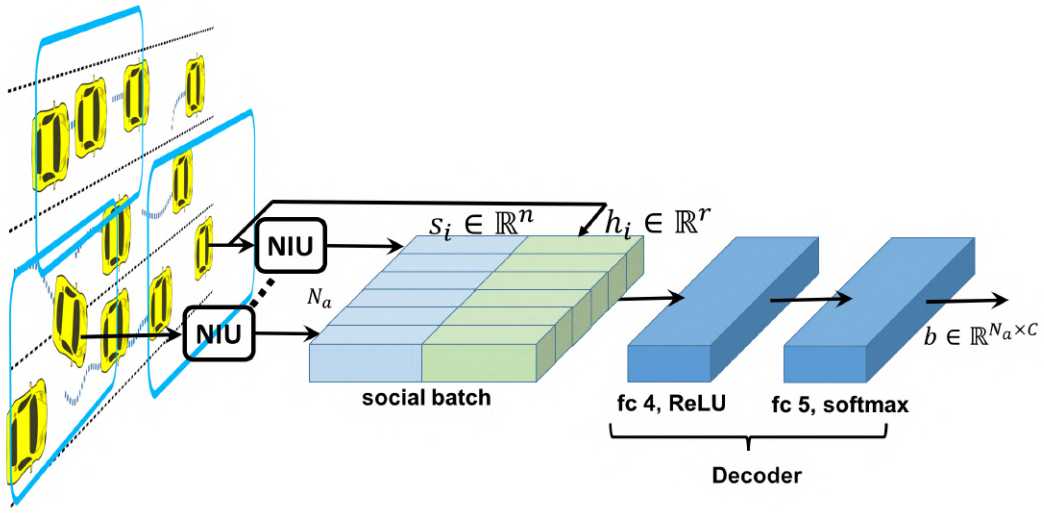


Figure 3.6: Illustration of the VBIN structure. All the NIUs are identical.

an index tensor of size $N_a \times N_{\text{nbr}} \times 1$, which records the index of the corresponding neighboring vehicle; 2) use the RNN to encode N_a local maneuver features and get the resulting hidden tensor of size $N_a \times r$. After the preparations, we can run a forward pass, as shown in Fig. 3.7.

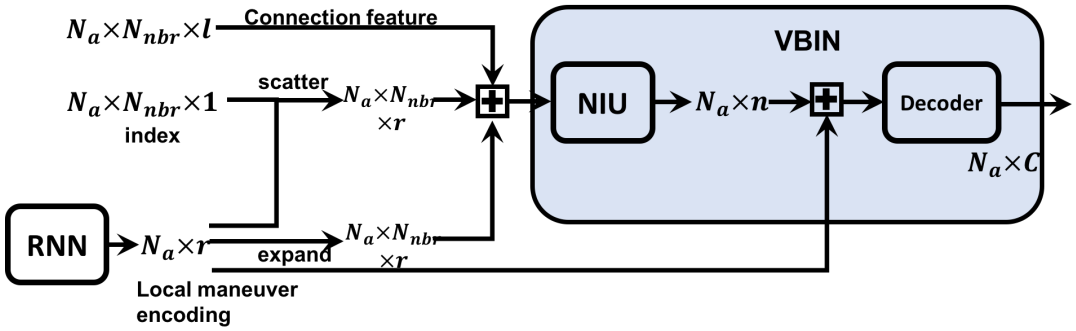


Figure 3.7: Illustration of the dataflow of the system.

The input tensor of the NIU is of size $N_a \times N_{\text{nbr}} \times (2r + l)$, and it combines each pair of hidden states and the corresponding connection feature. Inside the NIU, the PIU resizes and embeds the input tensor to size $N_a \times N_{\text{nbr}} \times d$, and its second and third dimension will be further flattened. After the output layers of the NIU, the tensor is of size $N_a \times n$, and it will be concatenated with the original hidden states and decoded, as shown in Fig. 3.6. N_a can be regarded as the batch size and will not affect the inference time.

3.4 Experimental Results

We apply our proposed method to a highway scenario to validate the performance and illustrate the implementation details for highway application.

3.4.1 Dataset

We use a real traffic dataset, NGSIM, which is publicly available online [77]. We use data collected from highways US-101 and I-80. The dataset consists of trajectories of real traffic captured at 10 Hz, and also provides mild, moderate and congested traffic conditions. We define the behavior classes as *lane keeping*, *lane change left* and *lane change right*. We extract $1,669$ LCs (and LKs) from the dataset and a total of 509 LCs (and LKs) are used for testing. The training and testing set are totally disjoint.

3.4.2 Baselines

We compare our method with the following methods:

- *Vanilla LSTM* (**VLSTM**). This is a simplified setting of our model, where we remove the interaction network VBIN.
- *Social LSTM* (**SLSTM**). This is from the seminal work [1] on interaction-aware trajectory prediction, and we change the output to the likelihood of behavior classes.
- *Convolutional social pooling LSTM* (**CLSTM**). This is a modification of SLSTM [58]. Both SLSTM and CLSTM use the social pooling strategy. The difference is that CLSTM uses convolutional layers for the pooled hidden states.
- *Semantic-based intention and motion prediction* (**SIMP**). We use the implementation from [57] and only adopt the intention prediction part.

CLSTM and SIMP are two state-of-the-art methods, originally verified on the NGSIM dataset. We adopt the publicly available implementation [78] of CLSTM and SLSTM, and train the model on our generated dataset. Since the source code of [57] is not officially available, we implement it according to the implementation details provided in [57].

3.4.3 Implementation Details

The output of the feature selection function $f_m(\mathbf{z}_{0:t}^{e,v})$ is shown in Tab. 3.1. The length of the observation window is set to 2 s (20 frames). For the sequence of local maneuver features to be fed to the RNN, we subtract the x^{lat}, x^{long} of the last frame of the sequence to remove the absolute translation. The length of the prediction window is set to 4s (40 frames). The criterion

Table 3.1: Feature selection for local maneuver encoding.

Feature	Description
x^{lat}	Lateral coordinate of the center of the vehicle
x^{long}	longitudinal coordinate of the center of the vehicle
$d^{\text{lat}}_{\text{clc}}$	Lateral distance to the current lane center normalized by lane width
v^{long}	longitudinal velocity in the lane reference frame
v^{lat}	Lateral velocity in the lane reference frame
θ	Vehicle orientation with respect to the lane longitudinal direction

for labeling LC is that in the prediction window the target vehicle crosses the lane dividing line. The labeling process is conducted in a sliding window manner starting from 8 s before the LC. All the methods output the likelihood of the three behavior classes, and the prediction decision is made by taking the class with the maximum likelihood. The structure of the RNN encoder is the same as [68] with 128 hidden states. The loss function is the negative log-likelihood (NLL) loss with three classes. All the models are implemented using Pytorch [79].

The connection feature consists of six elements: relative longitudinal and lateral distances, longitudinal and lateral velocities of the target vehicle, and longitudinal and lateral velocities of the neighboring vehicle. The detailed network specifications are shown in Tab. 3.2.

Table 3.2: Network specifications of the VBIN.

PIU				**NIU**			**Decoder**		
r	l	d	fc0 I/O	fc1 I/O	fc2 I/O	fc3 I/O	n	fc4 I/O	fc5 I/O
48	6	64	102/64	512/400	400/400	400/48	48	96/48	48/3

3.4.4 Evaluation Metrics

Accuracy. Precision, recall and F1-score are the three representative metrics quantifying the accuracy for classification problem. In the case of LC prediction, we define two positive classes (lane change left and right) and one negative class (LK), similar to [57] and [71]. However, we argue that for different times-to-lane-change (TTLCs), there is particular emphasis on different errors. We define the the false negatives (FNs) inside the duration of TTLCs less than 1.5 s as *critical FNs*, since it is too close to the actual LC and any mis-detection will be extremely dangerous. Note that the criterion is more strict than that adopted by [13], where the FN is tolerated as long as the vehicle hasn't crossed the dividing line. We define the false positives (FPs) inside the duration of TTLCs larger than 5.5 s as *critical FPs*, since a lane change generally takes 3.0

to 5.0 s [80] and a too early LC alarm will cause disturbance.

Negative log-likelihood (NLL). Apart from the traditional accuracy metrics, we adopt the NLL loss of multi-class classification to measure the classification uncertainty. Uncertainty constantly decreases as the TTLC approaches 0, but different methods perform differently for uncertainty reduction. Note that the evaluation of NLL loss is irrelevant to the definition of positive classes.

Average prediction time. Prediction time represents the average *effective* prediction horizon for LCs. Generally, the prediction time is obtained by recording the time of the first successful LC prediction. However, we consider a strict strategy and only record the starting point of a series of *consistent* LC predictions, where two consecutive LC predictions should not contain a gap larger than three frames.

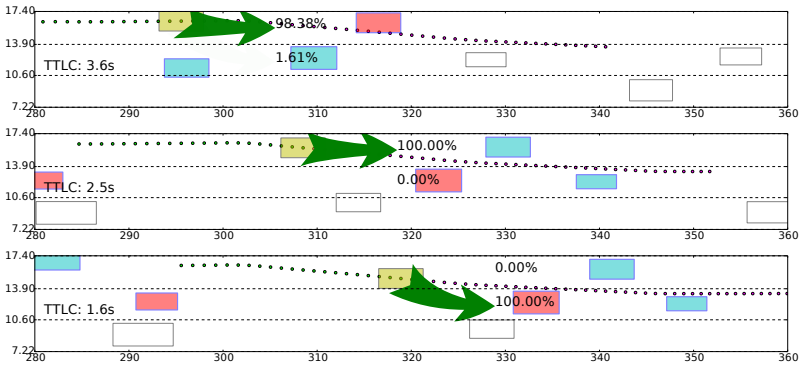
3.4.5 Analysis

Overall accuracy. In Tab. 3.3, we report the precision, recall and F1-score of all the methods. The calculation of the recall is based on the critical FNs representing the most unacceptable misclassifications. Note that we start the evaluation from an early phase, namely, 8 s before the target vehicle crosses the lane dividing line, which may introduce more FPs. Only the frames which contain enough ground truth observation and prediction are picked. There are a total of $13,338$ frames of prediction.

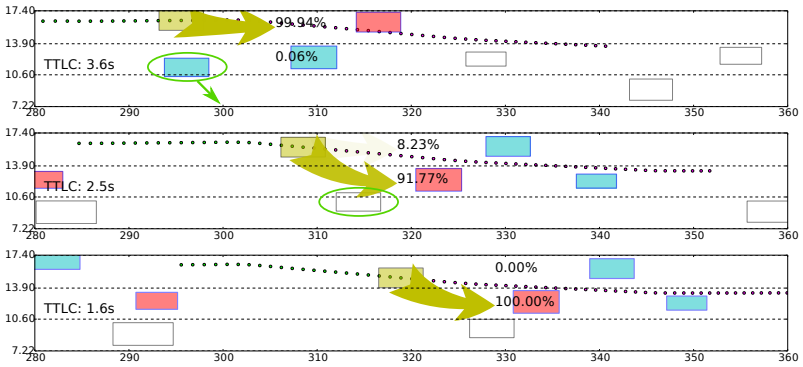
According to Tab. 3.3, the proposed VBIN has the highest precision, recall and F1-score among all the baseline methods. Note that the precision of SIMP reported in Tab. 3.8 is slightly lower than that in [57], since [57] does not include the FPs from 8 s to 4 s. Compared to CLSTM, our method achieves a higher recall of 0.967. Our conjecture is that CLSTM may underestimate

Table 3.3: A comparison of different prediction methods. The methods marked with † are interaction aware.

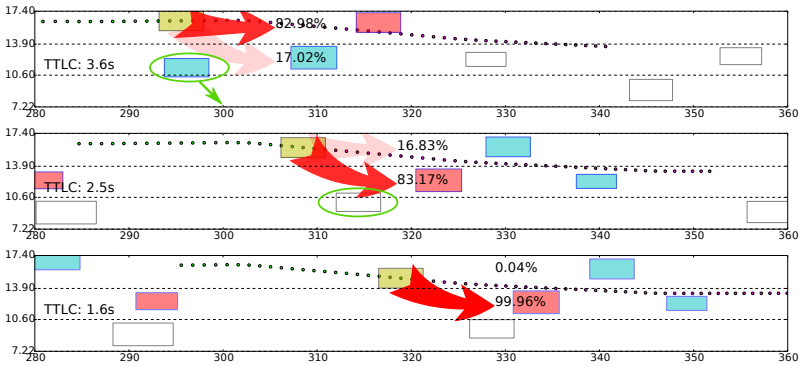
Method	Precision	Recall	F1-score	Ave. Predict Time (s)
VLSTM	0.802	0.920	0.857	1.999
SLSTM† [1]	0.890	0.900	0.895	2.355
CLSTM† [58]	0.885	0.932	0.908	2.294
SIMP† [57]	0.873	0.931	0.901	2.102
Our VBIN†	**0.922**	**0.967**	**0.944**	**2.622**



(a) Baseline: VLSTM



(b) Baseline: CLSTM



(c) Our VBIN

Figure 3.8: Illustration of the comparison on the NGSIM dataset. The vehicle marked in **yellow** is the prediction target. The vehicles filled with **cyan** and **red** are the selected neighboring vehicles. The pair of each **red** vehicle and target vehicle may have potential interaction according to the RSS safety model [3]. Note that we do not use any handcrafted models and the RSS is used to aid understanding of the scenario. The behavior predictions are marked in arrows, and the transparency represents the magnitude of the likelihood.

the likelihood of LC due to the social pooling strategy, especially when the neighboring lane is congested, which may result in more FNs.

The critical FPs and FNs are also listed in Tab. 3.4. Our proposed method has the least critical FNs, at 206, which indicates that our method preserves the accuracy of the near future. Also, our proposed method has the least critical FPs, at 322, which means that the interaction information is useful for rejecting false alarms. Compared to [13], which uses trajectory predic-

tion on an artificial potential field to reject FPs, our proposed method is completely data driven and free of hand tuning.

Table 3.4: Critical misclassifications over $13,338$ predictions.

Method	VLSTM	SLSTM†	CLSTM†	SIMP†	Our VBIN†
Critical FNs $(< 1.5s)$	435	597	400	477	**206**
Critical FPs $(> 5.5s)$	564	349	387	562	**322**

Ave. prediction time. Another concern is how early the LC can be recognized, which represents the length of the prediction horizon. Considering that the actual LC duration is about 3.0 s to 5.0 s, according to previous research [13, 80], a clear LC maneuver roughly occurs at 1.5 to 2.5 s before crossing the dividing line. The average prediction time of VLSTM is 1.999 s, which is reasonable if the prediction only relies on the maneuver history of the vehicle itself. The interaction-aware models all improve the average prediction time, which verifies the conjecture that interaction can help to predict behaviors over an extended horizon. Among all the interaction-aware models, our proposed method achieves the largest average prediction time of 2.622 s, which means that it is better at capturing behaviors in the distant future.

Accuracy with respect to TTLC. How prediction accuracy changes with respect to time is also a focus of our interest. As shown in Fig. 3.9, we measure the accuracy from two different perspectives: NLL loss and prediction distribution. The NLL loss generally represents the accuracy of the multi-class classification, and our proposed method achieves the lowest NLL loss for all different TTLCs. It can be observed that, compared to VLSTM, significant NLL loss reduction can be achieved, which confirms the usefulness of modeling interaction.

We also illustrate the distribution of the three behavior predictions at different TTLCs for *lane change right* (LCR) cases. The fractions of the three kinds of predictions of our method are stacked for each TTLC. We represent the distribution of VLSTM and CLSTM through lines in Fig. 3.9. It is shown that our method outputs the LCR prediction at the earliest phase, while outputting the least lane change left (LCL) predictions. The distribution also confirms that our proposed method has a superior prediction time.

Social Pooling vs. VBIN. In Fig. 3.8a, we provide an example of VLSTM showing the qualitative performance without modeling interaction. In Fig. 3.8b and Fig. 3.8c, we find that both CLSTM and our VBIN discover the LC intention earlier, and at a TTLC of 2.5 s, both methods predict LC with a likelihood of over 80 percent. However, at a TTLC of 3.6 s, CLSTM cannot capture the LC intention, despite LC being likely, since the target vehicle has a higher velocity

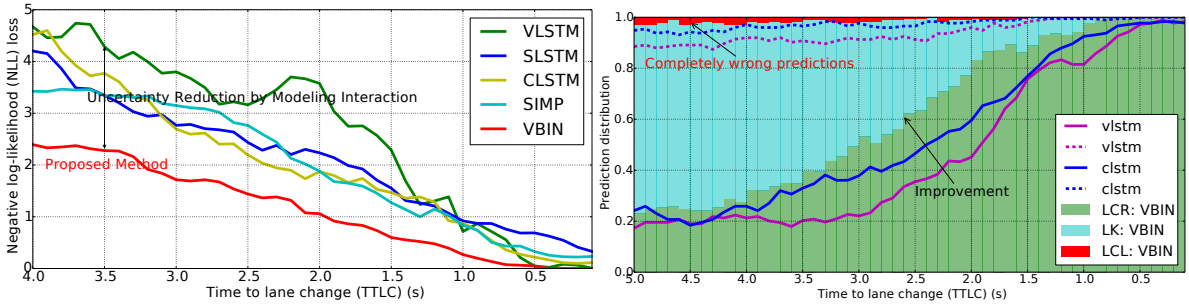


Figure 3.9: Illustration of the prediction accuracy w.r.t. the TTLC.

than the vehicle in front. Interestingly, in this case, the neighboring vehicle to the right of the target vehicle is actually conducting a rightward LC and leaving more room for the target vehicle. However, it seems that CLSTM fails to capture this information and thinks the neighboring vehicle to the right will block the target vehicle's LC, which matches our conjecture about the drawback of social pooling in Fig. 3.2. The proposed VBIN can address this problem and output a reasonable prediction.

3.5 Conclusion

In this chapter, we present a VBIN for vehicle behavior prediction. The proposed method is capable of predicting vehicle behaviors of an extended horizon by taking vehicle interaction into account. The VBIN is designed for highly dynamic on-road driving and has a novel interaction-aware network structure. It is also scalable and end-to-end trainable. We conduct extensive experiments and compare the proposed method with four baseline methods, including two state-of-the-art methods, showing significant improvements in accuracy and uncertainty reduction.

CHAPTER 4

EFFICIENT UNCERTAINTY-AWARE DECISION-MAKING FOR AUTONOMOUS VEHICLES

4.1 Scientific Background and Literature Review

4.1.1 Scientific Background

In recent years, work in the areas of multi-sensor perception, prediction, decision-making, trajectory planning and control has enabled autonomous driving in difficult environments with other traffic participants and obstacles. Reasoning about hidden intentions of other agents is the key capability to a safe and robust autonomous driving system. However, even given perfect perception, it is still challenging to make safe and efficient decisions due to the uncertain and sometimes unpredictable intentions of other agents. The situation is even worse when considering other system uncertainties such as imperfect tracking results and prediction errors.

There has been extensive literature on decision-making under uncertainty. Partially observable Markov decision process (POMDP) [81] provides a general and principled mathematical framework for planning in partially observable stochastic environments. However, due to the *curse of dimensionality*, POMDP quickly becomes computationally intractable when the problem size scales [82].

To address the computation difficulties, online POMDP planning algorithms [83] interleave the planning and execution and only reason about in the neighborhood of the current belief. Online POMDP solvers such as POMCP [84], DESPOT [85, 86] and ABT [87] have been proposed. The latest advances in the POMDP solvers are applied to many uncertainty-aware planning algorithms for autonomous vehicles [88–93]. However, despite that various simplifications and discretizations are applied, the efficiency of the existing methods is still inadequate for highly dynamic driving scenarios (see Sec. 4.1 for a detailed study).

It is essential to incorporate domain knowledge to efficiently make robust decisions. Multi-policy decision-making (MPDM) [75, 76, 94] conducts deterministic closed-loop forward simulation of a finite discrete set of semantic-level policies (e.g., lane change (LC), lane keeping

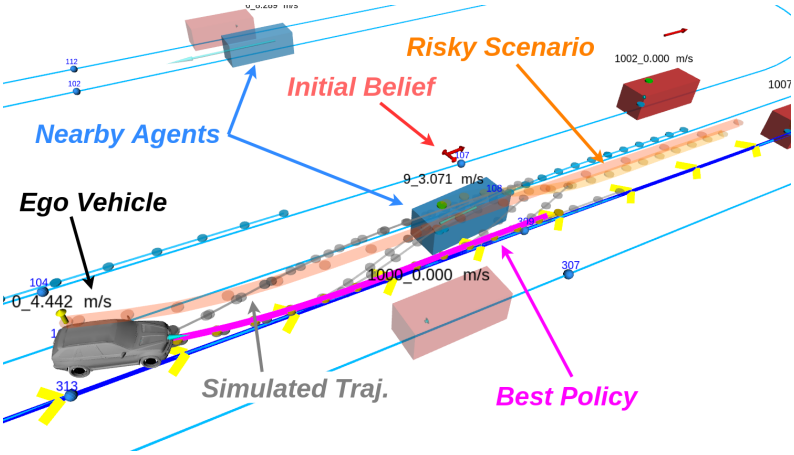


Figure 4.1: Illustration of the proposed decision-making framework. The initial belief for different intentions (i.e., LC, LK) is marked by the *red* arrows on the agent vehicles with length representing the probability. The ego vehicle generates a set of sequential semantic-level policies according to the DCP-Tree. Each behavior sequence is simulated in closed-loop (marked by *black* dots) considering nearby agents. The risky scenario (in *orange*) in which the leading vehicle inserts to the ego target lane is identified by our CFB mechanism.

(LK), etc.) for the controlled (ego) vehicle and other agents, rather than performing the evaluation for every possible control input for every vehicle. However, the semantic behaviors for all the agents are assumed to be fixed in the whole planning horizon, which may not be true in long-term decision-making. Moreover, risk can be underestimated if initial behavior prediction is inaccurate [95, 96], which may lead to unsafe decisions.

Our goal here is to control the computational complexity of the decision-making problem to enable real-time execution, while retaining sufficient flexibility and fidelity to preserve safety. In this chapter, we present an efficient uncertainty-aware decision-making (EUDM) framework. First, EUDM uses a *domain-specific closed-loop policy tree* (DCP-Tree) to construct a semantic-level action space. Each node in the policy tree is a finite-horizon semantic behavior of the ego vehicle. Each trace from the root node to the leaf node then represents a sequence of semantic actions of the ego vehicle. Each trace is evaluated in the form of closed-loop simulation similar to [75] but the ego behavior is allowed to change in the planning horizon.

The DCP-Tree essentially determines a preliminary semantic-level action sequence for the ego vehicle, however, the behaviors (intentions) of other agent vehicles remain undetermined. Since the combinations of the intentions of agent vehicles explode exponentially, it is inefficient to naively sample all possible combinations of agent intentions. To overcome this, EUDM uses the *conditional focused branching* (CFB) mechanism to pick out the potentially risky scenarios using open-loop safety assessment conditioning on the ego action sequence. EUDM is highly parallelizable, and can produce long-term (up to $8s$) lateral and longitudinal fine-grained

behavior plans in real-time (20 Hz).

The major contributions are summarized as follows:

- An efficient uncertainty-aware decision-making framework for autonomous driving.
- A real-time and *open-source*[1] implementation of the proposed decision-making framework.
- Comprehensive experiments and comparisons are presented to validate the performance, using both onboard sensing data captured by a real vehicle and an interactive multi-agent simulation platform

4.1.2 Related Works

There is extensive literature on decision-making for autonomous driving [97, 98]. Many researchers tackle the planning problem in a decoupled manner, namely, "predict and plan" [99–102]. Specifically, prediction results are fixed in one planning cycle. Several drawbacks may exist. First, it is problematic to handle interaction among agents in this decoupled design. Second, given onboard sensing, imperfect tracking may result in prediction errors, which affects the safety of the decision. Third, even given perfect perception, the prediction uncertainty still scales dramatically w.r.t. the prediction horizon [103] due to partial observability.

POMDP is a powerful tool to handle various uncertainties in the driving task using a general probabilistic framework [104]. However, due to the *curse of dimensionality*, POMDP quickly becomes computationally intractable when the problem size scales [82].

Benefit from the advance of online POMDP solvers [85–87, 105], several methods are proposed to tackle the decision-making problem by simplifying the system model and restricting the problem scale. Bai *et al.* [90] decouple the planning problem into pathfinding and velocity planning, and POMDP is only applied to the velocity planning. Hubmann *et al.* proposed POMDP-based decision-making methods for urban intersection [92] and merging [93] scenarios. However, human driving knowledge is not incorporated into the heuristic design. Meanwhile, the efficiency is still inadequate (less than 5 Hz) in fast-changing environments.

Cunninghan *et al.* [75, 94] proposed the multipolicy decision-making (MPDM) framework, which approximates the POMDP process into the closed-loop simulation of predefined semantic-level driving policies (e.g., LC, LK, etc.) for all the agents. The incorporation of

[1] `https://github.com/HKUST-Aerial-Robotics/eudm_planner`

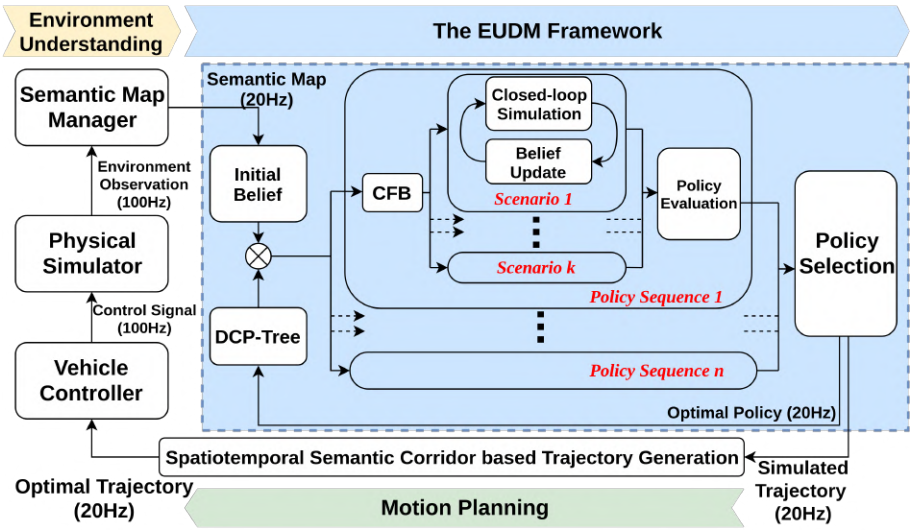


Figure 4.2: Illustration of the proposed behavior planning framework (in the *blue* box) and its relationship with other system components.

domain knowledge greatly accelerates the problem-solving. However, the ego behavior is fixed for the whole planning horizon, which may result in reactive decisions. Moreover, the hidden intentions (driving policy of other agents) are sampled according to initial behavioral prediction in the beginning and will not be updated during the simulation. As a result, risky outcomes may not be reflected in policy evaluation due to inaccurate initial behavior prediction [95].

In this chapter, we follow the idea of semantic-level closed-loop policies from MPDM. However, there are two major differences. First, the policy of the ego vehicle is allowed to change in the planning horizon according to the DCP-Tree, which makes it suitable for long-term decision-making. Second, focused branching is applied to pick out the risky scenarios, even given totally uncertain behavior prediction, which enhances the safety of the framework.

4.2 System Overview

An overview of our system is shown in Fig. 4.2, which is similar to our previous work [40]. The difference is that the focus of this chapter is the decision-making part. Different from the methods which decouple the prediction and planning module [99–102], for our method, the intentions of agents are tracked and updated in the planning horizon.

In EUDM, DCP-Tree is used to guide the branching in the action domain and update the semantic-level policy tree based on the previous best policy. Each action sequence of the ego vehicle is then scheduled to a separate thread. For each ego action sequence, the CFB mechanism is applied to pick out risky hidden intentions of nearby vehicles and achieves guided

branching in intention space. The output of the CFB process is a set of scenarios containing different hidden intentions combinations of nearby vehicles. Each scenario is then evaluated by the closed-loop simulation to account for interaction among agents in a sub-thread in parallel. All the scenarios are fed to the cost evaluation module and biased penalty is applied to risky branches. The output of the EUDM framework is the best policy which is represented by a series of discrete vehicle states (0.4 s resolution in the experiments) generated by the closed-loop forward simulation. The state sequence is fed to the motion planner to guide the trajectory generation process [40].

4.3 Decision-making via Guided Branching

4.3.1 Preliminaries on POMDP

A POMDP can be defined as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{Z}, O, \gamma \rangle$, which are the state space, action space, observation space, state-transition function, reward function, observation function, and discount factor, respectively. The state of the agent is *partially observable*, and is described as a belief b, which is a probability distribution over $\mathcal{S}$. The belief state can be updated given an action a and an observation z using Bayes' inference. The goal of the online POMDP planner is finding an optimal policy π^* that maximizes the total expected discounted reward, given an initial belief state b_0 over the planning horizon t_h. We refer interested readers to [84, 104] for more details.

The optimal policy is often pursued using a multi-step look-ahead search starting from the current belief b_0. A belief tree can be expanded using the *belief update* function after taking actions and receiving observations during the search. However, the scale of the belief tree grows exponentially ($\mathcal{O}(|\mathcal{A}|^h|\mathcal{Z}|^h)$) with respect to the tree depth h, which is computationally intractable given large action space $|\mathcal{A}|$ and observation space $|\mathcal{Z}|$. State-of-the-art online POMDP planners [85, 87, 105] use Monte-Carlo sampling to deal with the *Curse of Dimensionality* and *Curse of History* [105]. Meanwhile, generic heuristic search such as branch-and-bound [85] and reachability analysis [87] can be used to accelerate the search. Note that the focus of this chapter is to utilize domain-specific knowledge to achieve guided branching, which is also compatible with the generic heuristic search techniques.

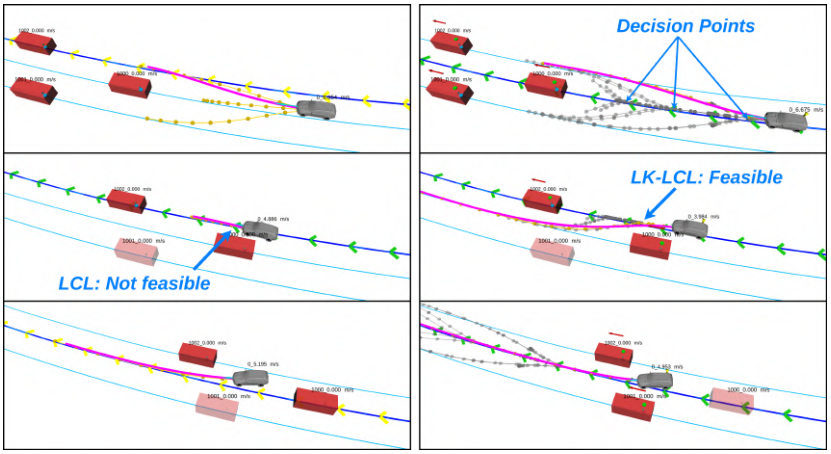


Figure 4.3: Comparison of MPDM (left) and EUDM (right). The ego vehicle (*gray*) needs to conduct an overtaking maneuver. The simulated behaviors of MPDM are fixed in the planning horizon. The ego vehicle cannot make a lane-change-left (LCL) decision until it passes the blocking vehicle, so the generated plan is local and reactive. EUDM considers the change of behavior in different future stages, which results in a consistent and farsighted plan.

4.3.2 Domain-specific Closed-loop Policy Tree

As pointed out by MPDM [76], for the decision-making problem, too much computation effort of POMDP is spent on exploring the space that is unlikely to be visited. The key feature of MPDM is using semantic-level policies instead of traditional "state"-level actions (e.g., discretized accelerations or velocities). By using semantic-level policies, the exploration of the state space is guided by simple closed-loop controllers (i.e., domain knowledge).

Motivated by MPDM, we also use semantic-level policies, as one source of the domain knowledge. However, as elaborated in Sec. 4.1, one major limitation of MPDM is that the semantic-level policy of the ego vehicle is not allowed to change in the planning horizon. For example, MPDM may simulate LC and LK policies for the whole planning horizon (e.g., 8 s). Typical patterns such as lane change in different stages (e.g., in 2 s, 4 s, 6 s) are not included in its decision space. As a result, the decision of MPDM tends to be local and may not be suitable for long-term decision-making (see Fig. 4.3 for an example).

In this chapter, DCP-Tree is utilized to generate future ego action sequences, which allows the semantic policy of the ego vehicle to change in the planning horizon. The nodes of DCP-Tree are pre-defined semantic-level actions associated with a certain time duration. The directed edges of the tree represent the execution order in time. DCP-Tree origins from an *ongoing* action $\hat{a}$, which is the executing semantic-level action from the best policy in the last planning cycle. Every time we enter a new planning episode, the DCP-Tree is rebuilt by setting $\hat{a}$ to the root

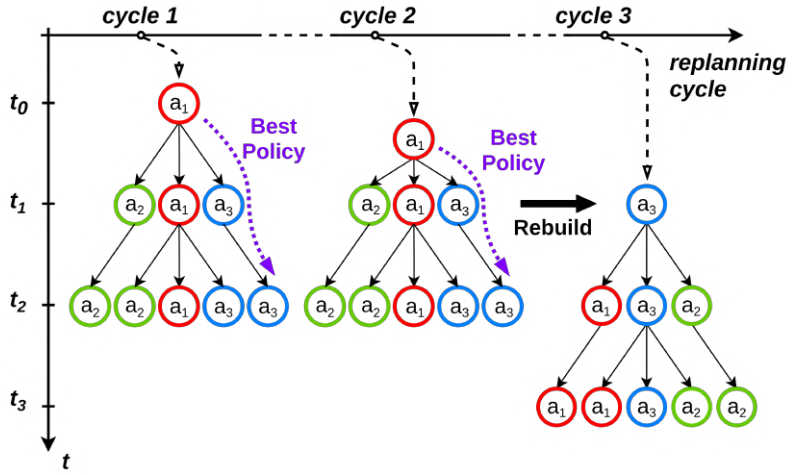


Figure 4.4: Illustration of the proposed DCP-Tree and the rebuilding process after its *ongoing* action changes. Suppose there are three discrete semantic-level actions $\{a_1, a_2, a_3\}$ and the height of the tree is three. The *ongoing* action for the left and middle tree is a_1, and the best policy is the dashed *purple* trace. After executing a_1, the *ongoing* action switches to a_3, and the DCP-Tree is updated to the right tree. Each trace only contains one change of action.

Algorithm 5: Process of EUDM

Inputs: Current states of ego and other vehicles s; *Ongoing action* $\hat{a}$; Pre-defined semantic action set $\mathcal{A}$; Planning horizon t_h;
$\mathfrak{R} \leftarrow \varnothing$; \\ set of rewards for each policy;
$\Psi \leftarrow \texttt{UpdateDCPTree}(\mathcal{A}, \hat{\texttt{a}})$; \\ DCP-Tree Ψ;
$\hat{\Pi} \leftarrow \texttt{ExtractPolicySequences}(\Psi)$;
foreach $\pi \in \hat{\Pi}$ **do**
  $\Gamma^{\pi} \leftarrow \varnothing$; \\ set of simulated trajectories;
  $\Omega \leftarrow \texttt{CFB}(s, \pi)$; \\ set of critical scenarios;
  foreach $\omega \in \Omega$ **do**
    $\Gamma^{\pi} \leftarrow \Gamma^{\pi} \cup \texttt{SimulateForward}(\omega, \pi, t_h)$;
  end
  $\mathfrak{R} \leftarrow \mathfrak{R} \cup \texttt{EvaluatePolicy}(\pi, \Gamma^{\pi})$;
end
$\pi^*, \hat{a} \leftarrow \texttt{SelectPolicy}(\mathfrak{R})$;

node.

Since the ego policy is allowed to change in the planning horizon, the challenge is that the number of possible policy sequences scales exponentially w.r.t. the depth of the tree (i.e., the planning horizon). To overcome this, DCP-Tree is expanded by a pre-defined strategy, which comes from the observation that, for human drivers, typically we do not frequently change the driving policy back and forth in a single decision cycle. For example, human drivers often evaluate whether it is feasible to conduct one policy change, e.g., switching from LK to LC in several seconds. This does not prevent human drivers from conducting complex maneuvers

since consecutive decisions from different decision cycles can be combined. Motivated by this, from the *ongoing* action, each policy sequence will contain at most one change of action in one planning cycle, as shown in Fig. 4.4, while the back-and-forth behavior is achieved by replanning.

For instance, suppose the ongoing action is LK, the resulting policy sequences may include (LK-LC-LC-LC...), (LK-LK-LC-LC...) and (LK-LK-LK-LC...), etc. Note that the size of leaf nodes in DCP-Tree is $\mathcal{O}[(|A|-1)(h-2)+|A|], \forall h > 1$, which grows linearly with respect to the tree height h. It is also notable that MPDM is only one branch of our DCP-Tree and DCP-Tree includes multiple future decision points. Compared to MPDM, DCP-Tree has much larger decision space resulting in more flexible maneuvers as shown in Fig. 4.3. Moreover, we also observe a significant improvement of decision consistency among consecutive planning cycles compared to MPDM.

4.3.3 Conditional Focused Branching

Essentially, DCP-Tree provides a guided branching mechanism in the action space. The remaining problem is to determine the semantic-level intentions of the nearby vehicles, namely, the branching in the intention space. The challenge here is that the combination of intentions of nearby vehicles scales exponentially w.r.t. the number of agents. In the case of MPDM, the intention of the nearby vehicles is fixed for the whole planning horizon, and the initial intention is sampled according to a behavior prediction algorithm. The limitation of MPDM is that, with a limited number of samples, influential risky outcomes may not be rolled out, especially when the initial intention prediction is inaccurate [95].

To overcome this, we propose the CFB mechanism. The goal here is to find the intentions of nearby vehicles which potentially lead to risky outcomes with as few branches as possible. The term "conditional" means conditioning on the ego policy sequence. The motivation comes from the observation that the attention of the human driver for nearby vehicles is biased differently when intending to conduct different maneuvers. For example, a driver will pay much more attention to the situation on the left lane rather than the right one when he intends to make an LCL. As a result, by conditioning on the ego policy sequence, we can pick out the a set of relevant vehicles w.r.t. the ego future actions. The selection process is currently based on ruled-based expert knowledge as detailed in Sec. 4.4. We point out that learning-based attention mechanisms can also be incorporated and we leave it as an important future work.

By conditioning on the ego policy sequence, we obtain a subset of vehicles that need to be

further examined. Instead of enumerating all the possible intentions for this subset of vehicles, we introduce a preliminary safety check to pick out the vehicles which we should pay special attention to. The preliminary safety assessment is conducted using open-loop forward simulation based on *multiple hypotheses*. For example, for the vehicle whose intention is uncertain, we anticipate what the situation will be if the vehicle is LC or LK, respectively. The anticipation is carried out using open-loop forward simulation under the intention hypothesis. The idea of using open-loop simulation is that by ignoring the interactions among agents, we check how the serious the situation will be if surrounding agents are completely uncooperative and does not react to the other agents. For the vehicles which do not pass the preliminary safety assessment, different scenarios are further examined in closed-loop forward simulation. And for the vehicles which pass the assessment, we use maximum a posteriori (MAP) from initial belief. As a result, the branching in intention space is guided to potentially risky scenarios.

The flow of EUDM is described in Algo.5. Evaluation for each policy sequence can be carried out in parallel (Line 5 to 11). Each critical scenario selected by CFB is examined by closed-loop forward simulation (Line 8 to 10) in parallel. Each policy is evaluated (Line 11) using the reward function detailed in Sec. 4.4 and the best policy is elected (Line 13).

4.4 Implementation Details

4.4.1 Semantic-level Actions

We consider both lateral and longitudinal actions to ensure the diversity of the driving policy. Similar to [94], we define the lateral actions as $\{$*LK, LCL, LCR*$\}$. For longitudinal action, we use $\{$*accelerate, maintain speed, decelerate*$\}$. Although semantic actions are discrete as shown in the DCP-tree, we can compute continuous control signals based on the predefined controllers for each semantic action. Specifically, for longitudinal control, we adopt intelligent driver model (IDM) as the predefined controller. We first find the leading vehicle for the simulated vehicle. If there is no leading vehicle, we place a virtual leading vehicle at a far distance. For vehicle c_r, x_r denotes its position at time t, and v_r denotes its velocity. Furthermore, l_r gives the length of the vehicle. To simplify notation, we define the net distance $d_r \coloneqq x_f - x_r - l_f$, where f refers to the vehicle directly in front of vehicle r, and the velocity difference, or approaching rate, $\Delta v_r \coloneqq v_r - v_f$. For a simplified version of the model, the dynamics of vehicle c_r are then

described by the following ordinary differential equation:

$$\dot{v}_r = a\left(1 - \left(\frac{v_r}{v_0}\right)^{\delta} - \left(\frac{d^*(v_r - \Delta v_r)}{d_r}\right)^2\right), \tag{4.1}$$

with $d^*(v_r - \Delta v_r) = d_0 + v_r T + \frac{v_r \Delta v_r}{2\sqrt{a_+ a_-}}$, where v_0, d_0, T, a_+, and a_- are model parameters which have the following meaning:

- Desired velocity v_0: the velocity the vehicle c_r would drive at in free traffic.
- Minimum spacing d_0: a minimum desired net distance. A car can't move if the distance from the car in the front is not at least d_0.
- Desired time headway T: the minimum possible time to the vehicle in front.
- Acceleration a_+: the maximum vehicle acceleration.
- Comfortable braking deceleration a_-: a positive number.

The exponent δ is usually set to 4. For different longitudinal actions {*accelerate, maintain speed, decelerate*}, we choose different v_0 based on the current speed. This is how discrete semantic actions are transformed to continuous control signals. We integrate the system described by Eq. 4.1 for one step and obtain the desired velocity $v(t)$ for this step, and use $v(t)$ as the control signal.

For lateral semantic actions {*LK, LCL, LCR*}, we adopt a simple pure pursuit controller with the control law given by,

$$\delta(t) = \tan^{-1}\left(\frac{2L\sin\Delta\theta(t)}{l_d}\right), \tag{4.2}$$

where $\delta(t)$ denotes the steer angle at time t, L denotes the wheel base length of the vehicle, $\Delta\theta(t)$ denotes the angle difference between the current vehicle angle and the tangent angle at the reference path with a look ahead distance of l_d. {*LK, LCL, LCR*} represent different reference centerlines for the pure pursuit controller.

Each semantic-level action is associated with time duration of 2 s, while the closed-loop simulation is carried out with 0.4 s resolution to preserve the simulation fidelity. Note the duration of the ongoing action is deducted by the replanning resolution (0.05 s) for each planning cycle.

4.4.2 Forward Simulation

The goal of the closed-loop simulation is to push the state of the multi-agent system forward while considering the potential interaction. The simulation model should achieve a good balance between simulation fidelity and inference efficiency. As we introduced in the implementation of semantic actions, we adopt the *intelligent driving model* [106] and *pure pursuit controller* [107] as the longitudinal and lateral simulation models, respectively. Based on these two predefined controllers, we can transform discrete semantic action to continuous control signals. Based on the control signals, namely, $v(t)$ for longitudinal control, and $\delta(t)$ for lateral control, we can use the vehicle model (i.e., kinematic bicycle model) to forward simulate the vehicle state using the differential equation as follows,

$$\begin{aligned} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= v\tan(\delta)/L, \end{aligned} \tag{4.3}$$

where $v(t)$ and $\delta(t)$ are used to calculate a constant steer rate and a constant acceleration for this micro step. Integrating the system using Eq. 4.3, we can obtain the next state for each vehicle after this micro step. It is notable that, for each simulation step, the leading vehicle for each simulated vehicle is updated before all the computation.

4.4.3 Belief Update

The hidden intentions considered in this work include lateral behaviors, such as {*LK, LCL, LCR*}. The belief over these intentions of agent vehicles are updated during the forward simulation as shown in Fig. 4.2. In this work, we adopt a rule-based lightweight belief tracking module that takes a set of features and metrics including velocity difference, distance w.r.t. the leading and following agents on the current and neighboring lanes and lane-changing model [4] as input. The belief tracker generates a probability distribution over the intentions (i.e., LK, LCL, LCR). The probability serves as the weight for each scenario and each frame during policy evaluation. In the experiments, we find the rule-based belief tracker works well despite its simple structure.

Mathematically, the belief tracker first computes the gain for LK, LCL and LCR, respectively and then normalizes the gains as a probabilistic representation. For the each behavior, the belief tracker uses the states of the predicted vehicle and its following and leading vehicle on the *target* lane to compute its gain according to MOBIL model [4].

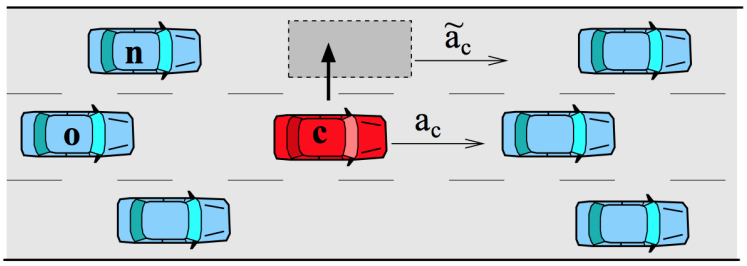


Figure 4.5: An illustrative example from MOBIL model [4] .

Take the gain of LCL as an example, MOBIL model firsts computes the IDM acceleration $\tilde{a}_c$ if the predicted vehicle (*red*) is placed at the left lane, the IDM acceleration a_c if the predicted vehicle keeps driving on its own lane, and also the IDM acceleration of the following vehicle at the nearby lane $\tilde{a}_n$ if the predicted vehicle inserts and a_n if the predicted vehicle does not insert. By comparing $\tilde{a}_n$ with $\tilde{a}_n$, we can evaluate the disturbance caused by the lane change of the predicted vehicle to the following vehicle n at the left lane. The gain of LCL for the predicted vehicle is weighted by the acceleration gain for the predicted vehicle c itself $\tilde{a}_c - a_c$ and the disturbance (which reflects the politeness of the lane change) to the following vehicle on the nearby lane $\tilde{a}_n - a_n$. In the same way, we can compute the gain for LCR. The gain for LK is directly measure by a_c. We refer the interested readers to [4] for more detailed description. After quantitatively computing the gains for LCL, LCR, LK, a softmax operation is enforced to normalize the gains as a probability measure. The belief tracker based on MOBIL can model the interaction between the predicted vehicle with its surrounding vehicles to some extent and can serve as a learning-free lightweight alternative to our learning-based method in Chap. 3.

4.4.4 CFB Mechanism

The first step of CFB is the key vehicle selection. For the current lane and neighboring lanes, we search forward and backward along the lanes for a certain distance w.r.t. the current speed of ego vehicle and the vehicles in this range are marked as key vehicles. The second step is uncertain vehicle selection according to the initial belief. Specifically, we pick out the vehicles, whose probabilities for the three intentions are close to each other, as uncertain vehicles. Note that for the vehicles with confident prediction, we select the MAP intention and marginalize the intention probabilities using the MAP selection result. The third step is using the open-loop forward simulation for safety assessment. For the vehicles which fail the assessment, we enumerate all the possible combinations of their intentions. Each combination becomes a CFB-selected scenario and the probability of scenario is calculated. The fourth step is picking out top

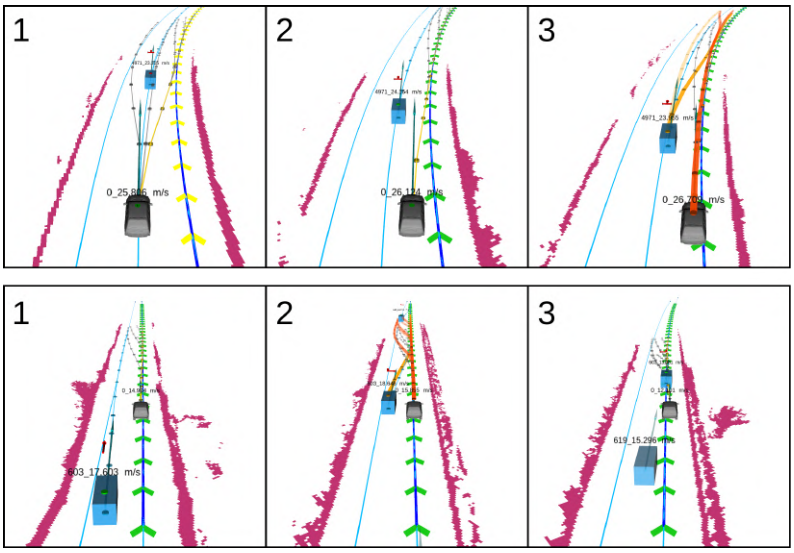


Figure 4.6: Open-loop test using onboard sensing data. Up: The ego vehicle is approaching another vehicle and making a LC decision (1 and 2). After the ego vehicle finishing LCR, EUDM detects several risky scenarios due to the uncertain intention of the front vehicle. Bottom: Another vehicle overtakes the ego vehicle and merges into our lane aggressively (1 and 2). EUDM captures the risky situation and takes a proper deceleration policy (3).

k scenarios according to user-preference, and we further marginalize the probabilities among the top-k scenarios. The marginal probabilities become the weights of CFB-selected scenarios during evaluation.

4.4.5 Policy Evaluation

The overall reward for a policy sequence is calculated by the weighted summation of the reward for each CFB-selected scenario. The reward function consists of a linear combination of multiple user-defined metrics including efficiency, safety and navigation. The design of reward function is essential for behavior planning. Inappropriate cost functions may result in inconsistent behaviors and poor generalization ability. Through extensive experiments, we find that the key to robust cost functions is to unify the units for different cost terms. Otherwise, it is impossible to understand the properties of the cost function and systematically tune the weights. Based on this observation, we design the cost function for safety, efficiency and navigation as follows.

Safety cost The safety cost we use in this chapter is derived based on responsibility-sensitive safety [3]. Given the state of the ego vehicle and the state of other agent vehicle, we first project both states onto the frénet frame of the reference lane. Depending on the different relative positions of the ego vehicle and the other agent, the calculation of safety distance is different.

Take longitudinal safety as an example, a longitudinal distance between a car c_r that drives behind another car c_f , where both cars are driving at the same direction, is safe w.r.t. a response time ρ if for any braking of at most $a_{\text{max,brake}}$, performed by c_f, if c_r will accelerate by at most $a_{\text{max,accel}}$ during the response time, and from there on will brake by at least $a_{\text{min,brake}}$, until a full stop then it won't collide with c_f. Then, the minimal safe longitudinal distance between the front-most point of c_r and the rear-most point of c_f is:

$$d_{\min} = \left[v_r \rho + \frac{1}{2} a_{\text{max,accel}} \rho^2 + \frac{\left(v_r + \rho a_{\text{max,accel}}\right)^2}{2 a_{\text{min,brake}}} - \frac{v_f^2}{2 a_{\text{max,brake}}} \right]_+ . \tag{4.4}$$

Instead of directly using safety distance which is with unit m inside the cost function, we transform the above calculation to a safety velocity (m/s) to unify the unit with the other cost terms. Mathematically, let $d_{\min}$ in Eq. 4.4 be equal to the current net distance between c_r and c_f, we can calculate the maximum velocity $v_{\max}$ for c_r to avoid the bump-to-rear collision. And the longitudinal safety cost $f^r_{\text{safety,lon}}$ for the vehicle c_r is based on the violation of the maximum velocity $v_{\max}$ given by $f^r_{\text{safety,lon}} = \max(0.0, v_r - v_{\max})$. Similarly, we can also calculate the lateral safety cost $f^r_{\text{safety,lat}}$ based on the RSS and we refer the interested readers to [108] for more details. Here we demonstration the computation of safety cost if the ego vehicle drives behind other vehicles. The source of the safety cost comes from the responsibility for a bump-to-rear collision. However, the computation is different if the relative position of ego vehicle and other vehicle is different. If the ego vehicle keeps its lane, there should be no safety cost between the ego vehicle with its rear vehicle if the ego vehicle does not exceed the maximum deceleration $a_{\text{max,brake}}$. The reason is that even there is a bump-to-rear collision between the ego vehicle with its rear vehicle, the ego vehicle is not responsible for the collision. We can observe that our safety cost is not designed to avoid all the collisions, but designed to avoid the responsibility for collision. This is essential for a multi-agent system (without vehicle-to-vehicle communication) since absolute safety can never achieved in this kind of system.

After taking the summation over all the safety costs between the ego vehicle and all the other agent vehicles, we obtain the safety cost for a particular frame.

Efficiency cost The efficiency cost is measured by the ego vehicle current velocity, the ego desired velocity, the leading vehicle velocity and relative distance between the ego vehicle and the leading vehicle. Mathematically, the efficiency cost for vehicle c_r is given by

$$f^r_{\text{efficiency}} = w_1 (v_0 - v_r) \mathbb{1}_{v_r < v_0} + w_2 l_r (v_0 - v_f) \mathbb{1}_{v_f < v_0}, \tag{4.5}$$

where v_0 denotes the desired velocity for vehicle c_r, l_r denotes a normalized (between 0-1)

scalar for the distance between ego vehicle c_r and its leading vehicle c_f, and v_r, v_f denote the current speed for c_r and c_f, respectively. w_1 and w_2 are two scalar weights. The efficiency cost represent whether the ego vehicle is blocked by its leading vehicle which results in unsatisfactory speed compared to its desired speed.

Navigation cost The navigation cost represents the navigation preference given by the user. For example, if the user specifies a certain destination, after route planning, each lane can be assigned a preference cost depending on how it contributes to the desired route. The design of navigation cost is simple in this chapter: $f^r_{\text{navi}} = w_3 v_0 + w_4 |v_0 - v_r|$, where w_3 represents the unit cost for a certain lane which can be specified by the scoring system of the route planner, and w_4 denotes the unit cost if the ego vehicle velocity does not match the user preference.

In summary, the safety, efficiency, and navigation costs share the same unit m/s for a particular frame. Multiplying the weighted summation of these cost terms by the frame duration (0.4 s in experiments) and weight decay parameter γ ($\gamma = 0.7$ in experiments) and summing over all the frames, we obtain the total cost for a particular policy sequence. The best policy sequence can then be picked out.

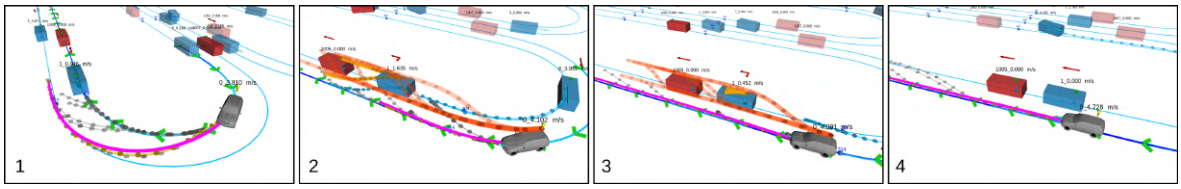
4.4.6 Trajectory Generation

The output of our behavior planner is a series of discrete states of the ego vehicle with 0.4 s resolution. The behavior plan is fed to the motion planner proposed in our previous work [40], which utilizes a spatio-temporal corridor structure to generate safe and dynamically feasible trajectories.

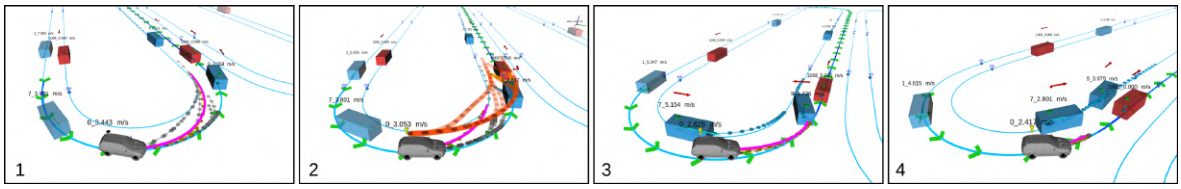
4.5 Experimental Results

4.5.1 Simulation Platform and Environment

The experiment is conducted in an interactive multi-agent simulation platform as introduced in Sect.4.2. All agents can interact with each other without knowing the driving model of other vehicles. The forward simulation model used in the ego vehicle may differ from the actual model running on the agents. The proposed decision-making method is implemented in C++11. All the experiments are conducted on a desktop computer equipped with an Intel i7-8700K CPU, and our proposed method can run stably at 20 Hz.



(a) Overtake the leading vehicle. First, our vehicle generates an overtaking action and tries to LCL, since it confidently believes the leading vehicle will LK (1). Second, the leading vehicle accelerates as the ego vehicle approaching the zone, making it uncertain that whether the leading vehicle will insert before the ego vehicle. This risky scenario is identified by CFB and the ego vehicle plan to decelerate according to the closed-loop simulation (2). Third, the leading vehicle decelerates. Although the previous risky scenario still does not pass the preliminary safety assessment using open-loop simulation, the closed-loop simulation succeeds under the overtaking action by considering interactions. After evaluation, the ego vehicle decides to accelerate and pass.



(b) Give up overtaking and give way for other vehicles. First, our vehicle decides to LCL and overtake (1). Second, the leading vehicle accelerates, upon the assessment of potential risk, the ego vehicle decides to decelerate to follow the leading vehicle while conducting the LCL (2). Third, the leading vehicle conducts LCL and the following vehicle tries to overtake us (3). The ego vehicle then decides to give up LCL and yield (3 and 4).

Figure 4.7: Illustration of different decision-making results in a conflict zone.

4.5.2 Qualitative Results

To verify that our EUDM can generate flexible and consistent behaviors in highly interactive scenarios, we show several cases using both real-world data and simulation.

Overtaking and Yielding in a Conflict Zone

Conflict zones are common in urban driving scenarios. As shown in Fig.4.7, the ego vehicle has to pass through a conflict zone where there is a leading vehicle also trying to LC and pass through the zone. Moreover, the initial belief of the leading vehicle is uncertain given the current observation. As shown in Fig.4.7, the EUDM framework can automatically select appropriate behaviors (i.e., overtaking or yielding) depending on the situation.

Table 4.1: Comparison of different decision-making approaches.

Map	Method	Safety CF	Efficiency Ave. Vel. (m/s)	Comfort ($\#/km$) UD	LCC
Double Merge	MPDM	0.048	4.9	1.85	4.63
	EDM	0.043	**5.3**	2.50	3.21
	EUDM	**0.025**	4.9	**0.38**	**1.53**
Ring	MPDM	0.042	13.36	1.09	0.0
	EDM	0.030	**14.37**	1.41	0.0
	EUDM	**0.003**	12.86	**0.48**	0.0

Testing using Real-world Onboard Sensing Data

In this case, our method is tested in an open-loop manner by using the data collected by a real autonomous driving vehicle. The goal is to verify whether the proposed method can capture risky scenarios and work under uncertain predictions and noisy perception. As shown in Fig. 4.6, EUDM can make appropriate decisions to overtake or decelerate depending on the situation.

4.5.3 Quantitative Results

We conduct a comprehensive quantitative comparison with the MPDM [94], which is one of the state-of-the-art decision-making methods for autonomous driving. We evaluate the two methods using two benchmark tracks, i.e., *Double Merge* and *Ring*. Detailed experiments can be found in the attached video, while the statistical results are shown in Table. 4.1.

Metrics

We introduce three major metrics to evaluate the performance of two methods, namely, safety, efficiency, and comfort. For safety, we count the fraction of frames that the distance between ego vehicle and other agents smaller than a threshold (i.e., safety distance). The efficiency is represented by the average velocity of the ego vehicle. The comfort is described by the number of *uncomfortable deceleration* (UD) and the number of *large curvature changing* (LCC) per kilometers. The threshold of UD and LCC is set to 1.6 m/s^2 and 0.12 $(s \cdot m)^{-1}$, respectively.

Benchmarking

We also conduct an ablative study by removing the CFB mechanism from EUDM, which results in the EDM method as shown in Table. 4.1. The goal is to verify whether the CFB can improve

the robustness and safety of the framework. As shown in Table. 4.1, EUDM makes safer decisions than EDM and MPDM according to the safety metric. The reason is that EUDM explicitly explores the risky scenarios and conducts biased branching. Note that the double merge map is a dense interactive scenario where unsafe situations are hard to be completely avoided due to aggressive behaviors of agent vehicles. For efficiency, our method and MPDM perform similarly, while EDM has a higher average velocity. It is because EDM enlarges the action space compared to MPDM, and it may take over-aggressive risky actions (see Fig.4.3). In terms of comfort, EUDM can generate much smoother behaviors than the other two baselines, since it takes conservative policy under risky scenarios beforehand and avoids hard brakes.

4.6 Conclusion

In this chapter, we proposed the EUDM framework for automated driving in dense interactive scenarios, by introducing two novel techniques, namely, the DCP-Tree and CFB mechanism. The complete framework is open-sourced and comprehensive evaluations are conducted using both real-world data and simulation.

CHAPTER 5

SPATIAL-TEMPORAL MOTION PLANNING FOR AUTONOMOUS VEHICLES

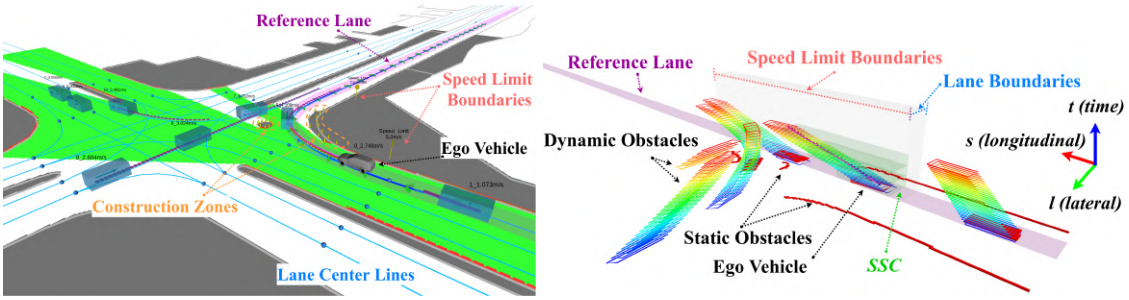
5.1 Scientific Background and Literature Review

5.1.1 Scientific Background

Trajectory generation for autonomous vehicles (AVs) in complex urban environments is challenging since there are many semantic elements (e.g., dynamic agents, traffic lights, speed limits, stop signs, and lane geometry). Different types of semantic elements may have different mathematical descriptions such as obstacle, constraint and cost [109]. It is non-trivial to tune the effects from different combinations of semantic elements so that the formulation can generalize well to all combinations of semantic elements [110]. Therefore, it is essential to describe diverse kinds of semantic elements in a unified way such that the type and combination of semantic elements do not affect the planning performance.

Apart from the representation issue of the semantic elements, another issue is how to *guarantee* the safety and feasibility of the generated trajectory. Most existing optimization-based [111, 112] and lattice-based [100, 113, 114] motion planners try to check or enforce constraints at a series of sampled points. However, these methods may fail to detect or resolve infeasible points between two sample points, and thus cannot provide safety guarantee for the entire trajectory.

To overcome the above challenges, we propose a unified trajectory generation framework with a theoretical safety and feasibility guarantee. The key to the framework is a novel spatio-temporal semantic corridor (SSC) structure. The SSC is motivated by the fact that most semantic elements can be either rendered as spatio-temporal obstacles or constraints within a certain range of the spatio-temporal domain. The key feature of the SSC is its abstraction for different types of semantic elements. Essentially, the SSC consists of a series of mutually connected collision-free cubes with dynamical constraints posed by the semantic elements. We propose an SSC generation process to generate and split the cubes so that the dynamical constraints can be correctly associated.



(a) Complex urban driving environments (b) Spatio-temporal semantic corridor (SSC)

Figure 5.1: Illustration of our trajectory generation framework. Complex semantic elements of the environment are projected to the spatio-temporal domain w.r.t. the reference lane. The SSC encodes the requirements given by the semantic elements and a safe trajectory is generated accordingly. Note that the visualization of the static obstacles is clipped to show the details of other components. More examples can be found in the video `https://www.youtube.com/watch?v=LrGmKaM3Iqc`.

Given the unified SSC representation, the trajectory generation problem boils down to generating the optimal trajectory within the SSC while satisfying the dynamical constraints. In this chapter, we contribute a quadratic programming (QP) formulation which guarantees the safety and feasibility of the generated trajectory by using piecewise Bézier curve parameterization. The proposed formulation is built on the top of the convex hull and hodograph properties of the Bézier curve. The contributions are summarized as follows:

- An SSC structure which provides a unified representation for diverse kinds of semantic elements in complex urban environments.
- An optimization-based trajectory generation formulation which ensures safety and feasibility for the entire generated trajectory.
- A complete and open-source trajectory generation framework and real-time implementation in a multi-agent urban simulation platform. Comprehensive experiments and comparisons are presented to validate the performance.

5.1.2 Spatio-temporal motion planning for AVs

There is extensive literature on spatio-temporal motion planning for autonomous vehicles. Ziegler *et al.* [113] sample a spatio-temporal state lattice on a space-time manifold [115] and use a search-based approach to obtain an executable trajectory. McNaughton *et al.* [100] adopt a spatio-temporal state lattice, which can automatically conform to the lane geometry by numerical optimization. Due to an unacceptable blowup in the size of the search space (i.e., the *curse of dimensionality*), GPU-accelerated dynamic programming is adopted in [100]. However, the

semantic elements in urban environments (such as speed limits, traffic lights, etc.) are not modeled in [100, 113, 115].

There are several approaches that attempt to model the semantic elements. Wolf *et al.* [116] associate semantic elements with specially designed cost functions and aggregate the cost terms as a potential field. However, this approach suffers from local minimums. Moreover, it is non-trivial to correctly balance the effects of different cost terms for different configurations of the semantic elements [110]. Hubmann *et al.* [117] render traffic lights and dynamic agents as obstacles in the longitudinal and time domain and apply a search-based method to obtain a generic driving strategy (i.e., a speed plan). Ajanovic *et al.* [109] extend the obstacle representation and render forbidden lane changes and solid lines as obstacles and speed limits as velocity constraints.

Built on top of the obstacle-like and constraint-like representations in [109], we further propose the SSC structure to generally represent different types of semantic elements. The key feature of the SSC is that it provides a level of abstraction which encodes all the information needed for later optimization. Adding a new semantic element or combining different semantic elements does not affect the cost formularization and constraint specification, which renders a unified and generalizable trajectory generation framework.

5.1.3 Corridor generation for AVs

The spatial corridor (i.e., convex free-space) is widely applied in trajectory generation. Zhu *et al.* [118] propose a convex elastic smoothing algorithm, which can generate a collision-free "tube" around the initial path and formulate the trajectory smoothing problem into a quadratically constrained quadratic programming (QCQP). Erlien *et al.* [119] consider not only spatial information but also vehicle dynamics to construct the convex tube. Both of these works, however, generate the corridor in a static environment and cannot deal with dynamic obstacles. Liu *et al.* [120] find a convex feasible set around the reference trajectory and leverage the convex feasible set to accelerate the non-convex optimization. However, the computation complexity is still prohibitively high for real-time applications. Moreover, collision-avoidance is their major concern and semantic elements are not considered. We are motivated by these corridor generation methods and further extend the spatial corridor to the spatio-temporal domain to cope with dynamic obstacles. Additionally, the proposed SSC can take various kinds of semantic elements into account.

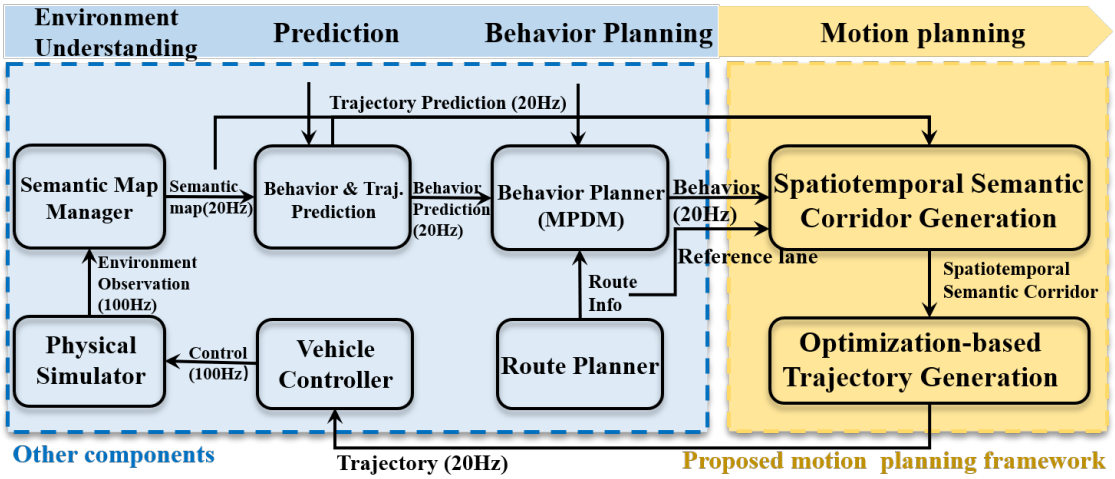
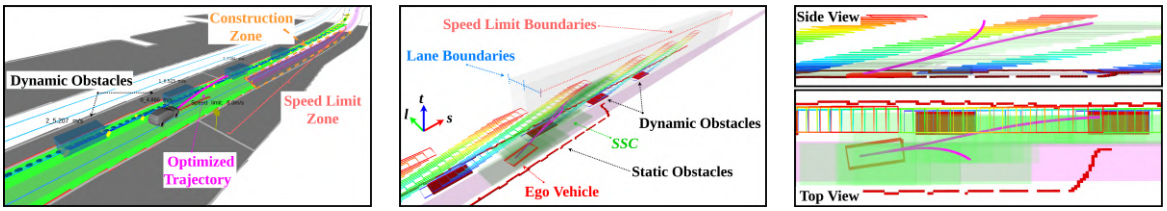


Figure 5.2: Illustration of the proposed trajectory generation framework and its relationship with other system components.



(a) Merging into congested traffic (b) Spatio-temporal representation (c) SSC and optimization

Figure 5.3: Illustration of merging into congested traffic under a speed limit. For the two potential behaviors, i.e., lane change and lane keeping, the optimal trajectories are generated inside the SSC for each behavior.

5.2 System Overview

The proposed trajectory generation framework (Fig. 5.2) belongs to the motion planning layer of an autonomous vehicle, and it requires necessary inputs from the upper layers, e.g., the behavioral layer. Apart from the proposed trajectory generation, the other system components are also illustrated to clarify the input and output of our framework.

As depicted in Fig. 5.2, there are four phases for a single planning cycle. The first phase is the environment understanding obtained by a semantic map manager which takes the responsibility to manage the semantic elements (e.g., occupancy grid map, dynamic agents, lanes, traffic rules, etc.) for local planning purposes. The second phase is the prediction, which not only provides high-level behavior anticipations (e.g., lane change, lane keeping, etc.) but also predicted trajectories for other dynamic agents. The third phase is the behavior planning, which is implemented using the multi-policy decision making (MPDM) method, as elaborated in Sect. 5.3. The fourth phase is our proposed motion planning, which takes discretized future simulated states from the behavior planner as seeds for corridor generation. Note that our trajectory gen-

eration framework can also work with other behavior planners, such as those in [117, 121, 122], as long as the behavior planner provides a preliminary initial guess about the future states.

To construct the SSC, four ingredients are needed, namely, a semantic map which consists of the semantic elements, predicted trajectories for dynamical agents, forward simulated states, and a reference lane given by the route information. Note that the trajectory prediction module may be optional if the forward simulated states already include the states for other agents such as the case of MPDM. In such case, we can use the simulated states of other vehicles as the predicted trajectories, which facilitates passing interaction anticipations from behavior planning to motion planning layer. However, since this is not a common feature in behavior planning, we still use the predicted trajectories from the trajectory prediction module in the experiments for generality, which may lose the interaction information from behavior planning. To summarize, the source of the seeds and the modeling of interaction depend on the choice of behavior planner.

5.3 Preliminaries on Multi-policy Decision Making

In this chapter, we adopt MPDM [76] as the behavioral layer. Recall that our trajectory generation method can also work with other behavior planning methods [117, 121, 122]. Since behavior planning is out of the scope of this chapter, only preliminary information about MPDM is provided here.

The MPDM model formulates the behavior planning problem as a general multi-agent partially observable Markov decision process (POMDP) to model the interaction and uncertainty in dynamic environments. Since solving the POMDP quickly becomes computationally intractable when the number of vehicles increases, MPDM relaxes the problem and assumes that both our vehicle and the other agents are executing a finite set of closed-loop discrete policies (e.g., lane change, lane keeping, etc.). Moreover, for each closed-loop policy, the future situation is anticipated via forward simulating all the vehicle states using a simplified simulation model, such as an idealized steering and speed controller. A comprehensive reward function is designed to assess the future situation and the best behavior is elected.

In this chapter, we use the forward simulated states of the ego vehicle as the seeds in the corridor generation process. Although the initial seeds are collision-free, they can not be directly executed by the vehicle due to a coarse resolution ($0.15\ s$ in the experiments) and a simplified simulation model (e.g., piecewise linear control in the experiments).

Since MPDM provides the forward simulated states for multiple behaviors (e.g., lane change

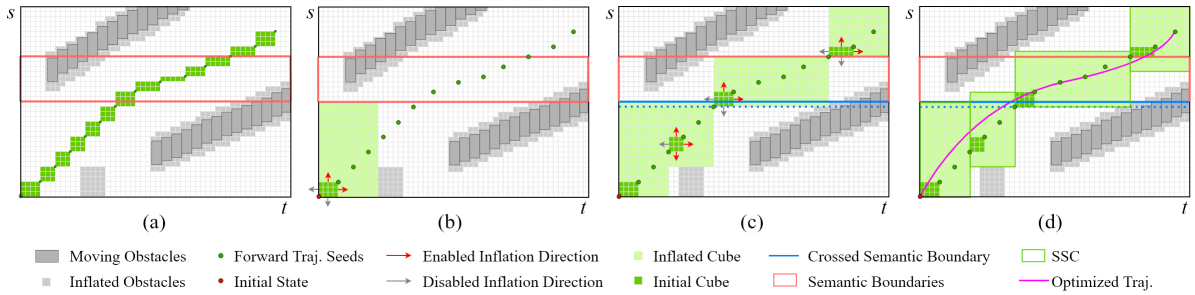


Figure 5.4: Illustration of a toy example of the SSC generation algorithm in the st domain. There is a speed limit which takes effect between the two *orange* boundaries, as shown in (a). To begin, the first initial cube is inflated until the two inflation directions touch the semantic boundary and the obstacle, as shown in (b). Next, the last seed in the first cube and the first seed outside the first cube are picked out to construct the second initial cube, as shown in (c). The inflation for the second initial cube terminates at the semantic boundary. Then for the third initial cube, the inflation direction opposite to the entry direction is disabled. After the cube inflation, a cube relaxation process is applied depending on the constraints associated and the free-space, as shown in (d).

left, lane change right, and lane keeping) at the same time, we fully utilize this feature and generate candidate trajectories for all the potential behaviors to enhance the robustness of the framework. For example, while executing a lane change trajectory, our trajectory framework always prepares the trajectory for switching back to the original lane, as shown in Fig. 5.3.

5.4 Spatio-temporal Semantic Corridor

5.4.1 Semantic Elements And Frenét Frame Representation

We deal with an slt 3-D configuration space which consists of the longitudinal direction s, the lateral direction l and the time t. The longitudinal and lateral directions are with respect to a Frenét frame, which is a dynamical reference frame constructed from the reference lane. Typically, the reference lane is extracted from the route information provided by a route planner, as illustrated in Fig. 5.2. For an unstructured environment where there is no lane available, the reference lane can also be provided by a path planner [123].

Rather than generating the corridor in Cartesian coordinates, we adopt the Frenét frame representation since most of the semantic elements are associated with the lane geometry. For example, speed limits, traffic lights and stop signs are typically associated with a certain longitudinal range of a lane. Moreover, since human-like driving behavior can typically be decoupled into lateral movements and longitudinal movements, modeling the free-space in these two directions is a more natural representation than modeling free-space in Cartesian coordinates. Time is another necessary dimension since many semantic elements are time-indexed. For instance,

the predicted trajectory is time-profiled and can be regarded as a series of spatio-temporal obstacles.

Two typical examples of projecting the semantic elements to a Frenét frame are depicted in Fig. 5.1 and Fig. 5.3, respectively. Diverse kinds of semantic elements can be generally divided into two categories: obstacle-like and constraint-like semantic elements. We elaborate on this in the following.

Obstacle-like semantic elements

Many semantic elements have the physical meaning that a certain portion of the slt domain is not allowed to be driven in. For example, static obstacles can be viewed as obstacles across whole time axes, and dynamic obstacles can be viewed as a series of static obstacles in the time domain according to the predicted trajectory, while a red light can be rendered as an obstacle occupying a particular longitudinal position and time period. After rendering obstacle-like semantic elements to the slt domain, the configuration space is a 3-D occupancy grid.

Constraint-like semantic elements

Apart from the obstacle-like semantic elements, many semantic elements represent dynamical constraints or time constraints. For example, speed limits and stop signs can be viewed as velocity constraints. There are also semantic elements which pose time constraints. For instance, when crossing lanes, the total time of the lane change should not be unreasonably long.

We propose a unified representation, i.e., *semantic boundaries*, for all the constraint-like semantic elements. For instance, a speed limit can be regarded as the velocity constraint applied to a longitudinal range $[s_{\text{begin}}, s_{\text{end}}]$, where s_{begin} and s_{end} are the two semantic boundaries. The lane change duration constraint can be regarded as a time constraint applied to the lateral range $[d_{\text{begin}}, d_{\text{end}}]$ of the current lane. Essentially, the semantic boundaries represent where a certain semantic element starts and stops taking effect.

Note that there is a minor difference in terms of the "hardness" of the constraints. Specifically, the constraints posed by traffic rules (e.g., speed limit) are hard constraints which should be followed without any compromise. Other constraints (e.g., lane change duration constraint) are required for a natural human-like behavior and there is no universal quantitative description of such constraints. We take the difference into account during the corridor generation process (Sect. 5.4).

Algorithm 6: Semantic Corridor Generation

1 Inputs: forward simulated states $\{x_0, x_1, \ldots, x_t\}$, initial state x_{des}, semantic boundaries $\mathcal{B}$, slt configuration space $\mathcal{E}$;
2 Initializes: seeds $\mathcal{S}^{\text{seed}} = \varnothing$;
3 $\mathcal{S}^{\text{seed}} \leftarrow \texttt{SeedGeneration}(\{x_0, x_1, \ldots, x_t\}, x_{\text{des}})$;
4 $\mathcal{C}^{\text{infl}} \leftarrow \texttt{CubeInflation}(\mathcal{S}^{\text{seed}}, \mathcal{B}, \mathcal{E})$;
5 $\mathcal{C}^{\text{infl}} \leftarrow \texttt{ConstraintAssociation}(\mathcal{C}^{\text{infl}}, \mathcal{B})$;
6 $\mathcal{C}^{\text{final}} \leftarrow \texttt{CubeRelaxation}(\mathcal{C}^{\text{infl}}, \mathcal{E})$;

5.4.2 Semantic Corridor Generation

As outlined in Algo. 6, the generation process consists of seed generation (Line 3), cube inflation (Line 4), constraint association (Line 5) and cube relaxation (Line 6).

Seed Generation

The seeds of the semantic corridor are generated by projecting the forward simulated states of the behavior planner to the slt configuration space. Since the forward simulated states are discretized, the feasibility of the corridor generation process depends on the complexity of environments and seed resolution. To guarantee the success of the corridor generation process, we require the initial cubes constructed from consecutive seeds to be collision-free (Fig. 5.4 (a) and Line 6, Algo. 7). In practice, this clearance requirement is reasonable and easy to achieve. For example, for a vehicle travelling at a longitudinal speed of $30\ m/s$ and a seed resolution of $0.15\ s$ (similar to [76]), the clearance required is roughly $4.5\ m$, which is much shorter than the emergency braking distance at such a high speed. Therefore, it is reasonable to directly reject the cases which violate the proposed requirement.

The motivation for generating the corridor around the seeds is to fully model topologically equivalent free space, while preserving the same high-level behavior. For example, as shown in Fig. 5.4 (a), the semantic meaning of the seeds is to pass between the two dynamic obstacles, which is preserved by the corridor generation. Since the motion planner should work with any given initial state, the initial state should also be included in the seeds.

Cube Inflation with Semantic Boundaries

The corridor is generated by iterating over the seeds. The seeds which are already contained in the last inflated cube are skipped (Line 4, Algo. 7) since they are topologically equivalent. The initial cubes are generated based on two consecutive seeds, by regarding the two seeds as two

Algorithm 7: `CubeInflation`$(\mathcal{S}^{\text{seed}}, \mathcal{B}, \mathcal{E})$

```
1  Inputs: cube seeds S^seed, semantic boundaries B, slt configuration space E;
2  Initializes: inflated cubes C^infl = ∅ ;
3  for i = 2, ..., |S^seed| do
4      if !IfContainedInLastCube(s_i^seed, C^infl) then
5          c ← GetInitialCubeBySeed(s_i^seed, s_{i-1}^seed);
6          if !IfInitialCubeFree(c, E) then
7              return;
8          end
9          D ← GetInflDirsBySemBoundaries(c, B);
10         c^infl ← InflateCubeInDirs(c, D, B, E);
11         C^infl ← C^infl ∪ c^infl
12     end
13 end
```

cube vertices (Line 5, Algo. 7).

The key feature of the cube inflation is the consideration of the semantic boundaries (Line 9, Algo. 7). The goal of the cube inflation process is to generate cubes which match the semantic boundaries so that the constraints can be conveniently associated. Specifically, when the initial cube intersects with a certain semantic boundary, the inflation direction opposite to the entry direction is disabled, so that the inflated cube can almost match the semantic boundaries. The inflation alternates among three slt directions for one step of inflation and terminates if this step collides with an obstacle or intersects with a certain semantic boundary. A toy example is provided in Fig. 5.4 (b) and (c). Since in the optimization (Sect. 5.5) each cube corresponds to one piece of the trajectory and to preserve convexity we do not explicitly optimize the durations of the pieces, the time upper bound of the current cube should coincide with the time lower bound of the next cube. One may consider optimizing the durations (which is non-convex) and in such case, a further inflation to increase overlapping in the t dimension can be beneficial.

Cube Relaxation

After the cube inflation process, the inflated cubes almost match the semantic boundaries, as shown in Fig. 5.4 (c). However, as mentioned in 5.4.1, some constraints, such as the lane change duration constraint, are soft and extra space should be left for optimization. To this end, we adopt a cube relaxation process to relax the cube boundaries while preserving the hard constraints and collision-free property, as shown in Fig. 5.4 (d). The maximum margin allowed for the relaxation is systematically determined by the constraints applied to the two consecutive cubes. For example, in the longitudinal direction, the margin can be dermined by velocity

matching distance according to the velocity constraints. For the lateral direction (i.e., the lane change case), the margin can be calculated by the allowed fluctuation of lane change duration.

5.5 Trajectory Generation With Safety and Feasibility Guarantee

Given the constraints specified by the SSC, we present an optimization-based trajectory generation method which can find the optimal trajectory within the SSC while satisfying the dynamical constraints. The optimization problem is also formulated in the Frenét frame, which is consistent with the SSC representation. In [99], Werling *et al.* use a quintic monomial polynomial for both the longitudinal and lateral direction based on the optimal control theory. However, the quintic monomial polynomial is not suitable for the optimization in the SSC for the following two reasons: 1) one segment of the polynomial only has limited representation ability and may fail to represent a highly constrained maneuver required by the SSC, and 2) the monomial basis polynomial is not well suited to problems with complex configuration space obstacles and dynamical constraints. In previous works on monomial basis polynomial trajectories [99, 124], the constraints are only enforced/checked on a finite set of sampled points. However, this method may fail to detect collision between sample points, and thus cannot provide any guarantee on safety and feasibility.

In this chapter, we remove the above two limitations by using a piecewise Bézier curve for the two-dimensional trajectory (i.e., the longitudinal direction $s(t)$ and lateral direction $l(t)$) along the reference lane. The reason for using the piecewise Bézier curve is its convex hull property and hodograph property [125].

5.5.1 Bézier Basis and Its Properties

A degree-m Bézier curve $f(t)$ is defined on a fixed interval $t \in [0, 1]$ by $m + 1$ control points as follows,

$$f(t) = p_0 b_m^0(t) + p_1 b_m^1(t) + \cdots + p_m b_m^m(t) = \sum_{i=0}^{m} p_i \cdot b_m^i(t), \tag{5.1}$$

where p_i denotes the control point and $b_m^i(t) = \binom{m}{i} t^i \cdot (1-t)^{m-i}$ is the Bernstein basis. Denote the set of control points $[p_0, p_1, \ldots, p_m]$ as $\mathbf{p}$.

The *convex hull* property is suitable for the problem of constraining the curve in a convex free-space. Specifically, the Bézier curve $f(t)$ is guaranteed to be entirely confined in the convex

hull supported by the control points $\mathbf{p}$. In other words, by constraining $\mathbf{p}$ inside the convex free-space, the resulting curve is guaranteed to be collision-free.

The *hodograph* property facilitates constraining high-order derivatives of the Bézier curve, which is useful for enforcing dynamical constraints. By the hodograph property, the derivative of a Bézier curve $\frac{df(t)}{dt}$ is another Bézier curve with control point $p_i^{(1)} = m \cdot (p_{i+1} - p_i)$. By applying the convex hull property on the derivative Bézier curve, the entire dynamical profile of the original curve $f(t)$ can be confined within a given dynamical range, as shown in Fig. 5.5.

5.5.2 Piecewise Bézier Curve Representation

In this chapter, we adopt a piecewise Bézier curve representation with each piece associated with one cube of the SSC. Accordingly, the j-th segment of an n-segment piecewise Bézier trajectory in one dimension $\sigma \in \{s, l\}$ is given by

$$f_j^\sigma(t) = \begin{cases} \alpha_1 \cdot \sum_{i=0}^{m} p_i^1 \cdot b_m^i(\frac{t-t_0}{\alpha_1}), & t \in [t_0, t_1] \\ \alpha_2 \cdot \sum_{i=0}^{m} p_i^2 \cdot b_m^i(\frac{t-t_1}{\alpha_2}), & t \in [t_1, t_2] \\ \quad\vdots & \quad\vdots \\ \alpha_n \cdot \sum_{i=0}^{m} p_i^n \cdot b_m^i(\frac{t-t_{n-1}}{\alpha_n}), & t \in [t_{n-1}, t_n], \end{cases} \tag{5.2}$$

where p_i^j denotes the i-th control point of the j-th segment and $t_0, t_1, \ldots, t_n$ are the time stamps of the start point and end point for each segment. Since the Bézier curve is defined on the fixed interval $[0, 1]$ while the trajectory duration for each segment may vary, we introduce a scaling factor α_j for each segment according to its duration, similar to [125].

Similar to [99], we minimize the cost function given by the time integral of the square of the jerk. Specifically, the cost J_j of the j-th segment can be written as,

$$J_j = w_s \int_{t_{j-1}}^{t_j} \left(\frac{d^3 f^s(t)}{dt^3} \right)^2 dt + w_l \int_{t_{j-1}}^{t_j} \left(\frac{d^3 f^l(t)}{dt^3} \right)^2 dt, \tag{5.3}$$

where w_s and w_l denote the weight for the control cost of the longitudinal direction and lateral direction, respectively. The objective is simple and invariant given different combinations of semantic elements thanks to the SSC, which allows the formulation to easily adapt to different traffic configurations.

Denote by $y_j^\sigma(t)$ the non-scaled Bézier curve in the interval $[0, 1]$ with $\mathbf{p}_j$ as the control points. Let $u = \frac{t-t_{j-1}}{\alpha_j}$ denote the normalized time of the non-scaled Bézier curve, the cost of the j-th segment on dimension σ can be rewritten using the non-scaled $y_j^\sigma(t)$ as follows,

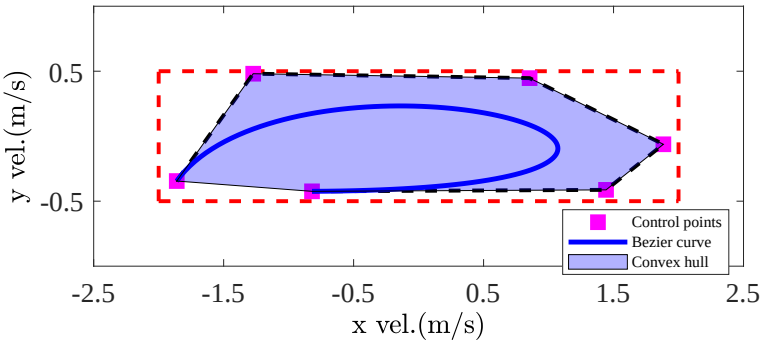


Figure 5.5: Illustration of using the convex hull property to constrain a velocity profile inside a feasible region (dashed red lines).

$$J_j^\sigma = \int_0^1 \alpha_j \cdot \left(\frac{d^3(\alpha_j \cdot y_j^\sigma(t))}{d(u \cdot \alpha_j)^3} \right)^2 du = \frac{1}{\alpha_j^3} \cdot \mathbf{p}_j^{\mathrm{T}} \mathbf{Q} \mathbf{p}_j,$$

where $\mathbf{Q}$ is the Hessian matrix of the non-scaled Bézier curve. We omit the detailed calculation of $\mathbf{Q}$ for brevity.

5.5.3 Enforcing Safety and Dynamical Constraints

In this chapter, we adopt a quintic (m=5) piecewise Bézier curve as the trajectory parameterization. According to the hodograph property, the k-th derivative of the non-scaled Bézier curve $\frac{d^k y_j^\sigma(t)}{dt^k}$ is supported by control points $\mathbf{q}_j^{\sigma,(k)}$ which can be calculated by induction as follows,

$$q_{j,i}^{\sigma,(0)} = p_i^j, q_{j,i}^{\sigma,(k)} = \frac{m!}{(m-k)!}(q_{j,i+1}^{\sigma,(k-1)} - q_{j,i}^{\sigma,(k-1)}). \tag{5.4}$$

Based on this property, the k-th-order derivatives at the boundaries of $f_j^\sigma(t)$ can be expressed as

$$\frac{d^k f_j^\sigma(t_{j-1})}{dt^k} = \alpha_j^{1-k} \cdot q_{j,0}^{\sigma,(k)}, \frac{d^k f_j^\sigma(t_j)}{dt^k} = \alpha_j^{1-k} \cdot q_{j,m}^{\sigma,(k)}, \tag{5.5}$$

respectively. Moreover, by further applying the convex hull property, we can constrain the entire derivative profile of $f_j^\sigma(t)$ using the following sufficient condition,

$$\beta_{j,-}^{\sigma,(k)} \leq \alpha_j^{1-k} \cdot q_{j,i}^{\sigma,(k)} \leq \beta_{j,+}^{\sigma,(k)}, \forall i \Rightarrow \beta_{j,-}^{\sigma,(k)} \leq \frac{d^k f_j^\sigma(t)}{dt^k} \leq \beta_{j,+}^{\sigma,(k)}, \tag{5.6}$$

where $\beta_{j,-}^{\sigma,(k)}$ and $\beta_{j,+}^{\sigma,(k)}$ denote the lower and upper bound on dimension σ for the k-th derivative of the j-th segment.

Desired state constraints

First of all, the generated trajectory should start from the given initial state $[\sigma_{t_0}^{(0)}, \sigma_{t_0}^{(1)}, \sigma_{t_0}^{(2)}]$ and terminate at the given goal state $[\sigma_{t_n}^{(0)}, \sigma_{t_n}^{(1)}, \sigma_{t_n}^{(2)}]$ for $\sigma \in \{s, l\}$, where $\sigma_t^{(k)}$ denotes the k-th-order derivative at time t. Specifically, this requires enforcing equality constraints for the first and last segment as follows,

$$\frac{d^k f_0^\sigma(t_0)}{dt^k} = \sigma_{t_0}^{(k)}, \qquad \frac{d^k f_n^\sigma(t_n)}{dt^k} = \sigma_{t_n}^{(k)}, \tag{5.7}$$

where $k = 0, 1, 2$. By applying Eq. 5.5, these constraints can be written as linear equality constraints w.r.t. $\mathbf{p}$.

Continuity constraints

The generated trajectory should be continuous for all the derivatives up to the k-th order at all the connecting points between two consecutive pieces. The continuity constraints between the j-th segment and the $j+1$-th segment can be written as

$$\frac{d^k f_j^\sigma(t_j)}{dt^k} = \frac{d^k f_{j+1}^\sigma(t_j)}{dt^k}, \tag{5.8}$$

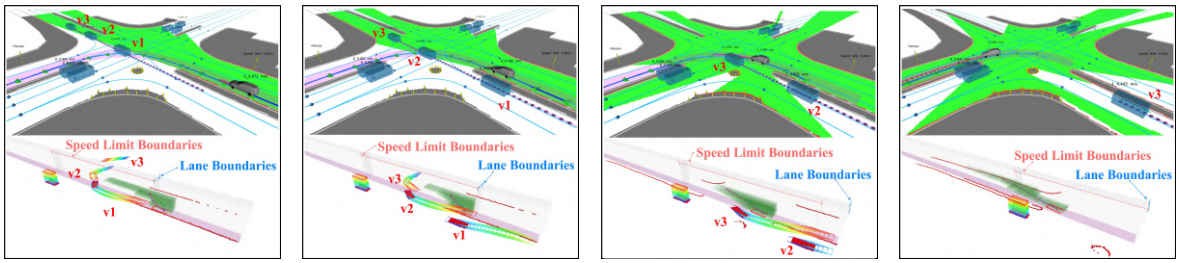
where $k = 0, 1, 2, 3$. By applying Eq. 5.5, these constraints can also be written as linear equality constraints w.r.t. $\mathbf{p}$.

Free-space constraints

To guarantee the generated trajectory is collision-free, we constrain each segment of the trajectory within the corresponding cube. The free-space constraint of the j-th segment on dimension σ can be enforced by using the sufficient condition (Eq. 5.6) under $k = 0$, where $\beta_{j,-}^{\sigma,(0)}$ and $\beta_{j,+}^{\sigma,(0)}$ represent the position bounds on dimension σ given by the shape of the cube.

Dynamical constraints

To comply with the environment semantics and dynamical feasibility constraint, we enforce the constraints on the derivatives of trajectories by using the sufficient condition (Eq. 5.6), where $k = 1, 2$. The physical meaning is that the maximum lateral/longitudinal velocity and acceleration is constrained. Summarizing all the linear equality and inequality constraints, the



(a) Approaching (b) Reducing speed to wait (c) Accelerating to pass (d) Completing the left turn

Figure 5.6: Illustration of an unprotected left turn in a busy urban intersection. When the ego vehicle is approaching the intersection, it finds the left turn is not feasible and it reduces speed to wait. Once feasible, the vehicle quickly accelerates to complete the left turn.

overall formulation can be written as a QP, which can be solved efficiently using off-the-shelf solvers (such as OOQP). Although Eq. 5.6 is a sufficient condition, in practice we find it does not result in over-conservative behavior, as shown in Sect. 5.6. In the case that no feasible solution can be found, the error is fed back to the behavior layer for further reaction.

5.6 Experimental Results

5.6.1 Implementation Details

The experiments are conducted in a multi-agent simulation platform, as illustrated in Fig. 5.2. In the simulation, dynamic agents potentially interact with each other, but the interaction model is unknown to the planner. The ego (our) vehicle only has a limited sensing range for the environment semantics. The route planner finds a random route for the ego vehicle at the beginning of the mission, and the route information of other agents is also unknown to the planner. The prediction method is similar to that in [60] which decouples the problem into behavior prediction and trajectory prediction. A long prediction horizon facilitates accounting for long-term future rewards, which potentially results in a more consistent output compared to using a short prediction horizon. However, the uncertainty also scales with the prediction horizon. Therefore, it is beneficial to characterize the long-term prediction uncertainty, and we provide an attempt in [126]. All the test environments are annotated from real satellite maps via QGIS. The planning method proposed in this chapter[1] is implemented in C++11. All the experiments are conducted on a desktop computer equipped with an Intel I7-8700K CPU, and our proposed method can run stably at 20 Hz.

[1] Source code is released at `https://github.com/HKUST-Aerial-Robotics/spatiotemporal_semantic_corridor`.

5.6.2 Qualitative Results

To verify that our proposed method can automatically adapt to different traffic configurations with different semantic elements, we choose three representative test cases.

Merging into congested traffic due to road construction

As illustrated in Fig. 5.3, this case is used to verify the capability of dealing with road construction, lane change (lane geometry), dynamic obstacles and the speed limit at the same time. The constructed SSC generally encodes the necessary information for optimization. The optimal trajectories are generated without explicitly caring about what types of semantic elements are present.

Overtaking on an urban expressway

This case is to validate the capability of dealing with high-speed traffic. The SSC is shown to be suitable for this time-critical scenario. As illustrated in Fig. 5.7, our method conducts a safe and smooth overtaking on an urban expressway with a speed of around $20\ m/s$. The limitation is that the prediction uncertainty is not sufficiently considered in the current SSC generation process, which is left as important future work.

An unprotected left turn at an intersection

This case is used to verify the capability of quickly responding to complex interactions with other agents during traffic negotiation. There is also a speed limit which poses hard speed constraints for the whole interaction process. As shown in Fig. 5.6, our method efficiently finds safe and feasible trajectories so that the vehicle precisely follows the behavior plan and navigates smoothly.

5.6.3 Comparisons and Analysis

We conduct a quantitative comparison with the seminal work [99], which is based on optimal primitives in the Frenét frame. In [99], the primitives are regularly sampled around a local target state with a certain resolution in the slt domain, and for different behaviors, the strategy for choosing the local target is different.

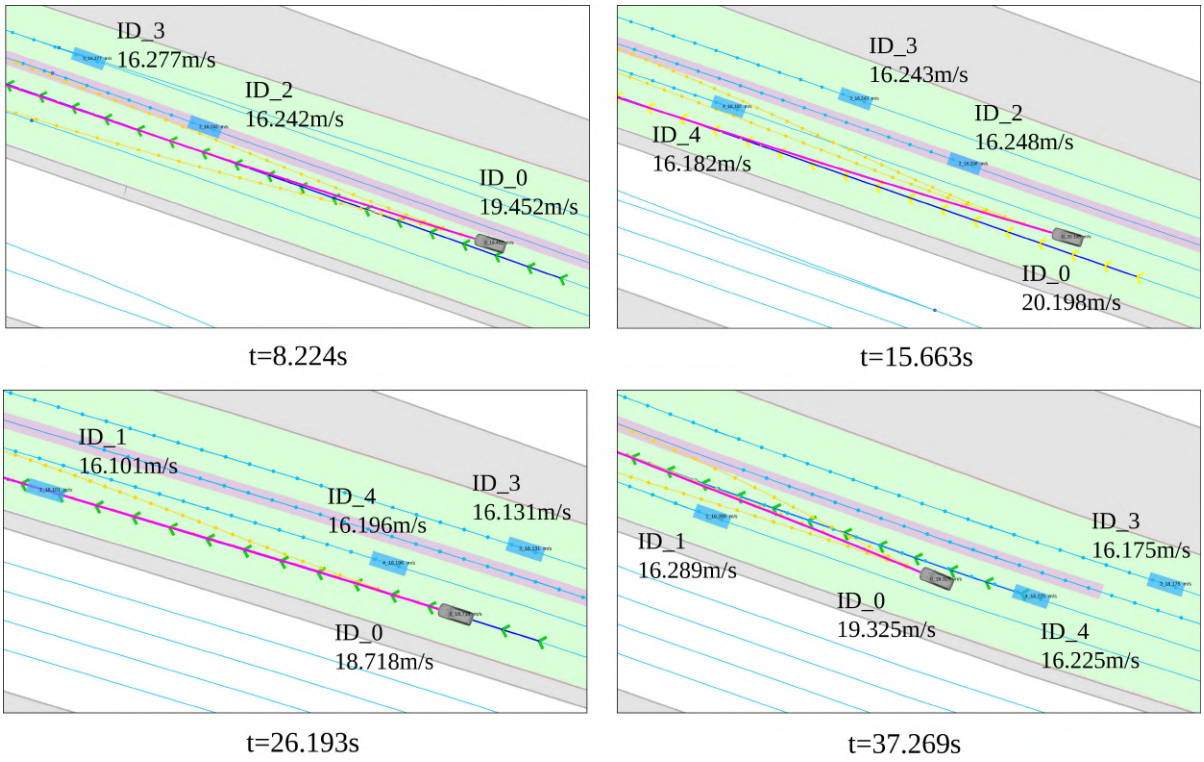
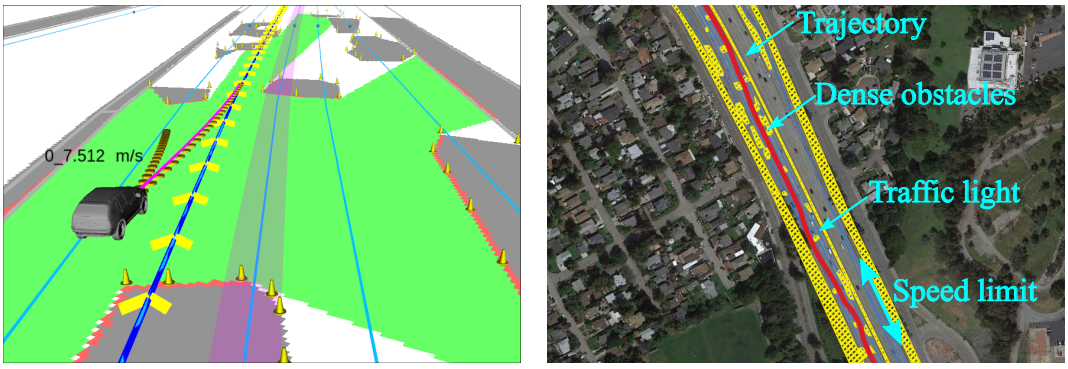


Figure 5.7: Illustration of overtaking on an urban expressway.

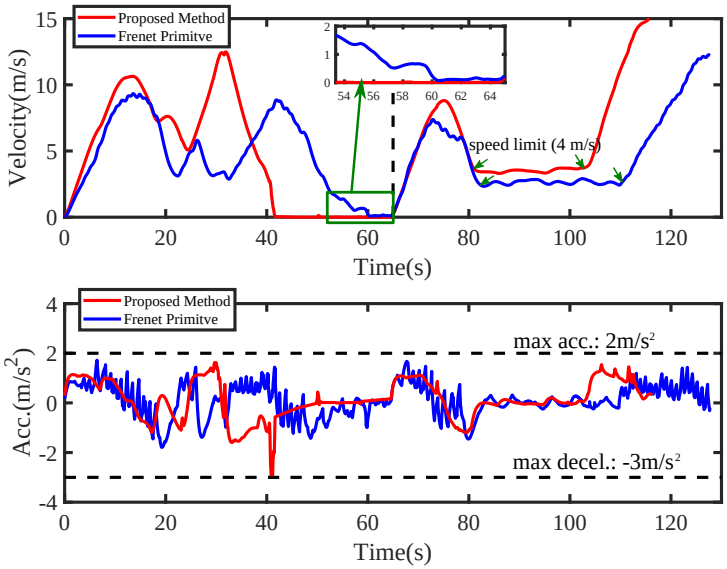
To conduct a fair comparison, we set up a benchmark track which is annotated from a real satellite map, as shown in Fig. 5.8b. To test the planner's response to semantic elements, we add a red light checkpoint and a speed limit to the track. Moreover, dense obstacles are placed on the track, as shown in Fig. 5.8a, to test the collision avoidance performance. Since the ego vehicle only has a limited sensing range (around 100 m), the collision avoidance task requires frequent replanning. The maximum acceleration and the maximum deceleration are set to 2 m/s^2 and 3 m/s^2, respectively. We use the same behavior planner (MPDM) for both our method and [99] to generate the lane change command. The user-desired velocity is set to 15 m/s for the behavior planner.

Collision-avoidance in cluttered environments

The first segment of the track is around 320 m from the starting point to the red light. As shown in Fig. 5.8c, our method can fully utilize the maneuverability of the vehicle and arrive at the red light at 42 s, about 14 seconds earlier than [99]. Moreover, our acceleration profile is smoother while staying within the dynamical limit. The reason is that our SSC representation models the continuous solution space, while the baseline method suffers from discretization and limited state space coverage. We observe that the benefit of using the corridor representation is more



(a) Benchmark track (b) Satellite map



(c) Comparison of the dynamic profile

Figure 5.8: Illustration of the comparison on a benchmark track.

obvious in the cluttered environments since many primitives of the baseline method become infeasible in this case.

Precise stop with a high entry speed

There is a red light checkpoint in the middle of the track, as shown in Fig. 5.8b, and the vehicle needs to complete a precise stop with a high entry speed. As shown in Fig. 5.8c, our method can reach the precise stop with an entry speed of $13\ m/s$ while the max deceleration is strictly bounded inside the dynamic range. However, for the baseline method [99], a stopping mode is needed to fix the local target state so that the replanning process can consistently reach the target boundary condition. If we do not manually fix the local target state and dynamically calculate it based on zero desired velocity, the initial sampled stopping trajectory may not be sampled in later replanning due to a minor change of the target state. This may cause rolling, as shown in

Fig. 5.8c. In contrast to [99], our method explicitly enforces the stopping boundary condition and achieves a precise stop.

Collision avoidance under a low speed limit

In addition to the study of high-speed collision-avoidance, we are also interested in the low-speed performance. To test this, a $4\,m/s$ speed limit is placed on the track, as shown in Fig. 5.8b. As depicted in Fig. 5.8c, our method strictly follows the speed limit.

We also conduct experiments in which obstacles are placed in an online manner (see our video for details).

5.7 Conclusion

In this chapter, we propose a trajectory generation framework for complex urban environments. Our main contribution is twofold. First, we present an SSC structure which copes with an arbitrary combination of semantic elements in a unified way. Second, we present a trajectory optimization formulation which guarantees the safety and feasibility of the output trajectory. The proposed method is extensively analyzed using various traffic configurations and complex semantic elements. The main limitation is that the prediction uncertainty and interaction uncertainty are not sufficiently modeled, which is the research direction we are currently working on [126]. Moreover, we find that the Bézier curve is also useful for non-linear trajectory optimization for AVs.

CHAPTER 6

SYSTEM INTEGRATION AND FIELD TESTS FOR AUTONOMOUS VEHICLES

6.1 Scientific Background and Motivation

Autonomous driving has been an emerging topic both in the industry and in the academic community nowadays. Planning, as one of the core components of the autonomous driving system, largely determines the intelligence and user experience of the system. Despite that there are massive industrial demos illustrating promising autonomy, few methodology details are provided to show how pain points in planning are systematically dealt with. On the other hand, various planning systems are presented in the academic community, which cover several pain points such as uncertainty and interaction modeling [40, 102, 109, 112, 118, 121, 127–129]. However, most of these methods are only validated through simulation or well-annotated datasets, leaving the question that whether these methods can work on a real autonomous vehicle.

Planning on a real vehicle with closed-loop execution is far more challenging than planning in simulation or open-loop validation using well-annotated datasets. Onboard planning requires dealing with an imperfect world and interacting with other traffic participants naturally. The imperfectness comes from a wide range of aspects, such as the occlusion in sensing, detection and tracking noises, and unavoidable stochastic behaviors of other traffic participants, which are hard to be reproduced in simulation. Social compliance is also difficult to be directly perceived without onboard validation. In this chapter, we target at building a robust and socially-compliant planning system which can work with the imperfect real world.

In this chapter, we present an Efficient Planning System for autonomous vehicles In highLy interactive envirONments (EPSILON). By presenting EPSILON, we systematically investigate several pain points of planning for autonomous driving, such as the modeling of uncertainty and multi-agent interaction, and attempt to achieve one little but a concrete step towards automated driving in the real-world. We validate the performance of EPSILON by conducting long-term closed-loop autonomous driving in complex driving environments.

EPSILON follows a hierarchical structure that consists of a behavior planning layer and a motion planning layer as in many previous methods [101, 102, 129]. In EPSILON, the role of

behavior planning is generating a preliminary decision which is represented by a sequence of states covering the planning horizon, while the role of motion planning is to wrap the sequence of states to a safe and smooth trajectory for closed-loop execution.

There is an extensive literature in behavior planning for automated vehicles. We observe that, in the early stage, most of the methods focused on a pure geometry perspective, and adopted various rule-based or search-based techniques [101, 102, 109, 117, 124, 129–131]. These methods typically assumed a behavior/trajectory prediction is equipped. The prediction is conducted independently of planning, and when the prediction is provided, the planner plans a trajectory with a sufficiently large safety margin. However, it is impossible to yield a perfect prediction due to the stochastic nature of other traffic participants and onboard perception noises. Moreover, with such independent prediction, the mutual influence of the ego vehicle's and other vehicles' future motion cannot be modeled. In cooperative and interactive scenarios, such as merging scenarios, where this mutual influence is essential, coupled prediction and planning are more promising [132]. Therefore, more methods turn to a probabilistic and multi-agent perspective. Partially observable markov decision process (POMDP) [104] provides a rigorous form of modeling uncertainties and multi-agent interactions, while suffering from prohibitive high computationally complexity. Several attempts [83–85, 133] have been to made to accelerate the problem solving but are still not efficient enough for driving in highly dynamic environments [88, 93]. In this chapter, we propose a guided branching technique to focus the exploration using domain knowledge. Our behavior planning method is much more efficient and can work in highly dynamic city traffic.

The output decision is fed to the motion planning layer, and the remaining problem is generating a safe and smooth trajectory which faithfully follows the decision. Our motion planning layer follows an optimization-based framework. All the constraints posed by complex semantics are encoded in a unified way using a spatio-temporal semantic corridor. The piecewise Bézier curve is adopted as trajectory parameterization for its convex hull property and hodograph property which can enforce safety and dynamical feasibility for the entire trajectory.

A preliminary version of the behavior planning method of EPSILON was originally presented in [134]. In [134], although the interaction among traffic participants is captured using multi-agent forward integration, it generally assumes other participants are *rational*. The rationality is reflected in the predefined multi-agent integration model. However, in real-world city driving, we find traffic participants are often *noisily rational*, especially in cities where the driving style is aggressive (e.g., China). There are common cases where other traffic partici-

pants are uncooperative and not interacting according to the pre-defined model. The resulting decision in [134] may be over aggressive and risky. In this chapter, we introduce the safety mechanism by incorporating responsibility-sensitive safety into the multi-agent forward integration, which better guarantees safety even encountering over-aggressive traffic participants. The motion planning method of EPSILON was originally proposed in [40] but multi-agent interaction was not modeled. There exists an inconsistency between interaction-aware behavior planning and interaction-unaware motion planning. In this chapter, we advance the motion planning method to incorporate multi-agent interaction and unify behavior planning and motion planning.

We summarize our contributions as follows.

1. We propose an efficient and complete planning system that systematically deals with several paint points of planning for automated vehicles.
2. Beyond validation using simulation and datasets, we extensively validate our system on a real vehicle in dense city traffic without an HD-map and purely relying on onboard sensor suites.
3. We advance the safety mechanism of the behavior planning method in [134] which better guarantees safety even encountering over-aggressive or uncooperative traffic participants.
4. We release all components in the planning system as open-source packages.

6.2 Related Work

Behavior planning for automated vehicles: Many works on behavior planning[1] for automated vehicles have been published in recent years. Many of them focus on a geometric perspective, namely, finding preliminary geometric collision-free paths/trajectories for the controlled vehicle. State-machine-based [101, 130, 131] behavior planning, which employed handcrafted rules for different driving conditions, was popular at an early stage. Based on the output state and action, reference paths can be extracted and fed to the motion planning layer. However, due to the complexity of real-world driving, it requires continuous engineering and tuning efforts to maintain the state machine. To this end, many methods then turned to a more general formulation for behavior planning, such as search-based methods [102, 109, 117, 124, 129].

[1] In this chapter, the two terms "behavior planning" and "decision-making" are used interchangeably.

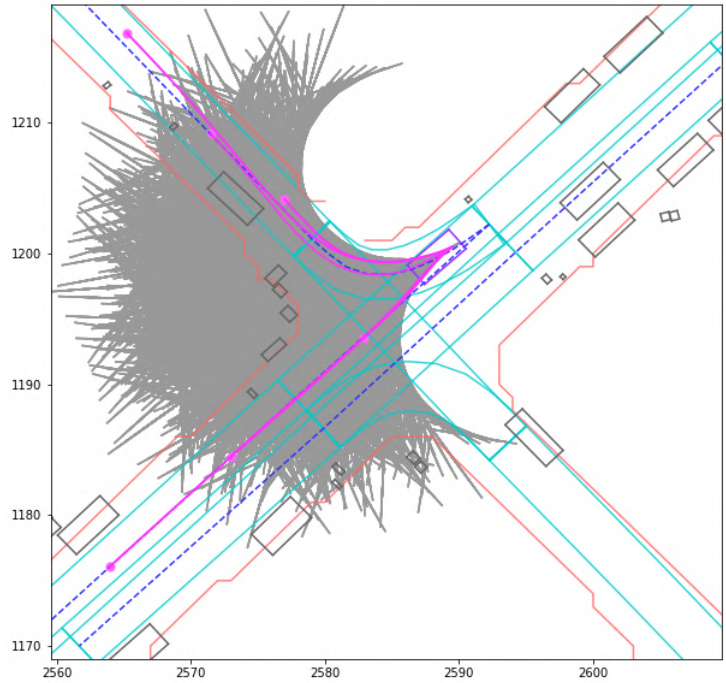


Figure 6.1: Illustration of using semantic-level action to guide the exploration of the action space. Even with coarsely discretized control actions (e.g., three discretized longitudinal velocities and three lateral accelerations), the action space to be explored (*gray*) of a POMDP-based planner is huge during a multi-step look-ahead search, while much of the space explored is of low-likelihood. The situation is much worse when considering the multi-agent setting. After incorporating semantic-level action (e.g., pursuing center lines with different aggressiveness) using predefined controllers, the exploration (*purple*) can be guided using domain knowledge. Although much of the exploration is tailed, the resulting exploration still faithfully captures the potential high-level decisions.

For example, Hubmann et. al. [117] uses A* graph search on a state lattice with static and dynamic events encoded as cost map. The methods which adopt pure geometric reasoning typically assume a deterministic trajectory prediction of other traffic participants is provided for the purpose of collision-checking. However, in real-world driving prediction should be modeled from a probabilistic perspective, due to the multi-modal and uncertain nature of the prediction problem. Moreover, different future actions of the controlled vehicle may result in a different future situation, which is also not modeled in pure geometric reasoning. In this chapter, we tackle the behavior planning problem from a probabilistic and multi-agent perspective, which fundamentally addresses the above issues.

There is extensive literature on behavior planning from a probabilistic and multi-agent perspective, and many of the works are formulated in POMDP [104]. POMDP is mathematically rigorous and outlines a principled way to solve the problem of planning under uncertainty. Due to the *curse of dimensionality*, POMDP quickly becomes computationally intractable when the problem size scales. To this end, many online POMDP solvers have been proposed to accelerate the problem solving, such as [84–86, 133]. Leveraging the latest advance of POMDP solvers,

several works such as [90, 92, 93, 121] have been applying POMDP to the behavior planning for automated vehicles. However, most of them are only validated in simulation with particular scenarios such as merging or intersection, except for [90] which has onboard experiments in a crowd. Nevertheless, [90] only deals with a tailed 1-D problem, namely, speed optimization problem under a given path and still yields a limited efficiency (around 3 to 5 Hz), which may be inadequate for solving a full decision-making problem in highly dynamic driving environments. Our behavior planning method originates from POMDP, but our method utilizes domain knowledge (as shown in Fig. 6.1) and guided branching to achieve a highly efficient solution. Moreover, our method is validated against real-world highly dynamic environments and potentially stochastic traffic participants.

Motion planning for automated vehicles: Motion planning for automated vehicles is relatively mature compared to behavior planning. Typical routines for motion planning include search-based methods based on state-lattices such as [100, 113, 114] or motion primitives [99], and optimization-based methods such as [111, 112, 116, 118, 127], and a combination of these two, such as [110, 128]. Our previous work [40] belongs to the optimization-based methods. The key feature of [40] is that it proposes a unified input representation, i.e., the spatio-temporal semantic corridor structure to wrap complex constraints posed by various semantic elements. This input representation saves considerable efforts in engineering and tuning the constraints. However, like most motion planning methods, [40] still requires a trajectory prediction module and does not capture the multi-agent interaction, which poses inconsistency when being used with interaction-aware behavior planning methods. In this chapter, since future anticipation is coupled inside our behavior planning, we can directly feed the decision together with the conditional anticipation of other traffic participants to the motion planning layer, which makes the motion planning of [40] amenable to modeling interaction.

6.3 System Overview

The structure of the proposed planning system is shown in Fig. 6.2. The planning system is built on top of the environment understanding and closed-loop execution modules. There are several modules in the perception pipeline, including freespace detection, semantics detection (e.g., lane detection, traffic light detection, etc.), object detection, and tracking. Note that HD-Map is *optional* for our planning system. For onboard experiments on a real vehicle, we do not use HD-Map and purely rely on online lane detections. The output of perception is synchronized and

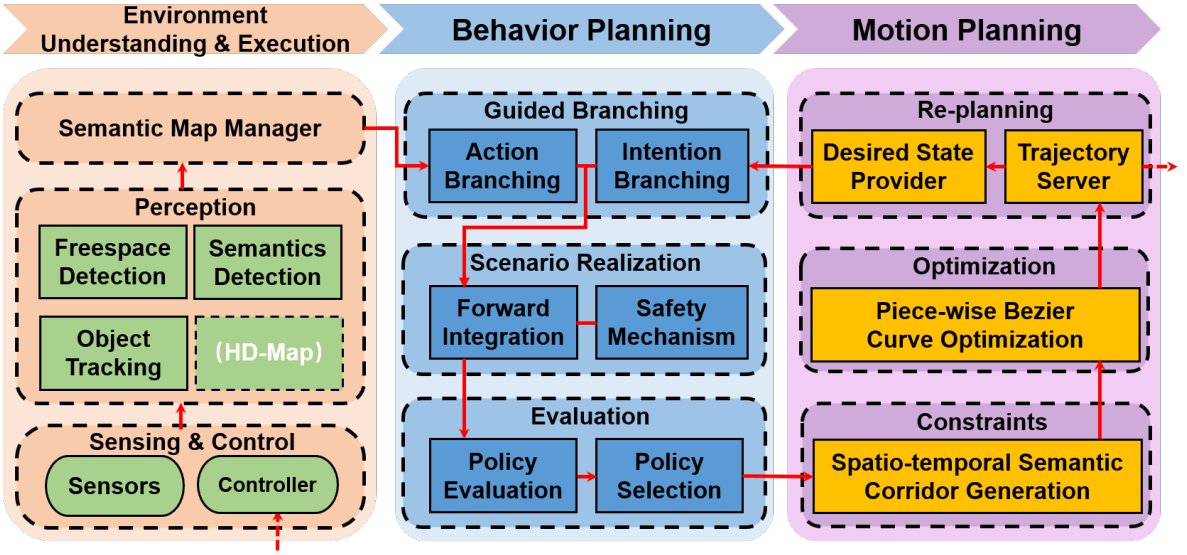


Figure 6.2: A diagram of EPSILON together with environment understanding and execution modules. HD-Map is optional for EPSILON.

fed to a semantic map manager module which is responsible for organizing the data structures and providing querying interfaces for planning. The semantic map is updated at 20 Hz and includes an occupancy grid for static obstacles, multiple tracklets for dynamical obstacles, and road structures for lanes and traffic semantics.

EPSILON follows a two-layer hierarchical structure, including a behavior planning layer and a motion planning layer. Note that there is no additional prediction module like in [40] since the prediction is coupled inside behavior planning. At a high level, our behavior planning consists of three processes, namely, guided branching, scenario realization and evaluation. Essentially, guided branching is responsible for expanding action sequences for the controlled vehicle and reasoning about the possible intentions of other traffic participants. By incorporating a particular ego action sequence with a particular intention combination of other traffic participants, we form a *scenario*. During scenario realization, we use closed-loop multi-agent forward integration to realize the scenario step-by-step. The novel part compared to our previous work [134] is the safety mechanism module, which is employed to ensure safety even when other traffic participants are noisily rational.

The motion planning layer basically follows our previous work [40]. First, static, dynamic obstacles and constraints posed by environment semantics are modeled using a spatio-temporal semantic corridor around the initial guess provided by the behavior planning layer. Then a piece-wise Bèzier curve is optimized with respect to the corridor. Using its convex hull property and hodograph property, the *entire* trajectory can be guaranteed to be safe and dynamically feasible. The generated trajectory is fed to a trajectory server for re-planning scheduling. The

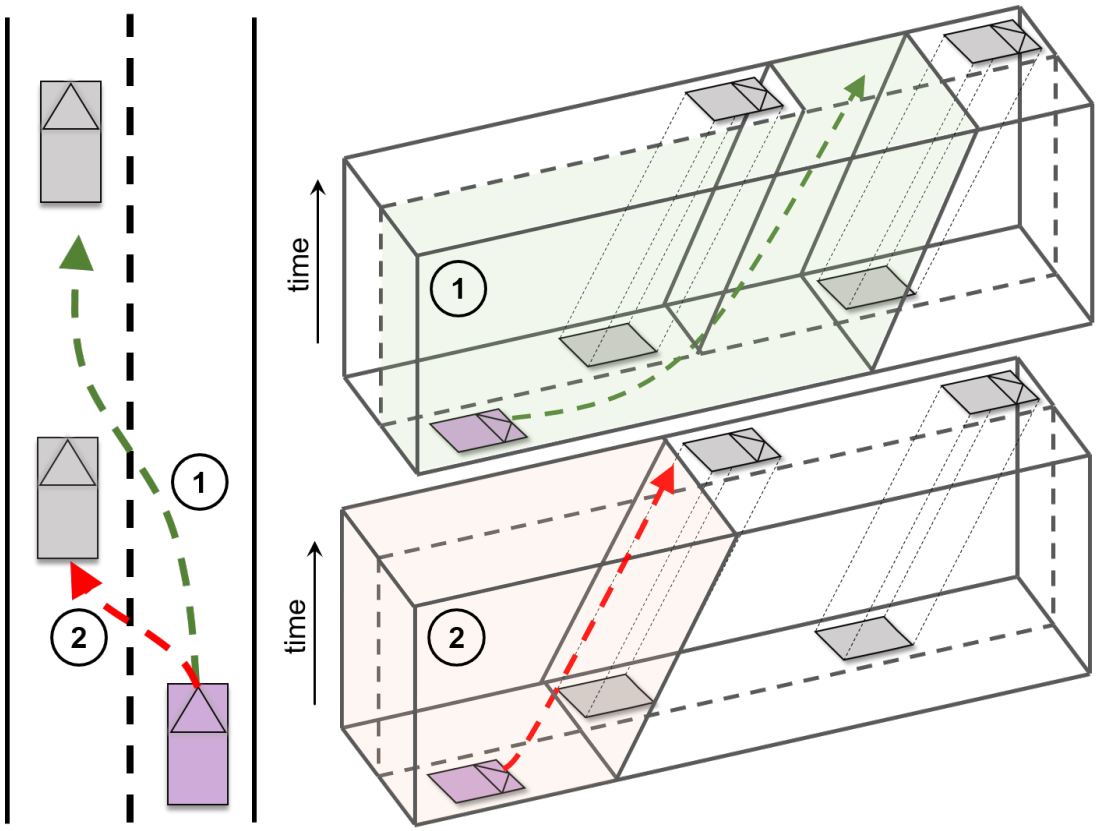


Figure 6.3: Illustration of the relationship between behavior planning and motion planning defined in this chapter. The controlled vehicle (marked in *purple*) can yield and insert behind the nearest vehicle (maneuver ①), or accelerate and insert between two nearby vehicles (maneuver ②). Visualized in the spatio-temporal domain, the two maneuvers belong to two different homotopy classes. The behavior planning targets at discovering diverse tactical maneuvers, while the motion planning is for generating a local smooth and safe trajectory.

desired state for re-planning can be queried from the trajectory server. The trajectory server is also responsible for sending out control commands to the vehicle controller to close the execution loop.

The design frequency for both layers is 20 Hz. In practice, the two layers can be assembled as a pipeline, which will increase the throughput of the whole planning system.

6.4 Complete System Formulation

In this section, we formulate the complete decision-making, intention prediction and motion planning problem for the proposed planning framework. As illustrated in Section 6.3, our framework follows a hierarchical structure with the behavior planning layer and motion planning layer. Let $\mathcal{E}$ denote the ego-centric local planning environment which is defined as the tuple $\mathcal{E} \coloneqq \langle \mathcal{O}, \mathcal{D} \rangle$ where $\mathcal{O} \in \mathbb{R}^2$ denotes the occupancy grid for static obstacles and $\mathcal{L}$ denotes

the road structures and traffic signals. Define s_t^i as the state of the vehicle $i \in V$ at time t and s_t^0 is reserved for the state of the ego vehicle. As a notational convenience, subscript absence denotes *all* time steps, and superscript absence denotes all agents. For example, s_t denotes the states of all vehicles at time t while s^i denotes the states at all time instances for vehicle i. Let $\mathcal{S}$ denote the collection of all the state variables for all the agents.

For the planning cycle at time t, the input to the behavior planning layer is $\langle \mathcal{S}, \mathcal{E}_t \rangle$. The output of the behavior planning layer is a *decision* D_t, which is parameterized by a sequence of discrete states $D_t \coloneqq [s_{t+1}, s_{t+2}, \ldots, s_{t+H}]$ for all agents where H denotes the planning horizon. The input to the motion planning layer is $\langle \mathcal{D}_t, \mathcal{E}_t \rangle$ while the output is a smooth and safe trajectory τ (typically parameterized by splines) for controlling the ego vehicle. The reason that we decouple the behavior planning layer and motion planning layer is for efficiency. After decoupling, the behavior planning layer only needs to reason about the future scenario at a coarse resolution, while the motion planning layer can work in the local solution space conditioning on the future decision $\mathcal{D}$.

As mentioned in Section 6.1, modeling uncertainty and interaction is critical for behavior planning. To this end, we formulate the behavior planning problem is the form of POMDP which provides a rigorous form for planning under uncertainty in a multi-agent setting. At time t, vehicle i can take an action $a_t^i \in \mathcal{A}^i$ to take the transition from state s_t^i to s_{t+1}^i. A state s_t^i is represented by a tuple of position, velocity, acceleration, angle and steer, while an action a_t^i is a tuple of controls for steering and throttle and brake. The vehicle dynamics is then represented by the conditional probability function $T(x_{t+1}, x_t, a_t) = p(x_{t+1}|x_t, a_t)$. The observation uncertainty (such as tracking error) can be represented as $Z(z_t^i, s_t) = p(z_t^i|s_t)$, where $z_t^i \in \mathcal{Z}^i$ is the observation made by vehicle i which contains the estimated position and velocities of other vehicles. Strictly speaking, solving a complete POMDP problem needs to find an optimal policy π^* that maximized the expected sum of rewards of the ego vehicle over the given planning horizon H, where a policy is a mapping $\pi : \mathcal{S} \times \mathcal{Z}^i \to \mathcal{A}^i$ that satisfied:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=t_0}^{t_0+H} R(s_t, \pi(x_t, z_t^i))\right], \tag{6.1}$$

where $R(s_t, a_t^i)$ is a real-valued reward function $R : \mathcal{S} \to \mathbb{R}$. In conclusion, a POMDP can be defined as a tuple of $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{Z}, \mathcal{E} \rangle$. In the context of driving without vehicle-to-vehicle communication, we can model the uncertainty on the behavior of other agents using a driving model function $D(a_t^i, s_t, z_t^i) = p(a_t^i|s_t, z_t^i)$ which implicitly assumes that the instantaneous action of each vehicle are independent of each other. The evolution formulation of $p(s_t)$ over time

is then described by

$$p^i(s_t^i, s_{t+1}^i, z_t^i, a_t^i) = p(s_t^i) \underbrace{p(z_t^i|s_t^i)}_{\text{observation}} \underbrace{p(s_{t+1}^i|s_t^i, a_t^i)}_{\text{transition}} \underbrace{p(a_t^i|s_t^i, z_t^i)}_{\text{driving model}}, \tag{6.2}$$

where $a_t^i \in \mathcal{A}^i, \forall i \neq 0$ is a latent variable which must be inferred from sensor observations, which is exactly the role of prediction. As a consequence, the POMDP formulation implicitly reasons about trajectory/intention prediction, which also indicates that behavior planning and prediction are coupled. We refer interested readers to [84, 104] for more details. It is also notable that most of methods which decouple prediction and behavior planning are simplifications of original POMDP formulation. Since solving the complete POMDP problem above is highly computationally expensive, we apply several domain-specific knowledge into the formulation to accelerate the problem solving.

Given the output of behavior planning D_t, the motion planning layer targets to generate a safe and smooth trajectory to realize the high-level decision. Note that since our behavior planning is formulated in a multi-agent setting, it naturally reasons about the future states for *all* the agents. The role of the motion planning boils down to a local trajectory generation problem, as shown in Fig. 6.3.

6.5 Implementation Details

Perception

We purely rely on onboard perception, namely, in real-world experiments, we do not use HD-maps. All the semantics including freespace, lane, and other dynamical objects are sensed in real-time and onboard. All the semantics are organized in a semantic map data structure. Note that all the perception results are subject to noise. From the planning perspective, the upstreaming uncertainties are propagated to the behavior prediction part, and all the uncertainties are reflected by the prediction uncertainty.

Prediction and Decision-making

We only require behavior prediction, which is coupled inside decision-making. We adopt the rule-based lightweight predictor introduced in Chap. 4. The reason why we do not adopt the behavior prediction method in Chap. 3 for onboard experiment is that currently the onboard sensing noise is too large. Large onboard sensing noise will reduce the performance gain of learning-based methods, especially when the noise distribution is different between training set

and onboard sensing noise. If high-end lidars are used in perception, the sensing noise (incl. detection and tracking) can be reduced significantly, and learning-based methods will probably outperform the rule-based predictor. Note that we do not need an independent prediction module. As introduced in Chap. 4, we utilize focused branching to overcome the risk due to uncertainty predictions.

The behavior planning (decision-making) module basically follows the details introduced in Chap. 4. The only difference is that for real-world experiments, other traffic participants are often nosily rational. As such, we introduce responsibility-sensitive safety to multi-agent forward simulation to account for the worse case risk. The working frequency of behavior planning is 20 Hz.

Motion Planning

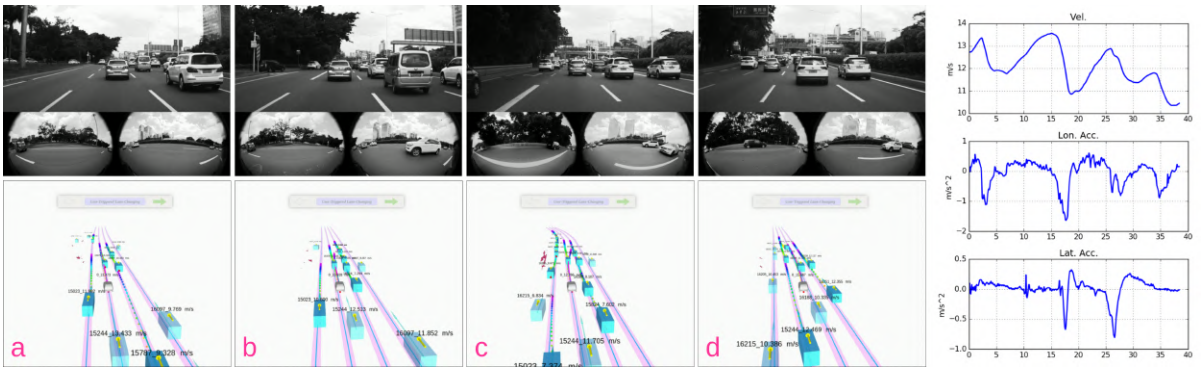
We use the motion planning technique introduced in Chap. 5. We add an additional cost term which represented the proximity of the optimized trajectory to initial seeds. We find this additional cost term beneficial for low-speed driving.

6.6 Field Tests

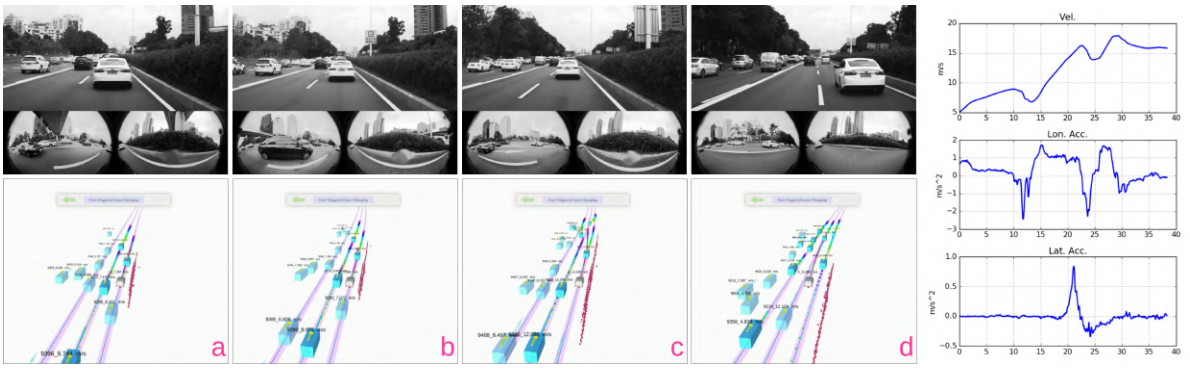
In this section, we provide four representative cases collected from our daily road tests. Note that during road test, we do not use HD-map. All the lanes and other traffic participants are detected and tracked online.

6.6.1 Intelligent gap finding and merging

In Chap. 4, we show that by incorporating advanced driving styles into semantic actions, we can fully exploit the power of semantic actions and achieve intelligent and flexible maneuvers. As shown in Fig. 6.4a (a), the user indicates a lane change requirement by stick, in (b), the controlled vehicle moves forward and automatically finds the appropriate gap to merge, in (c), the controlled vehicle finally finds a appropriate gap and starts merging, and in (d) the controlled vehicle completes the merging. From this example, we observe that by introducing advanced driving knowledge through intelligent driving controllers, the lane-change semantic action can be extremely flexible and automatically adapts to the traffic configuration. Compared to MPDM [75], we fully exploit the maneuverability of the ego vehicle. On the other hand, compared to pure rule-based decision making methods, EPSILON is more general and



(a) Automatic gap finding and merging.



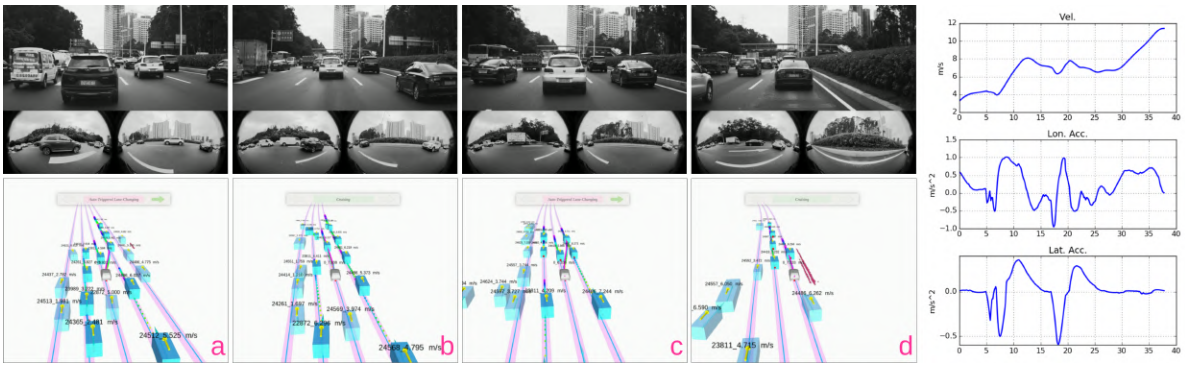
(b) Interaction-aware overtaking.

Figure 6.4: Illustration of two representative scenarios collected from road test. The tracklets are represented by *blue* bounding boxes, the static obstacles are marked in *red* and the controlled vehicle is represented using the model vehicle. In the visualization, we illustrate the forward simulation results both for the controlled vehicle and surrounding vehicles using *rainbow* dots where the color reflects the elapsed time. For a clear visualization, we only draw the forward simulation results for the most likely scenario, and the corresponding intention estimations are marked by *yellow* arrows. In these two cases, the user input a stick signal which requires the planning system to conduct a lane change for the user. EPSILON automatically finds the best time and the best gap to smoothly and safely complete the merging requirement. On the right we plot the dynamic profile (incl. velocity, longitudinal and lateral acceleration) during the process.

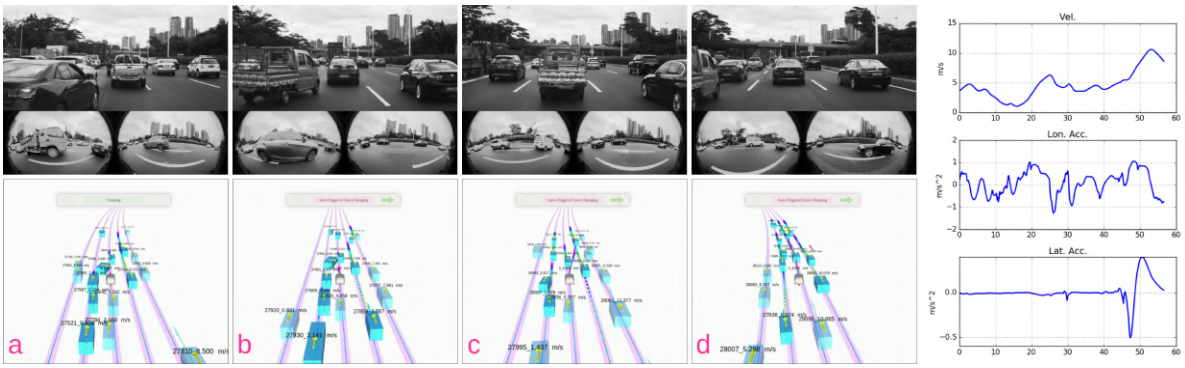
does not rely on complex handcrafted rules. The gap finding and merging is completed naturally through constant replanning.

6.6.2 Interaction-aware overtaking

In Chap. 4 we present the multi-agent forward simulation to consider interaction among agents. As illustrated in Fig. 6.4b, we show how interaction-aware overtaking can be achieved. In Fig. 6.4b (a), the user indicates a lane change requirement. However, in Fig. 6.4b we find that the following vehicle in the neighboring lane is not cooperative. What is worse, it tries to compete with the controlled vehicle and prevent our vehicle from lane changing. Actually, this is a common driving style in the area where the field tests are conducted. In Fig. 6.4b (c), the controlled vehicle further increases the speed and from the dynamic profile, we can



(a) Low-speed automatic lane change in dense traffic.



(b) Cut-in handling and automatic lane change.

Figure 6.5: Illustration of two automatic lane change scenarios dense traffic. The visualization scheme is the same as Fig. 6.4 and the difference is that the lane changes in Fig. 6.4 are upon user requests while here the lane changes are proposed by the planning system.

find that the acceleration is larger compared to the first phase, to increase the gap with respect the following vehicle. This acceleration is due to the active safety mechanism to conduct a longitudinal evasive maneuver so that even in the case of bump-to-rear collision, the controlled vehicle would not be blamed due to it should already complete the lane change at the collision. The controlled vehicle is confident about conducting the acceleration maneuver since there is still room for evasive braking maneuver to quit overtaking. In (d), finally, the following vehicle on the nearby lane gives up and the controlled vehicle finds a safe gap to complete the merge. From this example we find that interaction-aware planning does increase the flexibility of the decision and also avoids over-conservative behaviors. However, it is important to consider interaction in the envelope of safety.

6.6.3 Low-speed automatic lane change in dense traffic

The previous two cases are lane changes upon user requests. And in this case, we show how automatic lane change can be conducted. Automatic lane change is more challenging when in low-speed dense traffic. In Fig. 6.5a (a), the controlled vehicle tries to accelerate and conduct a active merge to the nearby lane which has a higher traveling efficiency. The controlled vehicle

considers the interaction with the following vehicle on the nearby lane and finishes the lane change safely as shown in Fig. 6.5a (b). Then the controlled vehicle finds the nearby lane is more efficient than the current lane again and then proposes another lane change as shown in Fig. 6.5a (c). The controlled vehicle completes the second lane change safely in (d). As we can see from these two consecutive active lane changes, we find that by considering interaction, EPSILON can achieve quite flexible maneuvers in this dense traffic. Actually, there are some interesting points in this case concerning the observation noise. The observed velocity of the following vehicle on the nearby lane is subject to large noise especially when the controlled vehicle is not tangent w.r.t. the lane. Even under this large observation noise, EPSILON can guarantee the safety via choosing evasive maneuvers when necessary and also avoid over-conservative behaviors.

6.6.4 Cut-in handling and automatic lane change

In this case we show how the controlled vehicle handles the aggressive cut-in maneuvers of other traffic participants. In Fig. 6.5b, the front vehicle on the nearby lane conducts an aggressive merge very close to the controlled vehicle. Then the controlled vehicle encounters another cut-in by another truck on the nearby lane in Fig. 6.5b. Both cut-ins are handled smoothly and the stop-and-go process is fluent as we can observe from the dynamic profile. After that, the controlled vehicle decides to overtake the vehicles which cut in previously since it is too congested in the current lane, as shown in Fig. 6.5b (c). The controlled vehicle automatically switches to a more efficient lane as shown in Fig. 6.5b (d), From this case we can observe that the intention estimation and forward simulation work well in dense traffic. Cut-in maneuvers can be quickly identified and handled smoothly. Moreover, the controlled vehicle is not being too conservative. It also actively overtakes other vehicles for its travel efficiency.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Thesis Summary

In this thesis, we present contributions to the state-of-the-art in autonomous navigation of MAVs and AVs in complex environments. We present a full stack of techniques, including motion planning (Chap. 2, Chap. 5), prediction (Chap. 3) and decision-making (Chap. 4). For MAVs, in Chap. 2, we present a kinodynamic replanning framework for quadrotors. While for AVs, in Chap. 3, we present a learning-based behavior prediction method for autonomous vehicles. Based on the behavior prediction technique, in Chap. 4 and Chap. 5, we present efficient decision-making and motion planning techniques for autonomous vehicles. Moreover, we integrate two complete systems for MAV (Chap. 2) and AV (Chap. 6), respectively. The two systems are validated in the physical world through extensive experiments.

Throughout this thesis, we are devoted to answering the question: How to accomplish the user-input task *safely*, *smoothly* and *optimally* with *minimum intervention*? All the methods and system designs are targeting at providing a better solution to this question. In this thesis, we develop methods which are systematically formulated and at the same time have practical impacts. In what follows, we summarize our contributions in this thesis:

- We contribute a kinodynamic replanning framework for quadrotors. This framework overcomes the limitation of traditional shortest path search that the initial path may lead to non-smooth or even dynamically infeasible transitions during replanning. We theoretically analyze the efficiency and optimality of our framework and validate the framework on two real quadrotor platforms.

- We contribute a novel and complete planning system for autonomous vehicles. For building this system, we contribute to the state-of-the-arts from all perspectives including behavior prediction, decision-making and motion planning. The integrated system, namely, EPSILON, is extensively validated in real-world dense city traffic demonstrating promising flexibility and intelligence.

7.2 Lessons Learned and Future Work

In this thesis, several problems related to motion planning, prediction and decision-making are studied. We provide primary solutions and we also note that there is still a long journey to satisfactory autonomous systems (both MAVs and AVs). In what follows, we point out what lessons can be learned from this thesis and what future research directions are promising for building a smarter autonomous system.

7.2.1 Thinking Beyond Path Search (MAV)

An important lesson which can be learnt from Chap. 2 is that the shortest path search is actually problematic for re-planning. For example, in Chap. 2, we show that due to the fact that the shortest path may fail to capture dynamic information, the resulting trajectory may fall into the local minimum where the dynamical feasibility and smoothness can not be guaranteed, especially when given non-static initial states. In Chap. 2, we provide our attempt to develop efficient kinodynamic search algorithms to bring dynamic information into the graph search phase. Generally speaking, our kinodynamic search achieves a good tradeoff between run-time efficiency and optimality, and overcomes the limitations of the traditional shortest path search. To this end, we think the kinodynamic search as the right direction to explore for quadrotor motion planning.

Technically speaking, there are still restrictions for the kinodynamic search. First, the kinodynamic search still targets at solving a "basic" problem, namely, collision avoidance. However, collision avoidance is only the most basic requirement for planning. When considering a slightly more complex task, for example, exploring unknown space to achieve maximum coverage while avoiding the obstacles, kinodynamic search is insufficient. Second, the kinodynamic search itself should be more powerful than what we demonstrate in Chap. 2. One important property of the kinodynamic search is that it directly reasons about time-indexed trajectories. The time information is highly beneficial for avoiding dynamic obstacles or swarm navigation. Given these two observations, we point out two research directions that are worth exploring in the future.

Motion planning for complex applications We observe that kinodynamic search is still restricted to collision avoidance. More efforts should be paid to bridge the gap between motion planning techniques in academia and real-world complex applications. Exploration, rescue, and advanced photography may pose additional requirements to motion planning, such as cost engi-

neering, and additional feasibility constraints. It remains unknown whether the behavior of the kinodynamic search will be affected when adding additional cost terms/constraints. In industry, these complex applications are implemented by extensive tuning. However, we think there should be a systematic and more elegant approach. Improving from path search to kinodynamic search is one little step towards advanced motion planning. And kinodynamic search should contain the potential to accommodate complex real-world applications.

Dynamic obstacle avoidance and swarm navigation Since kinodynamic search provides the time information during the graph search phase, it facilitates dealing with time-critical scenarios. There are two typical time-critical applications for quadrotor navigation, namely, dynamic obstacle avoidance and swarm navigation. The extension of kinodynamic search to these two applications should be beneficial to the community. For dynamic obstacle avoidance, the kinodynamic search itself should be capable of handling the problem without any modification. However, dynamic obstacle avoidance requires a complete system that includes object tracking and prediction. Therefore, the extension to dynamic obstacle avoidance is more like a system contribution instead of an algorithmic contribution. On the other hand, the extension to swarm navigation is non-trivial. Since swarm navigation may require decentralized control and limited communication among quadrotors, it is challenging to design a robust motion planning algorithm for controlling a swarm of quadrotors even based on the kinodynamic search. However, we think the time information of kinodynamic search will be a good starting point for solving this problem.

7.2.2 Uncertainty in Behavior Prediction (AV)

An important lesson which can be learnt from Chap. 3 is that long-term prediction is beneficial for planning but suffers from exploding uncertainty with respect to the prediction horizon. The uncertainty comes from noises from upstreaming modules, multi-modality and noisy labels. The first two sources are straightforward, while the third one is not well discussed in the literature. Noisy labels are actually quite common for behavior prediction, which means that given similar features, the labels can be totally different. In the context of behavior prediction, the problem means that given similar past observations, the future behavior label may be different. This is true since driving behaviors in the real-world are highly diverse and even stochastic. The reason why our interaction-aware behavior prediction can achieve performance gain is that by modeling interaction information the feature space is expanded, thus noisy labels are reduced. By reducing the noisy labels, the uncertainty in prediction is reduced.

It is worth noting that apart from interaction information, there is more information can be incorporated, such as environmental information (traffic signals, road structures, etc). By incorporating more information into prediction, the uncertainty of prediction can be further reduced. Therefore, we need an appropriate input representation for encoding diverse input sources. On the other hand, prediction can never be one hundred percent accurate no matter how much input information is incorporated. It remains a problem that how uncertainty in prediction should be propagated to downstreaming modules and what is the best output representation for quantitatively characterizing the uncertainty. Therefore, we point out two research directions regarding the uncertainty in prediction.

Input representation for behavior prediction By incorporating rich environmental information, prediction accuracy can be enhanced. It is essential to design a flexible and unified input representation to effectively fuse the environmental information. In Chap. 3, we use GRU, NIU and PIU as the encoding structure. However, our design is restricted to highway environments and various environmental information is missing. One straightforward extension is using graph neural network (GNN) for modeling the vehicle-to-vehicle interaction. In the literature, the trend is using rasterized bird-eye-view images and CNN as the encoder. This design can accommodate various semantics in urban driving environments by rendering semantics as color-coded elements on the rasterized image. However, it is questionable whether the rasterized image is the best direction since it is heavyweight during inference and suffers from information loss during rasterization. To this end, we think that input representation is critical for behavior prediction but is still under heavily developed in the current literature. Great improvements can be expected for the input representation.

Output representation for behavior prediction Apart from improving the accuracy of prediction, more attention should be paid to the communication between prediction and planning. A critical problem is that how to represent the uncertainty of prediction so that planning can efficiently understand. In Chap. 3, we adopt a probability representation over a predefined discrete set of behaviors. However, it cannot adapt to complex urban environments. In the literature, the trend is using anchor trajectories as the representation of "behavior". Each anchor trajectory is then sampled and the uncertainty is represented by the parameters of the Gaussian distribution for each time instance. However, this representation introduces the assumption that the uncertainty for each time instance is independent which is not true and often leads to infeasible and inconsistent predictions. Therefore, we think there should better output representation which effectively connects prediction and planning.

7.2.3 Incorporating Domain Knowledge in Decision-making (AV)

An important lesson which can be learnt from Chap. 4 is that domain knowledge is essential in behavior planning. By incorporating domain knowledge (incl. semantic actions, advanced controllers in implementing the semantic actions), advanced driving styles can be introduced into the behavior planning and computation efficiency can be improved. However, only basic domain knowledge is incorporated in Chap. 4 and Chap. 6. For example, we introduce a simple gap finding and velocity adaptation controller for lane changing, which achieves some flexible behaviors. In what follows, we introduce two possible ways of incorporating domain knowledge.

Policy design for forward simulation Following our practice in Chap. 4 and Chap. 6, we can introduce more advanced policies for forward simulation, so that the candidate behaviors can automatically adapt to different traffic configurations. Actually, there has been extensive literature in the policy design, mostly from the field of traffic simulation. We can leverage the knowledge in traffic simulation to help us design lightweight and flexible policies.

Cost and parameter learning for policy evaluation Given various flexible candidate behaviors, it is essential to have an appropriate cost function so that desired behaviors can be obtained. Currently, the weights of cost terms and parameters in the policies are tuned by hand. It is error prone and may fail to generalize to new scenarios. It is notable that another important source of domain knowledge is the human demonstration. We should make good use of human demonstrations so that "data" can accelerate and automate the cost tuning process. Candidate solutions may include inverse reinforcement learning and imitation learning.

7.2.4 Coupling Prediction and Planning (AV)

An important lesson which can be learnt from Chap. 6 is that coupling prediction and planning is beneficial. In most existing works, prediction and planning are decoupled. The mutual influence between the ego vehicle's and other vehicles' future motion cannot be modeled in this way. This issue will cause problems in highly interactive environments. In this thesis, we present our attempt to couple prediction and planning which has shown promising results in highly interactive environments. It is also notable that EPSILON can also be reduced to a decoupled pipeline if the ego decision is excluded in the multi-agent forward simulation. As a result, coupled prediction-planning framework seems more flexible in the sense that it is perfectly compatible with the decoupled one. To this end, we think it necessary to further exploit the

coupled framework design. In particular, we find that conditional prediction is a promising direction to seamlessly convert the existing decoupled pipeline to a coupled one. Conditional prediction means that conditioning the prediction process on a certain future candidate decision. Based on the current learning-based prediction methods in the literature, it is not difficult to extend their frameworks and support the conditional prediction. An additional encoder for the candidate ego decision can do the trick. In this way, we can fully utilize the existing advances of prediction and planning in the literature and switch to the coupled pipeline smoothly.

APPENDIX A

SUPPLEMENTARY MATERIAL FOR KINODYNAMIC SEARCH

A.1 Proof of Prop. 1

The correctness of the proposition follows from the fact that the derivative of the B-spline of degree k is another B-spline of degree $k-1$, which enjoys the convex hull property. Specifically, for the l-th derivative, let $\mathbf{C}_l$ map the basis $\mathbf{b}$ to the derivatives; i.e., $\frac{d\mathbf{b}}{d^l u} = \mathbf{C}_l \mathbf{b}$. It follows that

$$\frac{d\mathbf{s}_j(u)}{d^l u} = \frac{1}{(\Delta_t)^l} \frac{d\mathbf{b}^\intercal}{d^l u} \mathbf{M}_k \mathbf{P}_j = \frac{1}{(\Delta t)^l} \mathbf{b}^\intercal \mathbf{C}_l^\intercal \mathbf{M}_k \mathbf{P}_j. \tag{A.1}$$

Plug in $\mathbf{S}_l = \mathbf{M}_k^{-1} \mathbf{C}_l \mathbf{M}_k / (\Delta_t)^l$. It follows that

$$\frac{d\mathbf{s}_j(u)}{d^l u} = \mathbf{b}^\intercal \mathbf{M}_k (\mathbf{S}_j \mathbf{P}_j), \tag{A.2}$$

where $\mathbf{S}_j \mathbf{P}_j$ is the control point span of the derivative spline. By applying the convex hull property, we complete the proof.

A.2 Relationship between $\mathcal{G}$ and $\mathcal{G}_H$

Lemma 1. *Given the initial state π_s and goal state π_g, let $\pi = (v_0, v_1, \ldots, v_T)$ be an admissible control point placement of Problem 1. The extended sequence $\tilde{\pi}$ uniquely corresponds to an admissible path $\Phi = (\hat{v}_0, \hat{v}_1, \ldots, \hat{v}_Q)$ on the graph $\mathcal{G}_H$, with $\hat{v}_0 = \pi_s$, $\hat{v}_Q = \pi_g$ and $Q = K + T + 2$.*

Lemma 2. *Any admissible path $\Phi = (\hat{v}_0, \hat{v}_1, \ldots, \hat{v}_Q)$ on the graph $\mathcal{G}_H$ which satisfies $\hat{v}_0 = \pi_s$, $\hat{v}_Q = \pi_g$ and $(\hat{v}_{j-1}, \hat{v}_j) \in E_H$ for $j = 1, \ldots, Q$ uniquely corresponds to an admissible control point placement π of Problem 1.*

Proposition 2. *Given a strictly positive cost function $f_{k,\Delta_t} : [\tilde{\pi}]^k \to \mathbb{R}_+$, Problem 1 is equivalent to the shortest path problem on the graph $\mathcal{G}_H$, where the cost is defined on the vertices according to the function $f_{k,\Delta_t}(\cdot)$.*

A.3 Characterization of the inflation for the B-spline-based kinodynamic search

We denote by $\mathcal{C}^{\text{BK}}$ the configuration space in which we conduct the B-spline-based kinodynamic search. The configuration space $\mathcal{C}^{\text{BK}}$ is generated by inflating all the obstacles by δ^{BK} in the workspace. We take a 26-connected 3-D grid with fixed cell size $d_{\text{x}} \times d_{\text{y}} \times d_{\text{z}}$ and a fifth-degree B-spline as an example. There is a finite number of possible span patterns (27^5 in total). The minimum clearance $c_{\min}^{\text{BK}}$ of the configuration space $\mathcal{C}^{\text{BK}}$ can be expressed by the cell size and obstacle inflation δ_{BK} according to $c_{\min}^{\text{BK}} = \min\left(d_{\text{x}}/2 + \delta^{\text{BK}}, d_{\text{y}}/2 + \delta^{\text{BK}}, d_{\text{z}}/2 + \delta^{\text{BK}}\right)$. The problem of finding the sufficient condition such that the overall trajectory is collision free is equivalent to finding the minimum inflation δ^{BK} such that the trajectories for all the B-spline patterns are completely bounded inside the inflated cells. Since the total number of patterns is finite, δ^{BK} can be found by enumerating all the possible span patterns and picking out the one with the largest deviation. The script is available.[1] Note that the process of finding the sufficient inflation is one-time work prior to the planning process. Therefore, it does not affect the EBK search efficiency. Typically, for a 26-connected 3-D grid with fixed cell size $0.16\,m \times 0.16\,m \times 0.16\,m$, the inflation needed is less than $0.03\,m$, which is easy to satisfy in practice.

A.4 Performance analysis of the EBK search

To analyze the performance of the EBK search, we show than the modified INDEX function in Algo. 3 induces another search graph. The characterization of the search graph unveils the complexity of the EBK search. In the following, we give a formal definition of the search graph given by the modified INDEX function.

We begin with the definition of the nodes which are called *virtual nodes*. Specifically, given the encoding level d and encoding index e, we denote by $\mathcal{H}_e^d \coloneqq \{\hat{v} \in V_H | \textsc{Index}(\hat{v}, d) = e\}$ the virtual node aggregating all the vertex tuples which share the same encoding, i.e., the same last d coordinates. It follows that each vertex tuple $\hat{v} \in V_H$ belongs to exactly one virtual node due to the uniqueness induced by the function UNIQUEENCODE$(\cdot)$. For each vertex tuple $\hat{v}_i \in \mathcal{H}_{e_i}^d$, we can obtain the set of neighboring virtual nodes $\{\mathcal{H}_{e_j}^d | e_j = \textsc{Index}(\hat{v}_j, d), \forall (\hat{v}_i, \hat{v}_j) \in E_H\}$. The interesting part is that every vertex tuple of $\mathcal{H}_{e_i}^d$ has exactly the same set of neighboring virtual nodes. We denote by $\mathcal{E}$ the set of edges between virtual nodes, and we denote by $\mathcal{G}_D = (\mathcal{H}^d, \mathcal{E})$

[1]The corresponding script can be found at `https://github.com/WenchaoDing/kinodynamic_replanning.git`.

the *virtual graph* formed by the virtual nodes, where $\mathcal{H}^d$ denotes the set of all virtual nodes.

There are several unique transformations between $\mathcal{G}_D$ and $\mathcal{G}_H$. Given the initial state $\pi_s \in \mathcal{H}^d_{e_s}$ and the goal state $\pi_g \in \mathcal{H}^d_{e_g}$, an *admissible* path $\Theta = (\mathcal{H}^d_{e_0}, \mathcal{H}^d_{e_1}, \ldots, \mathcal{H}^d_{e_Q})$ on the virtual graph $\mathcal{G}_D$ should satisfy $\mathcal{H}^d_{e_0} = \mathcal{H}^d_{e_s}$, $\mathcal{H}^d_{e_Q} = \mathcal{H}^d_{e_g}$ and $(\mathcal{H}^d_{e_{j-1}}, \mathcal{H}^d_{e_j}) \in \mathcal{E}$ for $j = 1, \ldots, Q$. By the construction, any admissible path Φ on $\mathcal{G}_H$ uniquely corresponds to an admissible path Θ on $\mathcal{G}_D$ due to the uniqueness of encoding. Given a known initial state π_s, any admissible path Θ on $\mathcal{G}_D$ also uniquely corresponds to an admissible path Φ on $\mathcal{G}_H$.[2] The reason is that the encoding is based on the last $d > 1$ coordinates, and with the known initial virtual node, the original vertex tuple can be reconstructed.

We define the cost to the virtual node to be $c[\mathcal{H}^d_e] = \min\{c[\pi_s, \hat{v}] | \forall \hat{v} \in \mathcal{H}^d_e\}$, where $c[\pi_s, \hat{v}]$ is the minimum cost from the start vertex tuple π_s to $\hat{v}$ according to Eq. 2.5. The EBK search cannot reach the exact goal state π_g, and instead it can only reach the virtual node $\mathcal{H}^d_{e_g}$. In other words, the EBK search can only reach the relaxed goal state. Given the start and goal virtual nodes, the EBK search is complete (finds the optimal admissible path if one exists) with respect to the aggregated graph $\mathcal{G}_D$, as stated below:

Theorem 2. *Given the graph $\mathcal{G}_D = (\mathcal{H}^d, \mathcal{E})$, the start virtual node $\mathcal{H}^d_{e_s} \in \mathcal{H}^d$ and the goal virtual node $\mathcal{H}^d_{e_g} \in \mathcal{H}^d$, the EBK search finds the optimal admissible path $\Theta = (\mathcal{H}^d_{e_s}, \mathcal{H}^d_{e_1}, \ldots, \mathcal{H}^d_{e_g})$ on $\mathcal{G}_D$ if one exists.*

Proof. We prove by induction and contradiction. The proof follows a similar reasoning process to proving the correctness of Dijkstra's algorithm [44]. And the difference is that for one virtual node, there is one vertex tuple picked out to associate with the virtual node, and the association will be updated during the search process. For the expanded virtual node, the association will be fixed and is supposed to yield the minimum cost to the virtual node among all the aggregated vertex tuples.

Suppose the following hypothesis holds: for each expanded virtual node $\mathcal{H}^d_v$, $c[\mathcal{H}^d_v]$ is the lowest cost from the source virtual node to $\mathcal{H}^d_v$; and for unexpanded virtual node $\mathcal{H}^d_u$, $c[\mathcal{H}^d_u]$ is the lowest cost from the source virtual node to $\mathcal{H}^d_u$ via expanded virtual nodes only. The base case is that there is just the initial virtual node, and the hypothesis holds obviously.

Assume there are $n-1$ expanded virtual nodes, and the hypothesis holds, in which case, the lowest costs to the $n-1$ virtual nodes are known, and given the source virtual node, the vertex tuple associations for the $n-1$ virtual nodes are fixed accordingly. We choose $\mathcal{H}^d_v$ from the

[2]Without π_s, the uniqueness no longer holds.

$n-1$ nodes such that it has the least $c[\mathcal{H}_u^d] = c[\mathcal{H}_v^d] + f[\mathcal{H}_u^d]$ while satisfying $(\mathcal{H}_v^d, \mathcal{H}_u^d) \in \mathcal{E}_d$, where $f[\mathcal{H}_u^d]$ is the cost of the virtual node.

First, $c[\mathcal{H}_u^d]$ should be the shortest path from the source node to $\mathcal{H}_u^d$. Since if there is a shorter path reaching $\mathcal{H}_u^d$ via the nodes other than the $n-1$ expanded nodes, and $\mathcal{H}_w^d$ is the first unexpanded node on that path, it follows that $c[\mathcal{H}_w^d] > c[\mathcal{H}_u^d]$, which yields a contradiction.

Second, for any of the remaining unvisited nodes $\mathcal{H}_w^d$, $c[\mathcal{H}_w^d]$ should still be the shortest path via the expanded nodes. Since if a lower cost of $c[\mathcal{H}_w^d]$ is found by adding $\mathcal{H}_u^d$ to the expanded nodes, Line 17 of Algo. 1 will have updated it. Note that the update of $c[\mathcal{H}_w^d]$ will update the association with $\mathcal{H}_w^d$, so the cost and association are consistent. □

A.5 Proof of Theorem 1

The correctness of the theorem follows from the convex hull property of the B-spline. For brevity, we only consider one control point span which consists of $k+1$ control points, but can be generalized to a long control point sequence without any difficulty. The original tube of one control point span consists of $k+1$ balls, and the connectivity is already guaranteed by the two-level inflation scheme. As such, there are k intersection areas for the sequence of $k+1$ control points. Note that here we only consider the intersection between the two balls associated with two neighboring control points. In the extreme case, we add k control points to each of the intersection areas, and the overall control point sequence consists of $k+1+k^2$ control points, i.e., k^2+1 control point spans. By applying the convex hull property to each of the spans, the trajectory for each control point span is bounded inside one of the balls. As such, the iterative process can succeed in k^2 iterations if no violation of the dynamical feasibility is reported.

PUBLICATION LIST

Journal Papers

1. **Wenchao Ding**, Lu Zhang, Jing Chen, and Shaojie Shen, "EPSILON: An Efficient Planning System for Automated Vehicles in Highly Interactive Environments," to be submitted to *IEEE Transactions on Robotics (TRO)*, 2020.

2. **Wenchao Ding**, Wenliang Gao, Kaixuan Wang, and Shaojie Shen, "An Efficient B-spline-Based Kinodynamic Replanning Framework for Quadrotors," in *IEEE Transactions on Robotics (TRO)*, 2019.

3. **Wenchao Ding***, Lu Zhang*, Jing Chen, and Shaojie Shen, "Safe Trajectory Generation For Complex Urban Environments Using Spatio-temporal Semantic Corridor," in *IEEE Robotics and Automation Letters (RA-L)*, 2019. *Equal Contribution.

4. Wenliang Gao, Kaixuan Wang, **Wenchao Ding***, Fei Gao, Tong Qin, and Shaojie Shen, "Autonomous Aerial Robot Using Dual-fisheye Cameras," in *Journal of Field Robotics (JFR)*, 2020. ***Corresponding Author.**

5. Liu An, Vincent Lau, **Wenchao Ding**, and Edmund Yeh, "Mixed-Timescale Online PHY Caching for Dual-Mode MIMO Cooperative Networks," in *IEEE Transactions on Wireless Communications*, 2018.

Conference Papers

6. Lu Zhang*, **Wenchao Ding***, Jing Chen, and Shaojie Shen, "Efficient Uncertainty-aware Decision-making for Autonomous Vehicles Using Guided Branching," in *International Conference on Robotics and Automation (ICRA)*, 2020. ***Equal Contribution.**

7. Haoran Song, **Wenchao Ding**, Yuxuan Chen, Shaojie Shen, Michael Yu Wang, and Qifeng Chen, "PiP: Planning-informed Trajectory Prediction for Autonomous Driving," in *European Conference on Computer Vision (ECCV)*, 2020.

8. Jieru Zhao, Tingyuan Liang, Liang Feng, **Wenchao Ding**, Sharad Sinha, Wei Zhang and Shaojie Shen, "FP-Stereo: Hardware-Efficient Stereo Vision for Embedded Applications," in *IEEE International Conference on Field Programmable Logic and Applications (FPL)*, 2020.

9. **Wenchao Ding**, Jing Chen, and Shaojie Shen, "Predicting Vehicle Behaviors Over an Extended Horizon Using Behavior Interaction Network," in *International Conference on Robotics and Automation (ICRA)*, 2019.

10. **Wenchao Ding**, and Shaojie Shen, "Online Vehicle Trajectory Prediction using Policy Anticipation Network and Optimization-based Context Reasoning," in *International Conference on Robotics and Automation (ICRA)*, 2019.

11. **Wenchao Ding**, Wenliang Gao, Kaixuan Wang, and Shaojie Shen, "Trajectory Replanning for Quadrotors Using Kinodynamic Search and Elastic Optimization," in *International Conference on Robotics and Automation (ICRA)*, 2018.

12. Kaixuan Wang, **Wenchao Ding**, and Shaojie Shen, "Quadtree-accelerated Real-time Monocular Dense Mapping," in *International Conference on Intelligent Robots and Systems (IROS)*, 2018.

13. Liu An, Vincent Lau, **Wenchao Ding**, and Edmund Yeh, "Mixed Timescale Online PHY Caching and Content Delivery for Content-Centric Wireless Networks," in *IEEE Global Communications Conference (GLOBECOM)*, 2017.

REFERENCES

[1] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 961–971.

[2] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[3] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," *arXiv preprint arXiv:1708.06374*, 2017.

[4] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model mobil for car-following models," *Transportation Research Record*, vol. 1999, no. 1, pp. 86–94, 2007.

[5] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, 2011, pp. 2520–2525.

[6] F. Gao and S. Shen, "Online quadrotor trajectory generation and autonomous navigation on point clouds," in *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2016, pp. 139–146.

[7] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Intl. J. Robot. Research*. Springer, 2016, pp. 649–666.

[8] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-D complex environments," *IEEE Robotics and Automation Letters*, vol. 2, 2017.

[9] J. Chen, T. Liu, and S. Shen, "Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.* IEEE, 2016, pp. 1476–1483.

[10] A. Houenou, P. Bonnifait, V. Cherfaoui, and W. Yao, "Vehicle trajectory prediction based on motion model and maneuver recognition," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 4363–4369.

[11] H. M. Mandalia and M. D. D. Salvucci, "Using support vector machines for lane-change detection," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 49, no. 22. SAGE Publications Sage CA: Los Angeles, CA, 2005, pp. 1965–1969.

[12] J. Schlechtriemen, A. Wedel, J. Hillenbrand, G. Breuel, and K.-D. Kuhnert, "A lane change detection approach using feature ranking with maximized predictive power," in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*. IEEE, 2014, pp. 108–114.

[13] H. Woo, Y. Ji, H. Kono, Y. Tamura, Y. Kuroda, T. Sugano, Y. Yamamoto, A. Yamashita, and H. Asama, "Lane-change detection based on vehicle-trajectory prediction," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1109–1116, 2017.

[14] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *Intl. J. Robot. Research*, vol. 20, no. 5, pp. 378–400, 2001.

[15] D. J. Webb and J. van den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, 2013, pp. 5054–5061.

[16] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, 2015, pp. 3067–3074.

[17] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Intl. J. Robot. Research*, pp. 846–894, 2011.

[18] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *Intl. J. Robot. Research*, pp. 883–921, 2015.

[19] Y. Kuwata, J. Teo, S. Karaman, G. Fiore, E. Frazzoli, and J. How, "Motion planning in complex environments using closed-loop prediction," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008, p. 7166.

[20] C. Xie, J. van den Berg, S. Patil, and P. Abbeel, "Toward asymptotically optimal motion planning for kinodynamic systems using a two-point boundary value problem solver," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, 2015.

[21] Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," *Intl. J. Robot. Research*, vol. 35, no. 5, pp. 528–564, 2016.

[22] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, 2017.

[23] W. Ding, W. Gao, K. Wang, and S. Shen, "Trajectory replanning for quadrotors using kinodynamic search and elastic optimization," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.* IEEE, 2018, pp. 7595–7602.

[24] J. Van Den Berg, D. Wilkie, S. J. Guy, M. Niethammer, and D. Manocha, "LQG-obstacles: Feedback control with collision avoidance for mobile robots with motion and sensing uncertainty," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.* IEEE, 2012, pp. 346–353.

[25] D. Zhou and M. Schwager, "Vector field following for quadrotors using differential flatness," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.* IEEE, 2014, pp. 6567–6572.

[26] D. Bareiss, J. Van Den Berg, and K. K. Leang, "Stochastic automatic collision avoidance for tele-operated unmanned aerial vehicles," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.* IEEE, 2015, pp. 4818–4825.

[27] S. Liu, K. Mohta, N. Atanasov, and V. Kumar, "Search-based motion planning for aggressive flight in SE (3)," *arXiv preprint arXiv:1710.02748*, 2017.

[28] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *Intl. J. Robot. Research*, vol. 28, 2009.

[29] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev, "Multi-heuristic A*," *Intl. J. Robot. Research*, pp. 224–243, 2016.

[30] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," *Proc. of Robot.: Sci. and Syst.*, vol. 104, p. 2, 2010.

[31] R. E. Allen and M. Pavone, "A real-time framework for kinodynamic planning with application to quadrotor obstacle avoidance," Ph.D. dissertation, Stanford University, 2016.

[32] M. Pivtoraiko, D. Mellinger, and V. Kumar, "Incremental micro-UAV motion replanning for exploring unknown environments," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.* IEEE, 2013, pp. 2452–2458.

[33] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, "Continuous-time trajectory optimization for online UAV replanning," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, 2016, pp. 5332–5339.

[34] R. Deits and R. Tedrake, "Efficient mixed-integer planning for UAVs in cluttered environments," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, 2015, pp. 42–49.

[35] S. K. Kannan, W. M. Sisson, D. A. Ginsberg, J. C. Derenick, X. C. Ding, T. A. Frewen, and H. Sane, "Close proximity obstacle avoidance using sampling-based planners," in *AHS Specialists' Meeting on Unmanned Rotorcraft and Network-Centric Operations*, 2013.

[36] J. Chen and S. Shen, "Improving octree-based occupancy maps using environment sparsity with application to aerial robot navigation," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.* IEEE, 2017, pp. 3656–3663.

[37] K. Qin, "General matrix representations for b-splines," *The Visual Computer*, vol. 16, no. 3, pp. 177–186, 2000.

[38] K. Yang and S. Sukkarieh, "An analytical continuous-curvature path-smoothing algorithm," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 561–568, 2010.

[39] L. Yang, D. Song, J. Xiao, J. Han, L. Yang, and Y. Cao, "Generation of dynamically feasible and collision free trajectory by applying six-order Bezier curve and local optimal reshaping," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.* IEEE, 2015, pp. 643–648.

[40] W. Ding, L. Zhang, J. Chen, and S. Shen, "Safe trajectory generation for complex urban environments using spatio-temporal semantic corridor," *IEEE Robot. and Autom. Letters*, 2019.

[41] F. Gao, Y. Lin, and S. Shen, "Gradient-based online quadrotor safe trajectory planning in 3d complex environments," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, 2017.

[42] V. Usenko, L. von Stumberg, A. Pangercic, and D. Cremers, "Real-time trajectory replanning for mavs using uniform b-splines and 3d circular buffer," *arXiv preprint arXiv:1703.01416*, 2017.

[43] E. Verriest and F. Lewis, "On the linear quadratic minimum-time problem," *IEEE Transactions on Automatic Control*, vol. 36, no. 7, pp. 859–863, 1991.

[44] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

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

[46] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Malaysia; Pearson Education Limited,, 2016.

[47] M. Kleinbort, O. Salzman, and D. Halperin, "Collision detection or nearest-neighbor search? on the computational bottleneck in sampling-based motion planning," *arXiv preprint arXiv:1607.04800*, 2016.

[48] D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, and D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientific Belmont, MA, 1995, vol. 1, no. 2.

[49] T. H. Cormen, *Introduction to Algorithms*. MIT press, 2009.

[50] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.* IEEE, 1993, pp. 802–807.

[51] Z. Zhu, E. Schmerling, and M. Pavone, "A convex optimization approach to smooth trajectories for motion planning with car-like robots," in *Proc. of the IEEE Control and Decision Conf.*, 2015, pp. 835–842.

[52] T. Qin, P. Li, and S. Shen, "VINS-mono: A robust and versatile monocular visual-inertial state estimator," *arXiv preprint arXiv:1708.03852*, 2017.

[53] K. Wang, W. Ding, and S. Shen, "Quadtree-accelerated real-time monocular dense mapping," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.* IEEE, 2018.

[54] W. Gao and S. Shen, "Dual-fisheye omnidirectional stereo," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.* IEEE, 2017, pp. 6715–6722.

[55] F. Gao, W. Wu, J. Pan, B. Zhou, and S. Shen, "Optimal time allocation for quadrotor trajectory generation," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.* IEEE, 2018.

[56] S. G. Johnson, *The NLopt nonlinear-optimization package*, 2011. [Online]. Available: http://ab-initio.mit.edu/nlopt

[57] Y. Hu, W. Zhan, and M. Tomizuka, "Probabilistic prediction of vehicle semantic intention and motion," *arXiv preprint arXiv:1804.03629*, 2018.

[58] N. Deo and M. M. Trivedi, "Convolutional social pooling for vehicle trajectory prediction," *arXiv preprint arXiv:1805.06771*, 2018.

[59] N. Deo, A. Rangesh, and M. M. Trivedi, "How would surround vehicles move? A unified framework for maneuver classification and motion prediction," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 2, pp. 129–140, 2018.

[60] W. Ding and S. Shen, "Online vehicle trajectory prediction using policy anticipation network and optimization-based context reasoning," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.* IEEE, 2019.

[61] J. A. Nelder and R. J. Baker, "Generalized linear models," *Encyclopedia of Statistical Sciences*, vol. 4, 2004.

[62] S. Shalev-Shwartz, N. Ben-Zrihem, A. Cohen, and A. Shashua, "Long-term planning by short-term prediction," *arXiv preprint arXiv:1602.01580*, 2016.

[63] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Gaussian process dynamical models for human motion," *IEEE transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 283–298, 2008.

[64] J. Quiñonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate gaussian process regression," *Journal of Machine Learning Research*, vol. 6, no. Dec, pp. 1939–1959, 2005.

[65] T. Gindele, S. Brechtel, and R. Dillmann, "Learning driver behavior models from traffic observations for decision making and planning," *IEEE Intelligent Transportation Systems Magazine*, vol. 7, no. 1, pp. 69–79, 2015.

[66] S. Lefèvre, C. Laugier, and J. Ibañez-Guzmán, "Evaluating risk at road intersections by detecting conflicting intentions," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 4841–4846.

[67] S. Park, B. Kim, C. M. Kang, C. C. Chung, and J. W. Choi, "Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture," *arXiv preprint arXiv:1802.06338*, 2018.

[68] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker, "Desire: Distant future prediction in dynamic scenes with interacting agents," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 336–345.

[69] H. Zou, H. Su, S. Song, and J. Zhu, "Understanding human behaviors in crowds by imitating the decision-making process," *arXiv preprint arXiv:1801.08391*, 2018.

[70] A. Zyner, S. Worrall, J. Ward, and E. Nebot, "Long short term memory for driver intent prediction," in *Intelligent Vehicles Symposium (IV), 2017 IEEE*. IEEE, 2017, pp. 1484–1489.

[71] H. Q. Dang, J. Fürnkranz, A. Biedermann, and M. Hoepfl, "Time-to-lane-change prediction with deep learning," in *Intelligent Transportation Systems (ITSC), 2017 IEEE 20th International Conference on*. IEEE, 2017, pp. 1–7.

[72] F. Bartoli, G. Lisanti, L. Ballan, and A. Del Bimbo, "Context-aware trajectory prediction," *arXiv preprint arXiv:1705.02503*, 2017.

[73] D. Varshneya and G. Srinivasaraghavan, "Human trajectory prediction using spatially aware deep attention models," *arXiv preprint arXiv:1705.09436*, 2017.

[74] H. Xue, D. Q. Huynh, and M. Reynolds, "SS-LSTM: A hierarchical LSTM model for pedestrian trajectory prediction," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2018, pp. 1186–1194.

[75] A. G. Cunningham, E. Galceran, R. M. Eustice, and E. Olson, "MPDM: Multipolicy decision-making in dynamic, uncertain environments for autonomous driving," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.* IEEE, 2015.

[76] E. Galceran, A. G. Cunningham, R. M. Eustice, and E. Olson, "Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction." in *Proc. of Robot.: Sci. and Syst.*, 2015.

[77] U. D. of Transportation Intelligent Transportation Systems Joint Program Office (JPO), "Next generation simulation (ngsim) vehicle trajectories and supporting data," [Online] Available: https://www.its.dot.gov/data/.

[78] N. Deo and M. M. Trivedi, "Convolutional social pooling," [Online] Available: https://github.com/nachiket92/conv-social-pooling.

[79] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[80] T. Toledo and D. Zohar, "Modeling duration of lane changes," *Transportation Research Record*, vol. 1999, no. 1, pp. 71–78, 2007.

[81] R. D. Smallwood and E. J. Sondik, "The optimal control of partially observable markov processes over a finite horizon," *Operations research*, vol. 21, no. 5, pp. 1071–1088, 1973.

[82] O. Madani, S. Hanks, and A. Condon, "On the undecidability of probabilistic planning and infinite-horizon partially observable markov decision problems," in *AAAI/IAAI*, 1999, pp. 541–548.

[83] S. Ross, J. Pineau, S. Paquet, and B. Chaib-Draa, "Online planning algorithms for pomdps," *Journal of Artificial Intelligence Research*, vol. 32, pp. 663–704, 2008.

[84] D. Silver and J. Veness, "Monte-carlo planning in large pomdps," in *Advances in neural information processing systems*, 2010, pp. 2164–2172.

[85] N. Ye, A. Somani, D. Hsu, and W. S. Lee, "Despot: Online pomdp planning with regularization," *Journal of Artificial Intelligence Research*, vol. 58, pp. 231–266, 2017.

[86] P. Cai, Y. Luo, D. Hsu, and W. S. Lee, "Hyp-despot: A hybrid parallel algorithm for online planning under uncertainty," *Proc. of Robot.: Sci. and Syst.*, 2018.

[87] H. Kurniawati and V. Yadav, "An online pomdp solver for uncertainty planning in dynamic environment," in *Robotics Research*. Springer, 2016, pp. 611–629.

[88] H. Bai, D. Hsu, and W. S. Lee, "Integrated perception and planning in the continuous space: A pomdp approach," *Intl. J. Robot. Research*, vol. 33, no. 9, pp. 1288–1302, 2014.

[89] W. Liu, S.-W. Kim, S. Pendleton, and M. H. Ang, "Situation-aware decision making for autonomous driving on urban road using online pomdp," in *IEEE Intl. Veh. Sym.* IEEE, 2015, pp. 1126–1133.

[90] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, "Intention-aware online pomdp planning for autonomous driving in a crowd," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.* IEEE, 2015, pp. 454–460.

[91] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic decision-making under uncertainty for autonomous driving using continuous pomdps," in *Proc. of the Intl. Conf. on Intel. Trans. Syst.* IEEE, 2014, pp. 392–399.

[92] C. Hubmann, J. Schulz, M. Becker, D. Althoff, and C. Stiller, "Automated driving in uncertain environments: Planning with interaction and uncertain maneuver prediction," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 1, pp. 5–17, 2018.

[93] C. Hubmann, J. Schulz, G. Xu, D. Althoff, and C. Stiller, "A belief state planner for interactive merge maneuvers in congested traffic," in *Proc. of the Intl. Conf. on Intel. Trans. Syst.* IEEE, 2018, pp. 1617–1624.

[94] E. Galceran, A. G. Cunningham, R. M. Eustice, and E. Olson, "Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction: Theory and experiment," *Autonomous Robots*, 2017.

[95] D. Mehta, G. Ferrer, and E. Olson, "Fast discovery of influential outcomes for risk-aware mpdm," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.* IEEE, 2017, pp. 6210–6216.

[96] ——, "Backprop-mpdm: Faster risk-aware policy evaluation through efficient gradient optimization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1740–1746.

[97] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, 2018.

[98] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2015.

[99] M. Werling, S. Kammel, J. Ziegler, and L. Gröll, "Optimal trajectories for time-critical street scenarios using discretized terminal manifolds," *Intl. J. Robot. Research*, 2012.

[100] M. McNaughton, C. Urmson, J. M. Dolan, and J.-W. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.* IEEE, 2011.

[101] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller *et al.*, "Making bertha drive: An autonomous journey on a historic route," *IEEE Intelligent transportation systems magazine*, vol. 6, no. 2, pp. 8–20, 2014.

[102] J. Wei, J. M. Snider, T. Gu, J. M. Dolan, and B. Litkouhi, "A behavioral planning framework for autonomous driving," in *IEEE Intl. Veh. Sym.* IEEE, 2014, pp. 458–464.

[103] W. Ding, J. Chen, and S. Shen, "Predicting vehicle behaviors over an extended horizon using behavior interaction network," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.* IEEE, 2019.

[104] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[105] D. Silver and J. Veness, "Monte-carlo planning in large pomdps," in *Advances in neural information processing systems*, 2010, pp. 2164–2172.

[106] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical review E*, vol. 62, no. 2, p. 1805, 2000.

[107] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, Tech. Rep., 1992.

[108] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," *arXiv preprint arXiv:1708.06374*, 2017.

[109] Z. Ajanovic, B. Lacevic, B. Shyrokau, M. Stolz, and M. Horn, "Search-based optimal motion planning for automated driving," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.* IEEE, 2018.

[110] T. Gu, J. Atwood, C. Dong, J. M. Dolan, and J.-W. Lee, "Tunable and stable real-time trajectory planning for urban autonomous driving," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 250–256.

[111] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for bertha: A local, continuous method," in *IEEE Intl. Veh. Sym.* IEEE, 2014.

[112] W. Xu, J. Wei, J. M. Dolan, H. Zhao, and H. Zha, "A real-time motion planner with trajectory optimization for autonomous vehicles," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.* IEEE, 2012.

[113] J. Ziegler and C. Stiller, "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios," IEEE, 2009.

[114] M. Rufli and R. Siegwart, "On the design of deformable input- / state-lattice graphs," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.* IEEE, 2010.

[115] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *Intl. J. Robot. Research*, 2009.

[116] M. T. Wolf and J. W. Burdick, "Artificial potential functions for highway driving with collision avoidance," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.* IEEE, 2008.

[117] C. Hubmann, M. Aeberhard, and C. Stiller, "A generic driving strategy for urban environments," in *Proc. of the Intl. Conf. on Intel. Trans. Syst.* IEEE, 2016.

[118] Z. Zhu, E. Schmerling, and M. Pavone, "A convex optimization approach to smooth trajectories for motion planning with car-like robots," in *2015 IEEE Conference on Decision and Control*. IEEE, 2015, pp. 835 – 842.

[119] S. M. Erlien, S. Fujita, and J. C. Gerdes, "Safe driving envelopes for shared control of ground vehicles," in *7th IFAC Symposium on Advances in Automotive Control*. Elsevier, 2013, pp. 831–836.

[120] C. Liu, C.-Y. Lin, and M. Tomizuka, "The convex feasible set algorithm for real time optimization in motion planning," *SIAM Journal on Control and Optimization*, 2018.

[121] C. Hubmann, M. Becker, D. Althoff, D. Lenz, and C. Stiller, "Decision making for autonomous driving considering interaction and uncertain prediction of surrounding vehicles," in *IEEE Intl. Veh. Sym.* IEEE, 2017.

[122] J. Chen, C. Tang, L. Xin, S. E. Li, and M. Tomizuka, "Continuous decision making for on-road autonomous driving under uncertain and interactive environments," in *IEEE Intl. Veh. Sym.* IEEE, 2018, pp. 1651–1658.

[123] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *Intl. J. Robot. Research*, 2010.

[124] H. Fan, F. Zhu, C. Liu, L. Zhang, L. Zhuang, D. Li, W. Zhu, J. Hu, H. Li, and Q. Kong, "Baidu apollo em motion planner," *arXiv preprint arXiv:1807.08048*, 2018.

[125] F. Gao, W. Wu, Y. Lin, and S. Shen, "Online safe trajectory generation for quadrotors using fast marching method and bernstein basis polynomial," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.* IEEE, 2018.

[126] W. Ding, J. Chen, and S. Shen, "Predicting vehicle behaviors over an extended horizon using behavior interaction network," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.* IEEE, 2019.

[127] C. Liu, W. Zhan, and M. Tomizuka, "Speed profile planning in dynamic environments via temporal optimization," in *IEEE Intl. Veh. Sym.* IEEE, 2017.

[128] T. Gu, J. Snider, J. M. Dolan, and J.-w. Lee, "Focused trajectory planning for autonomous on-road driving," in *IEEE Intl. Veh. Sym.* IEEE, 2010.

[129] W. Zhan, J. Chen, C.-Y. Chan, C. Liu, and M. Tomizuka, "Spatially-partitioned environmental representation and planning architecture for on-road autonomous driving," in *IEEE Intl. Veh. Sym.* IEEE, 2017.

[130] 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.

[131] 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.

[132] M. Naumann, L. Sun, W. Zhan, and M. Tomizuka, "Analyzing the suitability of cost functions for explaining and imitating human driving behavior based on inverse reinforcement learning," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.* IEEE, 2020.

[133] H. Kurniawati and V. Yadav, "An online pomdp solver for uncertainty planning in dynamic environment," in *Robotics Research*. Springer, 2016, pp. 611–629.

[134] L. Zhang, W. Ding, J. Chen, and S. Shen, "Efficient uncertainty-aware decision-making for automated driving using guided branching," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.* IEEE, 2020.