Safe Intention-Aware Maneuvering of Autonomous Vehicles

by

Xin "Cyrus" Huang

B.S., Harvey Mudd College (2016)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2019

Author ..... Signature redacted
Department of Aeronautics and Astronautics
June 1, 2019

Certified by ..... Signature redacted
Brian C. Williams
Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by ..... Signature redacted
Sertac Karaman
Associate Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

Safe Intention-Aware Maneuvering of Autonomous Vehicles

by

Xin "Cyrus" Huang

Submitted to the Department of Aeronautics and Astronautics
on June 1, 2019, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

In order to improve driving performance, while achieving safety in a dynamic environment, it is crucial for a vehicle motion planner to be aware of the intentions of the surrounding agent vehicles. Many existing approaches that ignore the intentions of surrounding vehicles would produce risky or over-conservative plans. In this thesis, we describe a maneuver motion planning system that achieves both safety and efficiency by estimating the types of surrounding drivers and the vehicle motions being executed by them over a finite horizon in the future.

Our claim is that a vehicle is able to efficiently and safely navigate in dynamic traffic situations by estimating the possible types of drivers in its immediate vicinity, such as aggressive or careful, and by predicting their likely maneuvers and motions as probability distributions. To perform these predictions, we first employ a vehicle model that incorporates different driving styles, possible types of maneuvers, the likely trajectories that these maneuvers produce, and the likelihood of transitioning between successive maneuvers. The vehicle models are combined and encoded as hybrid partially observable Markov decision processes (POMDPs) whose discrete elements represent driving styles and maneuvers for each style and whose continuous parts represent vehicle motions.

We then frame the problem of recognizing a vehicle's current driving style and maneuver as a belief state update on the hybrid POMDP. The driving style is assessed using multinomial logistic regression classification, while the maneuver is estimated using Bayesian filtering over a variant of probabilistic hybrid automata and a library of pre-learned motion primitive models. Multinomial logistic regression classification allows us to predict driving styles probabilistically using multiple driving features, and Bayesian filtering provides robust estimation results based on the prior information. Given the recognition results and the learned motion primitive models, we provide probabilistically sound predictions of the future maneuver and trajectory sequence of each agent vehicle. Finally, we compute safe motion plans of the ego vehicle in light of recognized agent vehicle driver styles, intended maneuvers, and future vehicle trajectories, by performing risk-bounded planning on the hybrid POMDP model.

We demonstrate our system in a number of challenging simulated environments, including unprotected intersection left turns and lane changes with multiple dynamic vehicles. The demonstration shows that our intent recognition algorithms achieve an average driving style estimation accuracy of 89.89%, an average maneuver estimation accuracy of 98.9%, and an average trajectory prediction error of 2.12 meters. Furthermore, our maneuver planning system guarantees safety with respect to the safety constraint, while arriving at the goal 13.71% faster compared to a state-of-the-art planner without the intent recognition capability.

Thesis Supervisor: Brian C. Williams
Title: Professor of Aeronautics and Astronautics

Acknowledgments

This thesis would not have been possible without the advice, guidance, insights, and advice of my research mentor, Dr. Andreas Hofmann, and my research advisor, Professor Brian Williams. They have provided me with an opportunity to work on an exciting research topic and tremendous supports throughout the development of this thesis.

I am grateful to people at MERS for their support and friendship during my days at MIT. It is a pleasure to work in such a supportive environment, where people help each other. In particular, I would like to thank Tiago Vaquero and Ashkan Jasour for their mentorship, Matt Deyo and Sungkweon Hong who worked with me on the Geordi project, and those who provided comments that greatly improved my thesis: Jingkai Chen, Yuening Zhang, Nick Pascucci, Marlyse Reeves, and Eric Timmons.

I must thank my family, especially my parents Xu Huang and Qingping Zou, for their mental and financial support for the past two and a half years. I am thankful to all my friends who have encouraged me when my spirits were low and inspired me to work on challenging and exciting problems.

Finally, I would like to thank and acknowledge Toyota Research Institute for sponsoring this work.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

This thesis presents an approach to enable safe and efficient maneuver planning for an autonomous vehicle navigating in tight traffic situations, by recognizing the intentions of the other agent vehicles in its proximity consisting of their driving styles, maneuvers, and future trajectories.

In this chapter, we discuss the motivations for this thesis in Section 1.1 and present a short problem formulation in Section 1.2. We then provide an outline of the approach in Section 1.3, followed by a list of thesis contributions in Section 1.4 and an outline for the rest of this thesis in Section 1.5.

1.1 Motivation

Driving in dynamic scenes, such as intersections and busy streets, is stressful because drivers need to operate their vehicles safely among other vehicles with uncertain motions. A study [10] shows that 40% of crashes that occurred in the United States in 2008 were intersection-related crashes. Among these crashes, inadequate surveillance, false assumptions about the actions of other drivers, and misjudgment of the distance gap or others' speed are responsible for 44.1%, 8.4%, and 5.5% of crashes, respectively. Based on these statistics, we claim that it is crucial to recognize the probabilistic intentions of agent vehicles driving around our ego vehicle in order to improve driving experience, while ensuring safety in dynamic environments. In this thesis, we refer

intention as the possible future motions of surrounding agent vehicles, which include their likely maneuvers and trajectories, which are usually not directly observable. The intentions are recognized probabilistically in order to deal with the uncertainties from vehicle motions. The term *maneuver* refers to a high-level action that the driver chooses to operate a vehicle, such as going forward, turning left, and so on. The terms *ego vehicle* and *agent vehicle* refer to the vehicle that is controlled by our driving executive and the vehicle that is controlled by a third-party driver, respectively.

Since the first DARPA grand challenge in 2004, there has been significant research progress toward the development of motion planning techniques [38] that can be used in intelligent vehicle systems to either assist the driver or take over control completely. Many motion planners work only with static obstacles or assume simple models for the motions of dynamic obstacles, such as moving at a constant velocity towards a fixed direction. These assumptions can lead to risky or conservative plans. For instance, in order to find safe motion plans for an ego vehicle that is controlled by an intelligent system, the authors in [27] assume that each surrounding agent vehicle chooses its maneuver according to a predetermined probability distribution, and propose a method to produce risk-bounded conditional maneuver plans for the ego vehicle that provide an upper bound on the probability of near collisions with other surrounding agent vehicles. The limitation of this work is an unrealistic assumption that the agent vehicle will always choose its maneuver according to fixed likelihoods. To motivate the significance of this limitation, we show two examples that can be seen in common driving experience in Figure 1-1.

In the first case, our autonomous ego vehicle is driving in the left lane behind a slow agent vehicle driving in the right lane. It is a fair assumption that the agent vehicle is most likely to keep driving in the same lane. Hence the planner can generate a collision-free plan that tells the ego vehicle to go forward in order to pass the slow agent vehicle. However, the same assumption fails in the second scenario, where there are two agent vehicles in the right lane. Hence there is a high likelihood that the following agent vehicle will make a lane change to the left, if it is going about 5 miles per hour faster than the leading vehicle. In this case, a crash will likely

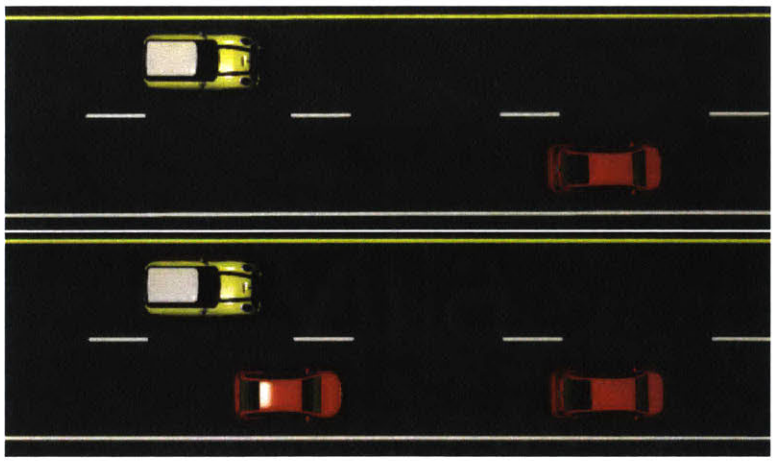


Figure 1-1: Two motivating scenarios, where the ego vehicle in yellow aims to go forward without colliding with the agent vehicles in red. Motion planners that fail to predict the likely maneuvers of the agent vehicles or assume a fixed maneuver distribution will produce results that are either risky or conservative. For instance, if all the agent vehicles are assumed to always go forward, the ego vehicle in the bottom scenario is likely to crash into the follower agent vehicle if the follower agent vehicle decides to make a lane change to pass the leading agent vehicle. On the other hand, an assumption that all agent vehicles will be more likely to make a lane change leads to a conservative plan in the top scenario.

happen if our motion planner asks the ego vehicle to pass the agent vehicles. On the other hand, if we assume that the agent vehicle is always more likely to make a lane change to the left, the planner will slow down the ego vehicle in the second scenario in order to avoid the crash, which causes the ego vehicle to act too conservatively in the first scenario and fail to make the pass. Therefore, in order to generate efficient and safe motion plans, it is critical for the ego vehicle to dynamically infer the future intention of each surrounding agent vehicle in real time, which consists of their driving styles, maneuvers, and trajectories. In this example, a classification of driving styles for the follower vehicle will help predict the likelihood that it will decide to pass. Furthermore, a maneuver estimation such that the follower agent vehicle will likely pass the leading agent vehicle will allow the ego vehicle to more accurately predict the motion of the follower vehicle.

1.2 Problem Formulation

This thesis focuses on three major problems: learning models of vehicle maneuvers, recognizing and predicting agent vehicle's driving style, maneuver sequence, and motion, and generating safe and efficient maneuver plans for an ego vehicle. By *efficient* we mean that the driving executive can find plans for our ego vehicle such that it can navigate to the goal quickly. By *safe* we mean that the plans can guarantee that the probability of near collisions with other agent vehicles is bounded by a threshold. An overview of how three problems relate to each other is depicted in Figure 1-2.

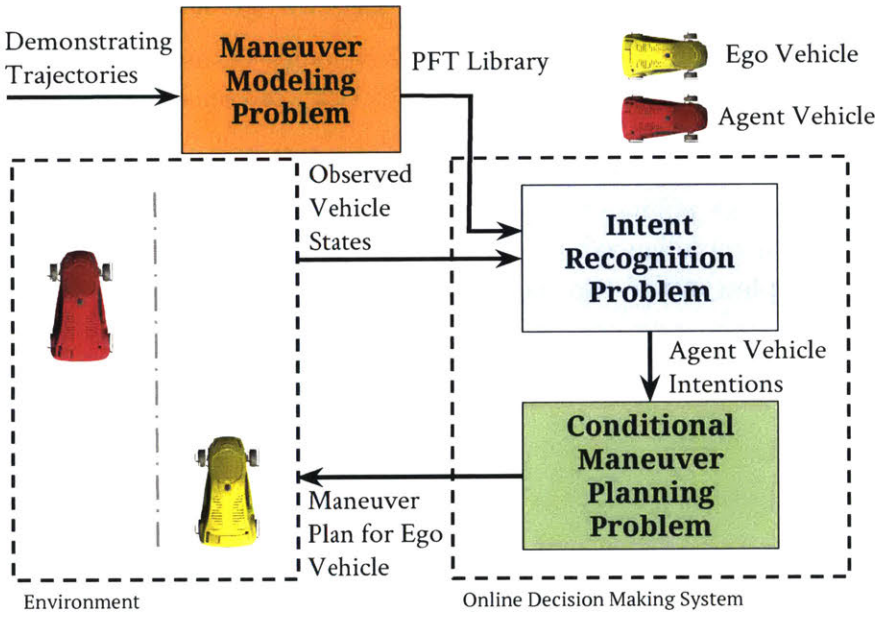


Figure 1-2: Architecture diagram of learning, recognition, and planning problems.

Vehicle maneuver modeling refers to the problem of generating motion models for high-level vehicle maneuvers, such as lane changes or turns, that can be executed by a driver. The input is a set of recorded user demonstration trajectories for each maneuver and driving style, where each trajectory is a sequence of continuous vehicle positions. The output is a library of motion models for each maneuver and driving style that capture the uncertainties of the vehicle motions at different stages. The learned library is stored for future use in intent recognition problem and maneuver planning problem.

Intent recognition refers to the problem of estimating and predicting vehicle driving styles, maneuvers, and trajectories. The input is a sequence of observed vehicle positions of a target vehicle and a library of vehicle models composed of maneuver models learned offline, and the output is a joint probability distribution on the vehicle intentions, consisting of the current maneuvers and driving styles, as well as a sequence of future maneuvers and trajectories of the agent vehicles.

Maneuver planning refers to the problem of generating a sequence of maneuvers that satisfy the optimality requirement under certain risk constraints, where optimality means that the ego vehicle spends the minimum time moving to the goal location, and risk is defined as the probability of near collisions with obstacles such as agent vehicles. The input to the problem is a set of maneuver options that the ego vehicle can choose from and their associated maneuver models, as well as the intentions of the agent vehicles from the output of the intent recognition problem. The output is a risk-bounded maneuver plan, conditioned on the intentions of other agent vehicles, for the ego vehicle to take to the goal location in the least amount of time.

1.3 Approach Overview

In this section, we present a brief overview of our intention-aware autonomous driving executive, *iGeordi*, that is comprised of a module for maneuver model learning, a module for intent recognition, consisting of estimation and prediction of maneuvers and trajectories, for agent vehicles, and a module for maneuver planning for the ego vehicle. Each module is used to solve a specific problem described in Section 1.2. For a detailed discussion of each module, please refer to Chapters 5 through Chapter 7.

1.3.1 iGeordi in Action

We start by showing how iGeordi can solve problems of maneuver model learning, intent recognition, and maneuver planning in the examples given in Figure 1-1. First, iGeordi generates a library of motion models for each maneuver, such as going forward, merging left, and merging right, and each driving style, such as aggressive and normal,

in a lane-change scenario, by collecting a number of demonstration trajectories for each maneuver from offline driving data.

Second, the driving executive accurately recognizes the intentions of the agent vehicles in real time, consisting of driving styles, maneuvers, and motions by observing their past trajectories. For instance, after observing that the follower vehicle approaches the leading vehicle with a large velocity, iGeordi estimates an aggressive driving style for the follower vehicle with high probability. Given the estimated driving style, iGeordi also infers that the follower vehicle tends to make a lane change since it is getting very close to the leading vehicle. Additionally, iGeordi predicts future maneuver and trajectory sequences for the follower vehicle and the leading vehicle to compute possible near collision risks given maneuver estimation results and learned vehicle motion models.

Third, iGeordi generates risk-bounded maneuver plans for the ego vehicle based on recognized intentions. For instance, iGeordi would ask the ego vehicle to go forward after recognizing that the agent vehicle does not intend to change its lane. On the other hand, the executive would slow down the ego vehicle in the bottom scenario to avoid near collisions after estimating that the following agent vehicle is aggressive and is likely to pass the leading agent vehicle. Overall, the online intention-aware planning capability allows iGeordi to make adjustments in real-time when the driving environment changes.

1.3.2 Learning Models of Vehicle Maneuvers

In this thesis, we consider the motion of the agent vehicle as a sequence of movements that depends on the high-level intended maneuvers by the driver. In order to accurately estimate and predict maneuvers and trajectories of the agent vehicles, we need to understand uncertainties in the motions of these human-driven vehicles so that our ego vehicle can safely navigate around them. Our solution is to generate a motion representation called Probabilistic Flow Tube (PFT) [13] to represent the motion of each maneuver, by learning from demonstration trajectories that capture that maneuver. We choose a probabilistic representation because some trajectories

are more likely to occur than others, when a human driver executes a particular maneuver. We choose a tube representation because it is able to capture both temporal and spatial aspects of the maneuver motions. Once we generate a library of PFTs for each maneuver, iGeordi's maneuver estimator module can use this library to estimate the likelihood that a particular trajectory is the result of a particular maneuver, such as turning left. Meanwhile, iGeordi's motion predictor can use the PFTs to generate realistic future motions for each agent vehicle with different driving styles to predict its next move.

1.3.3 Recognizing Intentions of Agent Vehicles

It is important to predict the likely motion of the agent vehicles in order to avoid possible near collisions with them. In order to have accurate predictions, we need to first estimate the current intentions of each agent vehicle, consisting of driving styles and maneuver types. As studied in [33], maneuver-based motion prediction approaches can produce reliable longer-term motion predictions with more reasonable computational time, as compared to physics-based approaches and interaction-aware approaches. In addition, an accurate estimation of driving styles would allow us to have better ideas of how the driver would execute a maneuver. For example, an aggressive driver tends to turn more sharply than a normal driver.

Our proposed intent recognition algorithm in iGeordi is divided into three steps: estimating the driving styles adopted by the driver, inferring the high-level maneuvers at the current step, and predicting a sequence of hybrid states for the agent vehicle, consisting of discrete maneuvers and continuous trajectories, over a finite horizon. The driving style estimation step allows us to select appropriate vehicle models used for maneuver estimation and motion prediction steps, the maneuver estimation step allows us to determine the current high-level action of the agent vehicle, and the hybrid prediction step allows us to update the probability distribution over the agent vehicle's future actions and compute near collision risk to identify risky plans.

1.3.4 Intention-Aware Maneuver Planning

Once we estimate and predict the intentions of the agent vehicles, we first update the belief state in a hybrid partially observable Markov decision process (POMDP) model whose discrete elements represent driving styles and maneuvers and whose continuous elements represent vehicle motions. The updated model captures the latest intentions of the agent vehicles, which can be used by our driving executive to generate safe maenvuer plans for our ego vehicle. Then, iGeordi finds the safe and efficient maneuver plans through a heuristic forward search algorithm called risk-bounded AO* (RAO*) [43]. RAO* outputs optimal plans with guarantees on the risk, which is defined as the probability of near collisions with other dynamic agent vehicles.

As a result, the driving executive finds safe and efficient plans for the ego vehicle navigating in tight traffic situations. Safety is achieved by estimating and predicting driving styles, maneuvers, and trajectories of the agent vehicles accurately with the help of a probabilistic motion representation that models the maneuver motions with uncertainties. Efficiency is achieved through a risk-bounded heuristic forward search algorithm that finds an optimal plan, while taking into consideration the risk of near collisions with other vehicles.

1.4 Thesis Contributions

In this thesis, we make three major contributions as follows:

1. We present a fast and accurate system for recognizing agent vehicle's intention, in terms of its estimated driving style and maneuver, as well as predicted maneuver sequence and trajectory. The system achieves an average driving style estimation accuracy of 89.89%, an average maneuver estimation accuracy of 98.9%, and an average trajectory prediction error of 2.12 meters over a horizon of 4.8 seconds in a simulated lane change test.

2. We present an autonomous driving executive that generates efficient and safe

plans by considering the intentions of surrounding agent vehicles. The safety and efficiency are achieved by using a heuristic forward search algorithm that finds solutions with minimum costs, while providing guarantees on the probability of near collisions with other agent vehicles. The efficiency in terms of average completion time to the goal is improved by 13.28% using our executive compared to a state-of-the-art risk-bounded planner without the intent recognition capability.

3. We show that our planner can produce plans that echo different driving styles in a number of challenging dynamic environments, by adjusting the upper bound probability of the risk constraints in the driver model.

1.5 Thesis Outline

The rest of the thesis is organized as follows: We first establish the background knowledge and present our thesis in a nutshell in Chapter 1, followed by a series of related work in Chapter 2. Next, we introduce a formal problem definition in Chapter 3, followed by an overview of our technical approach in Chapter 4. Each of the following three chapters elaborates on the design of the three primary modules used in iGeordi. In Chapter 5, we show a method to learn probabilistic vehicle motions using probabilistic flow tubes and show how the learned flow tubes are used for inference tasks. In Chapter 6, we discuss the intent recognition methods in several pieces, consisting of data-driven driving style estimation, discrete maneuver estimation, and probabilistic hybrid vehicle state prediction. In Chapter 7, we finish our intention-aware driving executive by integrating the intent recognition results into a maneuver planning problem and solving the problem with a risk-bounded heuristic forward search algorithm. In Chapter 8, we describe our experimental setup for a number of simulated challenging driving scenes, consisting of unprotected left turns and busy lane changes, and demonstrate the performance of different parts of our approach. In Chapter 9, we end the thesis with a summary of our work and a discussion of future research.

Chapter 2

Related Work

In this chapter, we review state-of-the-art research that is relevant to our work in four areas: motion prediction, vehicle planning, driving style estimation, and motion representation. Motion prediction refers to predicting future trajectories and maneuvers for a vehicle. Vehicle planning refers to motion planning or maneuver planning for autonomous vehicles. Driving style estimation refers to estimating the driving styles adopted by a vehicle. Motion representation refers to representing vehicle maneuvers using specific models. We discuss how related work addresses various aspects of our problem of intention-aware maneuver planning, how it fails to solve the problem completely, and how our approach fills the gap.

2.1 Maneuver-based Motion Prediction

In this section, we focus on reviewing maneuver-based prediction algorithms that predict the future trajectory of a target vehicle by estimating and predicting a sequence of high-level maneuver actions.

Vehicle maneuvers can be estimated through different sources of sensor inputs [16, 33]. The first common source is driver-oriented inputs, which are usually collected by onboard cameras pointing at the driver. For instance, in [17], eye gaze and head dynamics of the driver were collected by a camera to predict a driver's intention to change lanes. Although observing a driver's state can provide reliable clues

for estimating maneuvers, it cannot be extended further to predict the continuous trajectory of the vehicle without knowing extra information, such as vehicle states. Unlike these approaches, we do not assume the availability of driver-oriented sensors in other vehicles. Instead, we focus on recognizing the intentions of other vehicles through vehicle-based sensing.

Vehicle-based sensing measures state information about the target vehicle, such as position, velocity, and yaw angle. Some relevant work includes [50], where the authors use low-cost in-vehicle sensors like GPS, Inertial Measurement Unit (IMU), and odometer to predict latitudinal and longitudinal maneuvers, and [5], where pedal positions and vehicle locations are used to predict continuous lane change and turning actions. These approaches assume access to internal vehicle sensors, which is not realistic in estimating agent vehicles' motions. In our work, we use the vehicle state information that can be collected using external sensors, such as cameras and lidars, making our work more applicable for motion predictions of agent vehicles.

There are many ways to estimate maneuvers [16, 26, 44, 35, 32, 51, 11]. In one relevant work, the authors in [31] developed an object-oriented Bayesian Network that is able to detect 27 different maneuvers, including lane changing and object following. In another relevant work [35], an HMM was constructed to classify driver lane changing and lane keeping maneuvers using vehicle data collected from the Controller Area Network (CAN) bus. The approach was validated with naturalistic driving data and achieved an overall maneuver recognition rate over 90.8%. However, the approach was only able to detect offline maneuvers, and thus not applicable to real time predictions. In this thesis, we are inspired by [31, 35] to use Markov assumptions and Bayesian filtering to produce probability distributions over both discrete maneuvers and continuous trajectories in real time.

The maneuver estimation results are then used to predict continuous trajectories over longer horizons. For example, [26] generates a set of long-term trajectories based on the detected maneuver and selects the one with the minimum cost. In [11], the authors generate a probability distribution over future locations conditioned on the states of the vehicle and the estimated maneuver using deep neural networks. In [44],

a hand-crafted motion model is used to predict future trajectories given the estimated maneuver. Such maneuver-based methods have proved to work well in vehicle motion prediction, which inspires us to use a set of pre-learned motion models to predict the target vehicle's future positions probabilistically based on the estimated maneuvers. Compared to [26], we are able to produce multiple trajectories instead of only one trajectory, which makes our prediction more robust. Compared to [44], we learn our maneuver motion model through demonstration trajectories, instead of using a hand-crafted model. Compared to [11], our method is more data efficient since it generates accurate prediction results without requiring a large amount of training data.

In summary, our maneuver-based motion prediction method is inspired by the state of the art in a number of ways. First, it produces accurate long-term motion prediction results by first estimating maneuvers. Second, it does not rely on driver-oriented nor in-vehicle sensor data, which makes it more applicable for motion estimations of agent vehicles. Third, it predicts future motions efficiently through Markov assumptions and Bayesian filtering.

2.2 Vehicle Planning with Moving Obstacles

Many approaches use search-based methods to find a feasible path to the goal location, by reasoning over explicit trajectories of moving obstacles [41, 4, 1]. However, these methods rely on simple prediction models that are not realistic when planning over long-horizon, where the obstacles, such as other vehicles on the road, can change both high-level goals and low-level paths over time. For instance, [1] assumes that vehicles drive at current velocity and stay in the current lane, which can lead to inaccurate prediction over more than just a few seconds. Additionally, [41, 4] assume that goal states of the obstacles are known a priori, which is not realistic in dynamic scenes. Furthermore, these approaches either fail to consider the uncertainties in motion predictions, or convert the probabilistic predictions into a discretized costmap indicating the likely positions of the obstacles, which can lead to inaccurate approximations.

A widely used framework for maneuver planning with dynamic obstacles is Par-

tially Observable Markov Decision Processes (POMDPs). POMDPs provide a systematic model that incorporates the uncertainty in the environment, including the sensor noise and stochastic future motions of surrounding obstacles. [4, 3, 36] estimate pedestrian's goal locations as intention beliefs in order to predict their future motions. The predicted motions can be used to compute collision risk and generate safe motion plans in a POMDP framework. Such methods work effectively in cases where goal locations such as entrance, exit, and cross walk can be easily identified. However, the goal locations for a vehicle can be hard to obtain without using extra information such as a map. In this work, we focus on recognize intentions as high-level maneuvers as an alternative, since it is easier to evaluate and also provides useful information for future vehicle motions. Instead of estimating goals, [6] estimates future locations of surrounding agent vehicles by assuming a linear motion model, which is limited to only simple scenarios where vehicles always drive at a constant velocity. Instead, our approach uses a richer motion model to incorporate nonlinear motions of vehicles. Additionally, all the above methods solve the POMDP instance using approximation methods, due to the complexity required to find optimal solutions. On the other hand, we use a heuristic forward search algorithm to solve for exact solutions without approximations in order to guarantee the optimality of our solution. Another relevant work [9] provides exact POMDP solutions by decoupling the problem into multiple Markov Decision Processes. However, it assumes the agents have fixed or deterministic intentions, which would lead to conservative or risky outcomes, as discussed in Chapter 1.

Additionally, Bayesian reinforcement learning [54, 23] and deep reinforcement learning [20] are proposed to enable safe and efficient planning in dynamic environments. Unfortunately, these methods are limited because they either work in discrete state spaces or require a large amount of training data and computational resources. On the other hand, our approach is more data efficient, since it requires less training data and thus less computational resources. Additionally, it offers better maneuver plans by planning in continuous space.

Despite the success of POMDP-based approaches, they fail to provide a plan with

a guarantee on the probability of success, such as avoiding near collisions with moving agents in a dynamic environment. An extension called constrained POMDPs has been introduced to model risk explicitly and generate plans with bounded-risks [52, 39]. Using a similar idea, a chance-constrained POMDP (CC-POMDP) has been proposed to include a more systematic definition of execution risk and has been applied to different problem domains [43, 27]. Unfortunately, previous approaches using CC-POMDPs have assumed fixed or deterministic intentions of the dynamic agents [27], which can fail in many realistic driving scenarios, as discussed in Chapter 1. Instead, we improve the chance-constrained model by updating the belief states dynamically with a real time intent recognition system, and solve it by a heuristic search algorithm named Risk-Bounded AO* (RAO*) [43].

In summary, our proposed vehicle planning method is inspired by related work that solves vehicle planning problem in a POMDP framework, where the agent vehicle intentions are modeled as belief states. Our method combines the best aspects from different approaches by recognizing vehicle intentions as belief state update and solving for exact solutions to guarantee optimality with a chance-constrained model that upper bounds the risk of collisions with surrounding dynamic obstacles.

2.3 Driving Style Estimation

Estimating driving styles can be helpful to predict vehicle trajectories, because drivers with different styles will execute the same maneuver in various ways. Similar to Section 2.1, we do not assume the availability of driver-oriented sensors in other vehicles. Instead, we focus on approaches based on vehicle-based sensing.

Therefore, it is more realistic to estimate driving styles through vehicle states that can be collected from external sensors, such as cameras, lidars, and radars. Early methods use fuzzy logic [2, 14, 8] to estimate driving style. Instead of using hand-crafted fuzzification functions, recent supervised learning methods, such as k-nearest neighbor (k-NN) and support vector machines (SVMs) [53, 15], dynamic time warping [30, 19], and decision trees [21], learn a data-driven classification model

using a set of driving features. In this thesis, inspired by the success of supervised learning methods, we choose to infer driving styles using a classification method called multinomial logistic regression classification. Compared to other supervised learning methods, multinomial logistic regression classification is able to produce a probability distribution over possible driving style outcomes.

2.4 Vehicle Maneuver Motion Modeling

In this section, we focus on reviewing relevant methods that learn maneuver-based motion models. Other popular methods include inverse reinforcement learning that learns a reward function independent of vehicle maneuvers, which is discussed with more details in Section 5.3.2.

Maneuver motion representations are useful to estimate vehicle maneuver types and predict future vehicle trajectories. There are several ways to represent vehicle maneuver motions, including poly-lines [26], maneuver-specific models [44], funnels [49], and Gaussian mixture models [11, 13]. Among these maneuver motion representations, we favor probabilistic ones such as Gaussian mixture models since they incorporate uncertainties from human driver behavior and can be used to compute near collision probability in a chance-constrained framework. The difference between [11] and [13] is that the former learns a Gaussian mixture model through a deep neural network, which requires a large amount of training data and computational resources. On the other hand, the work in [13] proposes a more data efficient representation called Probabilistic Flow Tubes (PFT) that models human-like actions in humanoid robotics tasks, by computing common characteristics as Gaussian distributions defined over a time interval from a set of demonstration examples.

In summary, we decide to follow [13] and use PFT to model probabilistic vehicle motions for two predominant reasons. First, it is a compact representation that is parameterized by a sequence of Gaussian distributions defined by means and covariances over different time steps. Second, it learns human behavior through a number of demonstrating examples. In the vehicle domain, the PFT can capture different

driving styles efficiently given enough demonstrating trips. For example, if we collect enough data from two drivers with distinctive driving styles and learn a PFT for each driver, the resulting PFTs would represent different driving styles.

2.5 Summary

In summary, our work in this thesis is motivated by the state of the art work in several ways. First, it estimates vehicle intentions as maneuvers and future trajectories, using Markov assumptions and Bayesian filtering, which enables accurate motion predictions over the long term. Second, it estimates driving styles using a supervised learning method, which utilizes various driving features efficiently. Third, it learns maneuver motions using a probabilistic Gaussian mixture model-based representation, which incorporates uncertainties of different driving behaviors. Fourth, it uses a POMDP framework to enable safe and efficient maneuver planning for our ego vehicle. By combining these techniques together, our work represents an advancement in the state of the art that is critical for safe driving when sharing the road with other vehicles taking stochastic actions.

Chapter 3

Problem Statement

In this chapter, we begin by motivating the key needs that this thesis tries to fulfill with an example scenario. Next, we present the inputs and outputs formally by arguing that they are the right format and introducing each variable involved in the three constituent problems: maneuver model learning, intent recognition, and maneuver planning.

3.1 Importance of Intention-Aware Planning

An illustrated scenario of the intention-aware maneuver planning problem is shown in Figure 3-1, where an ego vehicle in yellow needs to navigate to a goal location at the end of the road as soon as possible, without colliding with the agent vehicles in red. This problem is challenging in two aspects. First, the ego vehicle must understand the intentions and predict the future motions of each agent vehicle in proximity, in order to avoid possible near collisions with them. Due to uncertainties in driver intentions and vehicle dynamics, the intentions and motions are stochastic and thus hard to predict. Our illustrated scenario is such an example. Second, after predicting the stochastic intentions of agent vehicles, we need to find safe maneuver plans such that our ego vehicle can navigate to its goal location with guarantees on the probability of near collisions. In the illustrated scenario, if we predict that the following agent vehicle is likely to change to its left lane, we must find plans for the ego vehicle that

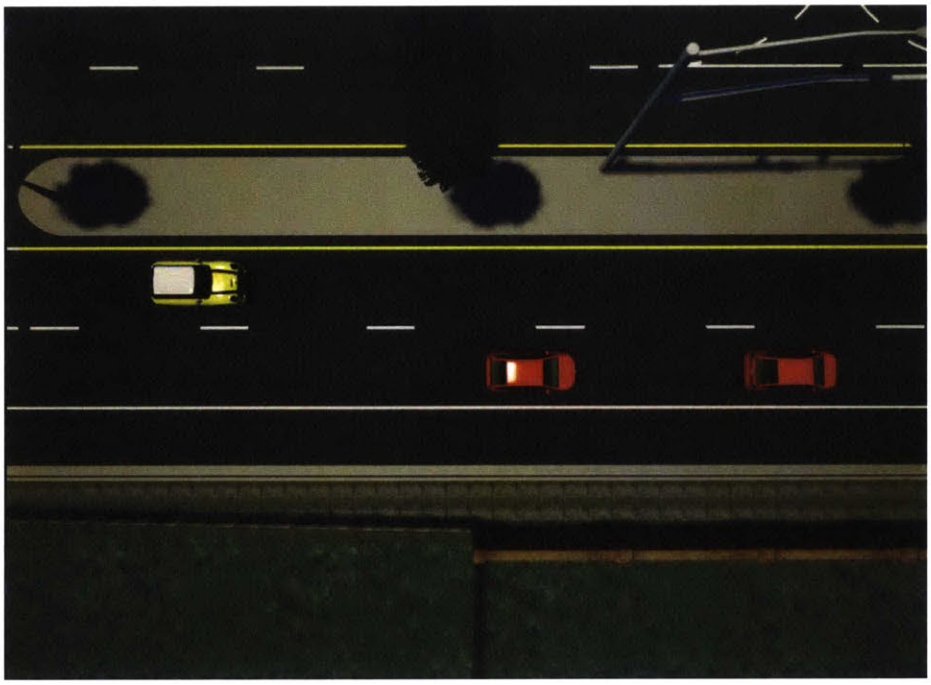


Figure 3-1: Illustrated example of an intention-aware maneuver planning problem. The ego vehicle in yellow needs to navigate to the end of the road as soon as possible, while keeping itself safe by not colliding with either of the agent vehicles in red that can either go forward or change lanes.

lead to the goal as soon as possible, while leaving time to confirm whether the lane change will take place before or after the ego vehicle has passed.

In this thesis, we identify three problems to solve based on the two challenges mentioned above. The first problem is to learn a probabilistic model for each maneuver that incorporates uncertainties from the driver and vehicle dynamics given a set of demonstration trajectories. The second problem is to recognize the intentions of each agent vehicle, including driving styles, maneuvers, and trajectories, given observed trajectories. The third problem is to find a safe and efficient maneuver plan for our ego vehicle, once the intentions of other agent vehicles are recognized accurately.

3.2 Problem Inputs and Outputs

The inputs to intention-aware maneuver planning problem include demonstration vehicle trajectories from offline data, observed vehicle states, and a set of available

actions for our ego car. Demonstration trajectories allow us to pre-learn maneuver models and vehicle models so that we can do online intent recognition. Observed vehicle states are important to determine the intentions of surrounding agent vehicles. Available action set allows us to compute the utility and risk for each option that our ego vehicle can choose, and eventually find the best to take.

The output is a risk-bounded optimal maneuver plan that helps facilitate safe and efficient maneuver planning for our ego vehicle.

In the rest of this chapter, we formally introduce all inputs and outputs used in this thesis, which include inputs and outputs for each subproblem.

3.3 Problem Definition: Intention-Aware Maneuver Planning

Given the input and output variables to our overall intention-aware maneuver planning problem, we present additional input and output variables in each subproblem used in this thesis.

3.3.1 Maneuver Motion Learning

We start by defining the input and output variables for the problem of maneuver motion learning.

A user demonstration trajectory is a sequence of continuous coordinates, which sufficiently capture the continuous vehicle motions. The coordinates are represented in 2D Cartesian space, where we assume that the vehicle is always driving on flat terrains so its height information is ignored. These coordinates are sufficient to represent vehicle motions for different maneuvers and driving styles.

Definition 1. A *user demonstration trajectory* D is a sequence of continuous Cartesian coordinates at discretized time steps $[(x_1, y_1), \ldots, (x_g, y_g)]$ with length g, where positive x points to the north and positive y points to the east in a global coordinate

system. Each trajectory is collected from user demonstrations with a sampling period of T_s.

In order to learn accurate maneuver models, we assign every demonstrated trajectory with a maneuver label, such as turning left, and a driving style label, such as aggressive. As introduced in Chapter 1, maneuvers and driving styles are two important intentions to be recognized for accurate motion prediction.

Definition 2. A *maneuver label* $m \in \mathcal{M}$ is a unique string associated with a vehicle maneuver, where $\mathcal{M}$ is a finite set of maneuver labels.

Definition 3. A *driving style label* $u \in \mathcal{U}$ is a unique string associated with a style adopted by a vehicle driver, where $\mathcal{U}$ is a finite set of driving style labels.

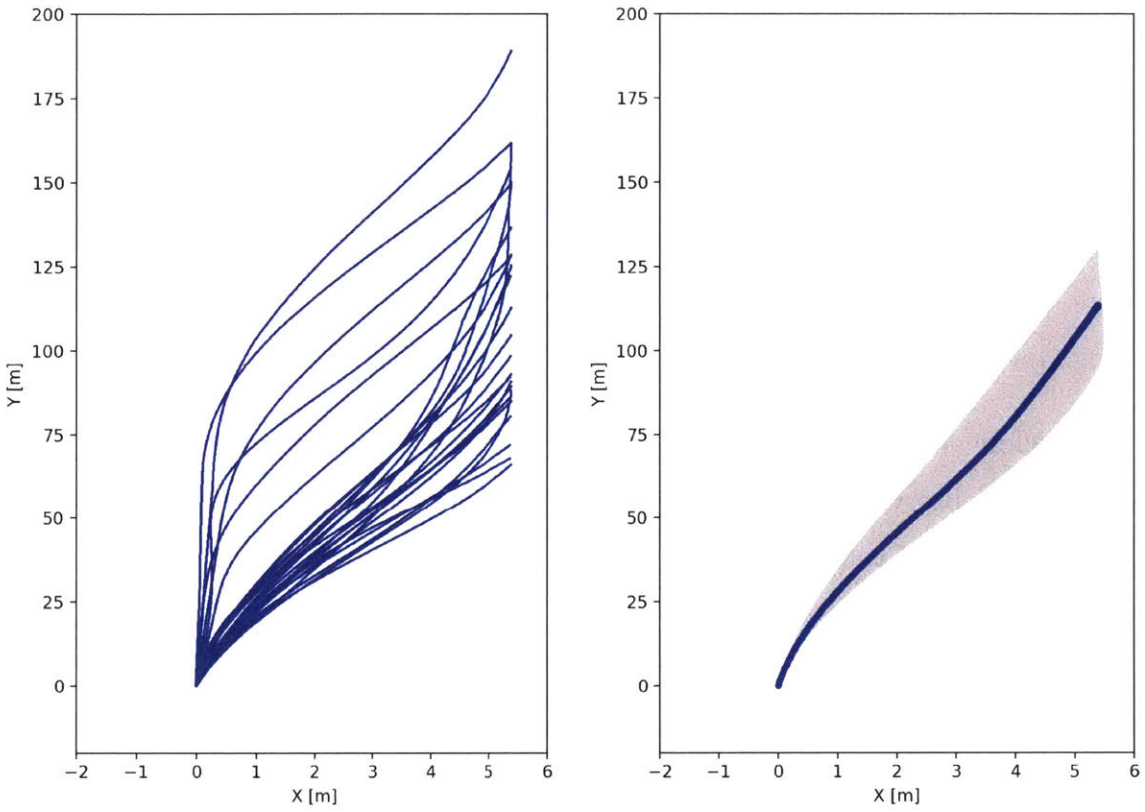
A maneuver model learning problem is typically provided with multiple trials of user demonstrations for that maneuver associated with a driving style.

Definition 4. A set of *training sequences for a particular maneuver type and driving style* $\mathcal{D}_{m,u}$ is the set of demonstrated trajectories $\{D^n_{m,u}\}_{n=1...N}$ with length N for a particular maneuver label m and driving style label u. An example training set is depicted in Figure 3-2a.

In order to learn maneuver models efficiently for multiple combinations of maneuvers and driving styles, we can provide a set of training sequences for all maneuver and driving style options.

Definition 5. A set of *all training sequences* $\mathcal{D}$ is the combined set of training sequences $\{\mathcal{D}_{m,u}\}_{m\in\mathcal{M},u\in\mathcal{U}}$ for all maneuver labels and driving style labels.

A probabilistic flow tube (PFT), formally presented below, is the output of a maneuver learning problem that represents motion models given training sequences. It is a sequence of nominal positions and a sequence of covariances at each time step, which is useful to capture both average behavior of a maneuver and uncertainties at different stages across that maneuver.



(a) Demonstration trajectories.

(b) Learned PFT.

Figure 3-2: Trajectories and the learned probabilistic flow tube of the merging right maneuver.

Definition 6. A *probabilistic flow tube* $L_{m,u}$ is a maneuver model associated with a maneuver label m and a driving style label u. It consists of a tuple (μ, Σ), where $\mu = [\mu_1, \ldots, \mu_g]$ is a sequence of points representing the mean trajectory, and $\Sigma = [\Sigma_1, \ldots, \Sigma_g]$ is a sequence of covariances across time t_i $(1 \leq i \leq g)$. An example PFT is depicted in Figure 3-2b.

A library of probabilistic flow tubes is used to store the maneuver models for all combinations of maneuver labels and driving styles.

Definition 7. A *library of probabilistic flow tubes* $\mathcal{L}$ is the combined set of PFTs $\{L_{m,u}\}_{m \in \mathcal{M}, u \in \mathcal{U}}$ for all maneuver labels and driving style labels.

3.3.2 Intent Recognition

In an intent recognition problem, the input is composed of observed data points by assuming that we observe a set of noise-free sensor data at each time step k. These data points are important for driving style estimation and maneuver estimation, since they represent both spatial and temporal aspects of vehicle motions.

Definition 8. An *observed data point* $\mathbf{o}^{(k)}$ is a set of inputs collected from vehicle sensors, including vehicle position, velocity, acceleration, yaw angle, and distances to driving lanes.

Meanwhile, we store a sequence of observed data points with length w to get a trajectory of observations at time step k. Such sequence is important to capture temporal aspects of vehicle driving.

Definition 9. An *observed data trajectory* $\mathbf{o}_k = [\mathbf{o}^{(k-w+1)}, \ldots, \mathbf{o}^{(k)}]$ is a sequence of sensor data tracked at time step $k - w + 1$ through k with sequence length w.

An observation history is a sequence of all observed data trajectories observed so far up to recognition time step k.

Definition 10. An *observed data history* $\mathbf{o}_{1:k} = [\mathbf{o}_1, \ldots, \mathbf{o}_k]$ includes all observed data sequences from time step 1 to time step k.

After defining the input data to the intent recognition problem, we then define the states being estimated. The clock variable is a non-observable random variable that indicates the progress of a maneuver that is finished.

Definition 11. A *clock variable* $c_k \in \mathbb{R}_{\geq 0}$ is continuous random variable at time step k, indicating how much a maneuver has finished.

The current driving style state is a non-observable random variable that represents possible driving styles at time step k.

Definition 12. A *driving style state* $\mathbf{x}_{U,k} \in \mathcal{U}$ is a random variable at time step k, indicating the driving style adopted by an agent vehicle.

The current maneuver state is a non-observable random variable that represents possible maneuver states, including maneuver types and maneuver clocks, at time step k.

Definition 13. A *maneuver state* $\mathbf{x}_{d,k}$ is a random variable at time step k, including a maneuver label variable $\mathbf{x}_{M,k} \in \mathcal{M}$ and a maneuver clock variable c_k.

The current continuous state is a random variable that represents possible locations of an agent vehicle.

Definition 14. A *continuous state* $\mathbf{x}_{c,k} \in \mathbb{R}^2$ is continuous random variable at time step k.

At current time step k, we will often refer to a future maneuver state and continuous state at some time step $k + r$ as $\mathbf{x}_{d,k+r}$ and $\mathbf{x}_{c,k+r}$, respectively.

In order to estimate all necessary states in a unified framework, we introduce a vehicle model called clocked Probabilistic Hybrid Automaton associated with a vehicle that adopts a certain driving style. The vehicle model consists of a clock variable, discrete and continuous state variables, observation variables, continuous functions governing the continuous state evolution, and transition functions governing the discrete state transition.

Definition 15. A *clocked Probabilistic Hybrid Automaton (tPHA)* [24] represents the vehicle model associated with driving style u and is encoded as a 6-tuple

$$\mathcal{H}_u =< c, \mathbf{x}, \mathbf{o}_M, F, T_d, \mathcal{M} >$$

to estimate and predict the hybrid states for each uncontrollable agent vehicle, where

- c denotes a non-negative continuous clock variable.
- $\mathbf{x} = \{\mathbf{x}_M \cup \mathbf{x}_c\}$ are the state variables, where $\mathbf{x}_M$ denotes the discrete maneuver label with domain $\mathcal{M}$, and $\mathbf{x}_c$ denotes continuous state variables as global positions (x, y) of the agent vehicle.

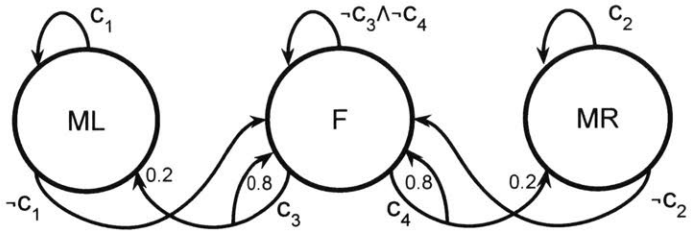


Figure 3-3: An example transition model in a clocked Probabilistic Hybrid Automaton.

- $\mathbf{o}_M$ is the observation variable that includes the observed locations of the agent vehicle $[o_x, o_y]$.

- F specifies the continuous evolution of the automaton, by mapping a maneuver type to a probabilistic flow tube (PFT), as defined in Definition 6. Each PFT is a function that maps the current clock variable and state variable to the next clock variable and state variable, as described in Chapter 5.

- T_d specifies a set of transition functions for each discrete state. Each transition function has an associated guard condition determined by the clock variable and state variables.

  Figure 3-3 shows an example of T_d that indicates the transition functions among three maneuvers (merge left ML, merge right MR, and go forward F). The transition probability is indicated by the number next to the arrow if the condition is satisfied, and is equal to 1.0 unless otherwise specified. Within a merging maneuver, such as ML and MR, the transition is deterministic to the same maneuver if the clock variable is smaller than a threshold c_{max}, which is usually the number of clock cycles included in a maneuver. In this case, c_1 and c_2 are true if $c \leq c_{max}$. Otherwise, the transition is deterministic to the forward maneuver. Within a forward maneuver, the transition depends on both the clock variable and the continuous state of the agent vehicle. If the clock variable is equal to c_{max} and the lateral position of the agent vehicle is larger than the lateral position of the middle of two lanes l_c, which indicates that the agent vehicle is

on the right lane, we assume it is likely to make a lane change to the left, so c_3 is satisfied and the transition goes to ML and F with probabilities 0.2 and 0.8, respectively. Similarly, c_4 is satisfied if the clock variable is equal to c_{max} and the lateral position of the agent vehicle is smaller than l_c. Otherwise, if neither c_3 nor c_4 are satisfied, the transition is deterministic to L.

After obtaining a clocked Probabilistic Hybrid Automaton for each driving style $u \in \mathcal{U}$, we combine them together into an automaton set.

Definition 16. A *set of clocked Probabilistic Hybrid Automata* $t\mathcal{PHA} = \{tPHA_u\}_{u \in \mathcal{U}}$ represents a set of vehicle models with different driving styles.

3.3.3 Risk-Bounded Maneuver Planning

After defining the necessary variables for the problems of maneuver motion learning and intent recognition for an agent vehicle, we next introduce the input and output variables for ego vehicle maneuver planning, starting with a set of available maneuver actions for the ego vehicle.

Definition 17. An *ego maneuver set* $\mathcal{A}$ is a set of high-level maneuver actions that can be taken by our ego vehicle.

For each maneuver action that the ego vehicle can choose from, we introduce a maneuver model that defines its characteristics, which can be used for computing the probability of collisions and determining a subset of actions that are available for planning.

Definition 18. *An ego maneuver model* is a tuple $AM = \langle Pre, Eff, c, Tr \rangle$, where

- *Pre* is a set of preconditions that defines the vehicle states in which the action can be executed.
- *Eff* is a set of effects that define the result of executing the action.
- $C \in \mathbb{R}_{\geq 0}$ is a cost value associated with the action.

- Tr is a sequence of control inputs, such as acceleration and steering angle, associated with the action.

Given the inputs of ego action models, it is necessary to introduce the output as a conditional maneuver plan that achieves the goal of safe and efficient maneuver planning for the ego vehicle, which is formally defined in Definition 19. Since we assume that the motions of agent vehicles in the environment are stochastic, it is necessary to generate a maneuver plan conditioned on their possible motions.

Definition 19. *A conditional maneuver plan* π is a function that produces a sequence of maneuver actions for the ego vehicle conditioned on the belief state over the intentions of surrounding agent vehicles $\mathcal{B}_k$ at time step k.

An optimal maneuver plan is defined in Definition 20, which denotes the best plan out of all possible maneuver actions that achieves the minimum expected cost over a finite horizon.

Definition 20. *An optimal maneuver plan* π^* is a maneuver plan that minimizes the expected cost conditioned on the current belief state. Mathematically, it is defined as:

$$\pi^* = \arg\min_{\pi} \mathbb{E}\left[\sum_{t=k}^{k+H} C(a_t) \middle| \mathcal{B}_k, \pi\right], \tag{3.1}$$

where $C(\cdot)$ is a cost function denoting the cost associated with an action $a_t \in \mathcal{A}$, as described in Definition 18, and H is the planning horizon.

Note that the planning horizon H is different from the predicting horizon h. The predicting horizon refers to the number of *time steps* ahead of the current time step that needs to be predicted, and the planning horizon refers to the number of *actions* the planner needs to look ahead. Usually, the predicting horizon is the product of the planning horizon and the number of clock steps contained in the maneuver. For example, if the planning horizon is 2, and each maneuver takes 25 time steps to finish, the predicting horizon would be 50.

A risk-bounded optimal maneuver plan is defined in Definition 21, which is an optimal plan subject to the risk constraint.

Definition 21. *An risk-bounded optimal maneuver plan* π^*_{rb} is an optimal maneuver plan that satifies the risk constraint defined as:

$$er(\mathcal{B}_k|\pi^*_{rb}) = 1 - \mathbb{P}(\bigwedge_{i=k}^{k+H} \text{Safe}_i|\mathcal{B}_k, \pi^*_{rb}) \leq \Delta, \tag{3.2}$$

where er is the execution risk of a policy π measured from the current belief state $\mathcal{B}_k$ over planning horizon H, Safe_i is an indicator function, which is true when the ego vehicle is near collision free, and Δ is a constant that upper bounds the execution risk. The way to compute er is provided with more details in Chapter 7.

3.4 Summary

In this chapter, we have provided a formal definition of the inputs and outputs appeared in different problems in this thesis, including maneuver model learning, intent recognition, and maneuver planning. In the next chapter, we give a formal problem definition for each problem using the defined inputs and outputs, as well as a series of pseudocodes to solve each problem.

Chapter 4

Technical Architecture

In this chapter, we provide an overview of our technical approach to the intention-ware maneuver planning problem. The major challenge of this problem comes from the stochastic intentions from the agent vehicles that drive around our ego vehicle. These intentions determine the future motions of the agent vehicles, which are necessary to consider if we want to achieve safe maneuver planning without near colliding with these vehicles. Another challenge is to generate a safe and efficient maneuver plan for the ego vehicle once we obtain the intent information. To address these challenges, we introduce an intention-aware driving executive iGeordi that consists of three components. The first component is an offline maneuver model generator that produces probabilistic maneuver models. The second component is an intent recognition module that estimates each agent vehicle's current intentions and future motions. The third component is a risk-bounded conditional maneuver planner that produces risk-bounded optimal maneuver plans. In the rest of this chapter, we provide an architecture diagram showing how our components connect and divide the work thesis work, define each component formally, and provide a series of pseudocodes that explain high-level ideas to solve each component.

4.1 Architecture Diagram of the Intention-Aware Driving Executive

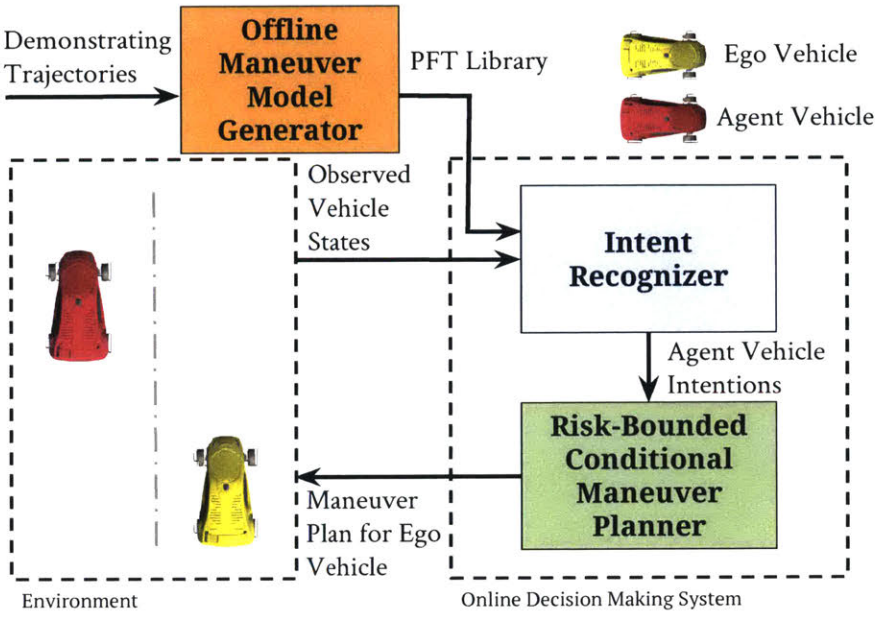


Figure 4-1: Architecture diagram of the Intention-Aware Driving Executive.

The architecture of our driving executive iGeordi is illustrated in Figure 4-1, which interacts with an environment that includes one ego vehicle in yellow and one agent vehicle in red. The driving executive takes the input from offline demonstration trajectories, as well as vehicle states from the environment, and produces maneuver plans for the ego vehicle. The overall architecture consists of three boxes that represent each of the three components in the executive. The first box in orange is a maneuver model generator that produces a library of learned maneuver models called probabilistic flow tubes (PFTs) offline given a set of demonstration vehicle trajectories. These learned PFTs are necessary for maneuver-based motion prediction approaches that produce reliable long-term prediction results. We then feed the PFT library to the next box in blue to produce agent vehicle intentions given its observed states. This box provides accurate intention estimations of the agent vehicles for our intention-aware driving executive. Third, the probability distributions over recognized intentions are fed into

a risk-bounded conditional maneuver planner, as illustrated in the green box, that produces a safe and optimal maneuver plan for our ego vehicle. This is the last and most important step to achieve our thesis goal for intention-aware maneuver planning. In the rest of this chapter, we describe briefly how the components come together by defining each of them formally and introducing our implementations for each component.

4.2 Maneuver Model Generator

In this section, we define the problem for learning maneuver models, and then provide our implementations.

4.2.1 Definition of A Maneuver Model Learning Problem

The problem of learning a library of maneuver models for the agent vehicle from user demonstrations is shown in Definition 22. The goal is to learn a nominal trajectory for each combination of maneuver types and driving styles and its corresponding covariances at different time steps.

Definition 22. Given a set of demonstration trajectories $\mathcal{D}$ with maneuver labels and driving style labels, generate a library of probabilistic flow tubes $\mathcal{L}$ that describes the agent vehicle maneuver motions.

This problem is essential in our driving executive, as the output PFT library can be used in several ways. First, we can use it to estimate the likelihood of vehicle maneuvers by comparing the observed trajectories with the nominal trajectory and uncertainties of each PFT in the library. Second, it can be used to predict future motions once the current driving style, maneuver state, and clock variable are estimated. Third, by modeling the uncertainties of vehicle motions, PFT can be used to compute the probability of near collisions between agent vehicles and our ego vehicle.

4.2.2 Pseudocode for Maneuver Model Generator

Similar to [12], the pseudocode for our maneuver model generator is provided in Algorithm 1, where we generate PFTs by iterating through all maneuver types and driving styles. For each combination of maneuver type and driving style, our algorithm temporally aligns the corresponding demonstration trajectories (Line 6-7), computes means and covariances at each time step (Line 8-9). The PFT, composed of a mean vector and a covariance vector, is added to our PFT library as the final output (Line 10-11). More details are provided in Chapter 5.

Algorithm 1 Pseudocode for Maneuver Model Generator

Input $\mathcal{M}$ (a set of pre-defined maneuvers), $\mathcal{U}$ (a set of pre-defined driving styles), $\mathcal{D}$ (a set of demonstration trajectories)
Output $\mathcal{L}$ (a library of PFTs)

```
1:  procedure GeneratePFTLibrary(M, U, D)
2:      L = []
3:
4:      for each maneuver m ∈ M do
5:          for each driving style u ∈ U do
6:              D_{m,u} = get demonstration trajectories with labels m and u
7:              D'_{m,u} = temporally align all trajectories in D_{m,u}
8:              μ_{m,u} = compute means of D'_{m,u} at each time step
9:              Σ_{m,u} = compute covariances of D'_{m,u} at each time step
10:             L_{m,u} = (μ_{m,u}, Σ_{m,u})
11:             L.append(L_{m,u})
12:
13:     return L
```

4.3 Intent Recognizer

In this section, we define the problem for recognizing intentions, including driving styles, maneuver states, and vehicle positions, and then provide our implementations.

4.3.1 Definition of An Intent Recognition Problem

The problem of recognizing the intention of an agent vehicle based on its observed trajectory and vehicle model is shown in Definition 23. The goal is to accurately

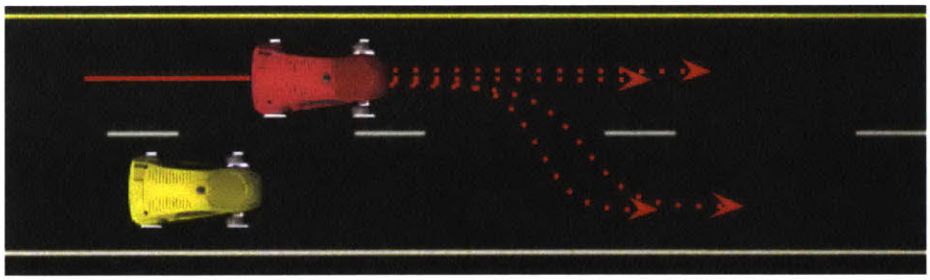


Figure 4-2: A sample intent recognition problem. Given its past trajectory in solid red line, we want to estimate the possible driving styles and maneuvers, as well as predict future motions of the agent vehicle. Some possible predictions are plotted in red dashed lines.

recognize the future motions of an agent vehicle so that our maneuver planning system can avoid near collisions with it.

Definition 23. Given an observed data history $\mathbf{o}_{1:k}$ and a set of vehicle models $t\mathcal{PHA}$, estimate the current driving style state estimate $\mathbf{x}_{U,k}$, the current maneuver state state $\mathbf{x}_{d,k}$, and current continuous position $\mathbf{x}_{c,k}$. Furthermore, generate a probability distribution over the future hybrid motion sequence, including maneuver states $\mathbf{x}_{d,k+1}, \ldots, \mathbf{x}_{d,k+h}$ $(\mathbf{x}_{d,k+1:k+h})$ and locations $\mathbf{x}_{c,k+1}, \ldots, \mathbf{x}_{c,k+h}$ $(\mathbf{x}_{c,k+1:k+h})$ up to horizon h.

The intent recognition problem is divided into three subproblems. The first subproblem is to select a subset of vehicle models that are most relevant to the target agent vehicle, through estimating its current driving style. In the second subproblem, given the selected vehicle models, we want to estimate the maneuver state of a vehicle at the current time step, including the maneuver type and the maneuver clock. Such estimation is helpful to predict the future motions of the vehicle, as the maneuver type indicates what high-level action a driver is likely to choose and the maneuver clock indicates how much of that action has finished. The third subproblem is to predict the hybrid states of a vehicle in the future, which is essential for an intention-aware driving executive, as it helps the executive to detect possible near collisions with other agent vehicles.

A sample problem is illustrated in Figure 4-2, where we want to recognize the intentions of an agent vehicle in red given its observed trajectory in a red solid line. In

Algorithm 2 Pseudocode for Intent Recognizer

Input $\mathbf{o}_{1:k}$ (observed vehicle data history), a set of vehicle models $t\mathcal{PHA}$

Output $p(\mathbf{x}_{d,k})$ (a probability distribution over current maneuver states), $p(\mathbf{x}_{c,k})$ (a probability distribution over current continuous states), $p(\mathbf{x}_{d,k+1:k+h})$ (a probability distribution over future maneuver sequences), $p(\mathbf{x}_{c,k+1:k+h})$ (a probability distribution over future continuous sequences)

1: **procedure** INTENTRECOGNIZER($\mathbf{o}_{1:k}, t\mathcal{PHA}$)
2: $p(\mathbf{x}_{U,k})$ = estimate current driving style state given $\mathbf{o}_{1:k}$
3: $t\mathcal{PHA}'$ = select the corresponding vehicle models from $t\mathcal{PHA}$ based on $p(\mathbf{x}_{U,k})$
4: $p(\mathbf{x}_{M,k}), c_k$ = estimate current clock variable and maneuver label given $p(\mathbf{x}_{U,k})$, $t\mathcal{PHA}'$, and $\mathbf{o}_{1:k}$
5: $p(\mathbf{x}_{d,k})$ = combine $p(\mathbf{x}_{M,k})$ and c_k into a single maneuver state
6: $p(\mathbf{x}_{c,k})$ = observe current vehicle position using $p(\mathbf{x}_{d,k})$ and $\mathbf{o}_{1:k}$
7:
8: **for** each future step i from 1 to h **do**
9: $p(\mathbf{x}_{d,k+i})$ = estimate next maneuver state using previous estimates $p(\mathbf{x}_{d,k+i-1})$, and vehicle models $t\mathcal{PHA}'$
10: $p(\mathbf{x}_{c,k+i})$ = estimate next continuous state using maneuver estimates $p(\mathbf{x}_{d,k+i})$ with non-zero probability, and vehicle models $t\mathcal{PHA}'$
11:
12: **return** $p(\mathbf{x}_{d,k}), p(\mathbf{x}_{c,k}), p(\mathbf{x}_{d,k+1:k+h}), p(\mathbf{x}_{c,k+1:k+h})$

the first subproblem, the objective is to estimate its driving styles, such as aggressive and normal, and select vehicle models weighted by the probabilities of their corresponding driving styles. In the second subproblem, given the selected vehicle models, we estimate the distribution over possible maneuver types, such as going forward and merging left, and the clock variables that indicate the percentage a maneuver has finished. In the third subproblem, the goal is to predict a sequence of maneuvers and trajectories for each estimation result, illustrated as red dashed lines in the sample problem, where each trajectory represents possible future motions of the red vehicle given a driving style, a maneuver type, and a clock variable.

4.3.2 Pseudocode for Intent Recognizer

The pseudocode for our intent recognizer is provided in Algorithm 2, where we first estimate the current driving style of an agent vehicle given observations (Line 2). Second, we select appropriate vehicle models based on the estimated driving styles,

and estimate the maneuver state and continuous state for the agent vehicle (Line 3-6). Third, we predict its future maneuver sequence and motion trajectory iteratively up to horizon h using the information from the previous time step, as well as the selected vehicle models (Line 8-10). More details are described in Chapter 6.

4.4 Risk-Bounded Conditional Maneuver Planner

In this section, we define the problem for risk-bounded maneuver planning, and then provide our implementations.

4.4.1 Definition of An Intention-Aware Risk-Bounded Maneuver Planning Problem

The problem of risk-bounded maneuver planning is provided in Definition 24. The goal is to generate safe maneuver plans for an ego vehicle that navigates to a goal location, while bounding risk of near collisions with other moving agent vehicles.

Definition 24. Given a probability distribution over each agent vehicle's future motions $p(\mathbf{x}_{d,k+1:k+h}, \mathbf{x}_{c,k+1:k+h})$, and a set of action models for each action in $\mathcal{A}$ that the ego vehicle can choose from, find a risk-bounded optimal maneuver plan π^*_{rb} as defined in Definition 21.

The intention-aware maneuver planning problem is the last piece in our driving executive, and its solution is a conditional maneuver plan that our ego vehicle can follow given intent recognition results. In Figure 4-3, we provide a planning scenario where the ego vehicle can choose to go forward or slow down, and the resulting conditional plan has two possible conditions. In the first condition, the agent vehicle is estimated to go forward, and our risk-bounded optimal maneuver plan is to go forward in order to move to the goal on the right as soon as possible without possible near collisions with the agent vehicle. In the second condition, the agent vehicle is estimated to merge to its right lane, and our output plan should slow the ego vehicle down in order to avoid possible near collisions.

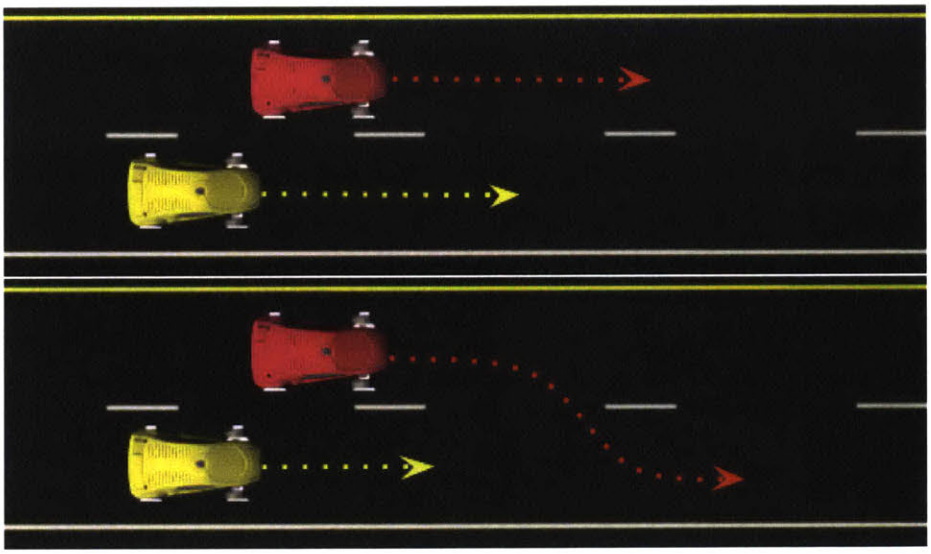


Figure 4-3: Two possible maneuver plans conditioned on different intentions of the agent vehicle. The top figure shows the maneuver plan for the ego vehicle to go forward in yellow dashed line if the agent vehicle is estimated to go forward. The bottom figure shows the maneuver plan to slow down if the agent vehicle is estimated to merge right.

Algorithm 3 Pseudocode for Risk-Bounded Conditional Maneuver Planner

Input Motion predictions $p(\mathbf{x}_{d,k+1:k+h}, \mathbf{x}_{c,k+1:k+h})$, available action set $\mathcal{A}$.
Output Risk-bounded optimal maneuver plan π^*_{rb}.

1: $\mathcal{P}$ = generate a maneuver planning model using $p(\mathbf{x}_{d,k+1:k+h}, \mathbf{x}_{c,k+1:k+h})$ and a set of action models for each action in $\mathcal{A}$
2: $\mathcal{B}_k$ = obtain current belief state over agent vehicle intentions from $\mathcal{P}$
3:
4: Initialize search graph $\mathcal{G}$ and policy π using $\mathcal{P}$, $\mathcal{A}$, and $\mathcal{B}_k$
5: Expand search graph and update policy iteratively until policy becomes optimal
6: π^*_{rb} = obtain the resulting optimal policy
7:
8: **return** π^*_{rb}

4.4.2 Pseudocode for Risk-Bounded Conditional Maneuver Planner

The pseudocode for our risk-bounded conditional planner is provided in Algorithm 3, in which we first generate a problem model containing all updated information such as agent vehicle intentions and the possible actions that the ego vehicle can choose (line 1-2), and then solve the problem model using a heuristic forward search algorithm

called RAO* (line 4-6). More details of the problem model generating and the solver can be found in Chapter 7.

4.5 Summary

In this chapter, we begin by reviewing our thesis problem and its associated challenges. We then provide an overview of our technical architecture, followed by a series of definitions and high level pseudocode algorithms to describe each component. More details of our technical approaches can be found in the next three chapters.

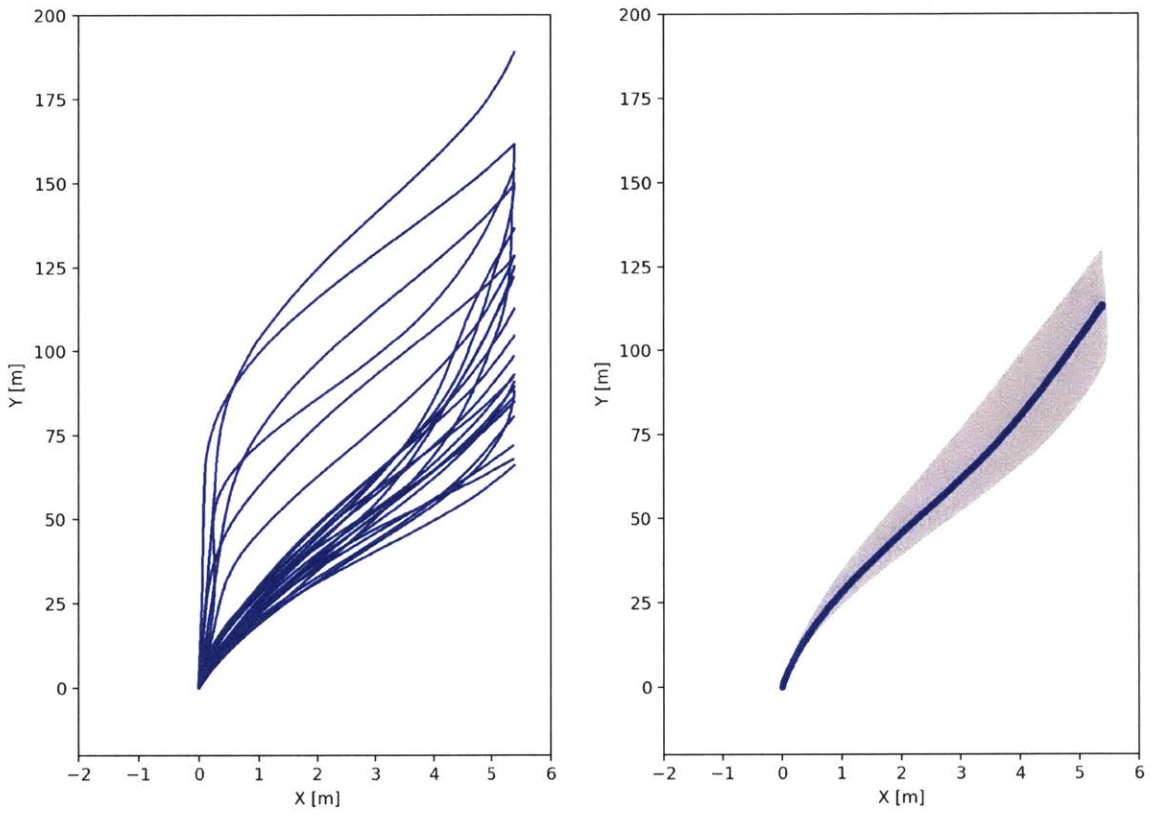
Chapter 5

Learning Models of Vehicle Maneuvers

In this chapter, we review Probabilistic Flow Tubes (PFTs) for learning motion models, first presented in [13], and introduce its applications to vehicle maneuver learning. The learned maneuver model is useful to estimate the maneuver executed by the vehicle and predict its future motions. As discussed in Chapter 2, we decide to use PFTs as our maneuver model because it has multiple advantages. First, a PFT is time dependent, which means it models the change of motions at different time steps during the evolution of a maneuver. Second, since there are many ways to execute the same maneuver and some are more likely than others, our learned model captures the probabilistic nature of the maneuver by representing the vehicle location as a Gaussian distribution at each discretized time step. For example, given many trajectories that can be executed to merge to the right lane as shown in Figure 5-1a, our PFT is able to represent their uncertainties as Gaussian distributions in Figure 5-1b. In addition, we generate multiple PFTs for the same maneuver, which help distinguish between different driving styles that can execute that maneuver. Each driving style represents a regional preference (e.g., Boston, Detroit) as well as attention and aggressiveness. It is important to learn different driving styles from human driver data, since we are trying to predict human driver behavior.

In the rest of this section, we first show how we learn a library of PFTs for

different combinations of maneuver types and driver styles given demonstration trajectories. Second, we present equations that support probabilistic inference based on PFT maneuver models.



(a) Demonstration trajectories.

(b) Learned PFT.

Figure 5-1: Trajectories and the learned probabilistic flow tube of the merging right maneuver.

5.1 Probabilistic Flow Tube Generation

Probabilistic Flow Tubes (PFTs) are used to model the probabilistic continuous motions for autonomous systems. As described in [13, 25], PFT represents a set of continuous trajectories with common characteristics defined over a time interval $[t_1, t_g]$. It is characterized as a set of cross-sectional regions, where each cross section CS_i stores

the mean and covariance of the associated common trajectories at time t_i ($1 \leq i \leq g$).

Algorithm 4 Generate a Probabilistic Flow Tube

Input m (a maneuver type), u (a driving style), $\mathcal{D}_{m,u} = \{D_{m,u}^{(n)}\}_{n=1...N}$, a set of N trajectories

Output $L_{m,u}$ (a PFT model)

procedure GENERATEPFT($\mathcal{D}_{m,u}$)
  $\mathcal{D}' = \text{FastDTW}(\mathcal{D}_{m,u})$

  for $i = 1$ to g **do**
    $\mathbf{x}_i \leftarrow (D_i^{'(1)}, \ldots, D_i^{'(N)})$
    $\mu[i] = \text{ComputeMean}(\mathbf{x}_i)$
    $\Sigma[i] = \text{ComputeCovariance}(\mathbf{x}_i)$

  for $i = 1$ to g **do**
    $\mu[i] = \mu[i] - \mu[1]$

  $L_{m,u} = (\mu, \Sigma)$
  return $L_{m,u}$

Algorithm 5 Generate a Library of Probabilistic Flow Tubes

Input $\mathcal{M}$ (a set of pre-defined maneuvers), $\mathcal{U}$ (a set of pre-defined driving styles), $\mathcal{D}$ (a set of demonstration trajectories)

Output $\mathcal{L}$ (a library of PFTs)

procedure GENERATEPFTLIBRARY($\mathcal{M}, \mathcal{U}, \mathcal{D}$)
  $\mathcal{L} = []$

  for $m = 1$ to $|\mathcal{M}|$ **do**
    for $u = 1$ to $|\mathcal{U}|$ **do**
      $L_{m,u} = \mathit{GeneratePFT}(\mathcal{D}_{m,u})$
      $\mathcal{L}.\text{append}(L_{m,u})$

  return $\mathcal{L}$

Since the trajectories are generated from human demonstrations, PFTs are useful to model realistic human behaviors and incorporate different driver styles. The algorithm to generate a PFT for a discrete maneuver action m with a driving style u, given a set of demonstration trajectories $\mathcal{D}_{m,u}$, is shown in Algorithm 4. Since the trajectories associated with the same action may have different lengths, we first perform fast dynamic time warping [42], a variant of dynamic time warping (DTW)

that has linear time and space complexity, on the input trajectories to temporally align them [37, 45] (Line 2). Assuming all aligned trajectories have a length of g, we compute the mean and covariance among all aligned trajectories at discretized time steps from 1 to g by assuming the time step always increments by 1 (Line 4-7). Finally, we shift the PFT by subtracting the mean at each step from the mean at the first step so that the mean of the first step is always at the origin, which makes all generated PFTs start at the same position (Line 9-10). An example of the generated PFTs is illustrated in Figure 5-1, where we align all demonstration trajectories so that they take the same number of steps, compute the mean and covariance at each step, and align the starting point to the origin.

In Algorithm 5, we generate a library of PFTs for each combination of maneuver type and driving style, by calling Algorithm 4 repeatedly.

Note that Algorithms 4 and 5 assume that each demonstration trajectory is labeled with a maneuver and a driving style. In cases where the demonstration trajectories are unlabelled, we can use clustering algorithms such as Expectation Maximization to cluster them into different groups.

5.2 Inference using Probabilistic Flow Tube

After generating a library of PFTs, we use them to infer the likelihood of possible continuous vehicle locations given a PFT label. Given a driving style label u, a maneuver label m, and a maneuver clock variable i, we can select the corresponding PFT and compute the probability of the continuous vehicle position as a bivariate Gaussian distribution:

$$\mathbb{P}(\mathbf{x}_{c,k}|\mathbf{x}_{M,k} = m, c_k = i; \mathbf{x}_{U,k} = u) = \mathcal{N}(\mu_{m,u}[i], \Sigma_{m,u}[i]), \tag{5.1}$$

where $\mu_{m,u}[i]$ and $\Sigma_{m,u}[i]$ can be obtained by looking up the ith entry in the associated PFT with driving style label u and maneuver label m.

A PFT is a function that maps the current vehicle state to the vehicle state at the

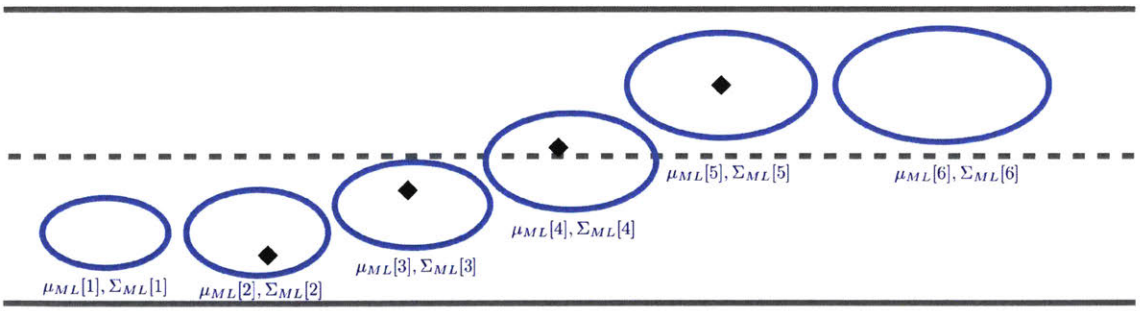


Figure 5-2: Example of aligning an observed trajectory with a probabilistic flow tube.

next time step, since the new clock variable is always incremented by one at each step and the new maneuver state stays the same, the continuous state can be computed using the new clock variable in the same PFT by Equation (5.2).

$$\mathbb{P}(\mathbf{x}_{c,k+1}|\mathbf{x}_{M,k} = m, c_k = i; \mathbf{x}_{U,k} = u) = \mathcal{N}(\mu_{m,u}[i+1], \Sigma_{m,u}[i+1]). \tag{5.2}$$

In addition, similar to [13], given an input state (m, u, i), the likelihood of observing a data trajectory $\mathbf{o}_k$ observed from time step $k - w + 1$ to k with length w, is the product of probability densities of each Gaussian distribution in the flow tube evaluated at the aligned points in the observed vehicle trajectory.

Since the flow tube is not necessarily aligned with the observed trajectory (e.g., the starting position of the vehicle is not the same as the starting position of the flow tube), we shift the flow tube such that the end of the trajectory is aligned with $\mu_{m,u}[i]$ and name the mean of the shifted flow tube $\mu'_{m,u}$. For instance, in Figure 5-2, we observe a past trajectory $\tau_{1:4}$ with length 4 marked by black diamonds. To compute the probability of observing that trajectory given the aggressive (Agg) merging left (ML) PFT shown as blue ellipses at time step 5, we align the center of the fifth ellipse from left in the flow tube to the end of the observed trajectory. The probability is then computed as the product of the probability densities of each Gaussian distribution represented as the aligned blue ellipses evaluated at the black diamonds:

$$\mathbb{P}(\tau_{1:4}|\mathbf{x}_{M,k} = ML, c_k = 5; \mathbf{x}_{U,k} = Agg)) = \prod_{j=1}^{4} \mathcal{N}(\tau_j \mid \mu'_{ML,Agg}[j+1], \Sigma_{ML,Agg}[j+1]). \tag{5.3}$$

In general, given the discrete state and maneuver clock (m, u, i), the likelihood of observing a trajectory $\mathbf{o}_{k-w+1:k}$ is computed as:

$$p(\mathbf{o}_k|\mathbf{x}_{M,k} = m, c_k = i; \mathbf{x}_{U,k} = u) = \prod_{j=k-w+1}^{k} \mathcal{N}(\mathbf{o}^{(j)}|\mu'_{m,u}[i-k+j], \Sigma_{m,u}[i-k+j]). \tag{5.4}$$

This observation likelihood can be used to compute the posterior distribution over the clock variable and maneuver state given an observed trajectory in Chapter 6.

5.3 Future Work for Maneuver Motion Modeling

Despite the advantages of Probabilistic Flow Tubes described at the beginning of this chapter, we would like to show some future work that makes it more applicable and compare it with an alternative method.

5.3.1 Probabilistic Flow Tubes with Flexible Time

In this thesis, although we perform dynamic time warping to temporally align all demonstration trajectories, we assume that all resulting PFTs have a fixed number of time steps, which is unrealistic in cases where vehicle drives with different velocities, because it takes a faster vehicle less time to finish a maneuver than a slower vehicle. Therefore, we would like to propose a few solutions to enable PFTs with flexible time.

One possible solution is to learn flow tubes for different ranges of velocities. For instance, multiple tubes can be created for the same maneuver with different velocity profiles. A PFT with fewer time steps can be learned for faster trajectories and vice versa. However, this solution would lead to a large number of maneuver models and slow down the maneuver estimation process since we need to compare the observations

with each model, as detailed in Chapter 6. An alternative is to add velocity or the duration of the maneuver as parameters in the PFT, which can prevent us from creating multiple PFTs.

Another method is to allow the same number of slices, but vary the step size between successive slices in the PFT. A PFT is essentially a Gaussian process (GP), which consists of a sequence of bivariate Gaussian distributions at discrete indices. Therefore, we can model motions with different velocities and make PFT a more applicable approach, by allowing the step size between successive Gaussian distributions to vary.

5.3.2 Inverse Reinforcement Learning

In addition to PFT, another state-of-the-art method to learn the motion model through demonstration trajectories is to use inverse reinforcement learning (IRL) [55]. IRL learns an immediate reward function, which maps from the current state and an action to a reward, under a Markov Decision Process (MDP) framework that models decision making problems consisting of a set of states, actions, transition functions, and reward functions. In an IRL problem, all elements in the MDP framework are known except reward functions, and the goal is to determine unknown reward functions by assuming the demonstration trajectories (or expert trajectories) always yield the best lifetime rewards. Once the reward function is learned, it is straightforward to use standard MDP approaches, such as value iteration and policy iteration, to find the best actions to take at each state.

Compared to PFT, IRL has a few advantages. First, it does not assume the vehicle states to have Gaussian distributions at discretized time steps. Second, although PFT is capable of incorporating different vehicle states, such as positions and velocities, in its representation, it does not model the relative importance of each component. On the other hand, IRL learns the weight of each state feature to determine the reward function, which is more accurate in some cases where one feature is more important than the other. Third, the predictions from IRL are usually more interpretable, since it explains the human preferences by computing the reward function explicitly for

each action at a given state. Fourth, instead of learning a motion model for each maneuver, we can learn a single IRL model that incorporates all possible maneuvers, which does not require clustering and defining maneuvers.

On the other hand, one of the largest challenges for IRL is that it requires the expert trajectories to inform us about the actions being taken at each state in order to learn a reward function in the MDP setting. In the vehicle case, the actions are usually the control inputs, including gas pedal values and steering wheel angles, and these control inputs are not usually available from uncontrollable agent vehicles. Additionally, IRL takes as input an MDP formulation with accurate transition functions that model the probability over next states given the current states and the control input, which requires an accurate vehicle dynamics model. Even if we have access to control inputs from demonstration trajectories and obtain an accurate MDP model, the performance of IRL still depends on the number of demonstration trajectories. In order to learn a comprehensive reward function, we would need the expert to demonstrate in all kinds of different scenarios that could be encountered, which can be time consuming.

One possible future work is to implement IRL by overcoming the challenges mentioned above and compare it with our current motion modeling method. This would allow us to identify the strength and weakness of our method as compared to state-of-the-art approaches.

5.4 Summary

In this chapter, we present a method to learn maneuver models as probabilistic flow tubes. The PFT representation is useful to capture how a vehicle moves at different stages within a maneuver. Furthermore, the representation incorporates the uncertainties of maneuver motions by learning a Gaussian distribution at each step, which can be used for further estimation and prediction tasks. In the next chapter, we present our method to recognize vehicle intentions using PFTs.

Chapter 6

Recognizing Intentions of Agent Vehicles

Recognizing the intentions of agent vehicles in proximity to our ego vehicle accurately is important if we want to achieve safe planning and execution, because inaccurate recognition results can lead to plans that collide with these agent vehicles. In this chapter, we present our solution to the intent recognition problem using a three-step approach as shown in Figure 6-1, which is able to produce probabilistic estimation and prediction results accurately in real-time. As describe in Chapter 3, the input to this problem is a set of learned vehicle models and observations on agent vehicles, and the output consists of estimations and predictions on agent vehicle driving styles, maneuvers, and trajectories.

In the first step, the objective is to compute a distribution over possible vehicle models. This is achieved by learning a probabilistic driving style estimator using inputs from the agent vehicle states and environmental states. Second, given the chosen vehicle models associated with different driving styles using Bayesian filtering, we estimate the intended maneuvers and clock variables, by assuming that it only depends on past trajectory of the agent vehicle. Third, given the estimated maneuver states, we compute the probability distribution over the future hybrid states, including maneuver sequences as well as position sequences, of the agent vehicle over a finite predicting horizon h using a recursive online Bayesian prediction approach.

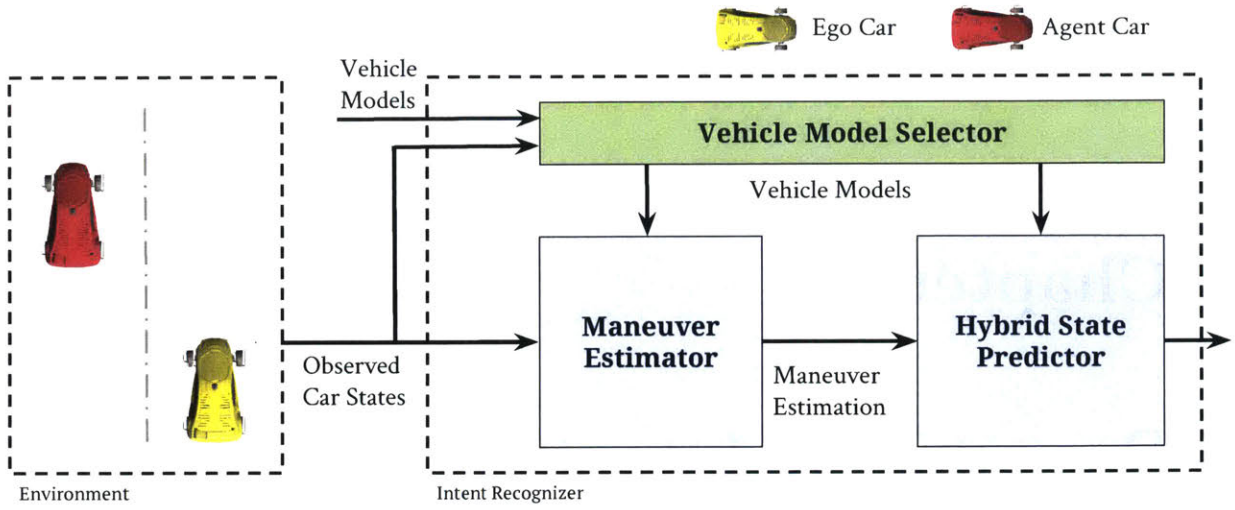


Figure 6-1: Architecture diagram of the intent recognition module.

6.1 Learning a Driving Style Classifier and Selecting Vehicle Models

After maneuver motion learning, we obtain a set of vehicle models for each driving style. In order to recognize the intentions accurately for an arbitrary agent vehicle, the first step is to choose appropriate vehicle models that capture its behavior by estimating its adopted driving styles. In this thesis, we assume that it is sufficient to estimate the driving style through the most recent observed data trajectory, and learn a model called *Multinomial Logistic Regression Classifier* to generate the probability over all possible driving styles that can be adopted by the driver based on various driving features, such as velocities, accelerations, and distances to lane center, from past five observations. This model is advantageous compared to other data-driven models as it outputs a probability distribution over possible outputs using a logistic function instead of the most likely outcome.

In a standard logistic regression classification task, we try to predict the probability of a dependent variable y taking two possible outcomes (0 or 1), based on a vector of independent variables or features $\mathbf{s}$. The probability is calculated using a logistic

function (or sigmoid function), defined in Equation (6.1):

$$\sigma_l(\alpha) = \frac{1}{1 + \exp(-\alpha)}. \tag{6.1}$$

The logistic function maps any real input $\alpha \in \mathbb{R}$ to a value between 0 and 1 so that we can interpret the result as a probability, and each outcome becomes mutually exclusive. The standard logistic regression classification probability is defined in Equation (6.2):

$$\begin{aligned} p(y = 1|\mathbf{s}) &= \sigma_l(\mathrm{w}^T\mathbf{s} + w_0) \\ p(y = 0|\mathbf{s}) &= 1 - \sigma_l(\mathrm{w}^T\mathbf{s} + w_0) \end{aligned} \tag{6.2}$$

The goal is to find weights w and w_0 such that when $y = 0$, $\mathbb{P}(y = 0|\mathbf{s})$ is large and $\mathbb{P}(y = 1|\mathbf{s})$ is small, and when $y = 1$, $\mathbb{P}(y = 0|\mathbf{s})$ is small and $\mathbb{P}(y = 1|\mathbf{s})$ is large. This is equivalent to minimizing the cost function, defined in Equation (6.3) given a set of training data $\{(\mathbf{s}^{(i)}, y^{(i)}), i = 1 \ldots n\}$:

$$J(\theta, \theta_0) = -\sum_i^n \left(y^{(i)} \log \left(\sigma_l(\mathrm{w}^T\mathbf{s}^{(i)} + w_0) \right) + (1 - y^{(i)}) \log \left(1 - \sigma_l(\mathrm{w}^T\mathbf{s}^{(i)} + w_0) \right) \right). \tag{6.3}$$

We can use the standard gradient descent method to find the desired weights. One challenge in the driving style estimation problem is that there are sometimes more than two types of driving styles, hence we need to use a multinomial logistic regression method, by generalizing the standard logistic regression method to multiple classes. To do so, we use a softmax function in Equation (6.4):

$$\sigma_s(i, \alpha_1, \ldots, \alpha_n) = \frac{\exp(s_i)}{\sum_{j=1}^n \exp(\alpha_j)}. \tag{6.4}$$

The multi-classification probability is defined in Equation (6.5):

$$p(y = i|\mathbf{s}) = \sigma_s(i, \mathrm{w}_1^T\mathbf{s} + w_1, \ldots, \mathrm{w}_n^T\mathbf{s} + w_n). \tag{6.5}$$

Similarly to the logistic function, the softmax function outputs a value between 0 and 1, and all resulting values sum up to 1, so that we can interpret the output value as a probability. Furthermore, by exponentiating the input values, the softmax function exaggerates the difference between them. To find the parameters $\mathrm{w}_1, w_1, \ldots, \mathrm{w}_n, w_n$, we use standard gradient descent method to minimize the cost function similar to Equation (6.3) given the training data, which works effectively to find the optimal parameters by iteratively moving in the direction of the minimized cost function.

After learning the classifier, we apply Equation (6.5) to compute the probability distribution over each driving style at time step k, assuming that the current driving style depends on the history driving data:

$$p(\mathbf{x}_{U,k} = u|\mathbf{o}_{1:k}) = p(\mathbf{x}_{U,k} = u|\mathbf{o}_{U,k}) = \sigma_s(u, \mathrm{w}_1^T\mathbf{o}_{U,k} + w_1, \ldots, \mathrm{w}_n^T\mathbf{o}_{U,k} + w_n), \quad (6.6)$$

where $\mathbf{o}_{U,k} \subseteq \mathbf{o}_{1:k}$ is a subset of the observed data trajectory that is helpful for determining the driving styles effectively. More details on the selection this subset are discussed in Chapter 8.

The estimated driving styles allow us to choose a subset of vehicle models that are suitable for the agent vehicle before we want to estimate its motions. More details are discussed in the next section.

6.2 Estimating Maneuver States

After estimating the driving style and selecting the corresponding *tPHA* models, we can use them to infer the probability distribution over the current vehicle states given observations using standard Bayesian filtering algorithms. Since the current positions are assumed to be fully observable, our problem is simplified to estimate the maneuver states including maneuver types and clock variables given the observations.

As described in Chapter 3, the maneuver state $\mathbf{x}_{d,k} = (\mathbf{x}_{M,k}, c_k)$ is a tuple consisting of a maneuver label variable and a clock variable. Although c_k is defined as a continuous variable in the problem formulation, we assume that it starts at 0 and

always increments by 1, which makes it essentially a discrete variable. The maneuver state $\mathbf{x}_{d,k}$ is estimated by summing up conditional probabilities weighted by vehicle model probabilities:

$$p(\mathbf{x}_{d,k}|\mathbf{o}_{1:k}) = \sum_{u \in \mathcal{U}} p(\mathbf{x}_{U,k} = u|\mathbf{o}_{1:k}) p(\mathbf{x}_{d,k}|\mathbf{o}_{1:k}; \mathbf{x}_{U,k} = u). \tag{6.7}$$

Each conditional probability is computed using the standard filtering algorithm as shown in Equation (6.8):

$$\mathrm{p}(\mathrm{x}_{d,k}|\mathbf{o}_{1:k}; \mathbf{x}_{U,k} = u) \propto p(\mathbf{o}_k|\mathbf{x}_{d,k}; \mathbf{x}_{U,k} = u) \sum_{\mathbf{x}_{d,k-1}} p(\mathbf{x}_{d,k}|\mathbf{x}_{d,k-1}; \mathbf{x}_{U,k} = u) p(\mathbf{x}_{d,k-1}|\mathbf{o}_{1:k-1}; \mathbf{x}_{U,k} = u), \tag{6.8}$$

where $p(\mathbf{o}_k|\mathbf{x}_{d,k}; \mathbf{x}_{U,k} = u)$ is given by observation function in Chapter 5, $p(\mathbf{x}_{d,k}|\mathbf{x}_{d,k-1}; \mathbf{x}_{U,k} = u)$ is given by transition function in the *tPHA* model corresponding to driving style u, and $p(\mathbf{x}_{d,k-1}|\mathbf{o}_{1:k-1}; \mathbf{x}_{U,k} = u)$ is estimation results from the previous step.

6.2.1 Pruning Maneuver States

Since the maneuver state contains two elements: a maneuver type and a clock variable, the number of possible values for $\mathbf{x}_{d,k}$ is the product of the domains of these two elements. For instance, in the lane change example shown in Figure 1-1, if there are three maneuvers (merge left, merge right, and go forward), each taking 25 clock cycles, the number of states would be 75, which is hard to track if we want to achieve online intent recognition.

One technique is to assign the probability of low-likely events (e.g., with probability smaller than a pre-defined threshold ϵ) to be zero. This also greatly reduces the number of states because many of them have negligible probabilities. For instance, an ϵ of 0.01% will filter out approximately 96% of possible states. After we apply this technique, the number of possible maneuver states is usually reduced drastically.

6.3 Predicting Hybrid States

Given the estimated belief state, including driving styles, maneuvers states, and vehicle positions, at the current step k, we wish to predict the future hybrid vehicle states over a predicting horizon of h using standard Bayesian prediction algorithms. To start, we compute the probability distribution over the next hybrid step at $k+1$ as a sum of conditional probabilities conditioning on all possible vehicle models:

$$p(\mathbf{x}_{d,k+1}, \mathbf{x}_{c,k+1}|\mathbf{o}_{1:k}) = \sum_{u \in \mathcal{U}} p(\mathbf{x}_{U,k} = u|\mathbf{o}_{1:k}) p(\mathbf{x}_{d,k+1}, \mathbf{x}_{c,k+1}|\mathbf{o}_{1:k}; \mathbf{x}_{U,k} = u), \tag{6.9}$$

Each conditional probability is computed as follows:

$$p(\mathbf{x}_{d,k+1}, \mathbf{x}_{c,k+1}|\mathbf{o}_{1:k}; \mathbf{x}_{U,k} = u) = p(\mathbf{x}_{d,k+1}|\mathbf{o}_{1:k}; \mathbf{x}_{U,k} = u) p(\mathbf{x}_{c,k+1}|\mathbf{x}_{d,k+1}, \mathbf{o}_{1:k}; \mathbf{x}_{U,k} = u), \tag{6.10}$$

The distribution over possible maneuvers in the next step, $p(\mathbf{x}_{d,k+1}|\mathbf{o}_{1:k}; \mathbf{x}_{U,k} = u)$, is computed as follows:

$$p(\mathbf{x}_{d,k+1}|\mathbf{o}_{1:k}; \mathbf{x}_{U,k} = u) = \sum_{\mathbf{x}_{d,k}} p(\mathbf{x}_{d,k+1}|\mathbf{x}_{d,k}, \mathbf{o}_{1:k}; \mathbf{x}_{U,k} = u) p(\mathbf{x}_{d,k}|\mathbf{o}_{1:k}; \mathbf{x}_{U,k} = u), \tag{6.11}$$

where $p(\mathbf{x}_{d,k+1}|\mathbf{x}_{d,k}, \mathbf{o}_{1:k}; \mathbf{x}_{U,k} = u)$ is computed using the discrete transition function T_d as specified in the corresponding *tPHA* model and the second term is the estimation result from the previous step.

The probability of next continuous state, $p(\mathbf{x}_{c,k+1}|\mathbf{x}_{d,k+1}, \mathbf{o}_{1:k}; \mathbf{x}_{U,k} = u)$, is assumed to be independent of the observations given discrete estimations, and can be computed using Equation (5.1).

Applying Equations (6.9), (6.10), and (6.11) recursively, we can estimate the future maneuver sequences as well as continuous states over a predicting horizon of h for the agent vehicle:

$$\mathrm{p}(\mathrm{x}_{d,k+h}, \mathbf{x}_{c,k+h})|\mathbf{o}_{1:k}) = \textstyle\sum_{u \in \mathcal{U}} p(\mathbf{x}_{U,k} = u|\mathbf{o}_{1:k}) p(\mathbf{x}_{c,k+h}|\mathbf{x}_{d,k+h}; \mathbf{x}_{U,k} = u) p(\mathbf{x}_{d,k+h}|\mathbf{o}_{1:k}; \mathbf{x}_{U,k} = u). \tag{6.12}$$

As mentioned in Chapter 4.3.1, the value of predicting horizon h can be very large, since it increases linearly with the number of clock cycles included in each maneuver. Additionally, the size of possible values for $\mathbf{x}_{d,k:k+h}$ increases exponentially with h. As a result, the motion planner has to search over a very large state space if the planning horizon is large. To alleviate this issue, we use a similar strategy to reduce the size of possible state sequences as shown in Section 6.2.1, by ignoring the very low likely events (e.g., with probability smaller than a pre-defined threshold ϵ) when predicting the hybrid vehicle states. For instance, given three maneuvers each taking 25 clock cycles, a predicting horizon of 50, and a transition function shown in Figure 3-3, the number of possible states is reduced by approximately 98% using an ϵ of 0.01%.

6.4 Future Work for Intent Recognition

In this section, we discuss a number of possible extensions that can strengthen our method by relaxing some assumptions made in this thesis, including adding noises in the observations, recognizing goals as part of intent, combining driving style estimation with maneuver estimation into a single coupled filtering problem, considering interaction between different agent vehicles and recognizing their intentions together, and recognizing intentions using deep neural network.

6.4.1 Intent Recognition with Noisy Observations

Since the major focus in this thesis is the uncertainty from the intentions of the agent vehicle, we decide to simplify the continuous state estimation problem by assuming that the continuous vehicle locations are fully observable, as stated in the problem formulation in Section 4.3.1.

However, this is not true in most real world applications, as there are always noises in the observations coming from perception sensors. Therefore, we would like to formulate the estimation problem as a *hybrid mode estimation problem* [24, 48] to estimate the hybrid state that contains both the discrete mode and the continuous state by considering noises in the state observations.

As introduced in [48], one typical way to estimate the continuous state is through a discrete-time Kalman filter, by assuming a discrete-time continuous-state time varying dynamical system for the agent vehicle:

$$\mathbf{x}_{c,k+1} = A_k \mathbf{x}_{c,k} + B_k \mathbf{u}_k + B_k^w \mathbf{w}_k \tag{6.13}$$

where $\mathbf{x}_c$ is the continuous state, $\mathbf{u}$ is the continuous control input, and $\mathbf{w}$ is the uncertainty of the agent vehicle that models disturbances, uncertain system model parameters, and sensor noises. Given the dynamical system, the Kalman filter estimates the current continuous state in two steps. First, it predicts the current state and covariance based on the previous estimate and the control input. Second, it computes the Kalman filter gain and refines the prediction based on the current measurement and the observation function.

Unfortunately, the control input of the agent vehicle is not available, which makes the problem of estimating the continuous state for the agent vehicle challenging. One possible solution is to estimate the control inputs based on the observed vehicle acceleration and angular rate.

Additionally, we would like to follow the work in [48] that uses best-first enumeration and conflict-directed search algorithms to identify important events earlier and speed up the estimation process if our intent recognition problem scales up to a handful of agents.

6.4.2 Intent Recognition with Goals

An important intention to consider is the goal state of each agent vehicle, since it provides useful cues for the future motions of the vehicle. In this thesis, we mostly consider recognizing the short-term maneuvers such as turning left and merging to the right lane. On the other hand, long-term goals, such as exiting the highway in a mile or going to the closest restaurant, are also important to be recognized, since it allows us to predict maneuver sequences in the future more accurately. For instance, if we know that the long-term goal is to exit the highway, our maneuver estimator

will output lane change maneuvers with higher probabilities.

Therefore, we would like to recognize the long-term goal states in the future by pre-defining a set of possible goal states $\mathbf{x}_G$ for the agent vehicle and adding them to the discrete state in our tPHA vehicle model. As a result, we can estimate the goal state together with the maneuver state given the vehicle model and the observations, by computing the following equation:

$$p(\mathbf{x}_{d,k}, \mathbf{x}_{G,k}|\mathbf{o}_{1:k}; \mathbf{x}_{U,k} = u) = p(\mathbf{x}_{G,k}|\mathbf{o}_{1:k}; \mathbf{x}_{U,k} = u)p(\mathbf{x}_{d,k}|\mathbf{x}_{G,k}, \mathbf{o}_{1:k}; \mathbf{x}_{U,k} = u), \quad (6.14)$$

where $(\mathbf{x}_{G,k}|\mathbf{o}_{1:k}; \mathbf{x}_{U,k} = u)$ can be computed by comparing the observations with the optimal actions to each possible goal state given the vehicle models [40]. Another way to do this is to define a set of macros, and use a planner to precompute maneuver sequences that implement each macro. The goal probability can be computed using an algorithm similar to [34].

6.4.3 Driving Style Estimation Using Bayesian Filtering

In this thesis, we assume that the driving style only depends on the current observations and ignore past observations that can help us get a more robust estimate. For instance, if the current observations are noisy and tell us that two driving styles, such as aggressive and normal, are equally likely. On the other hand, based on past observations, we compute a prior distribution that suggests the aggressive style is much more likely to happen. It makes more sense that our driving style estimator favors aggressive style with a higher probability, by reasoning both the current observations and the prior distribution. Such a method is called Bayesian filtering, as shown below:

$$p(\mathbf{x}_{U,k}|\mathbf{o}_{U,1:k}) = p(\mathbf{o}_{U,k}|\mathbf{x}_{U,k}) \sum_{\mathbf{x}_{U,k-1}} p(\mathbf{x}_{U,k}|\mathbf{x}_{U,k-1})p(\mathbf{x}_{U,k-1}|\mathbf{o}_{U,1:k-1}). \quad (6.15)$$

Moreover, we can combine driving style estimation and maneuver state estimation in a single Bayesian filtering framework, which generates even more robust results, as

past estimation on maneuver states is also a useful prior that indicates what driving style is more likely to be adopted by the target agent vehicle. The new Bayes filter can be computed with the equation below:

$$\mathrm{p}(\mathrm{x}_{M,k}, \mathbf{x}_{U,k}|\mathbf{o}_{1:k}) = p(\mathbf{o}_k|\mathbf{x}_{M,k}, \mathbf{x}_{U,k}) \sum_{\mathbf{x}_{M,k-1}, \mathbf{x}_{U,k-1}} p(\mathbf{x}_{M,k}, \mathbf{x}_{U,k}|\mathbf{x}_{M,k-1}, \mathbf{x}_{U,k-1}) p(\mathbf{x}_{M,k-1}, \mathbf{x}_{U,k-1}|\mathbf{o}_{1:k-1}), \tag{6.16}$$

where the observation function can be obtained from Equation (5.1), and the transition function in the *tPHA* vehicle model needs to be modified to incorporate driving style.

6.4.4 Interaction-Aware Intent Recognition

Another assumption made in this thesis is that the intention of each agent vehicle is independent of the intentions of other vehicles, which is true if they are far away. In cases where multiple agent vehicles are close, it is crucial to consider their interactions, as the future motions of one vehicle usually depend on the locations and intentions of nearby vehicles.

Therefore, we would like to estimate the intentions of all nearby agent vehicles together by combining their vehicle models together into a single model called concurrent probabilistic hybrid automaton (cPHA) [24, 47]. By allowing the information of each vehicle, such as locations and possible intentions, to be shared among each other, cPHA has two major advantages. First, it can provide more accurate recognition results over agent vehicles' intentions. For instance, if we know that the left side of agent vehicle A is blocked by agent vehicle B, we will rule out the case where A will merge left. Second, it can reduce the number of predictions over each agent vehicle's future motions, since no two agent vehicles should be predicted to arrive at the same position by a unified model.

6.4.5 Intent Recognition using Deep Neural Networks

Deep neural networks (DNN) have become very popular in recent years thanks to the advances in computational powers from the graphics processing unit (GPU). They have shown many successes in end-to-end classification and regression tasks in the domain of computer vision (e.g., image recognition) and natural language processing (e.g., voice recognition).

Since intent recognition is essentially a task of classification and regression, we would like to use DNN to estimate the current maneuver and driving style of a agent vehicle directly without generating maneuver motion models, because such models are sometimes hard to obtain. One possible method is to use a recurrent neural network (RNN) to produce estimations because it accounts for the temporal information in the input data. Since driving is a time-dependent task, RNN can extract more information from driving history and make meaningful estimations. Also, because of the success of DNN in computer vision tasks, we can also include camera image as one of the inputs to our estimator to extract more meaningful information such as road boundaries.

One challenge of using DNN-based methods for vehicle motion predictions is the requirement of a large labelled dataset [28]. The SHRP 2 NDS dataset [7] used for learning driving style estimator is a good candidate, but we need to spend more efforts on cleaning up the dataset because the raw dataset contains a lot of noise.

6.5 Summary

In this chapter, we present a three-step approach to the intent recognition problem that produces accurate probabilistic results over agent vehicle intentions, which include their driving styles, maneuver types, and future trajectories. Compared to the prior art, our method combines the problem of recognizing different aspects of vehicle intentions in a unified framework, where driving styles help identify the correct vehicle models, maneuvers help determine the agent vehicle's high level actions, and future motions help identify potential risk for our ego vehicle. These estimation results are used to update the maneuver planning problem model, and the updated model can be

used to find desired maneuver plans that achieve our safe and efficient requirements. More details are discussed in the next chapter.

Chapter 7

Intention-Aware Conditional Planning

In this chapter, we present our solution to the risk-bounded conditional maneuver planning problem, which is the last piece in our driving executive that is responsible for generating safe and efficient maneuver plans for our ego vehicle. Without the safety aspect, it is possible for the ego vehicle to collide with agent vehicles and lead to catastrophic outcomes. Without the efficiency aspect, the ego vehicle will likely to behave conservatively in tight traffic as illustrated in Chapter 1.

Our solution is divided into two steps. First, we model the maneuver planning problem into a structured model called chance-constrained partially observable Markov decision process with multiple agents (CC-POMDP-MA) [27] that encodes different vehicle models and constraint requirements. Using such a model allows us to utilize existing tools to find maneuver plans efficiently. Second, we solve the CC-POMDP-MA model with a state-of-the-art solver called risk-bounded AO* (RAO*) [43] based on the intent recognition results. These two steps would be called by our autonomous driving executive repeatedly until the ego vehicle reaches its goal.

7.1 Problem Modeling

Given the complexity of a maneuver planning problem that includes many vehicles and a set of objectives, it is important to have a unified model that satisfies the following requirements. First, the model needs to represent maneuvers and motions that each vehicle takes, which means it has to be hybrid. Second, the model needs to capture uncertainties of the agent vehicle intentions, so it has to be probabilistic. Third, it is required to model multiple vehicles operating at the same time, hence is concurrent. Fourth, since the maneuvers and driving styles cannot be observed directly, the model needs to be partially observable. Finally, the model needs to capture safety constraints as defined in the problem statement. Therefore, we introduce a model called chance-constrained partially observable Markov decision process with multiple agents *CC-POMDP-MA* [27] that satisfies the listed requirements. CC-POMDP-MA defines the problem of maneuver planning as a tuple

$$\mathcal{P} = \langle \mathcal{S}, \mathcal{A}, \mathcal{O}, T, O, C, \mathcal{B}_0, H, \Delta \rangle,$$

where

- $\mathcal{S}$ is a set of hybrid states for each vehicle, including the positions, and discrete states, including maneuvers and driving styles,
- $\mathcal{A}$ is the set of maneuvers that can be taken by the ego vehicle,
- $\mathcal{O}$ is a set of observations on the vehicle positions and environment states, such as distance to lane center,
- $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is a state transition function between hybrid states. The transition function depends not only on the action chosen by the ego vehicle, but also on the action of each agent vehicle that further depends on its intention,
- $O : \mathcal{S} \times \mathcal{A} \times \mathcal{O} \to \mathbb{R}$ is an observation function,
- $C : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a cost function associated with each action in $\mathcal{A}$,

- $\mathcal{B}_0$ is the initial belief state over hybrid states,
- H is the finite planning horizon,
- Δ is an upper bound probability for the risk constraint, where the risk is defined as the probability of our ego vehicle near colliding with any of the agent vehicles up to the planning horizon.

Given the CC-POMDP-MA model, we provide an algorithm in the next section that produces an optimal, deterministic, and chance-constrained policy π^*_{rb} at time step k.

7.2 Risk-Bounded AO* (RAO*)

As introduced in Chapter 2, there are a few methods to find efficient solutions to CC-POMDP models. In this thesis, we utilize a risk-bounded algorithm called RAO* to solve for POMDP instances effectively through a heuristic forward search.

The goal of RAO* is to find an optimal risk-bounded maneuver plan (or policy) π^*_{rb} that minimizes the expected cost over a finite horizon given the current belief state $\mathcal{B}_k$:

$$\pi^*_{rb} = \arg\min_{\pi} \mathbb{E}\left[\sum_{t=k}^{k+H} C(s_t, a_t) \middle| \mathcal{B}_k, \pi\right], \tag{7.1}$$

subject to

$$er(\mathcal{B}_k|\pi^*_{rb}) \leq \Delta, \tag{7.2}$$

where er is the execution risk up to horizon H:

$$er(\mathcal{B}_k|\pi^*_{rb}) = 1 - \mathbb{P}(\bigwedge_{i=k}^{k+H} \text{Safe}_i|\mathcal{B}_k, \pi^*_{rb}) = \mathbb{P}(\bigvee_{i=k}^{k+H} \text{Collision}_i|\mathcal{B}_k, \pi^*_{rb}). \tag{7.3}$$

The execution risk is a measurement of the probability of violating the risk over planning horizon H at any step from k to $k+H$, where the risk is defined as near collisions with obstacles. Finding the execution risk is equivalent to computing the probability of the ego vehicle colliding with any of the agent vehicles at any step from

k to $k+H$ using the following recursive form:

$$er(\mathcal{B}_k|\pi) = r_b(\mathcal{B}_k) + (1 - r_b(\mathcal{B}_k)) \sum_{o_{k+1} \in \mathcal{O}} \mathbb{P}^{\text{Safe}}(o_{k+1}|\pi(\mathcal{B}_k), \mathcal{B}_k) er(\mathcal{B}_{k+1}|\pi) \tag{7.4}$$

where $r_b(\mathcal{B}_k)$ is the immediate near collision probability at belief state $\mathcal{B}_k$, and the rest is the probability of near collisions from $k+1$ to $k+H$ given that the vehicle is collision free at $\mathcal{B}_k$. More explanations of computing Equation (7.4) can be found in Section 2.1 in the original RAO* paper [43]. The immediate risk probability $r_b(\mathcal{B}_k)$ is domain dependent, and in our thesis we compute this probability as the likelihood that any of the agent vehicle's position lies within the boundary of the ego vehicle modeled as a rectangle. Since the position of each agent vehicle is modeled as a Gaussian distribution given $\mathcal{B}_k$, we can either approximate the probability using a sampling-based Monte Carlo method or obtain an exact solution on the upper bound of the risk using a moment-sum-of-squares approach introduced in [29].

Given the definition of the execution risk, RAO* is able to find the desired policy effectively using a heuristic forward search method in the space of belief states with two heuristics: a utility heuristic that guides the policy search towards more promising branches, and a risk heuristic that helps prune over-risky branches and thus saves search time. In our vehicle maneuver planning problem, the utility heuristic is defined as the Euclidean distance to the goal location, which is admissible and consistent. In addition, we define the admissible risk heuristic at $\mathcal{B}_k$ to be the immediate risk $r_b(\mathcal{B}_k)$ so that the searching algorithm would never miss pruning safe branches.

The main algorithm is illustrated in Algorithm 6 [43]. Different from the original algorithm, we modify RAO* so that it begins the search with the belief state at time step k: $\mathcal{B}_k$, and then incrementally expands the graph $\mathcal{G}$ and updates the policy π using modules including ExpandGraph and UpdatePolicy as shown in Algorithms 7 and 8, respectively [43]. In ExpandGraph, we introduce a notion of Q-function that measures the expected cost after taking an action a_k at belief state

Algorithm 6 Main RAO* Algorithm

Input $\mathcal{P}$ (a CC-POMDP-MA instance), $\mathcal{B}_k$ (a belief state at time step k)
Output π^*_{rb} (an risk-bounded optimal policy)

procedure RAO*($\mathcal{P}, \mathcal{B}_k$)
    Initialize graph $\mathcal{G}$ and policy π consist of $\mathcal{B}_k$

    while not Terminated() **do**
        n, $\mathcal{G} \leftarrow$ *ExpandGraph*($\mathcal{G}$, π)
        $\pi \leftarrow$ *UpdatePolicy*(n, $\mathcal{G}$, π)

    return π^*_{rb}.

Algorithm 7 Expand Search Graph

Input $\mathcal{G}$ (Explicit graph), π (policy)
Output $\mathcal{G}'$ (Expanded explicit graph), n (Expanded node)

procedure EXPANDGRAPH($\mathcal{G}$, π)
    $\mathcal{G}' \leftarrow \mathcal{G}$, $n \leftarrow$ *ChoosePromisingLeaf*($\mathcal{G}$, π)

    for each action a available at n **do**
        $ch \leftarrow$ expand children of (n, a) based on the belief state and the transition function
        $\forall c \in ch$, estimate execution risk er and Q-function using (7.4) and (7.5)
        $\forall c \in ch$, update execution risk bounds

        if no $c \in ch$ violates its risk bound **then**
            $\mathcal{G}' \leftarrow$ add hyperedge $[(n, a) \rightarrow ch]$

    if no action added to n **then**
        mark n as terminal
    return $\mathcal{G}'$, n

$\mathcal{B}_k$:

$$Q(\mathcal{B}_k, a_k) = \sum_{s_k \in \mathcal{S}} C(s_k, a_k)\mathcal{B}_k + \sum_{o_{k+1} \in \mathcal{O}} \mathbb{P}(o_{k+1}|a_k, \mathcal{B}_k)Q^*(\mathcal{B}_{k+1}), \tag{7.5}$$

where the cost $C(s_k, a_k)$ is defined in the action model in Chapter 3, and $Q^*(\mathcal{B}_{k+1})$ is approximated by the utility heuristic at belief state $\mathcal{B}_{k+1}$.

Lastly, the main RAO* algorithm terminates until all of the leaf nodes of the policy π reach terminal.

The two important functions in RAO* are *ChoosePromisingLeaf* in Algorithm 7

Algorithm 8 Update Policy

Input n (expanded node), $\mathcal{G}$ (explicit graph), π (policy)
Output π' (updated policy)

procedure UPDATEPOLICY($n, \mathcal{G}, \pi$)
  $Z \leftarrow$ set containing n and its ancestors reachable by π

  while $Z \neq \emptyset$ **do**
    $n \leftarrow$ remove from Z a node with no descendant in Z

    while there are actions to be chosen at n **do**
      $a \leftarrow$ *ChooseBestAction*($\mathcal{G}$) satisfying the execution risk bound
      Propagate execution risk bound of n to the children of the hyperedge (n, a)

      if no children violates its execution risk bound **then**
        $\pi(n) \leftarrow a$; **break**
    if no action was selected at n **then**
      mark n as terminal
  return π'

and *ChooseBestAction* in Algorithm 8, because they help the algorithm decide the order of the search when there are multiple options. These two functions are designed to be flexible as they are domain dependent and any choice of them does not affect the optimally and risk-bound properties of the algorithm [43]. In this thesis, we choose to select an action that minimizes the expected Q-function, defined in (7.5). Additionally, we choose the most promising leaf with the best utility heuristic, defined as the Euclidean distance to the goal location.

By using a *CC-POMDP-MA* instance and specifying the domain-dependent pieces in the algorithm, such as the immediate risk probability and ways to choose the search order, we can utilize RAO* to find a risk-bounded policy that minimizes the expected cost to the goal.

7.3 Online Intention-Aware Maneuver Planning

In this section, we present our maneuver planning algorithm that is able to help our ego vehicle navigate to its goal safely and efficiently. Algorithm 9 describes the

method that combines the intent recognizer showed in Chapter 6 and the RAO* solver showed in Section 7.2. The novelty of this algorithm is updating the belief state online using intent recognition results to achieve real time safe planning.

Algorithm 9 Online Intention-Aware Maneuver Planning

Input CC-POMDP-MA $\mathcal{P}$, agent vehicle models $t\mathcal{PHA}$.
Output None.

1: $\mathcal{P}_0 \leftarrow \mathcal{P}$
2: $\mathcal{B}_0 \leftarrow$ GetInitialBeliefState($\mathcal{P}_0$)
3: $k \leftarrow 1$
4:
5: **while** True **do**
6: $\quad \mathbf{o}_k \leftarrow$ ObserveCurrentVehicleData()
7: $\quad \mathbf{o}_{1:k} \leftarrow$ UpdateObservation($\mathbf{o}_k, \mathbf{o}_{1:k-1}$)
8:
9: $\quad \mathbf{p}(\mathbf{x}_{U,k}) \leftarrow$ EstimateDrivingStyle($\mathbf{o}_{1:k}$, *MLRC*)
10: $\quad t\mathcal{PHA}' \leftarrow$ Select vehicle models from $t\mathcal{PHA}$ using $\mathbf{p}(\mathbf{x}_{U,k})$
11: $\quad \mathbf{p}(\mathbf{x}_{d,k}) \leftarrow$ EstimateManeuverState($\mathbf{o}_{1:k}, t\mathcal{PHA}', \mathcal{B}_{k-1}$)
12: $\quad \mathcal{B}_k \leftarrow$ UpdateBeliefState($\mathbf{o}_k, \mathbf{p}(\mathbf{x}_{d,k})$)
13:
14: $\quad$ **if** AtGoal($\mathcal{B}_k$) **then**
15: $\quad\quad$ Output success.
16: $\quad\quad$ **break**
17:
18: $\quad \mathbf{p}(\mathbf{x}_{d\rightarrow}, \mathbf{x}_{c\rightarrow}) \leftarrow$ PredictHybridState($\mathbf{o}_{1:k}, t\mathcal{PHA}', \mathbf{p}(\mathbf{x}_{d,k})$)
19: $\quad T_k \leftarrow$ UpdateTransition($\mathbf{p}(\mathbf{x}_{d\rightarrow}, \mathbf{x}_{c\rightarrow})$)
20:
21: $\quad \mathcal{P}_k \leftarrow$ UpdateModel($\mathcal{P}_{k-1}, \mathcal{B}_k, T_k$)
22: $\quad \pi^*_{rb} \leftarrow$ RAO*($\mathcal{P}_k, \mathcal{B}_k$)
23: $\quad a^*_{rb} \leftarrow$ GetAction(π^*_{rb}, $\mathcal{B}_k$)
24: $\quad$ Execute(a^*_{rb})
25:
26: $\quad k \leftarrow k + 1$

The algorithm is initialized with the necessary variables given the inputs to the problem (Line 1-3). At each execution step k, we begin by updating the observation sequence through collecting the latest vehicle data through perception sensors (Line 6-7). We assume that the driving executive always has access to enough observations from the past. Since the major uncertainty in this work comes from the intentions of agent vehicles, we assume that the perception sensors are perfect and thus the vehicle

positions are fully observable.

Then we estimate the driving style using a multinomial logistic regression classifier learned in Section 6.1 and select a subset of vehicle models corresponding to each possible driving style (Line 9-10). Next, we estimate the maneuver state, including the maneuver label and the maneuver clock for each agent vehicle based on the selected vehicle models using Equation (6.8) (Line 11). The estimation results and the most recent observed vehicle positions are used to update the current belief state (Line 12). If the belief state satisfies the goal condition (e.g., the ego vehicle arrives at the goal location), we output success and end the planning process. Otherwise, we need to continue to predict the future hybrid states $\mathbf{p}(\mathbf{x}_{d,k+1:k+h}, \mathbf{x}_{c,k+1:k+h})$ (abbreviated as $\mathbf{p}(\mathbf{x}_{d\rightarrow}, \mathbf{x}_{c\rightarrow})$) simultaneously for all agent vehicles (Line 18). The hybrid predictions are used in two ways: the discrete parts allow us to update the information on how the agent vehicles will move in the future, and the continuous part is useful for computing near collision risk with the ego vehicle (Line 19-21).

Once the POMDP is updated with the latest intent recognition results, our driving executive uses the RAO* algorithm as described in Section 7.2 to find an optimal policy. Although the approach is not limited to a specific solver and can in fact utilize any off-the-shelf POMDP solver to find a feasible policy, we choose RAO* because it is able to generate an optimal policy efficiently with an upper bound guarantee on the risk of near collisions with moving obstacles. In the next section, we discuss why using our method is more desirable compared to an alternative state-of-the-art method.

Given the output policy from the solver, our driving executive can quickly look up the optimal action to execute at planning step k (Line 23-24). The driving executive would continue incrementing the planning step and iterating through the same steps in a loop until the goal condition is satisfied or a time-out signal is triggered.

7.4 Comparison to Iterative RAO*

An alternate way to produce risk-bounded policies is to use an iterative RAO* algorithm that generates a number of offline policies using RAO* considering all possible scenarios in advance and iteratively updates the search graph during replanning [27]. In this section, we explain why using iterative RAO* with offline policies is not considered because of the space limitation in the driving-related problems considered in this thesis.

To compare our method with the iterative RAO* algorithm, we introduce two major performance metrics that are important for effective maneuver planning: time complexity and space complexity. The former indicates how fast the solver can produce satisfactory results, and the latter is important if we want to deploy our driving executive on reasonable hardware.

Generating policy using our method is efficient in space because it only considers one scenario that is currently encountered and does not require any hardware space to store extra information. However, it needs to build a search tree from scratch during each planning step, which can be time-consuming.

On the other hand, iterative RAO* aims to improve the computational time by generating a number of policies offline that cover all possible scenarios that the ego vehicle would encounter, where each scenario is dependent on the actions that the agent vehicle would take. Given a comprehensive library of offline policies, it only needs to look up the corresponding tree given the agent vehicle's intentions during the planning step. Unfortunately, because the probability distribution over the agent vehicle's intentions is continuous, we have to discretize them so that we only need to store a finite number of trees. Imagine in a simplified scenario where there is only one agent vehicle that chooses from two different maneuvers and two different driving styles, and each maneuver is divided into 20 steps (e.g., takes 20 clocks to finish). Thus, the total number of unique intentions is 80 for the agent vehicle. If we discretize the probability by 1%, which means that the percentage of each discrete state can only be integers, we want to know the number of offline policies that need

to be generated and stored. This is equivalent to find the number of non-negative integer solutions to the equation:

$$\sum_{i=1}^{80} x_i = 100, \tag{7.6}$$

where $0 \leq x_i \leq 100$ for all i.

The solution to the above problem is $\binom{100+80-1}{80-1} = \binom{179}{79} = 1.34e+52$, which means that we need to store $1.34e+40$ terabytes (TBs) of offline data if one policy tree only takes one byte of space. Even if we have enough storage space, it is not realistic to generate these trees with a reasonable amount of time in advance.

Additionally, the changing intentions of the agent vehicles restrict the amount of search graph that can be reused for iterative RAO*, which means in many cases we need to build a new graph from scratch. For example, if we use an offline policy that assumes the agent vehicle is making a left turn and suddenly detects that that vehicle is going forward instead, our policy tree needs to be re-evaluated drastically in the next step, by removing the left turn branch and expanding the going forward branch. Therefore, we decide not to iterative RAO* in this thesis because of the above limitations.

A challenge to run RAO* on the fly is that we need to make sure that it can produce results fast enough within the control cycle. To speed up RAO*, we have used a few other techniques in addition to the two heuristics used in the algorithm. First, the algorithm only considers the agent vehicle that is close enough to the ego vehicle, as the vehicles too far away do not impose much risk. This helps reduce the number of branches in the search tree as well as the time to compute near collision probability. Additionally, we use an action model that is similar to what is described in Section 6.2.1 for the ego vehicle, and ask the RAO* algorithm to only consider actions for the ego vehicle that satisfy the preconditions. Using these techniques, we show in Chapter 8 that our approach is able to achieve online planning in dynamic scenarios with multiple agent vehicles. In cases where the planning time is very restricted, we can possibly develop an anytime RAO* algorithm that first generates a

feasible solution quickly, and then updates the solution to achieve better performance when time permits.

In general, the online intention-aware planning ability allows our approach to be space efficient and make the adjustment quickly, while always having a risk-bounded plan when the surrounding agent vehicles change their intentions. This is a major advantage of our approach, as compared to other works using offline policy or assuming a fixed distribution over the intentions of dynamic agent vehicles.

7.5 Future Work for Maneuver Planning

One major drawback of RAO* is its scalability because of the large search space in a POMDP problem. Therefore, we would like to explore sampling-based planning methods in the future to improve the computational time when we need to generate plans for an ego vehicle that navigates with a handful of agent vehicles and pedestrians.

7.5.1 Sampling-Based Maneuver Planning

In this thesis, we have explored many methods to reduce computational time for our search-based POMDP solver. However, these methods are domain-specific and cannot be generalized to all problem instances. Furthermore, although they can find exact solutions, tree search methods suffer from the curse of dimensionality. In the worst case, they have to explore the entire search space, which can be exponential in terms of the depth of the tree. This can be a problem if we want to include more moving obstacles, such as pedestrians, in the future, because the number of pedestrians can be large in a crowded environment.

Instead of pruning the search space, we would like to investigate sampling-based methods that find approximate solutions with more efficiency in time. For instance, we can utilize Monte Carlo simulation to sample the behavior of each agent in the environment. This method has demonstrated great success in reinforcement learning. One of its most accomplishment is to help an artificial intelligence system called

AlphaGo dominate the most challenging classic games, Go, in the world [46].

7.6 Summary

In this chapter, we complete our driving executive by using a maneuver planning algorithm that models the problem using CC-POMDP-MA and then solves it using an RAO* solver iteratively until the ego vehicle reaches its goal. Next, we would like to demonstrate our executive in a set of challenging environments in the following chapter.

Chapter 8

Experimental Results

In this chapter, we demonstrate the performance of the proposed intention-aware driving executive iGeordi through a number of experiments in simulation and with a naturalistic driving dataset. We start by comparing a few candidate simulators before we decide to use the CARLA simulator, because of its accessibility. In the CARLA simulator, we design and implement two challenging and important scenarios in day-to-day driving: unprotected intersection left turn and lane change with multiple vehicles. In each scenario, we validate different pieces of our algorithm and compare them with the state of the art to show the advantage of our approach. In the end, we evaluate our driving style estimation algorithm in a large-scale naturalistic driving dataset.

8.1 Simulator Description

In order to evaluate the proposed approach effectively and systematically, it is crucial to build test cases where we can control an ego vehicle to navigate in a dynamic environment. One way is to implement our approach in a physical vehicle and test it in the real world. Although real-world experiments are able to demonstrate the success of our driving executive in a more convincing way, they require a large number of resources, because building (or buying) and maintaining a vehicle that is able to collect the necessary data can be very expensive and time-consuming. Therefore, we

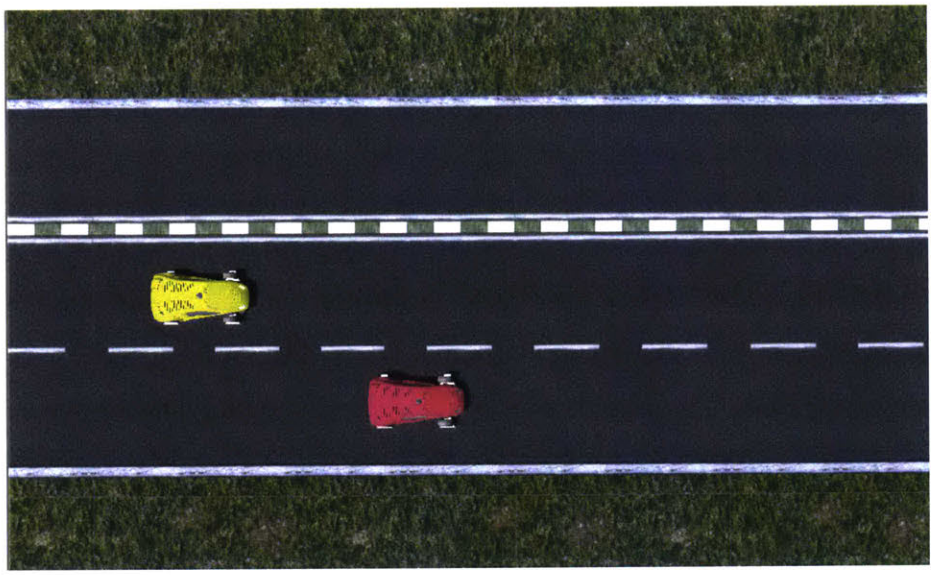


Figure 8-1: Top-down view of a simulated Unity driving scene.

decide to use a simulator with high fidelity that provides a realistic driving experience with little cost. Another advantage of using a simulator is that it allows us to place agent vehicles in designated places to simulate risky cases that are usually rare and hard to collect in real-world driving.

We have explored two candidate simulators in this thesis: the Unity simulator and the CARLA simulator. After comparing these two simulators in various ways, we have concluded that CARLA is better for our purpose because of its accessibility.

The Unity platform is widely used for building computer and video games with its 3D engine. It provides the user with a lot of power and freedom to create almost any driving scenarios through many built-in or third-party packages that model different physical objects. For instance, we can simulate a three-lane driving scene in Figure 8-1 with two vehicles created by the *SkyCar* package and a road environment created by the *EasyRoad3D* package. The simulator also models various physical aspects of vehicles, such as collisions and frictions, through its powerful physics engine. On the other hand, it is limited in a number of ways. First, there are many coordinate frames for different objects in Unity, and it is nontrivial to transform between different frames efficiently. The multiple frames also create a problem when we want to point our camera at the ego vehicle with consistent views for a better driving experience.

Second, since Unity is only supported on Windows and Mac systems, it is hard to wire software and algorithm packages on Linux platforms to the simulator. Last but not least, the software is not free, which can be problematic if we want to share our work with the outside research community.

Fortunately, we find the CARLA simulator [18] to be a good replacement because it is more accessible and does not have the limitations listed for Unity. CARLA is a state-of-the-art urban driving simulator that produces a realistic driving experience in a few small towns with traffic control (see Figure 8-2) and allows users to create multiple vehicles based on their needs. Each vehicle can be controlled individually through steering wheel angle and throttle inputs collected from either a user input device, such as a keyboard or programmed software that generates the control given the environment information. Although we do not have as much freedom to create any driving environment as we have in Unity, the three town environments in CARLA cover most driving scenarios of interest. Furthermore, users can add random vehicles and pedestrians in CARLA with reasonable behaviors and change the traffic lights periodically to make the simulator more realistic. Lastly, CARLA is open-source software that is in rapid development supported by many industry leaders, such as Toyota, General Motors, and Intel. It has become a very popular and recognizable platform for data collection and performance testing in the autonomous driving community.

In summary, because of the accessibility and better community support for the CARLA simulator, we decide to use it to collect data to learn the motion models, train the intent recognition estimators, and check the performance of our proposed intention-aware driving executive. In the rest of this chapter, we present three scenarios, by first describing the simulator setup and then demonstrating the results of different pieces of our system.

Figure 8-2: Top view of part of a CARLA town.

8.2 Intersection Left-Turn

According to [10], intersections are responsible for 40% of crashes occurred in the United States. Among all kinds of crashes happening at an intersection, unprotected left-turn is one of the most common and dangerous ones. Therefore, we are interested in validating our intention-aware driving executive in a scenario where the ego vehicle needs to make a left turn at an unprotected intersection when an agent vehicle is approaching from the opposite direction. We show that our approach is able to handle this challenging environment and outperform other baseline methods. Furthermore, we show that by tuning the upper bound on the risk-constraint, our approach can simulate different driving styles for the ego vehicle.

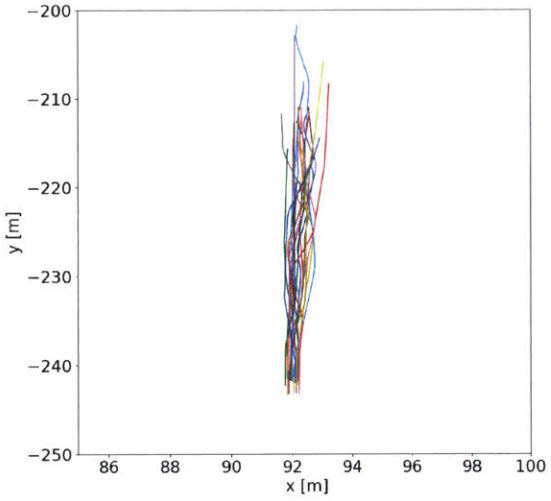
8.2.1 Scenario Setup

As shown in Figures 8-2 and 8-3, we start the test by placing the yellow ego vehicle in front of an intersection, and the red agent vehicle on the opposite side. The ego vehicle has a goal to turn left. At each step, it can choose to stop to yield to the

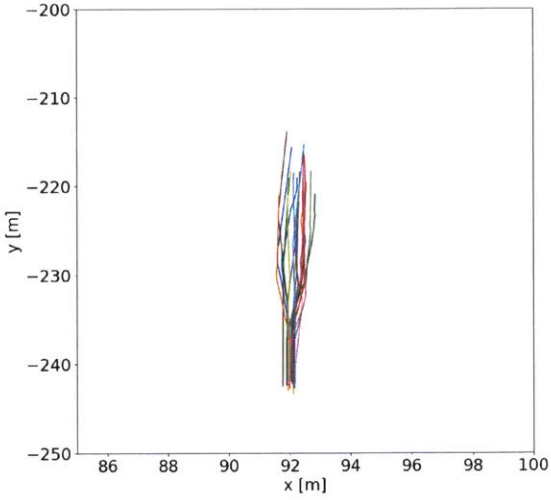
(a) View of the ego vehicle. (b) View of the agent vehicle.

Figure 8-3: Intersection left-turn scenario setup.

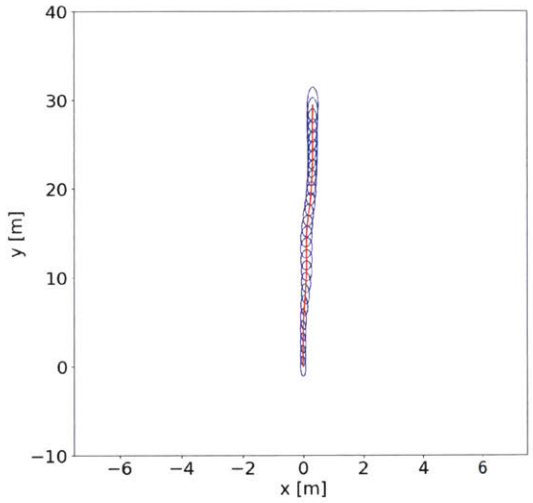
agent vehicle or perform a left-turn. In order to encourage turning sooner than later, we add a higher cost to the stop action. On the other hand, the agent vehicle chooses randomly from two maneuvers, including going forward or slowing down to yield to the ego vehicle. Although there are more maneuvers that can be considered in this situation, we would like to focus on the most common ones for simplicity. During the experiment, we observe from driving data that people with different driving styles usually execute each maneuver similarly (e.g., the covariances are small in the PFT model), hence we simplify this scenario by considering only one unique driving style and focus mostly on maneuver estimation. More discussions on the driving style estimation will be introduced in Section 8.3. Additionally, we use a value of 10 for the size of history data points w, and a value of 0.0001 for ϵ, as introduced in Chapter 6 used for discrete size pruning. As described in Section 4.3.1, we assume that the ego vehicle has access to the exact locations of all agent vehicles, since the major uncertainty comes from the maneuvers and driving styles of the surrounding vehicles.



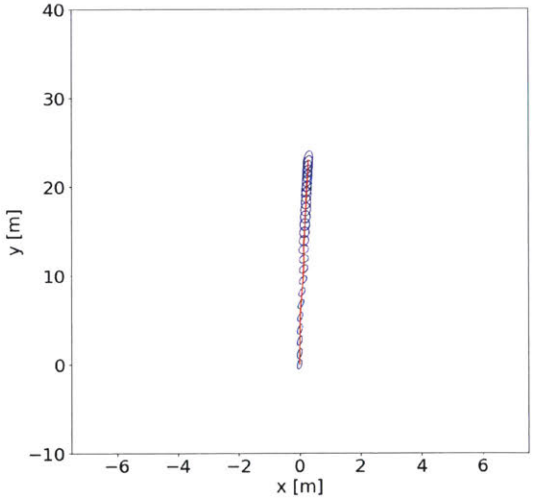
(a) Demonstrating forward trajectories.



(b) Demonstrating slow down trajectories.



(c) Forward PFT.



(d) Slow down PFT.

Figure 8-4: Trajectories (top row) and probabilistic flow tubes (bottom row) for maneuvers taken by the agent vehicle in the intersection left-turn scenario. In the PFT plots, the means (or nominal trajectories) are plotted in red, and the ellipses in blue represent one standard deviation at each step.

8.2.2 Data Collection

Given pre-defined maneuvers, we first generate a number of demonstration trajectories for each maneuver, by asking the human driver to teleop the agent vehicle for many times in the CARLA simulator using a Logitech steering wheel and pedal device with

force feedback. In each test, the agent vehicle starts at a fixed location with a certain distance to the intersection, and we start to record the trajectory once the vehicle reaches a stable speed (e.g., 40 kilometers per hour) and end recording after a fixed amount of time. The demonstration trajectories for the two maneuvers can be found on the top row of Figure 8-4. Note that trajectories are tilted towards the right side slightly because the road geometry in the simulator is not perfectly aligned with the y-axis.

8.2.3 Motion Learning

Given the demonstration trajectories, we learn a library of PFTs representing the uncertainties of the forward and slow down maneuvers using Algorithm 4 in Chapter 5. As a reminder, we shift the beginning of each PFT to the origin for consistency as shown on the bottom row in Figure 8-4. We observe that given a fixed amount of time, going forward maneuver travels further than a slowing down maneuver. Additionally, the gap between two consecutive steps in the slowing down PFT starts to decrease when time increases, which makes sense because the vehicle is moving slowly towards the intersection.

8.2.4 Intent Recognition Results

We investigate the performance of our intent-recognition algorithm, by separately measuring the accuracy of maneuver estimation and trajectory prediction. Since there is only one driving style behind the maneuvers, it is trivial to estimate the driving style in this scenario.

To measure the maneuver estimation accuracy, we compute the percentage of correctly identified maneuvers among the testing set. Since the intent recognition algorithm outputs a probability distribution over possible maneuvers, we choose the most likely maneuver and compare it with the ground truth maneuver. To measure the motion prediction performance, we sample from the predicted trajectory distribution and compute the average end displacement error over 4.8 seconds among all

samples. The predicting horizon is chosen as the time taken by the ego vehicle to make a left turn.

Among all the testing examples, our intent estimation has an average accuracy of 96.19%, and the average end displacement error is 2.02 meters. We observe that most maneuver misclassifications happen where a forward maneuver is misclassified as a slow maneuver because the forward trajectory contains a sequence of small gaps between consecutive positions. Such gaps confuse the maneuver estimator to produce a distribution that assigns more probability to the slow down maneuver and causes a large trajectory prediction error as a side effect. In the future, we would like to collect more trajectories to include these edge cases in our PFT representation.

8.2.5 Maneuver Planning Results

Next, we check the performance of our proposed intention-aware maneuver planner after recognizing the intentions of surrounding agent vehicles. In addition to our planner, we introduce three baseline methods for comparison. The first two planners push the extremes of confidence on the maneuvers estimated: complete uncertainty and complete confidence. The first baseline planner assumes that the agent vehicle will choose each action with equal likelihood. That is, the ego vehicle is maximally uncertain about the course of action performed by the agent vehicle, and hence will be conservative. The second baseline assumes that the agent vehicle is slowing down with 100% probability, and thus always produces a plan to turn immediately. The first two baseline planners all use the RAO* planner to find the policy offline. Instead of using a risk-bounded planner, the third baseline planner chooses the action for the ego vehicle based on the acceleration of the agent vehicle. If the acceleration becomes smaller than a negative threshold, it assumes the agent vehicle is slowing down and executes the ego vehicle to make a left turn.

We test each planner with 1000 trials, ask the agent vehicle to execute each maneuver randomly. Figure 8-5 shows one example where our approach successfully identifies the intentions of the agent vehicle and drives the ego vehicle to the destination efficiently. During the tests, we use two measurements to quantify the perfor-

Figure 8-5: Our intention-aware driving executive in an unprotected left turn test, where the executive recognizes that the agent vehicle is slowing down and controls the ego vehicle to turn left.

	Success Rate [%]	Completion Time [s]
Conservative	**100.00**	9.60
Risky	56.80	**4.80**
AccelerationBased	82.10	8.87
Ours ($\Delta = 0.1\%$)	**100.00**	8.64

Table 8.1: Performance of different planners in the unprotected intersection left turn scenario.

mance in terms of safety and efficiency: the percentage of collision-free turns and the average time to make turns among all collision-free turns.

Table 8.5 summarize the performance of all planners in the unprotected left turn scenario. We can see that the first baseline method is quite successful in avoiding all potential near collisions based on a conservative assumption. However, it sacrifices the efficiency by keeping the ego vehicle waiting even when the agent vehicle slows down and yields. On the other hand, the second baseline is efficient in making turns using the smallest amount of completion time, but leads to crashes almost half of the time because of its over-risky assumption. The acceleration-based method works in more than 80% of the test cases, but it is vulnerable in examples where the agent vehicle decelerates slightly in the middle of a forward maneuver. Compared to the methods above, our model is able to find safe plans in all cases, by allowing only a small risk bound of 0.1%, and completes the tasks more efficiently, by detecting the slow down intention of the agent vehicle quickly and reliably.

We further analyze the performance of our planner, by changing the risk bound Δ. As plotted in Figure 8-6, we see that when the upper risk bound is relatively small,

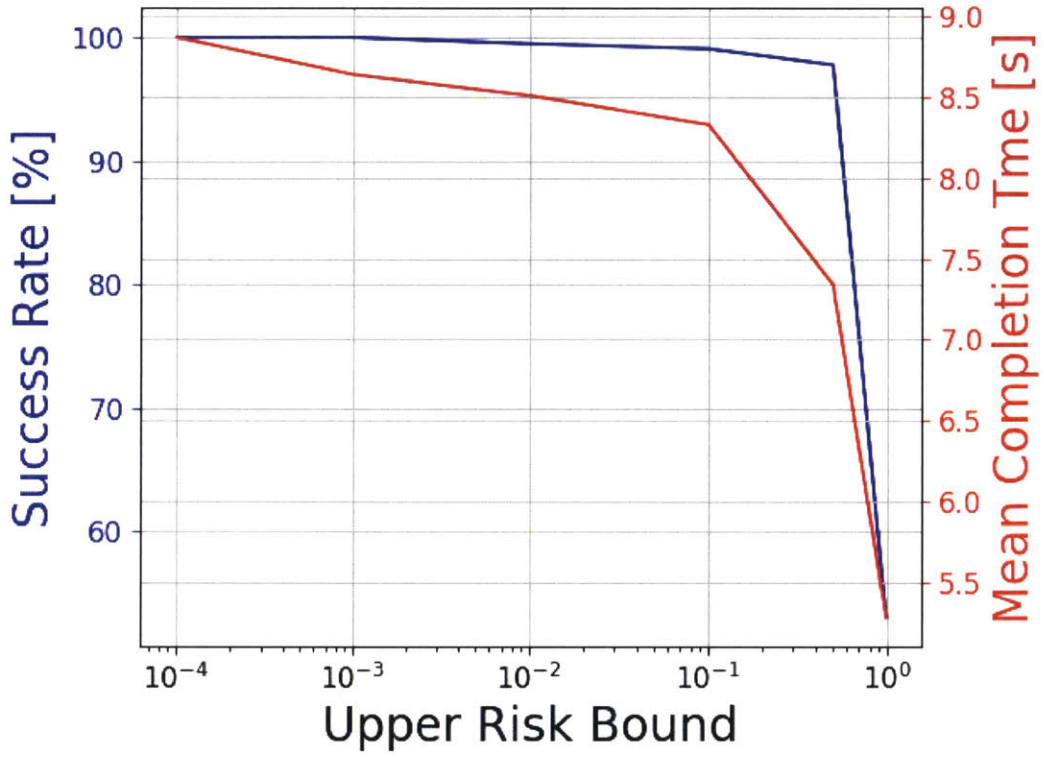


Figure 8-6: The performance of our planner in the intersection left-turn scenario as a function of the upper risk bound Δ, where the x-axis is plotted in the log scale. The performance is similar to the conservative approach at the left end, and the risky approach at the other end.

the planner can guarantee to find collision-free policies for the ego vehicle because it is very rare that our estimator classifies a forward action into a slow down action for the agent vehicle with great confidence. However, the smaller the risk bound, the more time it takes the ego vehicle to complete the task as the planner needs more time to be certain that it is within the safe bound to make a left turn. When the risk bound becomes larger, the success rate starts to drop because in some cases the intent recognizer provides inaccurate estimations, which leads our planner that takes large risk bound to choose the wrong action. When the risk bound is set to one, the behavior of the planner is essentially identical to the second baseline planner that tends to take risky decisions.

We can tune the *style* of our planner using different risk bound values, where a small risk bound leads to safer behaviors, and a large risk bound produces a risky

planner that is willing to sacrifice safety in order to achieve better efficiency.

8.3 Lane Change with Multiple Agent Vehicles

In this scenario, we test our planner in a more challenging dynamic scenario where the ego vehicle interacts with more than one agent vehicle on a busy road and needs to make lane changes to achieve better efficiency. Each agent vehicle can choose to execute maneuvers with two different driving styles, hence in this scenario, we focus on demonstrating the performances on estimating their driving styles. Similar to the unprotected left turn scenario, we compare our intention-aware driving executive with other baseline methods and show its advantage in terms of both safety and efficiency. Finally, because there are multiple agent vehicles in this scenario, we provide an analysis of the computational time of the planner, by varying the number of agent vehicles and the planning horizon.

8.3.1 Scenario Setup

As shown in Figure 8-7, we place the ego vehicle in the middle of a two-lane road at the beginning of each test. It can choose to change lane or go forward with two velocity options, and its goal is to reach the end of the road as soon as possible without near collisions. The cost of each action is based on its execution time. In addition, we place three agent vehicles randomly next to the ego vehicle, each of which can choose to change lane or go forward. Each agent vehicle is initialized with either a normal driving style or an aggressive driving style, and a normal vehicle will execute the same maneuver differently compared to an aggressive vehicle. The agent vehicles are designed to move more slowly than the ego vehicle, which encourages the ego vehicle to bypass them by changing lanes. Similarly to what was done in Section 8.2.3, a set of six PFTs with all possible combinations of maneuvers and driving styles are learned from human demonstrations.

The starting location of each agent vehicle is chosen carefully so that they can create potential risks to the ego vehicle. Figure 8-7 shows one of the testing examples,

Figure 8-7: A lane-change test example, where the yellow vehicle is the ego vehicle, and the red vehicles are the agent vehicles.

where an agent vehicle is placed behind the yellow ego vehicle in the adjacent lane to prevent the ego vehicle from making a lane change at the beginning. The other two agent vehicles are placed in the front on different lanes to create potential risks if the ego vehicle does not plan carefully. From many experiments, we find three cars are sufficient to cover most interesting lane change scenarios, since a vehicle interacts with at most three cars in proximity during a lane change.

8.3.2 Intent Recognition Results

In this scenario, we focus on demonstrating the performance of the driving style estimator as there are two possible styles associated with each agent vehicle. We then show the intent recognition results over the agent vehicles after driving styles are successfully estimated.

Driving Style Estimation Results

In order to obtain a successful driving style estimator, it is important to choose the features that can be used to distinguish between a normal style and an aggressive style. In this scenario, we select the features listed in Table 8.2, which includes the vehicle states, such as velocities and accelerations, and the environmental states, such as distances to the lane center and other vehicles. For each selected feature, we

Feature Name	Description
v_x	Longitudinal velocity
v_y	Latitudinal velocity
a_x	Longitudinal acceleration
a_y	Latitudinal acceleration
θ	Yaw angle
w_θ	Angular Velocity
d_l	Distance to lane center
d_{v1x}	Longitudinal distance to the closest vehicle
d_{v1y}	Latitudinal distance to the closest vehicle
d_{v2x}	Longitudinal distance to the second closest vehicle
d_{v2y}	Latitudinal distance to the second closest vehicle

Table 8.2: Features used in the multinomial logistic regression classifier.

compute the standard deviation of that feature in the past five observations as an additional feature, as we believe that an aggressive driver tends to have more variance in those features compared to a normal driver.

After selecting the features, we collect the training data from many demonstrating driving trips. In order to keep the data on a similar scale, we normalize the vehicle states between -1 and 1, and the environmental states between 0 and 1. To train the multinomial logistic regression classifier and prevent overfitting, we partition the collected data into two parts randomly. The first part, which consists of 80% of the data, is training data. The second part, which consists of the rest of the data, is validation data. Ten random partitions are generated. For each partition, the logistic regression classifier runs on the training data and finds the optimal parameters, including coefficients and the intercept in Equation 6.6. The output parameters are then evaluated on the validation data by computing the prediction accuracy. In the final step, we choose the output parameters with the best performance that achieves an average prediction accuracy of 89.89%, which is defined as the percentage of correctly identified driving style among all testing samples. Since the classifier outputs the probability likelihood given each sample, we choose the maneuver with the maximum

	Actual = Normal	Actual = Aggressive
Estimated = Normal	2326	331
Estimated = Aggressive	109	1588

Table 8.3: Confusion matrix on estimating two driving styles using the multinomial logistic regression classifier.

likelihood as the estimated result.

To determine how our classifier works on estimating each driving style, we summarize the estimation results in the confusion matrix as shown in Table 8.3, where each row represents the instances in a predicted class, while each column represents the instances in an actual class. The confusion matrix indicates that when the actual driving style is normal, our classifier performs well with only 4.48% misclassification rate. However, the aggressive driving style is more difficult to classify, because in many cases an aggressive driver behaves similarly to a normal driver when driving forward.

A similar conclusion can be reached by plotting the histograms of estimation probability for each driving style in Figure 8-8. The top histogram indicates that among all samples labelled with normal style, our classifier is able to confidently identify the correct label with more than 60% probabilities in most cases. Even in some cases where samples are misclassified as an aggressive style, the probability of being a normal style driver is close to 50%. On the other hand, when the actual label is aggressive, our classifier is confused by some samples that behave like a normal style driver and assigns a low probability in those cases.

Finally, we check the contribution of different features, by training a similar multinomial logistic regression classifier using only vehicle states and a classifier using only vehicle states and environmental states. The accuracy for each classifier is recorded in Table 8.4. The performance is increased by 4.98% after we add the environmental states, since they provide more information on how the driver reacts to other vehicles. For instance, an aggressive driver tends to keep a shorter distance to nearby vehicles. Furthermore, adding the standard deviation features also helps boost the estimation

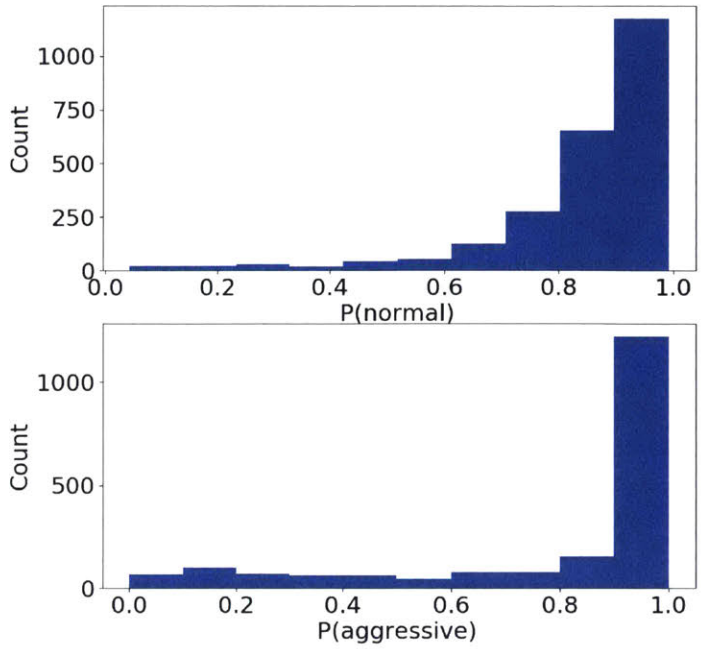


Figure 8-8: Estimation probability histograms. Top: Histogram of probabilities of being estimated to a normal style among all samples with normal driving style. Bottom: Histogram of probabilities of being estimated to an aggressive style among all samples with aggressive driving style.

Feature Set	Prediction Accuracy
Vehicle	79.97%
Vehicle+Environmental	84.92%
Vehicle+Environmental+StandardDeviation	89.89%

Table 8.4: Prediction accuracies in a lane change scenario with different features subsets.

accuracy because they provide more time-dependent cues on how a vehicle drives. For instance, an aggressive driver tends to change their acceleration more often or oscillates more around the lane center when following the lane than a normal driver.

In order to achieve online planning, in this scenario, we assume a binary driving model with only two driving styles to reduce the branching factor in the RAO* search tree. Although such an assumption would simplify the style estimation problem as a

Figure 8-9: Our intention-aware driving executive in a lane change test, where the executive recognizes the intentions of the agent vehicles and controls the ego vehicle to make a lane change. The risk bound is set to 1%, which makes our ego vehicle aggressive in this test.

binomial classification task, it still helps us demonstrate the basic capability of our approach. In Section 8.3.4, we use a driver model with more than two styles and a much larger naturalistic driving dataset to validate our driving style estimator.

Maneuver Estimation Results

After measuring the performance of driving style estimation, we compute the average maneuver estimation accuracy to be 98.9%, and the average trajectory prediction error to be 2.12 meters over a horizon of 4.8 seconds. We are able to achieve better maneuver estimation performance compared to intersection left-turn scenario as the lane change and forward maneuvers can be more easily distinguished in the lateral axis. Although accurate estimations over driving styles are helpful for reducing the trajectory prediction error, we do not see a lot of improvements on the trajectory predictor, as compared to the lane change scenario because a misclassified change-lane maneuver will result in a trajectory prediction in the opposite direction.

8.3.3 Maneuver Planning Results

Similar to the intersection left-turn scenario, we test our approach along with two baseline planners with 1000 trials. Figure 8-9 shows an example of our intention-aware driving executive working aggressively in a dynamic environment with two agent vehicles.

The results are summarized in Table 8.5. We see that our driving executive is

	Success Rate [%]	Completion Time [s]
Conservative	**100.00**	26.76
Risky	28.60	**19.20**
Ours ($\Delta = 0.1\%$)	**100.00**	24.62
Ours ($\Delta = 1\%$)	99.80	23.09

Table 8.5: Performance of different driving executives in the lane change scenario.

able to produce more efficient plans than the conservative planner, while maintaining safety all the time if the risk bound is set to be small. In addition, we can make an aggressive planner to further reduce the completion time, by increasing the risk bound. For example, compared to the conservative planner, our planner with a risk bound of 0.1% increases the efficiency in terms of completion time by 13.71%. Although the success rate is not perfect, our planner is able to respect the risk bound with guarantees. As a result, our driving executive exhibits performance similar to human drivers who usually balance safety and efficiency.

In this scenario, we choose a planning horizon of two because the planning time increases drastically with the horizon. In Figure 8-10, we run the planner on a computer with an Intel Core i7 processor and 39.2GB memory and plot the planning time in log scale as a function of the planning horizon for different number of agent vehicles, where each vehicle can choose from three actions and one planning horizon is equivalent to 4.8 seconds. The dashed black line indicates the maximum allowed planning time of 200 milliseconds if we want to keep the control frequency at 5Hz. With three agent vehicles and a planning horizon of two, it takes around 191 milliseconds for the planner to find a solution, which satisfies the planning time constraint.

Although a planning horizon of two is not ideal, our ego vehicle is able to make adjustments quickly if there exists possible risk further ahead thanks to the online capability of our planner. Furthermore, we observe that in many cases the ego vehicle only needs to interact with one or two vehicles. Therefore, we can adjust the planning horizon based on the number of interacting agents to improve the performance of the resulting plan.

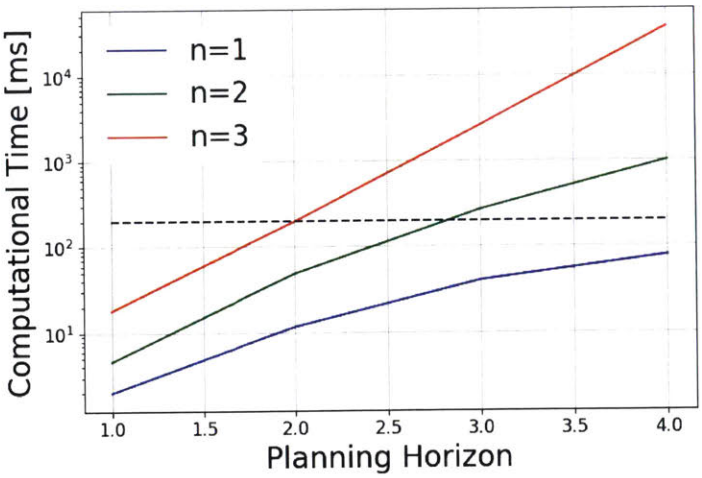


Figure 8-10: Planning time using the RAO* planner as a function of the planning horizon H and the number of agent vehicles n. The planning time is shown in log scale.

8.3.4 Driving Style Estimation on A Naturalistic Driving Dataset

In this section, our goal is to further validate the driving style estimator with more than two driving styles on a naturalistic dataset that was collected from the real-world driving experience.

The dataset used to evaluate the driving style estimator was extracted from the Second Strategic Highway Research Program (SHRP 2) Naturalistic Driving Study (NDS) trips [22, 7] [1]. In the SHRP 2 NDS study, about 3,400 drivers participated and drove in six cities across the United States, including Seattle, Tampa, Buffalo, Central Pennsylvania, Bloomington, and Durham. In this thesis, we choose a subset of the dataset consisting of 1,200 trips collected from 40 drivers, where each trip lasted for at least 10 minutes.

Each participating driver in the study was asked to enroll in a clinical attention-deficit/hyperactivity disorder (ADHD) screening assessment, and during the assessment they needed to answer a number of questions, such as how often they felt easily

[1]The findings and conclusions of this section are those of the author and do not necessarily represent the views of the VTTI, SHRP 2, the Transportation Research Board, or the National Academies.

Feature Set	Prediction Accuracy
Vehicle	42.65%
Vehicle+Environmental	49.81%
Vehicle+Environmental+StandardDeviation	51.26%

Table 8.6: Prediction accuracies on SHRP2 NDS dataset with different features subsets.

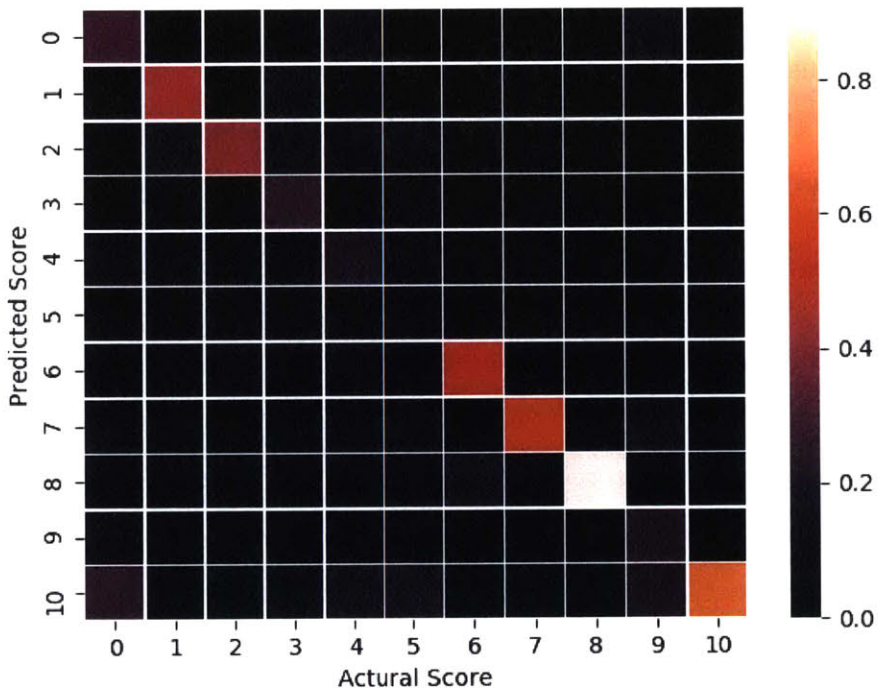


Figure 8-11: Normalized confusion matrix for the multinomial logistic regression classifier on the SHRP 2 NDS dataset.

distracted, how often they had difficulty waiting for a turn, and how often they felt restless. The answers to those questions are summarized as a Barkleys score ranging from 0 to 10 that measures the likelihood of having ADHD symptoms. Because Barkleys score covers many psychological states of the drivers, we decide to use it to represent different driving styles that need to be estimated.

Given the vehicle states and the environmental states collected in the SHRP 2 NDS study, we train a multinomial logistic regression classifier that provides a probability

distribution over all possible Barkleys score values. The training process is similar to that in Section 8.3.2, except the number of possible output values increases from 2 to 11. In order to keep a balanced dataset, the 40 drivers were selected such that the Barkleys scores are evenly distributed among those drivers.

The resulting estimator is able to achieve an estimation accuracy of 51.26%, which achieves better performance than a naive random estimator. The accuracy is lower than that in the simulated lane change scenario, because the driving styles are determined by the questionnaire answers, which can be subjective. Furthermore, the same driver may not exhibit the same driving style all the time when driving, which is one thing to consider for future work. Similarly to the lane change scenario, we verify that adding the environmental states and variance information helps improve the classification accuracy as shown in Table 8.6.

The normalized confusion matrix in plotted in Figure 8-11 as a heat map, which indicates that our estimator is able to achieve better accuracies when the Barkleys scores are high. On the other hand, when the scores are smaller or equal to 5, the estimator has difficulties finding the correct label. This can be explained by the fact the drivers with low Barkleys scores tend to behave more normal and similar, and the drivers with high scores usually behave more drastically and are thus easier to be distinguished. Although the estimation accuracy of 51.26% is not satisfactory, we argue that it is more important to detect aggressive drivers with high accuracies, as they usually introduce higher risks to our ego vehicle.

8.4 Summary

In this chapter, we show that our driving executive is able to achieve both safety and efficiency in different tasks in a number of testing environments. For instance, in the intersection left-turn scenario, the intent recognition algorithms are able to detect the correct maneuver 96.19% of the time and predict future trajectories with an error of 2.02 meters in terms of end displacement error. The maneuver planner is able to achieve a success rate of 100%, while improving the performance, in terms

of completion time to the goal, compared to a planner without intent recognition capability by 10%. In the lane-change scenario, the driving style estimation accuracy is 89.89%, the maneuver estimation accuracy is 98.9%, and the trajectory prediction error is 2.12 meters. The maneuver planner increases the efficiency by 13.71%.

As a result, we have demonstrated the capability of our driving executive iGeordi in this chapter. It exhibits performances similar to human drivers in two ways. First, by predicting the other agent vehicles' future motions in the environment, iGeordi balances safety and efficiency of its maneuver plan. Additionally, given a pre-determined risk bound, iGeordi will behave consistently as a safe driver or an aggressive driver, which makes it more predictable to other drivers. Future work includes comparing our intent recognition algorithms with state of the arts in a set of carefully designed experiments, identifying more useful metrics (such as how fast our intent recognition algorithms can detect the correct intentions) to further demonstrate the strength and weakness of our system, and testing iGeordi in real world experiments.

In the next chapter, we provide a series of discussions on different parts of our thesis, followed by a list of future work that we would like to accomplish.

Chapter 9

Conclusions

In this chapter, we conclude the thesis by summarizing our approach to intention-aware maneuver planning followed by discussions on different pieces in the approach. We then discuss future works that could extend the capability of the current approach.

9.1 Thesis Summary

In summary, we propose an intention-aware driving executive that generates risk-bounded maneuver plans for an autonomous vehicle driving in a dynamic environment, by considering the intentions of other surrounding vehicles, including driving style, maneuver type, and future trajectories. The intentions are recognized using a data-driven learning method and a Bayesian filter, and the recognition results are used to update the belief state in a multi-agent hybrid POMDP instance modeling the behavior of each vehicle in the driving environment. The POMDP instance is then solved by a state-of-the-art solver based on a risk-bounded heuristic forward search algorithm that utilizes a utility heuristic and a risk heuristic to find risk-bounded solutions efficiently.

We validate our driving executive in two challenging dynamic environments, and show that it outperforms baseline methods assuming static intentions over other agent vehicles. In addition, we demonstrate the performance of different pieces in our approach independently through a number of experiments in the CARLA simulator

as well as validate the driving style estimator on a naturalistic driving dataset. In the lane change scenario, the intent recognition algorithm achieves an average driving style estimation accuracy of 89.89%, an average maneuver estimation accuracy of 98.9%, and an average trajectory prediction error of 2.12 meters. The maneuver planner guarantees safety with respect to the safety constraint, while arriving at the goal 13.71% faster compared to a state-of-the-art planner without the intent recognition capability. Overall, the demonstration results show that intention-aware risk-bounded planning is necessary to enable safe driving in dynamic environments.

The key contribution of this thesis is proposing and implementing an idea of *proactive* driving by enabling the driving executive to reason and predict the behaviors of other agent vehicles before making decisions for the ego vehicle. Such idea achieves a balance between safety and efficiency, by estimating the risk in advance and taking efficient actions when risk is estimated to be small. We believe this idea of proactive planning has a great potential to help achieve fully autonomous driving because it imitates how human makes decisions by considering the uncertainties in advance.

9.2 Thesis Discussions

In this section, we provide discussions on various pieces of this work by identifying their strengths and directions for future work. We hope these discussions can serve as a guideline for those who want to follow and continue this line of work.

9.2.1 Motion Modeling

As discussed in Section 5.1, probabilistic flow tube is an efficient and simple tool to model continuous motions with uncertainties. By storing a sequence of Gaussian distribution parameters, such as means and covariances, a PFT keeps track of the uncertainties of a motion primitive at different stages. These parameters can not only be used to estimate the likely maneuvers given a sequence of observed trajectories, but also help predict the possible motions of the target agent in the future.

On the other hand, the PFT is limited in that it only extracts and stores spatial

information from demonstrating trajectories. It does not store velocities of the vehicle explicitly, which are important features if we want to improve the estimation and prediction performances. For instance, a vehicle travelling at different velocities may execute the same maneuver with different number of time steps. One possible extension to the flow tube representation used in this thesis is to incorporate velocities in it, as discussed in Section 5.3.1. Furthermore, the environmental states, such as distances to other vehicles, are not included in the flow tube representations. As presented in Section 8.3.2, such information is very helpful to determine the behaviors of the drivers. Therefore, despite the simplicity and efficiency of the motion model, its current formulation is not sufficient to store all the useful information that we desire.

9.2.2 Intent Recognition

In this thesis, we recognize the intentions of surrounding vehicles through maneuver-based motion prediction, in which we first estimate the likelihood over the agent vehicle's driving style and maneuver and then predict the trajectory based on the estimation results. Our current way of recognizing maneuvers does not consider the environmental states, which can be problematic because a vehicle typically drives based on its proximity and the surrounding environment. One possible future work is to incorporate environmental states when estimating maneuvers.

In addition, although maneuver-based method works quite efficiently if we assume that vehicle moves by following a sequence of maneuvers, it has a few challenges. First, it is challenging to define maneuvers given observed vehicle trajectories because there are many different types of vehicle maneuvers, and no standard definition exists for each maneuver. For instance, there are many ways to define the begin and the end of a turn left maneuver. One can define them based on the change of the steering wheel angle of the vehicle, and the other can use the distance to the crosswalk to find the start of that maneuver. Unfortunately, a method that works on one definition may fail to estimate maneuvers defined using another definition. Second, even if there exist standard definitions on all kinds of maneuvers, it is time-consuming to

label the dataset with maneuvers. One possible solution is to unsupervised learning approaches such as expectation minimization (EM) algorithm to cluster the input trajectories into different maneuvers, which can prevent us from hand defining and labeling maneuvers.

9.2.3 Maneuver Planning

By traversing through the search tree, the Risk-Bounded AO* algorithm is able to find an exact solution that ensures the risk of near collision with other vehicles is upper bounded by a safety constraint. As a result, our driving executive is able to provide a guarantee over the performance of the resulting plan, and the user can fully trust our executive after specifying a risk bound.

On the other hand, although the RAO* algorithm uses a utility heuristic and a risk heuristic to speed up the search by searching towards promising directions and pruning over-risky space, respectively, the search time still grows exponentially with respect to the depth of the tree in the worst case. As a consequence, it can fail to find solutions in real time if we add more vehicles and pedestrians in the environment, because the branching factor in the search tree increases linearly with the number of moving obstacles. As a result, it is necessary to reduce computational time by making approximations. Instead of traversing the entire search tree, we can use sampling-based methods to speed up the search as discussed in Section 7.5.1.

9.3 Future Work: Beyond Autonomous Vehicles

In this thesis, we have discussed possible future work relevant to each approach in Chapters 5 through 7. Overall, our system focuses on applications of autonomous vehicles because of the urgent need to improve transportation given the increasing number of traffic accidents. On the other hand, our approach is not limited to autonomous driving and can be extended to other problem domains where an intelligent agent needs to interact with other dynamic agents that take stochastic actions.

One excellent example is home robots whose goal is to assist human beings in

household tasks. By recognizing the intentions of their homeowners, the home robot can avoid potential risks of near collision with them and collaborate with them more efficiently. Therefore, we wish to expand our work to such domain, by estimating the intentions of the other humans in the environment and making safe plans accordingly. We believe our approach is advantageous in such tasks because it is able to establish trust between the homeowners and the robot as the approach is aware of the risk from the environment.

Bibliography

[1] Zlatan Ajanovic, Bakir Lacevic, Barys Shyrokau, Michael Stolz, and Martin Horn. Search-based optimal motion planning for automated driving. *arXiv preprint arXiv:1803.04868*, 2018.

[2] Ahmad Aljaafreh, Nabeel Alshabatat, and Munaf S Najim Al-Din. Driving style recognition using fuzzy logic. In *Vehicular Electronics and Safety (ICVES), 2012 IEEE International Conference on*, pages 460–463. IEEE, 2012.

[3] Haoyu Bai, Shaojun Cai, Nan Ye, David Hsu, and Wee Sun Lee. Intention-aware online pomdp planning for autonomous driving in a crowd. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 454–460. IEEE, 2015.

[4] Shray Bansal, Akansel Cosgun, Alireza Nakhaei, and Kikuo Fujimura. Collaborative planning for mixed-autonomy lane merging. *arXiv preprint arXiv:1808.02550*, 2018.

[5] Holger Berndt, Jorg Emmert, and Klaus Dietmayer. Continuous driver intention recognition with hidden markov models. In *Intelligent Transportation Systems (ITSC), 2008 11th IEEE International Conference on*, pages 1189–1194. IEEE, 2008.

[6] Sebastian Brechtel, Tobias Gindele, and Rüdiger Dillmann. Probabilistic decision-making under uncertainty for autonomous driving using continuous pomdps. In *Intelligent Transportation Systems (ITSC), 2014 17th IEEE International Conference on*, pages 392–399. IEEE, 2014.

[7] Kenneth L Campbell. The shrp 2 naturalistic driving study: Addressing driver performance and behavior in traffic safety. *TR News*, (282), 2012.

[8] German Castignani, Raphaël Frank, and Thomas Engel. Driver behavior profiling using smartphones. In *Intelligent Transportation System(ITSC), 2013 16th IEEE International Conference on*, pages 552–557. IEEE, 2013.

[9] Min Chen, Emilio Frazzoli, David Hsu, and Wee Sun Lee. Pomdp-lite for robust robot planning under uncertainty. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 5427–5433. IEEE, 2016.

[10] EH Choi. Crash factors in intersection-related crashes: An on-scene perspective (no. dot hs 811 366). *US DOT National Highway Traffic Safety Administration*, 2010.

[11] Nachiket Deo and Mohan M Trivedi. Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms. *arXiv preprint arXiv:1805.05499*, 2018.

[12] Shuonan Dong and Brian Williams. Motion learning in variable environments using probabilistic flow tubes. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1976–1981. IEEE, 2011.

[13] Shuonan Dong and Brian Williams. Learning and recognition of hybrid manipulation motions in variable environments using probabilistic flow tubes. *International Journal of Social Robotics*, 4(4):357–368, 2012.

[14] Dominik Dörr, David Grabengiesser, and Frank Gauterin. Online driving style recognition using fuzzy logic. In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pages 1021–1026. IEEE, 2014.

[15] Anup Doshi and Mohan M Trivedi. Examining the impact of driving style on the predictability and responsiveness of the driver: Real-world and simulator analysis. In *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pages 232–237. IEEE, 2010.

[16] Anup Doshi and Mohan M Trivedi. Tactical driver behavior prediction and intent inference: A review. In *Intelligent Transportation Systems (ITSC), 2011 14th IEEE International Conference on*, pages 1892–1897. IEEE, 2011.

[17] Anup Doshi and Mohan Manubhai Trivedi. On the roles of eye gaze and head dynamics in predicting driver's intent to change lanes. *IEEE Transactions on Intelligent Transportation Systems*, 10(3):453–462, 2009.

[18] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[19] Haluk Eren, Semiha Makinist, Erhan Akin, and Alper Yilmaz. Estimating driving behavior by a smartphone. In *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pages 234–239. IEEE, 2012.

[20] Michael Everett, Yu Fan Chen, and Jonathan P How. Motion planning among dynamic, decision-making agents with deep reinforcement learning. *arXiv preprint arXiv:1805.01956*, 2018.

[21] David Hallac, Abhijit Sharang, Rainer Stahlmann, Andreas Lamprecht, Markus Huber, Martin Roehder, Jure Leskovec, et al. Driver identification using automobile sensor data from a single turn. In *Intelligent Transportation Systems*

(ITSC), 2016 19th IEEE International Conference on, pages 953–958. IEEE, 2016.

[22] Jonathan M Hankey, Miguel A Perez, and Julie A McClafferty. Description of the shrp 2 naturalistic database and the crash, near-crash, and baseline data sets. Technical report, Virginia Tech Transportation Institute, 2016.

[23] Trong Nghia Hoang and Kian Hsiang Low. A general framework for interacting bayes-optimally with self-interested agents using arbitrary parametric model and model prior. In *IJCAI*, pages 1394–1400, 2013.

[24] Michael W Hofbaur and Brian C Williams. Mode estimation of probabilistic hybrid systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 253–266. Springer, 2002.

[25] Andreas Hofmann and Brian Williams. Exploiting spatial and temporal flexibility for plan execution of hybrid, under-actuated systems. In *AAAI Conference on Artificial Intelligence*, 2006.

[26] Adam Houenou, Philippe Bonnifait, Véronique Cherfaoui, and Wen Yao. Vehicle trajectory prediction based on motion model and maneuver recognition. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 4363–4369. IEEE, 2013.

[27] Xin Huang, Ashkan Jasour, Matthew Deyo, Andreas Hofmann, and Brian C Williams. Hybrid risk-aware conditional planning with applications in autonomous vehicles. In *Decision and Control (CDC), 2018 IEEE International Conference on*. IEEE, 2018.

[28] Xin Huang, Stephen McGill, Brian C Williams, Luke Fletcher, and Guy Rosman. Uncertainty-aware driver trajectory prediction at urban intersections. *arXiv preprint arXiv:1901.05105*, 2019.

[29] Ashkan Jasour, Andreas Hofmann, and Brian C Williams. Moment-sum-of-squares approach for fast risk estimation in uncertain environments. *arXiv preprint arXiv:1810.01577*, 2018.

[30] Derick A Johnson and Mohan M Trivedi. Driving style recognition using a smartphone as a sensor platform. In *Intelligent Transportation Systems (ITSC), 2011 14th IEEE International Conference on*, pages 1609–1615. IEEE, 2011.

[31] Dietmar Kasper, Galia Weidl, Thao Dang, Gabi Breuel, Andreas Tamke, Andreas Wedel, and Wolfgang Rosenstiel. Object-oriented bayesian networks for detection of lane change maneuvers. *IEEE Intelligent Transportation Systems Magazine*, 4(3):19–31, 2012.

[32] Stefan Klingelschmitt, Matthias Platho, Horst-Michael Groß, Volker Willert, and Julian Eggert. Combining behavior and situation information for reliably

estimating multiple intentions. In *Intelligent Vehicles Symposium (IV), 2014 IEEE*, pages 388–393. IEEE, 2014.

[33] Stéphanie Lefèvre, Dizan Vasquez, and Christian Laugier. A survey on motion prediction and risk assessment for intelligent vehicles. *Robomech Journal*, 1(1):1, 2014.

[34] Steven James Levine and Brian Charles Williams. Watching and acting together: Concurrent plan recognition and adaptation for human-robot teams. *Journal of Artificial Intelligence Research*, 63:281–359, 2018.

[35] Guofa Li, Shengbo Eben Li, Yuan Liao, Wenjun Wang, Bo Cheng, and Fang Chen. Lane change maneuver recognition via vehicle state and driver operation signalsresults from naturalistic driving data. In *Intelligent Vehicles Symposium (IV), 2015 IEEE*, pages 865–870. IEEE, 2015.

[36] Yuanfu Luo, Panpan Cai, Aniket Bera, David Hsu, Wee Sun Lee, and Dinesh Manocha. Porca: Modeling and planning for autonomous driving among many pedestrians. *IEEE Robotics and Automation Letters*, 3(4):3418–3425, 2018.

[37] Cory Myers, Lawrence Rabiner, and Aaron Rosenberg. Performance tradeoffs in dynamic time warping algorithms for isolated word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(6):623–635, 1980.

[38] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):33–55, 2016.

[39] Pascal Poupart, Aarti Malhotra, Pei Pei, Kee-Eung Kim, Bongseok Goh, and Michael Bowling. Approximate linear programming for constrained partially observable markov decision processes. In *AAAI Conference on Artificial Intelligence*, pages 3342–3348, 2015.

[40] Miquel Ramírez and Hector Geffner. Plan recognition as planning. In *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

[41] Andrey Rudenko, Luigi Palmieri, and Kai O Arras. Predictive planning for a mobile robot in human environments. In *IEEE Int. Conf. on Robotics and Automation (ICRA), Workshop on PlanRob*, 2017.

[42] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

[43] Pedro Santana, Sylvie Thiébaux, and Brian Williams. Rao*: An algorithm for chance-constrained pomdps. In *AAAI Conference on Artificial Intelligence*, pages 3308–3314, 2016.

[44] Matthias Schreier, Volker Willert, and Jürgen Adamy. Bayesian, maneuver-based, long-term trajectory prediction and criticality assessment for driver assistance systems. In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pages 334–341. IEEE, 2014.

[45] Pavel Senin. Dynamic time warping algorithm review. *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA*, 855:1–23, 2008.

[46] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[47] Eric Timmons. Fast, approximate state estimation of concurrent probabilistic hybrid automata. 2013.

[48] Eric Timmons and Brian Williams. Best-first enumeration based on bounding conflicts, and its application to large-scale hybrid estimation. Technical report, MIT CSAIL Technical Report, 2018.

[49] Mark M Tobenkin, Ian R Manchester, and Russ Tedrake. Invariant funnels around trajectories using sum-of-squares programming. *IFAC Proceedings Volumes*, 44(1):9218–9223, 2011.

[50] Rafael Toledo-Moreo and Miguel A Zamora-Izquierdo. Collision avoidance support in roads with lateral and longitudinal maneuver prediction by fusing gps/imu and digital maps. *Transportation research part C: emerging technologies*, 18(4):611–625, 2010.

[51] Quan Tran and Jonas Firl. Modelling of traffic situations at urban intersections with probabilistic non-parametric regression. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pages 334–339. IEEE, 2013.

[52] Aditya Undurti and Jonathan P How. An online algorithm for constrained pomdps. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3966–3973. IEEE, 2010.

[53] Vygandas Vaitkus, Paulius Lengvenis, and Gediminas Žylius. Driving style classification using long-term accelerometer information. In *Methods and Models in Automation and Robotics (MMAR), 2014 19th International Conference On*, pages 641–644. IEEE, 2014.

[54] Yi Wang, Kok Sung Won, David Hsu, and Wee Sun Lee. Monte carlo bayesian reinforcement learning. *arXiv preprint arXiv:1206.6449*, 2012.

[55] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.