POMDP-BASED BEHAVIOR PLANNING FOR ANTICIPATING AND RESPONDING TO LANE EXCURSIONS IN LARGE AUTONOMOUS VEHICLES

By

Batuhan Yücer

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science—Master of Science

2024

ABSTRACT

Large autonomous vehicles are rising in prevalence, and the lane excursion problem is a common occurrence in the navigation of large vehicles through urban roads. To the best of my ability, background research has not revealed any studies on solving this problem under the assumption that the excursion is unavoidable with a large vehicle body. This thesis introduces the lane excursion problem to the research space with a primary focus on dynamic vehicle interaction, and possible solutions are considered. The main solution provided is a POMDP (Partially Observable Markov Decision Process) method utilizing predictive planning given another vehicle with unknown intentions. A simulation environment is created for evaluation, and the POMDP method is evaluated using two measures introduced along with the problem. The POMDP method demonstrates good approximation ability of the hidden system variables and performs better than an uninformed non-predictive approach. With an additional section on the real-life implications of the problem, this study is expected to bring to light the importance of the lane excursion problem and prompt deeper discussion on the domain of large autonomous vehicles as a whole.

ACKNOWLEDGEMENTS

I would like to express my deepest thanks to my supervisor, Prof. Joshua Siegel, for being constantly supportive and constructive towards my work on this thesis. He has been a true inspiration and a great teacher to me during my time in the graduate program.

Special thanks to my committee members, Prof. Philip McKinley and Prof. Daniel Morris, for their enthusiasm towards my project and their invaluable expertise and time in evaluating this work.

In addition, I would like to acknowledge Dr. Ali Ufuk Peker and ADASTEC Corp. for encouraging me to pursue graduate studies and for their financial support throughout this endeavor.

I sincerely thank my friends and colleagues, who have been by my side whenever I needed them and helped me through times of stress.

Finally, I am incredibly grateful for my loving family. My parents are why I am where I am today, and I feel the deepest appreciation for them as I close another chapter of my life.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS vi

CHAPTER 1 INTRODUCTION 1

CHAPTER 2 BACKGROUND 5
- 2.1 Formal Definition of the Lane Excursion Problem 5
- 2.2 Previous Research 11

CHAPTER 3 METHODOLOGY 16
- 3.1 Modeling the Scenario 16
- 3.2 ABT Solver 24
- 3.3 TAPIR and ROS Integration 25

CHAPTER 4 EVALUATION AND RESULTS 28
- 4.1 Simulation of the Lane Excursion Scenario 28
- 4.2 Rule-Based Baseline Method 30
- 4.3 Performance Measures 31
- 4.4 Results and Analysis 32
- 4.5 Effects of Observation Uncertainty 36
- 4.6 Discussion on Real-life Application 38

CHAPTER 5 CONCLUSION 41

BIBLIOGRAPHY 43

LIST OF ABBREVIATIONS

POMDP	Partially Observable Markov Decision Process
NACTO	National Association of City Transportation Officials
MDP	Markov Decision Process
PBVI	Point-Based Value Iteration
POMCP	Partially Observable Monte-Carlo Planning
ABT	Adaptive Belief Tree
TAPIR	Toolkit for Approximating and Adapting POMDP solutions In Real-time
ROS	Robot Operating System

CHAPTER 1

INTRODUCTION

As autonomous driving technology continues to rise in prevalence, the idea of the autonomous vehicle also evolves alongside it. As with most emerging technologies, entry to earlier research on autonomous transportation was a higher hurdle to clear than it is today and a risky endeavor for the automotive industry at the time. Thus, the profit-driven, privately owned passenger vehicle has been the main product of the autonomous transportation industry for the last several decades. However, as the industry matures, a wider array of profitable business models emerge along with the decrease in upfront R&D costs. This is the reason that, recently, there has been a rise in the number of companies attempting to tackle newer challenges, such as autonomous public transportation.

Viewing the autonomous driving challenge through a public transportation perspective naturally expands the design of an autonomous vehicle to cover many more use cases than before, each with its own challenges. One such case is an autonomous bus transporting passengers throughout an urban route multiple times daily. Navigating urban areas with a large bus is, of course, a much different and arguably more difficult exercise in practice than it is with a standard passenger vehicle. This type of problem is also not explored in literature as extensively as it is for passenger cars because of the sparsity of large autonomous vehicles in the industry.

As a software engineer in the industry, my background includes several years of employment at ADASTEC Corp., a company developing a Level 4 autonomous driving software platform for public transportation vehicles. At ADASTEC, most of our research and development has been performed on typical public transit buses. A major part of the product is a full-stack autonomous driving software and sensor kit, which includes all the standard Level 4 automation capabilities from perception to localization and control. I personally work on the path planning and control of the vehicle, which involves route navigation, short and long-term decision-making, lane following, and comfortable driving. The outputs of these tasks depend heavily upon the kinematics and the geometry of the driven vehicle, which means that we regularly face challenges that arise due to the vehicle specifically being a bus. Previous research often makes the understandable assumption

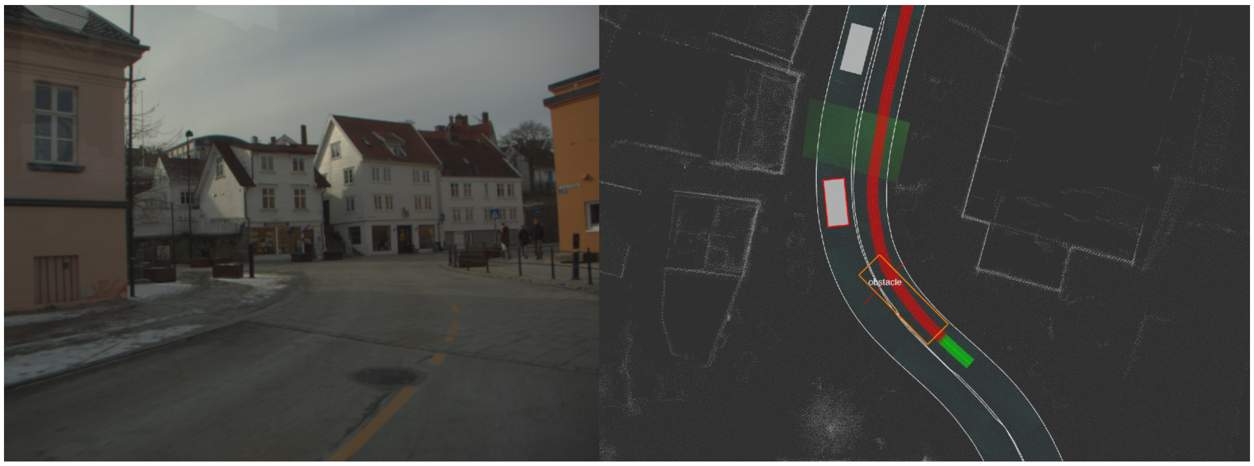


Figure 1.1 A part of the daily route taken by the bus running ADASTEC's software at the city of Stavanger, Norway. On the left is a camera snapshot of the general area, and on the right is the lane excursion scenario replicated in a simulation environment using collected sensor data.

that the vehicle in question is a passenger car that has standard dimensions and uses a well-known driving model, making it difficult to adapt to public transport where large vehicle dimensions and unusual vehicle geometries are commonly encountered, such as trucks and articulated buses. In this thesis, I address one such challenge where the long wheelbase and the wide frame of a bus cause lane excursions during sharp turns, a regular occurrence on urban roads.

Today, many other autonomous driving companies are also focusing on larger types of autonomous vehicles. Aurora's [1] hardware system product is designed to be fitted to primarily semi-trucks moving long distances. EasyMile [2] has a fleet of hundreds of autonomous passenger shuttles currently in operation, as well as truck and tow tractor projects. With the rising development efforts towards large autonomous vehicles, it becomes increasingly important to gain the unique perspectives required to address domain-specific challenges associated with these vehicle types. The lane excursion scenario to be defined is one such challenge.

Figure 1.1 depicts an example of a lane excursion from data collected by ADASTEC. When the autonomous bus attempts to clear the sharp turn, the front of the bus crosses the lane separation line and into the opposite lane. Even though this maneuver is unavoidable due to the shape of the road and the long wheelbase, it presents several possible risks: a collision with an oncoming vehicle in the opposite lane, a standoff where both the bus and the opposing vehicle are unable to move forward and one must reverse to clear the way, or at the very least, discomfort to the inhabitants of one (or both) of the vehicles involved due to emergency braking.

To prevent trajectories from conflicting with other vehicles in other scenarios, such as navigating intersections, many autonomous driving pipelines employ predictive methods so that said situations can be foreseen and appropriate actions can be performed by the autonomous vehicle before the system reaches that state. However, to the best of my knowledge, these methods have not yet been implemented in the context of the category of lane excursions where an area of collision risk exists directly as a consequence of the large size of the ego vehicle.

In this thesis, I introduce a formal definition of "the lane excursion problem" to the literature. Although this concept of lane excursion is described using physically large autonomous vehicle types as references, such as trucks and buses, the problem is generalizable to any vehicle. Lane excursions happen and are unavoidable whenever the curvature of the road is too high for the whole body of the vehicle to stay in lane boundaries. With this perspective, the "largeness" of the navigating vehicle is directly dependent on road shapes and sizes. I formally define the term "largeness" in the context of the lane excursion problem as well, for future reference in literature.

I propose two types of longitudinal controllers to navigate through the scenario under defined problem constraints. For the best solution, I model the lane excursion scenario as a POMDP, an optimization-based system model that enables accurately approximating the system state in future time steps and selecting the best action to take in an online fashion. The other solution is a configurable, rule-based controller which acts as a baseline. For evaluation, I introduce two performance measures and create a simulation environment. The POMDP method demonstrates high performance through quantitative and qualitative analysis.

The lane excursion problem is significant due to its high probability of occurrence during urban navigation, especially with commercial buses, as we at ADASTEC often experience. A meaningful example is the typical four-way intersection structure in the United States, where city buses need to encroach into the opposite lane for some time while turning right because it is impossible to clear the 90-degree turn without this encroachment. Figure 1.2 shows an example of this.

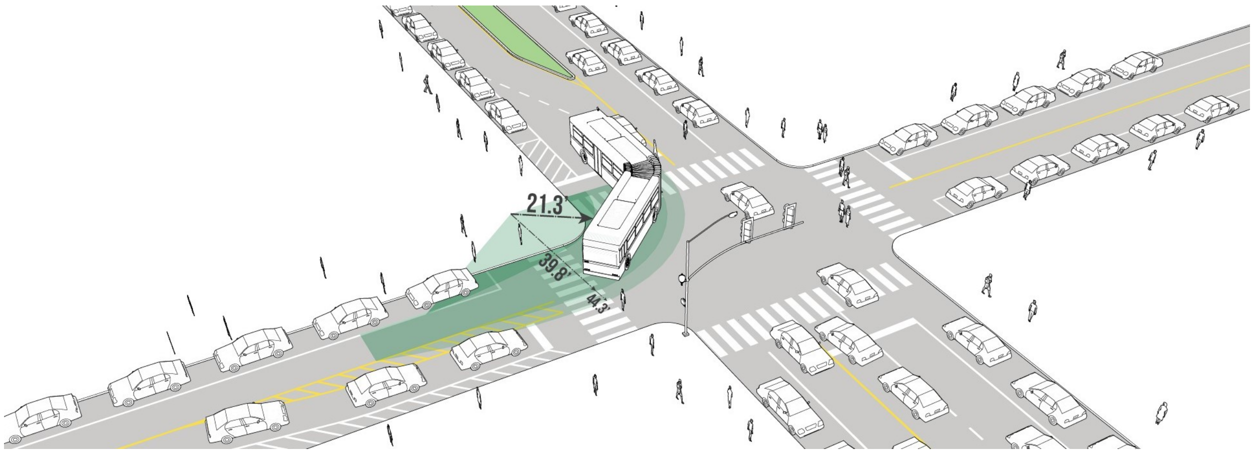


Figure 1.2 An example from a design guide by NACTO (National Association of City Transportation Officials) [3], displaying the extra lane space an articulated bus requires while making a sharp right turn in an urban four-way intersection. The front end of the bus draws a larger arc than the kinematic base of the leading bus segment.

A past article from the Transportation Research Record journal includes statistics from commercial bus accidents [4]. They report that rear-end crashes where one of the vehicles is stopped are among the most common types of accidents (24% of all recorded crashes), especially at intersections. They add that the bus is often the stopped vehicle in these incidents. They argue that the reason for this is the bus unexpectedly coming to a stop during right turns because of vehicles in the opposite lane. Their primary proposal to address accidents is to widen the turn radius of intersections so that buses can clear the turn without encroaching upon the opposite lane. Another study shows that narrower lane widths directly correlate to higher bus accident rates [5].

With the significance of the problem in mind, my goals in this thesis are twofold. First, I aim to present a successful real-time algorithm to address one category of lane excursions. Second, I aim to provide perspective on the real-life implications of the defined lane excursion problem and to prompt future research toward solving the additional challenges brought on by the much more complex real-life environments. To this end, I also include a discussion section detailing the considerations one must make when implementing the lane excursion planner in a real autonomous system. I expect that the proposed lane excursion planner will play a pivotal role in path planning pipelines in the future, especially for projects focused on large autonomous vehicle types in the growing industry.

CHAPTER 2

BACKGROUND

2.1 Formal Definition of the Lane Excursion Problem

2.1.1 Lane and Vehicle Geometry

We start by considering a two-lane road (right-hand traffic), with the lanes $\mathcal{L} = \{\mathcal{L}_e, \mathcal{L}_o\}$ in opposite directions. The ego vehicle moves along $\mathcal{L}_e$, the ego lane, and an independent actor vehicle moves along $\mathcal{L}_o$, the opposite lane. For this problem, both vehicles are assumed to follow their lane centers perfectly, and lateral motion will be ignored. The reason for this is explained more in detail in the section about problem boundaries.

In our scenario, the ego lane curves to the right, and the opposite lane naturally curves left along with it in a consistent circular motion until both lanes have turned 90 degrees, resulting in a quarter of a circle with radii $\mathcal{R}_e$ for $\mathcal{L}_e$, and $\mathcal{R}_o > \mathcal{R}_e$ for $\mathcal{L}_o$. As traffic rules dictate, both lanes have associated speed limits; for this study, the limit is set to 5 m/s everywhere. The speed limits also decrease on the curved parts of the lanes to limit the lateral acceleration experienced while taking the turn.

We define an ego vehicle navigating the turn on $\mathcal{L}_e$ with dimensions $\mathcal{D}_e = \{f_e, b_e, s_e\}$, where f_e is the distance between the rear axle and the front end of the vehicle, b_e is the distance between the rear axle and the rear end of the vehicle, and s_e is the distance from the driveshaft to the right or the left side of the vehicle. It is clear then that the vehicle's total length is equal to $f_e + b_e$, and its total width is equal to $s_e + s_e$. We similarly define an actor vehicle with dimensions $\mathcal{D}_a = \{f_a, b_a, s_a\}$, navigating through $\mathcal{L}_o$ in the opposite direction at the same time.

To represent a real-life situation, we refer to values from ADASTEC's collected data as well as the actual bus dimensions used on-site. Therefore, for the rest of this thesis, the reference vehicle dimensions are $f_e = 6.62$, $b_e = 1.70$, $s_e = 1.22$, $f_a = 3.72$, $b_a = 1.11$, $s_a = 1.05$ (all in meters). For the solutions proposed in this study, these values are arbitrary as long as they are geometrically feasible and the vehicle dimensions are large enough to cause the lane excursion problem, according to the equations provided throughout the rest of this definition. This maintains the transferability

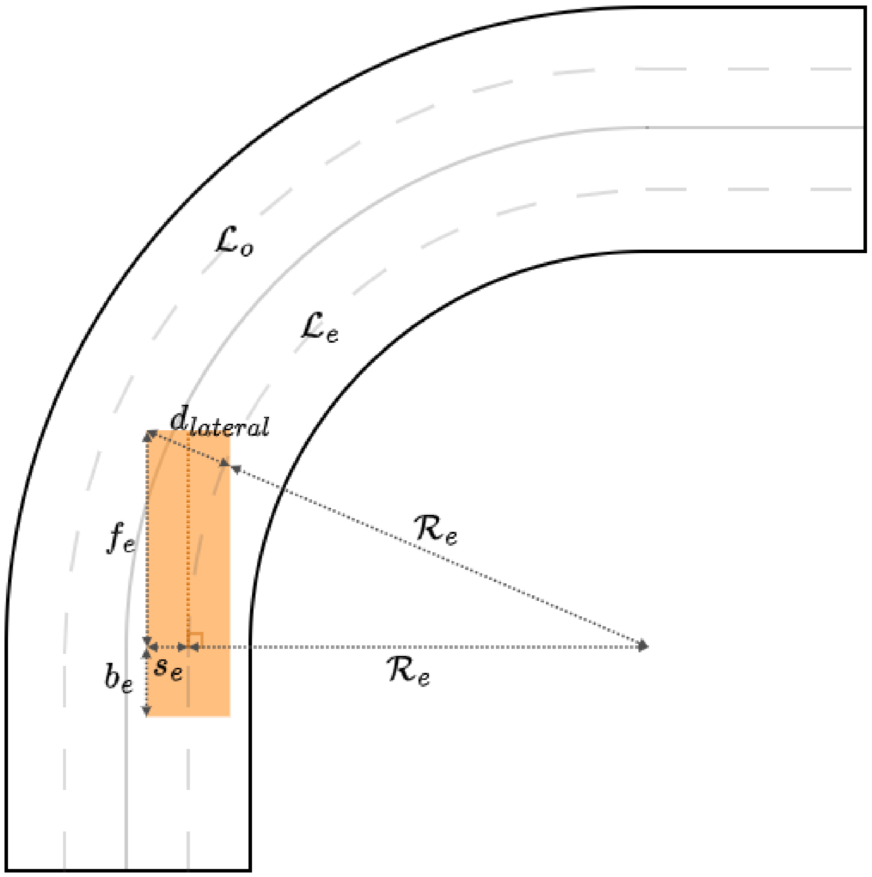


Figure 2.1 An instance of lane excursion due to the curvature of the road. All relevant vehicle and road dimensions are labeled.

of the problem when the vehicle sizes vary. However, it is assumed that the vehicle has a single rigid body, so vehicles with multiple segments, such as articulated buses and trucks with trailers, would need to be modeled differently, and they do not apply to this problem definition.

The standard bicycle model commonly used for most autonomous control frameworks [6] dictates that the center of the rear axle of the vehicle is the reference point that traces the lane to be followed, while the front axle is steered to keep the reference point on the lane center at all times. Given the circular shape of the curve in the scenario, at any point inside the curve, the ego vehicle's driveshaft will be tangential to the lane, which leads to the front of the vehicle being farther to the lane center compared to its rear. Figure 2.1 visualizes this scenario along with the relevant variables.

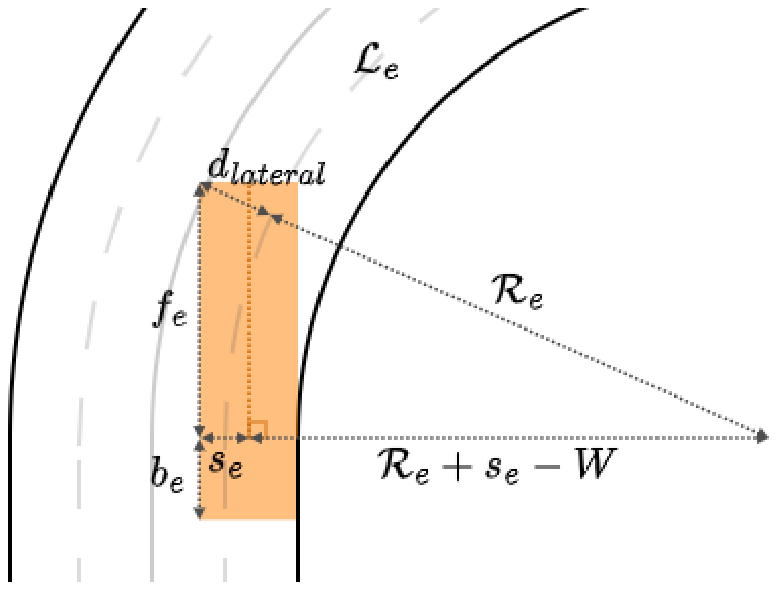


Figure 2.2 The vehicle in the scenario moved to the far right of its lane, to calculate if it is "large" in the context of a lane excursion. The image shows that it still slightly overextends at the front left point, which means that it is indeed a large vehicle.

While the ego vehicle is in this state, the center of the rear axle is exactly on the lane center, and the furthermost point to the lane center is the front-left corner, with a lateral distance of $d_{lateral} = \sqrt{(\mathcal{R}_e + s_e)^2 + f_e^2} - \mathcal{R}_e$. Given a turn radius of $\mathcal{R}_e = 15$ meters, which is far from an extreme case, this distance calculates to $d_{lateral} \approx 2.52$ meters.

Many standards regarding urban road design recommend lane widths of around 3.5 meters when accommodating large vehicles such as city buses [7, 8], meaning a half-width of 1.75 meters. The previously calculated $d_{lateral}$ value exceeds this half-width and invades the opposite lane by 0.77 meters, comfortably blocking the path of an opposing vehicle and causing risks. This becomes even more severe when the curve of the road is tighter or a lane width as high as 3.5 meters is not available for the lane in question, situations not at all uncommon on a highly variable urban route.

We take this opportunity to coin a new term, "largeness," to analyze lane excursions. We say that a vehicle "has largeness" or "is large" in the context of a possible lane excursion scenario if the excursion is impossible to avoid through trajectory shape optimization. This condition can be checked by assuming the vehicle is as far to the right of its lane as possible. This ensures that the vehicle can neither be translated further to the right to avoid the excursion nor steer to the right, because the right side of the vehicle body is currently at a tangent to the right lane boundary, and any steering would cross the lane on the right side. This can be seen in Figure 2.2. If, in this state, the vehicle still overextends to the opposite lane, then it is, by definition, a large vehicle for the purposes of the lane excursion problem. This condition is formalized as an inequality check below:

$$\text{Vehicle is large if } \ d_{lateral} = \sqrt{(\mathcal{R}_e + 2s_e - W)^2 + f_e^2} - \mathcal{R}_e > W \tag{2.1}$$

where W is equal to the half-width of the ego lane.

It is useful to emphasize that the defined "largeness" term for a vehicle is relative to the lane it navigates. With this definition, even a regular passenger vehicle can be large in the context of a lane excursion scenario if the lane width is too small or the curvature is too sharp. Given a half lane width of $W = 1.5$ meters and the vehicle dimensions previously provided, we calculate that the ego vehicle is large if the lane curve radius is $\mathcal{R} < 37.9$ meters, In contrast, if the actor vehicle were to navigate the same lane, it would be defined as large if the lane curve radius was $\mathcal{R} < 6.6$ meters.

2.1.2 Vehicle Interaction

To reiterate, the main risk while navigating this curve stems from a possible collision with the incoming actor vehicle, and the main challenge is that the intent of the other driver at any point is not known to the ego vehicle. For this scenario, the actor vehicle is assigned one of three driving styles: "standard," "aggressive," or "cautious." In simulating the scenario, their driving style affects their decision-making towards passing or yielding. By default, all drivers attempt to follow their reference lane velocity as closely as possible. However, when both the ego vehicle and the actor vehicle approach the conflict area at the same time, their interaction in a short span of time decides which driver takes priority in passing the conflict area and which driver waits to yield the way. An aggressive driver, as defined here, speeds up at this moment to attempt to pass before the ego vehicle. A cautious vehicle slows down and gives an opportunity to the ego vehicle to pass. A standard vehicle maintains its speed and reacts to the actions of the ego vehicle instead.

The existence of a set of different driving styles is a cornerstone of the definition of a lane excursion problem. Without varying styles, the actor vehicle becomes completely reactionary, at which point the ego vehicle assumes complete control of the whole interaction, and a major source of uncertainty is cleared. However, with varying driving styles, the ego vehicle needs to estimate a course of action by considering significantly different future states while the style is hidden from the autonomous system. The set of styles can be more diverse and complex depending on how much accuracy is needed in representing a real-life scenario. For this study, with random acceleration noise added in simulation, the given set of three driving styles is able to represent a sufficiently wide variety of driving behaviors such that a controller with an incorrect assumption fails to navigate through the scenario without conflicts.

An autonomous system may employ different types of automation frameworks to navigate the scenario. One such framework involves a prediction module to approximate the state of the actor vehicle on a finite time horizon. This allows the planning module to react accordingly and avoid maneuvering itself into a state that conflicts with the actor state in the future. A problem with this approach is that it ignores that the actor's actions heavily depend on the ego vehicle's actions because the human driver on the other side will make their own assumptions and predictions on how the ego vehicle will move in the foreseeable future. If all possible actions of the actor are considered with the same weight and many predictions are made at the same time, this leads instead to a very conservative speed planning by the autonomous system and possibly even to the "freezing robot" problem [9]. This is because whatever action is taken by the ego vehicle, there will always be a non-negligible chance that the actor vehicle will act in such an undesirable manner that the system reaches a conflicting state. Therefore, the best action to take becomes not to move at all.

Another possible framework involves performing prediction and planning in parallel. Here, a search space is established for the possible future states, with well-defined state transitions. This space can be searched up to a finite horizon, and optimal actions can be chosen by assigning values to states similar to Q-learning. This approach enables the system to discard state sequences with negligible probabilities, leading to a better approximation of what will be encountered in the future. A POMDP, as will be detailed in subsequent chapters, can be used to model the scenario in a way that is solvable by this second framework.

2.1.3 Problem Boundaries and Assumptions

Here, we list the assumptions made in formally defining the lane excursion problem:

- The road geometry is constrained to a two-lane road with lanes in opposite directions. There may be other lanes to the right of each lane, but the model would not include them, and the proposed solvers in this study would not handle lane changes. Different curvatures and lane widths are acceptable, and the lanes can also be curving in the other direction.

- There is only one actor vehicle, and it is visible to the autonomous system sensors. It is hypothetically possible to handle multiple actor vehicles using the solvers provided in this work by running the optimization multiple times. However, doing this instead of implementing both vehicles in a single optimization process is computationally infeasible.

- Vehicle shapes can be arbitrarily chosen as long as they form a geometrically plausible, single rigid body. The actor vehicle can also be large, which the solvers are expected to handle. Vehicles with segmented bodies (e.g., articulated buses) are excluded while defining the problem.

- For lane excursions to occur, there should be a notable curvature on the road. So, low urban speeds are assumed for both vehicles instead of highway speeds. The vehicles are assumed to be reasonable in staying under the speed limits provided by the traffic rules on the map.

- As an extension to the low-speed assumption, vehicle dynamics characteristics of the bus that affect high-speed driving, such as tire slip, are ignored for movement calculations. Instead, the bicycle kinematic model is assumed for both vehicles moving at low speeds.

- For the purposes of this study, lateral lane motion will be ignored since the problem assumes in the first place that a lane excursion is unavoidable due to the shape of the road. As some examples show in the introduction section, there are many cases in urban road navigation where the large vehicle is unable to progress without encroaching into adjacent lanes. Here, we assume that only a trajectory with such a shape is available, and the task becomes a type of velocity optimization over the provided trajectory. This is a common path planning framework known as the "path-velocity decomposition," where the ego path's shape and velocity profile are calculated consecutively instead of being optimized together [10].

In this work, the shape of the trajectory is provided as an input to the POMDP solver, which then acts as the velocity optimizer. The POMDP, as will be detailed further, models the system state using only the longitudinal distance traveled on the reference lane and not the lateral distance to the lane. The solver is capable of handling noise over the observed system. Since any lateral motion will only be observed by the solver as some noise on the longitudinal motion, the resulting change in behavior is expected to be insignificant. Thus, ignoring the lateral lane motion is, in fact, not a major issue in this study.

2.2 Previous Research

2.2.1 POMDP

The conceptualization of the Partially Observable Markov Decision Process starts with the assumption of incomplete state information being incorporated into the MDP [11] (Markov Decision Process). Later works reveal that solving for an optimal policy given a non-trivial POMDP model is a computationally intractable task [12, 13], due to what is now commonly referred to as "curses of dimensionality and history." The severity of these computational drawbacks is aggravated by modeling the POMDP with continuous state, observation, or action spaces, often the best options for representing real-life problems. Therefore, while the earlier POMDP solvers focus on the discretization of these spaces as well as the implicit belief state, the preferred approach later becomes approximation by belief state sampling, employed in popular methods such as PBVI (Point-Based Value Iteration) [14], POMCP (Partially Observable Monte-Carlo Planning) [15] and their many variants. Specifically in online policy iteration, the ABT (Adaptive Belief Tree) speeds up computation drastically by utilizing sampled belief states from the previous time step instead of resampling the entirety of the state pool each time. In addition, it is proven that an ABT will converge to an optimal solution with sufficient optimization time, meaning that the action sequence with the highest total expected reward will be approximated correctly [16].

With these latest developments in solving POMDPs, and the ability of the POMDP to represent the data gathering process under constraints of noise and incompleteness, they have become useful tools to address real-life problems in many different fields. Recent works utilize POMDPs to tackle problems in ecology [17, 18], networks [19, 20], infrastructural safety [21] and many aspects of robotics [22, 23, 24].

Due to the difficulty of coding complex POMDP problems in a reliable and computationally efficient manner, open-source frameworks for solving various problems using POMDP models have also been developed in software languages such as Python [25], Julia [26], and C++ [27].

2.2.2 Path Planning for Large Autonomous Vehicles

Some existing work on the consequences of large vehicle dimensions focuses on trajectory optimization and vehicle control. Yu et al. [28] design a comprehensive planning and control framework for autonomous buses of 12 meters in length. They focus on the uncommon kinematic properties of the autonomous vehicle in question as the primary constraint on trajectory generation and rule-based decision-making algorithms for various scenarios such as lane changing and overtaking. More recently, Oliviera et al. [29] address the navigation of narrow and constrained roads via autonomous buses. They model the lane following scenario as an optimization problem where the kinematics and dynamics of the bus are used as constraints, as well as the existence of the front and rear overhangs on a bus, which are free to move outside the drivable area during lane following as long as the wheels stay inside. The long wheelbase is also considered as one of the issues since the front of the bus creates a larger arc than its rear while turning, a crucial characteristic in the scenario this thesis work is built upon.

2.2.3 Path Planning as a POMDP Problem

In the context of autonomous driving, some major sources of uncertainty, such as sensor noise, occlusions, and the lack of knowledge about the intents of other actors in the environment, can all be taken into consideration when modeling the system as a POMDP. This allows solvers to perform high-level decision-making tasks such as navigating an intersection or roundabout [30, 31, 32], changing lanes [33, 34], or stopping to pedestrians at crosswalks [35]. These methods usually differ most in their state and observation representations, as well as the solvers used to calculate the target policy.

As mentioned in the problem definition, when the trajectory of the autonomous vehicle does contain an unavoidable lane excursion, some part of the drivable area is established as the boundaries in which a collision may occur. Then this problem can also be evaluated as a collision avoidance problem, where two incoming vehicles attempt to predict each other's behavior and safely navigate through the area with the collision risk. Re-framing the lane excursion problem in this manner is important since it reveals many similarities with the intersection navigation problem. Therefore, this thesis work takes inspiration from notable prior research on intersection navigation.

Brechtel et al. [36] formulate the POMDP under the assumption that uncertainties arise from noise and occlusions. The sensor noise is implicitly defined in the observation model, and to handle occlusions, a special type of observation is defined for vehicles outside the line of sight of sensors. The authors implement their own solver to calculate the policy offline. The belief space has to be discretized to make computation feasible for the offline approach. This discretization is performed a priori by a learned neural network. Unfortunately, this constrains the problem to previously known environments, and a dynamic solution becomes unavailable.

Liu et al. [37] model T-junction and roundabout scenarios using a POMDP with multiple obstacle vehicles. The states of the vehicles are their positions and orientations, as well as a hidden intention from a set of 4 different intentions: Stopping, hesitating, normal, and aggressive. The hidden intent directly affects how the driver's behavior differs from the expected motion, and the expected motion is encoded into the local map through a previously learned model utilizing geometric context from the drivable areas. The concept of a set of hidden intents leading to varying driving models is used as inspiration when defining the lane excursion problem for this thesis.

Hubmann et al. [38] focus on a four-way intersection scenario where the sources of uncertainties are sensor noise and driver intent. Other actors can choose one of three lanes to follow; this intent variable is hidden from the autonomous vehicle, leading to partial observability. Ego actions are selected depending on the hidden variables' beliefs, and the utilized ABT solver allows more accurate preventative decision-making at the current time step. The online policy computation and the modeling of the hidden variable make this work an appropriate analog to the lane excursion problem. Thus, it will be referenced frequently throughout this writing.

Zhang et al. [39] tackle the same problem in an intersection with unknown driver intents. However, they take a different approach under the assumption that the set of possible predicted paths that the actor vehicles may follow are already available through the prediction module of the autonomous stack. With this, state representation becomes trivial, with only the static intents of the vehicle being (hidden) variables. At each time step, a time series of possible future vehicle poses is used as input for policy optimization. This time series is discretized into a multi-step occupancy grid comprised of layered masks of the occupied cells over time. While representing the predictions as grid maps makes online computation feasible, accurate predictions are not always available in practice. Another major issue is that separating the prediction module prevents the ego vehicle from considering the future interaction between the two vehicles when deciding on a behavior.

2.2.4 Addressing Lane Excursions

Lane excursions are also referred to as lane departures in literature, although departures often describe completely exiting one lane to move into another in scenarios such as lane changing.

Preventing lane excursions before they occur is a vital constraint in most trajectory optimization problems, and there are many studies on this subject. However, to the best of my knowledge, no studies exist on specifically velocity optimization algorithms that act under the assumption that a lane excursion is unavoidable. To investigate prior art, I used the online tools Google Scholar, Science Direct, and SciSpace to conduct my searches. The most extensive search was done on Google Scholar. Many search terms were used to examine the available studies: To describe the vehicle type, size, and automation, I used the terms "Autonomous," "Automated," "Self driving," "Truck," "Bus," "Shuttle," "Large vehicle," and "Long vehicle." To describe the domain of the desired algorithm, I used the terms "Velocity planning," "Path planning," "Trajectory planning," "Motion planning," and "Decision making." To describe the road characteristics that lead to lane excursions, I used the terms "Turning," "Curvature," "Horizontal curve," "Urban roads," "Lane width," and "Turn radius." Finally, to describe the problem itself, I used the search terms "Lane departure," "Lane overflow," "Lane excursion," and "Encroachment." Among these terms, many combinations were also tried to filter through promising results, such as "Path planning for large autonomous vehicles" and "Lane departures of buses in urban roads." However, I found no results describing the lane excursion problem as defined in this study. Hence, in this thesis, I take the opportunity to define and address said problem, as well as to express the need for it in the industry and to coin necessary terms for future references in this branch of research.

CHAPTER 3

METHODOLOGY

3.1 Modeling the Scenario

3.1.1 Map Representation

As mentioned, the lane excursion scenario in question is simulated on a two-way, double-lane road with a 90-degree turn. The subsequent sections will define a solver to optimize an action policy for navigating this scenario as the ego vehicle. Due to the optimization process being a Monte-Carlo process, future states need to be sampled at a high frequency to solve the scenario as optimally as possible in real time. In addition, the primary goal is to avoid a conflict between two vehicles, and a conflict in this problem occurs when the two vehicles are at a state where neither can pass the other without one of them reversing to yield the way, or even worse, a state of collision. The calculation required to decide if there is a conflict requires many collision checks on simulated future vehicle positions using polygon intersection algorithms and is a computationally heavy process. Eliminating this burden during real-time optimization significantly boosts the processing ability of the method. To this end, the lane map is pre-processed and discretized. The distance ranges on the lanes where a conflict occurs are extracted, and whether a future state is a conflict state can be checked using a simple distance comparison. Pre-processing the map is a viable option in real-life applications since prior map data is usually available to autonomous systems.

First, N points are sampled on both the ego and the opposite lanes, using a sufficiently dense sampling interval to represent the curvature on both lanes. These points represent the possible vehicle positions on the lane, with the center of its rear axle aligning with the position and the orientation of a given point. Then, each point is paired with the opposing point on the other lane, creating N lines connecting the lane centers.

The body of each vehicle is simulated at each lane point. At each point, the vehicle frame intersects a given number of lines, and these intersection points correspond to a ratio of how much the vehicle covers the area between its own lane center and the opposing lane center. A ratio of 0.5 means that the intersection point is exactly in the middle of the two lane centers, 1.0 means that

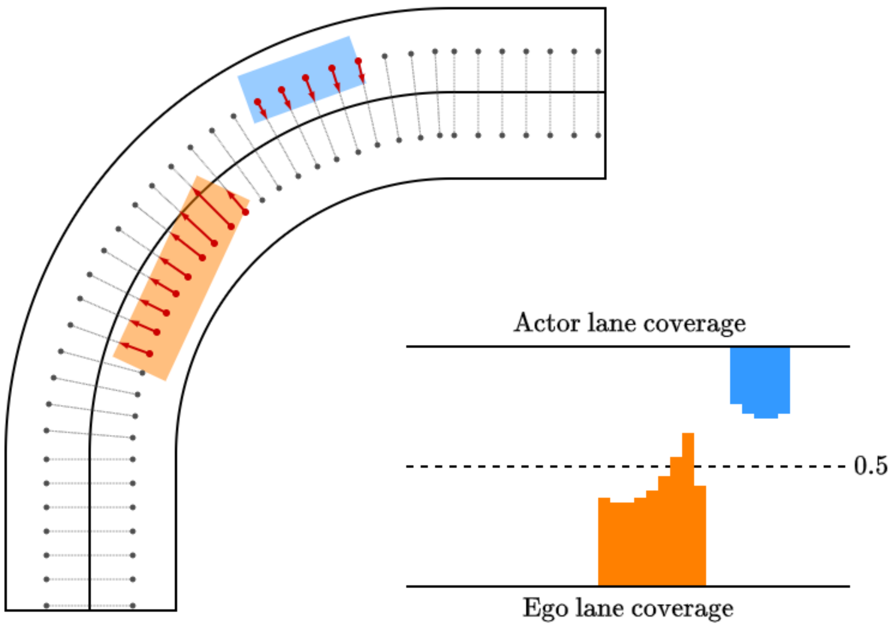


Figure 3.1 A visual representation of the map discretization and conflict checks. The red arrows depict how much each vehicle occupies the space between the two lane centers in different directions, creating a separate histogram at each point.

the vehicle reaches or exceeds the opposing lane center and 0.0 shows that the vehicle is currently not present at that location. This is visually depicted in Figure 3.1. These ratios create a histogram of lane coverage ratios at a given vehicle position. In the end, N histograms are created for each vehicle, where each histogram corresponds to a single lane point.

When the ego vehicle's histogram and the actor vehicle's histogram are summed up, if any value exceeds 1.0, the vehicles are in a physical collision state. This creates the basis of the conflict areas on each lane: If, at any state, the vehicles are propagated forward and there is no possibility that they pass each other without their histograms summing up to a collision, then the two vehicles are in conflict, and the solver must avoid this state. To avoid states that are uncomfortably close to a physical collision, this threshold is reduced to 0.8 instead when producing the conflict areas.

The conflict areas for each lane are extracted only once as a pre-processing step, so this computation does not affect the performance of the algorithm during runtime. Given a future state, the solver now only needs to check if the arc distances of the vehicles are inside the threshold of the conflict area, which is much easier to compute.

3.1.2 Definition of POMDP

A POMDP model is defined by the tuple of expressions $\langle S, A, O, T, Z, R, b_0, \gamma \rangle$, wherein S, A and O are respectively the sets of possible states, actions, and observations for the given problem. At any time step, the problem can be in a state $s \in S$, where taking an action $a \in A$ transitions the system into a new state $s' \in S$. This transition is probabilistic and is characterized by the transition function $T(s, a, s') = p(s_{t+1} = s' | s_t = s, a_t = a)$, which calculates the probability of reaching the state s' at the next time step $t + 1$ when action a is taken inside state s at time step t. Of course, this function and all of the other functions present in the POMDP model adhere to the Markovian property by being dependent on only the latest time step's state, action, and observation.

The property of the POMDP that extends the original MDP model is the partial observability of the system. While in an MDP, the whole system state s is observable from the perspective of a solver, in a POMDP, a part of this information is unavailable. Instead, an observation $o \in O$ is received at each time step after an action is taken and the corresponding state transition occurs. The observation function $Z(s', a, o) = p(o_{t+1} = o | a_t = a, s_{t+1} = s')$ denotes the probability of receiving the observation o when state s' is reached by taking the action a.

Since the state of the POMDP is unknown to a solver, a probability distribution over all possible states is maintained instead. This distribution is called a *belief* b, and $b(s)$ denotes the probability of being at state s under the current belief b. An observation received from the system changes the belief since more information has become available relating to the partially observable state. The initial belief b_0 is part of the model, and the belief update function, $\tau(b, a, o)$, calculates the next belief b' given an action and an observation:

$$b'(s') = \eta Z(s', a, o) \sum_{s \in S} T(s, a, s') b(s) \tag{3.1}$$

Every action in a POMDP is rewarded or penalized. The reward function $R(s, a)$ returns a scalar reward value for taking action a inside state s. The goal of a POMDP solver is to maximize the accumulative future rewards up to a finite or infinite time horizon, given a discount factor $\gamma \in [0, 1)$. Since the system state, transitions, and observations are probabilistic, the value to maximize becomes the expected total future rewards instead. To this end, the solver calculates a *policy* $\pi : B \longrightarrow A$, which maps each possible belief to an action. The optimal policy π^* maximizes the expected total rewards given the initial belief:

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, V(b_0, \pi) = \underset{\pi}{\operatorname{argmax}}\, E\left[\sum_{t=0}^{H} \gamma^t R\left(s_t, \pi(b_t)\right) \middle| b_0, \pi\right] \tag{3.2}$$

where $V(b_0, \pi)$ is the value function and H is the time horizon of the solution.

The belief space is a probability space and is always continuous, which makes optimizing a POMDP policy much more complicated than its MDP counterpart. In addition, the state, action, and observation spaces may also be continuous. Even though this is another source of high problem complexity, it is also a common choice for modeling real-life systems since the alternative is to discretize these spaces but lose crucial information in the process.

3.1.3 State Space

The specific lane excursion problem defined in this work depicts the ego vehicle and the actor vehicle moving in opposite directions in their respective lanes, $\mathcal{L}_e$ and $\mathcal{L}_o$. The state space for this problem is represented as $s = (s_e, s_a)^T \in S$, where s_e is the ego state, and s_a is the actor state. The ego state contains the following values:

$$s_e = (d_e, v_e, a_e)^T \tag{3.3}$$

where a_e and v_e are the magnitudes of the longitudinal acceleration and velocity of the ego vehicle. The acceleration value is included in the ego state because changes in acceleration will be penalized by the reward function, and tracking the current ego acceleration is required. d_e is the longitudinal distance traveled on $\mathcal{L}_e$ in the Frenet coordinate system. The vehicle's rear axle, being its kinematic base, is assumed to be the reference point for its position. Lateral motion is ignored for the purposes of this scenario, so the lateral distance to the lane is not included in the state.

Similarly, the actor state is as follows:

$$s_a = (d_a, v_a, \mathcal{I}_a)^T \tag{3.4}$$

where d_a and v_a represent longitudinal distance and velocity in the opposite lane, $\mathcal{L}_o$. $\mathcal{I}_a$ is the enumerated driving style (intent) of the actor vehicle; it is a constant hidden variable and cannot be

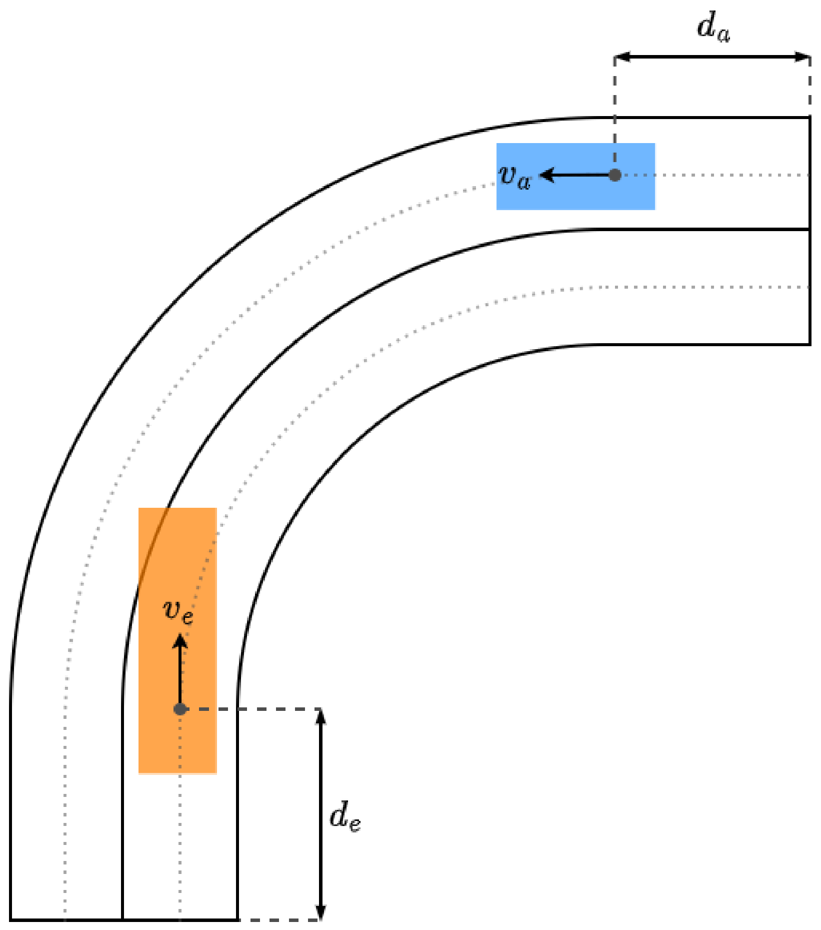


Figure 3.2 Distance and velocity state variables of the lane excursion POMDP model. In addition to these variables, ego acceleration, and actor driving style are also included in the system state.

observed. The driving styles defined for this problem are "cautious," "standard," and "aggressive." It characterizes the behavior of the actor vehicle when both vehicles are approaching the conflict area. In this situation, a cautious driver is more inclined to yield to the ego vehicle, whereas an aggressive driver is more inclined to attempt passing first.

Longitudinal movement categorization is more commonly employed in pedestrian-centric predictive models, with the hidden state being, for instance, the pedestrian's intent towards crossing a road or waiting [35]. Intersection navigation methods, even though they address a similar problem, often ignore this factor in modeling the intersection scenario [38, 39] and categorize driver intent by the turn direction they are heading towards. The longitudinal intent is also pivotal in the context of the lane excursion scenario because the driving style of the actor vehicle is the primary factor in how an interaction will occur between the ego and actor vehicles in future time steps.

The three driving styles defined in this scenario do not necessarily cover every possible individual behavior that one may encounter in real-life situations. However, the simulation of the scenario (detailed in further sections) is constructed in such a way that the actor vehicle can choose to move with acceleration values in the range of -4.0 m/s^2 to 2.0 m/s^2 depending on its driving style, which leads to high interaction variety with the aim to cover the most cases.

This POMDP model employs a continuous state space. Only the driving style is enumerated, whereas the rest of the values are continuous.

3.1.4 Action Space

Actions and observations are the direct sources of the branching factor when considering future belief states for the given problem. Keeping the action set discrete significantly improves the computational ability, as well as leading to closer-to-optimal solutions. For this problem, at each time step, the ego vehicle can take one of three discrete actions: accelerate (1.5 m/s^2), decelerate (-1.5 m/s^2), or maintain its speed (0 m/s^2). The chosen acceleration value is constantly applied until the next time step.

3.1.5 Observation Space and Observation Function

It is assumed for the base model that there is no noise introduced by perception, and the ego vehicle can completely observe its own state, as well as the position and velocity of the actor vehicle. The only hidden variable is the driving style of the actor vehicle. Thus, the observation space is represented as $o = (o_e, o_a)^T \in O$, with o_e being the ego observation and o_a being the actor observation. The specific values included are similar to the state space:

$$o_e = (d_e, v_e, a_e)^T \tag{3.5}$$

$$o_a = (d_a, v_a)^T \tag{3.6}$$

The observation function then simply maps each value to itself. Of course, the defined observation space is continuous in the same way as the state space.

3.1.6 Transition Function

For both the ego vehicle and actor vehicles, a constant acceleration motion model is employed at each time step. The ego vehicle state is updated with the following equation:

$$\begin{pmatrix} d'_e \\ v'_e \\ a'_e \end{pmatrix} = \begin{pmatrix} 1 & \Delta t & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} d_e \\ v_e \\ a_e \end{pmatrix} + \begin{pmatrix} \frac{(\Delta t)^2}{2} \\ \Delta t \\ 1 \end{pmatrix} a_{cmd} \tag{3.7}$$

where Δt is the duration of the time step and a_{cmd} is the chosen acceleration action under the current belief, $a = \pi(b)$. After the state is updated, the distance and velocity values are limited to valid thresholds: $d'_e \in [0, \text{length}(\mathcal{L}_e)]$ and $v'_e \in [0, \infty)$. This is to prevent invalid states, such as vehicles moving outside map limits and moving in reverse.

For the actor vehicle, a similar state update is performed. The only difference is that the driving style stays unchanged:

$$\begin{pmatrix} d'_a \\ v'_a \\ \mathcal{I}'_a \end{pmatrix} = \begin{pmatrix} 1 & \Delta t & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} d_a \\ v_a \\ \mathcal{I}_a \end{pmatrix} + \begin{pmatrix} \frac{(\Delta t)^2}{2} \\ \Delta t \\ 0 \end{pmatrix} a_{gen} \tag{3.8}$$

where a_{gen} is a rough approximation of the actor state behavior. It is the sum of three modifiers:

$$a_{gen} = a_{ref} + a_{style} + a_{unc} \tag{3.9}$$

a_{ref} is the required acceleration for the actor to reach the reference lane velocity at its current position. Its value is limited to the $[-3, 1]$ range (m/s^2). a_{style} approximates the difference in behavior given the driving style of the actor. It is -1.5 m/s^2 for "cautious", 0 m/s^2 for "standard", and 1.5 m/s^2 for "aggressive" actors. Lastly, a_{unc} is a random value sampled from a truncated normal distribution to cover a range of individual behaviors of actors. The distribution is $\mathcal{N}(0, 2)$ with a range limited to $[-2, 2]$. After the sum, the total a_{gen} is limited one last time to the $[-4, 2]$ m/s^2 range. The lower ranges for acceleration values are kept larger in magnitude due to the possibility of emergency brakes. It is assumed that the actor will attempt an emergency brake to avoid a collision if it is the only option available to not enter into a conflict state. This does not correlate to the driving style of the actor, as it is expected from any driver in real life to apply hard brakes when there is a hazard nearby, such as a fast oncoming vehicle without any space to swerve.

Several assumptions are made to approximate the driving behaviors. This approximation does not exactly match the simulated driving model, which will be detailed in the chapter on method evaluation. This is a deliberate choice to show that a rough approximation with general but reasonable assumptions about driver behavior is sufficient to distinguish drivers from each other.

3.1.7 Reward Function

Building upon Hubmann et al.'s approach [38], the reward function consists of several types of penalties (negative rewards):

$$R(s,a) = R_{conflict}(s) + R_{vel}(s) + R_{acc}(s,a) + R_{dist}(s) \tag{3.10}$$

$R_{conflict}$ is a high penalty equal to the coefficient $-K_{conflict}$ when the two vehicles are in a conflicting state according to their distances and the pre-processed conflict area thresholds previously described. When no conflict is detected, this penalty is 0.

R_{vel} is a penalty caused by undershooting or overshooting the reference speed limit v_{ref} at the position of the ego vehicle. Two different formulas are used for lower and higher velocities:

$$R_{vel} = \begin{cases} -K_{v+}(v_{ref} - v_e)^2 & \text{if } \ v_e > v_{ref} \\ -K_{v-}(v_{ref} - v_e) & \text{if } \ v_e < v_{ref} \end{cases} \tag{3.11}$$

with the coefficients K_{v+} and K_{v-} for the two different cases. The higher velocity case is punished with quadratic severity instead of linear severity, since exceeding the speed limit is much more dangerous than being too slow.

R_{acc} is equal to a comparably low negative coefficient $-K_{acc}$ when the action changes the acceleration of the ego vehicle, to prevent erratic behavior and increase comfort. When the acceleration stays the same, the value is 0.

Finally, the distance penalty is calculated as:

$$R_{dist} = \begin{cases} -K_{dist}(d_{conflict} - d_e) & \text{if } \ d_e < d_{conflict} \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$

where K_{dist} is the penalty coefficient and $d_{conflict}$ is the distance at which the collision area begins. This penalty induces the ego vehicle to move closer to the collision area before stopping, leading to a more realistic yielding behavior.

3.2 ABT Solver

Given the continuous state and observation spaces, solving the lane excursion problem online becomes a computationally demanding task. Exploring every possible belief state to generate a complete and optimal policy is infeasible. Thus, this type of problem is usually approximated using Monte-Carlo belief sampling. Among the existing sampling-based POMDP solvers, the ABT (Adaptive Belief Tree) is an appropriate choice to solve a discrete-action, continuous-observation POMDP problem [16]. It is also used in Hubmann et al.'s intersection problem, which is very similar in terms of POMDP characteristics [38].

The ABT functions by sampling many particles (states) from the initial belief, then propagating the particles through a "belief tree" comprised of different beliefs represented as nodes. To propagate a particle, a random action is sampled, a state transition is performed, and then an observation is sampled using the observation model. This process produces a new state, which is added to the tree by branching out from the previous belief to a new belief node. This propagation continues for each particle until the planning horizon is reached from the root. Particles travel through the same node if the same action is taken from the previous node and the received observation is similar enough according to a pre-defined similarity metric. In the end, all the particles in a given node form an approximation of the belief at that node. Using the taken action and the received observation, the corresponding node is used as the initial belief for the next time step in an online fashion. This way, the belief update function $\tau(b, a, o)$ need not be explicitly defined.

The ABT has computational advantages in that it only searches the reachable belief space, and particles sampled from previous time steps remain in the tree, so it is not necessary to sample all states from scratch at every time step. A visualization of this process can be seen in Figure 3.3.

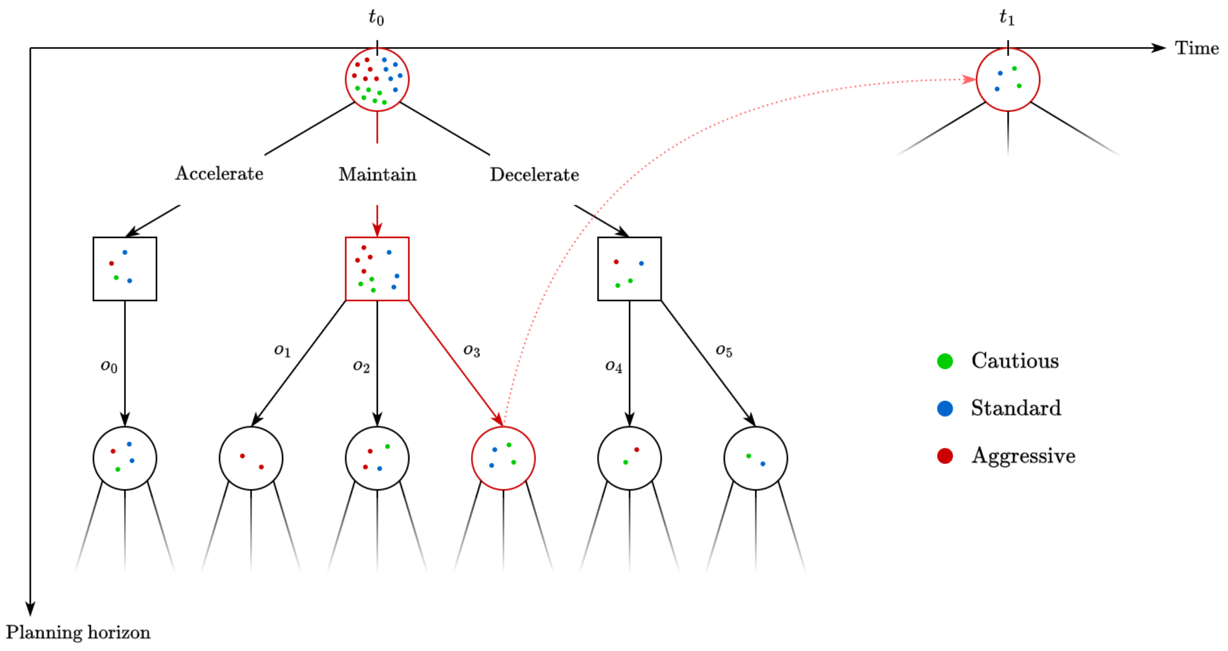


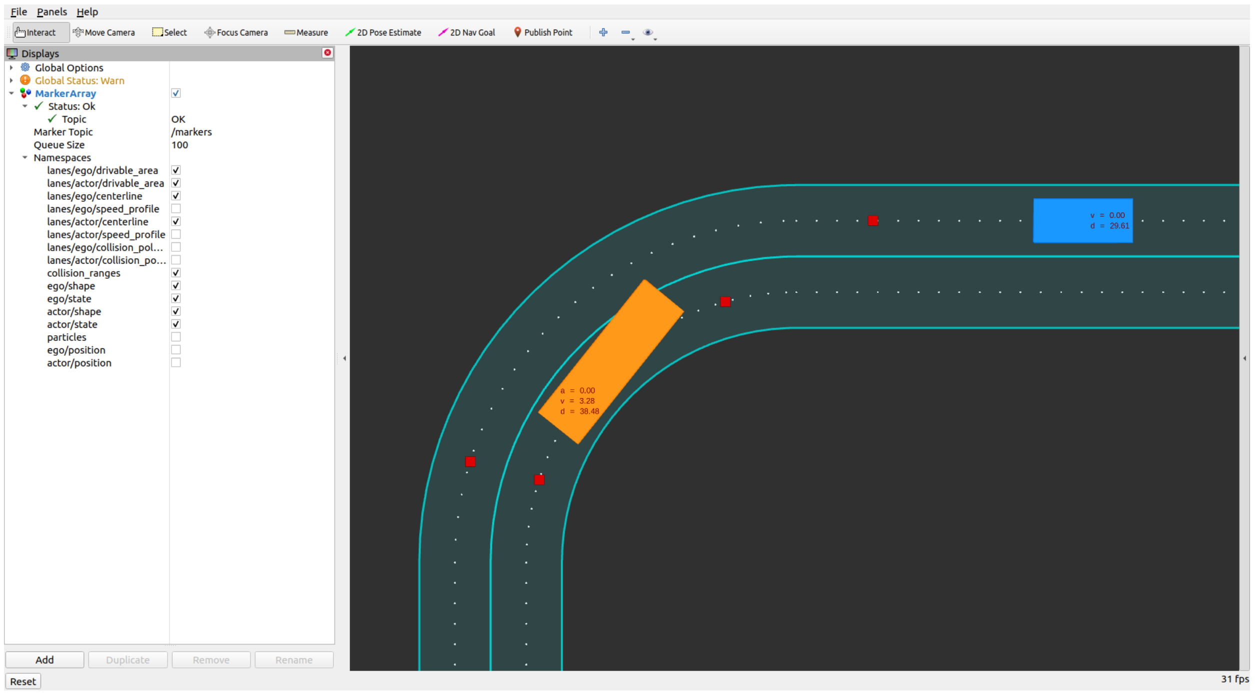
Figure 3.3 The process of sampling future states for the line excursion problem using the ABT method. The hidden driving style of the actor is sampled randomly at the initial belief, and then sampled particles are propagated through the belief tree. The belief node corresponding to the taken action and the received observation is used as the initial belief for the next step. The visual was adapted from Hubmann et al. [38] and redrawn for this problem.

3.3 TAPIR and ROS Integration

TAPIR (Toolkit for Approximating and Adapting POMDP solutions In Real-time) is an open-source software toolkit for solving continuous POMDP problems [27]. It implements the ABT method as a C++ library and provides inheritable classes for defining detailed POMDP models.

ROS (Robot Operating System) [40] is a software and communications framework for designing real-life and simulated robotics applications. It is currently used as a development platform for many autonomous vehicle projects. TAPIR provides the option to build its ABT library as a ROS package and implement the method as a ROS process. For this thesis, the ROS approach is employed, and the lane excursion solver is also implemented in ROS. It requires installing TAPIR separately and building packages together in a ROS workspace. The code can be found in `https://github.com/adastec-batuhan/lane_overflow_pomdp_planner` [41].

The ROS package is named *lopp_tapir*, which stands for "Lane Overflow POMDP Planner"



Global Options	
Global Status: Warn	
MarkerArray	✓
✓ Status: Ok	
✓ Topic	OK
Marker Topic	/markers
Queue Size	100
Namespaces	
lanes/ego/drivable_area	✓
lanes/actor/drivable_area	✓
lanes/ego/centerline	✓
lanes/ego/speed_profile	☐
lanes/actor/centerline	✓
lanes/actor/speed_profile	☐
lanes/ego/collision_pol...	☐
lanes/actor/collision_po...	☐
collision_ranges	✓
ego/shape	✓
ego/state	✓
actor/shape	✓
actor/state	✓
particles	☐
ego/position	☐
actor/position	☐

```
v  =  0.00
d  =  29.61
```

```
a  =  0.00
v  =  3.28
d  =  38.48
```

Figure 3.4 A snapshot of a POMDP simulation, visualized in RViz. Various visual markers are used to debug and track system states. On the left is the menu for selecting which of the published markers are to be currently viewed. The red squares correspond to the beginnings and ends of the conflict area on each lane.

and TAPIR. The term "lane overflow" was changed to "lane excursion" in the thesis later in time, but the package uses the abbreviation "Lopp" for most of the classes. We will refer to the package as "lane excursion planner" during this section.

Using an object-oriented approach, the TAPIR library provides everything related to POMDP as classes. The lane excursion planner inherits these classes and provides the necessary methods and variables to exhaustively model the lane excursion problem as a POMDP model. Some of the classes created are "LoppState," "LoppAction," "LoppObservation," and "LoppModel," which together define all of the POMDP characteristics in code. In addition, some classes unique to the lane excursion problem were needed to describe the scenario. "EgoState" and "ActorState" were created to represent the parts of the POMDP state. "Scenario" was created to design the lanes and extract information on the conflict area.

The "LoppNode" and "BaselineNode" classes inherit from the "TapirNode" class in the TAPIR library, which provides the basic ROS integration and functionalities. The inherited classes build upon this by implementing their own simulation algorithm.

The ROS environment is accompanied by several useful functionalities for the software development process. RViz, a built-in visualization tool of ROS, is used for debugging and visualizing the lane excursion simulation in real-time. The LoppNode and BaselineNode classes both publish a series of "markers" after running each time step of the simulation. A marker is the ROS message type that RViz recognizes and displays; various shapes (or text) in 2D or 3D can be drawn on the RViz screen. The lane excursion planner package publishes the map, vehicles, and states as marker messages to watch the ongoing simulation. The resulting view can be seen in Figure 3.4.

ROS also provides a data format called Rosbag, which records a series of ROS messages with timestamps so that the recordings can be played back and reviewed later. To extract performance measures for the POMDP and baseline controllers, each run is saved into a Rosbag containing the necessary simulation state data for each time step. The Rosbags are then processed using a simple analysis script in Python to produce the measures and plots that are shown in subsequent chapters.

CHAPTER 4

EVALUATION AND RESULTS

4.1 Simulation of the Lane Excursion Scenario

To test the performance of the POMDP, we utilize the built-in simulator class of the TAPIR library. The POMDP solver and the simulator are called in an alternating sequence within a single time step. After the POMDP solver decides on an action, it is passed to the simulation, which updates ego and actor states accordingly.

We implement driving models for both vehicles. The ego and actor vehicles' states are updated using the same kinematic formulas as the equations 3.7 and 3.8. The ego acceleration used is the corresponding value to the taken action. The actor acceleration is chosen through a more complex algorithm, depicted as a flowchart in Figure 4.1.

The actor, by default, aims to reach the reference speed limit at its current position, which produces the reference acceleration a_{ref}. If both vehicles are predicted to enter the conflict area under a time threshold (5 seconds), they are considered "close" in the flowchart. Then, the actor aims for a different target speed depending on its driving style. The new target speed is the lane speed limit multiplied by a coefficient of 0.5 for cautious behavior, 1.0 for standard behavior, and 1.25 for aggressive behavior, producing the acceleration a_{beh}. If the ego vehicle is already inside the conflict area, and the actor is close or inside, then the actor attempts to stop just before the conflict area with a maximum deceleration threshold and let the ego vehicle pass first, producing the acceleration a_{stop}. a_{ref} and a_{beh} are also perturbed with transition uncertainty, using a truncated normal distribution $\mathcal{N}(0, \sigma^2)$ with $\sigma = 2$ and a valid range of $[-2, 2]$ which is able to simulate behavioral variance under a given driving style.

The parameters used for the POMDP model and the ABT solver are given in Table 4.1. As the reward system is similar to Hubmann et al.'s intersection navigation POMDP [38], the values are similar in magnitude; although the penalty coefficients are specifically tuned to the defined lane excursion problem.

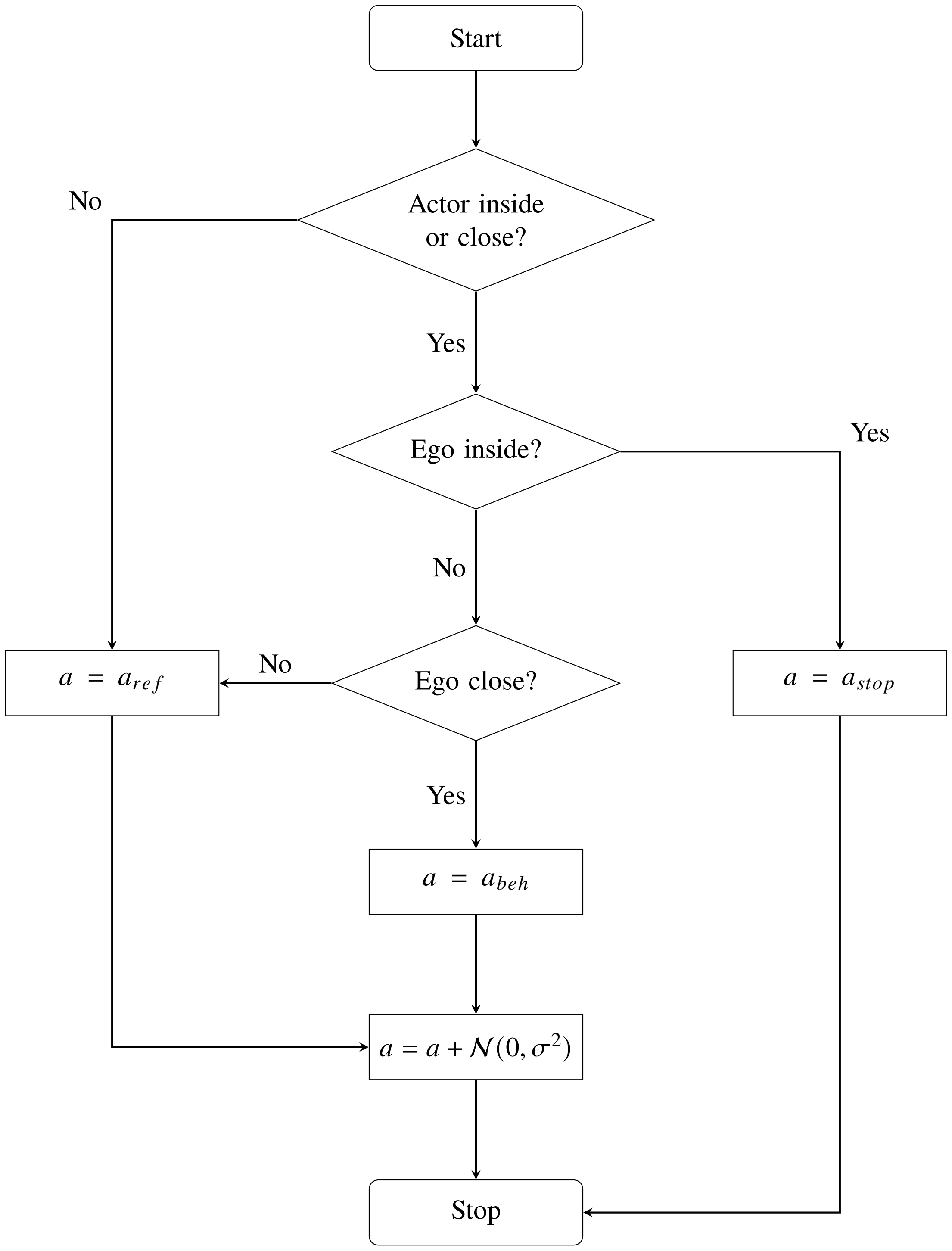


Figure 4.1 Algorithm for deciding the acceleration value chosen by the actor vehicle at each time step of the simulation.

Parameter	Value	Definition
γ	0.95	Discount factor
$K_{conflict}$	10000	Penalty coefficient for reaching a conflict state
K_{v+}	1000	Penalty coefficient for being above the reference velocity
K_{v-}	100	Penalty coefficient for being under the reference velocity
K_{acc}	200	Penalty coefficient for changing acceleration value
K_{dist}	50	Penalty coefficient for staying too far away from the conflict area
Δt	1	Time step (seconds)
H	8	Planning horizon in terms of time steps

Table 4.1 POMDP and ABT parameters.

4.2 Rule-Based Baseline Method

A rule-based driving controller is also created to provide a baseline for comparison to the POMDP solver. The rule-based controller does not simulate further in time to predict future states and interactions, unlike the POMDP. Instead, it assumes optimal ego and actor movements until the conflict area and makes a decision using only the current state of both vehicles. Optimal movement in this context is defined as the series of accelerations that follow the reference lane velocity as closely as possible without considering vehicle interaction. In other words, optimal movement minimizes the sum of the differences between the vehicle velocity and the lane reference velocity at every time step.

The rule-based controller calculates the times in which the ego and actor vehicles will reach the conflict area. Using this convention, if the ego is expected to enter first and after that the actor will have sufficient braking distance to stop without entering, then the ego takes the already calculated optimal actions. Otherwise, the ego vehicle will follow optimal actions until the last possible point, then brake to stop until the actor passes the conflict area.

The time in which the actor will enter the conflict area is calculated using knowledge of the actor's driving style. An aggressive actor is expected to enter earlier than a cautious actor, which ultimately affects the acceleration decision. The controller has three configurations in how it processes driving style information:

1. **Omniscient:** This configuration allows the controller to always know the driving style of the

actor vehicle. Using this knowledge, the omniscient controller is able to calculate the timing precisely and take the fastest path across the conflict area without causing conflicts.

2. **Uniform assumption:** This configuration makes the controller consider each of the three driving styles as valid possibilities. Even when the actor is cautious, the uniform assumption controller will act conservatively and move slower in case the style is, in fact, aggressive.

3. **False assumption:** This configuration causes the controller to misinterpret the actual driving style and randomly pick one of the two remaining styles as a certainty. When the actor is aggressive, the false assumption controller may presume that it is cautious and attempt to enter the conflict area, which is likely a mistake. This type of controller drives fast and tries to pass first, but sometimes leads to undesirable conflicts.

These configurations all have different advantages over each other, but their lack of predictive optimization prevents them from performing well in all possible cases.

4.3 Performance Measures

We define two different measures to evaluate the performance of the rule-based and POMDP controllers. The measures are defined over N simulation runs.

The first measure is the number of times in N runs where the simulation terminated with a conflict state instead of a goal state. The goal state means that the ego vehicle has successfully reached the end of the lane. Of course, fewer terminations with conflict states show better performance.

The second measure is the average velocity error during each run, where the velocity error at each time step is the absolute difference between the ego velocity and the reference lane velocity at the position of the ego. This value is also averaged over N runs. A lower average is better since it means that the ego vehicle was successfully able to follow the reference velocity. For the proposed methods, higher values are most often caused by the ego vehicle stopping to yield the way to the actor vehicle because exceeding the reference velocity is punished severely for the POMDP method, and it is outright impossible for the rule-based method, given its algorithm. This means that higher velocity errors directly correlate to longer clearing times of the lane excursion scenario.

Method	Cautious	Standard	Aggressive	All
Rule-Based, False Assumption	**0**	22	19	41
Rule-Based, Uniform Assumption	**0**	**0**	**0**	**0**
Rule-Based, Omniscient	**0**	**0**	**0**	**0**
POMDP	**0**	**0**	**0**	**0**

Table 4.2 Total numbers of rounds ending in conflicts. The data is over 100 rounds for each driving style, for 300 rounds in total.

Method	Cautious	Standard	Aggressive	All
Rule-Based, False Assumption	1.281	**1.246**	**1.394**	**1.306**
Rule-Based, Uniform Assumption	1.341	1.913	1.768	1.674
Rule-Based, Omniscient	**0.843**	1.587	1.768	1.399
POMDP	1.028	1.711	1.561	1.433

Table 4.3 Average absolute velocity errors of the ego vehicle (m/s). The data is over the successful rounds among 100 each driving style, for a maximum of 300 rounds in total.

Even when yielding the way, the ego vehicle can be controlled in a way that minimizes the wait time for the actor vehicle, which leads to the differences between the performances of the POMDP and rule-based methods, even in scenarios where yielding is the only feasible option.

The second measure, the average velocity error, is reported only for the successful runs, meaning the runs where the ego vehicle was able to avoid a conflict and reach the goal state. This allows for the complete distinction of the two measures regarding their purposes. The first measure shows if the methods can safely navigate the lane excursion scenario. The second measure shows how quickly it is navigated, as opposed to having an undesirable and conservative approach where it is unnecessary. Then, balancing these two measurements (keeping both values low) becomes the goal of a good-performing solution.

4.4 Results and Analysis

We test four methods: POMDP and the rule-based controller with its three different configurations. With each method, 100 simulation runs are executed for each driving style, for a total of 300 runs. The ego and actor vehicle states are randomly initialized within boundary conditions (valid ranges of arc distance and velocity) using a global seed. The list of seeds is reused for each method to ensure the same initial states when testing a new method.

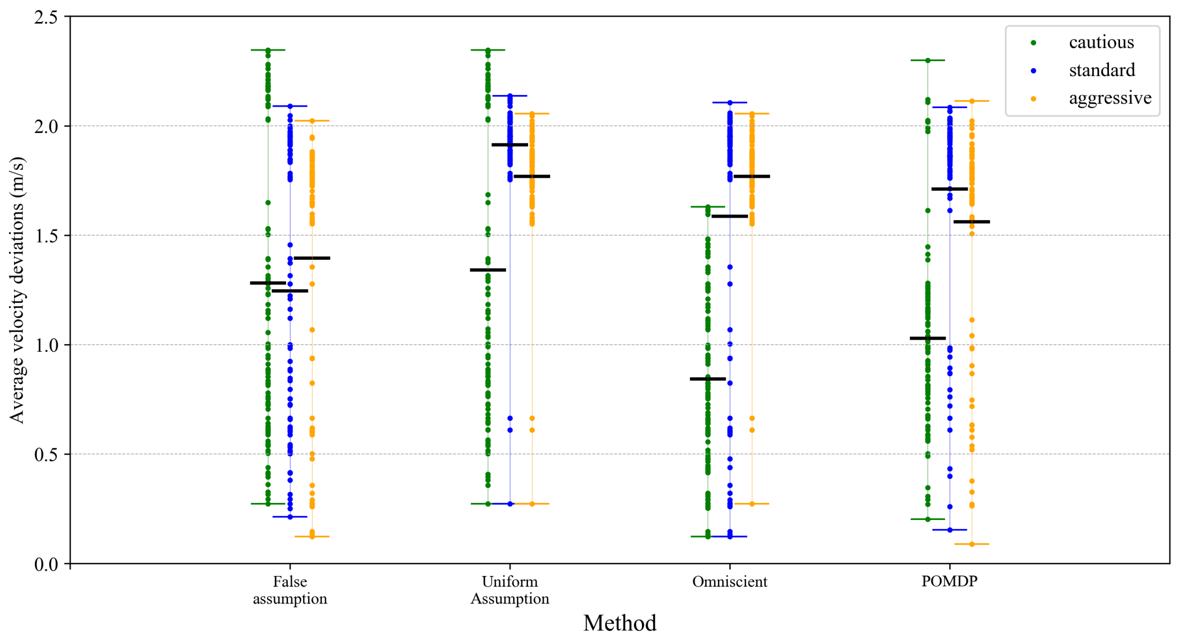


Figure 4.2 The average velocity error over each successful run, reported for each of the four proposed methods. Runs with cautious, standard, and aggressive actor styles are separated into different colored groups. The maximum and minimum values for each set of runs are highlighted at the top and bottom of the point groups with horizontal bars. The overall averages are highlighted with the wider black bars.

The measurements are reported in Tables 4.2 and 4.3. For each measure, a lower value shows higher performance.

The uniform assumption controller does not cause conflicts because it always acts conservatively, considering that there is always a possibility that the driving style is aggressive. The omniscient controller also does not cause conflicts because it knows the driving style with certainty and can act accordingly. The POMDP controller is also able to avoid any conflicts by accurately approximating the driving style of the actor in a short amount of time in all runs.

In the second measurement, the false assumption controller has the lowest velocity error overall. This is due to reckless driving in cases with standard or aggressive styles. In said cases, there is a chance that the controller assumes cautious behavior and takes the opportunity to pass. This leads to less stopping and yielding for the ego vehicle and lower velocity errors with respect to the reference on successful runs. However, there are many runs where this behavior for the ego leads to conflicts, as can be seen among the results for the first measurement. This disadvantage significantly overshadows its performance in the second measurement. The false assumption controller is a fine example where the balancing of the capabilities between navigating quickly and avoiding conflict fails in one direction.

The omniscient method performs second best since it possesses perfect knowledge about the actor and can select actions accordingly. Thus, this method's performance acts as a target for the POMDP method, which has to approximate the driving style dynamically.

The uniform assumption controller can be considered a realistic baseline for a lane excursion solution. It possesses no information about the actor's driving style and needs to act conservatively in every situation to avoid a conflict, given the chance that the driving style is more aggressive than cautious. This leads to more stopping and yielding behavior, which in turn increases the average velocity error in all types of runs. It can be seen that the POMDP controller performs better than this baseline in every case. This is because the POMDP can gather information while navigating the scenario and make better-informed decisions.

A deeper analysis of the average velocity error values can be made through Figure 4.2, where the average error is shown as a point for every successful run. Runs with different driving styles for the actor are grouped into different colored points, with their maximum, minimum, and average values highlighted as horizontal bars. It can be seen that there is a high variance in the error values; this is caused by the various randomized starting state velocities for the ego vehicle and the difference in yielding and passing behaviors. Yielding to the actor vehicle leads to higher velocity errors since the reference velocity guides the ego vehicle to move while the ego needs to stop until the actor has passed the conflict area. Passing first means that a lower average velocity error will be produced.

It can be observed in Figure 4.2 that most of the runs in the uniform assumption method end with the ego vehicle yielding and the error being higher as a result. As mentioned, the ego yields because, from the controller's point of view, there is always a chance that the actor will be aggressive and attempt to pass as well, which may lead to a conflict. In comparison, the POMDP controller produces more cases with correct yielding or passing decisions and, as a result, lowers the velocity errors for each driving style on average.

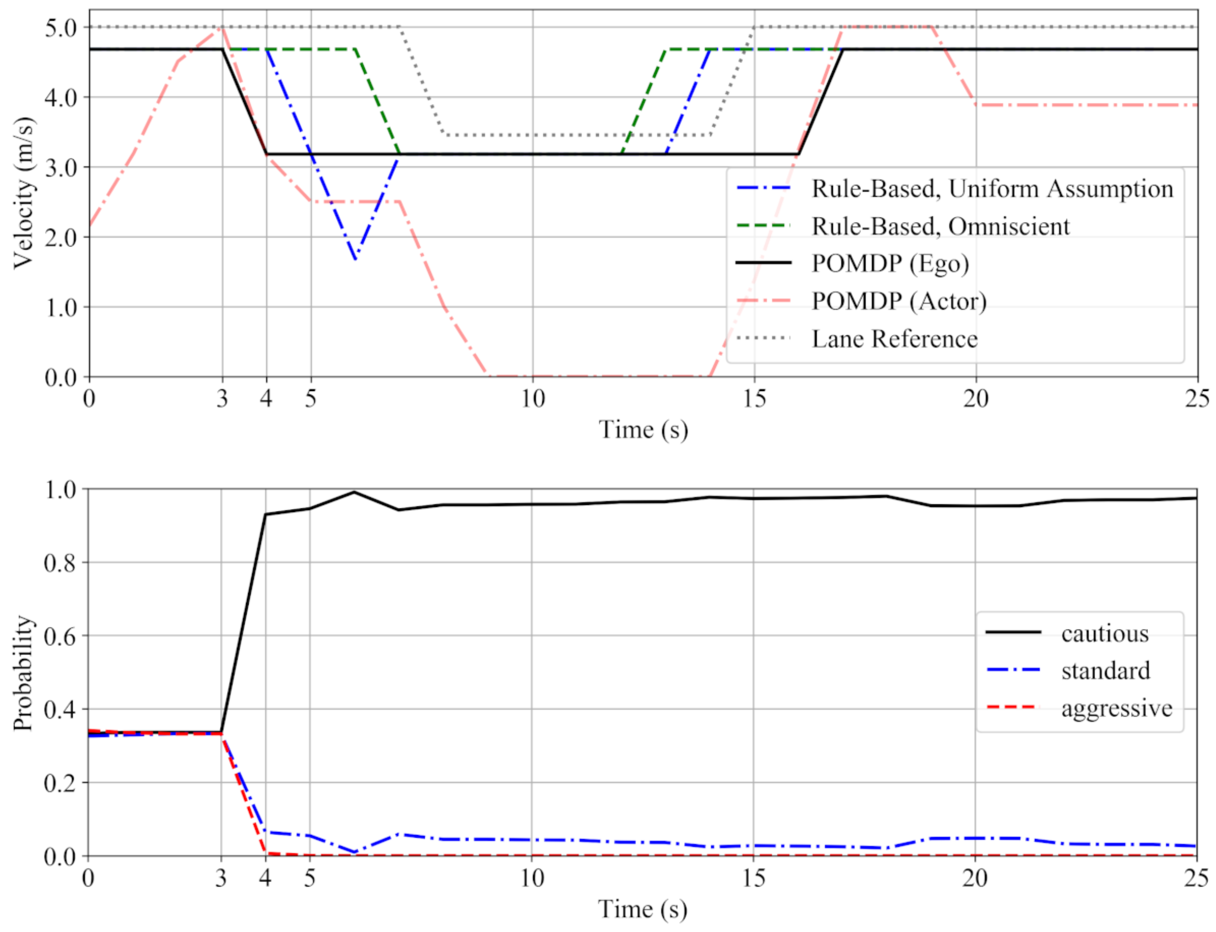


Figure 4.3 Data from a single run comparing the rule-based and POMDP methods, where the actor vehicle's driving style is "cautious." On the top, the velocity profiles over time are shown in comparison to the reference lane velocities. On the bottom, the POMDP belief (probability distribution) over the actor vehicle's driving style is shown.

For a detailed look into which actions the methods take during runs, the velocity profiles created during a single run are provided in Figure 4.3. The false assumption controller is left out for easier visualization. On the top, the ego velocities over time for each of the remaining methods are shown. The actor velocity for the POMDP method is also included. On the bottom, the belief state of the POMDP over the probabilities of the driving style of the actor is shown for the same run.

In this run, the actor vehicle is cautious, and the optimal approach is to pass the conflict area first with the highest velocity value allowed, which is seen as the velocity profile of the omniscient method. The POMDP controller starts to slow down one time step before the uniform controller ($t = 3$) since cruising at a lower speed allows it to gather information for a longer time, as opposed

Method	# Conflicts	Average velocity error
POMDP	0	1.433
POMDP + perception noise	1	1.418

Table 4.4 The POMDP evaluated in the default configuration, and with perception noise added in simulation. Conflicts are summed and velocity errors are averaged over 100 runs for each driving style, for 300 runs in total.

to reaching the conflict area as fast as possible and making a full stop after that. At $t = 4$, the actor vehicle starts to slow down due to its cautious behavior, and the belief of the POMDP solver shifts towards cautious as well. The future time steps confirm the POMDP solver's belief, and the ego takes the opportunity to pass the conflict area, while the uniform assumption controller has to slow down more to ensure that the actor vehicle cannot pass first, even if it is aggressive.

4.5 Effects of Observation Uncertainty

To test the capabilities of the POMDP model under added sources of uncertainty, another series of runs is performed where perception noise is introduced to the system. The noise is introduced during the observation step, making this a type of observation uncertainty. The observed arc distance is perturbed with a value taken from the normal distribution $\mathcal{N}(0, 2.5)$, and the observed velocity is perturbed using the distribution $\mathcal{N}(0, 0.5)$, which are plausible magnitudes of noise that could exist in a real-life system.

A basic delay-based error calculation was used to pick the variance for the velocity error: Given a very common autonomous system frequency of 10Hz, each observation will be 0.1 seconds later than its previous. In this duration, a vehicle may change acceleration values; then, the following observation becomes different from what is expected under a constant acceleration assumption. We consider the highest change in acceleration allowed by the simulation, ignoring emergency braking scenarios. This value is 3 m/s^2, leading to a maximum velocity change of 0.3 m/s in the 0.1s time frame. The distance error was modeled based on the size of the actor vehicle. In an extreme case where only one side of the vehicle was seen at a time step, the center of the bounding box could shift from its true position as much as half of the vehicle length, which is 1.86 meters in the simulation. These values both represent very improbable situations, but a higher variance than each of them was chosen as the error parameter to challenge the POMDP solver.

The measurements of the POMDP solver with and without perception noise are reported in Table 4.4. A consequence of integrating perception noise is that one of the 300 runs in total ends in a conflict, as opposed to none of them when the noise is not present. In this run, the actor vehicle is aggressive, and the ego vehicle estimates this correctly once again, then stops before the conflict area to yield the way. The actor vehicle passes through, and right before it completely clears, the ego vehicle starts moving and enters the conflict area as well, causing a conflict at the edge of the area. This is caused by the incorrect information provided to the ego vehicle, leading the solver to think that the actor vehicle is, in fact, completely past the conflict area. The incorrect information here causes the mistake in the first two steps of the time horizon and not the time steps farther into the future. Therefore, this failure does not indicate any weaknesses in the POMDP solver's prediction ability. Instead, it is an instant consequence of erroneous data provided.

Regarding the ability to follow the reference velocity profile of the lane, the POMDP seems to perform at the same level regardless of whether or not there is external observation uncertainty. This can be explained through the two following remarks: First, in the solution space that the POMDP solver searches, internal noise is introduced in the transition model, which leads to simulated variance in future time steps. When states similar to each other are considered the root belief, the beliefs in future time steps still cover very similar possibilities due to the high branching factor. This means that the future optimal actions and rewards reached from the root belief will likely be similar sequences when noise is introduced at the root step, which is what the observation uncertainty entails. Secondly, given a state in the lane excursion scenario, perturbing that state is not very likely to change the optimal action at closer time steps. More specifically, given an actor's velocity and distance, moving the actor or changing the actor's velocity without a major difference does not change the best action to take for the ego vehicle. For example, if the actor is aggressive, whether it is 10 or 12 meters away from the conflict area does not affect that the best action to take is to stop and yield. This is also true for velocities. When these two remarks are combined, it is observed that neither the closer time steps nor the time steps farther into the future are affected substantially, given a realistic amount of perception noise.

In conclusion, the predictive capabilities and performance of the POMDP controller are maintained even when some perception noise is introduced as the observation uncertainty. However, immediate effects, such as a conflict at the next time step, can be seen due to having inaccurate information. Though this problem is not specific to a POMDP solver and can happen with any other, future work can still be considered to address the higher possibility of conflicts.

4.6 Discussion on Real-life Application

As mentioned, research in autonomous vehicle literature about addressing problems specific to large vehicle types is scarce. Moreover, to the best of my knowledge, no effort has been made to address unavoidable lane excursions via longitudinal motion planning. Yet, with autonomous buses, trucks, and shuttles being more regularly deployed on roads, this scenario is likely to become a common occurrence during autonomous driving. Thus, the real-life implications of modeling this problem are discussed in this section.

The performance of the POMDP solver on navigating through a lane excursion scenario has been demonstrated throughout this paper. However, the evaluations were only done in a simulation environment. In real life, every step of communication and actuation introduces varying uncertainties to the whole problem. For instance, in a ROS environment, different packages are run under different processes, and there is always a communication delay between sending an action request and that action reaching the package that commands the vehicle. From the commandeering package to the vehicle, there are multiple physical layer steps that the message has to move through, which adds to the delay. Meanwhile, the simulation assumes that the action is instantly taken after being issued. To address this, in a real-life system the POMDP may need to estimate the delay to the actuator and simulate forward to issue the action at the delayed time earlier.

Another example is the vehicle dynamics being much more complicated than a bicycle model in real life. When doing particle propagation, the POMDP solver may need to consider the detailed vehicle dynamics equation. Especially for large autonomous vehicles, the vehicle's movement capabilities differ significantly from the common assumptions made in prior research and this thesis.

Even among a specific type of vehicle, such as a commercial bus, the dynamics characteristics differ according to factors such as tire placement [42].

Lateral motion, although ignored here, is also a source of uncertainty. As mentioned in the problem definition, the POMDP model is expected to be able to incorporate this range of motion without major changes since the conflict area is already pre-processed in this method, and the motion can be considered in this step. During the navigation of the scenario, the longitudinal arc distance will again be the important state variable, and the lateral motion will only add noise to the longitudinal distance traveled at each time step. The amount of uncertainty introduced this way is already handled by the variance in the transition model of the POMDP.

A major assumption made in defining the lane excursion problem is that the vehicles involved will adhere to traffic rules. In cases where these rules are broken by illegal maneuvers, the POMDP would not be able to forward simulate this into possible predictions since the state transitions are constrained by the traffic rules. This is an extreme case of uncertainty, but it is worth considering nonetheless when designing a generic approach. As a solution, the ABT solver may have a tiny portion of its belief particles propagated without the traffic rule constraints, allowing it to consider the possibility of reaching illegal states in the future. However, this may cause an unbalanced value estimation of each action, so a difficult fine-tuning process for reward parameters may be required. Another solution may be a higher-level decision tree reacting to illegal states in a rule-based manner and transferring control to the POMDP solver when the states are deemed legal, which may be the safer approach.

Moving on from uncertainties, there is also an issue of generalization. Multiple actor vehicles may need to be considered in a single environment simultaneously, in which case the current method would have to be run twice, whereas the better option is to extend the POMDP model to include multiple actors. In other cases, a single actor near a conflict area may not be visible to the ego vehicle due to occlusion spots. These cases have been addressed in previous works with intersection navigation problems [36, 39], which can also be adapted to this problem.

Generalization also requires the method to be activated dynamically whenever a possible conflict area is encountered. To this end, the conflict area can be extracted using prior map data as a pre-processing step or approximating the local map on demand through sensing. The POMDP method provided is able to use lane maps with different shapes as inputs. However, the type of scenario is limited to two-way, double-lane roads, so another challenge is to extend its capabilities to handle varying road types. In multi-lane roads, vehicles can change lanes which is a reliable way to avoid the conflict area, whereas this study was built on the assumption that passing through the conflict area is the only option for both vehicles involved. Multi-lane roads would require redefining the problem to accommodate vehicles moving in and out of the conflicting lanes.

As another generalization challenge, the driving styles of actor vehicles, although they cover a high variance of behaviors in this simulation, may not be able to completely represent the variance of driver behaviors encountered in real life. To address this, driving styles and their corresponding driving models can be extracted from real-life data, such as autonomous driving datasets, either with a rule-based data analysis method or a learning algorithm. After this, it must also be ensured that the POMDP solver is able to properly divide the belief space to distinguish different styles among a more extensive set.

One more interesting branch of autonomous driving research to consider is multiple autonomous systems interacting with each other. There may be a benefit to redesigning the lane excursion problem using two autonomous systems instead of one system interacting with an independent, manually driven vehicle. There have been many studies on managing intersections with conflict areas given multiple autonomous vehicles, with cases where they are able to communicate with each other or where a single infrastructure element makes the higher-level decision-making on pass priority, which can all be applied to the conflict area management in this problem as well [43].

In summary, many research opportunities exist in implementing the lane excursion problem in real-life autonomous vehicles. The set of assumptions made by the original problem definition is strict, but it can be relaxed by testing the removal of each assumption in isolated experiments. This way, the original problem definition can also be extended significantly to accommodate various road and vehicle geometries, vehicle interaction scenarios, and types of uncertainties.

CHAPTER 5

CONCLUSION

In this thesis, the lane excursion problem was introduced in the context of large autonomous vehicles. The main goal was to emphasize that as the research space on the autonomous driving industry expands, new perspectives on some long-standing problems need to be brought forward. One example is that large autonomous vehicles do not always have the capability of avoiding a lane excursion through trajectory shape optimization; thus, sometimes, it is required to assume that the excursion will inevitably take place and to plan longitudinal velocities under vehicle interaction constraints instead.

The lane excursion problem was formally defined, including a conflict area with undetermined passing priority and an opposing vehicle with a driving style that is non-observable to the autonomous system. Along with it, a "largeness" term is coined to be able to focus on vehicles with dimensions in specific ranges when addressing the lane excursion problem. Several methods to address the problem have been proposed. The main solution employs the capabilities of the POMDP framework to estimate the hidden driving style of the incoming vehicle and take close-to-optimal actions to navigate the conflict area. Other rule-based methods have been proposed as baselines for the POMDP method. Two measures to compare all methods have been proposed to evaluate them regarding safety and speed. Comparisons demonstrated the ability of the POMDP solver to generalize to different driving styles and randomized scenario states. Lastly, the behavior of the POMDP was analyzed in more detail with the aid of simulation data from specific runs and measures gathered under an additional observation uncertainty constraint.

The lane excursion problem is a frequent occurrence when the vehicle size is significant in the given road geometry, and it has the possibility to become even more pervasive as a higher number of large autonomous vehicles are deployed in the industry. My aim is to provide insight into how to solve this problem in a way that is generalizable to any type of autonomous vehicle deployment today. The proposed POMDP model successfully represents the fundamental challenges brought on by the problem definition: the non-observability of the whole system state and the complexity of dynamic vehicle interactions near conflict zones. However, there are still two major obstacles in transferring and generalizing the problem into real-life contexts: First, the POMDP model makes many assumptions to construct the problem, some of which need to be eliminated when building a generic solution. Second, real-life vehicle deployments are accompanied by a significantly larger number of characteristics and uncertainties that can affect the performance of a given solution, as opposed to the simulation environment presented in this study. To direct research efforts into clearing these two obstacles for the lane excursion problem, I included a discussion section detailing the major considerations associated with transferring the problem into real life.

My personal takeaways from this study were to widen my vision of domain-specific challenges for larger autonomous vehicles, to gain a deeper understanding of modeling and predicting vehicle interaction, and to achieve a level of expertise on MDP and POMDP solutions for the task of autonomous path planning. I first started my research on lane excursion due to it being a problem we frequently encounter during our autonomous deployments at ADASTEC. I wanted to improve my knowledge of a branch of planning algorithms that is applicable to real-life products in my career. As I worked on the thesis, I grew more aware of the potential of the proposed model to become a prevalent and useful tool to improve the safety of many autonomous deployments in operation today, ranging from regular passenger car types to buses and trucks. My next major goal is to focus on the aforementioned real-life implications and approach the POMDP method from an engineering perspective. I believe that this study in lane excursion modeling will be a consequential step toward safer and more efficient autonomous driving solutions in the future.

BIBLIOGRAPHY

[1] Aurora Innovation Inc., “The aurora driver.” [Online]. Available: https://aurora.tech/aurora-driver

[2] N. Verin and B. Sperling, “Ez10 passenger shuttle.” [Online]. Available: https://easymile.com/vehicle-solutions/ez10-passenger-shuttle

[3] National Association of City Transportation Officials, “Transit street design guide,” 2016. [Online]. Available: https://nacto.org/wp-content/uploads/2016/04/Turn-Radius_60ft-bus.jpg

[4] C. V. Zegeer, H. F. Huang, J. C. Stutts, E. Rodgman, and J. E. Hummer, “Commercial bus accident characteristics and roadway treatments,” *Transportation Research Record*, vol. 1467, pp. 14–22, 1994.

[5] D. Chimba, T. Sando, and V. Kwigizile, “Effect of bus size and operation to crash occurrences,” *Accident Analysis & Prevention*, vol. 42, no. 6, pp. 2063–2067, 2010.

[6] P. Polack, F. Altché, B. d’Andréa Novel, and A. de La Fortelle, “The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?” in *2017 IEEE intelligent vehicles symposium (IV)*. IEEE, 2017, pp. 812–818.

[7] R. F. Smith, “State of the practice and recommendations on traffic control strategies at toll plazas,” p. 24, 2006.

[8] National Association of City Transportation Officials, “Urban street design guide,” pp. 33–36, 2013.

[9] P. Trautman and A. Krause, “Unfreezing the robot: Navigation in dense, interacting crowds,” in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 797–803.

[10] K. Kant and S. W. Zucker, “Toward efficient trajectory planning: The path-velocity decomposition,” *The international journal of robotics research*, vol. 5, no. 3, pp. 72–89, 1986.

[11] K. J. Åström, “Optimal control of markov processes with incomplete state information i,” *Journal of mathematical analysis and applications*, vol. 10, pp. 174–205, 1965.

[12] C. H. Papadimitriou and J. N. Tsitsiklis, “The complexity of markov decision processes,” *Mathematics of operations research*, vol. 12, no. 3, pp. 441–450, 1987.

[13] O. Madani, S. Hanks, and A. Condon, “On the undecidability of probabilistic planning and infinite-horizon partially observable markov decision problems,” in *Aaai/iaai*, 1999, pp. 541–548.

[14] J. Pineau, G. Gordon, S. Thrun *et al.*, “Point-based value iteration: An anytime algorithm for pomdps,” in *Ijcai*, vol. 3, 2003, pp. 1025–1032.

[15] D. Silver and J. Veness, "Monte-carlo planning in large pomdps," *Advances in neural information processing systems*, vol. 23, 2010.

[16] H. Kurniawati and V. Yadav, "An online pomdp solver for uncertainty planning in dynamic environment," in *Robotics Research: The 16th International Symposium ISRR*. Springer, 2016, pp. 611–629.

[17] M. Memarzadeh, G. L. Britten, B. Worm, and C. Boettiger, "Rebuilding global fisheries under uncertainty," *Proceedings of the National Academy of Sciences*, vol. 116, no. 32, pp. 15 985–15 990, 2019.

[18] L. Pascal, M. Memarzadeh, C. Boettiger, H. Lloyd, and I. Chadès, "A shiny r app to solve the problem of when to stop managing or surveying species under imperfect detection," *Methods in Ecology and Evolution*, vol. 11, no. 12, pp. 1707–1715, 2020.

[19] R. Xie, Q. Tang, C. Liang, F. R. Yu, and T. Huang, "Dynamic computation offloading in iot fog systems with imperfect channel-state information: A pomdp approach," *IEEE Internet of Things Journal*, vol. 8, no. 1, pp. 345–356, 2020.

[20] Y. Zhu, S. Liang, M. Gong, and J. Yan, "Decomposed pomdp optimization-based sensor management for multi-target tracking in passive multi-sensor systems," *IEEE Sensors Journal*, vol. 22, no. 4, pp. 3565–3578, 2022.

[21] G. Arcieri, C. Hoelzl, O. Schwery, D. Straub, K. G. Papakonstantinou, and E. Chatzi, "Bridging pomdps and bayesian decision making for robust maintenance planning under model uncertainty: An application to railway systems," *Reliability Engineering & System Safety*, vol. 239, p. 109496, 2023.

[22] M. Chen, S. Nikolaidis, H. Soh, D. Hsu, and S. Srinivasa, "Trust-aware decision making for human-robot collaboration: Model learning and planning," *ACM Transactions on Human-Robot Interaction (THRI)*, vol. 9, no. 2, pp. 1–23, 2020.

[23] L. Holzherr, J. Förster, M. Breyer, J. Nieto, R. Siegwart, and J. J. Chung, "Efficient multi-scale pomdps for robotic object search and delivery," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6585–6591.

[24] J. Pajarinen, J. Lundell, and V. Kyrki, "Pomdp planning under object composition uncertainty: Application to robotic manipulation," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 41–56, 2022.

[25] K. Zheng and S. Tellex, "pomdp_py: A framework to build and solve pomdp problems," *arXiv preprint arXiv:2004.10099*, 2020.

[26] M. Egorov, Z. N. Sunberg, E. Balaban, T. A. Wheeler, J. K. Gupta, and M. J. Kochenderfer, "Pomdps. jl: A framework for sequential decision making under uncertainty," *Journal of Machine Learning Research*, vol. 18, no. 26, pp. 1–5, 2017.

[27] D. Klimenko, J. Song, and H. Kurniawati, "Tapir: A software toolkit for approximating and adapting pomdp solutions online," in *Proceedings of the Australasian Conference on Robotics and Automation, Melbourne, Australia*, vol. 24, 2014.

[28] L. Yu, D. Kong, and X. Yan, "A driving behavior planning and trajectory generation method for autonomous electric bus," *Future Internet*, vol. 10, no. 6, p. 51, 2018.

[29] R. Oliveira, P. F. Lima, G. C. Pereira, J. Mårtensson, and B. Wahlberg, "Path planning for autonomous bus driving in highly constrained environments," in *2019 IEEE intelligent transportation systems conference (ITSC)*. IEEE, 2019, pp. 2743–2749.

[30] W. Song, G. Xiong, and H. Chen, "Intention-aware autonomous driving decision-making in an uncontrolled intersection." *Mathematical Problems in Engineering*, 2016.

[31] K. H. Wray, B. Lange, A. Jamgochian, S. J. Witwicki, A. Kobashi, S. Hagaribommanahalli, and D. Ilstrup, "Pomdps for safe visibility reasoning in autonomous vehicles," in *2021 IEEE International Conference on Intelligence and Safety for Robotics (ISR)*. IEEE, 2021, pp. 191–195.

[32] H. Bey, M. Sackmann, A. Lange, and J. Thielecke, "Pomdp planning at roundabouts," in *2021 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops)*. IEEE, 2021, pp. 264–271.

[33] A. G. Cunningham, E. Galceran, R. M. Eustice, and E. Olson, "Mpdm: Multipolicy decision-making in dynamic, uncertain environments for autonomous driving," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 1670–1677.

[34] Z. Sunberg and M. J. Kochenderfer, "Improving automated driving through pomdp planning with human internal states," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 20 073–20 083, 2022.

[35] Y.-C. Hsu, S. Gopalswamy, S. Saripalli, and D. A. Shell, "A pomdp treatment of vehicle-pedestrian interaction: Implicit coordination via uncertainty-aware planning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 1984–1991.

[36] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic decision-making under uncertainty for autonomous driving using continuous pomdps," in *17th international IEEE conference on intelligent transportation systems (ITSC)*. IEEE, 2014, pp. 392–399.

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

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

[39] C. Zhang, S. Ma, M. Wang, G. Hinz, and A. Knoll, "Efficient pomdp behavior planning for autonomous driving in dense urban environments using multi-step occupancy grid maps," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2022, pp. 2722–2729.

[40] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, “Ros: an open-source robot operating system,” in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[41] B. Yücer, “Lane overflow pomdp planner,” https://github.com/adastec-batuhan/lane_overflow_pomdp_planner, 2024.

[42] Y. Zhang, A. Khajepour, and Y. Huang, “Multi-axle/articulated bus dynamics modeling: a reconfigurable approach,” *Vehicle System Dynamics*, vol. 56, no. 9, pp. 1315–1343, 2018.

[43] J. E. Siegel, D. C. Erb, and S. E. Sarma, “A survey of the connected vehicle landscape—architectures, enabling technologies, applications, and development areas,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 8, pp. 2391–2406, 2017.