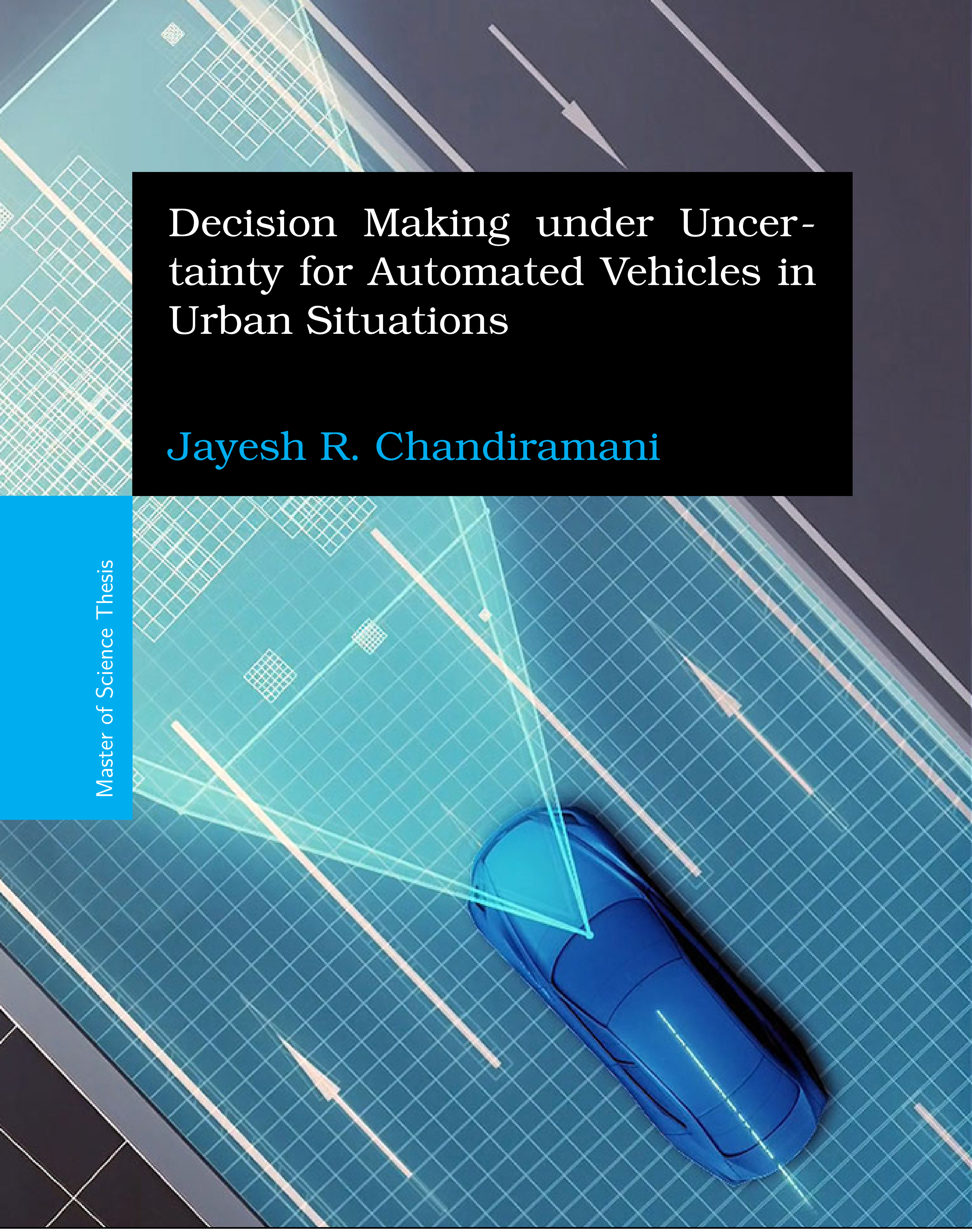


Decision Making under Uncertainty for Automated Vehicles in Urban Situations

Jayesh R. Chandiramani

Master of Science Thesis

Delft University of Technology

Delft Center for Systems and Control

Decision Making under Uncertainty for Automated Vehicles in Urban Situations

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Jayesh R. Chandiramani

November 7, 2017

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

The work in this thesis was supported by the PEM department at RWTH Aachen University. Their cooperation is hereby gratefully acknowledged.

Copyright © Delft Center for Systems and Control (DCSC)
All rights reserved.

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Decision Making under Uncertainty for Automated Vehicles in Urban Situations

by

Jayesh R. Chandiramani

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: November 7, 2017

Supervisor(s):

dr.ir. S. Baldi

ir. M. Sefati

Reader(s):

dr.ir. T. Keviczky

dr.ir. R. Happee

dr. L. Ferranti

Abstract

One of the most crucial links to building an autonomous system is the task of decision making. The ability of a vehicle to make robust decisions on its own by predicting and assessing future consequences is what makes it intelligent. This task of decision making becomes even more complex due to the fact that the real world is uncertain, continuous and vehicles interact with each other. Sensors that perceive the real world and measure quantities such as position and speed of other traffic objects are inherently noisy and are further susceptible to external conditions. On the other hand, the road users' intentions are stochastic and not measurable, and the presence of partial or complete vision based occlusions can make any measurements obtained useless. The decision making unit thus has to be aware of these issues and use the limited knowledge available to anticipate future situations that could unfold in an infinite number of ways to then maximize a reward (or minimize a cost).

In this thesis, a method to make automated longitudinal decisions along a predetermined path for autonomous vehicles in unsignalized urban scenarios is proposed. The decision making problem is formulated as a continuous Partially Observable Markov Decision Process (POMDP) with a discrete Bayesian Network estimating the behavior of other traffic objects. The future evolution of states are predicted using a multi-model Trajectory Estimator along with the digital map of the current driving area. Since continuous spaces make the belief space infinitely large, calculation of the value function for this problem becomes computationally intractable. The presented solver algorithm approximates the value function instead of computing it directly, and additional optimizations reduce the belief space exploration area in order to improve performance. The results show that this single generalized framework is able to handle all the tested urban scenarios with a good safety margin in scenes with multiple traffic objects, even under zero visibility of the other traffic objects due to the presence of occlusions.

Table of Contents

List of Figures

Acknowledgements

This MSc thesis marks the end of my work as an MSc student at Delft University of Technology and brings to conclusion two exciting years. I would like to take this opportunity to thank the entire DCSC department at TU Delft and the PEM department at RWTH Aachen University for providing a stimulating environment and for keeping student's motivation high.

I would specifically like to thank my supervisors Mohsen Sefati (Lehrstuhl fur Production Engineering of E-Mobility Components, RWTH Aachen), and Simone Baldi for their continuous support and guidance throughout the research project, while pushing me to achieve more than I could have done myself.

I would also like to express my deepest appreciation to my family for their unconditional love, support and trust. I would also like to express gratitude to all my friends for the support and the direct or indirect help they provided me with during my thesis.

Delft, University of Technology — Jayesh R. Chandiramani
November 7, 2017

Chapter 1

Introduction

1-1 Motivation

Globally, the number of on-road fatalities stand at a staggering 1.25 million per year [2]. While Europe provides a relatively small contribution to this number at 84,600 fatalities per year [2], it is still a significant number. Additionally, the number of vehicles on the road also steadily increases, as does the time spent by each individual traveling each year. This shows the importance of increasing the safety and efficiency of vehicles and this is reflected in the fact that development with regard to Automotive Driver Assistance Systems (ADAS) has been progressing at a rapid rate over the last few decades. More and more vehicles are being equipped with increasingly sophisticated ADAS systems and more recently, automated vehicles have gained the spotlight as fully autonomous vehicles become a reality.

The first big push towards research and development of automated vehicles came with the DARPA Urban Challenge in 2004 [3], where numerous research teams presented their autonomous vehicles against each other, and the efforts have been increasing substantially since then. When Google received permission to test its prototype autonomous vehicle for the first time in 2012 [4], it was a significant step forward to a safer and more efficient future. Currently, there are numerous research groups and companies working hard on bringing this technology to market, as governments around the world prepare to modify their road laws for this eventuality.

In order to describe the architecture of such a system, let us first look at the tasks performed by a driver. According to [5], these are: Navigation, Guidance and Stabilization. Navigation involves being able to use map data to find a path from point A to point B, with some consideration of traffic levels and length of route to minimize the overall time taken. Guidance refers to the process of perceiving the world, detecting other road user's intentions and then making decisions such as the desired velocity and choice of lane based on this information. The stabilization level relates to aligning the vehicle motion with the selected guidance parameters within an acceptable level of accuracy. Expanding these basic tasks into a generalized system architecture, this results in the structure seen in figure 1-1. According to this, the automated vehicle is expected to work on its own to localize itself globally, define its mission, plan a route using map data and cooperate with other road users by predicting their intentions. It also needs to take other static and dynamic objects into consideration to finally accomplish its mission while ensuring driver safety.

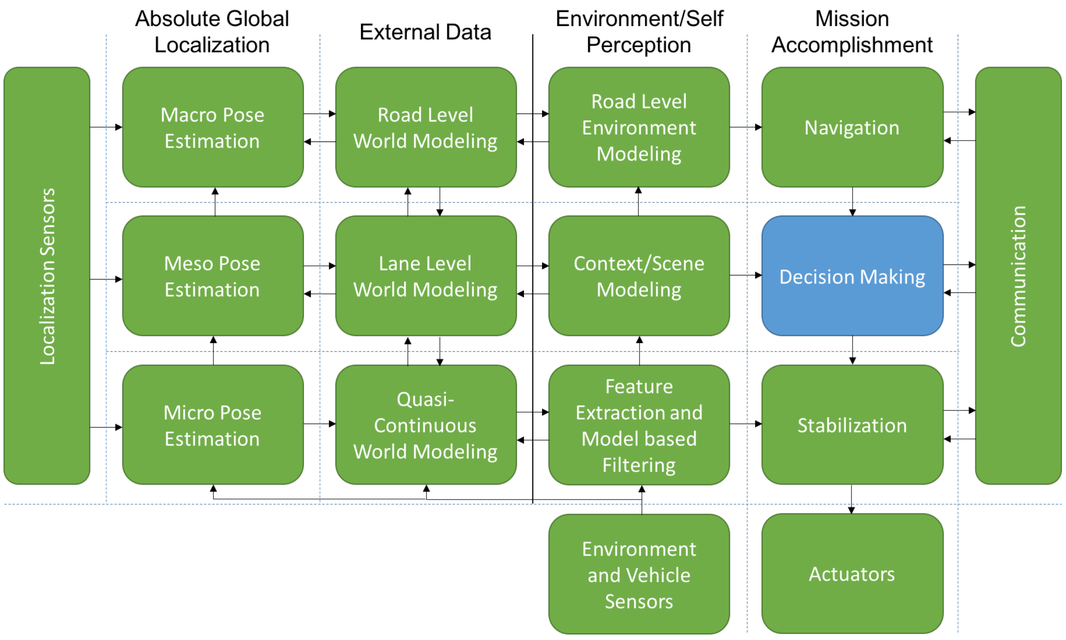

Absolute Global Localization
External Data
Environment/Self Perception
Mission Accomplishment
Localization Sensors
Macro Pose Estimation
Road Level World Modeling
Road Level Environment Modeling
Navigation
Meso Pose Estimation
Lane Level World Modeling
Context/Scene Modeling
Decision Making
Micro Pose Estimation
Quasi-Continuous World Modeling
Feature Extraction and Model based Filtering
Stabilization
Communication
Environment and Vehicle Sensors
Actuators

Figure 1-1: Autonomous Driving System Architecture [1]

Within this structure, the task of decision making is especially vital as it converts a thought (such as traveling from point A to point B along a path) into action. It does this by considering the evolution of the current scene and weighing the actions against each other to choose the most optimal one that would result in a safe and efficient journey. Generally speaking, there are two main categories of decision making methods - Group and Individual [6]. In autonomous driving terms this is referred to as centralized and decentralized decision making methods. Centralized methods are applied under the assumption that vehicles are able to communicate with each other or with some infrastructure, while decentralized methods rely on their perception of the environment to make decisions. Further, decision making for autonomous vehicles is split into two subcategories - for lateral and longitudinal motion. Lateral motion essentially deals with lane keeping or lane changing, while longitudinal motion deals with braking, accelerating or keeping a constant velocity along a path.

Although the process of decision making seems simple on paper, it must be considered that there are numerous challenges along this process when it comes to autonomous vehicles. The first challenge is the consideration of the uncertainty in the scene. Autonomous vehicles use sensors such as cameras, RADAR and LIDAR to gather information about the environment and thus predict the evolution of vehicles around it. The problem here is the fact that these sensors are always susceptible to noise, faulty measurements and obstacles blocking their vision. Further, information such as the driver's intention is not measurable and is usually stochastic as human drivers could change their intention at any moment. The second challenge is that along with safety and efficiency, the comfort of the passengers within the vehicle must also be accounted for and a smooth ride is highly desirable. For example, an overly conservative strategy in a crowded area could lead to uncomfortable, jerky vehicle behavior. Finally, the decision making unit must not be limited to certain scenarios such as single lane or highway driving. For fully autonomous driving, it must be applicable to all types of scenes and scenarios.

In this thesis, the goal is to develop a decentralized decision making framework for longitudinal motion within urban scenarios. To demonstrate that the framework is applicable to all urban situations, scenes such as the ones depicted in figure 1-2 are used for testing and validation of the work.

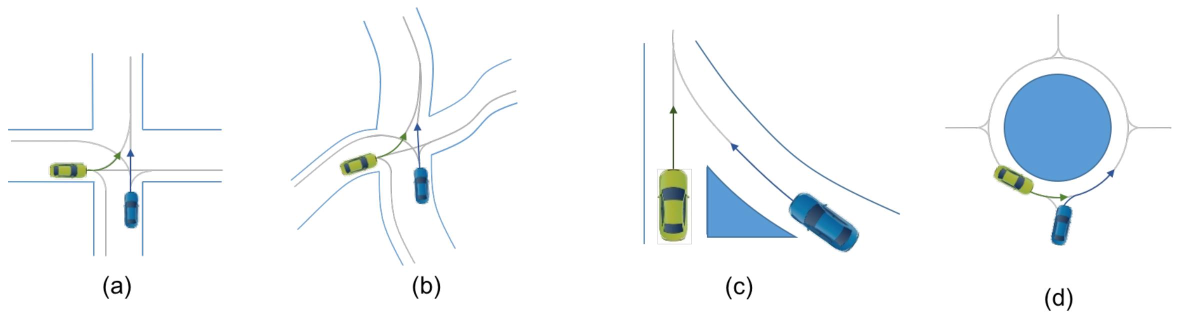


Figure 1-2: Longitudinal motion scenarios
(a) Perpendicular intersection (b) Curved intersection (c) Zipper Merge (d) Roundabout

1-2 Related Work

1-2-1 Future State Prediction

The first step in choosing the most optimal action to execute is being able to predict what the vehicles in the near vicinity intend to do and thus what their physical states would be. The current states (position, velocity, yaw angle) can be measured using sensors in the perception subsystem, using which a prediction must be made for each time step over the total horizon.

According to [7], future state prediction models can be divided into three categories:

Physics based models

These are the most direct models, as they simply model vehicles as dynamic entities that obey the laws of physics. Future positions are calculated using dynamic or kinematic models using the current state measurements.

The most commonly used models from this category are the Constant Velocity (CV) and Constant Acceleration (CA) models [8][9][10] which use basic equations of motion to make linear predictions of future states along with velocity and acceleration based simplifications respectively. They usually result in errors in applications with longer time horizons or slow sampling rates, but are computationally inexpensive and can be applied to certain cases such as constant highway driving very effectively.

The Gaussian noise representation of the uncertain vehicle state is popular [11][10] as it is used in the Kalman Filter, which is a method to iteratively update a vehicles state using a linear evolution and sensor model, along with a normal distribution for consideration of uncertainty. The combination of prediction and update steps for each new measurement results in a mean trajectory for the vehicle.

Another method in this category is Monte Carlo simulations [12], where random sampling of variables from the evolution model generates future trajectories for the vehicle. The advantage of this method is that future trajectories can be obtained using certain or uncertain vehicle state knowledge.

Maneuver based models

These models assume that a vehicle is an independent entity performing a specific maneuver such as a lane change or a turn. In order to do this, either prototype trajectories must be generated [13] or the intention or behavior of the vehicle is required. This calls for methods such as a Hidden Markov Model [14], Bayes classifier [15] or a Support Vector Machine [16] to be used for the task of behavior prediction.

In [17], 8 distinct maneuver classes are identified: Turn left, Turn right, Lane Change left, Lane Change right, Follow road, Follow vehicle, Target brake and Trash that are the outputs of a discrete Bayesian network (BN). The trash class relates to highly uncertain maneuvers and is meant to model highly stochastic behavior such as that of a drunk driver. The authors use a separate model for each of these classes and include a noise term to model the variation and uncertainty of the state variables over longer time horizons. This is very advantageous as it does not simply apply a single model to various maneuvers and also the inherent consideration of uncertainty expands the potential decision making methods that could be used (at the cost of the decision making approach needing to perform multiple iterations).

Overall these models are much more accurate than the simple physics based models especially over longer prediction horizons, but they come with an added computational overhead and require careful selection of training data.

Interaction based models

This class of models assume vehicles to be entities that interact with each other and influence each other's maneuvers. This leads to a more sophisticated approach with a better understanding of the vehicle motion within the scene. An example of such an approach is seen in [18], where a Dynamic Bayesian Network (DBN) is deployed to consider all the available information and then predict the intention of the driver. Using this, the future state of the vehicle is predicted along with all the associated uncertainty. This makes the approach much more reliable over longer prediction horizons but again at the cost of higher computational overhead, which becomes even more significant in densely populated urban areas with a lot of traffic. Additionally, the problems relating to collecting a large amount of training data and making assumptions while building the network itself remain.

1-2-2 Decision Making

The earliest autonomous vehicles focused on computer vision, control architecture and integration rather than on outright decision making [19]. While the complexity of the decision making task was recognized to some extent at the time, the first DARPA Challenge in 2004 brought it into focus and propelled development in this area [3]. In this part some of the main decision making approaches specifically in the field of autonomous driving are explored. Note that some of the methods discussed in this section employ behavior and future state prediction methods that are similar to the ones discussed earlier.

Manual Programming

Finite State Machines (FSM) and Rule Based manual programming approaches are the simplest type of decision making approaches but they are very effective even today as they allow for a clear method of failure analysis and testing. Some of the best teams at the DARPA Challenges such as Team AnnieWay [20] used an FSM based approach where states included driving behavior and state transitions occurred as per manually written conditions. Another example is Tartan Racing Team ("BOSS") [21] who used an event triggered automatic state machine based approach.

In Germany, Mercedes-Benz has demonstrated the S500 Intelligent Drive research vehicle that successfully covered a distance of 100km between Mannheim and Pforzheim, encountering some complex urban traffic situations along the way [22]. They used a hierarchical state machine approach to generate time and space constraints based on assuming the worst case scenario for each and fed this information directly into the trajectory planner.

These methods are very efficient and easy to understand and debug. The drawback is that the only way that uncertainty can be considered in the method itself is by assuming the worst case scenario at all times.

Optimisation

The first method is by defining a cost function and evaluating each action or sequence of actions using optimization algorithms in order to find the one with the lowest cost. The autonomous driving use case usually results in a highly non linear cost function so the simpler optimization algorithms (such as linear programming or convex optimization) cannot be used. Instead, methods such as Chance Controlled Optimization in [23] are used to efficiently solve lane change overtaking in urban areas. In this case, other road user's behavior is modeled using a Markov linear system but formulated as a discrete time stochastic hybrid system. In [24] the authors integrate receding horizon control into game theory in order to make the decision dependent on both the current and future predicted uncertain information. A game is defined for lane changes and reachability analysis is then used to include an upper and lower bound on the vehicle position at each time step while evaluating actions.

Graph Search

Graph search methods are another class of approaches that have been used in many fields. Here a tree-like graph is created to model different decisions and their consequences in order to choose the optimal action. In [25], a simple rule based classifier is used to model other road users and the A* graph search method is then used with invisible collision states as a heuristic. This method was implemented in BMW's Highly Automated Driving Framework simulator as well on the road. Another approach is using Rapidly Explored Random Trees (RRTs). In [26], the authors use cooperative perception to perform motion planning via an RRT based framework. First the map is divided into a grid using the Occupancy Grid Map and then a cost map is created to keep the vehicle in the center of its lane on the road and planning is then handled by the Anytime RRT algorithm. In [27], the motion planner uses Sparse Stable Trees with the RRT* method to simplify the problem by pruning non-useful nodes and MPC.

Model Predictive Control

The next method is using Model Predictive Control (MPC), which is a continuous space control method that uses a dynamic model to predict future events and minimizes a cost function to find the optimal solution. [28] first models driving as a hybrid system, and uses inverse reinforcement learning to predict behavior. An MPC like cost function is then sought to be minimized in order to make lateral lane change decisions. In [29], automation of lane change maneuvers is looked at as a longitudinal planning problem. By first using a trajectory planner to check if a longitudinal trajectory exists in order for the ego vehicle to be positioned safely in a gap in the target lane and then planning a lateral trajectory, the problem complexity is reduced and MPC is applied more effectively in the lateral and longitudinal direction. [30] uses the Interactive Multi Model-Extended Kalman Filter (IMM-EKF) method to estimate and a maximum likelihood prediction of other road user's states is got using the Kalman filter equations. Lane change and lane keeping cost functions are defined and the driving mode is derived by comparing these two costs. Finally, the lower level MPC considers disturbances and uncertainty to complete the decision making task.

In general, these approaches are advantageous because they allow for a large action set to be used but they ordinarily do not have the provision to consider uncertainty.

Partially Observable Markov Decision Process

The next method is using the Partially Observable Markov Decision Process (POMDP) which is based on probability theory. The aim is to maximize the total expected reward over a time period by evaluating all action sequences and considering their resulting effects. A probability distribution over the states is used in order to allow for the consideration of uncertainty in the decision making process. This is usually computationally expensive to solve but a lot of research has been done to try and find ways to simplify the problem, such as the team from TU Braunschweig that uses relative distances, velocities and time to collision with various objects within a signal processing network to provide the POMDP with inputs such as "lane change possible" and "lane change beneficial" to then find the optimal lane change action [31]. The work done by this research group was further extended with the inclusion of a behavior planning framework, a situation prediction model and also enabled variable time steps in the planning process [32]. Other approaches try to reduce the number of branches in the decision tree and reduce the time needed to find the optimal solution. For example, "DESPOT" is one of these solvers that has gained popularity due to its effectiveness [33][34]. In [35], the authors use Monte Carlo planning and reduce the complexity of the continuous POMDP autonomous driving problem at an intersection by iteratively developing an effective low dimensional discrete representation of the state space. This allows for a significant speed up of the decision making problem but this representation needs to be formulated offline, which is a problem as it needs to be continuoually updated if there is a change in the map.

Overall, this approach is extremely advantageous as it allows all sources of uncertainty to be modeled but real time implementation is usually a problem due to the computational complexity.

Deep Learning

Finally, there are deep neural networks, or simply "Deep Learning" (DL) approaches, which have been gaining popularity in recent years across a variety of applications. However, due to the secretive nature of automotive companies, there is a lack of scientific publications in this area with respect to autonomous driving.

For example, Google/WayMo has been developing an approach in this direction since 2009 and have been collecting a few million miles of driving data [36]. Tesla has also been collecting driving data in 2014 and then after they rolled out their driver assistance package in 2015, they have continued to collect more data especially when driver intervention is required [37]. Unfortunately neither of these companies have published any scientific papers regarding their work. nVidia has released some information about their end to end approach in [38], where they present a neural network with 9 layers that determines steering angles based on video input. The network is said to have been trained on images from a front facing camera and steering wheel input from a human driver, but further details are minimal.

The problems with DL approaches are the amount of training data required to actually train the network under all possible scenarios and the fact that the resulting system is basically a black box [39]. This is a major problem, as it hinders failure testing and performance under untrained scenarios cannot be guaranteed. In the case undesired behavior is detected, more data specific to that scenario must be obtained and the system must be retrained. Automotive companies in general are opposed to this approach, especially as end-to-end solutions as there is no clear understanding of how the underlying network works. Instead, there is a general preference for transparency and well defined rules when it comes to subsystems interacting with each other.

1-3 Research Questions

From the current state of the art, it is clear that there are a multitude of approaches to solving the decision making and behavior modeling problem. The main criteria for choosing an approach in this thesis are:

- Generality: an approach that could be applied to all types of driving scenarios
- Traceability: allowing for easy failure analysis and debugging

- Uncertainty handling: allowing for sensor, environmental and behavioral uncertainties to be taken into consideration
- Real Time Capability: so that the method can be used in real world application

Narrowing the approaches down based on this, POMDP appears to meet most of our requirements for decision making, while BNs paired with maneuver based models are a good fit for future state prediction. Our further research questions are formulated as follows:

1 Is it possible to adapt and optimize state of the art continuous online POMDP solver algorithms to the autonomous driving problem such that they can be run in real time?

2 Can a probabilistic approach effectively address the stochastic nature of the problem using the predicted intentions and future states of traffic vehicles?

3 Along with a good safety margin, is it possible to obtain an acceptable level of driving efficiency from this approach?

4 Can invisible vehicles, occluded from the sensors via obstacles in the environment be addressed within the decision making approach?

1-4 Outline

Chapter 2 introduces the concept of Bayesian networks and details the Junction Tree algorithm that simplifies the process of making inferences within these networks. Following this, the behavior prediction problem is formulated within the Bayesian network structure and a multi motion model approach to future state prediction using the output of the network is covered in detail.

Chapter 3 introduces POMDPs and the underlying complexity of finding a solution to a problem modeled as a continuous POMDP. Following this, a state of the art online solver is presented, along with further performance improvements to bring it closer to real time performance. Finally, the autonomous driving problem is modeled as a POMDP and the finer details within this structure are discussed in detail.

Chapter 4 begins with the description of the simulation setup and the parameters used for the decision making framework. Then several test scenarios are defined and the results of the behavior prediction and decision making framework is presented and analyzed in detail.

Chapter 5 presents a summary of this MSc thesis, highlighting the contributions and discussing areas of further development.

Chapter 2

Behavior Prediction and Trajectory Estimation

The main reason a behaviour estimator or predictor is required is to provide the decision making algorithm with information regarding the future positions and velocities of traffic objects. Without this information, it would only be possible to proceed by weighing all possible maneuvers equally, but this is a worst case approach. In this work a discrete Bayesian Network is used to provide the decision making unit with a set of probabilities corresponding to a set of discrete maneuver classes for each of the n traffic objects.

This chapter is organized as follows: Section 2-1 introduces Bayesian networks and the related algorithm used to make inferences from a set of observations. Section 2-2 presents an approach for behavior prediction adapted from [17]. Section 2-3 discusses how the output of this network can be used to predict the future evolution of the scene.

2-1 Bayesian Network Theory

2-1-1 Introduction to Bayesian Networks

Bayesian networks (BNs) [40] are a probabilistic graphical representation of knowledge in an uncertain domain. They are essentially a directed acyclic graph (DAG), which is a graph in which all edges connecting nodes have a specific direction with no cycles formed within it. In these graphs, each node represents a variable and the edge that links two nodes represents a probabilistic dependency between the corresponding variables. They allow for effective computation of the joint probability distribution (JPD) over the set of variables used.

A BN, uses the DAG structure and the parameters of the model that have to be specified - ie. the qualitative and quantitative parts. These parameters follow the Markov property, wherein the conditional probability distribution (CPD) of each node depends only on its parent's nodes [41]. For discrete variables, this results in a table of values which lists the local probability that a child node takes on a specific value for each combination of its parent's values. The JPD of a set of variables can be uniquely derived from these CPD tables.

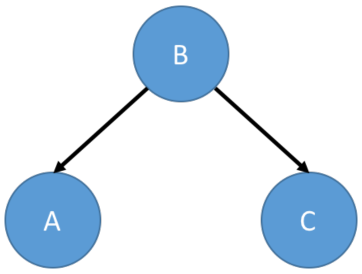


Figure 2-1: Basic Bayesian Network

Figure 2-1 depicts a simple example of a BN [42]. The set of edges $E = (B, A), (B, C)$ along with the nodes $X = A, B, C$ form a DAG as they satisfy the two requirements: all edges have a direction and form no cycles. From the figure, it can be concluded that A and C are conditionally independent and thus $P(A|B,C) = P(A|B)$. This means that the probability of A is conditioned only based on the value of B and the value of C has no relevance here. The same is true for C: $P(C|B,A) = P(C|B)$. The joint distribution of all the variables in the BN is:

$$P(A,B,C) = P(A|B)P(B)P(C|B) \tag{2-1}$$

Putting this in a general form, for nodes $X = X_1, X_2, ..., X_n$, the joint probability function is:

$$P(X) = \prod_{i=1}^{n} P(X_i|parents(X_i)) \tag{2-2}$$

Thus, the directed edges give the graph meaning by telling us what other variables a specific variable is conditioned upon. Depending on whether X_i has parents or not, it is said to be conditional or unconditional respectively and if the node (variable) is observed then it is called an evidence node, else it is said to be hidden.

2-1-2 Inference in BNs

The information about the previous nodes that appears as the posterior in the mathematical expression is termed "evidence" and these evidence nodes are filled using the observations received. The process of computing the values of the other nodes in the BN using this evidence is called inference. Making inferences in a BN is usually NP-hard but there are various methods such as Monte Carlo, Gibbs Sampling and Helmholtz machine that either perform belief propagation or approximate inferences to make this process more efficient [43].

In this work, the Junction Tree Algorithm (JTA) [42][44] is used for inferences. The idea behind the algorithm is to create a representation of the graph such that variables are clustered together in a singly connected graph. This allows for message passing and variable elimination schemes to be applied easily in order to make inferences.

The JTA basically consists of the following steps:

- Moralization - it is the process of "marrying" the child nodes to the parents. For each child node in the initial graph, undirected edges are added between all its parents. This would result in an undirected graph, with all parents connected of a given node to each other, as in figure 2-2.

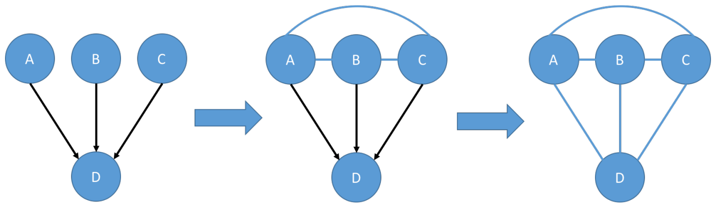


Figure 2-2: Moralization

- Triangulation - this is the process of creating a chordal graph (also called a triangulated graph), in which each cycle with more than 3 vertices has an edge that connects two vertices of the cycle. This is also called a chord. Thus it essentially is the addition of edges to the undirected graph.

  If the graph is not triangulated, the "fill-in" process is performed on the basis of the maximum number search to create a triangulated graph structure. In the cycle ABCD in Figure 2-3, it is seen that AC and BD are not directly connected to each other. Therefore, the fill-in must be performed, in which case either a new edge BD or AC is added, so that the graph structure is triangulated.

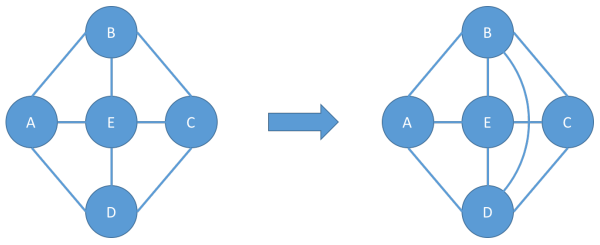


Figure 2-3: Triangulation

- Create the junction tree - The first step is the formation of cliques. A clique is a subgraph in which all vertices are connected in pairs. Several nodes of the subgraph are then combined to form a common node called a clique node. The clique nodes then represent a new quantity which is used in the so-called potential representation instead of the original conditional probabilities. With this formulation the probability distribution can then be expressed in the form with several potential functions $\psi_i(C_i)$:

$$P(V) = K \prod_{i=1}^{m} \psi_i(C_i) \tag{2-3}$$

  where K is a normalization factor such that $\sum_V P(V) = 1$.

  In order to connect cliques to each other and to create a clique graph (which is illustrated as a non-directed graph with cliques as nodes), separators are needed. This is understood to be the number of nodes that describe the intersection of two arbitrary cliques. It can be seen from Figure 2-4 that nodes ABC, BCD, and CE are interconnected. They are then transformed into common cliques, which are linked via separators BD and C.

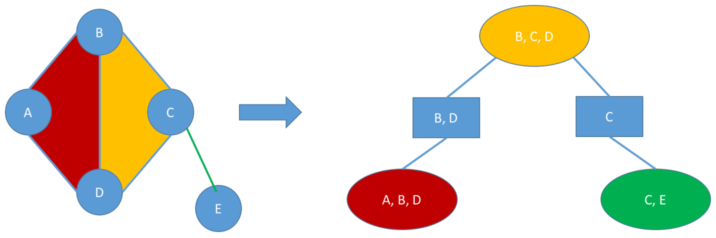


Figure 2-4: Clique graph formation

With a triangulated graph the distribution P can also be factorized according to:

$$P(V) = \frac{\prod_{C_i} P(C_i)}{\prod_{S_j} P(S_j)} \tag{2-4}$$

where C_i are the cliques and S_j are the separators of a clique tree of the graph. This global product representation can be split into smaller local distributions to reduce complexity [45].

Since the algorithm is a tree, some edges or separators must be eliminated to create a clique tree that fulfills the "running intersection property" (RIP) [44]. According to this property, if all nodes between a pair of nodes V and W in the clique tree contain the intersection $V \bigcap W$, then the clique tree is a junction tree. From the clique graph, a maximum spanning tree T_{max} is now sought for the RIP property. The weighting of the edges w_i is defined as the number of elements in the separator i. Mathematically this looks like:

$$\sum_{w_i \in G} w_i(T_{max}) = max \tag{2-5}$$

Since the standard methods are applied only to determine the minimum span, the weights in the clique graph can be negated or subtracted from a number. The problem is now transformed into a search for a minimal span tree. In Figure 2-5, cliques are connected by weighted edges. In order to form the tree, the left and lower edges are required in this example since the sum of weights in the spanning tree reaches a maximum value.

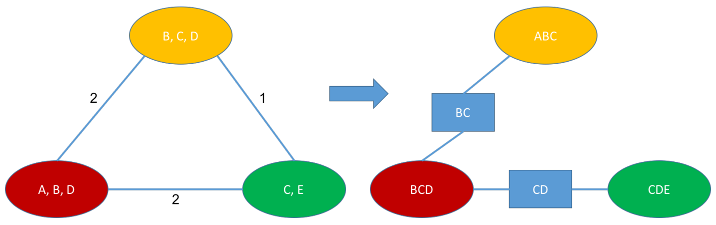


Figure 2-5: Building a clique tree

- Potential assignment - In order to calculate further with a potential representation, parameter assignment has to be done. The clique is chosen such that it's set and it's parents belong to the same clique.

  If there are p cliques in total, the potential function of Clique i with $1 \leq i \leq p$ is described in the potential representation by:

$$\psi(C_i) = \prod_{V:V=C_i} P(V|parents(V)) \tag{2-6}$$

- Absorption - The process of message passing from one clique to another is called absorption. A root clique is first chosen arbitrarily from the set of cliques, then information is passed from the leaf cliques up toward the root. Then the information is passed down to the leaf cliques again. Thus all cliques receive the information from all other cliques and the calculation of the joint probability is reduced to the elimination of variables in individual cliques.

  For the clique tree in figure 2-6, C_i is the child clique of C_j and they are connected by separator S_{ij} with the property $C_i \setminus S_{ij} = U_i$, $C_j \setminus S_{ij} = U_j$. For the up-calculation (marked with a single asterisk), the first update of the cliques and separators is used:

$$\psi^*(S_{ij}) = \sum_{U_i} \psi^*(C_i) \tag{2-7}$$

$$\psi^*(C_j) = \frac{\psi^*(S_{ij})}{\psi(S_{ij})} \psi(C_j) \tag{2-8}$$

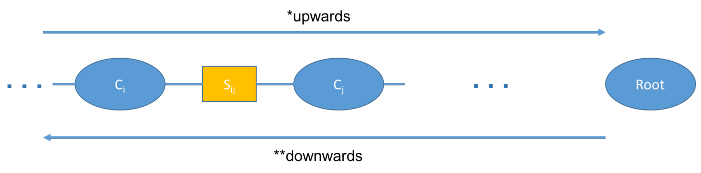


Figure 2-6: Message passing in a junction tree

If clique C_j absorbs information from several separators at the same time, then:

$$\psi^*(C_j) = \frac{\prod_i \psi^*(S_{ij})}{\prod_i \psi(S_{ij})} \psi(C_j) \tag{2-9}$$

In the case of the downward calculation (marked with a double asterisk), the second update is used:

$$\psi^{**}(S_{ij}) = \sum_{U_j} \psi^*(C_j) \tag{2-10}$$

$$\psi^{**}(C_i) = \frac{\psi^{**}(S_{ij})}{\psi^*(S_{ij})} \psi(C_j) \tag{2-11}$$

After the absorption is completed:

$$\psi^{**}(C_i) = P(C_i) \tag{2-12}$$

$$\psi^{**}(S_{ij}) = P(S_{ij}) \tag{2-13}$$

Thus, the global composite probability $P(V)$ is reduced to two single composite probabilities $P(C_i)$ and $P(S_{ij})$ and then the probability calculation is performed by variable elimination.

The JTA can be summarized in the following figure:

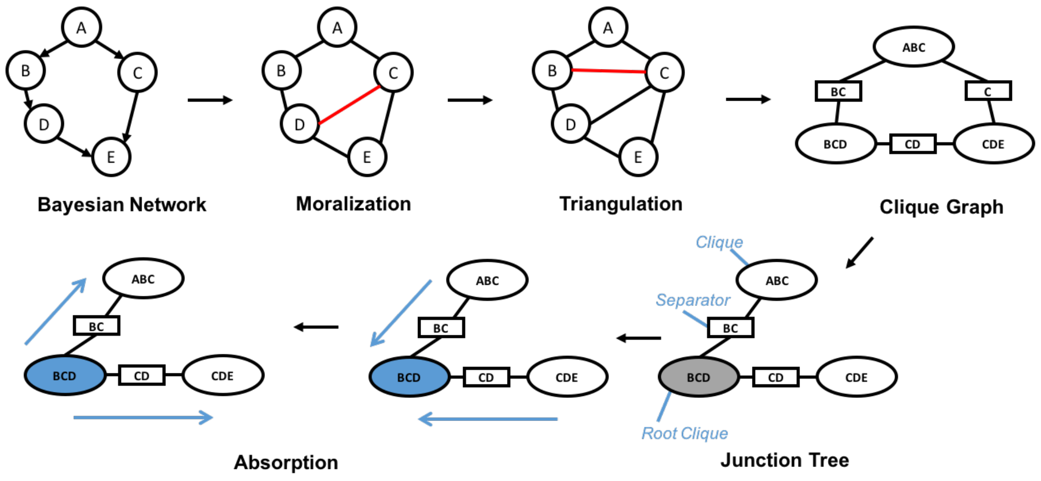


Figure 2-7: Summary of the junction tree algorithm

2-2 Applying discrete BNs to estimate driver behavior

In order to make effective predictions regarding the intention of either a densely populated scenario or a single other vehicle on the road, the fact that drivers make mistakes must be considered [17]. While the idealistic assumption could work in simulation, it could be dangerous out in the real world when a driver does not follow the traffic rules perfectly, leading to a rare event. Thus, the regular approaches that end up modeling the average, sensible driver are not always suitable as they would exclude these rare events or emergency situations. Instead, if a separate network is instantiated for each vehicle and each network receives observations/evidence independent from other traffic participants then the network would designed under the premise that drivers overlook each other and make mistakes. If there is physical evidence that proves otherwise, then that evidence plays a role in the behavior prediction.

The basic set of maneuvers that drivers use are: Lane change (left/right), Turn (left/right), Follow road, Follow vehicle, Target Brake and an additional Trash maneuver class for highly stochastic driving behavior. While the names of these maneuvers make them self explanatory, a visualization of these classes is seen in figure 2-8.

These maneuvers are formulated as nodes within the network and the resulting network, adapted from [17] can be seen in figure 2-9. It consists of three main layers, with 8 maneuver nodes corresponding to the 8 classes (green), 6 helper nodes (gray) and 12 evidence nodes (blue). Continuous variables such as position and velocity discretized into intervals to make the entire network a discrete BN. The hidden layer is embedded between the causal layer

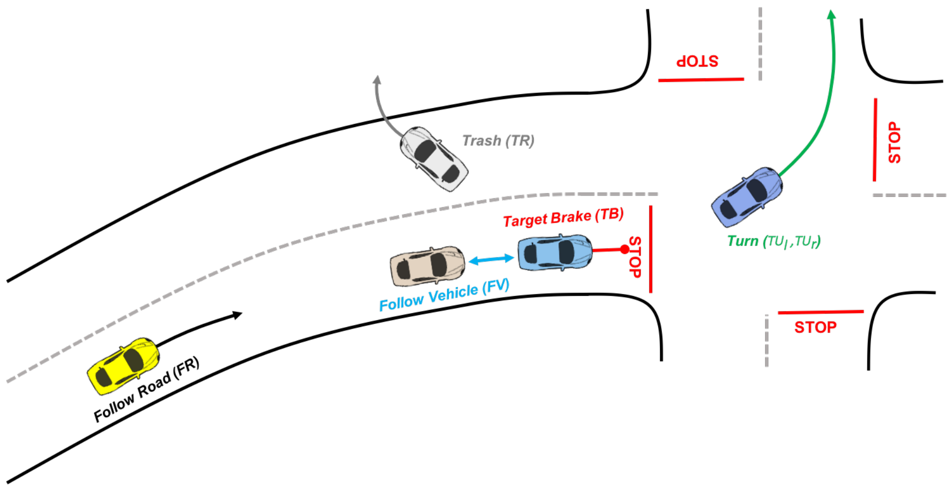
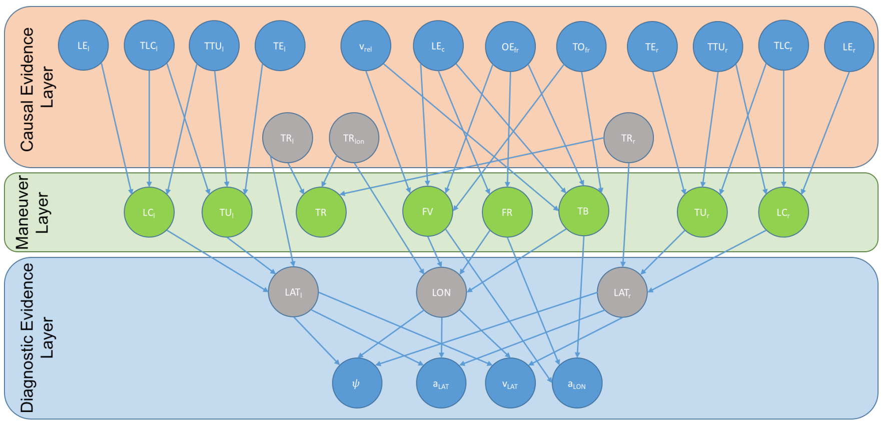


Figure 2-8: Comparison of maneuver classes



Variable	Explanation
$LE_{l/r/c}$	Lane existence (left/right/current)
$TLC_{l/r}$	Time to lane crossing (left/right)
$TTU_{l/r}$	Time to Turn (left/right)
$TE_{l/r}$	Turn evidence (left/right)
v_{rel}	Relative velocity to object ahead
OE_{fr}	Object existence in front
TO_{fr}	Time to object in front
$TR_{l//LON/r}$	Trash helper node (left/longitudinal/right)
$LC_{l/r}$	Lane Change (left/right)
$TU_{l/r}$	Turn (left/right)
TR	Trash
FV	Follow Vehicle
FR	Follow Road
TB	Target Brake
$LAT_{l/r}$	Lateral motion (left/right)
LON	Longitudinal motion
ψ	Yaw angle
a_{LAT}	Lateral Acceleration
v_{LAT}	Lateral Velocity
a_{LON}	Longitudinal Acceleration

Figure 2-9: Discrete Bayesian network for vehicular behavior prediction

and the diagnostic reasoning layer as this allows for inter-causal reasoning to be applied. The causal layer models the requirement for a specific maneuver, such as the existence of a neighboring lane being a requirement for a lane change maneuver. On the other hand, the diagnostic layer models each maneuver's consequences as measurable states, such as vehicle acceleration (lateral/longitudinal), yaw rate and velocity (lateral). This greatly simplifies the corresponding CPTs and allows for explaining away - if the diagnostic layer's observed states are common to both a turn and a lane change maneuver, then the causal knowledge of a turn not existing increases the probability of the lane change maneuver. This also allows for the creation of the Trash maneuver class, when all of the other maneuvers have been explained away - for example, in the case of a drunk driver's movement. The Trash helper nodes assist with the explaining away.

The entire network is updated with the available evidences and measurements at each new timestep and the JTA is applied to make inferences for the discrete random variables. Then the maneuver nodes' values are normalized so that they form a single state with a valid probability for each of the 8 maneuvers.

For this specific vehicular behavior prediction application, the formulation of the junction tree itself can be done offline, and only the absorption steps need to be performed online at each timestep when new observations are obtained. Thus the offline process was done in MATLAB using the Bayes-net Toolbox [46] and the resulting network along with the online processes were programmed using C++. The network's tables were filled with values obtained from [17], where a similar BN was trained with actual road data to derive the values.

2-3 Multi motion model approach for future state prediction

Since the output of the behavior predictor contains the probabilities for 8 different driving maneuvers, a different maneuver based motion model can be applied per maneuver while taking into account the environmental evidences to calculate the future state. Since the uncertainty in the position will be accounted for in the decision making algorithm, it does not need to be considered at this point. The models chosen are thus relatively simple, and can be replaced by more sophisticated models in the future, if deemed necessary.

In general, there are two main coordinate systems to choose from: the Global Coordinate System (GCS) and the Road Coordinate System (RCS). The difference between these two is the reference and what the coordinates represent. The advantage of the GCS is that it is slightly more intuitive and allows for easier collision detection but it tends to be quite tedious when calculating vehicle positions in scenes in complex road networks. In these cases, the RCS ends up being easier to use along in order to calculate the future position of the vehicle along the route. Thus, the GCS is first transformed to the RCS, the state transition is performed and then the coordinates are transformed back to the GCS. Note that the map information is vital to performing this transformation and the corresponding transition in the RCS.

Thus the models corresponding to each of the maneuver classes are:

- **Follow Road:** A Constant Acceleration (CA) model is chosen as it able to calculate the future longitudinal position of a vehicle that is following the road in its own lane.

$$\begin{bmatrix} s'_i \\ v'_{i,lon} \\ a'_{i,lon} \end{bmatrix} = \begin{bmatrix} 1 & T & 0.5T^2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_i \\ v_{i,lon} \\ a_{i,lon} \end{bmatrix} \tag{2-14}$$

  where s_i refers to the longitudinal position (in the RCS), $v_{i,long}$ and $a_{i,long}$ refers to the longitudinal velocity and acceleration respectively of the i^{th} vehicle. For convenience and easier reading, henceforth the extra notation is omitted. The acceleration value used is directly from the initial state and is passed down untouched, while the velocity and position is calculated each time. The lateral offset t (in the RCS) is assumed to remain the same over the entire prediction.

- **Follow Vehicle:** Here it is assumed that the driver reacts to the vehicle in front in order to keep a reference time gap between the vehicles. The difference between the predicted time gap τ_i and the reference time gap τ_r: $\Delta\tau = \tau - \tau_r$ is used to find the corresponding acceleration to maintain this gap:

$$a_{lon} = \begin{cases} c_{FV} a_{min} & if \quad \Delta\tau < 0 \\ c_{FV} a_{max} & if \quad \Delta\tau \geq 0 \end{cases} \tag{2-15}$$

where a_{max} and a_{min} are the maximum and minimum comfort acceleration/deceleration values for a general vehicle, and c_{FV} is the scaling factor based on the deviation from the reference time gap:

$$c_{FV} = min\left(1; \frac{\Delta\tau^2}{2\tau}\right) \tag{2-16}$$

This allows for the assumption that the driver will accelerate more smoothly when there are positive time gaps, and decelerate harder when there are negative time gaps. The velocity and position is then calculated using the corresponding CA model equations.

- **Target Brake:** This model assumes that the vehicle is going to stop at the designated braking target, such as a stop line at an intersection for example. Thus the predicted acceleration value must be calculated such that the vehicle actually stops at this point from its current velocity and position:

$$a_{i,lon} = max\left(\frac{-v_{i,lon}^2}{2\Delta d}; a_{min}\right) \tag{2-17}$$

where Δd is the distance to the braking target from the current position of the vehicle. The velocity and position are then calculated using the CA model.

- **Lane Change (right/left):** A half sine model can effectively simulate a vehicle performing a lane change in the RCS, using the maneuver start coordinate s_{start}, the width of the lane w_{lane} and the reference maneuver length l_{ref}. The lateral offset is calculated by:

$$t = 0.5 w_{lane} sin\left(\frac{\pi}{l_{ref}}\Delta s - 0.5\pi\right) + w_{lane} \tag{2-18}$$

note that t is the lateral offset in the RCS, and $\Delta s = s - s_{start}$. The longitudinal offset requires the yaw angle to be calculated first:

$$\psi = \arctan\left(\frac{dt}{d\Delta s}\right) = \arctan\left(\frac{w_{lane}\pi}{2l_{ref}}\cos\left(\frac{\pi}{l_{ref}}\Delta s - 0.5\pi\right)\right) \tag{2-19}$$

Now at each time step k the above equations are re-used and the model is adapted to the current lateral position and yaw angle by setting $t = t_k$ and $\psi = \psi_k$ tp solve for l_{ref} and Δs:

$$l_{ref} = \frac{w_{lane}\pi}{2\tan\psi_k}\cos\left(\arcsin\left(\frac{2t_k}{w_{lane}} - 2\right)\right) \tag{2-20}$$

$$\Delta s_k = \left(0.5 + \frac{1}{\pi}\arcsin\left(\frac{2t_k}{w_{lane}} - 2\right)\right) l_k \tag{2-21}$$

Once the end of the half sine is reached, the CA model is used.

- **Turn (right/left):** This maneuver is split into three parts - the approach to the turn, the turn itself and the part after the turn has been executed. For the first and third parts, the road is considered to be straight near the turn itself and the information about the turn, such as the radius r_c, and start s_{start} and end s_{end} points of the turn are known from the digital map.

  For the first phase, the current velocity is compared to the reference turn velocity which is calculated such that the driver does not experience more than a pre-set value of lateral acceleration during the turn:

  $$v_{lon,max} = \sqrt{a_{comf,max}(r_c + \Delta r_c)} \tag{2-22}$$

  If the current velocity is lower than or equal to this reference, the acceleration value is set to zero but if the velocity exceeds it, then the deceleration is calculated to correspond with the driver adjusting his speed to match this reference velocity. This is then used in a CA model.

  For the second phase, a Constant Turn Radius (CTR) model is used. The map information is used to calculate the future position of the vehicle by first calculating the angle around the turn and the longitudinal distance covered at that speed:

  $$\psi_{turn} = \frac{s_{end} - s_{start}}{r_c} \tag{2-23}$$

  $$x_{k,lon} = v_{k,lon} \Delta t_k \tag{2-24}$$

  $$x_k = x_{k-1} + x_{k,lon} \cos \psi_{turn} dir \tag{2-25}$$

  $$y_k = y_{k-1} + x_{k,lon} \sin \psi_{turn} dir \tag{2-26}$$

  Where $dir = \dfrac{v_k}{|v_k|}$

  Finally, in the third phase, the driver is assumed to accelerate back up to the speed limit thus a comfort acceleration value is then used in the CA model.

- **Trash:** Since this maneuver class describes highly irrational behavior, there is no single model that can accurately predict the future position of such a vehicle. Thus, a CTR and CA model is combined for this, directly using the measurements from the state and the corresponding model equations above.

2-4 Summary

The future state prediction problem has been investigated and a solution has been proposed by first predicting the behavior of the driver, followed by applying a state estimation model specific to the predicted maneuver.

Bayesian networks took up the role of behavior prediction, and the junction tree representation allowed for inferences within the network to be made as effectively as possible. The vehicle's future state is then predicted with higher accuracy than standard approaches by using a motion model specific to the each chosen maneuver.

Chapter 3

Decision Making

Based on the overview on the current state of the art decision making approaches in the field of autonomous driving in Chapter 1, it is clear that a probabilistic approach such as POMDPs are the most suitable approach as it can be applied to all driving scenarios, has a good level of transparency and most importantly allows for the full consideration of uncertainty.

This chapter introduces the concept of POMDPs, followed by the proposed solver algorithm. Following this, the autonomous vehicle decision making problem is formulated as a POMDP and the approaches taken towards dealing with occluded vehicles and detection of collisions between vehicles are described.

3-1 Introduction to Partially Observable Markov Decision Processes

A POMDP is a decision making method that models an agent that has to perform a series of actions to maximize its expected rewards over a specific time horizon. The main advantage of POMDPs is the consideration of uncertainty via the concept of "partial observability", which is the assumption that the agent cannot directly observe the underlying state and therefore a probability distribution over the set of possible states is used. This probability distribution is then updated based on a set of transitions, observations and their respective probabilities.

Formally, POMDP is defined by the tuple

$$(S, A, T, Z, O, R, \gamma)$$

where:

- S represents the set of states
- A corresponds to the action set
- O is the observation space

- $Z(s', a, o) = p(o|s')$ is the observation model which is the probability that the agent in receives observation o when action a was taken to move to state s'. This is based on sensor uncertainty
- $T(s, a, s') = p(s'|s, a)$ is the transition model which is the probability that the agent ends up in state s' if action a is taken from state s. This is based on uncertainty in the system dynamics and the environment
- $R(s, a)$ is the reward function which is a real number returned when an action a is taken in state s
- γ is the discount factor in the range $[0, 1)$ that weighs the current reward and the rewards further in the future relative to each other.

At $t = 0$, the agent begins in state s and executes action a. It then moves to state s', receives a reward r and observes o. The agent's imperfect knowledge of its state is represented as a probability distribution over the states, called a belief. The initial belief is created directly from s and is then updated based on the action executed and observation received:

$$b'(s') = \tau(b, a, o) = \frac{p(o|s')}{p(o|b, a)} \int_s p(s'|s, a)b(s) \tag{3-1}$$

This is similar to a particle filter update, with η chosen to represent the normalization factor:

$$b'(s') = \eta Z(s', a, o) \sum_s T(s, a, s')b(s) \tag{3-2}$$

This process then repeats until all possible actions and corresponding observations have been explored. This results in a large tree containing nodes that represent belief states, and the edges connecting these nodes are the actions and observations that result in the child belief node from the parent belief node. The goal of the POMDP is to explore all these possibilities and assess the value of the expected rewards in each branch to finally choose the branch that results in the highest expected reward over the entire horizon. A visualization of the corresponding tree is shown in figure 3-1.

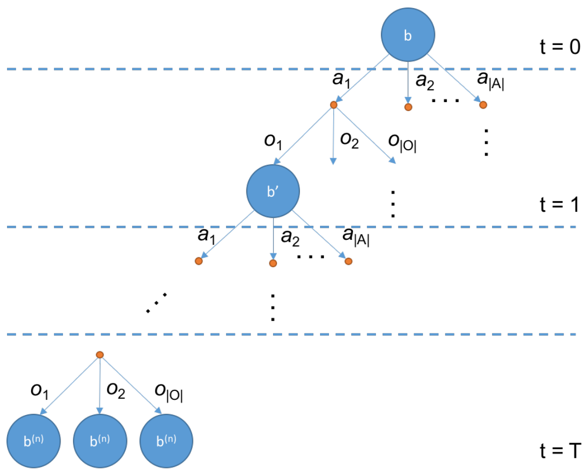


Figure 3-1: POMDP Belief Tree

In a POMDP, a policy $\pi : B \rightarrow A$ represents a mapping of beliefs $b \in B$ to actions $a \in A$. The optimal policy is thus the branch within the belief tree that is predicted to result in the largest expected reward. The value function represents the sum of the current and future rewards as follows:

$$V_n(b) = \max_a Q_n(b, a) \tag{3-3}$$

where n is the planning horizon and

$$Q_n(b, a) = \int_s r_a(s)b(s) + \gamma \sum_o p(o|b, a)V_{n-1}(b'(s')) \tag{3-4}$$

This recursion is called the Bellman equation and it converges to the optimal value V^*. The corresponding optimal policy is thus defined as:

$$\pi^*(b) = \operatorname*{argmax}_a Q^*(b, a) \tag{3-5}$$

for the optimal Q function Q^* associated with V^*.

It must be mentioned here that the above equations are applicable to continuous POMDP spaces. In the case of discrete state, action and observation spaces, the equations change slightly as the integrals are replaced by summation terms over the respective spaces. While this makes it clear that solving a discrete POMDP is much easier, continuous spaces are chosen in this thesis as they are much more intuitive for the problem at hand. Another reason is that in order to use discrete spaces, the entire real continuous world would need to be discretized in a way that the Markov property is still considered valid [47]. An approach for this is seen in [35], but requires a massive amount of computation and has to be done offline. This would be problematic in a new or modified area in the map.

Moving forward to solving the POMDP, there is a choice between online and offline solvers. Offline solvers are not bound to real time constraints and develop a policy by exploring the entire state space over thousands of iterations. This is far from ideal when looking at a large problem such as autonomous driving, as the size of the continuous state space is infinitely huge and thus the computation time and storage requirement becomes too large. This brings us to online solvers, which start only with a default policy and are expected to solve the POMDP in real time. This is of course computationally intensive, especially when continuous spaces are used and integrals are to be dealt with. While there is no solution to solve the continuous autonomous driving problem in real time via a POMDP, there has been significant progress in this regard to approximate a continuous value function via discrete sampling [48].

Although on first glance it seems that the value function calculation is computationally intractable, it has been shown that for a discrete POMDP, the value function calculation can be reduced to [49]:

$$V_n(b) = \max_{\alpha \in \Gamma} \sum_s \alpha(s)b(s) \tag{3-6}$$

where Γ is the set of α vectors. Saving these vectors in a set for future use is a form of learning and simplifies exploration of similar states in the future. An intuitive explanation of this can be found in the appendix. The main reason this is being discussed is that there exist approaches for solving continuous POMDPs that make use this property, thus resulting in a form of learning and result in lower computational expense.

The following section presents a state of the art solver algorithm that uses this approach and further modifications relating to the problem at hand are proposed in order to increase performance efficiency.

3-2 Solver algorithm

There has been a lot of research in the area of solving POMDPs efficiently and as of today, the fastest approaches to solve a continuous POMDP are point based methods that approximate the continuous value function instead of evaluating it directly. These methods are based on the principle of sampling the continuous belief and solving a discrete POMDP problem to approximate the continuous value function [50]. At each time step, the belief is sampled, the expected reward for each sample is obtained and weighed with the probability corresponding to each sample. The sum of these weighted rewards are then the approximated value function. As the number of samples increase, a more accurate approximation is obtained at the cost of a higher computational demand.

The current state of the art methods in this category are: Monte Carlo Value Iteration (MCVI) [48], Determinized Sparse Partially Observable Tree (DESPOT) [51], Real-Time Belief Space Search (RTBSS) [31] and Heuristic Search Value Function (HSVI) [52]. While these approaches are similar in the sense that they are all point based methods, they are vastly different in their actual implementation. More information about these solvers and a comparison between them can be found in the literature review done for this thesis.

In this thesis, the MCVI is chosen as a base solver because of its novel concept of policy graphs, which allow for previously explored states to be learned from in an iterative manner. It uses a so called Monte Carlo (MC) backup operation along with point based planning that allows the error in the point based planner to be bound while finding the optimal value function. Further performance improvements are added to this algorithm to bring it closer to real time performance.

3-2-1 Monte Carlo Value Iteration

As the same suggests, this POMDP solver uses random Monte Carlo sampling of the belief state to perform point based approximation of the value function [48]. The main concepts within this approach are the MC backup and the idea of learning from explored states using policy graphs.

Policy Graphs

While α vectors are advantageous with discrete POMDPs and show a considerable speed up, they cannot be used with continuous spaces as they result in hardware storage problems due to the amount of data produced. [48] proposes the concept of representing these α vectors implicitly in a so called policy graph G whose nodes are labeled with actions $a \in A$ and edges are labeled with observations $o \in O$. This can represent a policy in the POMDP by leveraging the advantages of α vectors from discrete POMDPs but without the data storage problems that are associated with using α vectors with continuous spaces. When such a policy graph is executed, the agent performs action a from an initial belief b, then receives an observation o and transitions from node v to node v' following the edge corresponding to the observation and this process repeats. Each node v of the policy graph represents an α function α_v, and the value $\alpha_v(s)$ is the reward of executing a policy π from an initial state s:

$$\alpha_v(s) = R(s, \alpha_v) + E(\sum_{t=1}^{T} \gamma^t R(s_t, a_t)) \tag{3-7}$$

Thus the value function V_n can be completely determined using the α functions associated with G:

$$V_n(b) = \max_{v \in G} \int_s \alpha_v(s) b(s) ds \tag{3-8}$$

MC Backup

From the basic concepts, it is known that the optimal value function V^* can be found through value iteration. Each iteration is called a backup H and is represented as:

$$V_t(b) = HV_{t-1}(b) = \max_{a \in A}(R(b,a) + \gamma \sum_{o \in O} p(o|b,a) V_{t-1}(b')) \tag{3-9}$$

where $R(b,a) = \int_{s \in S} R(s,a)b(s)ds$. At each belief $b \in B$, the backup H looks ahead and chooses the optimal action that maximises the sum of expected rewards over the current and next step.

As mentioned earlier, the value function explicitly is not represented explicitly as a set of α functions, but instead implicitly as a policy graph. Let us denote this value function as V_G for the current policy graph G [48]:

$$HV_G(b) = \max_{a \in A}(R(b,a) + \gamma \sum_{o \in O} p(o|b,a) \max_{v \in G} \int_{s \in S} \alpha_v(s) b(s) ds) \tag{3-10}$$

Considering that α vectors are not used explicitly, it seems that the original problem of evaluating the integral on the right hand side of the above equation still exists. However, the entire right hand side of this equation can be evaluated via sampling and MC-simulation to construct a new policy graph G'. This is then called MC-Backup of G at belief b. Algorithm 1 elaborates this process in detail. It must be noted that a strong advantage of this is that the MC backup only requires a relatively small number of samples to work effectively [48].

Upper and Lower Bounds

In order to have a measure of the accuracy of the approximated value function, the concept of an upper and a lower bound is introduced, which are represented by $\overline{V}$ and $\underline{V}$ respectively. The lower bound is essentially the maximum value function extended over the belief space but the upper bound is represented by a set of points that are calculated separately and

Algorithm 1 Backing up a policy graph G with N samples taken from belief b

```
1: function MC_BACKUP($b, N, G$)
2:     for all $a \in A$ do
3:         $R_a \leftarrow 0$
4:     end for
5:     for all $a \in A$ do
6:         for all $o \in O$ do
7:             for all $v \in G$ do
8:                 $V_{a,o,v} \leftarrow 0$
9:             end for
10:        end for
11:    end for
12:    for all $a \in A$ do
13:        for $i = 1$ to $N$ do
14:            Sample $b$ to get state $s_i$, with probability $b(s_i)$
15:            Apply transition model to action $a$ taken in state $s_i$. Sample from transition
16:            model $T(s_i, a, s_i')$ and observation model $Z(s_i, a, o_i)$ to get $o_i$ and $s_i'$.
17:            $R_a \leftarrow R_a + R(s_i), a$
18:            for all $v \in G$ do
19:                $V' \leftarrow$ Reward of policy represented by $G$
20:                $V_{a,o,v} \leftarrow V_{a,o,v} + V'$
21:            end for
22:        end for
23:        for all $o \in O$ do
24:            $V_{a,o} \leftarrow \max_{v \in G} V_{a,o,v}$
25:            $v_{a,o} \leftarrow \text{argmax}_{v \in G} V_{a,o,v}$
26:        end for
27:        $V_a \leftarrow (R_a + \gamma \sum_{o \in O} V_{a,o})/N$
28:    end for
29:    $V^* \leftarrow \max_{a \in A} V_a$
30:    $a^* \leftarrow \text{argmax}_{a \in A} V_a$ Create $G'$ by adding new node $a^*$ to $G$.
31:    for all $o \in O$ do
32:        Add edge $(a^*, v_{a^*,o})$ to $G$.
33:    end for
34:    return $G'$
35: end function
```

interpolated over the space. As the belief space is explored, these bounds are updated and the difference between these two bounds at a certain point express the uncertainty of the expected value at that specific belief point. When the difference is below a threshold ϵ, an accurate enough approximation is achieved and no advantage would be obtained by continuing exploration with more samples taken from that belief.

There are a number of ways to initialize these bounds, varying in complexity depending on the amount of information used to form a potentially tighter bound. For the lower bound, the simplest method is the Blind policy method [31] that essentially takes a single action and repeatedly applies it regardless of belief state, until the horizon is reached. Another approach is to always assume that the agent is in the worst state to choose action a from, which gives us equation 3-11 for an infinite horizon problem [52].

$$\underline{V} = \sum_{t=0}^{\infty} \gamma^t \min_s r(s,a) = \frac{\min_s r(s,a)}{1-\gamma} \tag{3-11}$$

For the upper bound, there are numerous initialization techniques proposed in [53]. First, there is the fixed strategy approach wherein the best possible value is assigned to every belief, but this is a relatively loose initialization of the upper bound compared to other approaches such as the MDP method, which assumes full observability and then solves the MDP version of the problem. With this, the bound would look like equation 3-12

$$\overline{V}(b) = \sum_{s \in S} b(s) V^*_{MDP}(s) \tag{3-12}$$

where

$$V_{i+1}^{MDP}(s) = \underset{a}{max} \left\{ \rho(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) V_i^{MDP}(s') \right\} \tag{3-13}$$

Further approaches exist, such as the Fast Informed Bound (FIB) method which accounts for the partial observability to some extent and improves the approximation, but although it produces a 5x tighter initial bound than the MDP method, it takes roughly 50x longer to compute [53].

In this thesis a compromise between bound quality and computation time is made by choosing the MDP method for the upper bound and the blind policy for the lower bound.

3-2-2 Performance improvements

In general, online POMDP algorithms start from the initial belief b_0 and explore the entire reachable belief space $\mathcal{R}(b_0)$ to converge to the optimal value function V^*. This is of course more efficient than exploring the entire belief space for a solution, such as is done in offline methods when real time performance during the training phase is not necessary. Considering at computational speed is usually a problem with continuous POMDP problems that need to be solved online, this process can be made even more efficient by directing the belief space exploration to the *optimally* reachable space $\mathcal{R}^*(b_0)$ [54]. In this thesis the following methods are proposed to reduce the size of the explored area by pruning branches that are potentially suboptimal during the exploration phase:

Lower Bound Criterion

As the main aim of the POMDP exploration process is to evaluate all branches to find the optimal one, it is possible that there would be several suboptimal branches, such as those that result in collisions with other vehicles for example. In order to quickly prune these branches, a lower bound L can be set on the expected reward, which if not met marks that branch as suboptimal and discontinues exploration along that path. This basically removes branches with low rewards which would ordinarily be explored but not chosen as the optimal branch. It must be noted that this lower bound value must always be chosen somewhat conservatively. This is because during exploration based on sampling of a probability distribution, there could be belief states in which a small percentage of samples result in a collision but could still turn out to be part of the optimal branch in a worst case scenario.

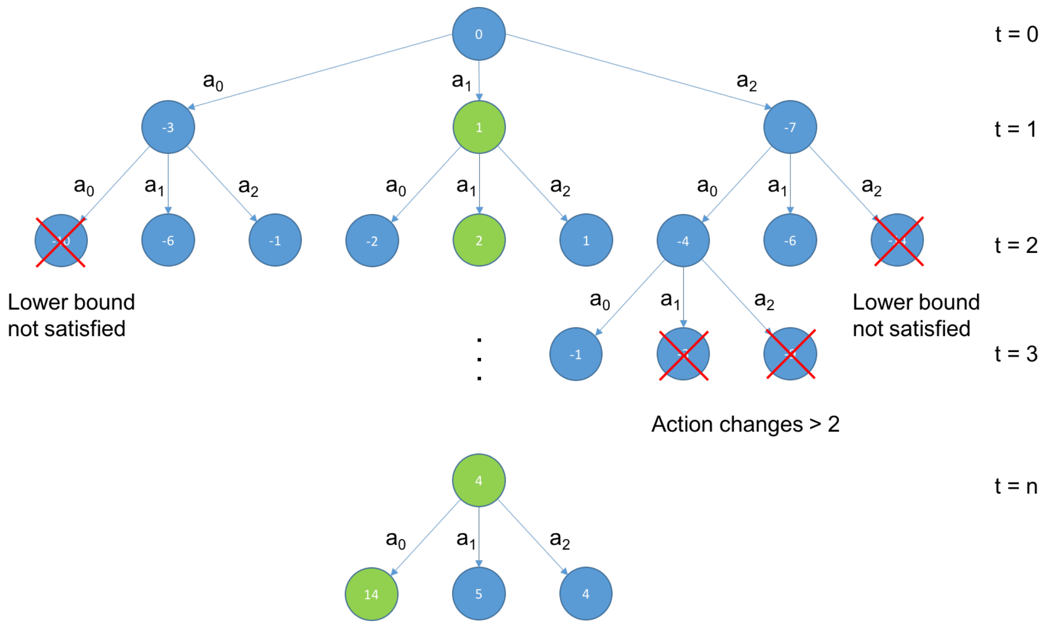


Figure 3-2: Optimally reachable belief space exploration

Gap Termination Criterion

In general, the upper and lower bound give us a measure of the accuracy of the value function approximation but general algorithms wait till a full backup is performed to update the bounds and check if the gap is below a threshold at the root so that it is possible to terminate sampling and further exploration. Instead, passing this gap termination value ϵ down the tree to the leaves by applying the inverse of the discount factor makes this more efficient as it is now possible to check if the termination criterion is satisfied at the leaves rather than only at the root. Thus the value $\gamma^{-t}\epsilon$ is passed down the tree.

Action Change Criterion

Another idea proposed here specifically for the autonomous driving use case is that it is unlikely that a branch that consists of numerous action changes over the horizon will turn out to be optimal, especially from a comfort viewpoint. Thus exploration of these branches can be terminated once the number of action changes exceeds a pre specified limit, thus pruning branches of the tree and making the process far more efficient.

These ideas can be visualized in figure 3-2.

3-3 Formulating the autonomous driving problem as a POMDP

States, actions and observations

For the autonomous driving problem, a continuous state space is chosen and the basic states for the ego vehicle and other traffic objects are position (x, y), velocity (v_x, v_y) and acceleration a_x, a_y. Considering that for the traffic objects these states depend on their uncertain behavior (b), the output of the behavior prediction is added as an extra state for these vehicles.

$$s = \Big(s_{ego}, \quad s_1, \quad ..., \quad s_n\Big) \tag{3-14}$$

$$s_{ego} = \Big(x, \quad y, \quad v_x, \quad v_y, \quad a_x, \quad a_y\Big), s_i = \Big(x, \quad y, \quad v_x, \quad v_y, \quad a_x, \quad a_y, \quad b\Big) \tag{3-15}$$

The initial belief is constructed from the states and observations by creating a normal distribution for the positions, velocities and acceleration. The probability density function (PDF) for a normal (Gaussian) distribution is defined as:

$$p(x|\hat{x}, \Sigma) = \frac{1}{(2\pi)^2\sqrt{det(\Sigma)}} e^{-\frac{1}{2}((x-\hat{x})^T\Sigma^{-1}(x-\hat{x}))} \tag{3-16}$$

where $\hat{x}$ is the mean of the distribution and Σ is the defined covariance matrix:

$$\Sigma = \begin{bmatrix} \sigma_x^2 & \rho_{xy}\sigma_x\sigma_y \\ \rho_{xy}\sigma_x\sigma_y & \sigma_y^2 \end{bmatrix} \tag{3-17}$$

It must be noted that for continuous distributions, the corresponding probability value for x cannot be obtained directly from the PDF. In order to obtain the corresponding probability value of x from the continuous PDF, the obtained value must be multiplied with a small area around x.

For the behavior, each motion intention is assigned to an integer from 0 to 7 and a discrete distribution is created using the probability associated with the behavior (ie., a large random distribution of the values with the corresponding discrete variable's frequency in the distribution is directly proportional to its probability).

For the action set, more care must be taken as the problem grows exponentially with the addition of more actions. Thus the action set is chosen to be discrete, and contains three distinct actions:

$$a = \{Accelerate, Stay\ Constant, Brake\} \tag{3-18}$$

The transitions are also part of the continuous space and the new positions are obtained using the multi model approach discussed in the previous chapter. The transition model simulates the system dynamics and the conditional probability function $p(s'|s, a)$ describes the probability of an initial state s and an action a resulting in the future state s'.

The observations are essentially measurements of position and velocity received from the vehicle sensors, along with the accelerations for the other traffic vehicles measured using the rate of change of velocity between the previous sample and the current one. Thus this is also a continuous space, as it basically follows the state space.

$$o = \Big(o_{ego},\ \ o_1,\ \ ...,\ \ o_n\Big),\quad where\quad o_i = \Big(x,\ \ y,\ \ v_x,\ \ v_y,\ \ a_x,\ \ a_y\Big) \tag{3-19}$$

The observation model simulates the sensors obtaining measurements and the conditional probability function $p(o|s', a)$ describes the probability of receiving an observation o when the action a is applied to get state s'.

3-3-1 Transition model

The objective of the transition model is to effectively describe the system dynamics relative to the chosen actions and consider the uncertainty in these transitions as the conditional probability distribution $p(s'|s, a)$.

In the case of the ego vehicle, knowledge of the chosen action allows the effect it has on the state transition to be calculated. In the case of the traffic objects, their behavior must be considered in order to obtain realistic predictions as this directly has an effect on their transition. For example, the interaction of two vehicles where one follows the other can only be detected when the "Follow Vehicle" behavior is predicted. Further, the digital map data must also be included in this process so that the predictions are more accurate and uncertainty is reduced.

In summary, the transition model for a tactical decision making framework in urban scenarios needs to consider all the static and dynamic information in the scene. The static information includes the road information, road signs and markings, while the dynamic information includes the vehicle's measurable states and predicted behavior. This is where the behavior prediction and multi model transition approaches discussed in the previous chapter come into play.

3-3-2 Observation model

The role of the observation model is to describe the capabilities and limitations of the perception unit and its sensors. The model is basically required to consider the uncertainty related to the sensors, which exist in two areas - inaccuracies and occlusions.

The first problem is fairly straightforward to deal with. The measurements of different traffic objects are independent from each other and a simple additive normal distribution of noise can model sensor inaccuracies quite well. The covariance matrix Σ_O accounts for velocity and position for each traffic object. To model this, the conditional distribution is used as follows:

$$p(o_i|x'_{ego}, x'_1, ...x'_n) = N(x'_{ego}, \Sigma_{O_{ego}}) \tag{3-20}$$

The second problem is slightly more complex. In urban scenarios, it is very common to have vision based occlusions in the path of the sensors and thus vehicles that are present could be completely invisible to the decision making framework. It could also be possible that sensor limitations such as angle or range could cause these traffic objects to be invisible. Since these

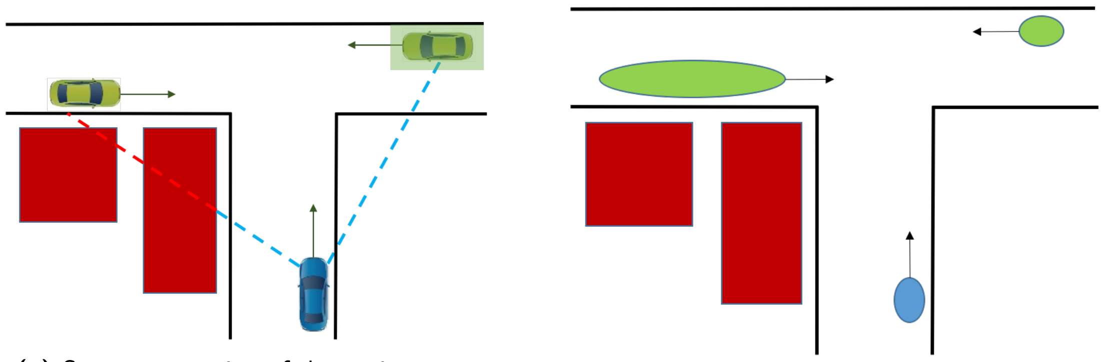

(a) Sensor perception of the environment **(b)** Resulting belief state

Figure 3-3: Observation model in urban areas

problems are related directly to the sensors, they must be considered in the observation model as it allows the decision making framework to be aware of this uncertainty.

An obstruction (such as a building or trees) can be detected by the perception unit/sensors. Then, using the map information it can be checked if this could cause a traffic object to be occluded. If such an obstruction exists, then a new traffic object state is created and added, with the position set behind the obstruction and velocity set to the speed limit of the area. Finally, the Σ_O value is modified to account for a large amount of uncertainty for this specific state. Figure 3-3 gives a rough visualization of the resulting belief state.

In order to augment the modeling of invisible traffic objects in the observation model, a slight modification to the future state prediction is made, wherein the behavior prediction is ignored for the added state. Instead the acceleration, velocity and position is calculated assuming that this vehicle could reach the point of collision (for example, the center of the intersection) at the same moment as the ego vehicle.

3-3-3 Multi Resolution Planning

Measurements taken in the current time step are relatively more certain than future steps where another road user could deviate significantly from his initial intention and predicted state. Thus it is unreasonable to treat prediction steps further into the future with the same resolution as the steps closer to $t = 0$ by using fixed time steps $\Delta t = T/N$ where N is the total number of time steps and T is the horizon.

Thus, the idea of using exponentially increasing time steps is proposed here, wherein the first time step is chosen to be $\Delta t = 0.2$ seconds and every n^{th} time step thereafter is $\Delta t_n = 2^n \Delta t$. This also reduces computational expense while still allowing for a relatively long time horizon.

3-3-4 Reward function

The reward function assigns values to specific situations so that the decision making framework can choose which action would be the most preferred or optimal. The main goals are related to the ego vehicle reaching a preset destination as quickly as possible, while adhering to the speed limit and not crashing into another vehicle. Secondary goals include a comfort factor, which is split into two parts - the direct switching of acceleration and braking actions (rather than using a constant velocity action between them, which would be more comfortable) and continuing to use the same action as the previous time step. Of course, in many cases these goals clash and thus a hierarchy is born, wherein one goal is weighed more strongly than the other. A good policy creates a balance between these potential conflicting goals.

The reward function used in this work is split into two parts - depending on whether a collision has been predicted or not. If there is a collision, only the corresponding collision reward is assigned. If there is no collision then if conditions are used to check whether the ego vehicle has reached its destination, if it is under the set speed limit and if the action chosen obeys the two comfort criteria. An overview of the functions is shown in algorithm 2.

Algorithm 2 Reward function for autonomous driving

```
Precondition: $previous\_action, speed\_limit$
function REWARD FUNCTION($state, action$)
    if $collision$ then        ▷ Checks for collisions between ego vehicle and all traffic objects
        $reward+ = r_{collision} * number\_of\_collisions$
    else
        if $goal$ then        ▷ Checks if the ego vehicle is within the goal area
            $reward+ = r_{goal}$
        end if
        if $speed < speed\_limit$ then        ▷ Checks if the ego vehicle is under the speed limit
            $reward+ = r_{speed} * \frac{speed}{speed\_limit}$
        end if
        if $abs(action - previous\_action) < 2$ then        ▷ Checks if there is a direct switch
                                                              between acceleration and braking
                                                              actions (0 and 2)
            $reward+ = r_{comfort1}$
        end if
        if $action = previous\_action$ then        ▷ Checks if the current action is the same as
                                                     the previous step
            $reward+ = r_{comfort2}$
        end if
    end if
    return $reward$
end function
```

Terminal state

In the event that the destination is reached or a collision occurs with high certainty within the prediction horizon, the prediction along that branch must be stopped as it is not required to explore further and undesirable to potentially assign multiple rewards for the these criteria.

Hence if more than 50% of samples from the belief end up in either of these two criteria, then a terminal flag is set so that the branch is not explored further.

Collision detection

An important tool to detect a terminal state is the collision detection method. Choice of method here is critical, as this must be performed n times, for each sample of each belief of the POMDP. There is also the issue of vehicles tunneling each other [55] between time steps due to relatively high velocities as illustrated in figure 3-4.

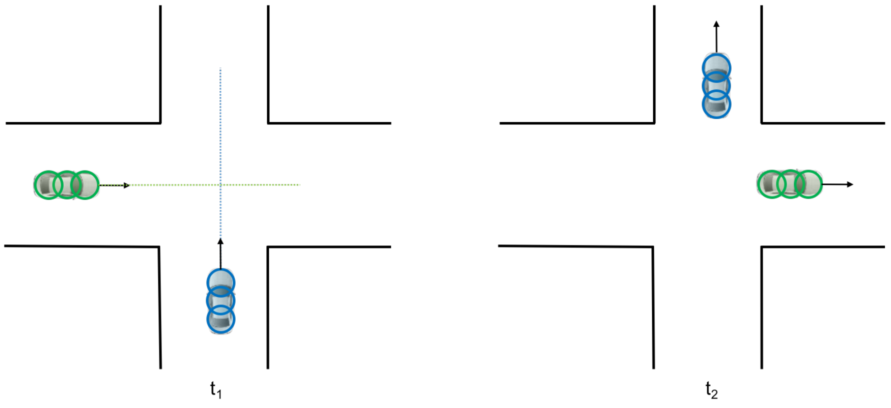


Figure 3-4: Vehicles tunneling each other between two time steps

A simple method to check for collisions is to perform a distance check between two vehicles, which is analogous to drawing a circle over each vehicle and checking for intersections between pairs of these circles. The problem here is that this is quite inaccurate in the lateral direction and would result in numerous false collision predictions when vehicles are traveling in adjacent lanes. According to [56], a minimum of three circles along the vehicle axis can represent the vehicle with minimal error using the vehicle's length l, width b and wheelbase w. The equations to calculate the positions c_i and radius r of these circles using the half axle distance d_{ha} as follows:

$$d_{ha} = 0.5(l - w) \tag{3-21}$$

$$c_1 = 0.5 * d_{ha} \tag{3-22}$$

$$c_3 = l - (1.5 * d_{ha}) \tag{3-23}$$

$$c_2 = 0.5 * (c_1 - c_3) \tag{3-24}$$

$$r = \sqrt{0.25 * b^2 + 0.25 * d_{ha}^2} \tag{3-25}$$

While the three circle representation is more accurate, it is also much more expensive than the single circle check, as this results in 3^n distance comparisons when considering n vehicles, compared to n distance comparisons with the single circle representation.

Thus a two level collision check is proposed, wherein the first level consists of the single circle representation and the second level consists of the three circle representation. If a collision is detected by the first level, the second level check performed to verify if a collision is actually going to occur. This balances accuracy and computational expense between the two methods.

In order to solve the issue of tunneling with this two level check, two modifications are made. First, to better detect the possibility of vehicles being close enough that a collision is possible at a specific speed, the diameter of the first circle is made directly proportional to the speed of the vehicles. At the second level, it is clearly not enough to simply perform the collision check with the three circle representation at the position at that discrete time step. Instead, a "continuous" check is performed by moving the vehicles along their intended direction in smaller steps between the two larger POMDP prediction time steps to check for a collision. This can be visualized better in figure 3-5 using the previous example shown in figure 3-4.

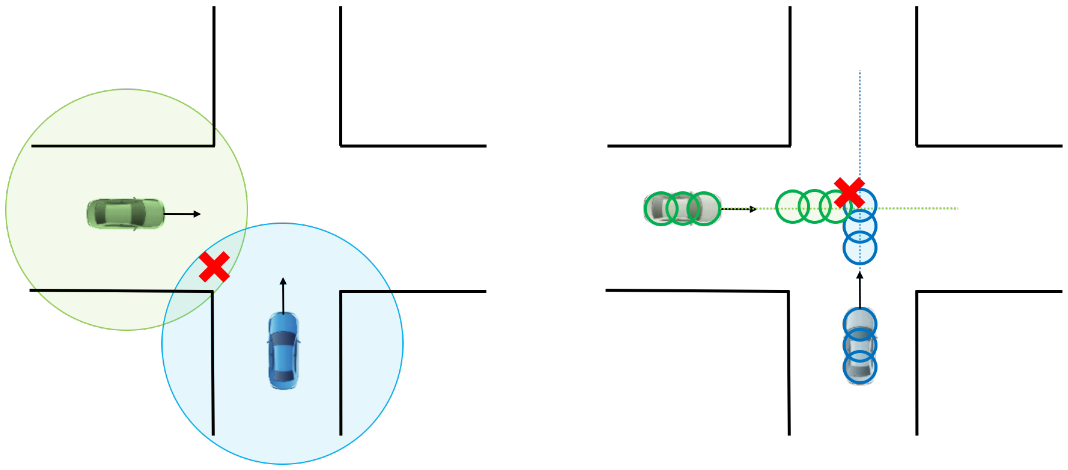


Figure 3-5: Two level continuous collision detection

While this takes care of the challenges discussed so far, there is a possibility of collisions still occurring if a traffic object performs a sudden action such as braking hard while ahead of the ego vehicle. This is due to the limited deceleration value in the discrete action set, and can result in a collision if the traffic object decelerates continuously with a higher value than the ego vehicle's braking action. To work around this, if a collision terminal state is reached for all possible root action choices, the braking value is modified to a larger "brake harder" value. This results in a negligible computational overhead and accounts for such emergency situations where collisions would be unavoidable with a limited action set.

3-4 Summary

A state of the art online POMDP solver was chosen and modified to converge to a solution faster by directing the solver towards the belief space that will most likely result in the optimal solution. Additionally, the approximation of the value function is stopped as soon as it has achieved a specific level of accuracy, thus making the entire solver far more efficient.

Following this, more specific problems relating to the autonomous driving use case were discussed, such as vehicle tunneling and invisible vehicles in urban areas. Solutions to these problems were proposed and incorporated withing the POMDP structure.

Chapter 4

Simulation results

This chapter begin with a description of the simulation setup, the parameters chosen and some evaluation criteria. Following this, the test scenarios are described and are accompanied by the corresponding results along with an analysis of the various graphs.

4-1 Simulation Setup

In order to test and verify the work done, the simulation software CarMaker by IPG is chosen. CarMaker is a sophisticated vehicle simulation package that allows the user to model a vehicle accurately using the multitude of parameters spanning the suspension, engine and transmission characteristics. The user can then create the desired road network, insert Traffic objects and import a 3-dimensional vehicle model into the software for visualization.

A road network is created using a text file in which all necessary data relating to the scene has to be defined. This includes the length, radius, join angle, etc of the road segments and junctions. Further details such as road markings, signs and traffic lights can also be defined. Finally, routes must be created along these road segments, vehicles are assigned to a route and maneuvers can be set for the traffic objects. For the ego vehicle the user is given the option of using a fixed maneuver or external input from Simulink or a C++ program. The digital map information corresponding to these road networks and routes is written in a C++ file and read by the behavior predictor and future state estimator when needed. This file includes essential information regarding the scene, such start and end points of a junction, the existence and position of road markers such as stop lines and the radius of turns.

The decision making framework is set to run every 0.2 seconds and corresponding synchronization with the simulation software is set up. In the case that the built executable of the C++ source code runs slower than 0.2 seconds per planning iteration, the simulation software waits to receive an input before proceeding. The output of the simulation can then be viewed in the IPGMovie window, and specific output parameters can be viewed as the simulation progresses in the IPGControl window.

4-1-1 Parameter Configuration

Before proceeding further, the various parameters of the decision making framework must be described for a better understanding of the results. These are listed in table 4-1 and divided into categories for ease of viewing.

Category	Parameter	Value
General	Time step (Δt)	0.2 s
	Total time horizon (T)	6.2 s
	Speed limit	50 km/h ($\pm$5%)
	Discount factor (γ)	0.9
Actions	Accelerate	+2 m/s^2
	Constant	0 m/s^2
	Brake	-2 m/s^2
Sensor	RADAR - Range	150 m
	RADAR - Field	140deg
Belief Prediction	Number of samples (N)	50
	Transition model variance (Σ_T)	0.05
	Observation model variance (Σ_O)	0.05 (visible) or 0.5 (invisible)
Rewards	Collision	-1000*number of collisions
	Goal reached	100
	Speed	20*(current speed/speed limit)
	Comfort1	10
	Comfort2	10
Termination Criteria	Minimum bound gap	20
	Lower bound	-250
	Maximum number of action changes	2
Collision Check	Circle diameter (level 1)	40 m

Table 4-1: Framework parameters

4-1-2 Evaluation Criteria

The first and foremost goal of a decision making method for autonomous vehicles is to ensure the safety of its passengers. Along with this, it must also result in a high level of efficiency and comfort for the passengers inside the vehicle. Unfortunately there are no standard criteria or tests defined by governing bodies for autonomous vehicle or in the related literature yet, so the following criteria are defined to quantify the performance of the decision making approach:

- Minimum distance to vehicles: This parameter measures the closest distance (from bumper to bumper) between the ego vehicle and any other vehicle in the scenario. This will be accompanied by the ego vehicle's speed at that moment as the safety distance is directly proportional to the speed of the vehicles. A larger distance between vehicles is preferred as it represents a higher safety margin.
- Time to accelerate: This refers to the time the ego vehicle spends below the maximum speed without accelerating after a braking maneuver has been performed and there is

no obstacle in front of the ego vehicle anymore (or if the speed of the vehicle ahead is significantly higher than that of the ego vehicle). The lower the value, the better is the efficiency of the approach.

- Number of action changes: This is the number of times the ego vehicle changes its chosen action over the entire simulation period. In order to have a smooth and comfortable ride, this value must be as low as possible.

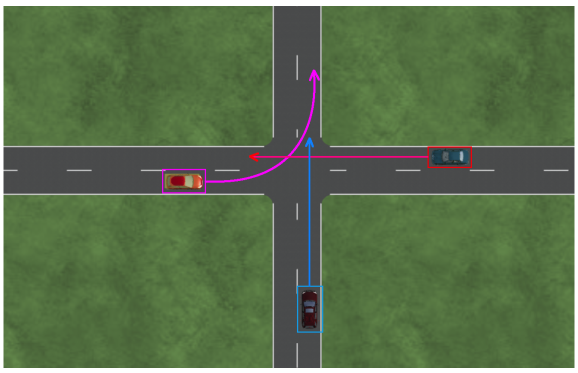
4-2 Results

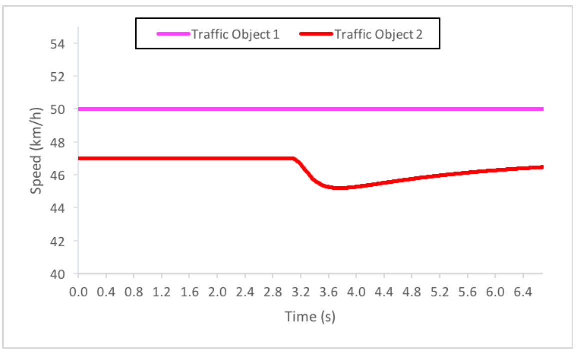
The goal in this work is to test the decision making framework within various urban scenes similar to those shown in figure 1-2 in Chapter 1. While the total set of tested scenes and scenarios cannot be displayed here, a chosen set of the results are displayed to give the reader a relatively wide range of possible scenarios. In each subsection the scenario will be described, along with figures and the velocity profile of each traffic object. In these figures, the ego vehicle is always marked with a blue box, while the traffic objects are marked with red, magenta and orange boxes. The first result presented is the behavior prediction for each traffic vehicle, followed by the decision making results (ie., the chosen action along with the associated velocity profile of the ego vehicle and the reward values rescaled relative to each other).

It is of value to note that in order to create the worst possible scenarios to test and validate the work done, usual traffic rules are broken by the traffic objects in some of the scenarios. The aim here is to prove that the framework is able to achieve an adequate level of safety and performance even when there is a large amount of uncertainty and unpredictability.

4-2-1 Perpendicular Intersection

Topographically, this is the simplest of the tested scenes but it allows for inferences to be made more easily before progressing to more complicated scenes. This scene is a perfect intersection with full visibility, and the results from two scenarios are presented.


(a) Intended routes of all vehicles

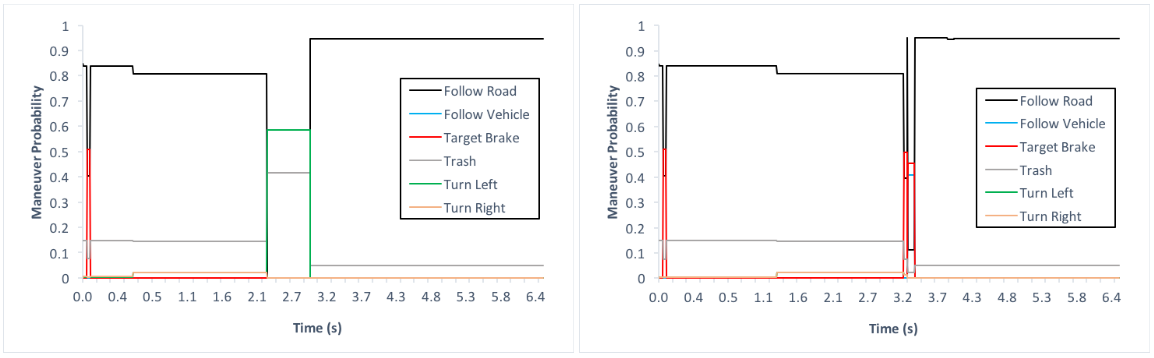


(b) Velocity profile of traffic

Figure 4-1: Test scenario 1: Setup

The first scenario uses two traffic objects that start from opposite directions. The first one turns left at the intersection, while the second continues straight ahead. The ego vehicle is set to continue straight along its path to cross the intersection. Figure 4-1 depicts the scenario visually and provides the velocity profiles of the traffic objects.

The corresponding behavior predictions of the traffic objects are seen in figure 4-2. Looking at the prediction for traffic object 1, a follow road maneuver is predicted initially and then as it nears the turning point the Turn Left maneuver is predicted as soon as it begins turning as the vehicle does not brake before making the turn. For traffic object 2, the prediction is that it continues to follow the road and this is of course accurate. At the point when traffic object 2 is performing a turn and is an obstacle in front, the vehicle brakes slightly to avoid a potential collision and thus a relatively high possibility of a target brake maneuver is predicted.

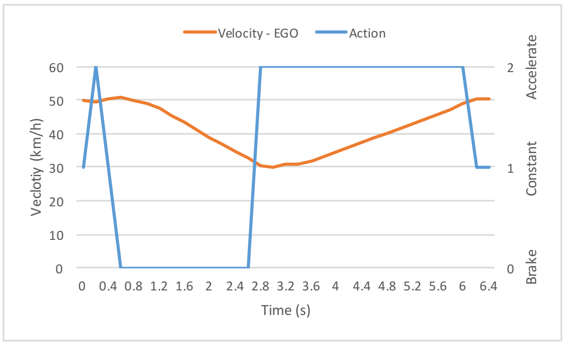


(a) Traffic object 1 **(b)** Traffic object 2

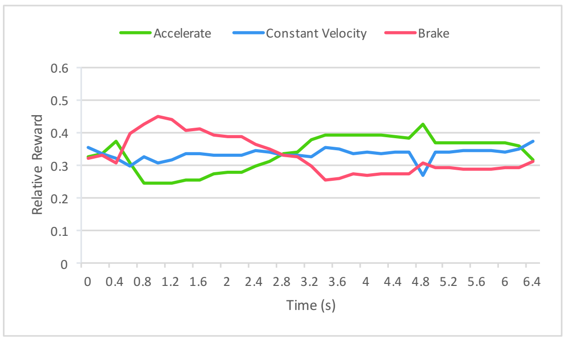
Figure 4-2: Test scenario 1: Results - Behavior prediction

Focusing on decision making, the ego vehicle has two basic option to choose from in this scenario. Either it can accelerate and cross the intersection before either of the other vehicle arrive or it must brake to allow them to pass and then continue. Since this requires a gross violation of the speed limit (and thus a relatively lower reward), the ego vehicle's only viable choice here is to brake. This must be kept in mind for all the remaining tested scenarios.

The output (chosen action) of the decision making framework along with the corresponding velocity of the ego vehicle over the course of the simulation is seen in figure 4-3. At the first instant the ego vehicle chooses to accelerate in an effort to maintain the highest speed as per the set speed limit. A moment later it is more certain that a collision is imminent and the decision switches to a braking maneuver. After this, the ego vehicle chooses to accelerate back to the speed limit at the 2.8 second mark, even though traffic object 1 has not fully crossed its path. This is because it is able to predict with enough certainty that a collision is not possible with any of the vehicles. This is unlike the average human driver where one feels more comfortable accelerating only once a clear path ahead is visible. On one hand this result is more efficient, but on the other hand more human like behavior may be preferred by the occupants of the vehicle.



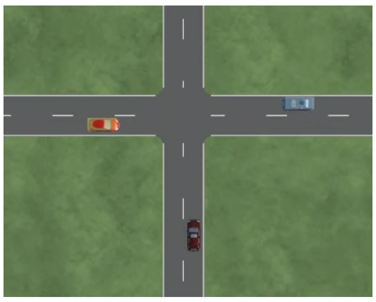
(a) Chosen action and Ego vehicle velocity



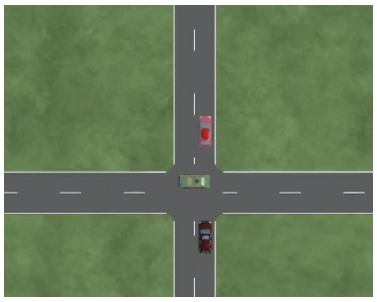
(b) Relative rewards for each action

Figure 4-3: Test scenario 1: Results - Decision Making

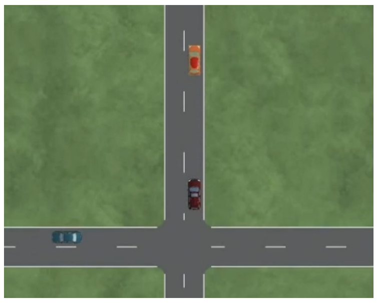
A visualization of the simulation output is shown in figure 4-4.



(a) t = 2.5s



(b) t = 3.6s



(c) t = 5.1s

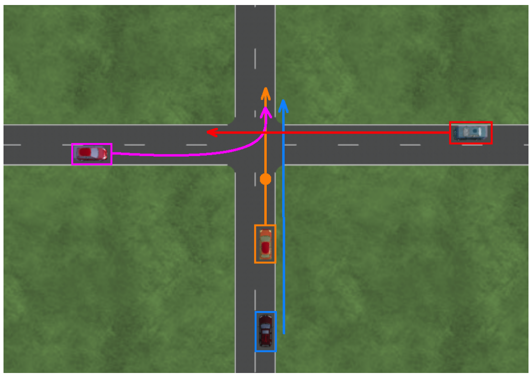
Figure 4-4: Test scenario 1: Visualization

The overall performance of the decision making approach is quantified in in table 4-2:

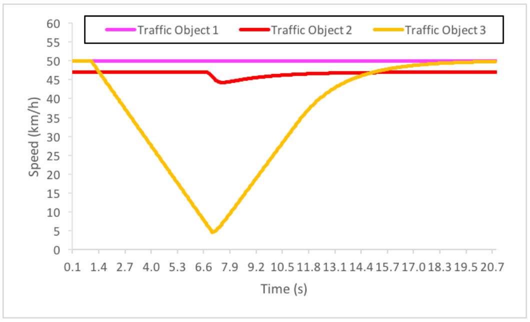
Criterion	**Value**
Minimum distance to vehicles	3.8m (31.0km/h)
Time to accelerate	0s
Number of action changes	4

Table 4-2: Performance evaluation - Decision Making

The second scenario is set up with one more traffic object traveling straight ahead, located in front of the ego vehicle. This vehicle slows down to a stop before the intersection to give way to the other vehicles before continuing ahead. Figure 4-5 depicts the scenario along with the velocity profile of the traffic objects.



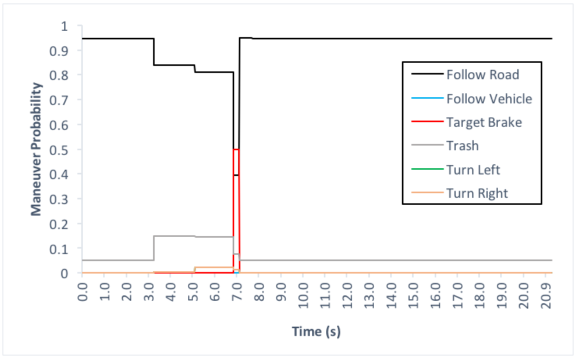
(a) Intended routes of all vehicles



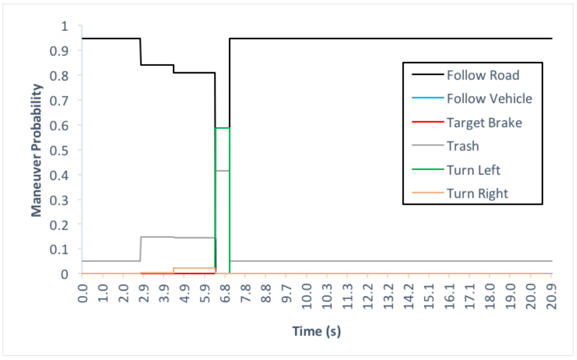
(b) Velocity profile of traffic

Figure 4-5: Test scenario 2: Setup

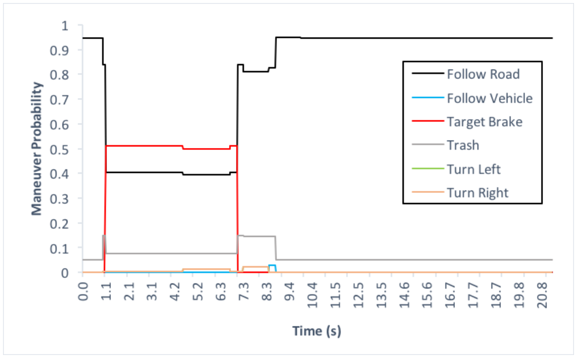
The corresponding behavior predictions of the traffic objects are seen in figure 4-6. The predictions of traffic object 1 and 2 are similar to that of scenario 1. With regard to traffic object 3, which begins braking (with the stop line as a target stop point) to give way to the other vehicles before accelerating and continuing ahead, a Target Brake maneuver is accurately predicted after which the most likely maneuver is again the Follow Road maneuver.



(a) Traffic object 1



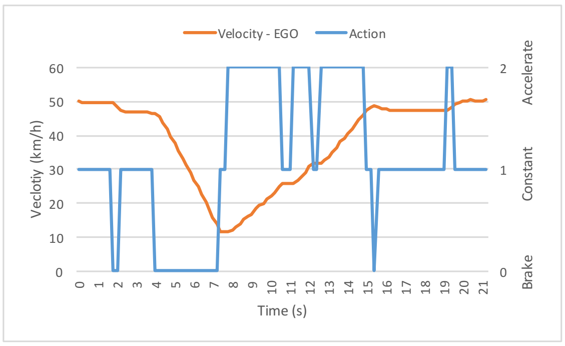
(b) Traffic object 2



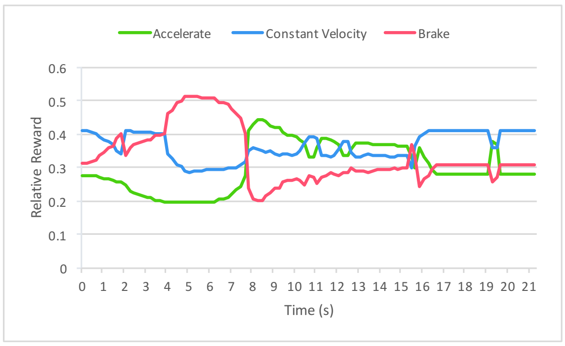
(c) Traffic object 3

Figure 4-6: Test scenario 2: Results - Behavior prediction

The output (chosen action) of the decision making framework along with the corresponding velocity of the ego vehicle over the course of the simulation is seen in figure 4-7. At the start, the ego vehicle keeps a constant velocity as it is following another vehicle (traffic object 3). Once this vehicle begins to brake at the intersection, the ego vehicle predicts the possibility of an earlier collision but due to the uncertainty, it only chooses to brake slightly as this is the most efficient choice. Once the certainty of a collision increases, the ego vehicle begins a more consistent braking maneuver. Finally, while accelerating back up to speed, the ego vehicle switches between accelerate and stay constant a few times as it attempts to keep a sufficient gap between itself and traffic object 3. Overall, the ego vehicle is able to maintain not only a safety margin but also a high level of efficiency in its decisions.



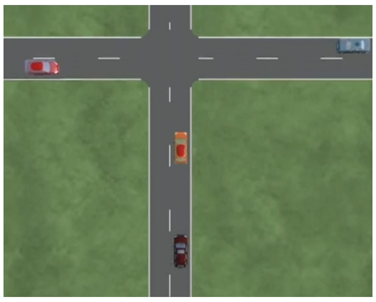
(a) Chosen action and Ego vehicle velocity



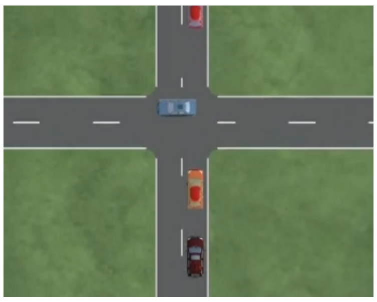
(b) Relative rewards for each action

Figure 4-7: Test scenario 2: Results - Decision Making

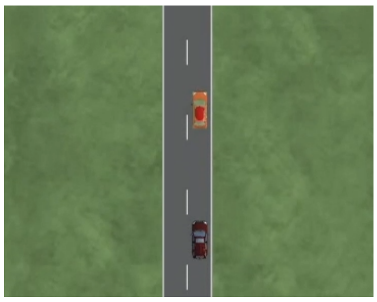
A visualization of the simulation output is shown in figure 4-8.



(a) t = 5.7s



(b) t = 7.6s



(c) t = 16.0s

Figure 4-8: Test scenario 2: Visualization

The overall performance of the decision making approach is quantified in in table 4-3:

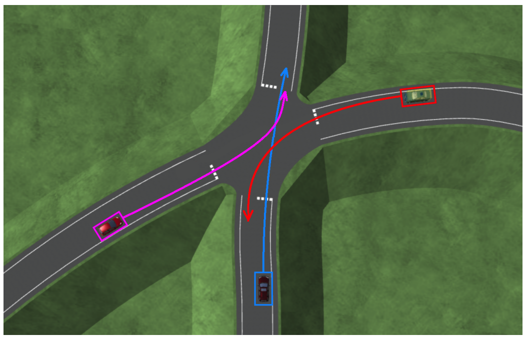
Criterion	Value
Minimum distance to vehicles	3.02m (10.9km/h)
Time to accelerate	0.2s
Number of action changes	13

Table 4-3: Performance evaluation - Decision Making

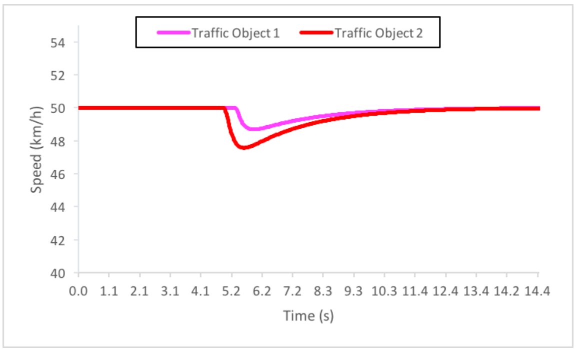
4-2-2 Curved Intersection

Here the topography is complicated by adding curves to the road segments, and making them meet at the intersection at varying angles. This is to further test and validate the decision making framework, since the amount of uncertainty makes it very difficult even for a human driver to predict if a vehicle approaching from the left will continue straight or turn left at the intersection. It is also critical here that the future state estimation takes into account the map information, due to the curvature and angles of the road segments.

The first scenario within this scene uses two traffic objects that start from opposite directions. In this case, both vehicles turn left at the intersection. The ego vehicle is set to continue straight along its path to cross the intersection. Figure 4-9 depicts the scenario visually and provides the velocity profiles of the traffic objects.

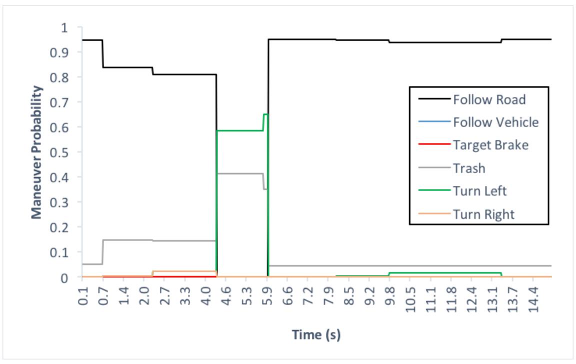


(a) Intended routes of all vehicles

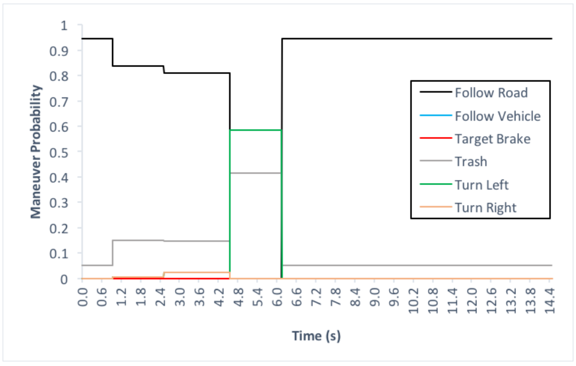


(b) Velocity profile of traffic

Figure 4-9: Test scenario 3: Setup



(a) Traffic object 1

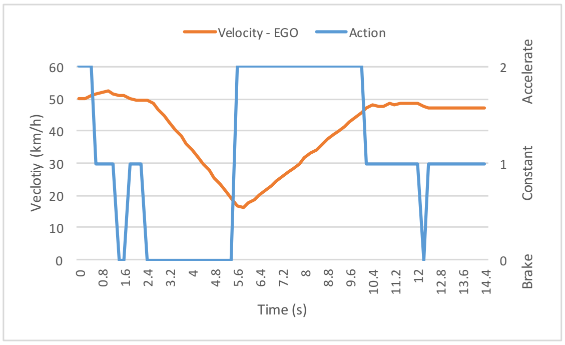


(b) Traffic object 2

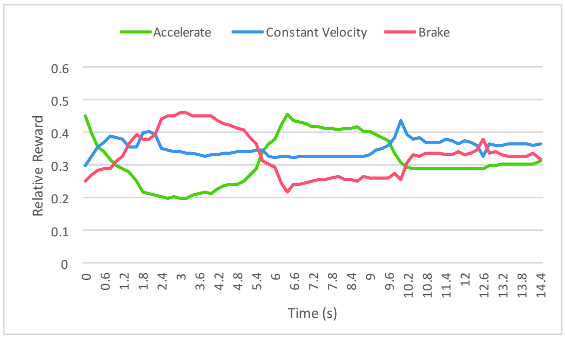
Figure 4-10: Test scenario 3: Results - Behavior prediction

The corresponding behavior predictions of the traffic objects are seen in figure 4-10. The turn left maneuvers were successfully predicted for both traffic objects. The decrease in speed along with the yaw angle and the map information easily lead to this prediction. The relatively high trash maneuver prediction is due to the uncertain nature of the scene and the vehicles brake relatively late into their respective maneuvers compared to human drivers. This is a simulation artifact, as the Bayesian network was trained on real data.

The output (chosen action) of the decision making framework along with the corresponding velocity of the ego vehicle over the course of the simulation is seen in figure 4-11. The ego vehicle is able to predict a collision based on the movements of the vehicles within the scene and performs the necessary braking maneuver to give way to the traffic objects before it accelerates ahead and approaches its destination. After passing the intersection, the ego vehicle chooses to stay below the maximum speed limit due to the fact that it needs to keep a sufficient gap to traffic object 2.



(a) Chosen action and Ego vehicle velocity



(b) Relative rewards for each action

Figure 4-11: Test scenario 3: Results - Decision Making

A visualization of the simulation output is shown in figure 4-12.

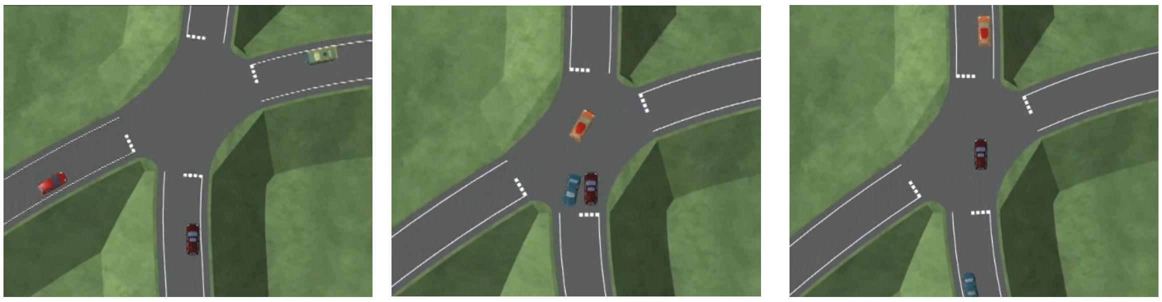

(a) t = 3.8s (b) t = 5.6s (c) t = 6.7s

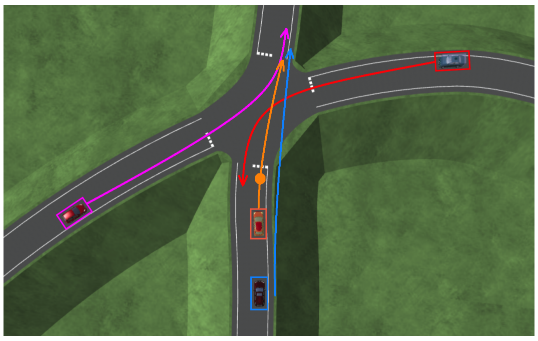
Figure 4-12: Test scenario 3: Visualization

The overall performance of the decision making approach is quantified in in table 4-4. Note that the low distance between vehicles is due to the ego vehicle and traffic object 1 passing each other in adjacent lanes.

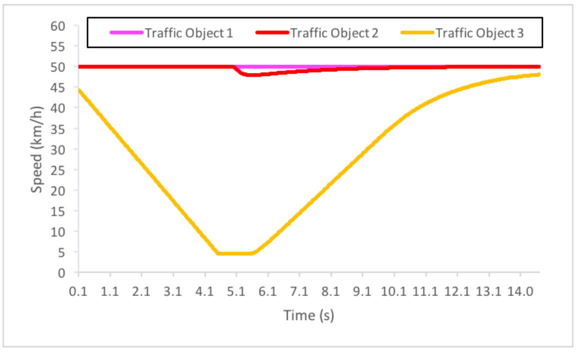
Criterion	Value
Minimum distance to vehicles	0.9m (17.7km/h)
Time to accelerate	0.4s
Number of action changes	8

Table 4-4: Performance evaluation - Decision Making

The next scenario is set up with one more traffic object placed in front of the ego vehicle. It slows down to a stop before the intersection to give way to the other vehicles before continuing ahead. Figure 4-13 depicts the scenario along with the velocity profile of the traffic objects.

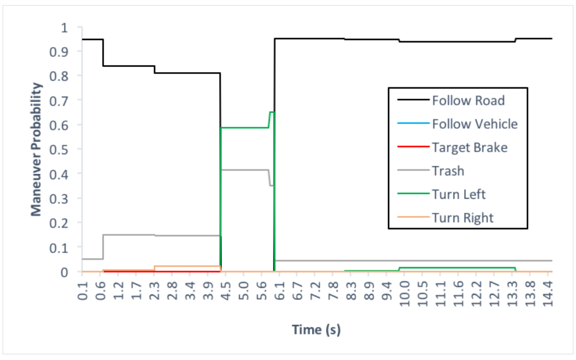


(a) Intended routes of all vehicles

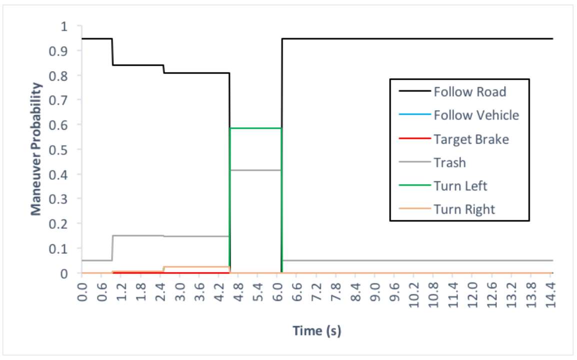


(b) Velocity profile of traffic

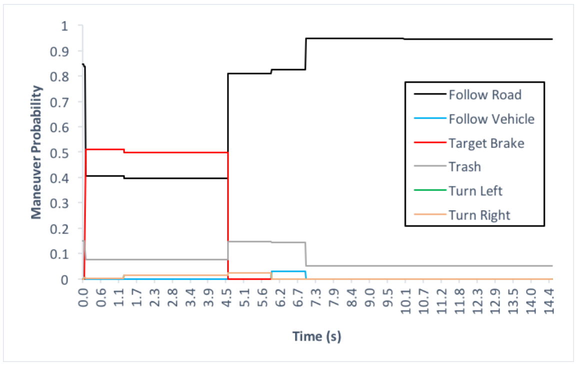
Figure 4-13: Test scenario 4: Setup



(a) Traffic object 1



(b) Traffic object 2



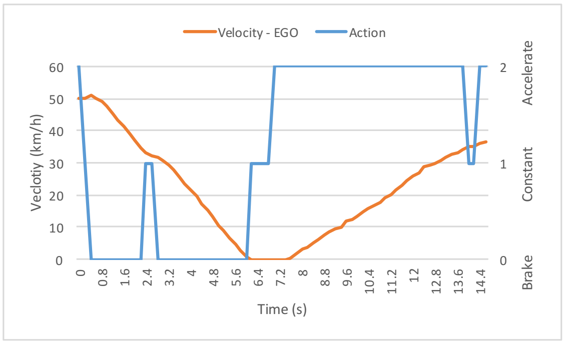
(c) Traffic object 3

Figure 4-14: Test scenario 4: Results - Behavior prediction

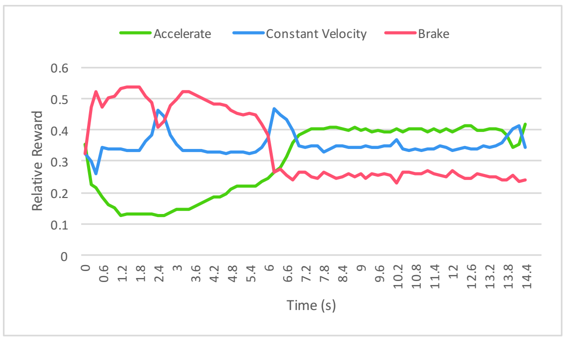
The behavior predictions of the traffic objects are seen in figure 4-14. There is no significant change for the prediction of the first two vehicles compared to the previous scenario. Their turn left maneuvers are predicted using their states and the map information. For the third traffic object, the target brake maneuver is identified immediately as it brakes (with the stop line as a target) to give way to the other vehicles before continuing ahead.

The output (chosen action) of the decision making framework along with the corresponding velocity of the ego vehicle over the course of the simulation is seen in figure 4-15.

Corresponding to the behavior prediction, the ego vehicle performs a braking maneuver and comes to a complete stop behind traffic object 3. At the 6 second mark when traffic object 3 has begun moving again, the ego vehicle also begins to accelerate and follows behind it.



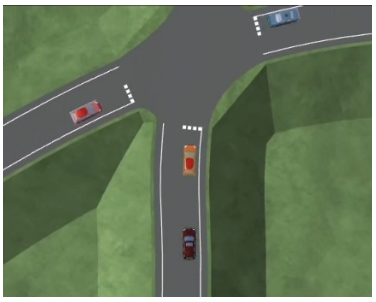
(a) Chosen action and Ego vehicle velocity



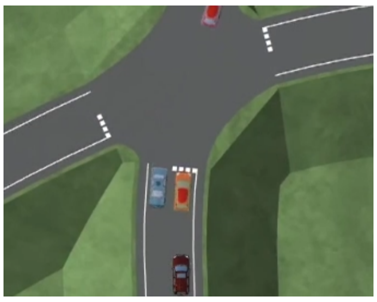
(b) Relative rewards for each action

Figure 4-15: Test scenario 4: Results - Decision Making

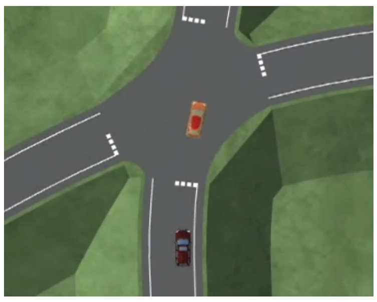
A visualization of the simulation output is shown in figure 4-16.



(a) t = 4.3s



(b) t = 6.0s



(c) t = 8.5s

Figure 4-16: Test scenario 4: Visualization

The overall performance of the decision making approach is quantified in in table 4-5:

Criterion	Value
Minimum distance to vehicles	0.9m (0.2km/h)
Time to accelerate	0.8s
Number of action changes	7

Table 4-5: Performance evaluation - Decision Making

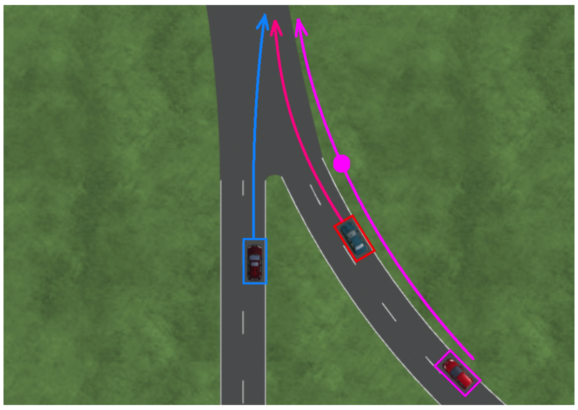
Again, the low distance between vehicles here is due to the ego vehicle and traffic object 1 passing each other in adjacent lanes.

4-2-3 Zipper Merge

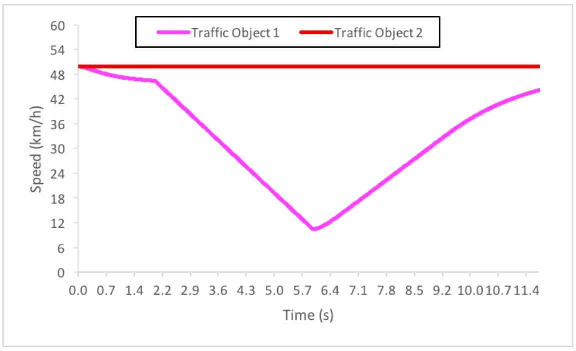
Here a scene which simulates vehicles approaching from different roads/lanes and then merging into a single lane. Due to the unsignalized nature of this scene in particular, there exists a lot of uncertainty in the intention of the traffic as the vehicles approach the merge point.

The first scenario in this scene deals with two traffic objects approaching the merge point from the same lane. The first traffic vehicle does not brake and continues ahead without interruption, while the second traffic vehicle ends up slowing down to allow the ego vehicle to merge ahead of it.

Figure 4-17 depicts the scenario visually and provides the velocity profiles of the traffic objects.



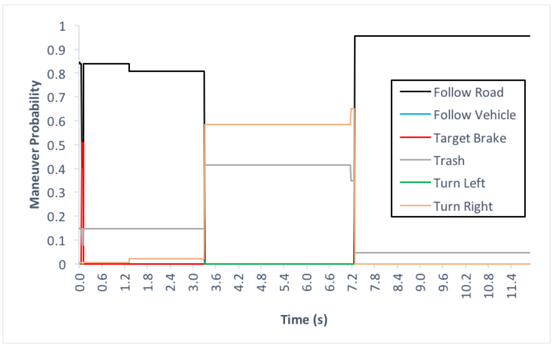
(a) Intended routes of all vehicles



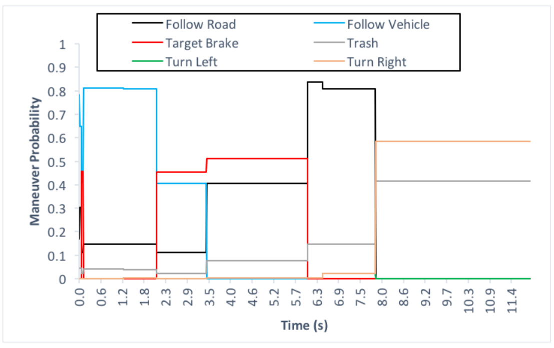
(b) Velocity profile of traffic

Figure 4-17: Test scenario 5: Setup

The corresponding behavior predictions of the traffic objects are seen in figure 4-18. Here the turn right/merge maneuver for traffic object 2 and the target brake maneuver for the traffic object 1 are both predicted accurately as they approach the merge point.



(a) Traffic object 1



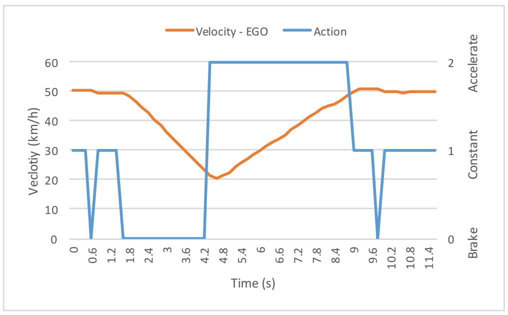
(b) Traffic object 2

Figure 4-18: Test scenario 5: Results - Behavior prediction

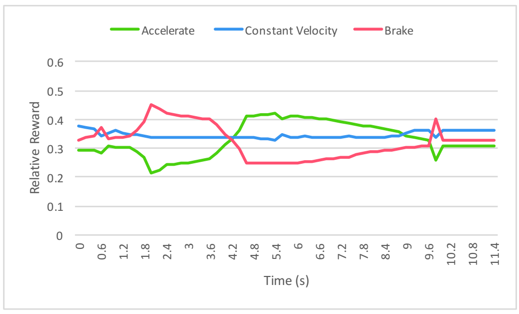
The output (chosen action) of the decision making framework along with the corresponding velocity of the ego vehicle over the course of the simulation is seen in figure 4-19.

This scenario displays the robustness of the overall approach as the ego vehicle is able to consider these predicted intentions and the general uncertainty to make the appropriate braking and acceleration decisions so that it can easily merge between the two traffic objects instead of waiting inefficiently for both vehicles to pass and then moving ahead.

A visualization of the simulation output is shown in figure 4-20.

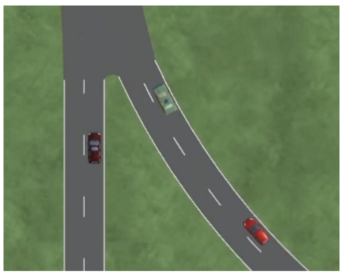


(a) Chosen action and Ego vehicle velocity

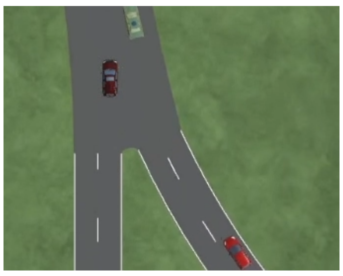


(b) Relative rewards for each action

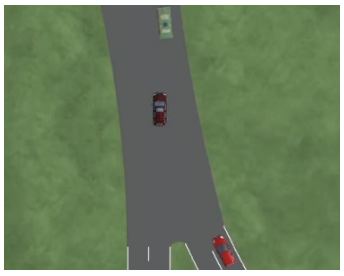
Figure 4-19: Test scenario 5: Results - Decision Making



(a) t = 1.2s



(b) t = 2.6s



(c) t = 4.4s

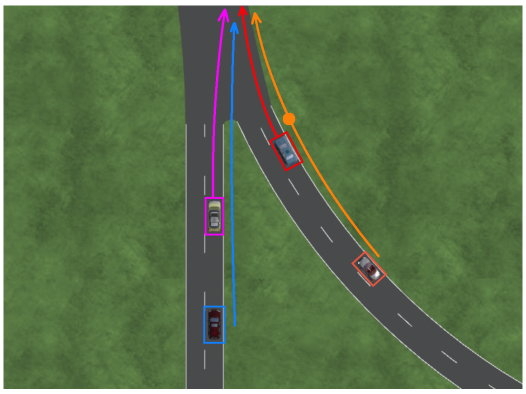
Figure 4-20: Test scenario 5: Visualization

The overall performance of the decision making approach is quantified in in table 4-6:

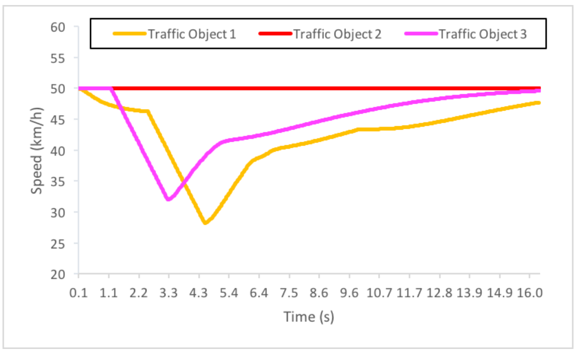
Criterion	Value
Minimum distance to vehicles	3.87m (41.0km/h)
Time to accelerate	0.8s
Number of action changes	7

Table 4-6: Performance evaluation - Decision Making

The next scenario is set up with traffic object 3 added in front of the ego vehicle. It is set to slow down before the merge point to give way to traffic object 2 (approaching from the right), after which it merges into the lane. Finally, traffic object 1 approaches from the right and slows down to avoid colliding with the other vehicles, then accelerates ahead without giving way to the ego vehicle. This complicated merge scenario pushes the overall approach to check that a safe and efficient behavior is achievable for the ego vehicle. Figure 4-21 depicts the scenario along with the velocity profile of the traffic objects.

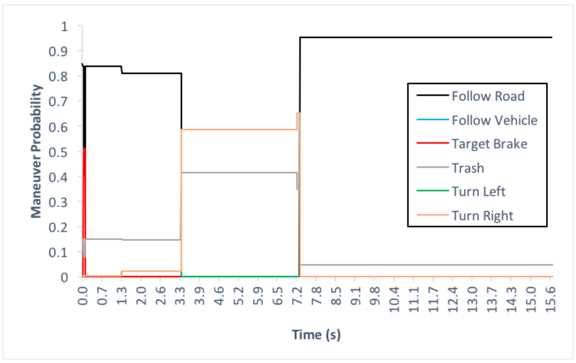


(a) Intended routes of all vehicles

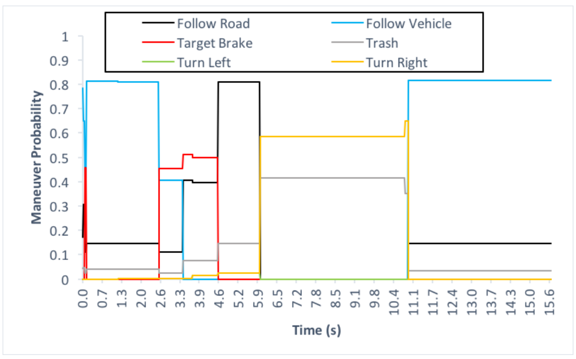


(b) Velocity profile of traffic

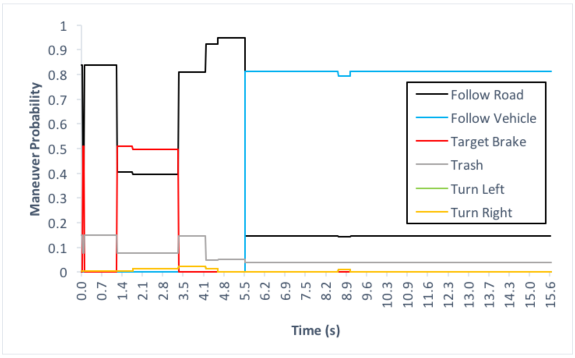
Figure 4-21: Test scenario 6: Setup



(a) Traffic object 1



(b) Traffic object 2



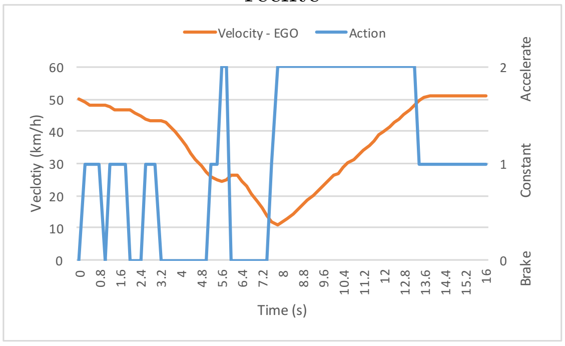
(c) Traffic object 3

Figure 4-22: Test scenario 6: Results - Behavior prediction

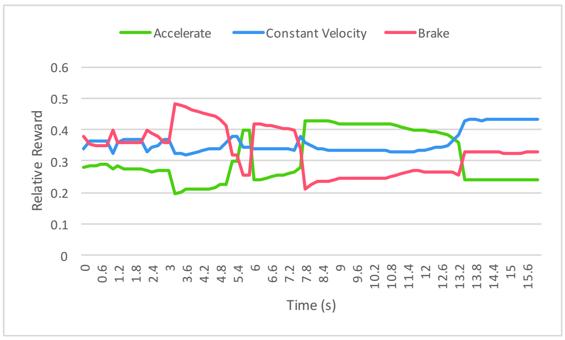
The corresponding behavior predictions of the traffic objects are seen in figure 4-22. The slowing down behavior of traffic objects 1 and 3 is formulated as a target braking prediction. While this is partly true, these vehicles do not intend to come to a complete halt and this is reflected by the fact that the target brake prediction is completely eliminated after the vehicles stop braking, at which point the merge maneuvers are predicted and then the follow vehicle maneuver is predicted for these two vehicles.

The output (chosen action) of the decision making framework along with the corresponding velocity of the ego vehicle over the course of the simulation is seen in figure 4-23. The ego vehicle brakes sufficiently behind the vehicle in front of it, and then starts accelerating with the assumption that traffic object 1 will continue with its target brake maneuver. But once the maneuver prediction changes, it sees the high possibility of a collision as it is no longer given a gap and it chooses to brake again. Finally, it accelerates again to merge behind the entire platoon of vehicles.

rechte



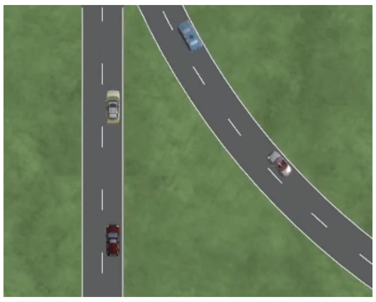
(a) Chosen action and Ego vehicle velocity



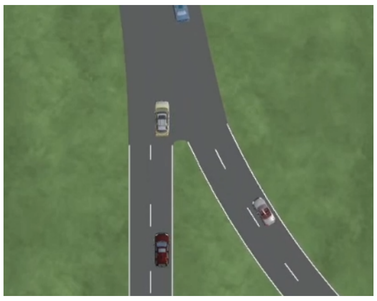
(b) Relative rewards for each action

Figure 4-23: Test scenario 6: Results - Decision Making

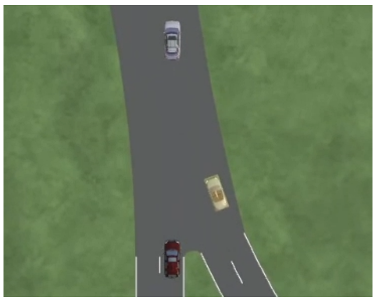
A visualization of the simulation output is shown in figure 4-24.



(a) t = 3.1s



(b) t = 4.9s



(c) t = 7.2s

Figure 4-24: Test scenario 6: Visualization

The overall performance of the decision making approach is quantified in in table 4-7:

Criterion	Value
Minimum distance to vehicles	4.39m (17.7km/h)
Time to accelerate	0.8s
Number of action changes	11

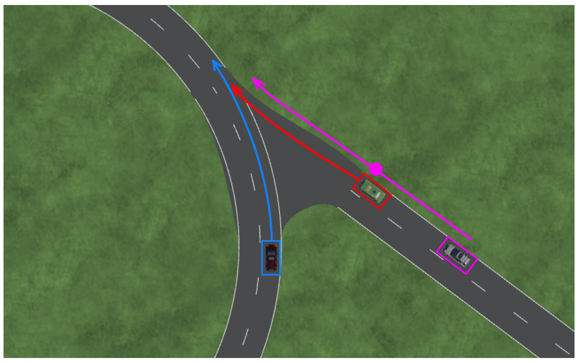
Table 4-7: Performance evaluation - Decision Making

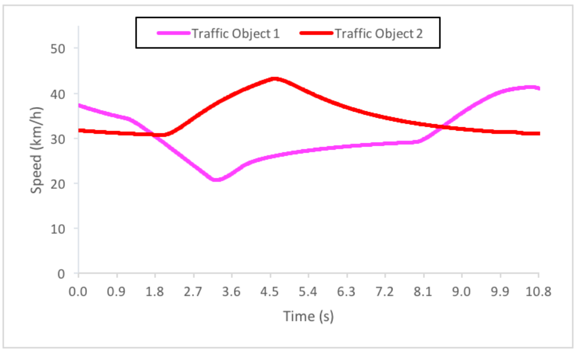
4-2-4 Roundabout

The core of this scene is a roundabout, wherein vehicles merge onto a common curved path. This is also a fairly common scene in urban areas and there is usually a high amount of interaction and cooperation between vehicles here, with traffic rules usually dictating a hierarchy between vehicles. For the sake of testing and validation, two worst case scenarios are created and tested within the roundabout scene.

The first scenario deals with two traffic objects attempting to enter the roundabout from the same lane. The first traffic object does not give way to the ego vehicle, while the second traffic vehicle ends up slowing down before merging to allow the ego vehicle to merge ahead of it.

Figure 4-25 depicts the scenario visually and provides the velocity profiles of the traffic objects.


(a) Intended routes of all vehicles

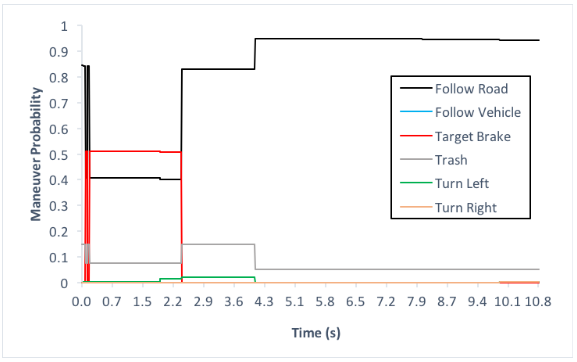


(b) Velocity profile of traffic

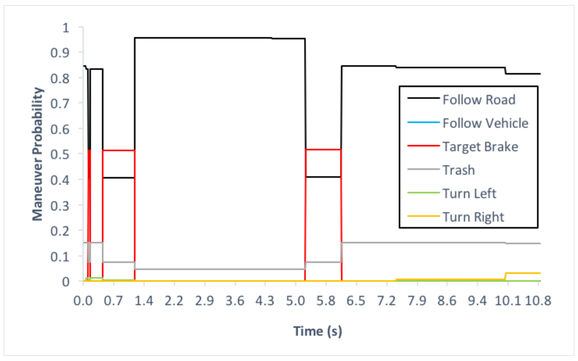
Figure 4-25: Test scenario 7: Setup

The corresponding behavior predictions of the traffic objects are seen in figure 4-26. The traffic vehicles in this scenario slows down as they approach the roundabout, and then accelerate back up to speed once they enter it. This causes the behavior to be predicted as a target brake maneuver because of the possibility that the vehicle can stop before entering the roundabout. This is also seen in the results, along with a slightly lower probability that the maneuver is follow road or follow vehicle.

The output (chosen action) of the decision making framework along with the corresponding velocity of the ego vehicle over the course of the simulation is seen in figure 4-27.

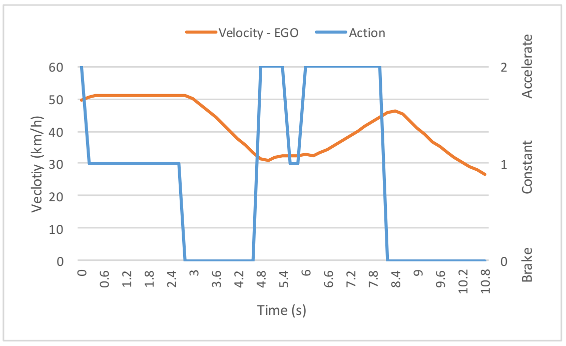


(a) Traffic object 1

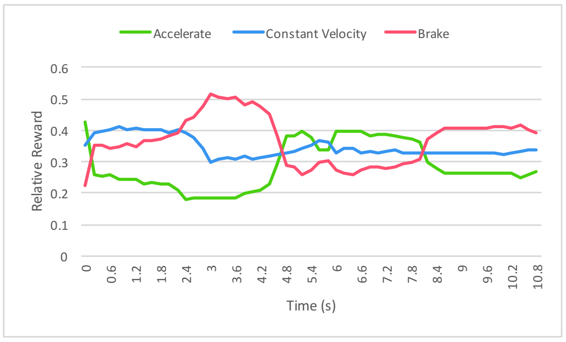


(b) Traffic object 2

Figure 4-26: Test scenario 7: Results - Behavior prediction



(a) Chosen action and Ego vehicle velocity

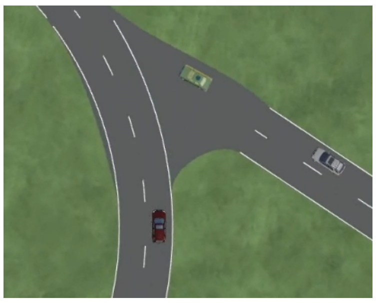


(b) Relative rewards for each action

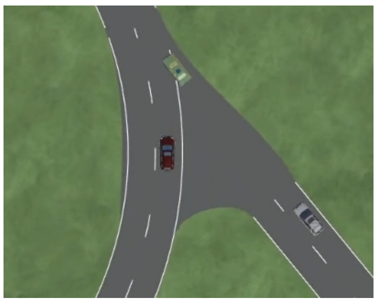
Figure 4-27: Test scenario 7: Results - Decision Making

Due to the consideration of uncertainty the ego vehicle is able to avoid a collision with the first traffic object by braking sufficiently and following behind it to continue along the path to its destination. It does not brake further for the second traffic vehicle because it uses the target brake maneuver prediction to predict its future state at the stop line. It does, however brake again at around the 8 second mark in response to the the traffic vehicle ahead of it.

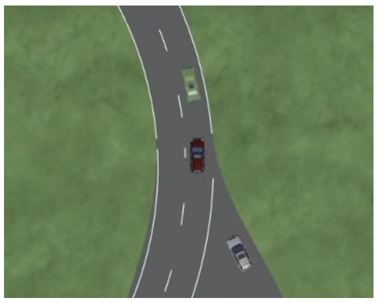
A visualization of the simulation output is shown in figure 4-28.



(a) t = 3.1s



(b) t = 4.4s



(c) t = 6.2s

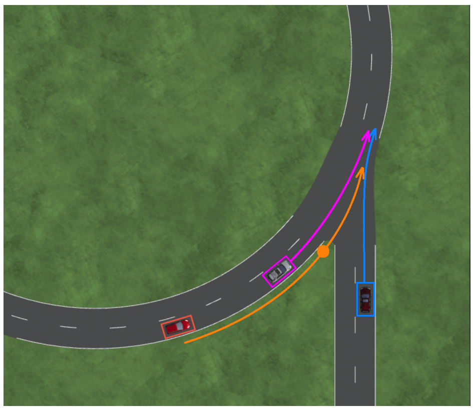
Figure 4-28: Test scenario 7: Visualization

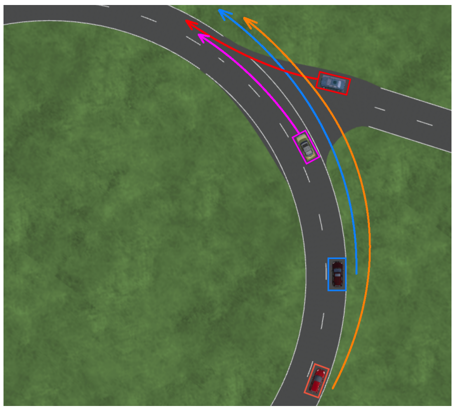
The overall performance of the decision making approach is quantified in in table 4-8:

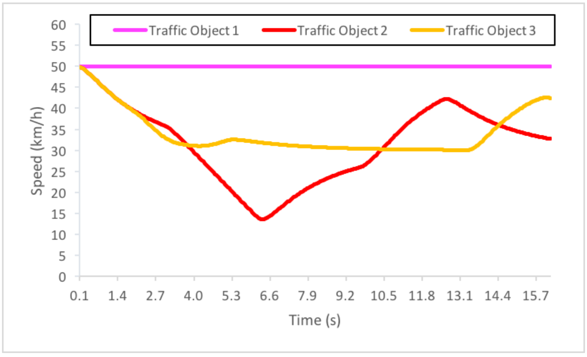
Criterion	Value
Minimum distance to vehicles	4.15m (30.5km/h)
Time to accelerate	0.2s
Number of action changes	6

Table 4-8: Performance evaluation - Decision Making

The next scenario within this scene is set up with two traffic objects already on the roundabout and another one approaching the roundabout from an on ramp ahead. This vehicle is set up to enter the roundabout in front of the ego vehicle as a worst case scenario. Figures 4-29 (a) and (b) depict the scenario and 4-29 (c) displays the velocity profile of the traffic objects.


(a) Intended routes: part 1

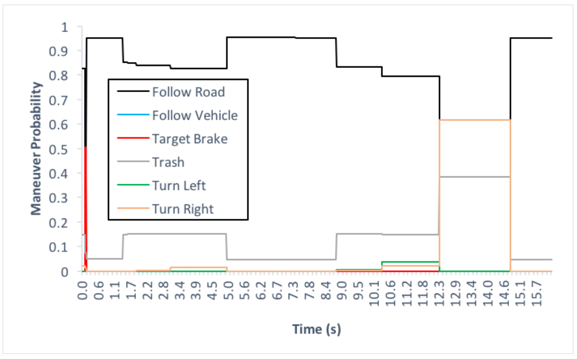

(b) Intended routes: part 2



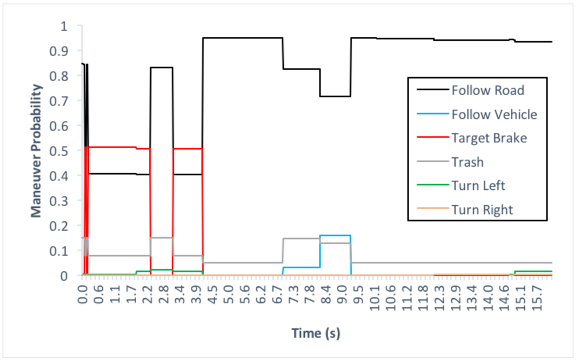
(c) Velocity profile of traffic

Figure 4-29: Test scenario 8: Setup

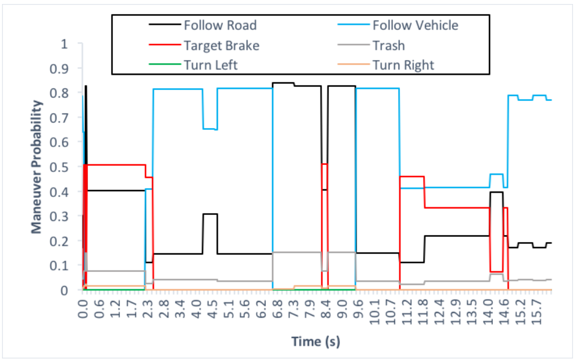
The corresponding behavior predictions of the traffic objects are seen in figure 4-30. At each stage, the Bayesian network is able to predict the Follow Road and Target Brake intentions of the traffic well enough to aid the decision making algorithm with the required information to make informed choices to avoid collisions and remain efficient.



(a) Traffic object 1



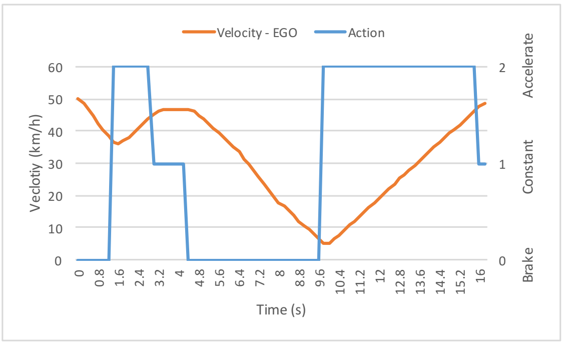
(b) Traffic object 2



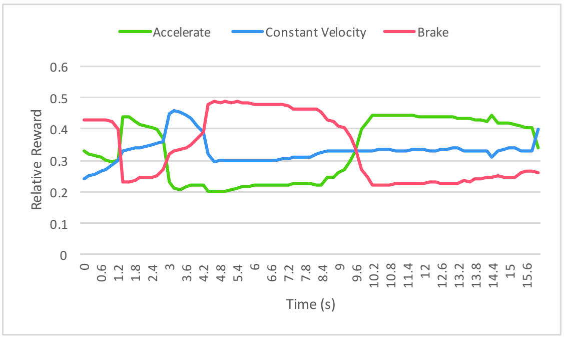
(c) Traffic object 3

Figure 4-30: Test scenario 8: Results - Behavior prediction

The output (chosen action) of the decision making framework along with the corresponding velocity of the ego vehicle over the course of the simulation is seen in figure 4-31.



(a) Chosen action and Ego vehicle velocity

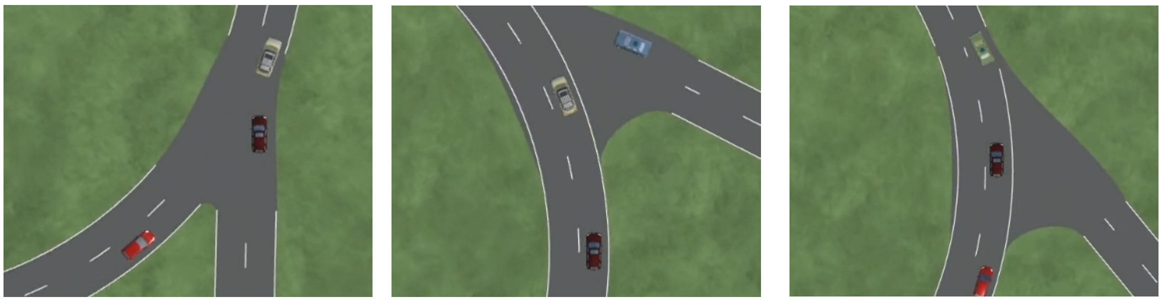


(b) Relative rewards for each action

Figure 4-31: Test scenario 8: Results - Decision Making

At the start of the simulation, there is a direct collision predicted between the first traffic object on the roundabout and the ego vehicle. Thus the ego vehicle brakes to avoid this, and notices that traffic object 3 is slowing down to give way so the ego vehicle continues to enter the roundabout and accelerates to follow the first vehicle. At the point when traffic object 2 is seen trying to enter the roundabout, the ego vehicle again predicts that the vehicle does not intend to give way so it chooses to brake and allow it to enter the roundabout. Following this, the ego vehicle accelerates back up to the speed limit.

A visualization of the simulation output is shown in figure 4-32.



(a) t = 1.6s **(b)** t = 6.0s **(c)** t = 9.6s

Figure 4-32: Test scenario 8: Visualization

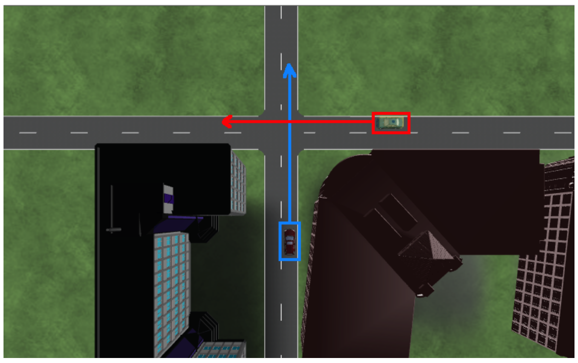
The overall performance of the decision making approach is quantified in in table 4-9:

Criterion	**Value**
Minimum distance to vehicles	3.76m (38.8km/h)
Time to accelerate	0.8s
Number of action changes	5

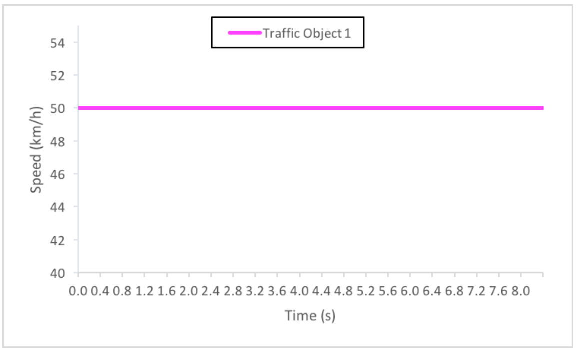
Table 4-9: Performance evaluation - Decision Making

4-2-5 Occlusions

So far all the evaluated scenarios have been with no occlusions blocking the traffic. In urban areas, it is of course very common to approach an intersection with no visibility of the approaching vehicles. Thus the final scenario is set up with multiple occlusions completely blocking any approaching vehicles at the intersection. One traffic vehicle crossing the intersection is added as shown in 4-33.



(a) Intended route of invisible traffic

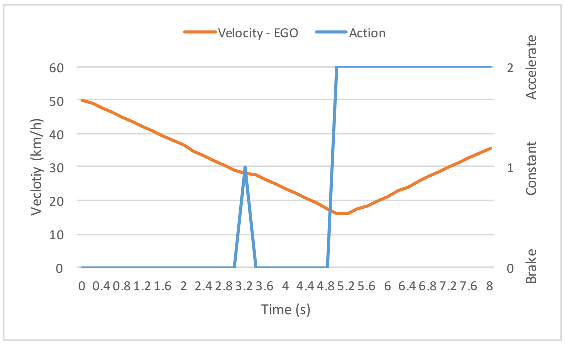


(b) Velocity profile of traffic

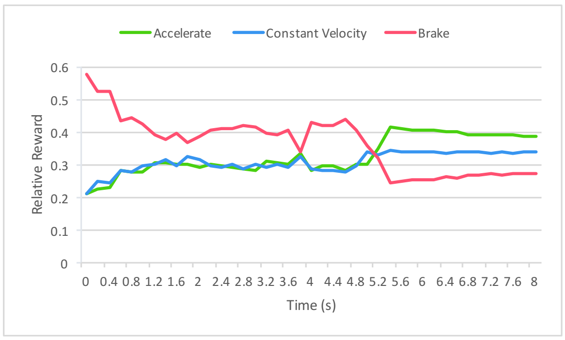
Figure 4-33: Test scenario 9: Setup

In this case, there is no behavior prediction because there is no state information of the traffic. Instead, the only vehicle states that exist are ones with high uncertainty behind each occlusion.

The output (chosen action) of the decision making framework along with the corresponding velocity of the ego vehicle over the course of the simulation is seen in figure 4-34.



(a) Chosen action and Ego vehicle velocity



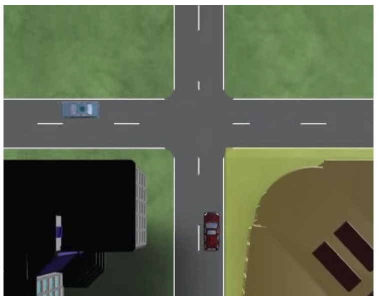
(b) Relative rewards for each action

Figure 4-34: Test scenario 9: Results - Decision Making

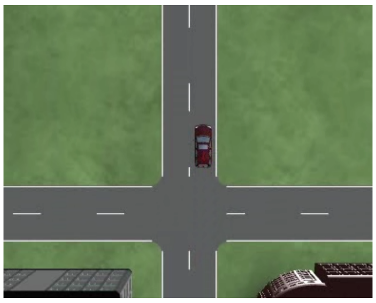
The added states behind each occlusion cause the ego vehicle to reduce it's speed sufficiently as it approaches the intersection, so that it is able to avoid any potential collision with currently invisible vehicles at the intersection. This turns out to be a good choice as the previously occluded traffic object appears and continues past the intersection. Once the ego vehicle has complete visibility of the intersection, it begins to accelerate back to the speed limit as it sees no danger. This is similar to a human driver's behavior when approaching an unsignalized intersection with no visibility.

A visualization of the simulation output is shown in figure 4-35.

(a) t = 4.6s



(b) t = 6.75s



(c) t = 9.0s

Figure 4-35: Test scenario 9: Visualization

The overall performance of the decision making approach is quantified in in table 4-10:

Criterion	Value
Minimum distance to vehicles	16.2m (20.0km/h)
Time to accelerate	0s
Number of action changes	3

Table 4-10: Performance evaluation - Decision Making

4-2-6 Computation time

Shifting our focus to the applicability of this approach to the real world, one of the most important criteria is real time capability. The work done in this thesis was demonstrated on an Intel Core i7-4770 processor using a single core at a clock speed of 2.5GHz. A comparison of execution times between the standard MCVI algorithm and the proposed approach are shown in table 4-11. Note that the average execution time is averaged over the entire simulation.

POMDP Solver	**Execution Time (s)**		
	Min.	Max.	Avg.
Standard Algorithm	0.468	5.128	2.23
Proposed Algorithm	0.11	2.13	0.657

Table 4-11: Computation time with 4 vehicles

From the values shown, it can be seen that while the proposed approach shows an average speedup of 3.39x over the standard MVCI algorithm, it falls short of real time capability on the tested hardware. Thus further development of the algorithm and/or parallelization approaches must be looked at to improve on this.

For example, discrete graphics processing units (GPUs) are a viable option to achieve a significant speed up. In [57], a basic POMDP algorithm along with the localization and perception units were deployed on a high performance NVIDIA GPU using CUDA, which allowed for parallelization to be exploited. The results of this work showed a speed up to the tune of 213x. Thus via extrapolation one can predict that a similar performance gain can be achieved using the algorithm proposed in this thesis to achieve real time performance.

Chapter 5

Conclusions and Future Work

This chapter brings this MSc thesis to a conclusion. Section 5-1 highlights the challenges and the contributions of this study. Section 5-2 identifies potential topics for future research.

5-1 Conclusion

This thesis began with the goal of developing a tactical decision making framework to deal with uncertain, dynamic environments (ie., urban areas). The main challenges were being able to accurately predict the future states of the traffic, while also considering the uncertainty associated with their intentions and the sensor measurements. The Partially Observable Markov Decision Process (POMDP) approach was chosen as the most suitable method for the task of decision making along with Bayesian networks to predict the behaviour of the traffic, and a multi model approach for future state prediction.

The problem was that online, continuous POMDPs are not always real time capable, so methods to make the solver more efficient had to be explored. Additionally, the general approach is not applicable to invisible vehicles, which is a common occurrence in dense urban areas. Thus the main contribution of this thesis is a highly optimized POMDP algorithm with a novel approach to dealing with occlusions resulting in a tactical decision making framework for autonomous driving in urban environments. The results of simulation at speeds upto 50km/h show that the approach works in all possible urban traffic scenarios, with a good level of efficiency and safety.

Regarding the real time capability of the work, while major improvements in the algorithm were obtained, the framework was short of real time capable on an Intel Core i7 processor. However, extrapolating using results from related work with parallelization of a basic POMDP algorithm, a safe prediction can be made that the approach in this thesis would be real time capable on a GPU. Coincidentally, this is congruent with the hardware proposed to be used for the autonomous vehicle at the PEM department of RWTH Aachen University.

5-2 Future Work

The defined work in this thesis was limited to longitudinal decision making, but in the real world lateral lane changes are also important in order to create an efficient path and have more options to avoid collisions. Thus the action set must be expanded and this would have a negative effect on the run time, but it must be considered that there would be additional headroom on the target hardware to preserve real time capability.

Having said this, it is prohibitive to use high end hardware in autonomous vehicles when looking at commercial production. Thus further improvements to the POMDP solver must be sought, such as the scenario sampling approach from DESPOT [51], or approximating the POMDP using deterministic closed loop forward simulations as in [58].

Another area of development - particularly for urban areas is the detection of pedestrians, prediction of their intentions and future states. The movement of pedestrians is considered even more stochastic than that of human drivers so consideration of uncertainty is paramount in this case. It must be noted that due to the generality and transparency of the chosen approach, the main requirement for expansion of the decision making framework to handle pedestrians is a pedestrian model for state transitions within the POMDP. This also traces back to the original justification of using this approach and not need extra training data or a retraining phase for expansion of the framework.

Finally, real world testing of the approach is vital as simulation of scenarios has its own inherent limitations. In the real word, drivers cooperate and react to each others' movements and also make mistakes. While scenarios that try and mimic stochastic driving behavior has been considered in this thesis, real world testing must be performed for actual validation.

Appendix A

Preliminaries

A-1 Probability theory

In the measurement world, disturbances such as noise are inevitable. The measured values and the hidden unobservable variables are referred to as random variables X. If this variable has a value of x, the Probability of this event is assigned to an expression $P(X = x)$.

The most important properties in probability theory are:

$$0 \leq P(A) \leq 1$$

and

$$\sum_{a_i \in A} P(A = a_i) = 1$$

This implies that the single probability never takes a negative value and does not exceed 1. Also the sum of all probabilities over all possible values from any random variable X always sum up to 1.

Now, consider two events A and B. If both events have already occurred, the conditional probability of $P(B|A)$ is considered. If the two events occur simultaneously then the joint probability is $P(A, B)$. The relation between the conditional probability and the joint probability is:

$$P(B|A) = \frac{P(A, B)}{P(A)}$$

Bayes' Theorem allows us to infer the probabilities of a series of events as:

$$P(B|A) = \frac{P(B)P(A|B)}{P(A)}$$

The total probability eliminates unconsidered random variables from the joint probability. Thus, the probability of the event B is determined from the compound probability $P(A, B)$ over the event space:

$$P(B) = \sum_{a_i \in A} P(B, A = a_i)$$

This resulting probability is also called a Marginal and the process is called Marginalization. We can now reformulate Bayes' Theorem as follows:

$$P(B|A) = \frac{P(B)P(A|B)}{\sum_{b_i \in B} P(b_i)P(A|b_i)}$$

Using the conditional probability relation, the above equation can be rewritten as:

$$P(B) = \sum_{a_i \in A} P(B, A = a_i) = \sum_{a_i \in A} P(B|A = a_i)P(A = a_i)$$

If the variable A is no longer varied and $A = a'$ has already been defined before the set of total probabilities is calculated, then in the discrete event space A:

$$P(B) = P(B, A = a')$$

Thus, only one term with $A = a'$ is calculated instead of the entire summation.

The compound probability can be determined using several interconnected conditional probabilities via a chain rule. For example:

$$P(A, B, C, D) = P(A)P(B, C, D|A) = P(A)P(B|A)P(C, D|A, B) = P(A)P(B|A)P(C|A, B)P(D|A, B, C)$$

If the events A and B are independent:

$$P(A, B) = P(A)P(B)$$

Thus:

$$P(B|A) = P(B)$$

A-2 Value function approximation in POMDPs

Let us begin by modeling the POMDP as a belief state Markov Decision Process (MDP). Simply put, it is an MDP that needs to be solved with states replaced by beliefs. The probability of belief state b transitioning to $b'_{b,a,o}(s')$ is given by the probability of receiving observation o in b. Thus:

$$p_b(b'|b, a) = p(o|b, a) \tag{A-1}$$

And the immediate reward would be $r_b(b, a) = \sum_{s \in S} b(s)r(s, a)$.

Focusing on value iteration in a POMDP, if a policy tree is used to represent all the possible actions that could be taken in response to receiving an observation (each of these is called a "plan"), the agent would have to choose from all these plans and develop a policy based on the belief it is in. While the policy tree is not as efficient as a decision tree, it is convenient to introduce the concept of α vectors.

If s is the initial state and the agent follows a policy PT for some time t, then it expects a reward of α, which is computed recursively by moving down a policy tree and choosing

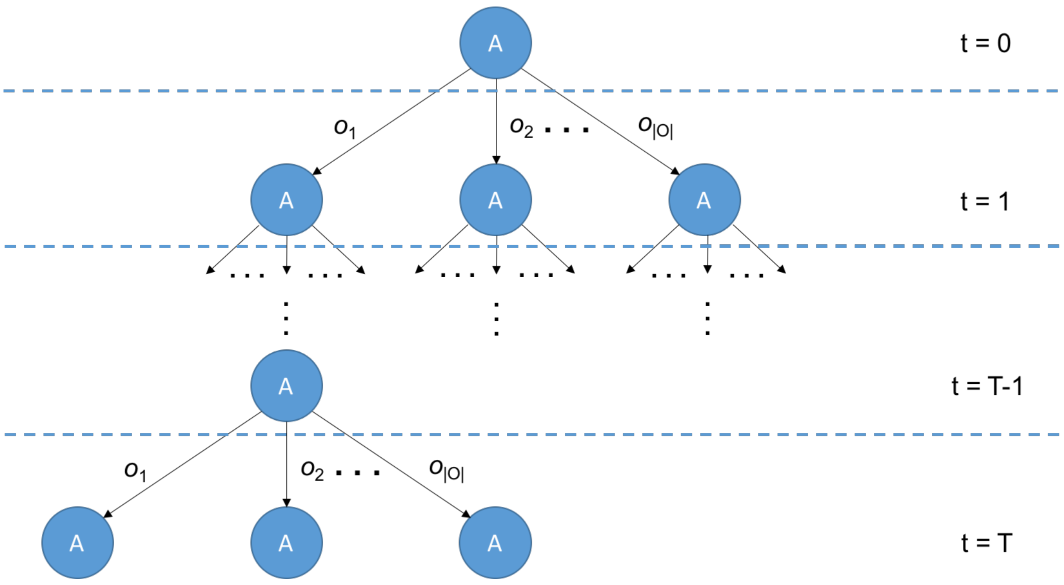


Figure A-1: Policy Tree

actions from it and is essentially the sum of immediate rewards and the discounted future rewards:

$$\alpha_{PT}(s) = r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) \sum_{o \in O} p(o|s') \alpha_{PT_{(o)}}(s') \tag{A-2}$$

Here the true state in not known and is estimated as a belief $b(s)$ so the value for initial belief b_0 is sought for after t steps, and this can be derived from the $\alpha_{PT}(s)$ for any belief b as:

$$V_{PT}(b) = \sum_{s \in S} b(s) \alpha_{PT}(s) \tag{A-3}$$

The optimal value function for a belief b and horizon t is thus the best policy tree amongst all policy trees:

$$V_t(b) = \max_{PT} V_{PT}(b) \tag{A-4}$$

Combining the equations A-3 and A-4, the finite horizon value function can be represented using a set of α vectors Γ as:

$$V(b) = \max_{\alpha \in \Gamma} \sum_{s \in S} b(s) \alpha(s) \tag{A-5}$$

Thus, these α vectors contain the value that a policy tree can receive and the value function for t-steps forms a piecewise linear convex simplex. The *curse of history* shows us that the total number of policy trees is infinite and the number of vectors representing the value function can become extremely large, and while a subset of these policy trees and their α

vectors actually have an impact on the optimal policy in the real world, this is till significant for problems with continuous spaces.

Now, since the belief state space is actually continuous and not discrete, the MDP cannot be solved directly. The derived POMDP value iteration from the previous belief state MDP results shows that the POMDP backup is a linear operator, which serves as a justification for using these α vectors.

Just as with regular MDPs, the nth Bellman backup of the value function can be calculated using the value function of the $n-1$th value function:

$$V^0(b) = r_b(b,a) \tag{A-6}$$

$$V^n(b) = \max_{a\in A} V_a^n(b) \tag{A-7}$$

$$V_a^n(b) = r_b(s,a) + \gamma \sum_{b'\in B} p_b(s'|s,a)V^{n-1}(b') \tag{A-8}$$

As per equation A-5, the $n-1$th value function can be represented by the α vectors in the set Γ^{n-1} and it is PWLC. Thus:

$$V^{n-1}(b'_{b,a,o}) = \sum_{s'\in S} b'_{b,a,o}\alpha^{n-1}_{b,a,o} \tag{A-9}$$

where

$$\alpha^{n-1}_{b,a,o} = \arg\max_{\alpha\in\Gamma^{n-1}} \sum_{s'\in S} b'_{b,a,o}\alpha \tag{A-10}$$

Rewriting equation A-8 using the transition, observation and reward functions, instead of the belief state MDP's reward $r_b(s,a)$ and transition $p_b(s'|s,a)$ function and simplifying:

$$V_a^n(b) = \sum_{s\in S}\left(r(s,a) + \gamma \sum_{o\in O}\sum_{s'\in S} p(o|s')p(s'|s,a)\alpha^{n-1}_{b,a,o}(s')\right) b(s) = \sum_{s\in S}\alpha^n_{b,a}(s)b(s) \tag{A-11}$$

$$V^n(b) = \max_{a\in A} V_a^n(b) \tag{A-12}$$

It turns out that the value function after the Bellman backup operation stays convex and if the value function in the $n-1$th step is PWLC, then it remains so in the nth step as well [59].

The policy that is derived using the nth value function's results is then:

$$\pi^n(b) = \arg\max_{a\in A} V_a^n(b) \tag{A-13}$$

Bibliography

[1] R. Matthaei and M. Maurer, "Autonomous driving–a top-down-approach," *at-Automatisierungstechnik*, vol. 63, no. 3, pp. 155–167, 2015.

[2] WHO, "Violence and Injury Prevention." http://www.who.int/violence_injury_prevention/road_safety_status/2015/GSRRS2015_data/en/, 2017. [Online; accessed 02-August-2017].

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

[4] J. Markoff, "Google cars drive themselves, in traffic," *The New York Times*, vol. 10, no. A1, p. 9, 2010.

[5] E. Donges, "A conceptual framework for active safety in road traffic," *Vehicle System Dynamics*, vol. 32, no. 2-3, pp. 113–128, 1999.

[6] P. Ramser, *Review of Decision Making in Action: Models and Methods.* American Psychological Association, 1993.

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

[8] S. Ammoun and F. Nashashibi, "Real time trajectory prediction for collision risk estimation between vehicles," in *Intelligent Computer Communication and Processing, 2009. ICCP 2009. IEEE 5th International Conference on*, pp. 417–422, IEEE, 2009.

[9] A. Barth and U. Franke, "Where will the oncoming vehicle be the next second?," in *Intelligent Vehicles Symposium, 2008 IEEE*, pp. 1068–1073, IEEE, 2008.

[10] A. Polychronopoulos, M. Tsogas, A. J. Amditis, and L. Andreone, "Sensor fusion for predicting vehicles' path for collision avoidance systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 3, pp. 549–562, 2007.

[11] T. Batz, K. Watson, and J. Beyerer, “Recognition of dangerous situations within a cooperative group of vehicles,” in *Intelligent Vehicles Symposium, 2009 IEEE*, pp. 907–912, IEEE, 2009.

[12] M. Althoff and A. Mergel, “Comparison of markov chain abstraction and monte carlo simulation for the safety assessment of autonomous cars,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1237–1247, 2011.

[13] D. Vasquez and T. Fraichard, “Motion prediction for moving objects: a statistical approach,” in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 4, pp. 3931–3936, IEEE, 2004.

[14] T. Streubel and K. H. Hoffmann, “Prediction of driver intended path at intersections,” in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, pp. 134–139, IEEE, 2014.

[15] M. Bahram, A. Wolf, M. Aeberhard, and D. Wollherr, “A prediction-based reactive driving strategy for highly automated driving function on freeways,” in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, pp. 400–406, IEEE, 2014.

[16] H. M. Mandalia and M. D. D. Salvucci, “Using support vector machines for lane-change detection,” in *Proceedings of the human factors and ergonomics society annual meeting*, vol. 49, pp. 1965–1969, SAGE Publications Sage CA: Los Angeles, CA, 2005.

[17] M. Schreier, V. Willert, and J. Adamy, “An integrated approach to maneuver-based trajectory prediction and criticality assessment in arbitrary road environments,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 10, pp. 2751–2766, 2016.

[18] T. Gindele, S. Brechtel, and R. Dillmann, “Learning context sensitive behavior models from observations for predicting traffic situations,” in *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*, pp. 1764–1771, IEEE, 2013.

[19] D. A. Pomerleau, “Efficient training of artificial neural networks for autonomous navigation,” *Neural Computation*, vol. 3, no. 1, pp. 88–97, 1991.

[20] S. Kammel, J. Ziegler, B. Pitzer, M. Werling, T. Gindele, D. Jagzent, J. Schröder, M. Thuy, M. Goebl, F. v. Hundelshausen, *et al.*, “Team annieway's autonomous system for the 2007 darpa urban challenge,” *Journal of Field Robotics*, vol. 25, no. 9, pp. 615–639, 2008.

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

[22] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, *et al.*, “Making bertha driveâĂŤan autonomous journey on a historic route,” *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 2, pp. 8–20, 2014.

[23] M. P. Vitus and C. J. Tomlin, “A probabilistic approach to planning and control in autonomous urban driving,” in *52nd IEEE Conference on Decision and Control*, pp. 2459–2464, IEEE, 2013.

[24] F. Meng, J. Su, C. Liu, and W.-H. Chen, “Dynamic decision making in lane change: game theory with receding horizon,” in *Control (CONTROL), 2016 UKACC 11th International Conference on*, pp. 1–6, IEEE, 2016.

[25] C. Hubmann, M. Aeberhard, and C. Stiller, “A generic driving strategy for urban environments,” in *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, pp. 1010–1016, IEEE, 2016.

[26] W. Liu, S.-W. Kim, Z. J. Chong, X. Shen, and M. H. Ang, “Motion planning using cooperative perception on urban road,” in *Robotics, Automation and Mechatronics (RAM), 2013 6th IEEE Conference on*, pp. 130–137, IEEE, 2013.

[27] A. Arab, K. Yu, J. Yi, and D. Song, “Motion planning for aggressive autonomous vehicle maneuvers,” in *Automation Science and Engineering (CASE), 2016 IEEE International Conference on*, pp. 221–226, IEEE, 2016.

[28] K. Driggs-Campbell and R. Bajcsy, “Communicating intent on the road through human-inspired control schemes,” in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pp. 3042–3047, IEEE, 2016.

[29] J. Nilsson, M. Brännström, E. Coelingh, and J. Fredriksson, “Longitudinal and lateral control for automated lane change maneuvers,” in *2015 American Control Conference (ACC)*, pp. 1399–1404, IEEE, 2015.

[30] J. Suh, B. Kim, and K. Yi, “Stochastic predictive control based motion planning for lane change decision using a vehicle traffic simulator,” in *Transportation Electrification Asia-Pacific (ITEC Asia-Pacific), 2016 IEEE Conference and Expo*, pp. 900–907, IEEE, 2016.

[31] S. Ulbrich and M. Maurer, “Probabilistic online POMDP decision making for lane changes in fully automated driving,” *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pp. 2063–2070, 2013.

[32] S. Ulbrich and M. Maurer, “Towards Tactical Lane Change Behavior Planning for Automated Vehicles,” *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, vol. 2015-Octob, pp. 989–995, 2015.

[33] D. K. Kye, S. W. Kim, and S. W. Seo, “Decision making for automated driving at unsignalized intersection,” *ICCAS 2015 - 2015 15th International Conference on Control, Automation and Systems, Proceedings*, no. Iccas, pp. 522–525, 2015.

[34] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, “Intention-aware online POMDP planning for autonomous driving in a crowd,” *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June, pp. 454–460, 2015.

[35] S. Brechtel, T. Gindele, and R. Dillmann, “Solving continuous pomdps: Value iteration with incremental learning of an efficient space representation,” in *International Conference on Machine Learning*, pp. 370–378, 2013.

[36] C. Isidore, “Google looking to hire self-driving car ’drivers’.” http://money.cnn.com/2016/05/13/autos/google-self-driving-drivers/, 2016. [Online; accessed 02-January-2017].

[37] F. Lambert, “Tesla has now 1.3 billion miles of Autopilot data going into its new self-driving program.” https://electrek.co/2016/11/13/tesla-autopilot-billion-miles-data-self-driving-program/, 2016. [Online; accessed 02-January-2017].

[38] M. Bojarski, P. Yeres, A. Choromanska, K. Choromanski, B. Firner, L. Jackel, and U. Muller, “Explaining how a deep neural network trained with end-to-end learning steers a car,” *arXiv preprint arXiv:1704.07911*, 2017.

[39] D. Castelvecchi, “Can we open the black box of ai?,” *Nature News*, vol. 538, no. 7623, p. 20, 2016.

[40] F. V. Jensen, *An introduction to Bayesian networks*, vol. 210. UCL press London, 1996.

[41] I. Ben-Gal, “Bayesian networks,” *Encyclopedia of statistics in quality and reliability*, 2007.

[42] T. A. Stephenson, “An introduction to bayesian network theory and usage,” tech. rep., IDIAP, 2000.

[43] V. Mihajlovic and M. Petkovic, “Dynamic bayesian networks: A state of the art,” 2001.

[44] D. Barber, *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

[45] G. K.-I. C. Beierle and G. Kern-Isberner, *Methoden wissensbasierter Systeme*. Springer, 2003.

[46] K. Murphy, “Bayes Net Toolbox for Matlab.” https://github.com/bayesnet/bnt, 2014. [Online; accessed 21-May-2017].

[47] D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, and D. P. Bertsekas, *Dynamic programming and optimal control*, vol. 1. Athena scientific Belmont, MA, 1995.

[48] H. Bai, D. Hsu, W. S. Lee, and V. a. Ngo, “Monte Carlo Value Iteration for Continuous State POMDPs,” *Int. Workshop on the Algorithmic Foundation of Robotics (WAFR)*, pp. 175–191, 2011.

[49] M. S. J. Porta, M. Spaan, and N. Vlassis, “Value iteration for continuous-state pomdps,” *IAS Technical Report, University of Amsterdam, Tech. Rep. IAS-UVA-04-04*, 2004.

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

[51] A. Somani, N. Ye, D. Hsu, and W. Lee, “DESPOT : Online POMDP Planning with Regularization,” *Advances in Neural Information Processing Systems*, pp. 1–9, 2013.

[52] T. Smith and R. Simmons, “Heuristic search value iteration for pomdps,” in *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pp. 520–527, AUAI Press, 2004.

[53] M. Hauskrecht, “Value-function approximations for partially observable markov decision processes,” *Journal of Artificial Intelligence Research*, vol. 13, pp. 33–94, 2000.

[54] H. Kurniawati, D. Hsu, and W. S. Lee, “Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces.,” in *Robotics: Science and Systems*, vol. 2008, Zurich, Switzerland, 2008.

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

[56] J. Ziegler, P. Bender, T. Dang, and C. Stiller, “Trajectory planning for berthaâĂŤa local, continuous method,” in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, pp. 450–457, IEEE, 2014.

[57] F. J. Ramirez-Rodriguez, “Parallelization of existing algorithms of self-driving cars: Decisions, tracking, visualization,” Master’s thesis, PEM, RWTH Aachen, 04 2017.

[58] E. Galceran, A. G. Cunningham, R. M. Eustice, and E. Olson, “Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction.,” in *Robotics: Science and Systems*, 2015.

[59] E. J. Sondik, “The optimal control of partially observable markov processes.,” tech. rep., DTIC Document, 1971.