THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

# Learning When To Drive in Uncertain Scenarios

*A deep Q-learning approach*

Tommy Fujita Tram

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden, 2024

**Learning When To Drive in Uncertain Scenarios**

*A deep Q-learning approach*

Tommy Fujita Tram

ISBN 978-91-8103-089-1



Doktorsavhandlingar vid Chalmers tekniska högskola
Ny serie nr 5547
ISSN 0346-718X

Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg, Sweden
Phone: +46 (0)31 772 1000
Email: tommy.tram@chalmers.se; tommy.tram@gmail.com

Cover:
An illustration of two vehicles approching an intersection. The blue vehicle has to make a decision to yield or drive while the intention of the red vehcile is uncertain.

Printed by Chalmers Reproservice
Gothenburg, Sweden, August 2024

*To my loving family*

**Learning When To Drive in Uncertain Scenarios**

*A deep Q-learning approach*

Tommy Fujita Tram
Department of Electrical Engineering
Chalmers University of Technology

# Abstract

The main focus of this thesis is tactical decision-making for autonomous driving (AD) through intersections with other road users. Human drivers can navigate diverse environments and situations, even those they have never encountered before. Autonomous vehicles are expected to have similar capabilities. This thesis specifically addresses the challenge of navigating intersections where the intentions of other drivers are unknown, as these intentions can be influenced by factors such as driver mood, attention, right-of-way, and traffic signals.

To tackle the complexity of manually specifying reactions for every possible situation, this thesis adopts a learning-based strategy using reinforcement learning (RL). The problem is formulated as a partially observable Markov decision process (POMDP) to account for the uncertainty of unknown driver intentions. A general decision-making agent, based on the deep Q-learning algorithm, is proposed. The contributions of this thesis include the development and application of this method to various simulated intersection scenarios, demonstrating its adaptability and effectiveness in different environments with minimal modifications. By accounting for the inherent uncertainty in driver behavior, this approach enhances the robustness and reliability of the autonomous driving system.

**Keywords:** Autonomous driving, reinforcement learning, decision making, uncertain environments, Partially observable Markov decision process, deep Q-learning, transfer learning, model predictive control, neural networks

# List of Publications

This thesis is based on the following publications:

[A] Tommy Tram, Anton Jansson, Robin Grönberg, Mohammad Ali, and Jonas Sjöberg, "Learning Negotiating Behavior Between Cars in Intersections using Deep Q-Learning". *Published in 2018 21st International Conference on Intelligent Transportation Systems (ITSC).*

[B] Tommy Tram, Ivo Batković, Mohammad Ali, and Jonas Sjöberg, "Learning When to Drive in Intersections by Combining Reinforcement Learning and Model Predictive Control". *Published in 2019 IEEE Intelligent Transportation Systems Conference (ITSC).*

[C] Carl-Johan Hoel, Tommy Tram, and Jonas Sjöberg, "Reinforcement Learning with Uncertainty Estimation for
Tactical Decision-Making in Intersections". *Published in 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC).*

[D] Tommy Tram, Maxime Bouton, Jonas Fredriksson, Jonas Sjöberg, and Mykel Kochenderfer, "Belief State Reinforcement Learning for Autonomous Vehicles in Intersections". *Submitted to IEEE Transactions on Intelligent Transportation Systems.*

[E] Hannes Eriksson, Tommy Tram, Debabrota Basu, Mina Alibeigi, and Christos Dimitrakakis, "Reinforcement Learning in the Wild with Maximum Likelihood-based Model Transfer". *Published in 2024 International Conference on Adaptive Agents and Multi-Agent Systems (AAMAS).*

# Acknowledgments

This thesis is the result of many years of hard work, lots of travel and a great learning experience. Even in times when things did not work as expected or unexpected problems emerged, I never felt that I was alone. That is why I would like to acknowledge the people who have made this work possible and memorable.

First, I want to thank my supervisor and examiner Professor Jonas Sjöberg. Thank you for believing in me and for the opportunity of pursuing a doctoral degree as your student. It has been an honor and a pleasure. I would also like to thank Jonas Fredriksson that assisted in the final year and made it possible for me to cross the finish line.

A special thanks to my industrial supervisor Dr. Mohammad Ali, who chose and encouraged me to pursue a PhD in this field. I still remember the pitch that convinced me to do it. You said: "Pilot assist is mostly done now, we need someone to look towards the future of decision-making and control". We had a lot of fun in the beginning of this journey and I will always be grateful for the guidance and support you provided, and most of all for believing in me.

The work presented in this thesis would not have been possible without the financial support from VCC, Zenuity, Zenseact, and the Wallenberg AI, Autonomous Systems and Software Program (WASP). I am also grateful for the community and environment created by both Zenseact and WASP. The Advanced Graduate Program led by Dr. Mats Nordlund and Dr. Carl Lindberg created an environment within the company which cultivated a safe space to share ideas, talk amongst peers and fun to go to work in the morning.

I would especially like to thank Dr. Carl Lindberg for the coaching during these final years of my PhD. At the start of the pandemic, you paid attention to our casual conversations, identified problems and proposed solutions I did not think was possible, but more importantly you cared. Thanks to you, I was able to finish my final year of my PhD together with my wife on the other side of the world and as a direct result of that, I now have a beautiful son. For this, I am eternally grateful.

I especially want to thank my great friends and co-authors with whom I embarked on this PhD journey with. Carl-Johan Hoel, my research twin, thank you for all the support over the years, the enlightening conversations and most of all for being a friend. Ivo Batkovic, thank you for all the good times we shared. I have always admired your determination and focus. Working

with you has been a pleasure and a privilege. I am happy to call you my friend. I would also like to thank Anton Jansson and Robin Grönberg for their well executed master thesis from where I found the direction of this thesis. Additionally, I am grateful to Hannes Eriksson and Debabrota Basu for our collaboration on the transfer learning experiments.

Special thanks go to Christian Rodriguez for being a great friend and supporting me whenever times were tough.

I would like to thank WASP for all the interesting study trips, courses, and events that not only widened my skills, but also introduced me to a vast network of colleagues and friends across the world. In addition, the WASP exchange program gave me the opportunity to spend six months as a visiting research student at the Stanford Intelligent Systems Laboratory (SISL). To that end, I am deeply grateful to Professor Mykel Kochenderfer who thought me that life is a POMDP and gave me the opportunity to spend time and work with his research group. Additionally, special thanks to Maxime Bouton for the fun we had during my time at SISL. I learned a lot, both on and off campus.

*Tommy Fujita Tram,*
*Göteborg, August, 2024.*

# Acronyms

| | |
|---|---|
| AD: | Autonomous driving |
| ADAS: | Advanced driving assistance systems |
| AV: | Autonomous vehicles |
| ASIL: | Automotive Safety Integrity Level |
| CNN: | Convolutional neural network |
| DQN: | Deep Q-network |
| DRQN: | Deep Recurrent Q-network |
| EQN: | Ensemble quantile networks |
| IDM: | Intelligent driver model |
| LSTM: | Long short-term memory |
| LQR: | Linear Quadratic Regulator |
| MDP: | Markov decision process |
| MPC: | Model predictive control |
| MCTS: | Monte Carlo tree search |
| NN: | Neural network |
| POMDP: | Partially observable Markov decision process |
| SGD: | Stochastic gradient descent |
| RPF: | Randomized prior functions |
| RL: | Reinforcement learning |

# Contents

# Part I

# Overview

# CHAPTER 1

# Introduction

The way we transport ourselves is currently evolving, and Autonomous Driving (AD) technology is expected to have a big impact on this transformation [1], [2]. With Autonomous Vehicles (AV) the efficiency of traffic can be improved by scheduling commercial transports outside of rush hours [3]. The number of parking spots in cities can be reduced if the vehicles can autonomously drive itself to a less crowded area when not in use and drive back when needed. Congestion and traffic jams could also be reduced if a large amount of vehicles in traffic are autonomous and optimize around the same goal e.g., traffic flow or fuel efficiency.

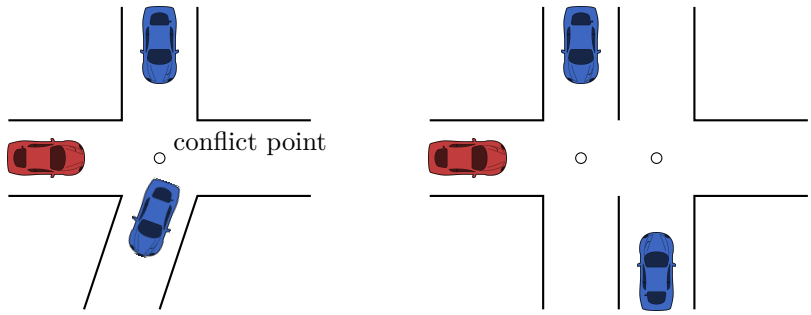
The rapid success of Machine Learning (ML) during the last decades has lead to major progress towards deploying AVs in the real world. One clear beneficiary of these new ML techniques are the perception systems [4]. Better perception enables more accurate representation of the environment. However, navigating complex scenarios such as urban intersections and roundabouts with dense traffic remain challenging for AVs because it requires a higher level of interaction between road users, as summarized in a review paper [5]. When human drivers approaches an intersection, they normally assess who has the right of way, before deciding whether to proceed or yield to other road

users. However, human drivers do not always follow the rules. According to the Insurance Institute for Highway Safety [6], in 2019, an estimated 115,741 people were injured by drivers running a red light, whereas 928 of them were killed. While these accidents were mainly caused by driver inattention or reckless driving, it motivates the development of decision-making algorithms for AVs which not only follow the traffic rules but can also take into account other drivers future actions and inattention, which is the main focus in this thesis.

## 1.1 Autonomous Driving Levels

When talking about autonomous driving, it is first important to specify which level of autonomy that is being discussed. The Society of Automotive Engineers has classified these different levels of autonomy ranging from zero to five [7], also referred to as L0-L5. The first level L0 is a vehicle with no autonomy, whereas a fully AV that can operate in any environment and without any human supervision is defined as L5. Popular Advanced Driver Assistance Systems (ADAS) functions today, like lane centering or adaptive cruise control are classified as L1, while the Volvo Pilot Assist and Tesla Autopilot that provide both steering and acceleration/braking are classified as level 2. The main criterion for L2 systems is that the driver is in control and only supported by the system. This requires the driver to always supervise the vehicle and take over when needed to ensure safety. For L3 and higher, the responsibility of driving shifts to the system. At L3, the driver still has to take control over the vehicle, but only when the systems request it. At L4 and L5, the autonomous driving features no longer require the driver to take over. The main difference between L4 and L5 is the capability of driving anywhere under all conditions. Examples of L4 are robot taxis developed by Waymo, Zoox, Cruise and Toyota which only operate in a specified area or city.

The methods presented in this thesis are aimed at an autonomy level L3-L5. At the highest level the system is expected to handle all aspects of driving within a specific task such as crossing an intersection at any location.




**(a)** Single intersection **(b)** Double intersection

**Figure 1.1:** Examples of different intersections

## 1.2 Intersections, intentions and scenarios

This section aims to clarify the use of the terms' *intersection*, *intention*, and *scenario* within the context of this thesis. An intersection refers to the geometric layout of roads intersecting each other, encompassing elements such as the number of junctions, conflict points, turns, and angles of incidence, as illustrated in Figure 1.1. Intersections can be categorized as signalized or unsignalized. A signalized intersection is equipped with infrastructure to designate the right-of-way, such as regulatory signs (e.g., STOP or YIELD) or traffic signals. In contrast, an unsignalized intersection lacks such features, relying on local driving rules, such as giving the right-of-way to vehicles approaching from the right. However, as emphasized in the introduction, human drivers do not always follow these right-of-way rules, which can result in accidents. Therefore, this thesis defines intentions as the anticipated actions of other vehicles in the future, such as stopping or proceeding through the intersection.

Assume there are two main intentions: "take way" and "give way" (yield). In Figure 1.2, three different agents with a starting velocity 12m/s are approaching an intersection. At time 0s, agents with a give way intention will start to brake. With these two main intentions, other intentions can be derived; for example, a cautious intention can be modeled as a give way agent that changes its intention to take way at 2 seconds. The reason for the change can be that a cautious agent lowers the speed initially to have the possibility to give way,

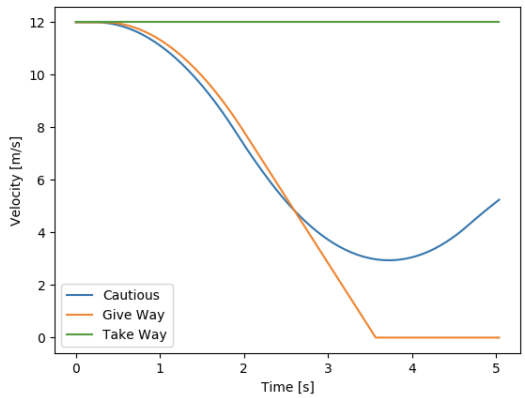



**Figure 1.2:** Velocity profiles illustrating three different intentions for an agent approaching an intersection from the same initial position and velocity. The green curve represents an agent with a 'take way' intention, while the orange curve shows an agent intending to 'give way,' stopping before the intersection. The blue curve depicts a cautious intention, starting with a slowdown but without coming to a complete stop.

and later speed up only if there is no potential conflict. Distinguishing between the 'give way' and 'take way' intentions poses a challenge because even after the initial braking at 0 seconds, it is not certain that the agent will stop until 3 seconds after the initial braking. This uncertainty may be considered too conservative by the person riding the AV.

If driver intentions can be accurately predicted, intersections could be managed without the need for traffic lights, as right-of-way decisions can be made based on intentions rather than relying solely on infrastructure. A Reinforcement Learning (RL) approach could enhance driving in scenarios where understanding intentions is crucial, such as at intersections with both stop signs and traffic lights, or when a vehicle violates traffic rules by running a red light. By training an RL agent in a simulation environment with varying driver intentions, it can learn to infer these intentions and make optimal decisions through trial and error.

## 1.3 Research questions

This thesis focuses on investigating and evaluating RL agents for navigating complex intersection scenarios, with a particular emphasis on managing uncertainty. The intersection navigation problem is formulated as a Partially Observable Markov Decision Process (POMDP), where observable states include positions and velocities of vehicles, while unobservable states pertain to the intentions of surrounding drivers. With this in mind, this thesis aims to answer the following research questions:

**Q1.** How can RL techniques be used to develop a decision-making agent that effectively navigates intersections without explicitly estimating the intention state of other vehicles?

A deep Q-learning approach is used to solve the POMDP, with short-term goal as actions. These short-term goals are translated into reference points and constraints for a controller. Paper A uses a sliding mode controller, while Paper B uses a Model Predictive Control (MPC) to generate the vehicles' acceleration. To address the challenge of unobservable intentions, the hidden state in the Long Short-Term Memory (LSTM) layer of the RL agent is designed to incorporate estimations of these intentions.

**Q2.** How can an RL agent utilize the uncertainty in its predictions and actions to enhance decision-making in complex environments?

Two main sources of uncertainty are tackled in this thesis: Q-value uncertainty and intention state uncertainty. For Q-value uncertainty, Paper C estimates the uncertainty in the Q-values predicted by using an ensemble of neural networks on different subsets of the available data. This ensemble provides a distribution over the estimated $Q$-values and the uncertainty in choosing different actions can be defined as the coefficient of variation of the $Q$-values. In Paper D, the hidden intention states of other vehicles are represented using a belief state generated via a particle filter. This belief state provides a probability distribution over possible true states, which informs the agent's decisions.

Two methods are proposed to handle intention estimation: QMDP-IE and QID. QMDP-IE, derived from the QMDP algorithm, uses a single estimated intention state, reducing computational complexity while blending elements of POMDPs and MDPs. In contrast, QID represents the intention state using a

belief state distribution, allowing for more adaptive and robust decision-making in uncertain environments.

**Q3.** How can an RL agent handle situations it has not been trained on?

A confidence criterion is applied to the uncertainty of the estimated $Q$-values in Paper C, making actions with high uncertainty invalid and if no action satisfies the confidence criterion a backup action e.g., emergency braking, can be applied avoiding potential collisions. If the uncertainty measure from Paper C and Paper D can be used to identify that the agent is in the wrong environment, the transfer RL approach from Paper E can be employed to determine which environment out of a set of environments the agent is currently in, or to select the policy that best fits the current situation

## 1.4 Scope and limitations

The following aspects of creating a tactical decision-making agent for autonomous driving in uncertain environments are not considered in this thesis.

1. Guaranteeing safety in AD systems is an important open question that is out of scope for this thesis.
2. The work in this thesis is tested in simulation environments and not real world.
3. This work considers the control of one vehicle and not multiple agents.

## 1.5 Contributions

The main contributions of this thesis are:

1. A deep Q-learning approach for creating a decision-making agent navigating intersection that considers the intentions of other drivers.
2. A neural network architecture that is invariant to permutations of the order of which surrounding traffic participants are observed, which speeds up training and improves the quality of the trained agent.
3. A belief state representation of driver intentions using a particle filter.

4. A belief state Deep Q-network (DQN) method that can adjust the aggressiveness of the policy using one threshold parameter.
5. Extension of RL methods that provide an uncertainty estimate of the proposed decisions and use it to create a confidence criterion that can identify situations with high uncertainty.
6. A transfer learning method that is able to identify which Markov Decision Process (MDP) the agent is in from a set of MDPs.

## 1.6 Thesis outline

The outline of the thesis is as follows: in Chapter 2 other research in the same field is presented. Chapter 3 introduce the mathematical framework MDP and POMDP with a brief theory of RL, deep Q-learning and the Intelligent Driver Model (IDM). Chapter 4 is where the problem is formulated by defining the components of the POMDP. Results from using deep Q-learning to solve the POMDP is presented and later combined with an MPC to improve the actions. Later in Chapter 5 two approaches to handle the uncertainty is presented. First the uncertainty in the decisions from the RL algorithm and then an empirical study of how well a DQN can handle uncertainty of others driving intentions. Chapter 6 present an approach to generalize over different MDPs more specifically policies learned from different transfer functions. The synergies and differences of the different methods are highlighted in Chapter 7. Finally, Chapter 8 provides some concluding remarks and future research directions.

# CHAPTER 2

# Related work

In recent years, decision-making for AVs in structured scenarios like intersections has attracted a lot of attention in the literature. This chapter provides a broad introduction to the primary research directions and outlines how the contributions in this thesis relate to existing work. However, it does not aim to provide a comprehensive survey of every approach.

## 2.1 Rule-based methods

A skilled engineer can sometimes solve the decision-making problem for structured traffic scenarios using rule-based methods. One example of a rule-based method was implemented using hierarchical state machines to switch between predefined behaviors depending on what scenarios was encountered [8], [9]. These methods were successful for a limited and controlled environment such as the Urban Challenge event, but it is difficult for an engineer to anticipate every situation that may occur in the real world and design a suitable strategy that can solve all of them, in particular when drivers are not following the law [10]. The limitations of rule-based approaches motivate the choice of more adaptive and flexible methods, such as a planning-based or learning-based

method.

## 2.2 Planning-based methods

Planning-based methods treats the decision-making task as a motion planning problem. Commonly, a prediction model is used to predict the motion of the other agents, and then the behavior of the ego vehicle that is being controlled is planned accordingly. Liebner *et al.* [11] used to the IDM infer driver intent in urban intersections and Hoermann *et al.* [12] used a particle filter to estimate the parameters of the IDM, both works showed promising results when evaluated on real-world data. Consequently, the IDM is used in this thesis to model the driving behavior of other vehicles.

One planning-based method is using Monte Carlo Tree Search (MCTS) [13], but since the predictions are independent of the ego vehicle plan results in a reactive behavior [14], [15]. Therefore, the interaction between the ego vehicle and other agents is not explicitly considered, but may happen implicitly by frequent replanning. MCTS also requires extensive online computation and can be hard to scale in complex traffic situations with an increasing number of traffic participants.

Another approach to solve the motion planning problem is to use optimal control, which was applied to highway driving scenarios by Werling *et al.* [16]. Since human behavior is complex and varies between individuals, a study by Damerow *et al.* [17] use a probabilistic prediction as input to the motion planning, which aims to minimize the risk of collision during an intersection scenario. While Batkovic *et al.* [18] used a robust scenario MPC approach to handle uncertain multi-modal road users. Other approaches to motion planning for autonomous driving are provided in the surveys by González *et al.* [19] and Paden *et al.* [20]. However, these planning-based methods rely on the accuracy of the prediction models and require a lot of on-board computing power which may be limited in an AV.

## 2.3 Learning based methods

Learning-based approaches offer the ability to learn from experience, adapt to new situations, and make decisions based on a wide range of scenarios, rather than relying on predefined rules or models. RL methods can help relieve

the burden of designing hand-crafted solutions for all possible scenarios [21], [22]. The work by Mnih *et al.* [23] showed a DQN that achieved impressive results in training agents to play Atari games from raw pixel inputs using experience replay, highlighting DQNs ability to learn complex behaviors from high-dimensional sensory data, a key requirement for autonomous vehicles navigating intersections. In order to handle the uncertainty of predicting other traffic participants' behaviors or intentions, the literature formulates the problem of driving under uncertainty as a POMDP [24]. Deep Recurrent Q-Network (DRQN) approaches, such as the ones from Hausknecht *et al.* [25] and Zhu *et al.* [26], showed some promise solving POMDP with non-observable states by leveraging past observations or actions.

Another approach by Bouton *et al.* [27] used belief states to capture uncertainties in the environment. The belief state can be used to model the probability distribution over the uncertain world states, e.g., the intention of other road users. Wang *et al.* [28] decoupled the belief state modeling (via unsupervised learning) from policy optimization (via RL) and Littman *et al.* [29] claimed that having full observability at learning time, combined with knowing what will not be observable at deployment time, enables an RL agent to learn a policy that is more robust to its unobservable states.

Advantage of DQN methods, compared to planning based methods, is that DQN learns through interaction with the environment. Experience replay allows DQN to learn efficiently from past experiences, improving its performance with less real-world data compared to methods without it. This capability enables autonomous vehicles to adapt to unseen situations and make informed decisions based on accumulated experiences at intersections. While DQN doesn't directly address driver intentions, it can learn to infer them indirectly from traffic patterns and historical data. This allows for adaptive decision-making based on the perceived likelihood of driver actions at intersections. Therefore, DQN is a suitable choice for this thesis due to its adaptability and efficiency in learning from complex, dynamic environments.

# CHAPTER 3

# Technical background

This chapter briefly introduces the MDP framework and its extension POMDP and RL. A more comprehensive overview of POMDPs and RL is given in the books by Kochenderfer [24] and Sutton and Barto [21], upon which this chapter is based. The purpose of the chapter is to summarize the most important concepts and introduce the notation that are used in the subsequent chapters.

## 3.1 Markov decision process

A MDP is a mathematical framework for modeling discrete time sequential decision-making problems. It involves an agent making decisions in an environment evolving over time according to a stochastic process. The state of the environment contains all the information necessary about the agent and environment at a given time to be able to transition to any given state. This property is referred to as the Markov property.

The MDP is formally defined as the tuple $(\mathcal{S}, \mathcal{A}, T, R, \gamma)$, described by the following list [24]:

- The state space $\mathcal{S}$ represents the set of all possible states of the environ-

ment. This set could consist of both discrete and continuous states.

- The action space $\mathcal{A}$ represents the set of all possible actions the agent can take. The action space can consist of both discrete and continuous actions. Since this thesis focuses on high-level decision-making, only discrete actions are considered.

- The state transition model $T(s' \mid s, a)$ describes the probability $\Pr(s' \mid s, a)$ that the system transitions to the next state $s' \in \mathcal{S}$ from state $s \in \mathcal{S}$ when action $a \in \mathcal{A}$ is taken.

- The reward function $R(s, a)$ returns a scalar reward $r$ for each action $a$ an agent takes in a given state $s$. The design of the reward function should reflect the overall objective that the agent should maximize.

- The discount factor $\gamma \in [0, 1)$ is a scalar that discounts the value of future rewards. The discount factor $\gamma$ will affect the results of the optimization problem. A discount factor set close to 0 will make immediate rewards more important while a $\gamma$ closer to 1 would give some weight to expected future reward as well.

A policy $\pi$ is defined as the mapping from state $s$ to action $a$ and the goal of the agent is to take a sequence of actions that maximizes the accumulated reward $r$. The value of being in a state while following a policy is described by the value function

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R(s_t, a_t) | s_0 = s, \pi\right]. \tag{3.1}$$

The optimal value function $V^*$ is unique and follows the Bellman equation:

$$V^*(s) = \max_a \left[R(s, a) + \gamma \sum_{s'} T(s, a, s') V^*(s')\right]. \tag{3.2}$$

From the bellman equation one can deduce a state-action value function $Q(s, a)$ that satisfies $V^*(s) = \max_a Q(s, a)$. Given this Q function, a policy can be derived as $\pi(s) = \operatorname{argmax}_a Q(s, a)$.

### 3.1.1 Partially observable Markov decision process

Sometimes the agent does not have direct access to the entire state of the environment. In these cases, it is more common to model the problem as a POMDP, which is an extension to the MDP. A POMDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, T, O, R, \gamma)$, where the state space, action space, transition model, reward and discount factor is the same as the MDP, but a POMDP has two additional elements:

- The observation space $\mathcal{O}$, which represents all possible observations that the agent can receive. This can be both discrete and continuous.
- The observation model $O(o|s', a)$, which describes the probability of observing $o \in \mathcal{O}$ in a given state $s'$ after taking an action $a$: $O(o, s', a) = \Pr(o|s', a)$.

In a POMDP, the agent takes an action $a$ from a given state $s$ and the environment transitions to the next state $s'$ according to the transition model $T$. The agent then receives an observation $o$ related to $s'$ and $a$ according to the observation model $O$. After the agent takes an action $a$ from a given state $s$ and the environment transitions to the next state $s'$ according to the transition model $T$, the agent receives an observation $o$. This observation $o$ is drawn according to the observation model $O(o|s', a)$, which specifies the probability of observing the given new state $s'$ and taken action $a$.

Since the state is not observable, policies in a POMDP are no longer described by mapping from states to actions. Instead, the agent must reason about the history of observations and actions. This history can sometimes be summarized in a statistic referred to as a belief state (or belief). A belief state $b$ is a probability distribution such that

$$b(s) = \Pr(s \mid o_{1:t}) \tag{3.3}$$

is the probability of being in state $s$ at time $t$, given observations $o_{1:t} := \{o_1, o_2, ..., o_t\}$ up to time $t$. At each time step $t$, the agent updates its belief using a Bayesian filtering approach given the previous belief and the current observation as follows:

$$b'(s') \propto O(o \mid s', a) \int_{s \in S} T(s' \mid s, a) b(s), \tag{3.4}$$

where $b'$ is the updated belief state after taking action $a$ and receiving observation $o$. By summarizing all relevant information into the current belief state, a POMDP also satisfy the Markov property [21, Ch. 17], ensuring that future states depend only on the current belief state and not on the entire history of actions and observations. This approximation is referred to as a $k$-Markov approximation, where $k$ defines the length of the included history. With a sufficiently long history, the Markov property is assumed to approximately hold, even in a partially observable environment.

Policies are now described as mappings from beliefs to actions. The optimal belief state value function $V^*(b')$ that satisfies the Bellman equation can be formulated as:

$$V^*(b) = \max_a \left[ R(b,a) + \gamma \sum_{o \in \mathcal{O}} \Pr(o \mid b,a) V^*(b') \right] \tag{3.5}$$

where $b'$ is computed using (3.4), and $R(b,a) = \int_{s \in \mathcal{S}} b(s) R(s,a)$ is the expected reward in a belief state $b$. Similarly, as in the MDP one can deduce a belief-action function $Q(b,a)$ that satisfy $V^*(b) = \max_a Q(b,a)$ and the policy a policy can be derived as $\pi(b) = \operatorname{argmax}_a Q(b,a)$.

The observable states in this work are information that sensors on the ego vehicle can provide e.g., distance to intersection, position and speeds of other vehicles. While the unobservable states are the intentions of other drivers that are approaching the same intersection as ego. Chapter 4 will later formulate the POMDP studied in this work.

## 3.2 Reinforcement learning

In some problems, the state transition probabilities or the reward function are unknown. These problems can be addressed using reinforcement learning, where the agent learns how to behave by interacting with the environment [24, Ch. 5]. The data available to an RL agent depends on its current policy, necessitating a balance between exploring the environment and exploiting existing knowledge. Moreover, the reward an agent receives might hinge on a crucial decision made earlier, making it essential to attribute rewards to the correct decisions.

RL algorithms are categorized into two approaches: model-based and model-free [24, Ch. 5]. In model-based methods, the agent first estimates a represen-

tation of the state transition function $T$ and then uses a planning algorithm to find a policy. Conversely, model-free RL algorithms, as the name suggests, do not explicitly construct a model of the environment to determine actions.

Model-free approaches can be further divided into value-based and policy-based techniques. Value-based algorithms, such as $Q$-learning [30], aim to learn the value of each state, thereby implicitly defining a policy. Policy-based techniques [31] directly search for the optimal policy within the policy space, either through policy gradient methods [32] or gradient-free methods like evolutionary optimization. Hybrid techniques, such as actor-critic methods [33], combine both policy and value-based approaches.

RL algorithms typically assume that the environment is modeled as a MDP, where the agent knows the state of the environment. However, in many practical situations, only partial information about the state of the environment is available, which is modeled within the POMDP framework. In such cases, it is common to approximate the state as a belief $b$ [34].

### 3.2.1 Deep Q-learning

Q-learning [30] is a model-free and value-based RL algorithm, where the objective of the agent is to learn the optimal state-action value function $Q^*(s, a)$. This function is defined as the discounted expected return when taking action $a$ from state $s$ and then following the optimal policy $\pi^*$, i.e.,

$$Q^*(s, a) = \max_{\pi} \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R(s_k, a_k) | s_0 = s, a_0 = a, \pi\right]. \tag{3.6}$$

The optimal state-action value function follows the Bellman equation

$$Q^*(s, a) = \mathbb{E}_{s' \sim T(s'|s,a)}\left[R(s, a) + \gamma \max_{a'} Q^*(s', a')\right], \tag{3.7}$$

which recursively defines the $Q$-values of the state-action pairs $(s, a)$. The equation can intuitively be understood by the fact that if $Q^*$ is known, the optimal policy is to select the action $a'$ that maximizes $Q^*(s', a')$.

In the DQN algorithm, a Neural Network (NN) with weights $\theta$ is used as a function approximator of the optimal state-action value function, $Q(s, a; \theta) \approx Q^*(s, a)$ [35]. The weights of the network are adjusted to minimize the temporal difference (TD) error in the Bellman equation, typically with some kind of Stochastic Gradient Descent (SGD) algorithm. Mini-batches with size $M$ of

experiences, $e = (s, a, r, s')$, are drawn from an experience replay memory, and the loss is calculated as

$$L(\theta) = \mathbb{E}_{\mathrm{M}}\Big[(r + \gamma \max_{a'} Q(s', a'; \theta^{-}) - Q(s, a; \theta))^2\Big]. \tag{3.8}$$

Here, $\theta^{-}$ represents the NN parameters of a target network, which is kept fixed for a number of training steps, in order to stabilize the training process. Several of improvements to this standard form of DQN have been proposed and are compared by Hessel et al. [36]. See Paper A for the details of the DQN implementation in this thesis.

# CHAPTER 4

# Modeling Intersection Driving Scenarios

Navigating an intersection is a sequential decision-making problem that can be mathematically modeled using an MDP, as introduced in Section 3.1. This thesis focuses on driving through intersections in the presence of other drivers, emphasizing not only to follow traffic rules but also the ability to adapt to the intentions of other drivers. Given that current sensors cannot directly observe other drivers' intentions, a POMDP is a more suitable framework for formulating this problem.

This chapter explores the modeling of intersection driving scenarios for autonomous vehicles.

## 4.1 POMDP formulation

Effectively modeling the intersection problem is key for developing an optimal decision-making policy. This section outlines each component of the POMDP framework as applied to the intersection problem in this thesis. While specific details of the POMDP may vary between Paper A-D, the general description remains consistent.

### 4.1.1 State space

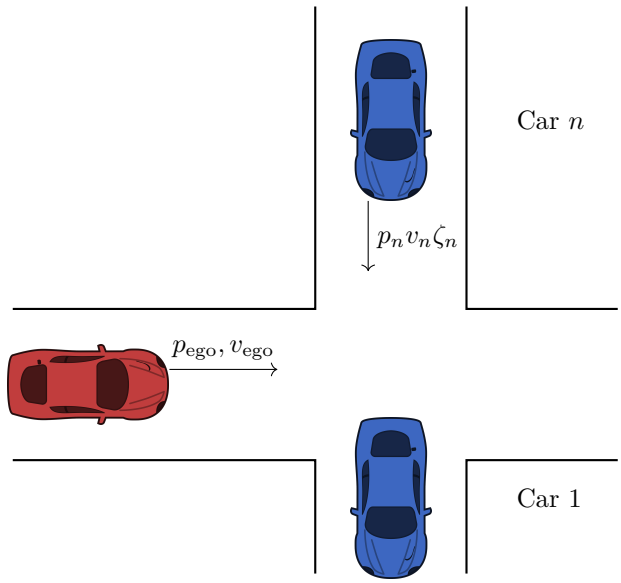



**Figure 4.1:** General description of the states in a simple intersection. The ego vehicle in red is controlled by the agent, while the blue vehicles are the other vehicles crossing the same intersection. Each blue vehicle is described by an index $n$, a position $p_n$, a velocity $v_n$ and hidden intention $\zeta_n$

.

From Section 3.1, the state space contains all the information necessary about the agent and environment to be able to transition from any given state $s$ to the next state $s'$. In the scenario shown in Figure 4.1, the red car on the horizontal lane represents the ego vehicle controlled by the agent while the blue cars on the vertical lane are the other vehicles which the ego vehicle needs to interact with in order to cross the intersection.

A simplified description of the state,

$$s = (p_{\text{ego}}, v_{\text{ego}}, \{p_n, v_n, \zeta_n\}_{n=1}^{N}), \tag{4.1}$$

consists of positions state $p_{\text{ego}}$ and $p_n$, where subscript *ego* and $n$ denotes the ego vehicle and the index of the surrounding vehicle up to $N$ vehicles.

Instead of using a Cartesian coordinate system to describe the position $p_{\text{ego}}$ and $p_n$, relative distance measures are proposed. This way, the state space is generalizable to different intersection designs, e.g., the angle of incidence and the number of crossing points. The velocity of ego $v_{\text{ego}}$ and of all the other traffic participants $v_n$ are also necessary to be able to predict what position they will be in the next state. Finally, the intention of all the other participants $\zeta_n$. As mentioned in Section 1.2, $\zeta_n$ encapsulates information such as stop sign, traffic light or even inattention in to one variable. Paper D shows a comparison between two fully observable MDPs, one with intention and the other one without and the results show that having an intention state reduce number of collisions.

### 4.1.2 Action space

One limitation of deep Q-learning is the requirement for a discrete action space. In various AD studies [37], it is common practice to define the action space in terms of different discretized acceleration requests. To address the challenge of managing a potentially high-dimensional action space with fine discretization or a coarse action space with large steps between accelerations, Paper A proposes using short-term goals as actions: {*'take way', 'yield', 'follow car* $\{1, \ldots, N\}$*'*}.

These short-term goals represent high-level objectives, such as driving through an intersection (take way), stopping at the start of an intersection (yield) or drive behind a specific car (follow car $n$). Each high-level action is translated into a set of parameters that are input into a sliding mode controller in Paper A and a MPC in Paper B, which then generates the appropriate acceleration to control the ego car.

### 4.1.3 Transition model

The transition model is initially unknown, and RL is employed to implicitly learn this model by taking actions in the environment from different states, recording the resulting rewards, and noting the subsequent state transitions. In this work, the environment is a simulator, and the primary objective for the agent is to learn the transition dynamics of other vehicles, which depend on their intentions $\zeta$. These intentions are modeled as predetermined actions governed by an IDM. Using the IDM to guide predetermined actions enhances the

dynamics of vehicle interactions, bringing them closer to real-world scenarios.

### Intelligent Driver Model

The IDM is a widely used car-following model in traffic flow theory and simulation [38]. It describes how drivers adapt their speed and spacing based on the distance to the vehicle ahead. IDM helps in understanding and predicting traffic dynamics, optimizing traffic flow, and developing ADAS functions. In this thesis the IDM is used to model the general behavior of surrounding vehicles. The IDM models a vehicle $n$'s position $p_n$ and velocity $v_n$ as

$$\dot{p}_n = v_n \tag{4.2}$$

$$\dot{v}_n = a_{\max}\Big(1 - \Big(\frac{v_n}{v_n^{\text{desired}}}\Big)^{\delta} - \Big(\frac{d^*(v_n, \Delta v_n)}{d_n}\Big)^2\Big) \tag{4.3}$$

$$\text{with } d^*(v_n, \Delta v_n) = d_0 + v_n T_{\text{gap}} + \frac{v_n \Delta v_n}{2\sqrt{a_{\max}\alpha_b}}$$

where $v_n^{\text{desired}}$ is the desired velocity, $d_0$ is the minimum distance between cars, $T_{\text{gap}}$ is the desired time gap to the vehicle in front, $a_{\max}$ is the maximum vehicle acceleration, $d_n$ is the distance to the vehicle in front, $\Delta v_n = v_n - v_{n-1}$ is the velocity difference between vehicle $n$ and the vehicle directly in front $n-1$, and $\alpha_b$ and $\delta$ are model parameters for comfortable deceleration or acceleration.

The acceleration can be simplified into two terms: an interaction term for when there is a vehicle in front

$$\begin{aligned} a_n^{\text{int}} &= -a_{max}\Big(\frac{d^*(v_n, \Delta v_n)}{d_n}\Big)^2 \\ &= -a_{max}\Big(\frac{d_0 + v_n T_{\text{gap}}}{d_n} + \frac{v_n \Delta v_n}{2\sqrt{a_{\max}\alpha_b} d_n}\Big)^2 \end{aligned} \tag{4.4}$$

and free road term, when there is no leading vehicle

$$a_n^{\text{free}} = a_{\max}\Big(1 - \Big(\frac{v_n}{v_n^{\text{desired}}}\Big)^{\delta}\Big). \tag{4.5}$$

In this thesis, IDM's application ensures realistic simulation of surrounding vehicle behavior, which is crucial for testing and validating the proposed decision-making algorithms for autonomous vehicles.

### 4.1.4 Observation model

The observation space is closely aligned with the state space $\mathcal{S}$, but it includes some added noise and excludes the intention state $\zeta_n$, because current sensors cannot directly detect the intentions of other drivers. The observation

$$o = (p_{\text{ego}}, v_{\text{ego}}, \{\hat{p}_n, \hat{v}_n\}_{n=1}^{N}), \tag{4.6}$$

encompasses all observable elements of the state, detailed in (4.1). The ego vehicle accurately observes its own states, while it observes noisy measurements of the positions $\hat{p}_n$ and velocities $\hat{p}_n$ of the surrounding vehicles. These measurements are given by:

$$\hat{p}_n = p_n + \epsilon_{\text{p}}, \tag{4.7}$$

$$\hat{v}_n = v_n + \epsilon_{\text{v}} \tag{4.8}$$

where, $\epsilon_{\text{p}} \sim \mathcal{N}(0, \sigma_p^2)$ and $\epsilon_{\text{v}} \sim \mathcal{N}(0, \sigma_v^2)$.

### 4.1.5 Reward function

The design of the reward function is pivotal as it determines the value associated with each state, ultimately shaping the driving policy of the agent. A well-crafted reward function is instrumental in guiding the agent towards achieving its objectives effectively. The reward model in this thesis is formulated based on terminal states, including reaching the goal $r^{\text{succ}}$, collision events $r^{\text{fail}}$, timeouts $r^{\text{t.o.}}$, and on non-terminal states $r^{\text{comf}}$ e.g, every step update the agent has not reached a terminal state. These terminal states play a critical role in defining the success or failure of the agent's driving behavior and are accordingly reflected in the reward structure. One example of a reward function used in this thesis, from Paper B, is

$$r_t = \begin{cases} r^{\text{succ}} & \text{on success,} \\ r^{\text{fail}} & \text{on failure,} \\ r^{\text{t.o.}} & \text{on timeout, i.e. } \tau \geq \tau_m, \\ r^{\text{comf}} & \text{on non-terminating updates,} \end{cases} \tag{4.9}$$

where the reward for reaching the goal is typically assigned a relatively high value, such as $r^{\text{succ}} = 1$, while the reward for colliding is assigned a very low

value, such as $r^{\text{fail}} = -1$. These two rewards establish the maximum and minimum possible rewards for a single simulation run. If the defined reward values are too large, the $Q$-values in (3.8) can become large and cause the gradients to grow and potentially lead to instability [39]. Although the absolute values can be chosen arbitrarily, the most important aspect is their relative values to each other. This relative scaling ensures that the agent prioritizes achieving the goal and avoiding collisions appropriately. The other rewards for timeout and non-terminal states are then usually set somewhere between $r^{\text{succ}}$ and $r^{\text{fail}}$, such as $r^{\text{t.o.}} = -0.1$. The reward for non-terminal states $r^{\text{comf}}$ was utilized differently: Paper A used $r^{\text{comf}}$ to account for comfort by assigning a higher negative reward for high acceleration jerk, while Paper B added a negative reward was given if the MPC predicted a collision.

## 4.2 Simulation environment

The simulation environment in this thesis, first introduced in Paper A, places an agent at an intersection tasked with reaching the goal across the intersection while interacting with up to $N$ other cars on the intersecting lane. At the beginning of each episode, up to $N$ vehicles are initialized with initial positions $p_n^0$ distributed along the intersecting lane, starting velocities $v_n^0$ and a deterministic policy that defines their intention $\zeta_n$. Each vehicle, including the ego vehicle, adheres to the IDM (Section 4.1.3) aimed at maintaining specific speeds and safe distances, with the maximum acceleration is capped at $5m/s^2$ to ensure comfort and safety under normal driving conditions. For example, if a vehicle's intention $\zeta_n$ is yield, it would set the IDM for the distance to the object in front $d_n$ to the distance to the start of the intersection, and its velocity $v_{n-1}$ would be set to 0. Alternatively, a vehicle with a take way intention would follow the IDM only considering the vehicle directly in front of it.

During the simulation, whenever a vehicle in the perpendicular lane crosses the intersection, it is removed from the environment. Subsequently, a new vehicle is spawned at the start of that lane at a random time, with new initial values for position $p_n^0$, velocity $v_n^0$, desired velocity $v_n^{\text{desired}}$ and intention $\zeta_n$. This dynamic spawning process ensures that the traffic scenario continually evolves, presenting varying challenges and interactions for the agent navigating the intersection. Each episode continues until a terminal state is reached,

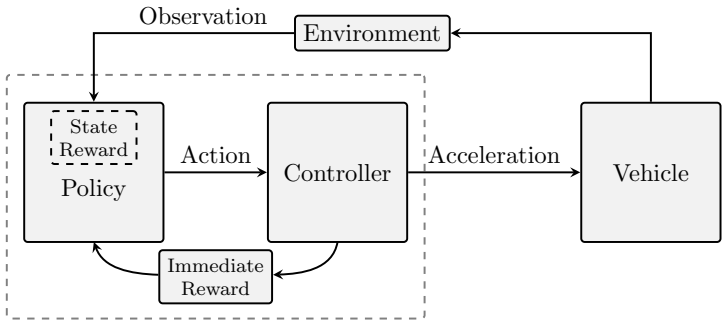



**Figure 4.2:** Representation of the decision-making architecture. Policy represents the DQN that chooses an action that is sent to a Controller that can either be a sliding mode controller or MPC. The Environment represents the simulation environment in which the vehicles operate, but can be replaced by the real world. The State Reward is the reward given by the terminal state ($r^{\text{succ}}, r^{\text{fail}}, r^{\text{t.o.}}$) from the environment and the Immediate Reward $r^{\text{comf}}$ is given by the controller.

which can be reaching the goal, collision, or timeout for Paper A-C and goal, collision, safe stop, or deadlock for Paper D. Rewards are assigned based on a predefined reward function designed to reinforce desired behaviors.

## 4.3 Deep Q-learning approach

To find a driving policy $\pi$ for the POMDP detailed in the previous section, both Paper A and Paper B used deep Q-learning, described Chapter 3.2.1, using a decision-making architecture shown in Figure 4.2. Paper A proposed a network architecture with shared weights and LSTM layer to approximate the Q-function. The shared weight effectively reduces the input space, as the state for each car $n$ can be initially weighted the same as any other car, independent of the order it comes into the neural network. While the LSTM layer has the role of utilizing the previous states to implicitly estimate a hidden state that could possibly encapsulate the intention of other cars.

As mentioned in Chapter 4.1.2, the actions in Paper A are controlled by a sliding mode controller while Paper B uses an MPC to generate a velocity profile for a short time horizon. The sliding mode controller, which is used

interchangeably with the IDM in this thesis, aims to maintain a minimum distance from a target vehicle by controlling acceleration in a manner similar to IDM.

### 4.3.1 Model Predictive Control

MPC is an optimization-based control technique where an Optimal Control Problem (OCP) is repeatedly solved over a receding limited time horizon, starting from the current system state. For every time instance, a mathematical model of the controlled system is used to simulate future states over a finite horizon, while a sequence of control inputs is selected and optimized given an objective cost function. The first element in the sequence of control inputs is then applied to the real system, and a new OCP with an updated state is solved at the next time instance.

MPC faces challenges as a mixed integer problem, where calculating the optimal path for all possible actions is computationally intensive. DQN on the other hand, only handles discrete actions. While DQN cannot guarantee safety, it excels at choosing actions with the highest utility (Q-value). The reward function in DQN can incorporate the predicted outcome from the MPC model, penalizing suboptimal actions. However, if experience shows a better outcome than the model predicts, DQN can opt for an action that leads to a better total reward. Paper B integrates the MPC cost as a negative immediate reward, allowing the DQN to balance this cost with the high-level goal of reaching the target. This enables the DQN to choose actions that are generally beneficial at a high level, even if they may not seem optimal to the MPC.

The agents from Paper A and Paper B are evaluated on two scenarios: a single intersection and a double intersection with two crossing points, both shown in Figure 1.1. The next section presents the experimental results from Paper A and Paper B for both intersections.

## 4.4 Results from simulation

The results from Table 4.1 show that the proposed DQN agent from Paper A found a policy that successfully crossed a single intersection 96.1% of the time, with 2.8% resulting in collisions and 1.1% resulting in timeouts. When comparing agents trained with and without an LSTM layer, those with the

LSTM succeeded in 3 out of 4 attempts where those without it would fail. This demonstrates that deep Q-learning has great potential for creating decision-making agents capable of navigating intersections. OA key contribution of this thesis is the proposal of network architectures specifically designed for this problem. In Paper A, an architecture employing shared weights is introduced, and the results shown in Figure 4.3 reveal that utilizing a network with shared weights for processing information about observed vehicles notably enhances convergence speed. The combination of shared weights with other improvements, such as dropout and experience replay, is crucial for achieving these performance gains.


**Figure 4.3:** The figure shows that the success rate for a network with shared weights (brown line) converge faster than the fully connected network structure which do not share weights (turquoise line).

As mentioned in Section 4.1.2, the controller from Paper A could only consider one car at a time e.g., follow car $n$. If the agent needs to achieve a velocity between two discrete actions, it can do so by switching between them in a manner similar to pulse-width modulation. However, this placed a heavy burden on the DQN compensate by frequently switching actions. In contrast, in Paper B, the MPC showed significant improvement in handling more complex intersections. For instance, in the double intersection, the MPC agent succeeded 95.2% of the time with a collision rate of 3.6%, compared to the sliding mode controller, which only succeeded 90.9% of the time with a collision rate of 8.3%.

| Controller | Success Rate | | Collision Rate | |
|---|---|---|---|---|
| | Single | Double | Single | Double |
| SM (Paper A) | 96.1% | 90.9% | 2.8% | 8.3% |
| MPC (Paper B) | 97.3% | 95.2% | 1.2% | 3.6% |

**Table 4.1:** Average success rates and collision rates for a fully trained DQN agent driving through a single and double intersections. If the agent failed to reach the goal or collide within a given time, the terminal state was classified as a timeout.

## 4.5 Discussion

Observing the behavior of a fully trained agent from Paper A and Paper B provides the insight that the path of the ego vehicle can be segmented into four zones, illustrated in Figure 4.4. Starting from the right, Zone 0 represents the *safe zone*, where the ego vehicle is out of danger and can resume nominal driving. Zone 1 is the *conflict zone*, where a collision with another vehicle is possible. Zone 2, the *critical decision zone*, is the final opportunity for the vehicle to either stop or proceed through the intersection. The size of zone 2 is determined by the minimum distance required for the vehicle to come to a complete stop before entering the conflict zone, ensuring sufficient time for safe decision-making. Lastly, Zone 3, the *information gathering zone*, is situated furthest from the intersection. Here, the agent can observe how other vehicles behave over time to estimate their intentions.

The goal is to reach Zone 0. To achieve this, the agent aims to minimize the time spent in Zone 1 if there is a chance of intersection with another car. Our actions are formulated as short-term goals, designed for comfortable use with lower acceleration rates. The size of Zone 2 depends on the vehicle's current speed, which is influenced by its behavior in Zone 3.

Now, two conflicting strategies emerge: to minimize time in Zone 1, the agent desires a high speed entering the intersection. However, it also seeks a low speed to reduce the size of Zone 2 and the critical decision period. If the intentions of other vehicles are known, the stochasticity in Zone 1 would be eliminated, transforming the problem into a scheduling task aimed at creating a velocity profile that minimizes the time required to cross. However, since the intentions of other vehicles are inherently stochastic, the next chapter offers a promising approach by accounting for this uncertainty and optimizing

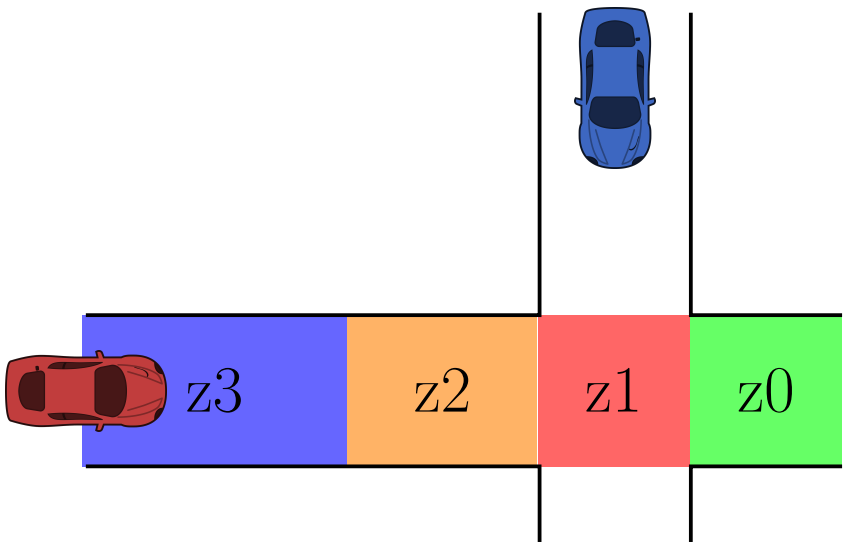



**Figure 4.4:** Intersection scenario divided into zones describing what is required of the decision maker in different zones

decision-making in dynamic traffic scenarios.

# CHAPTER 5

## Accounting for the uncertainty

The previous chapter formulated the POMDP and introduced how deep Q-learning methods can be utilized to create decision-making agents capable of navigating intersections. A significant advantage of RL methods is their scalability to different scenarios through appropriate training. However, a drawback of deep Q-learning methods, is the use of neural networks, which provide a black-box solution without indicating any confidence or uncertainty in their decisions [40].

This chapter presents two approaches to estimating the uncertainty: Paper C addresses the uncertainty in the output of the DQN, and Paper D addresses the uncertainty in the intention estimation that is fed as an input to the DQN. To demonstrate the effect of accounting for uncertainty, the results include two types of experiments. In the first type, the trained agent is evaluated on scenarios within the training set. In the second type, the agent is evaluated on scenarios outside the training set. Both types of experiments are conducted using the same simulation environment described in Section 4.2.

## 5.1 Estimating the uncertainty of the Q-value

To estimate the uncertainty in the $Q$-value of the trained agent, Paper C employed statistical bootstrapping to train an ensemble of neural networks on different subsets of the available experience. This ensemble provides a distribution over the estimated $Q$-values for any input. A better Bayesian posterior is obtained by adding different Randomized Prior Functions (RPF) to each ensemble member [41]. The $Q$-values of each ensemble member $k$ is then calculated as the sum of two neural networks, $f$ and $p$, with the same architecture but different values on the weights $\theta_k$ and $\hat{\theta}_k$, i.e.,

$$Q_k(s,a) = f(s,a;\theta_k) + \beta p(s,a;\hat{\theta}_k). \tag{5.1}$$

Here, the weights $\theta_k$ of network $f$ are trainable, and the weights $\hat{\theta}_k$ of the prior network $p$ are fixed to the randomly initialized values. A parameter $\beta$ scales the importance of the networks. With the two networks, the loss function in Eq. 3.8 becomes

$$\begin{aligned} L(\theta_k) = \mathbb{E}_M \Big[ & (r + \gamma \max_{a'} (f_{\theta_k^-} + \beta p_{\hat{\theta}_k})(s',a') \\ & - (f_{\theta_k} + \beta p_{\hat{\theta}_k})(s,a))^2 \Big]. \end{aligned} \tag{5.2}$$

The training process of an ensemble RPF is described by Algorithm 1. An ensemble of $K$ DQN are first initialized randomly. Each ensemble member is also assigned a separate experience replay memory buffer $m_k$. For each new episode, a random ensemble member $\nu$ is selected and used to take greedy actions throughout the episode, which corresponds to an approximate Thompson sampling approach to the exploration vs. exploration dilemma. Each new experience $e = (s_i, a_i, r_i, s_{i+1})$ is then added to the separate replay buffers $m_k$ with probability $p_{\text{add}}$. The trainable weights of each ensemble member are then updated by uniformly sample a mini-batch $M$ of experiences and using SGD.

The agent's uncertainty in choosing different actions can be defined as the coefficient of variation $c_{\text{v}}(s,a)$ of the $Q$-values of the ensemble members [42]. A threshold $c_{\text{v}}(s,a)$ can then be used to determine whether an action has an acceptable level of uncertainty.

A high uncertainty, where $c_{\text{v}}(s,a) > c_{\text{v}}^{\text{safe}}$, indicates that $(s,a)$ is outside the training distribution. When the agent is fully trained, the policy $\pi$ chooses

**Algorithm 1** Ensemble RPF training process

```
for k ← 1 to K do
    Initialize θ_k and θ̂_k randomly
    m_k ← {}
i ← 0
while networks not converged do
    s_i ← initial random state
    ν ~ U{1, K}
    while episode not finished do
        a_i ← argmax_a Q_ν(s_i, a)
        s_{i+1}, r_i ← StepEnvironment(s_i, a_i)
        for k ← 1 to K do
            if p ~ U(0, 1) < p_add then
                m_k ← m_k ∪ {(s_i, a_i, r_i, s_{i+1})}
            M ← sample mini-batch from m_k
            update θ_k with SGD and loss L(θ_k)
        i ← i + 1
```

actions by maximizing the mean of the $Q$-values of the ensemble members, with the restriction $c_{\mathrm{v}}(s,a) < c_{\mathrm{v}}^{\mathrm{safe}}$, i.e.,

$$\begin{aligned} \pi = \operatorname*{argmax}_a \frac{1}{K} \sum_{k=1}^{K} Q_k(s,a), \\ \text{s.t.} \quad c_{\mathrm{v}}(s,a) < c_{\mathrm{v}}^{\mathrm{safe}}. \end{aligned} \tag{5.3}$$

If no action within the action space satisfies the restriction, a predefined backup action $a_{\mathrm{safe}}$ is chosen instead. This $a_{\mathrm{safe}}$ can be something like a collision avoidance measure, such as emergency braking.

### 5.1.1 Results for scenarios within the training set

This section shows the results of the ensemble RPF method compared to the DQN method used in Paper A, but without the LSTM layer. The ensemble RPF method outperforms the DQN method when the agents are tested on scenarios that are similar to the training scenarios. When the fully trained ensemble RPF agent is exposed to situations that are outside the training

distribution, the agent indicates a high uncertainty and chooses safe actions, whereas the DQN agent sometimes collides with other vehicles.

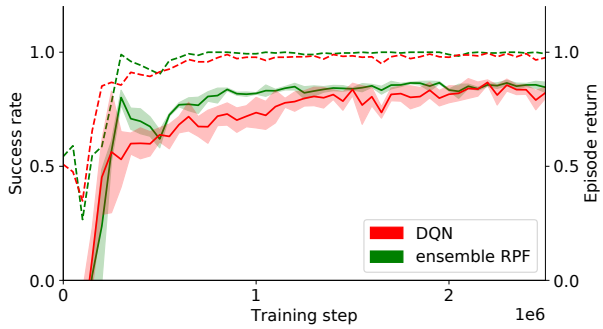



**Figure 5.1:** Proportion of test episodes where the ego vehicle reached its goal (dashed), and episode return (solid), over training steps for the ensemble RPF and DQN methods. The shaded areas show the standard deviation for 5 random seeds.

The average return and the average proportion of episodes where the ego vehicle reached the goal, as a function of number of training steps, is shown in Figure 5.1, for the test episodes. The figure also shows the standard deviation for 5 random seeds, which generates different sets of initial parameters of the networks and different training episodes, whereas the test episodes are kept fixed. The results show that the ensemble RPF method both learns faster, yields a higher return, and causes fewer collisions than the DQN method.

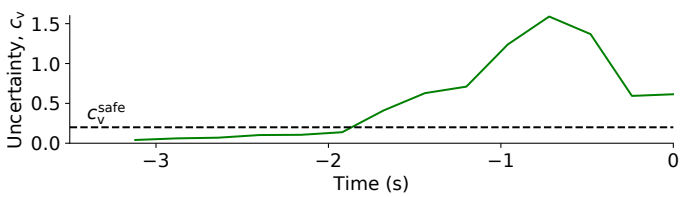
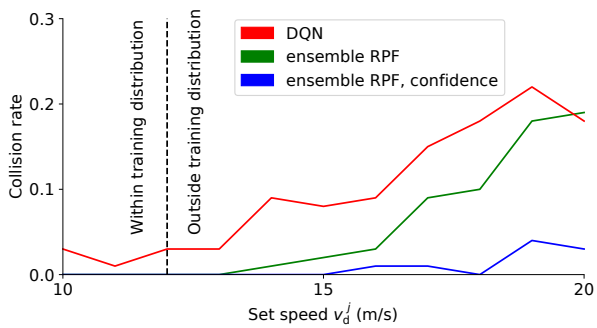


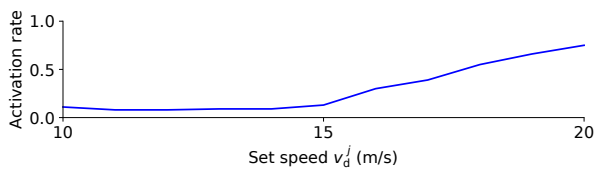

**Figure 5.2:** Uncertainty $c_v$ during the time steps before one of the collisions in the test episodes, within the training distribution. The collision occurs at $t = 0$ s.

Upon analyzing the scenarios with collisions using the DQN agent, it was observed that in one particular example, the agent fails to brake sufficiently in zone 3 (from Figure 4.4) and takes an incorrect action in zone 2, resulting in a collision in zone 1. In contrast, in the same scenario with the ensemble RPF agent, Figure 5.2 show that the estimated uncertainty increases significantly during the time before the collision (zone 2), when the incorrect actions was taken. By applying the confidence criterion, the agent, aware of high uncertainty, brakes early enough in zone 2, thus avoiding collisions. This confidence criterion was applied to all test episodes, effectively eliminating all collisions for scenarios within the training set.

### 5.1.2 Results for scenarios outside the training set




**(a)** Proportion of collisions.




**(b)** Proportion of episodes where $a_{\text{safe}}$ was used at least once.

**Figure 5.3:** Performance of the ensemble RPF agent, with and without the confidence criterion, and the DQN agent, in test episodes with different velocities for the surrounding vehicles.

To evaluate the agents' ability to detect unseen situations, a fully trained

ensemble RPF agent was tested in scenarios outside the training distribution. The same testing scenarios as in the previous section were used, with the exception that the speed of the surrounding vehicles was set to a single deterministic value, which was varied during different runs in the range $v_n \in [10, 20]$ m/s. The proportion of collisions as a function of set speed of the surrounding vehicles is shown in Figure 5.3, along with the proportion of episodes where the confidence criterion was violated at least once. The figure demonstrates that by accounting for the uncertainty of the $Q$-value and the use of the confidence criterion some collisions can be avoided.

## 5.2 Estimating the uncertainty of the intention state

The ensemble RPF agent avoided collisions by not taking a bad action in zone 2, by accounting for the uncertainty of the Q-value estimate. However, can further reductions in collisions be achieved by accounting for uncertainty in the input? In Paper D, the uncertainty regarding the intention state $\zeta_n$ is managed using a belief state represented as a probability distribution over possible states. This distribution, referred to as intention distributions, is generated using a particle filter. Based on these distributions, two methods were proposed: QMDP-IE and QID.

QMDP-IE is derived from the QMDP algorithm [29]. This derivation leverages the assumption that the intention state $\zeta_n$ can be determined after the ego vehicle has crossed the intersection analyzing the actions taken by the other vehicle, i.e., the intention can be determined by post-data analysis. This assumption allows the QMDP algorithm to blend elements of POMDPs and MDPs. The QMDP algorithm is comprised by the following steps:

1. Oracle DQN Training: The weights $\theta_{\text{MDP}}$ used in $Q(a, s; \theta_{\text{MDP}})$, are trained using the true state $s$, which includes the true intention state $\zeta_n$.

2. Expected Q-value Computation: QMDP calculates the Q-value for each belief state, which accounts for uncertainty in the true state. It then computes the expected Q-value for each action by averaging the Q-values of the potential states, weighted by their probabilities in the belief state.

QMDP-IE modifies the QMDP approach:

1. Utilization of Oracle DQN: QMDP-IE uses the same Q-values $Q(a, s; \theta_{\text{MDP}})$ as calculated by the Oracle DQN.

2. Intention State Estimation $D^{\text{IE}}(b)$: Instead of considering the full distribution of potential states (belief state $b$), QMDP-IE estimates a single intention state. This estimation $D^{\text{IE}}(b)$ is done using a threshold $\zeta_{\text{threshold}}$ on the intention distribution

$$D^{\text{IE}}(b) = \zeta_n^{\text{IE}} = \begin{cases} \text{yield} & \text{if } \zeta_n^{\text{yd}} > \zeta_{\text{threshold}}, \\ \text{take way} & \text{otherwise.} \end{cases} \tag{5.4}$$

   This $\zeta_n^{\text{IE}}$, together with the observable state $o$, is then passed as to the NN to estimate the Q-value.

The threshold value $\zeta_{\text{threshold}}$ in the QMDP-IE method is a critical parameter that directly influences the agent's behavior, ranging from aggressive to passive. The agent is tested with different $\zeta_{\text{threshold}}$ values, and the outcomes are observed in terms of success rate and success time shown in 5.4. A lower threshold value ($\zeta_{\text{threshold}} = 0,5$) means the agent relies on the most likely estimation of the intention state, resulting in aggressive behavior. This can lead to quicker success times but might increase the risk of collisions or other negative outcomes. A higher threshold values ($\zeta_{\text{threshold}} = 0.9$) makes the agent more cautious, potentially leading to safer but slower success times. In summary, while both QMDP and QMDP-IE rely on the Oracle DQN to predict Q-values, QMDP-IE reduces the complexity of decision-making by focusing on a single estimated intention state rather than a full belief state distribution reducing the computational complexity.

In the QID method, the intention state $\zeta_n^{\text{ID}}$ is derived from the belief state $b$ using an approximator $D^{\text{ID}}(b)$. The derived intention state $\zeta_n^{\text{ID}}$ is obtained as the marginal distribution of intention from the set of particles $\{x_m\}_{m=1}^{M}$:

$$D^{\text{ID}}(b) = \zeta_n^{\text{ID}} = [\zeta_{\text{tw}}^{\text{ID}}\ \zeta_{\text{yd}}^{\text{ID}}] = \sum_{m=1}^{M} w_{n[m]} [x(\zeta)]_{n[m]}, \tag{5.5}$$

where each particle $x_m$ represents a possible state, and each particle has an associated weight $w_{n[m]}$, indicating its probability out of $M$ number of total particles. The intention state used by the particle $[x(\zeta)]_{n[m]}$ is specified using a one-hot vector, which represents the intention state in a binary format where

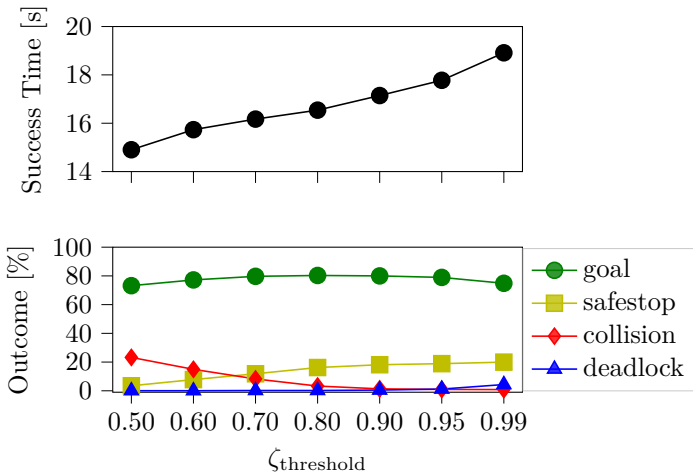



**Figure 5.4:** The success time and outcome percentages of QMDP-IE for different values of $\zeta_{\text{threshold}}$ values. The top figure show that increasing $\zeta_{\text{threshold}}$ result in slower success times, whereas the lower figure indicates that higher $\zeta_{\text{threshold}}$ reduces the collision rate while simultaneously increasing the safe stop rate.

one element is '1' (indicating the active intention) and the others are '0'. The intention state for "take-way" is then $\zeta^{\text{ID}}_{\text{tw}} \in [0, 1]$ and consequently, "yield" becomes $\zeta^{\text{ID}}_{\text{yd}} = 1 - \zeta^{\text{ID}}_{\text{tw}}$. This process allows the QID method to derive a clear and actionable intention state from a distribution of possible states.

Both QMDP-IE and QID were evaluated on the intersection scenario introduced in Section 4.2, specifically designed to include at least one car on the intersecting lane with the same time to intersection as the ego vehicle. This scenario increases the likelihood of a collision if the intentions of the other vehicles are not accurately considered, thereby encouraging more safe stops.

### 5.2.1 Results within the training set

This section presents the experimental results, starting with a comparison of the results of QMDP-IE and QID. QMDP is used as a baseline algorithm for QMDP-IE, while QPF, an algorithm trained on the whole belief state $b$ is used

**Table 5.1:** Result summary for scenarios within the training set

| Experiments | Goal reached | Safe stop | Collision | Deadlock | Success Time |
|---|---|---|---|---|---|
| | % | % | % | % | s |
| Oracle DQN | 84.50 | 14.45 | 1.05 | 0.00 | 15.49 |
| QMDP-IE | 80.00 | 18.15 | 1.35 | 0.50 | 17.14 |
| QMDP | 71.95 | 15.05 | 5.70 | 7.30 | 20.56 |
| QID | 85.60 | 10.40 | 3.70 | 0.30 | 16.64 |
| QPF | 63.50 | 21.80 | 12.15 | 2.55 | 16.61 |

as the baseline for QID. Unlike QMDP-IE, the QID algorithm does not require access to the true intent during training.

As shown in Table 5.1, both QMDP-IE and QID beat their respective baseline algorithm in both higher goal reached and lowest collision percentage. QMDP-IE collision rate 1.35 % was very close to the oracle DQN 1.05 % making it as good as knowing the intention in these test cases. QID experienced more collisions than QMDP-IE but fewer than QMDP. This indicates that while training on ground truth is preferable, QID is a suitable alternative when ground truth data is not available.

### 5.2.2 Results for scenarios outside the training set

To investigate how the algorithms perform when exposed to behaviors they are not trained for, a scenario is introduced where a car is allowed to overtake another car in front, thereby violating the assumption of the IDM. In the previous experimental scenario, the order of the other cars was structured such that if one car yielded, all following cars would stop behind it. This makes it possible to collide with a car that started behind a yielding car, but it also creates larger gap in between cars, that was not possible before.

In this overtake scenario, both QID and QPF exhibit relatively low collision rates at 2.53 % and 3.7 % respectively. QMDP-IE and QMDP show slightly higher collision rates of 4.6 % and 5.95 % respectively, which is higher than QID and QPF and slightly higher than their own performance in the trained scenarios, detailed in Table 5.1. Because QMDP-IE uses the network from the Oracle DQN, it performs well when the intention is within the training

**Table 5.2:** Results with an overtake agent

| Experiments | Goal reached % | Safe stop % | Collision % | Deadlock % | Time s |
|---|---|---|---|---|---|
| Oracle DQN | 89.80 | 0.15 | 10.05 | 0.00 | 16.01 |
| QMDP-IE | 94.55 | 0.20 | 4.60 | 0.65 | 16.62 |
| QMDP | 79.70 | 10.20 | 5.95 | 4.15 | 19.84 |
| QID | 96.89 | 0.11 | 2.53 | 0.47 | 16.38 |
| QPF | 96.15 | 0.10 | 3.70 | 0.05 | 17.62 |

set, while QID and QPF was trained on the uncertainty making the policy more cautious in zone 3 which lead to an opening in zone 2 leading to the goal. This suggests that QID is slightly better at taking passive actions in zone 3, which can change the intention distribution and reduce the uncertainty for the next state. This makes QID more effective at handling scenarios outside the training set than QMDP-IE.

Overall, QID proves to be better than QMDP-IE in scenarios outside the training set, offering more robustness to untrained behaviors. Meanwhile, QMDP-IE provides a structured approach with moderate collision rates and higher computational efficiency, performing well when the intentions are within the training set.

# CHAPTER 6

## Generalizing over different scenarios

In the previous chapter, the experiments demonstrated how an agent can utilize the uncertainty measurement to reduce the number of collisions. This chapter explores how to identify when an agent is acting in a scenario outside its training set.

Let's consider that a company is designing Level 5 AD agents. As mentioned in Section 1.1, L5 requires the AV to be able to drive anywhere. The company has already designed two RL agents that have learned to drive at L4 in the USA and UK. Now, the company wants to deploy a new RL agent in India. Though all the RL agents are concerned with the same task, i.e. driving, the models encompassing driver behaviors, traffic rules, signs, etc., can differ for each. For example, UK and India have left-handed traffic, while the USA has right-handed traffic. However, learning a new controller specifically for every new geographic location is computationally expensive and time-consuming, as both data collection and learning take time. Thus, the company might use the models learned for UK and USA, to estimate the model for India, and use it further to build a new RL agent. Hence, being able to transfer the source models to the target environment allows the company to use existing knowledge to build a new agent faster while using less resources. This problem

**Algorithm 2** Maximum Likelihood Estimation for Model-based Transfer RL (MLEMTRL)

1: **Input:** weights $\boldsymbol{w}^0$, $m$ source MDPs $\mathcal{M}_s$, data $D_0$, discount factor $\gamma$, iterations $T$.
2: **for** $t = 0, \ldots, T$ **do**
3: // Stage 1: Model Estimation //
4: $\boldsymbol{w}^{t+1} \leftarrow$ Optimiser$(\log \mathbb{P}(D_t \mid \Sigma_{i=1}^m w_i \mu_i), \boldsymbol{w}^t)$
5: Estimate the MDP: $\mu^{t+1} = \sum_{i=1}^m w_i \mu_i$
6: // Stage 2: Model-based Planning //
7: Compute the policy: $\pi^{t+1} \in \arg\max_{\pi} V^{\pi}_{\mu^{t+1}}$
8: // Control //
9: Observe $s_{t+1}, r_{t+1} \sim \mu^*(s_t, a_t), a_t \sim \pi^{t+1}(s_t)$
10: Update the dataset $D_{t+1} = D_t \cup \{s_t, a_t, s_{t+1}, r_{t+1}\}$
11: **return** An estimated MDP model $\mu^T$ and a policy $\pi^T$

of model transfer from source models to a target environment to plan efficiently is address in Paper E.

## 6.1 Approach

In this example, the driving model for each country is defined by its own MDP. A set of source MDPs $\mathcal{M}_s \triangleq \{\mu_i\}_{i=1}^m$ is used to create a convex hull of MDPs $\mathcal{C}(\mathcal{M}_s)$, where each MDP $\mu_i$ acts as a boundary. The proposed approach, Maximum Likelihood Estimation for Model-based Transfer RL (MLEMTRL), involves using model transfer RL to determine the unknown target MDPs $\mu^* \in \mathcal{C}(\mathcal{M}_s)$ position within this convex hull of MDPs and then find an optimal policy $\pi^*$ for that MDP. This method relies on having a diverse set of MDPs to effectively span the convex hull of models, thereby facilitating the discovery of optimal policies between MDPs. For example, downtown driving behavior in one country may closely resemble that in another. For the case when $\mu^*$ is close but outside $\mathcal{C}(\mathcal{M}_s)$, see Paper E.

The MLEMTRL algorithm, outlined in Algorithm 2, consists of a *model estimation* stage and a *planning* stage.

**Model estimation:** This stage estimates a MDP $\hat{\mu}$ from a compact subset

of $m$ source MDPs, $\mu' \subset \mathcal{M}_s$, utilizing data from experience $D_t$ gathered by taking actions in the environment. Let's define the convex hull as:
$\mathcal{C}(\mathcal{M}_s) \triangleq \{\mu_1 w_1 + \ldots + \mu_m w_m \,|\, \mu_i \in \mathcal{M}_s, w_i \geq 0, i = 1, \ldots, m, \sum_{i=1}^{m} w_i = 1\}$.
Then, the corresponding MLE problem with the corresponding likelihood function is

$$\hat{\mu} \in \underset{\mu' \in \mathcal{C}(\mathcal{M}_s)}{\arg\max} \; \log \mathbb{P}(D_t \,|\, \mu'), D_t \sim \mu^*. \tag{6.1}$$

Since $\mathcal{C}(\mathcal{M}_s)$ induces a compact subset of model parameters $\mathcal{M}' \subset \mathcal{M}$, (6.1) leads to a constrained maximum likelihood estimation problem [43]. It implies that if the parameter corresponding to the target MDP is in $\mathcal{M}'$, it can be correctly identified. For continuous state-action MDPs, a linear-Gaussian likelihood is used. In this context, let $d_s$ be the dimensionality of the state-space, $\boldsymbol{s} \in \mathbb{R}^{d_s}$ and $d_a$ be the dimensionality of the action-space. Then, the mean function $\mathbf{M}$ is a $\mathbb{R}^{d_s} \times \mathbb{R}^{d_a + d_s}$ matrix. The mean visitation count to the successor state $\boldsymbol{s}'_t$ when an action $\boldsymbol{a}_t$ is taken at state $\boldsymbol{s}_t$ is given by $\mathbf{M}(\boldsymbol{a}_t, \boldsymbol{s}_t)$. The corresponding covariance matrix, denoted by $\mathbf{S}$, is of size $\mathbb{R}^{d_s} \times \mathbb{R}^{d_s}$. The log-likelihood is then expressed as follows:

$$\log \mathbb{P}(D_t \,|\, \mathbf{M}, \mathbf{S}) = \log \prod_{i=1}^{t} \frac{\exp\Big(-\frac{1}{2}\boldsymbol{v}^\top \mathbf{S}^{-1} \boldsymbol{v}\Big)}{(2\pi)^{d_s/2} |\mathbf{S}|^{1/2}},$$
$$\text{where } \boldsymbol{v} = \boldsymbol{s}'_i - \mathbf{M}(\boldsymbol{a}_i, \boldsymbol{s}_i).$$

As the optimization problem involves weighing multiple source models together, a weight vector $\boldsymbol{w} \in [0, 1]^m$ is introduced, with the usual property that $\boldsymbol{w}$ sum to 1.

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \log \mathbb{P}(D_t \,|\, \Sigma_{i=1}^{m} w_i \mu_i), D_t \sim \mu^*, \mu_i \in \mathcal{M}_s, \\ \text{s.t.} \quad & \sum_{i=1}^{m} w_i = 1, w_i \geq 0. \end{aligned} \tag{6.2}$$

Because of the constraint on $\boldsymbol{w}$, this is a constrained nonlinear optimization problem. Any optimizer algorithm, denoted by Optimizer, can be used for this purpose. When an appropriate model $\mu^{t+1}$ has been identified at time step $t$, the next stage of the algorithm involves model-based planning in the estimated MDP.

**Planning:** This stage computes the policy $\pi^{t+1}$ that maximizes the expected reward given the value function $V^{\pi}_{\mu^{t+1}}$ based on $\mu^{t+1}$. After computing $\pi^{t+1}$,

the agent executes actions in the environment to acquire new experiences $D_{t+1}$. These steps are repeated $T$ times before producing a final estimated MDP model $\mu^T$ and a policy $\pi^T$ is given.

Given the model, $\mu^t$ and the associated reward function $\mathcal{R}$, the optimal value function of $\mu^t$ can be computed iteratively as [21]:

$$V^*_{\mu^t}(s) = \max_a \sum_{s'} \mathcal{T}^a_{s,s'} \Big( \mathcal{R}(s,a) + \gamma V^*_{\mu^t}(s') \Big). \tag{6.3}$$

The fixed-point solution to Eq. (6.3) is the optimal value function, where $\mathcal{T}$ is the transition function introduced in Section. 4.1.3. When the optimal value function has been obtained, one can simply select the action maximizing the action-value function. Let $\pi^{t+1}$ be the policy selecting the maximizing action for every state, then $\pi^{t+1}$ is the policy the model-based planner will use at time step $t+1$.

## 6.2 Experiments and results

The proposed algorithm is evaluated on 10 different tasks in Deepmind Control Suite [44], and the results for two Linear Quadratic Regulator(LQR) tasks, *dm_LQR_2_1* and *dm_LQR_6_2*, are presented. These environments have continuous states and actions, where the tasks involve controlling a two-joint, one-actuator system and a six-joint, two-actuator system toward the center of the platform, respectively. They feature unbounded control inputs and rewards, with state spaces $s \in \mathbb{R}^4$ and $s \in \mathbb{R}^{12}$, respectively. In the Deepmind Control Suite, each task varies by seed, which determines the stiffness of the joints.

The performance is compared to baseline algorithms such as a posterior sampling RL method (PSRL), multi-task soft-actor critic (MT-SAC) [45], [46] and a modified MT-SAC-TRL that uses data from the novel task during learning. In PSRL, a new model is sampled from the prior at every round, learning in the target MDP from scratch. The aim of this experiment is to identify improvements in learning speed, jumpstart and asymptotic improvements. *Learning Speed Improvement:* A learning speed improvement would be indicated by the algorithm reaching its asymptotic convergence with less data. *Jumpstart Improvement:* A jumpstart improvement can be verified by the behavior of the algorithm during the early learning process. In particular, if

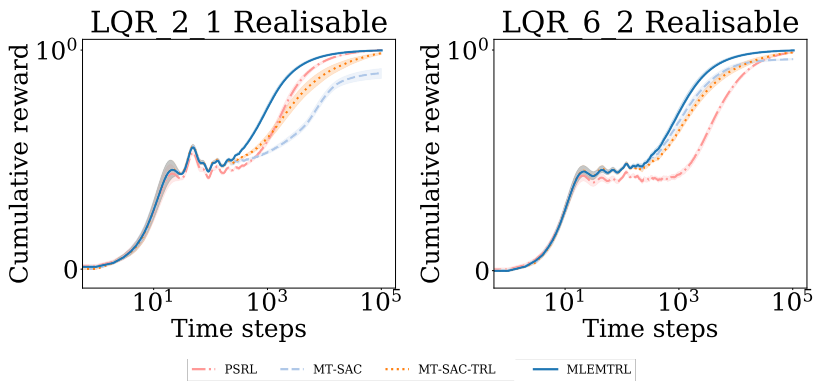



**Figure 6.1:** The average cumulative reward at every time step computed for two LQR tasks from Deepmind control suit. The shaded regions represent the standard error of the average cumulative reward at the time step.

the algorithm starts at a better solution than the baseline, or has a simpler optimization surface, it may more rapidly approach better solutions with much less data.*Asymptotic Improvement:* An asymptotic improvement would mean the algorithm converges asymptotically to a superior solution to that one of the baseline.

The results experiment are shown in Figure 6.1, where the performance metric is the average cumulative reward at each time step over $10^5$ time steps, with the shaded region representing the standard deviation. In these experiments, MLEMTRL demonstrates a clear advantage over all baselines in terms of learning speed improvement and asymptotic performance when compared to MT-SAC, but shows negligible differences in jumpstart improvements.

This shows that MLEMTRL is able to construct a model for an RL agent using a set of source models, provided that the target model is sufficiently similar to the source models.

# CHAPTER 7

# Discussion

Chapter 4, 5 and 6 introduce different methods to create a decision-making agent for navigating through intersections using deep Q-learning, while considering the uncertainty of its own decisions and the prediction of the other drivers intentions. This chapter highlights some differences and synergies of the different methods.

## 7.1 Guaranteeing safety

Safety is the most important factor of a AD system, but since RL is a data driven approach it is very difficult to guarantee safety in the same way e.g., a MPC can. Safety validation in a data driven approach can be very costly as it is usually done by driving many miles and showing statistical confidence in the safety metrics [47], [48], e.g., number of interventions. That is why this section will clarify where the work in this thesis would fit in by first defining a system architecture for AD, its modules and a short summary of the ISO 26262 standard. Then, place itself within the system and motivate how safety can be guarantied.

The architecture of an autonomous driving system can be divided into

perception, planning and control [5], [49]. The perception module is responsible for sensing and mapping the environment with the use of sensors such as LIDARs, cameras, radars etc. The raw data from the sensors are then processed though various sensor fusion techniques to generate a representation of the environment, e.g., position, velocity of other traffic participants while also describing the road such as width and distance to the next intersection. This information is then used by the planner to create a driving strategy of how to transverse through the world. However, the information from the sensors are often noisy, with false positives and false negatives making it difficult for the planner.

Tactical planning can be divided into three categories, the proactive, active and reactive [50]. A proactive module would be something like a precautionary safety module that interprets the information about the environment and create constraints that is sent to the active planner, like driveable area, allowed speeds and actions [51]. These constraints are generated from a set of safety goals and rules, making this the first layer of protection that can ensure safety. The role of the active planner is to take this sets of allowed actions and prescribe the behavior of the vehicle through decisions such as drive, yield or stop. The goal of these high level decisions is to optimize metrics such as comfort, fuel consumption and time to goal. These decisions are then sent to a motion planner that generates a safe dynamically feasible path for the vehicle for a shorter planning horizon of around 0.1s. At the same time, a reactive, collision avoidance, module make sure that the chosen decision and path does lead to any collisions [52]. Unlike the decision maker, the collision avoidance module main goal is to identify imminent danger [53] and therefore has access to more aggressive actions like emergency braking to ensure safety.

In the industry today the main standard for functional safety in motorized vehicles is the ISO 26262 standard, titled "Road vehicles – Functional safety" [54]. It uses a Automotive Safety Integrity Level (ASIL) to classify the inherent safety risk in an automotive system and the functions or modules of such a system. The ASIL classification is used to express the level of risk reduction required to prevent a specific hazard, from ASIL D to ASIL A. ASIL D represents the highest hazard level and ASIL A the lowest. There is a level with no safety relevance and only standard Quality Management processes are required, this level is referred to as QM.

Although safety is the most important requirement for enabling autonomous

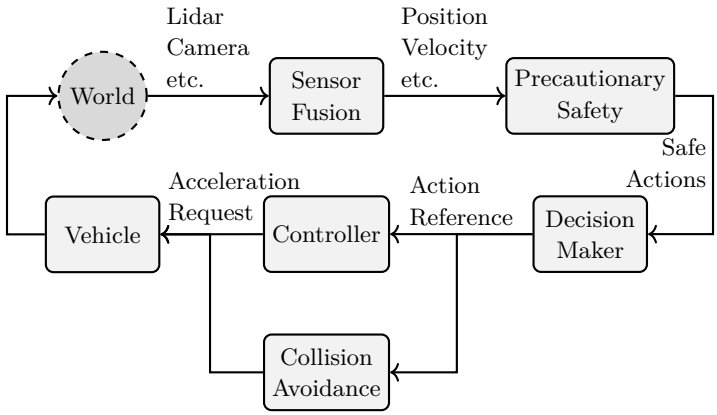



**Figure 7.1:** Representation of a proposed the system architecture for AVs.

driving, the work in this paper does not make any safety guarantees. Instead, it is proposed that the decision-making algorithms presented in this paper be used in the system architecture shown in Figure 7.1. This approach allows higher ASIL to be applied to the precautionary safety and collision avoidance modules, while the decision-making algorithms focus primarily on comfort. As a result, the ASIL classification for the decision-making components could be at lower levels, potentially even classified as QM in the best case.

## 7.2 Designing the reward function and terminal states

The reward function introduced in Chapter 4.1.5 is a crucial component that significantly influences the behavior and performance of a reinforcement learning agent. By carefully designing and tweaking the reward function in (4.9), the agent can be guided towards desirable behaviors and optimize its decision-making policy.

The results from Chapter 4 and 5 show a collision rate of $1 - 3\%$, which may seem high for AVs. However, within the proposed system architecture from Figure 7.1, this collision rate can be interpreted as interventions by a collision avoidance system, such as emergency braking. This interpretation can

be achieved by adjusting the simulation parameters, for example, increasing the size of the cars or redefining the collision state to represent a collision avoidance intervention state.

## 7.3 Modular models in autonomous vehicles

There are two common strategies for creating and deploying DQNs to the real world: training a comprehensive model that encapsulates everything or training smaller, specialized models and switching between them as needed. The proposed MLEMTRL algorithm from Chapter 6 is a step towards the latter approach of using smaller models. Combined with the uncertainty measurements in Chapter 5, instead of reverting to a default action when uncertainty is high, the agent can instead trigger a model change. MLEMTRL can then be used to determine which model to switch to, ensuring a more adaptive and robust decision-making process.

This approach allows for modularity and flexibility, enabling easier updates and maintenance since individual models can be refined or replaced without affecting the entire system. Additionally, smaller models are less likely to overfit to irrelevant details present in a larger, comprehensive dataset, leading to more generalizable and robust performance in their specific domains. They also require less computational power and memory, making them more suitable for deployment on AVs with limited resources. Smaller models are generally easier to interpret and debug, facilitating understanding of the decision-making process and identifying any issues or biases, which is an important property to have when developing AVs.

# CHAPTER 8

## Concluding remarks and future work

This thesis introduces how RL-based methods can be used to develop a decision-making agent and evaluates their efficacy in learning when to drive across intersections, with a focus on managing the uncertainty of other drivers' intentions. By answering the three research questions presented in Chapter 1.3.

In response to Q1: *How can RL techniques be used to develop a decision-making agent that effectively navigates intersections without explicitly estimating the intention state of other vehicles?* The intersection navigation problem is formulated as a POMDP, where observable states include positions and velocities of the vehicles in the scenario, and unobservable states is the intentions of surrounding drivers. In Chapter 4, a deep Q-learning approach is introduced to solve the POMDP, with short-term goals, as discreet actions, translated into reference points and constraints for a controller. Two controllers are implemented and compared: one using sliding mode and another using MPC. The results show that the DQN effectively generates a policy for driving across an intersection crossing with dynamic behavior of other traffic participants, and its performance improved when combined with a robust controller like MPC. Additionally, the hidden state in the LSTM layer of the RL agent incorporates estimations of driver intentions, which improved the performance of the RL

agent in dynamic traffic environments.

A significant advantage of RL methods is their scalability to different scenarios through appropriate training. However, a drawback of deep Q-learning methods is the use of neural networks, which provide a black-box solution without indicating any confidence or uncertainty in their decisions. This limitation is addressed by answering Q2: *How can an RL agent utilize the uncertainty in its predictions and actions to enhance decision-making in complex environments?* Chapter 5 presents two approaches to address the uncertainty: an ensemble method addresses the uncertainty in the output of the DQN, and a belief-based method addresses the uncertainty in the intention estimation that is fed as an input to the DQN. The results demonstrate that accounting for uncertainty can significantly improve the agent's performance, especially in scenarios outside the training set, by avoiding collisions and improving decision-making robustness.

Chapter 6 addresses Q3: *How can an RL agent handle situations it has not been trained on?* The MLEMTRL algorithm is introduced to address the challenge of deploying RL agents in new environments by leveraging knowledge from previously trained models. The approach involves constructing a convex hull of source MDPs and using model transfer RL to identify the most relevant model for the target environment. This method is evaluated in various tasks, including a continuous state-action MDPs, demonstrating improved learning speed and asymptotic performance. The results show that MLEMTRL can construct a new model for an RL agent using a set of source models, provided the target model is sufficiently similar to the source models. This capability can potentially enable the efficient deployment of AD agents in new environments, reducing the need for extensive data collection and training specific to each new geographic location.

Overall, the research presented in this thesis contributes to the advancement of RL techniques for autonomous driving in diverse and dynamic environments. The findings show the critical roles of accurate intention estimation, effective uncertainty management, and the strategic use of transfer learning to build robust and versatile autonomous driving systems. While RL methods alone may still have challenges in guaranteeing safety for autonomous driving, integrating them with MPC and other active safety systems can transform RL into a powerful tool for creating comfortable and enjoyable rides for the passengers, paving the way for safer and more efficient transportation solutions.

## 8.1 Future work

Future research should focus on transitioning from simulation environments to real-world implementations. While simulations offer a controlled setting for developing and testing algorithms, real-world driving presents unpredictable variables and complexities. Implementing and refining these models in actual driving scenarios will be crucial for validating their effectiveness and reliability. This step will involve rigorous testing, continuous learning, and adaptation to ensure the autonomous systems can handle diverse and dynamic real-world conditions, ultimately moving closer to the widespread adoption of safe and efficient autonomous vehicles.

Additionally, the integration of language prediction models, specifically using transformers and attention mechanisms, along with driver monitoring systems (DMS), can enhance the prediction of driver intentions. Transformers, with their powerful attention mechanisms, can effectively handle sequential data and capture complex dependencies. By leveraging these models, it may be possible to interpret and predict driver behaviors based on a broader range of contextual cues.

Driver monitoring systems can provide critical real-time data on driver behavior, including eye movement, head position, and other physiological indicators. Combining this data with sensor inputs and verbal communication or textual descriptions of driver actions can create a comprehensive understanding of driver intentions. Transformers can process and correlate these different data types, improving the accuracy of intention estimation, particularly in complex or ambiguous driving scenarios.

Moreover, employing transformers trained on extensive datasets could facilitate the development of more sophisticated algorithms capable of predicting and adapting to diverse driving behaviors. This approach could significantly enhance the overall safety and efficiency of autonomous driving systems by providing a more nuanced and dynamic understanding of the driving environment. Integrating DMS with advanced language models could also help in identifying potential risks and ensuring timely interventions, thus improving the overall robustness and reliability of autonomous vehicles.

# CHAPTER 9

# Summary of included papers

This chapter provides a summary of the included papers.

## 9.1 Paper A

Tommy Tram, Anton Jansson, Robin Grönberg, Mohammad Ali, and Jonas Sjöberg
Learning Negotiating Behavior Between Cars in Intersections using Deep Q-Learning
*Published in 2018 21st International Conference on Intelligent Transportation Systems (ITSC)*,
pp. 3169–3174, Nov. 2018.
 DOI: 10.1109/ITSC.2018.8569316.

This paper concerns automated vehicles negotiating with other vehicles, typically human driven, in crossings with the goal to find a decision algorithm by learning typical behaviors of other vehicles. The vehicle observes distance and speed of vehicles on the intersecting road and use a policy that adapts its speed along its pre-defined trajectory to pass the crossing efficiently. Deep Q-

learning is used on simulated traffic with different predefined driver behaviors and intentions. The results show a policy that is able to cross the intersection avoiding collision with other vehicles 98% of the time, while at the same time not being too passive. Moreover, inferring information over time is important to distinguish between different intentions and is shown by comparing the collision rate between a Deep Recurrent Q-Network at 0.85% and a Deep Q-learning at 1.75%.

The thesis author contributed with the problem formulation, proposed algorithms and the writing of the paper.

## 9.2 Paper B

Tommy Tram, Ivo Batković, Mohammad Ali, and Jonas Sjöberg
Learning When to Drive in Intersections by Combining Reinforcement Learning and Model Predictive Control
*Published in 2019 IEEE Intelligent Transportation Systems Conference (ITSC)*,
pp. 3263–3268, Oct. 2019.
 DOI: 10.1109/ITSC.2019.8916922.

In this paper, we propose a decision making algorithm intended for automated vehicles that negotiate with other possibly non-automated vehicles in intersections. The decision algorithm is separated into two parts: a high-level decision module based on reinforcement learning, and a low-level planning module based on model predictive control. Traffic is simulated with numerous predefined driver behaviors and intentions, and the performance of the proposed decision algorithm was evaluated against another controller. The results show that the proposed decision algorithm yields shorter training episodes and an increased performance in success rate compared to the other controller.

The thesis author contributed with the problem formulation, proposed algorithms, implementation of the simulation and reinforcement learning algorithm, and the writing of the paper.

## 9.3 Paper C

Carl-Johan Hoel, Tommy Tram, and Jonas Sjöberg
Reinforcement Learning with Uncertainty Estimation for
Tactical Decision-Making in Intersections
*Published in 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*,
pp. 1-7, Sep. 2020.
 DOI: 10.1109/ITSC45102.2020.9294407.

This paper investigates how a Bayesian reinforcement learning method can be used to create a tactical decision-making agent for autonomous driving in an intersection scenario, where the agent can estimate the confidence of its decisions. An ensemble of neural networks, with additional randomized prior functions (RPF), are trained by using a bootstrapped experience replay memory. The coefficient of variation in the estimated Q-values of the ensemble members is used to approximate the uncertainty, and a criterion that determines if the agent is sufficiently confident to make a particular decision is introduced. The performance of the ensemble RPF method is evaluated in an intersection scenario and compared to a standard Deep Q-Network method, which does not estimate the uncertainty. It is shown that the trained ensemble RPF agent can detect cases with high uncertainty, both in situations that are far from the training distribution, and in situations that seldom occur within the training distribution. This work demonstrates one possible application of such a confidence estimate, by using this information to choose safe actions in unknown situations, which removes all collisions from within the training distribution, and most collisions outside of the distribution.

The thesis author contributed with the problem formulation, proposed algorithms, simulation implementation and the writing of the paper.

## 9.4 Paper D

Tommy Tram, Maxime Bouton, Jonas Fredriksson, Jonas Sjöberg, and Mykel Kochenderfer
Belief State Reinforcement Learning for Autonomous Vehicles in Intersections
*Submitted to IEEE Transactions on Intelligent Transportation Systems*,

This paper investigates different approaches to find a safe and efficient driving strategy through an intersection with other drivers. Because the intentions of the other drivers to yield, stop, or go are not observable, we use a particle filter to maintain a belief state. We study how a reinforcement learning agent can use these representations efficiently during training and evaluation. This paper shows that an agent trained without any consideration of the intentions of others is both slower at reaching the goal and results in more collisions. Four algorithms that use a belief state generated by a particle filter are compared. Two of the algorithms have access to the intention only during training while the others do not. The results show that explicitly trying to predict the intention gave the best performance in terms of safety and efficiency.

The thesis author contributed with the problem formulation, proposed algorithms, simulation and experimental implementations, and the writing of the paper.

## 9.5 Paper E

Hannes Eriksson, Tommy Tram, Debabrota Basu, Mina Alibeigi, and Christos Dimitrakakis
Reinforcement Learning in the Wild with Maximum Likelihood-based Model Transfer
*Published in 2024 International Conference on Adaptive Agents and Multi-Agent Systems (AAMAS)*,
pp. 516–524, May. 2024.

For decision-problems with insufficient data, it is imperative to take into account not only what you know but also what you do not know. In this work, ways of transferring knowledge from known, existing tasks to a new setting is studied. In particular, for tasks such as autonomous driving, the optimal controller is conditional on things such as, the physical properties of the vehicle, the local and regional traffic rules and regulations and also on the specific scenario trying to be solved. Having separate controllers for every combination of these conditions is intractable. By assuming problems with similar structure, we are able to leverage knowledge attained from similar tasks to guide learning for new tasks. We introduce a maximum likelihood estimation procedure

for solving Transfer Reinforcement Learning (TRL) of different types. This procedure is then evaluated over a set of autonomous driving settings, each of which constitutes an interesting scenario for autonomous driving agents to make use of external information. We prove asymptotic regret bounds for proposed method for general structured probability matrices in a specific setting of interest.

The thesis author contributed with the problem formulation and experimental analysis of the paper.

# References


[1] O. Y. I-Cheng Lin and C. Joe-Wong, "Mixed-autonomy era of transportation," Traffic21, Tech. Rep., 2022.

[2] K. Heineke, N. Laverty, T. Möller, and F. Ziegler, *The future of mobility: Mobility evolves*, Apr. 2023.

[3] D. J. Fagnant and K. Kockelman, "Preparing a nation for autonomous vehicles: Opportunities, barriers and policy recommendations," *Transportation Research Part A: Policy and Practice*, vol. 77, pp. 167–181, 2015, ISSN: 0965-8564.

[4] J. Janai, F. Güney, A. Behl, and A. Geiger, *Computer Vision for Autonomous Vehicles: Problems, Datasets and State of the Art*. Now Publishers Inc., 2020.

[5] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 187–210, 2018.

[6] "2019 traffic safety culture index," AAA Foundation for Traffic Safety, Washington DC, Tech. Rep. 202-638-5944, 2019.

[7] "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," On-Road Automated Driving (ORAD) Committee, Tech. Rep., 2021.

[8] L. Fletcher *et al.*, "The mit–cornell collision and why it happened," *Journal of Field Robotics*, vol. 25, pp. 775–807, Oct. 2008.

[9] S. Kammel *et al.*, "Team annieway's autonomous system for the 2007 darpa urban challenge," *Journal of Field Robotics*, vol. 25, pp. 615–639, Sep. 2008.

[10] L. Gressenbuch and M. Althoff, "Predictive monitoring of traffic rules," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2021.

[11] M. Liebner, M. Baumann, F. Klanner, and C. Stiller, "Driver intent inference at urban intersections using the intelligent driver model," in *IEEE Intelligent Vehicles Symposium (IV)*, 2012.

[12] S. Hoermann, D. Stumper, and K. Dietmayer, "Probabilistic long-term prediction for autonomous vehicles," in *IEEE Intelligent Vehicles Symposium (IV)*, 2017.

[13] C. B. Browne *et al.*, "A survey of Monte Carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 1–43, 2012.

[14] C. Hubmann, M. Becker, D. Althoff, D. Lenz, and C. Stiller, "Decision making for autonomous driving considering interaction and uncertain prediction of surrounding vehicles," in *IEEE Intelligent Vehicles Symposium (IV)*, 2017.

[15] Z. N. Sunberg, C. J. Ho, and M. J. Kochenderfer, "The value of inferring the internal state of traffic participants for autonomous freeway driving," in *American Control Conference (ACC)*, 2017.

[16] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a Frenét frame," in *IEEE Int. Conf. on Robot. and Automat.*, 2010.

[17] F. Damerow and J. Eggert, "Risk-aversive behavior planning under multiple situations with uncertainty," in *IEEE Int. Conf. on Intell. Transp. Syst. (ITSC)*, 2015.

[18] I. Batkovic, M. Zanon, M. Ali, and P. Falcone, "Real-time constrained trajectory planning and vehicle control for proactive autonomous driving with road users," in *European Control Conference (ECC)*, 2019.

[19] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2016.

[20] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[21] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018.

[22] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura, "Navigating occluded intersections with autonomous vehicles using deep reinforcement learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[23] V. Mnih *et al.*, *Playing atari with deep reinforcement learning*, 2013.

[24] M. J. Kochenderfer, *Decision Making Under Uncertainty: Theory and Application*. MIT Press, 2015.

[25] M. J. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2015.

[26] P. Zhu, X. Li, P. Poupart, and G. Miao, *On improving deep reinforcement learning for pomdps*, 2018.

[27] M. Bouton, A. Cosgun, and M. J. Kochenderfer, "Belief state planning for autonomously navigating urban intersections," in *IEEE Intelligent Vehicles Symposium (IV)*, 2017.

[28] A. Wang, A. C. Li, T. Q. Klassen, R. T. Icarte, and S. A. Mcilraith, "Learning belief representations for partially observable deep RL," vol. 202, Jul. 2023.

[29] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling, "Learning policies for partially observable environments: Scaling up," in *International Conference on Machine Learning (ICML)*, 1995.

[30] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.

[31] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[32] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, pp. 229–256, 2004.

[33] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 12, 1999.

[34] A. R. Cassandra, M. L. Littman, and N. L. Zhang, "Incremental pruning: A simple, fast, exact method for POMDPs," in *Conference on Uncertainty in Artificial Intelligence (UAI)*, 1997.

[35] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[36] M. Hessel *et al.*, "Rainbow: Combining improvements in deep reinforcement learning," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[37] M. Bouton, A. Nakhaei, K. Fujimura, and M. J. Kochenderfer, "Cooperation-aware reinforcement learning for merging in dense traffic," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2019.

[38] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical Review E*, vol. 62, pp. 1805–1824, 2 2000.

[39] H. V. Hasselt, A. Guez, M. Hessel, V. Mnih, and D. Silver, "Learning values across many orders of magnitude," in *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[40] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT Press, 2005.

[41] I. Osband, J. Aslanides, and A. Cassirer, "Randomized prior functions for deep reinforcement learning," in *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[42] C.-J. Hoel, K. Wolff, and L. Laine, "Tactical decision-making in autonomous driving by reinforcement learning with uncertainty estimation," in *IEEE Intelligent Vehicles Symposium (IV)*, 2020.

[43] J. Aitchison and S. Silvey, "Maximum-likelihood estimation of parameters subject to restraints," *The annals of mathematical Statistics*, vol. 29, no. 3, pp. 813–828, 1958.

[44] Y. Tassa *et al.*, "Deepmind control suite," *arXiv preprint:1801.00690*, 2018.

[45] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, 2018.

[46] T. Yu *et al.*, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *Conference on robot learning*, 2020.

[47] P. Koopman and M. Wagner, "Challenges in autonomous vehicle testing and validation," in *SAE International Journal of Transportation Safety*, 1. 2016, vol. 4, pp. 15–24.

[48] D. Åsljung, J. Nilsson, and J. Fredriksson, "Using extreme value theory for vehicle level safety validation and implications for autonomous vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 2, no. 4, pp. 288–297, 2017.

[49] D. Kortenkamp, R. G. Simmons, and D. Brugali, "Robotic systems architectures and programming," in *Springer Handbook of Robotics, 2nd Ed.*, 2008.

[50] K. Berntorp, T. Hoang, R. Quirynen, and S. Di Cairano, "Control architecture design for autonomous vehicles," in *IEEE Conference on Control Technology and Applications (CCTA)*, 2018.

[51] Z. Shiller, F. Large, S. Sekhavat, and C. Laugier, "Motion planning in dynamic environments," in *Autonomous Navigation in Dynamic Environments*, C. Laugier and R. Chatila, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 107–119, ISBN: 978-3-540-73422-2.

[52] M. Brännström, E. Coelingh, and J. Sjöberg, "Model-based threat assessment for avoiding arbitrary vehicle collisions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 3, pp. 658–669, 2010.

[53] M. Brännström, E. Coelingh, and J. Sjöberg, "Decision-making on when to brake and when to steer to avoid a collision," *International Journal of Vehicle Safety*, vol. 7, no. 1, pp. 87–106, 2014.

[54] "Road vehicles — functional safety," International Organization for Standardization, Standard, Dec. 2018.