Distracted Driver Detection: A Safer Reinforcement Learning Approach.

Master Thesis by
Pieter El Sharouni
5930499

Utrecht University

Department of Natural Sciences
Artificial Intelligence
Utrecht University
The Netherlands

October 2023

Supervisors:
First: Shihan Wang
Second: Mihaela Mitici

Abstract

Both driver inattention and driver distraction present significant challenges in road safety, leading to an increasing number of accidents and fatalities every year. As drivers periodically become distracted, driving performance declines, and accidents become more likely. A cooperative lane-keeping assistance system could enhance safety while retaining most of the driver's autonomy. To train this system, Safe Reinforcement Learning with a memory component will be utilized. Safe Reinforcement Learning involves learning policies that maximize rewards in scenarios where maintaining reasonable system performance and safety is crucial during the learning or deployment stages. Adding memory-based Deep Reinforcement Learning improves performance in partially observable environments. Since the driver's psychological state is unknown to the system, the problem will be formulated as a Partially Observable Markov Decision Process (POMDP). However, during experimentation, memory-based DRL did not show any improvement compared to regular DRL. Therefore the problem was later reformalized as a Block Markov Decision Process (BMDP). We will use First Order Constrained Optimization in Policy Space (FOCOPS) extended with a Long Short-Term Memory (LSTM) layer, to address the distracted driver issue. The problem will be divided into three subsections: first, exploring the advantage of using memory-based Deep Reinforcement Learning in BMDPs; second, examining the benefit of using Safe Reinforcement Learning for the distracted driver problem; and finally, comparing FOCOPS with an LSTM layer to state-of-the-art reinforcement learning methods. We chose Recurrent Safe Reinforcement Learning to increase the learning rate and policy safety, making it more suitable for real-world applications.

Keywords— Distracted Driver, Safe Reinforcement Learning, Memory-based, Lane-keeping

Contents

1 Introduction

1.1 Motivation

Both driver inattention and distraction present significant challenges in road safety, leading to a rising number of accidents and fatalities each year (Kashevnik, Shchedrin, Kaiser, & Stocker, 2021; Regan, Lee, & Young, 2008; Young, Regan, & Hammer, 2007). In 2020, 38,824 individuals were killed in motor vehicle traffic accidents on US roadways. Of these fatalities, 3,142 or 8.1% involved at least one distracted driver (Stewart, 2023). Other sources suggest that approximately a quarter of vehicle crashes result from driver distraction (Young et al., 2007). Addressing the distracted driver problem is of paramount importance.

One potential solution to these driving accidents is autonomous driving. Autonomous driving offers numerous benefits, such as fewer traffic accidents and relieving vehicle occupants of driving and navigation tasks (Ondruš, Kolla, Vertal', & Šarić, 2020). If cars could operate entirely autonomously, driver attentiveness would become irrelevant. However, achieving full autonomy faces several challenges, including multi-sensory data synchronisation, failure detection, cyberattack protection, and interacting with human drivers (L. Liu et al., 2020). It may be some time before fully autonomous vehicles become a reality (Palade & Deo, 2022; Rajasekhar & Jaswal, 2015). By 2040, members of the Institute of Electrical and Electronics Engineers (IEEE) estimate that 75% of all vehicles will be autonomous (Ondruš et al., 2020). Until such vehicles are available, driving assistance systems with increasing autonomy will be employed transitionally (Stewart, 2023). Minor tasks like lane-keeping or cruise control are delegated to these systems, while the driver retains responsibility for other tasks in a supervisory capacity. This paper focuses on the lane-keeping task.

By developing a lane-keeping assistance system that adapts to the drivers' attentiveness, a major contribution to road accidents and fatalities can be tackled. Moreover, Improved lane-keeping assistance systems can contribute to the overall development of autonomous driving technology by serving as a stepping stone, enabling researchers to build on these advancements to create more sophisticated systems (Nidamanuri, Nibhanupudi, Assfalg, & Venkataraman, 2021). Although driver fatigue is not the focus of this thesis, it also impairs driving behaviour and is a major contributor to traffic accidents (Philip et al., 2005). Improved lane-keeping assistance systems could help mitigate the risks associated with driver fatigue by providing additional support during long journeys.

1.2 Current Lane Assistance

Presently, lane assist systems employ a basic rule-based approach that detects when the driver is nearing the edge of the road, providing a warning or correcting the steering (Kashevnik et al., 2021; Zakaria et al., 2023). However, issues arise when the vehicle encounters a situation that does not fit any of the predefined rules. In such cases, the vehicle may function properly most of the time but struggles with edge-cases for which no rules exist. Consequently, these systems have limited utility, or drivers may rely too heavily on them, leading to potential malfunction or inattentiveness when supervision is needed (Beckers, Siebert, Bruijnes, Jonker, & Abbink, 2022; van de Merwe, Mallam, & Nazir, 2022). Therefore, to combat the issues with a rule-based approach such as these edge-cases, a policy-based approach could be utilized.

Another limitation of current lane-keeping assistance systems is the lack of adapta-

tion to the driver's behaviour. Research has shown that continuous assistance based on the distance from the centerline enhances lane-keeping performance and is the most preferred style of lane-keeping assistance, even for drivers with better lane-keeping skills while engaged in secondary distracting tasks (Blaschke, Breyer, Färber, Freyer, & Limbacher, 2009; Pohl, Birk, & Westervall, 2007; Roozendaal, Johansson, Winter, Abbink, & Petermeijer, 2021).

1.3 Task Delegation

While assigning tasks to an assistance system could enhance driving performance, it may also limit the driver's autonomy (Zahabi, Razak, Shortz, Mehta, & Manser, 2020). This loss of autonomy could turn driving from an engaging and enjoyable task into a monotonous and tedious supervisory chore. Consequently, drivers may be more prone to distraction (Shahini & Zahabi, 2022), have slower reaction times (Capallera, Angelini, Meteier, Abou Khaled, & Mugellini, 2022), be more likely to speed (Monfort et al., 2022), experience reduced situational awareness (O'Neill, McNeese, Barron, & Schelble, 2022) or place excessive trust in the assistance system (Detjen, Faltaous, Pfleging, Geisler, & Schneegass, 2021). Furthermore, it may take drivers too long to react and regain control from the assistance system in critical situations (Beckers et al., 2022; van de Merwe et al., 2022).

Adaptive lane-keeping, based on the driver's attention level, can help keep drivers engaged throughout the driving process. For example, the assistance system would become more sensitive when the driver is inattentive and increase torque if the driver takes incorrect actions. Previous studies have shown that human-machine cooperation improves driving performance (Q. Li et al., 2021; C. Liu et al., 2022) and that online adaptation of assistance according to the driver's state enhances driving performance while reducing workload during distracted phases (Benloucif, Sentouh, Floris, Simon, & Popieul, 2019).

1.4 Improved Lane Assistance

To address these issues, an improved lane assist system should be capable of handling all situations, including unexpected ones, rather than relying on a rule-based approach while the majority of autonomy remains with the driver. A more advanced system should learn a policy that can manage cases it has not encountered before. Furthermore, given the preference for continuous cooperative lane control, we propose a cooperative policy-based continuous lane-keeping system.

However, a problem with a policy-based system is that it has to learn said policy. A way for such a lane assistance system to learn a policy is reinforcement learning, a method where an agent is placed in an environment and either rewarded or punished for taking certain actions (Kaelbling, Littman, & Moore, 1996). This way the agent learns which actions are considered beneficial in certain situations and which are considered detrimental. However, to learn which action to take in which situation, a lot of exploration in necessary. In the case of a lane assistance system this exploration will involve taking actions that may result better driving, but may also involve taking actions that may result in crashing. As crashing is the absolute worse result for a lane assistance system, some safety measures need to be taken. These safety measures can be implemented by incorporating them in the reward function. However, as the agent only gets a single reward based on its action and this reward often consists of many parts, it may be difficult for the agent to distinguish the rewards

and punishments from a single reward. Therefore having the safety measures part of a separate cost function the agent can more easily learn which actions will maximise the reward and minimise the costs. This sub-field or reinforcement learning is called Safe Reinforcement Learning (Garcıa & Fernández, 2015).

1.5 Cooperative control

Combining these issues, we propose a cooperative control scheme in which the driver and the assistance system share control over the car. The assistance system adapts to the driver's behaviour, exerting minimal to no influence on the vehicle when the driver is attentive and on the centerline, while increasing its influence when the driver is attentive and further from the centerline. This approach allows the system to support the driver when they are distracted or unable to drive safely, while still maintaining the majority of control with the driver, thereby enhancing autonomy, situational awareness, and driver safety. As safety is a crucial part of a functional lane-keeping assistant, the proposed lane-keeping system will utilized a safe reinforcement learning algorithm to ensure safety.

1.6 Outline

The remaining content of the thesis is organized as follows:

- Chapter 2: Introduces the main concepts of the theoretical background related to POMDPs and Safe Reinforcement Learning.
- Chapter 3: Shows the related work and research questions.
- Chapter 4: Presents the methodology, defining the POMDP and BMDP used in our problem statement and the methods how we will answer the research questions.
- Chapter 5: Presents the results to the research questions
- Chapter 6: Analyses the results and discussed the limitations to the experimental setup and several possible adaptations for further research
- Chapter 7: Conclusion to the thesis and summarizes the findings

2 Theoretical background

This chapter focuses on the theoretical background of the problem and introduces the basic concepts that serve as a foundation of the problem and solution approaches in this thesis. The problem is assisted lane-keeping with shared control by a potentially distracted human driver and assistant agent. In Section 2.1 introduces MDPs and shows how Reinforcement Learning (RL) can be used to solve them. However, as the driver's attention is unknown to the agent but does effect driving performance, the agent observes only partial information. Therefore an extension of MDPs, Partially Observable Markov Decision Processes (POMDP) are explained to deal with the partially observable information. Notation and information is gathered from fundamental research papers and books by Kaelbling, Littman, and Cassandra (1998); Kaelbling et al. (1996) and Sutton and Barto (2018). Section 2.2 introduces the different methods of incorporating safety in Reinforcement Learning. In Section 2.3 the challenges and possible solutions for POMDP problems are introduced with reinforcement learning algorithms and memory-components to solve POMDP's.

2.1 Reinforcement Learning

2.1.1 Markov Decision Process

Lane-keeping of a car is a sequential decision making task as every driving action directly influences the choice of the best following driving actions. Such a sequential decision problem is typically modelled as a Markov Decision Process (MDP) (Kaelbling et al., 1996). In such an environment it is assumed that the agent does not know the parameters of this process, but has to learn how to act directly from experience. At every time step, t, the environment is in a certain state, s_t, fully observable by the agent. The agent interacts with the environment by taking an action a_t, which determines in which state, s_{t+1}, the agent comes next. The underlying assumption of MDPs is that the next state of the agent is only determined by the current state and the action of the agent. To account for the randomness of the environment the transition can be probabilistic. After taking some action, a_t, and arriving at some state, s_{t+1}, the agent receives a reward r_t. Thus a MDP problem, illustrated in Figure 1, can be formalizes as a tuple $< S, A, T, R >$ where:

- S is the set of states called the state space.
- A is the set of actions called the action space.
- $T : S \times A \longrightarrow \Delta(S)$ is the transition dynamics. For each action a and state s, $T(s, a)$ is the probability distribution over states that the system may transition into when taking action a in state s
- $R : S \times A \times S \longrightarrow R$ is the reward function. The value $r(s, a, s')$ is the reward associated with transitioning from state s to s' by taking action a.

The agent's goal is to maximize this reward, also called the return. The difficulty of taking the action with the highest immediate reward in each state is that it may not result in the highest cumulative return over a longer sequence of actions. Thus the agent needs to find the optimal policy that decides the best action in each state with respect to the cumulative reward of time. If the state transitions are known, the optimal policy can be found using model-based techniques such as value or policy iteration. If the transition model is unknown, model-free reinforcement learning can be applied to learn an optimal policy.

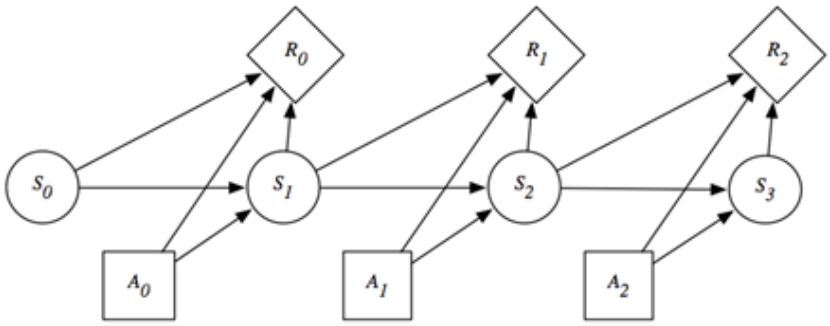


Figure 1: A decision network representing part of an MDP.
Source: https://artint.info/html/ArtInt_224.html

Reinforcement learning has been used to solve MDP's in a wide variety of complex tasks with excellent success rate such as playing games like DOTA 2 or Atari (Berner et al., 2019; Mnih et al., 2013), autonomous driving (Chen, Yuan, & Tomizuka, 2019) and robotics (Singh, Kumar, & Singh, 2022). In the standard reinforcement-learning model, an agent is connected to its environment via perception and action, as seen in Figure 2. On each step of interaction the agent receives the current state of the environment as input, s_t. The output created by the agent is some action, a, determined by the policy of the agent. The action changes the state of the environment, and the agent learns the value of this state transition the reward, r_t. The agent's policy, π, dictates which actions to take based on the current state. The agent learns which actions increase the long-run sum of values through systematic trial and error. There are a wide variety of algorithms associated with optimizing how the agent learns (Bertsekas & Tsitsiklis, 1996; Kaelbling et al., 1996; Kearns & Singh, 2002; Sutton & Barto, 2018).

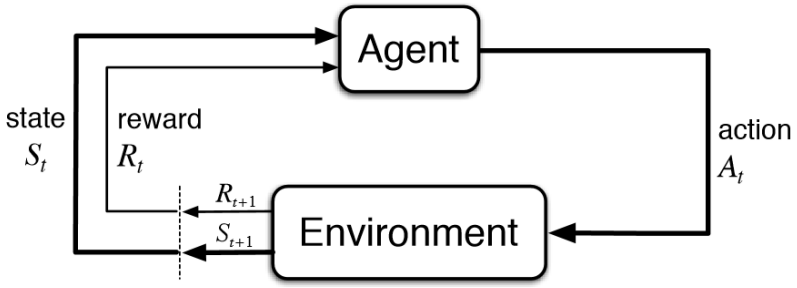


Figure 2: A simple Reinforcement Learning model.
Source: Georgeon et al. (2015)

In order to learn which actions increase the long-run sum of values the most, the agent has to take the future into account in the decisions it makes about how to behave now. The simplest way to model this is the finite-horizon model, where the agent optimizes its expected reward for the next h steps:

$$E\left(\sum_{t=0}^{h} r_t\right) \tag{1}$$

Here the agent only needs to attend to the next h amount of steps and does not concern itself with what happens after. A more realistic model is the infinite horizon discounted model. Here the model takes into account the long-run reward of the agent, but rewards that are received in the future are discounted according to some discount factor γ, where $(0 \leq \gamma \leq 1)$:

$$E\left(\sum_{t=0}^{\infty} \gamma^t r_t\right) \tag{2}$$

Each state will have an optimal value, the expected infinite discounted sum of reward that the agent will receive if it starts in that state and executes the optimal policy:

$$V^*(s) = \max_{\pi} E\left(\sum_{t=0}^{\infty} \gamma^t r_t\right) \tag{3}$$

This optimal value function is unique and can be defined as the solution to the equation:

$$V^*(s) = \arg\max_{a}\left(R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')V^*(s')\right) \tag{4}$$

This function states that the optimal value of a state is the immediate reward the agent receives from taking the best available action a in state s plus the expected discounted value of the next state. With this optimal policy can be denoted as:

$$\pi^*(s) = \arg\max_{a}\left(R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')V^*(s')\right) \tag{5}$$

Reinforcement learning can be split up into two sections, model-based and model-free methods. In model-based methods the model is already to know and we can use this to find the optimal policy. The model consists of the state transition probability function $T(s,a,s')$ and the reward function $R(s,a)$. Reinforcement learning and this thesis are mainly concerned with how to obtain the optimal policy when such a model is not available. The agent must interact with the environment to learn the state transitions and the reward function. These are the model-free methods, which can be split up into two sections, value iteration and policy iteration.

One way to find the optimal policy is to find the optimal value function. Value iteration algorithms known as Q-learning learn the optimal policy by looping through all the actions of all the states, updating the state-action value, until the policy is good enough based on some criteria:

$$Q(s,a) = R(s,a)\gamma \sum_{s' \in S} T(s,a,s')V^*(s') \tag{6}$$

Methods of this family learn an approximator, $Q_\Theta(s,a)$, for the optimal action-value function, $Q^*(s,a)$. This optimization is almost always performed off-policy, which means that each update can use data collected at any point during training,

regardless of how the agent was choosing to explore the environment when the data was obtained.

From all these Q-values, the agent needs to choose the optimal action to get the optimal reward. This can be difficult as the Q-function is a value based function with high variability. In order to reduce this variability, the advantage function can be used. The advantage function is the difference between the Q-value and the average of actions which it would have taken in that state:

$$A(s,a) = Q(s,a) - V(s) \tag{7}$$

Instead of finding the optimal policy indirectly through optimal value function, policy iteration algorithms manipulate the policy directly. First the value function of some policy π is computed, after which the policy is improved at each state:

$$V_\pi(s) = R\left(s, \pi(s)\right) + \gamma \sum_{s' \in S} T\left(s, \pi(s), s'\right) V_\pi(s') \tag{8}$$

$$\pi'(s) = \arg\max_a \left(R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V_\pi(s') \right) \tag{9}$$

Methods in this family represent a policy explicitly as $\pi_\Theta(a|s)$. They optimize the parameters Θ either directly by gradient descent on the performance function $J(\pi_\Theta)$, or indirectly by maximizing local approximations of $J(\pi_\Theta)$. This optimization is almost always done on-policy, which means that each update only uses data collected while acting according to the most recent version of the policy. Policy optimization also usually involves learning an approximator $V_\Theta(s)$ for the on-policy value function $V^\pi(s)$, which gets used in figuring out how to update the policy. The value function of a policy is the expected infinite discounted reward that will be gained, at each state, by executing that policy. Once the value of each state under a certain policy is known, this value can be improve by changing the first action taken. If it can, the policy is updated to take the new action whenever in that situation. Figure 3 shows a taxonomy of algorithms of modern reinforcement learning.

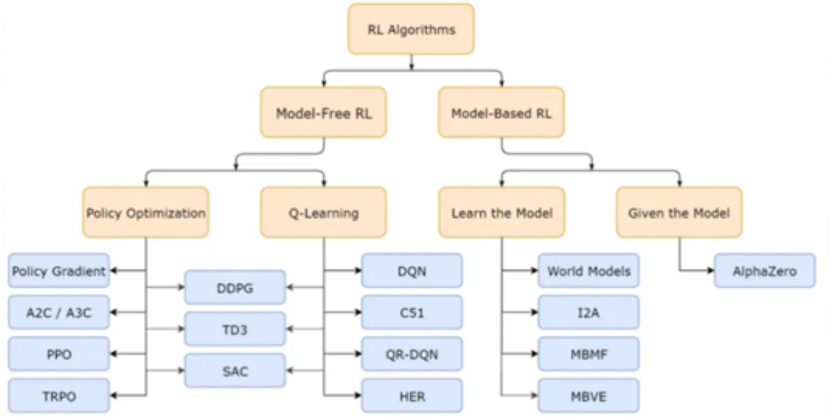


Figure 3: A taxonomy of modern RL algorithms.
Source: https://spinningup.openai.com

In the example of a lane-keeping assistance agent, the agent will be rewarded for choosing the correct actions based on the vehicles sensors. The reward will be based driving performance such as lane-centeredness and speed, which are the most common parameters for the reward function (Zhu & Zhao, 2021). If the driver model is not distracted, the vehicle will remain in the centre of the lane. Therefore, if the agent takes any action, it will indirectly be punished for acting when the driver is attentive, thus adapting to the driver's model distraction level. The agent will have to learn when to take which actions.

In some situations, such as expensive robots in real environments, safety of the agent is particularly important. In such scenarios researchers have to pay attention not only to optimizing long term rewards, but also to damage avoidance (Garcia & Fernández, 2012; Koppejan & Whiteson, 2011; Martín H & de Lope, 2009). Naturally, safety is of vital importance in a lane-keeping assistance agent. By limiting the actions during the exploration process or by constraining the optimization function, safety can be achieved.

2.1.2 Constrained Markov Decision Process

A Constrained Markov Decision Process (CMDP) is an MDP with an additional set of constraints C which restrict the set of allowable policies (Y. Zhang, Vuong, & Ross, 2020). Specifically, the MDP is augmented with a set C of auxiliary cost functions, $C_1, \ldots, C_m$, with each one a function $C_i : S \times A \times S \to R$ mapping transition tuples to costs, and limits $d_1, \ldots, d_m$. Let $J_{C_i}(\pi)$ denote the expected discounted return of policy π with respect to cost function:

$$C_i : J_{C_i}(\pi) = E_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t C_i(s_t, a_t, s_{t+1}) \right] \tag{10}$$

The set of feasible stationary policies for a CMDP is then:

$$\Pi_C = \{\pi \in \Pi : \forall i, J_{C_i}(\pi) \leq d_i\} \tag{11}$$

and the reinforcement learning problem in a CMDP is:

$$\pi^* = \arg \max_{\pi \in \Pi_C} J(\pi) \tag{12}$$

In Constrained Markov Decision Processes (CMDP) the objective is to maximize the agents' reward while making the agents satisfy safety constraints (Achiam, Held, Tamar, & Abbeel, 2017). Similar to the standard V^π, Q^π and A^π for reward in unconstrained reinforcement learning, the cost value function, cost action-value function and cost advantage function are defined by replacing the reward R with cost C.

While updating the policy in unconstrained policy space can lead to more stable behaviour and better sample efficiency, solving CMDPs directly within the context of local policy search can be challenging and sample inefficient since after each policy update, additional samples need to be collected from the new policy in order to evaluate whether the constraints are satisfied. A surrogate cost function which evaluates the constraint $J_C \pi_\theta$ using samples collected from the current policy π_{θ_k} is shown to be a good approximation (Achiam et al., 2017) when π_θ and π_{θ_k} are close w.r.t the KL divergence $D_{KL}(\pi_\theta || \pi_{\theta_k})$. A new policy in a CMDP is obtained by solving the optimization problem:

$$\max_{\pi_{\theta_k} \in \Pi_\theta} \mathbb{E}_{(s,a)\sim\pi_{\theta_k}}[A_C^{\pi_\theta}(s,a)] \tag{13}$$

subject to:

$$J_C(\pi_{\theta_k} + \frac{1}{1-\gamma}\mathbb{E}_{(s,a)\sim\pi_{\theta_k}}[A_C^{\pi_{\theta_k}}(s,a)] \leq b \tag{14}$$

$$D_{KL}(\pi_\theta||\pi_{\theta_k}) \leq d \tag{15}$$

2.1.3 Partially Observable Markov Decision Process

For MDP's the optimal policy can be computed and executed by simply executing the policy in every state s. However, what would happen if the agent would no longer be able to accurately determine the state it was in with complete reliability? In order to effectively behave in a partially observable world, it is necessary to use memory of the previous actions and observations to aid in the disambiguation of the states of the world.

The Partially Observable Markov Decision Process (POMDP) is a mathematically principled framework to model decision-making problems in the non-deterministic and partially observable scenarios. According to Kaelbling et al. (1998), a POMDP can be describes as the tuple $< S, A, T, R, \Omega, O >$, illustrated in Figure 4, where:

- S, A, T and R describe a Markov Decision Process.
- Ω is the finite set of observations the agent can experience of its world.
- $O : S \times A \longrightarrow \Pi(\Omega)$ is the observation function which gives, for each resulting state and action, a probability distribution over possible observations.

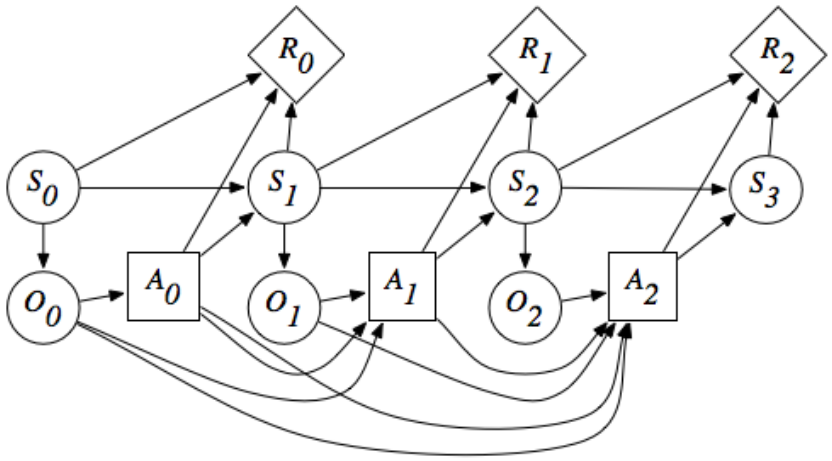


Figure 4: A POMDP framework.
Source: https://artint.info/html/ArtInt_230.html

A POMDP is an MDP in which the agent is unable to observe the current state. Instead, it takes actions based on observations, which are part of the true state of the environment. In the distracted driver problem the unobservable is the level of

distraction of the driver. The agent cannot know if the driver model is distracted, but has to infer this from parameters that can be measured, such as distance sensors and steering input from the driver. The agent utilizes the observations over time to estimate the true state of the environment and choose the correct actions. At time t, it takes into account the complete history h_t of actions and observations until t:

$$h_t = \{a_0, o_1, ..., o_{t-1}, a_{t-1}, o_t\} \tag{16}$$

However, keeping a memory of all the past actions and observations will become immensely expensive memory wise. Therefore a probability distribution over the states is kept, called belief b. the probability of being in state s, given history h is given by $b(s, h)$.

$$b_t(s, h) = \Pr(s_t = s | h_t = h) \tag{17}$$

Thus only the belief needs to be kept and recursively updated as the agent performs actions and receives new observations. The agent starts with an initial belief b_0 about the initial state of the environment. At each subsequent step, the new belief b' is updated using the previous belief b, the last action a and the current observation o. The agent chooses an action based on the policy and the current belief. Solving a POMDP consists of finding the optimal policy π^* that maximizes the cumulative reward obtained over some time horizon N with initial belief b_0 using a discount factor $0 \leq \gamma \leq 1$ according to the following equation:

$$\pi^* = \arg\max_{\pi} E\left[\sum_{t=0}^{N} \sum_{s \in S=0} b_t(s) \sum_{a \in A} \lambda^t R(s, a) \pi(b_t, a) | b_0\right] \tag{18}$$

The return received by following a policy π with a certain belief b can be calculated with the value function $V^{\pi}(b)$:

$$V^{\pi}(b) = \sum_{a \in A} \pi(b, a) \left[\sum_{s \in S} b(s) R(s, a) + \lambda \sum_{o \in P} \Pr(o|b, a) V * \pi(b')\right] \tag{19}$$

The optimal policy π^* maximizes $V^{\pi}(b)$, which can also be written as:

$$\pi^*(b_{t_0}) = \arg\max_{\pi} E_{\pi}\left[\sum_{t_0}^{T} \gamma^{t-t_0} r_t | b_{t_0}\right] \tag{20}$$

For any POMDP there exists at least one optimal policy. However, due to the unobservable nature of POMDPs, finding the optimal policy is notoriously difficult. The goal of the agent remains to maximize the expected discounted future reward. As the internal state of the driver is unknown to our lane-keeping system, this will result in a POMDP problem. Deep Reinforcement Learning and more specifically Safe Reinforcement learning has been used before to solve POMDP's (Andriotis & Papakonstantinou, 2021; Carr, Jansen, Junges, & Topcu, 2023; Shen, Ausin, Mostafavi, & Chi, 2018; Simão, Suilen, & Jansen, 2023).

2.2 Reformalization

While the distracted driver problem conforms to the characteristics of a POMDP, it is plausible to reframe the problem to enhance tractability for the algorithms. A potential approach involves leveraging the observations, which may reveal sufficient

information to the agent, enabling effective policy learning without explicitly considering the underlying belief. In theory, given the car's position on the track, the car's angle, and the previous driver's steering and acceleration inputs at each time step, a sufficiently sample efficient algorithm should be capable of making correct decisions, regardless of the latent state of the driver.

2.2.1 Preliminary Results

While a POMDP formalization seems to fit due to the unknown state of the driver, there are other formalizations that may be more appropriate. This subsection critically examines the assumption that the distracted driver problem can be formulated as a Partially Observable Markov Decision Process (POMDP). As this thesis builds upon the work of J. Jansen (2021), we have adopted similar assumptions, primarily that due to the unknown attentive state of the driver, leading to partial observability, which may impede conventional reinforcement learning methods from approximating an optimal solution. Nevertheless, we hypothesize that the car's position, angle, and past driver actions could provide sufficient information for the algorithm, rendering the attentive state of the driver irrelevant.

To challenge the POMDP assumption, we conduct a comprehensive analysis using preliminary results by comparing two variants of the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm. One variant includes a Long Short-Term Memory (LSTM) layer, intended to better handle a POMDP environment. In Section 5, the obtained results are presented, indicating that the problem may not be a true POMDP environment. Subsequently, in Section 2.2.2, we propose an alternative problem formalization, considering the promising outcomes of the experiments.

2.2.2 Block MDP

Given the preliminary results, we will now re-formalize our problem from a POMDP to a Block Markov Decision Problem (BMDP). In a BMDP, the environment is characterized by a finite, yet unobservable latent state space S, a finite action space A, and a potentially infinite, but observable context space X. Although this model has been implicitly assumed in previous works (Azizzadenesheli, Lazaric, & Anandkumar, 2016; Dann et al., 2018; Krishnamurthy, Agarwal, & Langford, 2016), it was first formally introduced by Du et al. (2019).

The dynamics of a BMDP are described by an initial state $s_1 \in S$ and two conditional probability functions: the state-transition function p, and the context-emission function q, which define conditional probabilities $p(s'|s,a)$ and $q(x|s)$ for all $s, s' \in S, a \in A, x \in X$. In each episode, the environment initiates in state s_1. During step $h \in H$ of an episode, the environment generates a context $x_h \sim q(\cdot|s_h)$, and the agent observes this context x_h, while the underlying state s_h remains unobserved. The agent then takes an action a_h, causing the environment to transition to a new state $s_{h+1} \sim p(\cdot|s_h, a_h)$. A BMDP can be describes as the tuple $< S, A, R, X, p, q >$, where:

- S, A, R describe a Markov Decision Process.
- p is the state-transition function, similar to T in a MDP or POMDP
- X is the context space, observed by the agent, generated by the unknown true state s. In the distracted driver problem, the context space is the observed inputs, while the unknown true state s is the vehicle state plus the driver's inputs and the driver's attentive state.

- q is the context-emission function, which maps each context $x \in X$ to their generated state s. For our problem this means that each observed input is generated from a single state and can be this state can be backtracked through the context x.

Though the initial description might resemble that of a POMDP, we differentiate the BMDP by making the following general assumption:

Assumption 1 *Each context x is uniquely generated by each state s. That is, the context space X can be partitioned into disjoint blocks X_s, each containing the support of the conditional distribution $q(\cdot|s)$. The block structure implies the existence of a perfect decoding function $f : X \longrightarrow S$, which maps contexts into their generating states (Du et al., 2019).*

Adapting this assumption to our distracted driver problem formulates the following assumption:

Assumption 2 *Each generated input to the agent x is uniquely determined by the true state s, which include the driver's attentive state. That is, each input X can be partitioned into disjoint blocks X_s, supporting the conditional distribution $q(\cdot|s)$. The block structure implies the existence of a perfect decoding function $f : X \longrightarrow S$, which maps inputs to the driver's attentive state.*

In the context of the distracted driver problem, the observable context space X comprises essential inputs to the algorithm, namely the track position, track angle, and the previous driver actions. The action space A consists of the algorithm's steering and acceleration outputs. On the other hand, the unobservable latent state S encompasses both the car's state and the unobservable state of the driver. For state transitions, we utilize a state transition function p that quantifies the likelihood of transitioning to another state given the current state and actions. Additionally, the context-emission function q characterizes how the observable context space x corresponds to the unobservable true state s.

Notably, the context space x is influenced by the actual state and will exhibit variation between a distracted driver and an attentive driver. A distracted driver's steering and acceleration inputs will consistently differ from those of an attentive driver. Thus, in theory, we can partition the context space X into two distinct blocks: one representing inputs resulting from a distracted driver, and the other representing inputs arising from an attentive driver. This separation allows for the fulfillment of the assumption that each context x uniquely determines its generating state s, enabling us to formally address our problem as a Block Markov Decision Problem (Block MDP). Since a Block MDP is essentially a POMDP where the observational space can be separated into disjoint blocks, no alterations are required in the environmental setup or code for implementing this formalization.

For visual clarity, Figure 5 provides an illustrative representation of the BMDP framework within the context of the distracted driver problem. The framework operates through an iterative process:

- True State (S_0): The actual state of the environment at a given time step is represented as S_0. This state encapsulates all relevant information about the vehicle's position, speed, and other critical factors.

- Observed Context (X_0): The agent observes the context X_0, which provides insight into the prevailing environmental factors and potential distractions influencing the driver. This observed context serves as essential input for the agent's decision-making process and is generated the context-emission function q and the previous state.
- Driver Model: Instead of directly determining the agent's action based on the observed context, a driver model intervenes. This driver model determines the driver's attentiveness or distracted state and influences the action to be taken. The action taken by a distracted driver will be different from the action taken by an attentive driver.
- Action (A_0): The driver model influences the action A_0 chosen by the agent. This action involves decisions related to acceleration, braking and steering.
- Next State (S_1): After executing the chosen action A_0, the environment transitions to a new state, denoted as S_1. This state results from the combination of the previous state S_0 and the action taken A_0.
- Reward (R_0): The reward R_0 is computed based on the previous state S_0, the current state S_1, and the action taken A_0. This reward serves as feedback for the agent, indicating the quality of its decision and behavior.

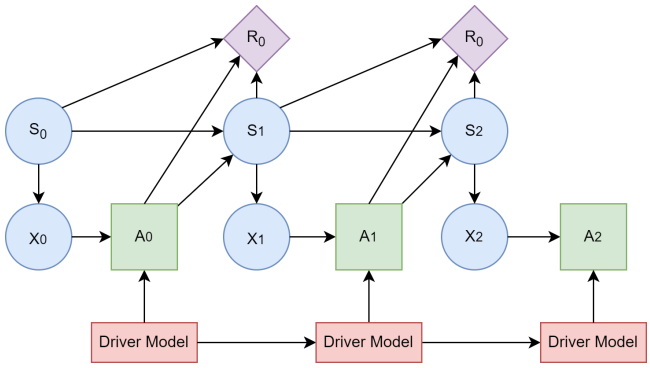


Figure 5: A BMDP framework.

2.3 Safe Reinforcement Learning

As maximizing the long-term reward does not necessarily avoid negative outcomes, another criteria is necessary to evaluate risk. This sub-field within Reinforcement Learning is called Safe Reinforcement Learning (SRL). According to an extensive survey by Garcıa and Fernández (2015), Safe RL algorithms can be separated into two fundamental tendencies and can be defined as the process of learning policies that maximize the expectation of the return in problems. An additional important factor in Safe RL is to ensure reasonable system performance by employing safety constraints during learning and execution. While Garcıa and Fernández (2015) explores the ways that SRL algorithms can be changed to ensure safety, Zhu and Zhao (2021) talks about how methods or regular reinforcement learning methods are modified to ensure safety.

2.3.1 Gracia and Fernández

The Safe Reinforcement learning categories of Garcıa and Fernández (2015) are more oriented towards reward and tasks and are separated into two tendencies. The first tendency consists of transforming the optimization criterion, the second consists of modifying the exploration process in two ways: i) through the incorporation of external knowledge and ii) through the use of a risk metric. Deep Safe RL for high dimensional and continuous MDP optimization problems is a relatively new area that has emerged in recent years, where safe states or actions are represented using neural networks (Gu et al., 2022).

Optimization Criterion The object of traditional RL algorithms is to find a function which specifies an action or a strategy for some state of the system to optimize a criterion. However, this optimization criterion is not always the most suitable one in dangerous or risky tasks (Geibel & Wysotzki, 2005; Mihatsch & Neuneier, 2002). There are several alternatives to this optimization criterion in order to consider risk and thus employ Safe Reinforcement Learning. These criteria can be categorized in four groups:

- Worst-Case Criterion: In the Worst-Case Criterion, a policy is considered to be optimal if it has a maximum worst-case return (Gaskett, 2003; Nilim & El Ghaoui, 2005; Tamar, Xu, & Mannor, 2013). This criterion is useful when avoiding rare occurrences of large negative return is imperative, but it can be overly pessimistic. Additionally the true long term utility of the actions are lost and does not detect risky situations from the early steps.
- Risk-sensitive Criterion: In other approaches the optimization criterion is transformed so as to reflect a subjective measure balancing the return and the risk. It can be transformed into an exponential utility function, a linear combination of return and risk or the probability of entering into an error state (Geibel & Wysotzki, 2005; Howard & Matheson, 1972; Sato, Kimura, & Kobayashi, 2001). The advantages of this criterion is that it is easier to switch between risk-averse and risk-seeking behaviour and long-term risk situations are detected. However, the policy may also be overly pessimistic and it does not detect risky situations from the early steps.
- Constrained Criterion: The objective of these methods is to maximize the return subject to one or more constraints resulting in a constrained optimization criterion. In these cases the objective is to maximize the return while keeping other types of expected measures higher than some given bound (Kadota, Kurano, & Yasuda, 2006; Moldovan & Abbeel, 2012). This is an intuitively solution as the exploration is carried out only in a region of space considered safe. The total policy space is constrained to those policies that are considered safe. A representation of this can be see in Figure 6. However, these constraints may be difficult to formulate in RL algorithms and may therefore be difficult to correctly select the parameter constraints.
- Other Optimization Criteria. Other approaches are based on the use of optimization criteria falling into the area of financial engineering, such as the r-squared, value at risk or the density of return (Kashima, 2007; Morimura, Sugiyama, Kashima, Hachiya, & Tanaka, 2012).

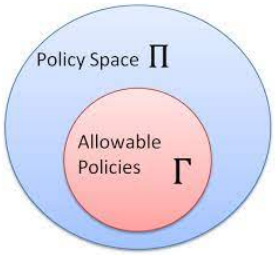


Figure 6: A representation of constraining the total policy space.
Source: Garcıa and Fernández (2015)

Exploration Process During the learning process, the agent makes decisions about which action to choose, either to explore more about the environment or exploit what the agent knows results in a high reward. Part of the goal is therefore to explore the state space efficiently. However, most of these exploration methods do not consider the risk of actions. To avoid risky situations, the exploration process can be modified by including prior knowledge of the task, which can be some through several approaches. As most RL algorithms begin learning with no external knowledge, random exploration of the environment is necessary to gain knowledge. While the agent does learn more about the environment, this random exploration leads the agent to undesirable or irrelevant states, which wastes a significant amount of time. There are two methods of modifying the exploration process to ensure safety: i) through the incorporation of external knowledge and, ii) through the use of risk-direct exploration. Prior external knowledge can be incorporated into the exploration process by:

- Providing Initial Knowledge, which can be used to bootstrap the algorithm, so it is exposed to the most relevant regions of the state and action spaces, eliminating random and risky exploration (Driessens & Džeroski, 2004). The value function approximation is bootstrapped and leads the agent through the more relevant regions of the space from the earliest steps of the learning process. The disadvantage of providing initial knowledge is that the bias introduced may produce sub-optimal policies and the exploration phase following the initial training phase may result in visiting catastrophic states.
- Deriving a policy from a set of Demonstrations, where the external knowledge is not used to bootstrap the learning algorithm, but to derive a policy in an offline, safe manner (Abbeel, Coates, & Ng, 2010; Tang, Singh, Goehausen, & Abbeel, 2010). While this seems like a safe and decent option, the performance is heavily limited by the quality of the demonstrations. It is also unknown how the agent will act if it encounters a state which did not exist in the demonstrations.
- Teacher Advice. The teacher provides actions or information to the agent, which can be requested by the agent (Garcia & Fernández, 2012), or provided but the teacher whenever it feels it is necessary (Quintía Vidal, Iglesias Rodríguez, Rodríguez González, & Vázquez Regueiro, 2013; Thomaz & Breazeal, 2008). The advantages of the teacher advice methods is that the agent is guided through the exploration process by being kept far from catastrophic states from the earliest steps of the learning process. However, approaches where the agent requests advice, only short term risk situations are detected. In approaches

where the teacher provides actions, constant monitoring of the agent by the teacher is needed, which can be time intensive. An example of how a teacher would interact with the agent can be viewed in Figure 7.

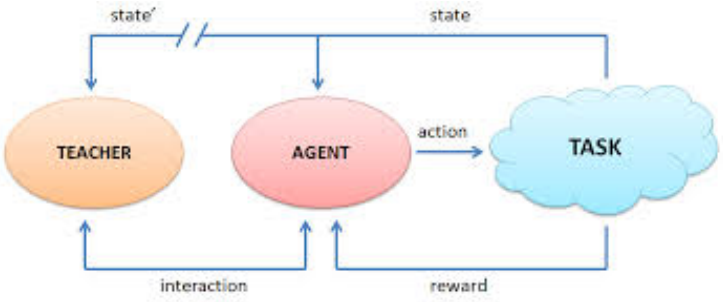


Figure 7: General Teacher-Learner Agent Interaction Scheme.
Source: Garcıa and Fernández (2015)

In Risk-direct Exploration approaches prior knowledge is not incorporated and the classic optimization criterion remains the same. However risk measure is used to determine the probability of selecting different actions during the exploration process (Law, 2005). In these approaches long term risk-situations are detected but it does not detect risky situations from the earlier steps.

2.3.2 Zhu and Zhao

Instead of focusing or task and rewards, the summarizing of Safe Reinforcement Learning by Zhu and Zhao (2021) is more oriented towards how regular RL methods can be modified to ensure safety. According to Zhu and Zhao (2021), autonomous driving applications have high requirements for safety. As Deep Reinforcement Learning methods lack interpretability compared to traditional rule based methods, it becomes difficult to predict if the agent will produce a safe policy. A general strategy to solve this problem is to combine traditional methods to ensure safety in Deep Reinforcement Learning. This can be done in three way:

- Modified Methods: The most common way to enhance safety in the standard DRL algorithms is to modify them, typically by constraining the exploration space. A safety model checker is introduced to identify the set of actions that satisfy the safety constraints at each state through probabilistic prediction, prior knowledge and constraints. Examples of such modified methods are shown in N. Jansen, Könighofer, Junges, and Bloem (2018) and Fulton and Platzer (2018).
- Combined Methods: Combining traditional rule-based methods to enhance safety, but without modifying the learning process. In these methods, a rule-based tracking and safe set controller ensures safety control. These combined methods are used particularly often in autonomous driving, such as in Xiong, Wang, Zhang, and Li (2016) and Shalev-Shwartz, Shammah, and Shashua (2016).
- Hybrid Methods: Here Deep Reinforcement learning is integrated into traditional heuristic search or POMDP planning methods, such as combining DRL and approximate POMDP planning. Studies using these hybrid methods are Bernhard, Gieselmann, Esterle, and Knol (2018) and Pusse and Klusch (2019).

3 Related Work

This chapter gives an overview of the current work related to the subject of the thesis such as driving assistance systems and related research, modelling the drivers' attention as a POMDP and solving such problems using Safe Reinforcement Learning and Recurrent Neural Networks. After this the suggested solution to the distracted driver problem and research questions are given.

3.1 Current Lane Assistance

Driving assistance systems has been a topic of strong scientific interest, research and development (Ayyasamy, 2022; Khan & Lee, 2019). Most car companies already employ some level of driving assistance such as cruise control or lane-keeping (Greenwood, Lenneman, & Baldwin, 2022; Oviedo-Trespalacios, Tichon, & Briant, 2021). As an in-depth research into all the possible possibilities of such driving assistance systems is out of the scope of this research, it will only focus on lane-keeping assistance. To the best of our knowledge, none of the currently commercially available lane-keeping assistants adapt to the driver's attentiveness.

While none of the commercially available lane-keeping assistance systems adapt to the driver's attentiveness, the effect of adapting the intensity of lane-keeping assistance based on the driver's distraction is the subject of recent studies. Examples of this are through haptic lane-keeping assistance, which suggest that continuous lane-keeping assistance is both the preferred style of assistance and improves lane-keeping performance (Benloucif et al., 2019; Roozendaal et al., 2021). These methods, prove to be effective at keeping the driver engaged when attentive while keeping a distracted driver safe at the same time. Therefore we propose a continuous cooperative lane-keeping agent based on the driver's level of attention.

3.2 Lane Assistance Research

For such an lane-keeping assistance system we would need to be able to detect when the driver is distracted. Several studies have utilized continuous video recordings or analysis of the driver to detect distraction using in-vehicle monitoring systems such as head-tracking (S. Zhang & Abdel-Aty, 2022), eye-tracking data (Liang, Reyes, & Lee, 2007; Sri Mounika, Phanindra, Sai Charan, Kranthi Kumar Reddy, & Govindu, 2022) or analysis of the driver using heart rate (Seenivasan, 2023) and even brain activity (Kashevnik et al., 2021; Perera, Wang, Lin, Nguyen, & Chai, 2022). Such data has been processed using Machine Learning methods such as Support Vector Machines (Liang et al., 2007) and Long Short-Term Memory (Wollmer et al., 2011). While these methods have seen success, constant in-vehicle monitoring can be seen as invasive and unpractical as this will require vehicle modifications and extensive data processing. Research suggests that driver distraction can lead to a change in driving performance. Distracted drivers show certain behaviours, such as lane position deviations, infrequent steering wheel movements, and reduced reaction times (Kashevnik et al., 2021; Qin, Li, Chen, Bill, & Noyce, 2019).

Tango and Botta (2013) show that monitoring driving performance measures such as steering wheel angle, heading angle and lateral deviation can be sufficient to detect driver distraction accurately. Other papers have used these metrics, such as speed, longitudinal and lateral acceleration, yaw rate and throttle position in a supervised learning method to detect driver distraction (Chai, Lu, Jiang, Shi, & Zeng, 2021;

Ye, Osman, Ishak, & Hashemi, 2017). Instead of a supervised learning methods a reinforcement learning method will be used in this research. Reinforcement learning had the benefit of learning from its own action and the environment, giving it the ability to continuously increase performance and adapt to each driver individually (Sutton & Barto, 2018).

A driving assistance system that adapts to each individual driver would increase acceptability and trust in the system, which would result in more frequent use of the system (Yi et al., 2019). Therefore, we propose training the lane-keeping assistant system using reinforcement learning with only the car's own metrics as input.

3.3 Solving BMDPs

This section focuses on modeling the unknown latent states in the distracted driver problem and addressing them using either recurrent reinforcement learning or safe reinforcement learning techniques. While the problem has been reformulated as a Block Markov Decision Process (BMDP) based on preliminary results, it's important to note that there is significantly more research on Partially Observable Markov Decision Processes (POMDPs) compared to BMDPs. POMDPs have been the primary framework for modeling problems with latent states, including driver behaviour. Given the similarities in the formalization and the presence of latent driver states in both POMDPs and BMDPs, it's reasonable to assume that solutions developed for POMDPs can be adapted and applied to BMDPs.

To illustrate this, we will draw examples from research on modeling driver problems with unknown states in POMDPs. These examples demonstrate how recurrent reinforcement learning and safe reinforcement learning methods have been utilized in similar contexts. The underlying assumption is that insights and techniques developed for POMDPs can be valuable for addressing similar challenges in BMDPs.

3.4 Distracted Driver as BMDP

As the physiological state of the driver has to be taken into account by the algorithm, we have some unobservable influence on the problem, resulting in a Block Markov Decision Process (BMDP). While BMDPs are relatively less researched compared to POMDPs in terms of unobservable influence on the problem, POMDPs are well suited to model driving problems in which the internal psychological state of a human is important but unknown such as her intend, goals, drowsiness, or attentiveness. However, as both formalizations are similar and involve latent states, we will use POMDP research as an example for modelling driver problems with unknown states.

By employing a POMDP to model the uncertainty about humans' internal states, L. Li, Zhao, and Wang (2022) improved driving policies by taking into account the unknown driving intentions of surrounding vehicles. Using a memory-based Actor-Critic framework called Recurrent Deterministic Policy Gradient, they were able to solve the optimal driving policy under partial observability and generate the optimal trajectory, showing improved performance compared to state-of-the-art algorithms.

Pant et al. (2022) designed a POMDP model that captures the latent cognitive state of the driver. Learning the drivers' level of attention, the driver can be brought into the decision making loop, ensuring timely and safe decision making.

Danesh, Cai, and Hsu (2023) also use a POMDP to model for the uncertainty of human intentions and behaviour. In order to compensate for some of the errors in Monte Carlo sampling, such as missing rare but crucial events that could lead to

potential safety concerns, they propose their own algorithm, LEADER, a critic and generator network. LEADER learns to attend to critical human behaviours during planning, thereby lowering the collision rate while keeping the efficiency and smoothness of driving compared to other state-of-the-art algorithms such as DESPOT and Soft-Actor-Critic (SAC).

As the previous studies show, a POMDP can be used to model driving intentions, latent cognitive state of the driver and to model the uncertainty of human behaviour. However, as our version of the distracted driver problem is a BMDP, we propose modelling the distracted driver problem as a BMDP, which to our knowledge has not been done yet. The following section will introduce methods of solving POMDPs and due to the similarity of BMDPs and POMDPs, these methods will be applied to the BMDP distracted driver problem.

3.4.1 Recurrent Deep Reinforcement Learning for BMDP

In Recurrent Deep Reinforcement Learning methods, an additional memory-component is added to reinforcement learning methods in order to integrate information across time steps. This implementation has shown much success in recent studies in order to solve POMDPs. The following paragraphs show how a memory-component can aid in solving POMDPs.

Meng, Gorbet, and Kulić (2021) test their proposed algorithm Long-Short-Term-Memory-based Twin Delayed Deep Deterministic Policy Gradient (LSTM-TD3) by introducing a memory component to TD3. Using OpenAI Gym environments they compare its performance with other state-of-the-art DRL algorithms such as SAC, Deep Deterministic Policy Gradient (DDPG) and TD3 without a LSTM component. The results show significant advantages of the memory component in addressing POMDPs, including the ability to handle missing and noisy observation data.

Hausknecht and Stone (2015) replaced the first post-convolutional fully-connected layer of a Deep Q-Network with a recurrent LSTM and compared it to resulting Deep Recurrent Q-Network. The two algorithms are compared on Atari games with a flicker condition with a probability of receiving observations. If the probability of receiving observations was one, the Atari game was effectively a MDP and there was no significant difference between the DQN and DQRN. However, by lowering the probability of receiving inputs, creating a POMDP, the harder it becomes for the algorithm to play the game without a recurrent layer, resulting in rapidly declining performance for the DQN network and less dramatic decline for the DQRN due to the LSTM layer.

Ni, Eysenbach, and Salakhutdinov (2022) show that Recurrent Reinforcement Learning can be a strong baseline for solving POMDPs. By comparing six specialized models and their own model in 21 environments, they show through careful hyperparameter tuning and architecture decisions, that augmenting model-free reinforcement learning with a memory-based architectures, such as LSTMs, provides a general approach to solving all many types of POMDPs.

Another study uses a recurrent version of Proximal Policy Optimization (Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017) called GRU-PPO in order to solve POMDPs in the OpenAI benchmark for Deep Reinforcement Learning algorithms called Memory Gym. By comparing the memory-less PPO with a memory-compenent enhanced version, GRU-PPO, training and generalization performances show a strong dependence on memory (Pleines, Pallasch, Zimmer, & Preuss, 2023). According to a recent survey conducted by Ni et al. (2022), it is apparent that most studies utilize an

LSTM as their memory component. Although both GRU and LSTM have their respective advantages and disadvantages, we have decided to utilize LSTM in our study since it has been extensively investigated in recent studies.

As these studies show, the addition of a memory-component to existing Deep Reinforcement Learning algorithms aids in solving POMDPs. Therefore, we hypothesize that adding such a memory-component will aid in solving the potentially distracted driver problem by compensating for the unobservable state of the driver.

3.4.2 Safe Reinforcement Learning for BMDPs

For the solution to the distracted driver problem to be viable in a real-world scenario, safety during training needs to be considered. For this Safe Reinforcement Learning (SRL) can be utilized. SRL is a sub-field of Reinforcement Learning where policies are learned that maximize the expectation of the return in problems as well as ensure reasonable system performance and safety constraints. SRL is a development in the reinforcement learning domain by either adapting the optimization criterion to make it more suitable for risky situations or modify the exploration process, often through prior knowledge (Garcıa & Fernández, 2015). While SRL has been utilized in autonomous driving, it has been under-utilized in the distracted driver problem. The following paragraphs are examples of SRL used in autonomous driving that can be potential solutions for the distracted driver problem.

Examples of recent studies that utilize Safe Reinforcement Learning are by Lin et al. (2023). Here the authors suggest to use the learning from demonstrations paradigm to tackle the problem of sample inefficiency in regular reinforcement learning. However, instead of providing an extensive number of demonstrations, they propose using a teacher-advice with Gaussian process (TAG) mechanism. The teacher provides both an advice action and its associated confidence value, guiding the agent through the environment. Both TAG-PPO and TAG-DDPG algorithms, show improved performance compared to their counterparts lacking the Teacher Advice on several delayed reward and complicated continuous control environments.

Wen, Duan, Li, Xu, and Peng (2020) explores Parallel Constrained Policy Optimization (PCPO), an extension of the Constrained Policy Optimization (CPO) algorithm by Achiam et al. (2017) in two autonomous driving tasks through the addition of a risk function bounded above by risk limit to guarantee policy safety. PCPO and CPO both show improved performance for both the single vehicle lane-keeping task and multi-vehicles at a crossing compared to PPO.

Kong, Zhang, and Xu (2021a, 2021b) shows that CPO outperforms a simple Actor-Critic algorithm in the lane-keeping task. The constraints in CPO are the safety constraints, which is the distance from the center-line of the lane. CPO explores the environment without going out of the lane while the non-constrained Actor-Critic algorithm has to in order to discover the optimal policy. As shown by these studies, the addition of a safety mechanism such as constraints or Teacher Advice can improve performance as well as safety during exploration. Such methods are an important aspect to add to the solution of the distracted driver problem in order to remain viable as a real-world solution. In this study we will use First Order Constrained Optimization in Policy Space (FOCOPS) Y. Zhang et al. (2020) for the constrained optimization criterion. FOCOPS is an improvement upon CPO, where the policy update is project unto an optimal one in terms of the minimum KL divergences, a measure of how much one probability distribution is different from another one.

3.5 Algorithmic Approaches

The choice of TD3 and FOCOPS as the algorithms utilized in this study was motivated by their applicability and success in addressing Partially Observable Markov Decision Problems (POMDPs). Given the structural similarities between POMDPs and Block Markov Decision Problems (BMDPs), it was hypothesized that techniques proven effective in POMDPs could also be relevant for solving BMDPs.

While providing a detailed explanation of these algorithms is beyond the scope of this document, the key aspects of TD3 and FOCOPS are highlighted below.

3.5.1 Algorithmic Approaches: TD3

TD3 was chosen as the algorithm for unconstrained policy optimization due to its widespread use and extensive study in the field of autonomous driving and reinforcement learning (Okuyama, Gonsalves, & Upadhay, 2018; Zhu & Zhao, 2021). TD3 is an extension of the Deep Deterministic Policy Gradient (DDPG) algorithm, which combines elements of Deep Q-Networks (DQN) and Deterministic Policy Gradient (DPG) methods.

DQN, rooted in Q-learning, is designed to handle problems with high-dimensional observation spaces (Mnih et al., 2015). However, it is limited to discrete and low-dimensional action spaces, which is insufficient for solving the distracted driver problem. On the other hand, DPG can handle continuous action spaces but is known to be less stable, especially in challenging problem domains. To address these limitations, DDPG combines the strengths of DPG and DQN in a unified actor-critic architecture and learning algorithm TD3 (Lillicrap et al., 2015).

A common issue with DDPG is the overestimation of Q-values, which can lead to policy instability. TD3 mitigates this problem by concurrently training two Q-functions, denoted as Q_{ϕ_1} and Q_{ϕ_2}, using mean square Bellman error minimization. This involves minimizing the expected difference between the Q-value of a state-action pair and the advantage with respect to the action a. The Bellman equation describes the optimal action-value function:

$$Q^*(s,a) = E_{s' \sim \pi}[R(s,a) + \gamma \max_{a'} Q^*(s',a')] \tag{21}$$

The actions used to create the Q-learning target in TD3 are derived from the target policy $\mu_{\theta_{targ}}$, but with clipped noise added to each dimension of the action. Following the addition of this clipped noise, the target action is further clipped to ensure it falls within the valid action range, which satisfies $a_{Low} \leq a \leq a_{High}$. The resulting target actions are represented as follows:

$$a'(s') = clip(\mu_{\theta_{targ}}(s') + clip(\epsilon, -c, c), a_{Low}, a_{High}), \epsilon \sim N(0, \sigma) \tag{22}$$

This aspect of TD3, known as target policy smoothing, functions as a regularizer within the algorithm. Its purpose is to prevent the Q-function approximator from exploiting a sharp, incorrect peak for certain actions. Instead, it smoothes out the Q-function over similar actions to improve stability.

Moreover, both Q-functions in TD3 share a single target value, determined by whichever of the two Q-functions produces the smaller target value:

$$y(r, s', d) = r + \gamma(1-d) \min_{i=1,2} Q_{\phi_{i,targ}}(s', a'(s')) \tag{23}$$

and then both are learned by regression to this target:

$$L(\phi_i, D) = E_{s,a,r,s',d)\sim D}[(Q_{\phi_i}(s,a) - y(r,s',d)^2], i = 1,2 \tag{24}$$

Algorithm 1 Pseudocode of TD3

Require: initial policy parameter θ, Q-function parameters ϕ_1, ϕ_2, empty replay buffer D.

Require: Target parameters equal to main parameters:
$\theta_{targ} \leftarrow \theta, \phi_{targ,1} \leftarrow \phi_1, \phi_{targ,2} \leftarrow \phi_2$

repeat
  Observe state s and select action a according to
    $a =$clip$(\mu_\theta(s) + \epsilon, a_{Low}, a_{High})$ where $\epsilon \sim N$
  Execute a in the environment
  Observe next state s', reward r, and done signal d to indicate whether s' is terminal
  Store $(s_t, a_t, r_{t+1}, s_{t+1})$ in buffer D
  If s_{t+1} is terminal, reset environment
  if it's time to update **then**
    for j in range(however many updates) **do**
      Randomly sample a batch of transitions:
        $B = (s, a, r, s', d)$ from D
      Compute target actions:
        $a'(s') = \text{clip}(\mu_{\theta_{targ}}(s') + \text{clip}(\epsilon, -c, c)a_{Low}, a_{High}), \epsilon \sim N(0, \sigma)$
      Compute targets:
        $y(r, s', d) = r + \gamma(1-d)\min_{i=1,2} Q_{\phi_{targ,i}}(s', a'(s'))$
      Update Q-functions by one step of gradient descent:
        $\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d)\in B}(Q_{\phi_i}(s,a) - y(r,s',d))^2$ for i = 1,2
      if j mod policy delay = 0 **then**
        Update policy by one step of gradient ascent using:
          $\nabla_\theta \frac{1}{|B|} \sum_{s\in B} Q_{\phi_1}(s, \mu_\theta(s))$
        Update target networks with:
          $\phi_{targ,i} \leftarrow \rho\phi_{targ,i} + (1-\rho)\phi_i$ for i = 1,2
          $\theta_{targ} \leftarrow \rho\theta_{targ} + (1-\rho)\theta$
      end if
    end for
  end if
until convergence

Using the smaller Q-value for the target and performing regression towards it helps mitigate the issue of overestimation in the Q-function. This helps stabilize the learning process and leads to better policy optimization.

Lastly the policy the policy is optimized by maximizing Q_{ϕ_1}:

$$\max_\theta E_{s\sim D}[Q_{\phi_1}(s, \mu_\theta(s)] \tag{25}$$

In TD3, the policy is updated less frequently than the Q-functions, which helps dampen the volatility that can arise in DDPG due to the way a policy update affects the

target. The introduction of a memory component to TD3, as demonstrated by Meng et al. (2021), provides significant advantages compared to other deep reinforcement learning methods in PODMPs. The pseudocode of TD3 can be viewed in Algortihm 1. The author's PyTorch implementation used and adapted for this thesis has been taken from `https://github.com/sfujim/TD3`. The hyperparameter settings for TD3 can be seen in Table 5.

3.5.2 Algorithmic Approaches: FOCOPS

There are many examples of algorithms to solve a Constrained Markov Decision Process, both model-based (Paternain, Chamon, Calvo-Fullana, & Ribeiro, 2019; Yu, Yang, Kolar, & Wang, 2019) and model-free RL (Achiam et al., 2017; Y. Liu, Halev, & Liu, 2021) both with Actor-Critic and Policy Gradient methods (Gu et al., 2022). As far as we know no Safe Reinforcement Learning algorithm has been applied to the distracted driver problem. First Order Constrained Optimization in Policy Space (FOCOPS) is chosen because it both guarantees constraint satisfaction throughout training, works for arbitrary policy classes such as neural networks and because of its ease of implementation. The pseudo-code of FOCOPS can be viewed in Algorithm 2. The authors implementation used and adapted for this thesis was taken from `https://github.com/ymzhang01/focops`. The hyperparameter settings for FOCOPS can be seen in Table 5.

FOCOPS is designed to maximize an agent's overall reward while simultaneously ensuring that the agent satisfies a set of cost constraints. This involves solving a Constrained Markov Decision Process (CMDP), as described by Equation 13. Instead of directly solving this equation, FOCOPS adopts a two-step approach:

- Policy Update in Nonparameterized Space: FOCOPS first employs data generated by the current policy to find the optimal update policy. This is achieved by solving a constrained optimization problem within the nonparameterized policy space.
- Projection into Parametric Space: Once the optimal update policy is determined, FOCOPS then projects this policy back into the parametric policy space.

The key advantage of this approach is that it provides an approximate upper bound for worst-case constraint violation throughout the training process. Additionally, it is relatively simple to implement because it relies on first-order gradient information for training iterations, as opposed to second-order Hessian methods (Tan & Lim, 2019; Y. Zhang et al., 2020).

Policy Update in Nonparameterized Space In the first step of solving equation 13, a slightly different optimization problem is considered, where the parameter of interest is the non-parameterized policy π and not the policy parameter. Say some policy π_{θ_k} is a feasible solution. Then the optimal policy takes the form of:

$$\pi^*(a|s) = \frac{\pi_{\theta_k}(a|s)}{Z_{\lambda,\upsilon}(s)} \exp(\frac{1}{\lambda}(A^{\pi_{\theta_k}}(s,a) - \upsilon A_C^{\pi_{\theta_k}}(s,a))) \tag{26}$$

Where $Z_{\lambda,\upsilon}(s)$ is the partition function that insures that the equation is a valid probability distribution and λ and υ are solutions to the optimization problem where we minimize λ, as to maximise reward-advantage parameter and minimize υ, as to minimise the cost-advantage parameter.

In the process of solving for the optimal policy in the CMDP, it's important to note that this optimal policy, denoted as π^*, tends to assign high probabilities to areas of the state-action space that yield high returns. However, this preference for high returns must be balanced with a penalty term that accounts for the cost advantage.

Projection into Parametric Space After solving the CMDP equations, π^* is not necessarily restricted to the parameterized policy space. Consequently, it becomes impractical to sample directly from this policy. To address this, an important step is to project the optimal policy back into the parameterized policy space. This is accomplished by minimizing a loss function, which helps align the optimal policy with the parameterized policy. The specific form of this loss function is given by the equation:

$$L(\theta) = \mathbb{E}_{s\sim\pi_{\theta_k}}[D_{KL}(\pi_\theta||\pi^*)[s]] \tag{27}$$

In the context of projecting the optimal policy π^* back into the parameterized policy space, a projected policy π_θ is employed to approximate this optimal update policy. To achieve this, first-order methods are utilized to minimize the loss function.

The gradient of the loss function, which represents the direction and magnitude of change needed to minimize the loss, is a crucial component in this optimization process. The gradient for the loss function is expressed as:

$$\nabla_\theta L(\theta) = \mathbb{E}_{s\sim\pi_{\theta_k}} \nabla_\theta[D_{KL}(\pi_\theta||\pi^*)[s]] \tag{28}$$

Where:

$$\nabla_\theta[D_{KL}(\pi_\theta||\pi^*)[s] = \nabla_\theta[D_{KL}(\pi_\theta||\pi_{\theta_k})[s] - \frac{1}{\lambda}\mathbb{E}_{a\sim\pi_{\theta_k}}[\frac{\nabla_\theta\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}(A^{\pi_{\theta_k}}(s,a) - \upsilon A_C^{\pi_{\theta_k}}(s,a))] \tag{29}$$

As this method employs first-order optimization, its accuracy is mainly guaranteed in the vicinity of the initial condition, where $\pi_\theta = \pi_{\theta_k}$. To enforce this constraint, an indicator function is added.

$$I(s_j) := \mathbf{1}_{D_{KL}(\pi_\theta||\pi_{\theta_k}[s_j]\geq\delta} \tag{30}$$

Consequently, states that are sampled with KL-divergence values that are too large are rejected from the gradient update. The resulting sample gradient update term is thus:

$$\tilde{\nabla}_\theta L(\theta) \approx \frac{1}{N}\sum_{j=1}^{N}[\nabla_\theta D_{KL}(\pi_\theta||\pi_{\theta_k})[s_j] - \frac{1}{\lambda}\frac{\nabla_\theta\pi_\theta(a_j|s_j)}{\pi_{\theta_k}(a_j|s_j)}(\tilde{A}(s_j,a_j) - \upsilon\tilde{A}_C(s_j,a_j))]I(s_j) \tag{31}$$

During training an early stopping criteria is used to prevent trust region constraint violations for the new updated policy:

$$\frac{1}{N}\sum_{i=1}^{N} D_{KL}(\pi_{theta}||\pi_{\theta_k})[s_i] < \delta \tag{32}$$

The constraints in the lane-keeping problem will be set at a certain distance from the centerline to ensure that the vehicle never comes near the edge of the road. It's

worth noting that FOCOPS has not been used in combination with an LSTM layer before to solve a POMDP or BMDP problem. This combination of FOCOPS and LSTM represents a novel approach to addressing the lane-keeping problem in a BMDP framework.

Algorithm 2 Pseudocode of FOCOPS

Initialize: Initial policy π_{θ_0}, Value networks $V_{\phi_0}, V_{\psi_0}^C$.
while stopping criteria not met **do**
  Generate trajectories $\tau \sim \pi_{\theta_k}$
  Estimate value and action-value functions V and Q using:
  $V^\pi(s) := \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t)|s_0 = s]$
  $Q^\pi(s,a) := \tau \sim \pi[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t)|s_0 = s, a_0 = a]$
  Estimate cost value and cost action-value functions:
  $V_C^\pi(s) := \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty}\gamma^t C(s_t,a_t)|s_0 = s]$
  $Q_C^\pi(s,a) := \tau \sim \pi[\sum_{t=0}^{\infty}\gamma^t C(s_t,a_t)|s_0 = s, a_0 = a]$
  Estimate C-returns, advantage and cost advantage functions:
  $J_{C_i}(\pi) := \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty}\gamma^t C_i(s,a)]$
  $A^\pi(s,a) := Q^\pi(s,a) - V^\pi(s)$
  $A_C^\pi(s,a) := Q_C^\pi(s,a) - V_C^\pi(s)$
  Update cost penalty term ν using:
  $\nu \leftarrow$proj $[\nu - \alpha(b - J_C(\pi_{\theta_k}))]$
  for K epochs **do**
    for each minibatch **do**
      Update value networks by minimizing MSE of $V_{\phi_k}, V_{\phi_k}^{target}$ and $V_{\psi_k}^C, V_{\psi_k}^{C,target}$ using $\frac{1}{N}\sum_{i=1}^{N}(V_i - V_i^{target})^2$
      Update policy network using:
      $\tilde{\nabla}_\theta L(\theta) \approx \frac{1}{N}\sum_{j=1}^{N}\nabla_\theta D_{KL}(\pi_\theta||\pi_{\theta_k})[s_j] -$
      $\frac{1}{\lambda}\frac{\nabla_\theta \pi_\theta(a_j|s_j)}{\pi_{\theta_k} a_j|s_j)}(\tilde{A}(s_j,a_j) - \nu\tilde{A}_C(s_j,a_j))]I(s_j)$
      if $\frac{1}{N}\sum_{j=1}^{N} D_{KL}(\pi_\theta||\pi_{\theta_k})[s_j] > \delta$ **then**
        Break out of inner loop

3.6 Problem Overview and Research Questions

3.6.1 Problem Overview

The problem in this thesis is assisted lane-keeping for a potentially distracted driver in a shared control scheme. The agent acts as an assistant to the human driver by keeping the car centred in the lane. The driver is simulated using a driver model that periodically becomes distracted. During such periods of inattentiveness, driving performance will decline. The state of the driver is unknown to the agent, which is has to be inferred from the driver's steering actions and the car's metrics. The dynamics of the car are simulated using The Open Racing Car Simulator (TORCS). A BMDP model is employed to account for the uncertainty about the driver's attentiveness.

3.6.2 Research Question

The main research aim to be answered in this thesis is finding an optimal solution to the distracted driver problem. Solutions to the problem will be compared by their lane-keeping ability on the highway. As both safety and the unobservable state of the driver pose distinct problems to the algorithm, our proposed solution is extending the safe reinforcement learning algorithm FOCOPS with a LSTM layer, called FOCOPS-LSTM. Our main research question is therefore formulated as follows:

- Can the recurrent safe reinforcement learning in the form of FOCOPS-LSTM be considered a viable solution to the distracted driver problem modelled as a Block Markov Decision Process?

Our implementation will be compared to an off-policy, state-of-the-art, commonly used algorithm, TD3 and its extension TD3-LSTM. The main research aim is split into three sub-questions:

- What are the advantages of using a memory-based DRL method compared to a memory-less method in the distracted driver problem?
- How do different safe and unsafe reinforcement learning methods compare to each other in the distracted driver problem?
- How does our proposed method solution FOCOPS-LSTM compare to another comparable state-of-the-art algorithm, TD3-LSTM?

4 Methodology

This chapter formally defines the POMDP that is used to model the shared-control lane-keeping problem that is considered in this thesis. Furthermore, the methods used to solve the problem are explained in detail. First Section 3.1 gives an overview of the simulation The Open Racing Car Simulator (TORCS) that is used to simulate the dynamics of a car driving on a highway and the driver model used. To answer the research questions, several algorithms will be utilized which are explained in Section 3.2.

4.1 Overview

The issue addressed in this thesis pertains to a shared control lane-keeping task in which both the driver and the assisting agent collaborate to maintain the vehicle centred within the lane. As the driver model becomes periodically distracted, the agent must adapt to the driver's latent state. It is assumed that the driver model demonstrates near-optimal driving behaviour; hence, the agent need not intervene as long as the driver remains attentive. Nevertheless, when the driver becomes distracted, both horizontal and lateral driving will alter, necessitating the agent's intervention to achieve the desired driving behaviour. Since the agent is unaware of the driver's attentive state, it must infer this from the driving behaviour to keep the vehicle centred. This inference can be made based on the information gathered over time. Using this estimation, the agent determines the actions the driver is likely to take and selects the appropriate actions itself.

Rather than conducting experiments with actual humans and cars, which can be both time-consuming and costly, simulation models are employed for both the human and the vehicle in the experiments. Three components collaborate in these simulations. Firstly, there is the simulation software TORCS (Wymann et al., 2000), which simulates the car's dynamics. Secondly, there is the driver model, simulating a human driver who becomes periodically distracted. Finally, there is the agent, which receives inputs from TORCS, such as sensor data, car metrics, and driver inputs, and returns the actions it believes will maximise the reward.

An important note to the Methodology and thesis as a whole is that much inspiration has been taken from another Master Thesis by Jokke Jansen J. Jansen (2021). There are many similarities between the two thesis such as using the TORCS envrionment, the driver model, the highway settings and the POMDP formalization. However, we improve upon it by employing the more natural continuous instead of discrete sensory information and steering actions. Additionally we compared several distinct algorithms and as such, test the assumption that the environment is truly a POMDP environment.

4.2 TORCS

The Open Racing Car Simulator is a modern, modular and highly portable multi-agent car simulator. Its high degree of modularity and portability renders it ideal for artificial intelligence research. Over 300 research papers have been written employing TORCS as a basis, primarily for testing AI algorithms, including driver attention and stress (Wymann et al., 2000). The simulator is relatively simple, but it handles all basic elements of vehicle dynamics. This includes basic properties of vehicle systems such as mass, rotational inertia of the car, wheels, and other components, mechanical

details such as suspension types and links as well as dynamic and static friction profiles of tires for different road types.

In TORCS, the participating players are referred to as 'robots'. They are loaded as external modules in TORCS. This means that new artificially intelligent agents can be developed independently and they only have to satisfy the basic API requirements. The robots have the opportunity to interact with the simulation every 0.02s. The default interface is through a low-level API which can provide detailed information about the race status to the robot, exact position, distance from the edge of the track, etc. However, there are many parts of the simulation state to which the robots have no direct access. With only a single car on the track, the overall problem can be formalised as a partially observable Markov decision processes. An adaptation to OpenAI gym named gym-TORCS was utilized which enables Python usage taken from `https://github.com/ugo-nama-kun/gym_torcs`. Gym-TORCS has been used in many reinforcement learning papers for autonomous driving Ly and Akhloufi (2020); Santara et al. (2021).

As TORCS is based on racing tracks, for the cooperative lane-keeping problem we will have to implement our own custom track. To simulate a highway track we will have a single lane with constant width of 3m with only moderate bends. There are no other cars and the track is completely flat. An example of a highway track is given in Figure 8. If this proves too easy for the algorithm, we can implement sharper bends or a more narrow track.

In order to have our code interact with the TORCS environment a python wrapper for RL experiment with the simple, similar interface compatible with OpenAI-gym environments called Gym-TORCS was used.

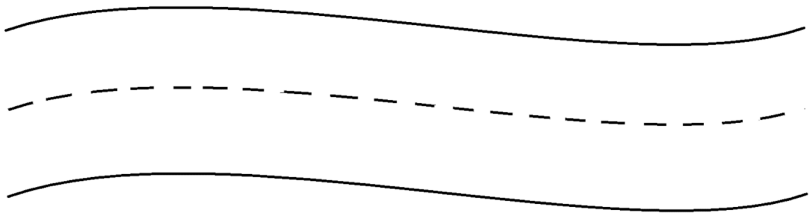


Figure 8: An example of the custom highway used in TORCS.

4.2.1 Driver Model

To emulate a human driver experiencing periodic distractions, a driver model is constructed to interface with the TORCS simulator. The model governs the driver's actions, the onset of distractions, and their duration. Figure 9 provides a schematic representation of the interaction between the driver model, the agent, and TORCS.

The driver's state comprises three variables: current attentiveness (attentive or distracted), time until the driver regains attention after being distracted, and driving behaviour, which consists of two action inputs – steering input and speed regulation input. The time until an attentive driver becomes distracted is randomly chosen between 15-20 seconds, while the duration of distraction ranges from 5-15 seconds. Each time

the driver becomes distracted or attentive, the duration is randomly selected. While this seems unrealistic compared to a human driver, a more realistic driver model in terms of attention span would also mean that our algorithm will have less training data on the distracted driver state and will also be tested much less often. When the driver becomes distracted, the last executed steering action is repeated, and the speed input is steadily decreased until the driver regains attention, simulating realistic driving behaviour.

If the task of finding the optimal solution was too easy, it would not be possible to clearly see which algorithm works better. Therefore, to make it more difficult for the algorithms some noise is added by multiplying the previous steering with a factor randomly chosen from a range of 0.85 to 0.95 if the steering action was to the left. If the previous steering action was to the right side of the road, it is multiplied by factor randomly chosen from a range of 1.05 and 1.15.

4.3 Problem Formulation

In this section, the cooperative lane-keeping control is formulated as a POMDP. The subsequent subsections will define the state space, state transition probabilities, action space, rewards, and observation space.

4.3.1 State Space: S

The overall state space comprises all possible states in which the agent can exist within its environment. Given that the vehicle and driver are distinct entities, the state space can be partitioned into the possible states of the vehicle and the possible states of the driver. Consequently, the POMDP's state space consists of all potential combinations of the driver's and vehicle's states. The state transition probabilities are not provided but are implicitly defined by the car dynamics determined by TORCS and must be learned by the agent.

As TORCS encompasses fundamental properties of vehicle systems, including mass, inertia, suspension types, and tyre profiles, enumerating them all would be excessively exhaustive. The car states that the agent can measure and modify its driving behaviour according to include the car's position on the track, velocity, yaw angle, and acceleration. Although other attributes, such as wheel friction, drag, and engine pressure, affect the vehicle, the agent cannot measure these. The state values are continuous.

The lane position of the vehicle spans a continuous interval between -1.5 (left lane border) and +1.5 (right lane border), with zero indicating that the car is perfectly in the lane's centre, as desired. The car's yaw angle, defined over an interval of [-90, +90] degrees with respect to the track's direction, implies that at -90 degrees, the car is heading directly towards the left lane border, and at +90 degrees, it is heading straight for the right lane border. A value of zero indicates that the car is moving in the exact same direction as the track, as desired. The vehicle's speed, which falls within a continuous interval of [-100, +100] km/h, signifies that a value of zero means the vehicle is travelling at the desired 100 km/h.

The total state space is therefore a combination of the vehicle's state, determined by TORCS and the driver's state, determined by the driver model.

4.3.2 Action Space: A

The action space encompasses the actions that the agent can execute to maximise the vehicle's lane centredness. The human driver and the agent share control of the vehicle, which is why the combined action is influenced by both the agent and the driver model. Since the driver should remain active during driving to ensure timely reaction to dangerous situations if necessary, the lane-assistant system should exert reduced control as long as the driver is not distracted. However, the system must also intervene when the vehicle is too far from the lane's centre to ensure safety. Therefore, both steering inputs from the agent and driver model will be averaged. The turn input of the driver $t_{driver} \in [-45, +45]$ and the turn input of the agent $t_{agent} \in [-45, +45]$ are added to the car $t_{car} \in [-45, +45]$ and averaged according to the following equations:

$$\begin{aligned} t_{car} &= (t_{driver} + t_{agent})/2 \\ s.t. t_{car} &[-45, +45] \end{aligned} \tag{33}$$

A maximum steering angle of 45 degrees is set, where a steering action of -45 means steering fully to the left and a steering action of +45 means steering fully to the right. In addition to the steering input, the agent and driver will also control the vehicle's speed. The desired speed is set at 100 km/h. Once more, the agent's input will depend on how far off the driver is from the desired speed. The speed input of the driver $s_{driver} \in [0, 100]$ and the speed input of the agent $s_{agent} \in [0100]$ are added to the car $s_{car} \in [0, 100]$ and averaged according to the following equations:

$$\begin{aligned} s_{car} &= (s_{driver} + s_{agent})/2 \\ s.t. s_{car} &[0, 100], \end{aligned} \tag{34}$$

All the actions taken by the agent are shown in Table 1

Actions	Description	Unit	Range	Symbol
Steering	The steering output by the agent	Degrees	-90, 90	a
Velocity	The velocity output by the agent	Km/h	0, 100	s

Table 1: Actions taken by the agent

4.3.3 Transition Function: $T : S \times A \longrightarrow \Delta(S)$

The transition function states that for each action a and state s, $T(s, a)$ is the probability distribution over states that the system may transition into when taking action a in state s. The transition function is not explicitly given to the agent but are defined by TORCS' dynamics and the driver model. The agent will have in implicitly learn the transition function in order to take the action with the highest return value.

4.3.4 Reward Function: R

The agent must learn optimal actions based on the reward function R. This reward function is derived from the car's distance to the centre of the lane, the vehicle's speed, and its yaw angle. The driver's attentiveness is not considered in the reward function, as the agent must infer this from the driver's behaviour alone. However, the attentiveness of the driver influences driving performance, so the quality of the agent's

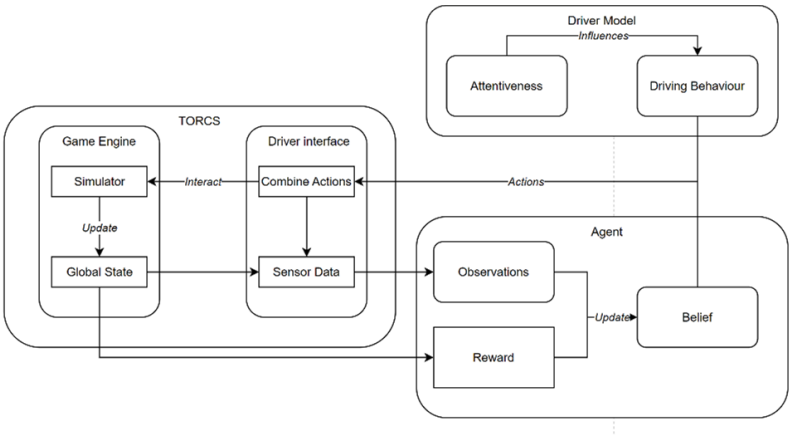


Figure 9: A diagram showing the interactions between the driver, agent and TORCS environment

estimate correlates with the rewards received. When the driver is attentive and drives optimally, any action taken by the agent results in sub-optimal driving and a penalised reward. Consequently, the agent learns when not to intervene with the driver, granting them more autonomy. However, if the driver is inattentive, driving performance deteriorates, and the agent must act to receive the highest possible reward. To further discourage the agent from taking unnecessary actions, a small penalty is applied for each action taken. According to a meta-study by Zhu and Zhao (2021), these are all common parameters for the reward function in autonomous driving systems. The reward is calculated according to the following equation:

$$R = s_{car}(\cos\Theta - |d| - \sin|\Theta|) - 0.1(s_{agent} + t_{agent}) \tag{35}$$

Here, d represents the distance from the centre of the lane, Θ is the vehicle's yaw angle, and s_{car} is the car's speed. The agent is rewarded for the speed it travels parallel to the lane and the agent is penalised if the vehicle is to the left or right of the lane's centre, the speed it travels perpendicular to the lane. To stimulate the agent to only interact when it is necessary and if it does interact, to do so minimally, the total reward of the agent is also decreased by a small amount relative to the size of the actions taken by the agent. The maximum reward is achieved by being in the lane's centre, parallel to the sides, maintaining the desired speed, and minimising the inputs from the agent.

4.3.5 Cost Function

In addition to the reward function, some safe reinforcement learning algorithms, such as CPO and FOCOPS, utilise a cost function that restricts the set of allowable policies. For the distracted driver problem, two constraints will be added to the POMDP problem. The cost function will limit policies that permit the vehicle to cross a specific distance from the centre-line by setting a cost boundary. The constraints also appear in the reward function as penalties; satisfying these constraints results in a higher reward. The agent's task is to maximise the reward function while minimising the cost

functions. Optimising the reward and cost functions will be part of the experiments. The cost function for crossing the cost boundary distance Δ, is calculated according to the following equations:

$$C(s) = \begin{cases} 1, & \text{if } \Delta \geq \delta \\ 0, & \Delta < \delta \end{cases} \tag{36}$$

An important note for setting the parameter for cost boundary δ is that the distance from the centerline in TORCS is measured from the center of the car. At the same time, the vehicle crashes is the edge of the vehicle hits the side of the road. Therefore, in order to find the optimal distance for the cost boundary parameter δ, several experiments have been conducted as can be seen in Section 5.

4.3.6 Observations: Ω

The observation space Ω encompasses all the observations made by the agent. The agent receives sensory information about the car's current lane centredness and yaw angle. Additionally, the driver's last actions are observed. The agent must estimate the driver's most likely next action by considering the history of past observations and learn which action to take to maximise the reward. While more observations are available using the TORCS environment, in order to make it difficult enough for the algorithm, only a few have been chosen. All the observations made by the agent are given in Table 2. In order to conform to a POMDP, the driver's attentive state is unknown to the agent.

Observation	Description	Unit	Range	Symbol
Lane Centeredness	The distance that the vehicle is from the centerline	Meters	-1.5, 1.5	d
Yaw Angle	The angle that the vehicle is compared to the centerline	Degrees	-90, 90	θ
Steering Angle	The steering input from the driver	Degrees	-1, 1	a
Velocity	The velocity input from the driver	km/h	0, 100	s

Table 2: Observations for the agent

4.3.7 Observation Function: $O : S \times A \longrightarrow \Pi(\Omega)$

The observation function gives a probability distribution over possible observations for each resulting state and action. Each input given to the agent is therefore based on the state of the driver model, the vehicle and the action taken by the agent. As the true state is unknown to the agent, the next observation will also be unknown to the agent due to the latent states. The observation function is unknown to the agent.

4.4 Block MDP Formulation

While the official tuples describing a BMDP uses different notations, the POMDP and BMDP tuples are essentially the same. Obviously the state space S, action space A and reward function R are the same for both formulations. While POMDP uses observation space Ω and observation function O to describe the transition from the true state to the observations of the agent, BMDP describes this using context X and emission function q. Also similar to how transition function T determines how the current state and action determine the next state in POMDPs, the latent transition distribution p does this for BMDP. The major difference between the two formulations

is Assumption 1, which states there exists an implicit decoding function $f : X \longrightarrow S$. As this decoding function is implicit no changes need to be applied to the experimental setup or models to adapt the BMDP model.

4.5 Solution approach

To solve the distracted driver problem, it is divided into two sub-problems. The first sub-problem concerns the driver's unobservable state. A distracted driver will have worse performance than an attentive one, but the agent cannot directly detect this. The agent can only partially observe the real state and must infer the unobservable from what it can detect. To tackle this issue, Recurrent Neural Networks (RNNs) are employed, as they can identify dependencies of inputs over arbitrary time periods, aiding in dealing with unobservable environments.

The second sub-problem relates to safety. As the agent explores possible actions, it should avoid actions that could lead to disastrous states, such as the vehicle going off-lane. To tackle this issue, safe reinforcement learning will be employed. Finally, by combining these two approaches, the agent can effectively handle the distracted driver problem. It can estimate the driver's attentiveness based on their behaviour and learn to intervene when necessary while prioritising safety. This ensures that the vehicle maintains optimal driving performance and safety, even when the driver becomes periodically distracted. For the algorithmic choices Deep Reinforcement Learning methods First Order Constrained Optimization in Policy Space (FOCOPS) and Twin Delayed Deep Deterministic Policy Gradient (TD3) have been chosen.

4.5.1 Processing Observations

The process of handling observations involves interactions between the TORCS environment, the LSTM network, and the reinforcement learning agent. There are several steps to this process, which are visually represented in Figure 10

- TORCS Environment: Within the TORCS environment, the agent operates and receives observations. These observations encompass various aspects of the current state, such as the car's position, speed, and relevant environmental information. These observations serve as essential input for the agent's decision-making.
- LSTM Network: The observations from the TORCS environment undergo processing through an LSTM network, well-suited for handling sequential data. In this context, it serves to process both known states of the environment and the unknown attentive state of the driver. The LSTM's key function is to capture temporal dependencies in the data, enabling the agent to retain and utilize historical information to inform its present decisions.
- Reinforcement Learning Agent: The output of the LSTM network consists of encoded observations, which are subsequently fed into the reinforcement learning agent. The primary role of the RL agent is to select actions that maximize the expected cumulative reward. To do so, it relies on the encoded observations, which encapsulate information from the LSTM regarding the current state as well as past states and contexts.
- Action Execution: Based on its policy, representing a strategic approach to action selection, the agent makes a decision on the next action to take. This chosen action is then executed within the TORCS environment.

- Observation Processing Loop: The described sequence forms a continuous loop. The RL agent continually interacts with the environment, receiving new observations, selecting actions, and observing rewards. This iterative process enables the agent to learn and adapt its policy over time, aiming for improved performance.
- Replay Buffer: As the agent engages in interactions with the environment, it accumulates data, including observations, actions, and corresponding rewards. These data points are systematically stored within a replay buffer, which functions as a memory reservoir for the agent. Subsequently, this replay buffer is employed during the neural network training phase.

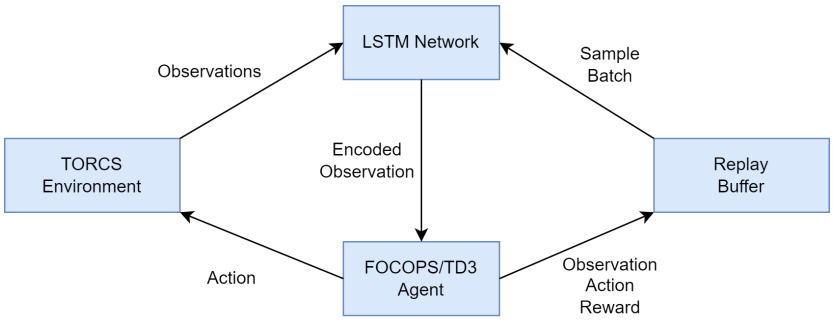


Figure 10: The process of handling observations between the TORCS environment, LSTM network and the reinforcement learning agent.

4.5.2 Flicker condition

As Meng et al. (2021) and Hausknecht and Stone (2015) have shown that a recurrent Deep Reinforcement Learning method shows significant improved performance in a POMDP compared to a MDP, the same method of introducing a flicker condition will be implemented to show that the distracted driver environment is not a POMDP, but and BMDP. As Figure 11 shows the performance declines rapidly for the non-recurrent neural network while performance for the recurrent neural network declines linearly with the probability of receiving inputs. A similar method is used to prove that the distracted driver problem is not a POMDP.

4.5.3 Memory-based DRL

Recurrent Neural networks are capable of learning features and long term dependencies from sequential and time-series data. RNNs are made of high dimensional hidden states with non-linear dynamics with one or more feedback loops. The structure of the hidden states work as a memory of the network and the state of the hidden layer at a time is conditioned on its previous state. This structure enables the RNN to store, remember and process complex past signals for long time periods. RNNs can map an input sequence to the output sequence at the current timestep and predict the sequence in the next timestep. However, the memory produced from the recurrent connections

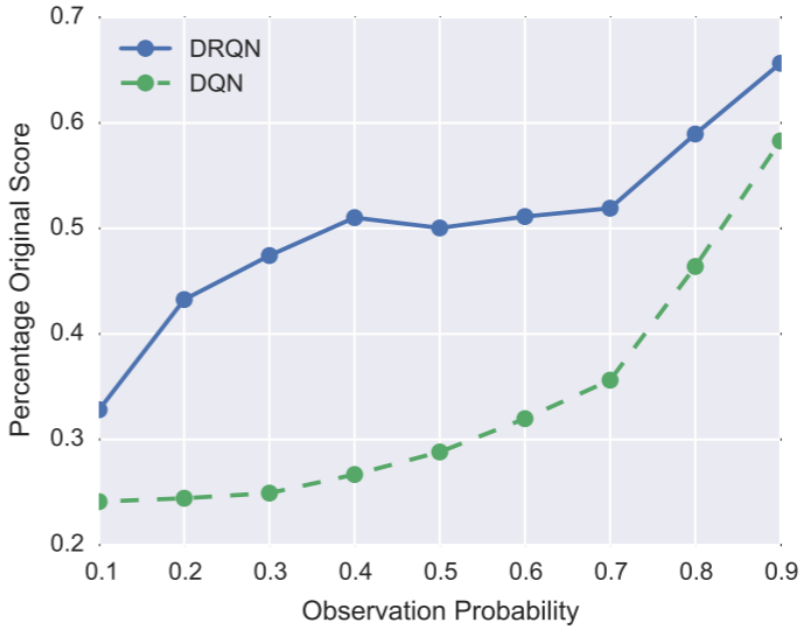


Figure 11: Performance of DRQN compared to DQN on flicker condition.
Source: Hausknecht and Stone (2015)

can severely be limited by the algorithms employed. The network can fail to learn long-term sequential decencies in the data due to exploding or vanishing gradients during training. The Long-Short Term Memory (LSTM) RNNs were designed to tackle this problem (Salehinejad, Sankar, Barfett, Colak, & Valaee, 2017) and an example of such a LSTM can be seen in Figure 12. The hyperparameter settings for TD3 can be seen in the Appendix in Table 5.

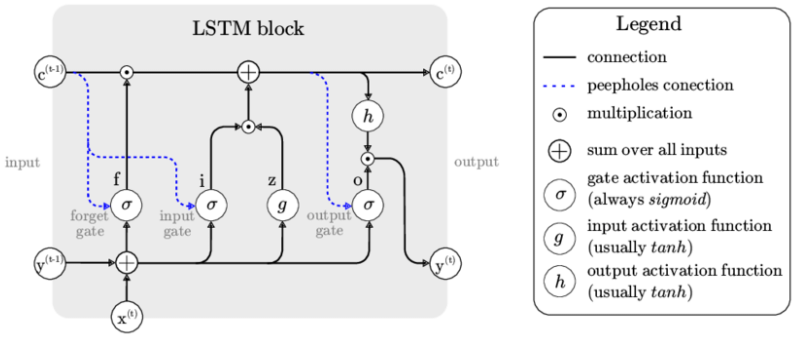


Figure 12: Example of a LSTM cell.
Source: Van Houdt et al. (2020)

LSTM changes the structure of hidden units to memory cells, in which their inputs and outputs are controlled by gates. These gates control the flow of information to hidden neurons and preserve extracted features from previous timesteps. A typical LSTM cell is made of input gates, forget gates, output gates and a cell activation component, which can be seen in Figure 12. These units receive the activation signals from different sources and control the activation of the cell. The LSTM gates can prevent the rest of the network from modifying the contents of the memory cells. This allows LSTM networks to process data with complex and separated interdependencies and excel in a range of sequence learning domains. The input gate of LSTM is defined as

$$g_t^i = \sigma(W_{Ig^i} x_t + W_{Hg^i} h_{t-1} + W_{g^c g^i} g_{t-1}^c + b_{g^i}) \tag{37}$$

Where W_{Ig^i} is the weight matrix from the input layer to the input gate, W_{Hg^i} is the weight matrix from hidden state to the input gate, $W_{g^c g^i}$ is the weight matrix from cell activation to the input gate and b_{g^i} is the bias of the input gate. The forget gate is defined as:

$$g_t^f = \sigma(W_{Ig^f} x_t + W_{Hg^f} h_{t-1} + W_{g^c g^f} g_{t-1}^c + b_{g^f}) \tag{38}$$

Where W_{Ig^f} is the weight matrix from the input layer to the forget gate,W_{Hg^f} is the weight matrix from hidden state to the forget gate, $W_{g^c g^f}$ is the weight matrix from cell activation to the forget gate and b_{g^f} is the bias of the forget gate. The cell gate is defined as:

$$g_t^c = g_t^i \tanh(W_{Ig^c} x_t + W_{Hg^c} h_{t-1} + b_{g^c}) + g_t^f g_{t-1}^c \tag{39}$$

Where W_{Ig^c} is the weight matrix from the input layer to the cell gate, W_{Hg^c} is the weight matrix from hidden state to the cell gate and b_{g^c} is the bias of the forget gate. The output gate is defined as:

$$g_t^o = \sigma(W_{Ig^o} x_t + W_{Hg^o} h_{t-1} + W_{g^c g^o} g_{t-1}^c + b_{g^o}) \tag{40}$$

Where W_{Ig^o} is the weight matrix from the input layer to the output gate, W_{Hg^o} is the weight matrix from hidden state to the output gate, $W_{g^c g^o}$ is the weight matrix from cell activation to the output gate and b_{g^o} is the bias of the forget gate. Finally the hidden state is computed as:

$$h_t = g_t^o \tanh g_t^c \tag{41}$$

LSTMs have been implemented in many solutions for various POMDPs such as flickering Atari games (Hausknecht & Stone, 2015), online driver distraction detection, and various other POMDP environments (Meng et al., 2021; Ni et al., 2022; Wierstra, Foerster, Peters, & Schmidhuber, 2007). In the distracted driver problem the agent has to learn when the driver is distracted or not based on past experiences which are arbitrarily long in the past. Therefore it stand to reason that the addition of a memory unit in the network should be helpful in a BMDP as well. In order to discover the advantages of using memory components such as LSTMs in the distracted driver problem, memory-based Deep Reinforcement Learning methods will be compared to their base methods. Figure 13 visualizes how the LSTM layer is added to TD3 and FOCOPS.

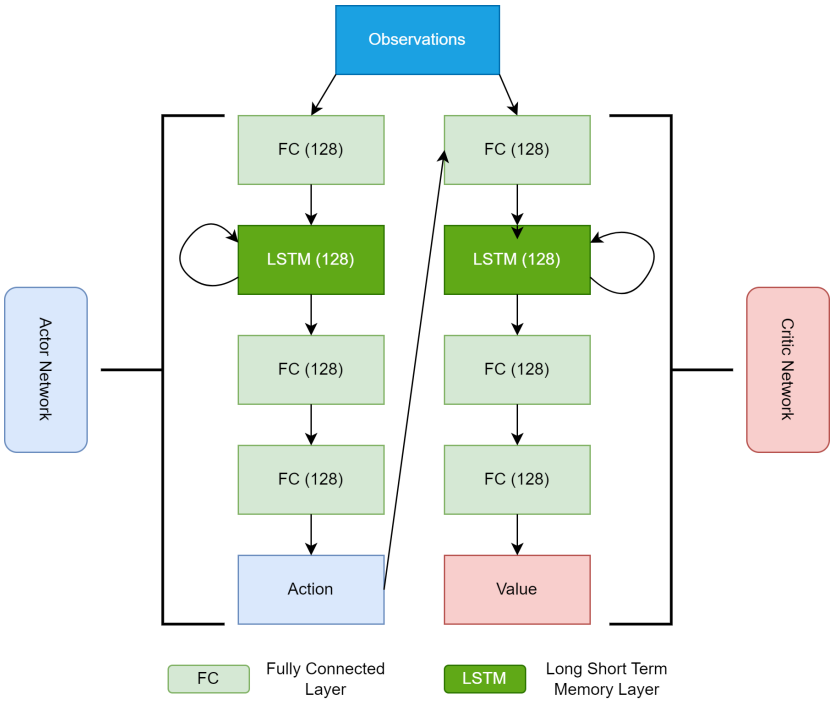


Figure 13: Network structure of the Actor and Critic Networks. Arrows show how the layers are connected with a loop on the LSTM layers to indicate recurrence.

4.6 Experimental Setup

To address the research questions, a comparative analysis will be conducted among the algorithms to evaluate their lane-keeping performance in the highway environment. As the driver model becomes periodically distracted, the lane-keeping ability declines, requiring the agent to learn appropriate actions to keep the vehicle centered. The primary focus will be on comparing the algorithms based on their learning capabilities to adapt to the changing conditions and maintain proper vehicle control during distracted periods. By assessing their lane-keeping performance, we can gain insights into the effectiveness of each algorithm and their potential for assisting distracted drivers in real-world scenarios.

4.6.1 Performance Metrics

Two metrics will be used to determine the performance of the algorithms: the average reward over all episodes and the average reward in a 100 episode window. The average reward over all episodes is calculated by dividing the cumulative reward by the number of episodes in order to compare the learning rate. If two algorithms have the same end performance but one has a much slower learning rate, this will show in the average reward over all episodes as this will be lower in the algorithm with the lower learning

rate. The average reward of a 100 episode window is calculated by taking the average reward of the current episode and the next 100 episodes. This metric is used to compare the final performance of the algorithms. As a another metric of driving performance and costs the average distance from the centerline over all episodes will be used.

4.6.2 FOCOPS-LSTM as Solution

As both a memory-component and safety are important for the solution to the distracted driver problem, we propose a combination of the two, called Long Short-Term Memory First Order Constrained Optimization in Policy Space (FOCOPS-LSTM) to train our lane-keeping assistance system. While CPO has been used before as a solution for autonomous driving (Kong et al., 2021a, 2021b), FOCOPS has not yet been used for the distracted driver problem.

By extending FOCOPS with a LSTM layer the unobservable state of the driver can be accounted for. Our system will only use the car's metrics so it does not require labelled data and minimizes the intrusiveness of the system. The system will be able to learn from previous experiences how the car's metrics change over time and therefore will be able to determine if the driver is attentive or not. The driving dynamics are simulated using a realistic driving simulator. As far as we know, this is the first work to use a Recurrent Safe Reinforcement Learning algorithm to solve the distracted driver problem and account for uncertainty about a driver's distraction in a shared-control, lane-keeping scenario while relying only on commonly available driving performance metrics as observations.

4.6.3 TD3 vs TD3-LSTM

To address the first sub-question regarding the advantages of using a memory-based Deep Reinforcement Learning method in the distracted driver problem, we will extend the TD3 algorithm by incorporating a Long Short-Term Memory (LSTM) layer, resulting in TD3-LSTM. The objective is to assess the impact of adding a memory component to the algorithm in the context of the distracted driver problem, which involves partial observability due to the driver's distracted state. We hypothesize that the inclusion of the LSTM layer will enhance the algorithm's performance and contribute to better handling of the partial observability challenge.

4.6.4 FOCOPS vs TD3

In order to address the second sub-question concerning the efficacy of safe reinforcement learning methods in preventing unsafe actions, we will compare the performance of FOCOPS with TD3. The hypothesis is that FOCOPS, incorporating a cost-critic, will result in fewer violations compared to TD3, which primarily focuses on maximizing the reward without considering the associated costs. By evaluating the violation rates and overall safety of both algorithms, we aim to gain insights into the effectiveness of FOCOPS in promoting safer actions during the distracted driver problem.

4.6.5 FOCOPS-LSTM vs TD3-LSTM

To evaluate the effectiveness of our proposed solution FOCOPS-LSTM for the distracted driver problem, we will conduct a comparative analysis against TD3. To ensure a fair comparison, we will also extend TD3 with a LSTM layer, leading to

TD3-LSTM. Building upon the insights from the previous research questions, our hypothesis suggests that the inclusion of a recurrent layer in FOCOPS will leverage the safety aspects of FOCOPS and its capacity to handle the partial observability inherent in the distracted driver problem. Consequently, we anticipate that FOCOPS-LSTM will outperform TD3-LSTM in terms of overall performance and safety. This investigation will provide valuable insights into the potential advantages of our proposed FOCOPS-LSTM solution for addressing the challenges of cooperative lane-keeping in the presence of a distracted driver.

Answering these three sub-questions will result in the answer to the main question whether recurrent safe reinforcement learning can be considered a viable solution to the distracted driver problem. The adapted code can be viewed at https://github.com/PieterES/Master-Thesis

5 Results

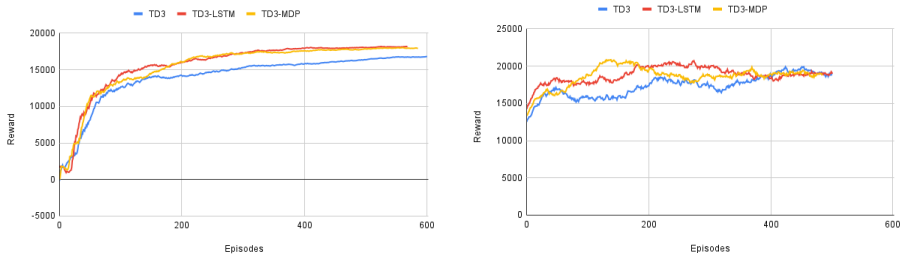
To tackle the research inquiries at hand, we initiate a comprehensive comparative analysis involving TD3 and FOCOPS, serving as our unconstrained and constrained Reinforcement Learning methodologies, respectively. Furthermore, we delve into their LSTM-augmented variants as integral components of our evaluative framework. Preceding the actual experiments designed to address the primary research questions, we executed a series of preliminary investigations aimed at optimizing the parameters governing TD3 and FOCOPS.

5.1 Preliminary Results

A comparison was conducted to assess whether the driver's hidden state truly influences the algorithm, potentially aligning with a genuine POMDP. In this evaluation, inputs were limited to include solely the car's position, angle, and previous driver actions. In a standard POMDP scenario, the expectation was for the LSTM component to provide advantages in handling partial observability, potentially resulting in higher average rewards. Figure 14 presents the results, showcasing both the average reward across all episodes and the average reward in a 100-episode window. This setup allowed for a fair comparison of learning rates and the final algorithm performance.

5.1.1 Distracted driver as MDP

Figure 14 presents a comparison of average rewards between TD3, utilizing solely track position, angle, and previous driver actions as inputs, and TD3 with the additional inclusion of the driver's distracted state as inputs. Surprisingly, the performance disparity between these two setups is minimal. This observation challenges our initial hypothesis, which anticipated a substantial performance improvement if the distracted driver problem were indeed a POMDP owing to the driver's undisclosed distracted state.



(a) (b)

Figure 14: (a) Average reward of TD3, TD3-LSTM and TD3 with the distracted state over all episodes (b) Average reward of TD3, TD3-LSTM and TD3 with the distracted state over 100 episode window

5.1.2 Flicker condition

The results reveal that TD3-LSTM does not exhibit a significant increase in average rewards across all episodes or within a 100-episode window, contrary to what one might expect in a POMDP environment. To delve deeper into this matter, we conducted a flicker condition experiment, akin to those conducted by Meng et al. (2021) and Hausknecht and Stone (2015). In this experiment, we compared the performance of TD3-LSTM and TD3 in a true POMDP environment created by introducing a flicker condition.

In this flicker condition, the algorithm is subjected to receiving only 0's with a certain probability, as opposed to the regular input. This condition simulates the presence of faulty sensors. In scenarios where the algorithm receives only 0's, it has no information about the vehicle's position or the driver's inputs. Consequently, choosing the correct actions becomes an arduous task unless there is a memory component such as an LSTM layer.

Hence, within the experiments involving the flicker conditions, there exists a probability that the algorithm will receive only 0's, effectively creating a genuine POMDP environment where the position and driver actions remain entirely unknown. The outcomes of these experiments are visually depicted in Figure 15.

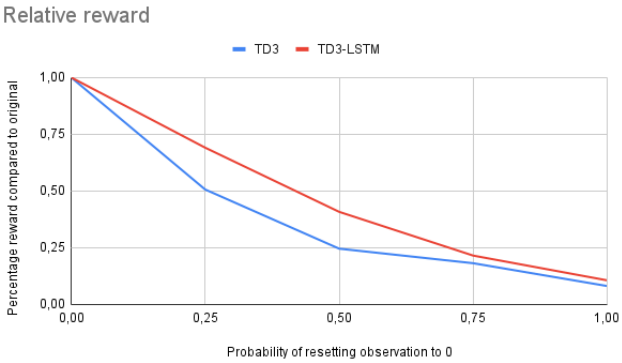


Figure 15: Relative Reward of TD3 and TD3-LSTM compared to original score

In this analysis, it becomes evident that as the probability of receiving accurate inputs diminishes, TD3-LSTM consistently outperforms TD3 relative to their respective baseline scores. This observation highlights that our implementation of TD3-LSTM excels in an environment that can be unambiguously categorized as a POMDP. In contrast, TD3-LSTM's performance remains similar to TD3 in the original distracted driver problem setting. This observation challenges the initial premise that the distracted driver problem inherently aligns with the characteristics of a POMDP, primarily rooted in the uncertainty surrounding the driver's distracted state.

Given that the act of revealing the driver's distracted state did not yield the substantial performance improvement anticipated, it is evident that our assumption characterizing the problem as a POMDP is not well-founded. Therefore, the reformalization of the distracted driver problem from a POMDP problem to a BMDP is imperative.

5.1.3 Additional BMDP proof

The fundamental distinction between a Block MDP (BMDP) and a Partially Observable MDP (POMDP) pivots on the notion that the inputs can be readily organized into discrete blocks. To substantiate this critical distinction, a comprehensive demonstration of the feasibility of transforming observed contexts into distinct blocks is imperative. In essence, this entails establishing the existence of a decoding function capable of mapping observed contexts to their respective underlying true states. To this end, a neural network was trained, employing the identical architectural framework as deployed in our experimental setup. Remarkably, this neural network attained a commendable accuracy rate of 83.8% in predicting the attentive state of the driver based on historical data.

Consequently, a compelling inference can be drawn: there indeed exists a decoding mechanism inherently capable of mapping contextual inputs to their originating states. This inference, in turn, implies that the agent, when navigating the original distracted driver problem, implicitly performs this decoding operation while seeking to optimize its policy. To facilitate a visual comprehension of the data segregation achieved, the Appendix includes Figure 28, illustrating a 2D data plot, along with Figures 29, 30, 31 and 32, presenting 3D data plots, as elaborated in Section 8.1.

5.2 Parameter Optimization

Before conducting the experiments, several parameters were optimized. Some parameters, like neural network size and the reward function, were based on similar papers (Meng et al., 2021). However, other parameters, such as cost, cost boundary, and experimental length, are more specific to this research and were fine-tuned before proceeding with the experiments.

5.2.1 Timesteps

In the context of resource constraints and time limitations, a critical parameter in the experimentation process is the total runtime allocated for each trial. To determine this parameter, FOCOPS was employed due to its relatively lower sample efficiency compared to TD3, making it an appropriate benchmark for defining the minimum requisite timesteps. The findings presented in Figure 16 illustrate the average reward computed over a 100-episode window, revealing that the algorithm achieves convergence after approximately 100,000 timesteps. Subsequently, although some degree of variance persists in the average reward, further improvement is not observed.

Consequently, the decision was made to establish a maximum timesteps threshold of 150,000. This choice provides the algorithm with an extended duration to converge, if such a prolonged period proves necessary.

5.2.2 Cost and Rewards

To ascertain the most suitable balance between reward and costs in the FOCOPS algorithm, the original reward function is subjected to division by factors of 1, 10, 100, and 1000. Similarly, the agent's costs are manipulated, assuming values of 1, 10, 100, and 1000 for transgressing the cost threshold. The outcomes of these experiments are presented in Table 3. It is noteworthy that the configurations where the reward is divided by 1000 while maintaining a cost value of 1 yield the highest average reward

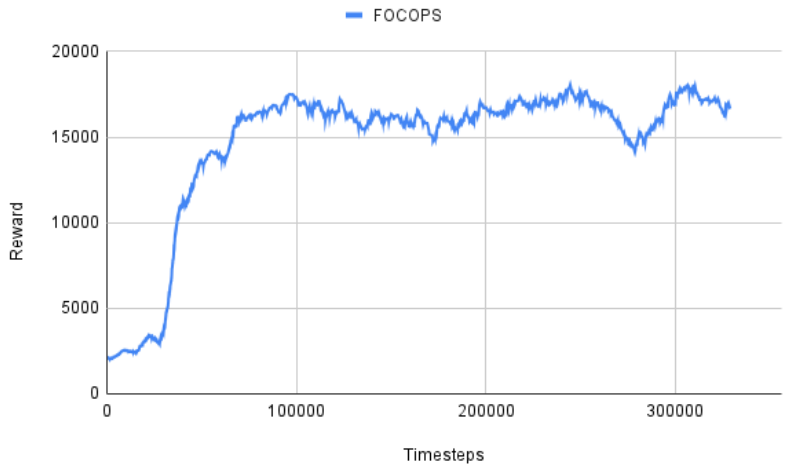


Figure 16: Average reward of FOCOPS over a 100 episode window

over the last 100 episodes. Consequently, these parameter settings have been selected as the optimal configuration.

Cost/Reward	/1	/10	/100	/1000	Average
1	16779	16267	15544	15284	15968,5
10	14838	16104	13271	16996	15302,25
100	16221	13689	15722	15873	15376,25
1000	14520	15682	14621	16125	15237
Average	15589,5	15435,5	14789,5	16069,5	

Table 3: Average reward of the last 100 episodes for different reward and cost ratios.

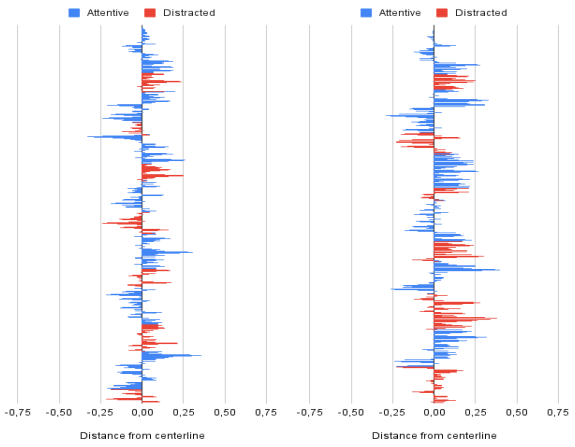
5.2.3 Cost Boundary

The costs assigned to the FOCOPS agent are contingent on the vehicle's deviation from the centerline, prompting an exploration of different cost boundaries set at 0.25m, 0.5m, 0.75m, and 1m. The outcomes of these variations are detailed in Table 4. Moreover, the distinct driving behaviours of the FOCOPS agent are depicted in Figure 17 and Figure 18, while Figure 19 showcases the driving actions of the TD3 agent, including the steering maneuvers executed by both the driver and agent for a specific episode. Notably, a cost boundary of 0.5m exhibits the most promising results, aligning with the rewarding behaviour observed with a 0.25m cost boundary. Details of the driving actions of the FOCOPS agent, including the steering behaviours of both the driver and the agent are viewed in Figure 20. Consequently, the 0.5m cost boundary has been designated as the optimal configuration. Further insights into these parameters,

alongside other hyperparameters, are enumerated in Table 5.

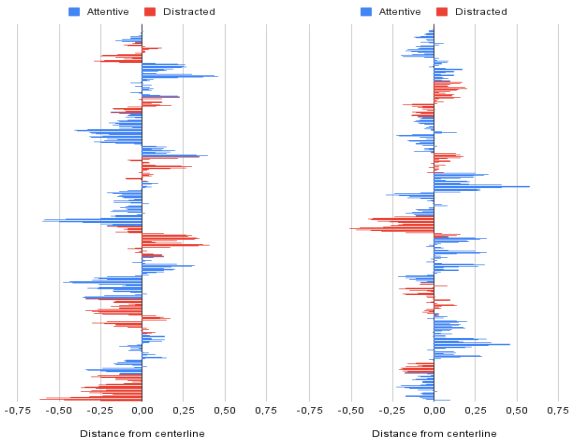
	Distance			
	0.25	0.5	0.75	1.00
Reward	16570,12317	17098,19059	15032,30416	13691,12792
Cost	16782,17822	5831,683168	4940,594059	3356,435644

Table 4: Average reward and costs of the last 100 episodes for different cost distances.



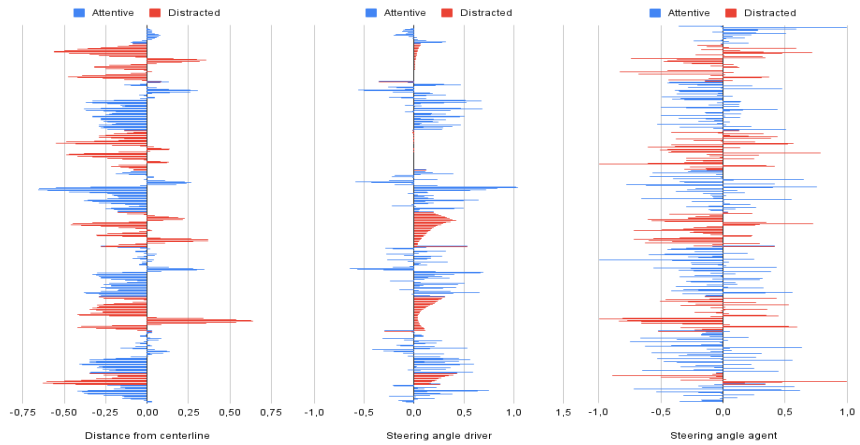
(a) (b)

Figure 17: (a) Driving behaviour of FOCOPS agent with cost boundary of 0.25m (b) Driving behaviour of FOCOPS agent with cost boundary of 0.50m



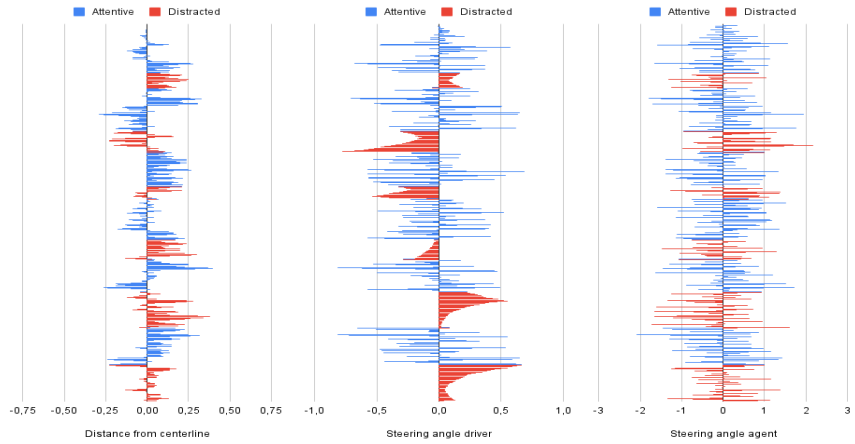
(a) (b)

Figure 18: (a) Driving behaviour of FOCOPS agent with cost boundary of 0.75m (b) Driving behaviour of FOCOPS agent with cost boundary of 1.00m



(a) (b) (c)

Figure 19: (a) Combined driving behaviour of TD3 agent (b) Steering actions of the driver (c) Steering actions of the agent



(a) (b) (c)

Figure 20: (a) Combined driving behaviour of FOCOPS agent and driver (b) Steering actions of the driver (c) Steering actions of the agent

Hyperparameter	TD3	FOCOPS
No. of hidden layers	4	4
No. of hidden nodes	128	128
LSTM Layer	128 (2nd Layer)	128 (2nd Layer)
Activation	tanh	tanh
Maximum timesteps	150.000	150.000
Std of Gaussian exploration noise	0.25	-0.5
Batch size	100	2048
Mini-batch size	NA	64
Memory size	1e6	NA
Number of Epochs	NA	10
Learning Rate (λ)	3e-4	3e-4
Nu (Cost coefficient) (ν)	NA	1
Nu-learning rate (λ_ν)	NA	0.01
Max Nu (ν_{max})	NA	2.0
Rho (ρ)	0.005	NA
Gamma (γ)	NA	0.99
Inverse lambda	NA	1.5
KL bound (D_{KL})	NA	0.02
Noise added to target policy during critic update (ϵ)	0.25	NA
Range to clip target policy noise (c)	0.5	NA

Table 5: Hyperparameters for both algortihms

5.3 Analysis of Parameter Optimization

The outcomes of the initial optimization experiment, as documented in Table 3, underscore that the maximum reward is achieved when the reward remains unaltered, coupled with a cost factor of 1. Nevertheless, it is crucial to acknowledge that the learning process is inherently influenced by stochastic elements. Consequently, the

parameter settings for reward and costs have been determined based on the highest average rewards within their respective domains. Specifically, the configuration yielding the highest average reward involves setting the cost to 1 and dividing the reward by 1000. These designated settings are uniformly employed for FOCOPS in all subsequent comparative analyses.

Table 4 provides a comprehensive overview of the distances in meters from the centerline at which point costs are imposed. Naturally, as the constraint distances become more stringent, the associated costs escalate, in line with the expectation that the vehicle will transgress these boundaries more frequently. Nevertheless, it is noteworthy that the configuration yielding the highest average reward over the last 100 episodes corresponds to a constraint distance of 0.5 meters. Consequently, this particular parameter setting is consistently employed in all subsequent analyses involving FOCOPS and FOCOPS-LSTM.

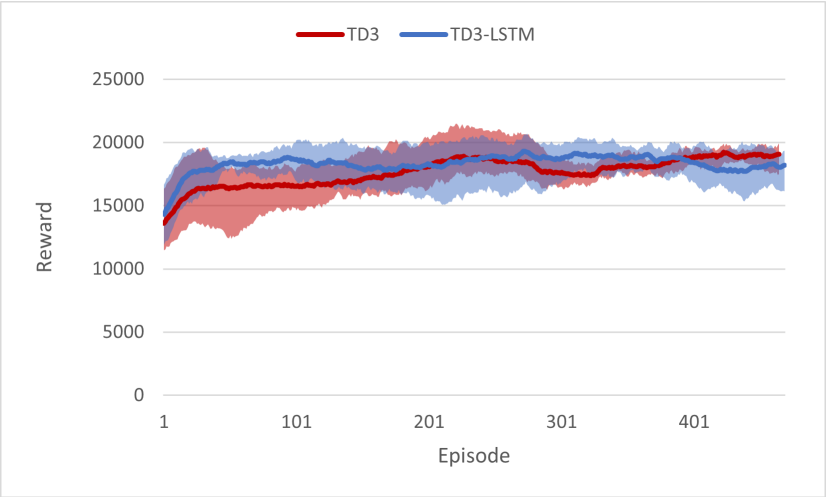
5.3.1 Driving behaviour

Furthermore, a detailed analysis of cooperative driving performance can be gleaned from Figure 17 and 18. Notably, these figures offer a comparative view of the agent's interaction with the distracted driver. In all comparisons, a randomly selected completed episode from the last 100 episodes is used for assessment. While marginal disparities in driving performance are evident in Figure 17, a more pronounced distinction becomes apparent when contrasting the 0.25m and 0.5m constraint settings against the 0.75m and 1.00m settings, particularly in terms of proximity to the centerline. Consequently, it is evident that the chosen constraint setting significantly impacts driving performance.

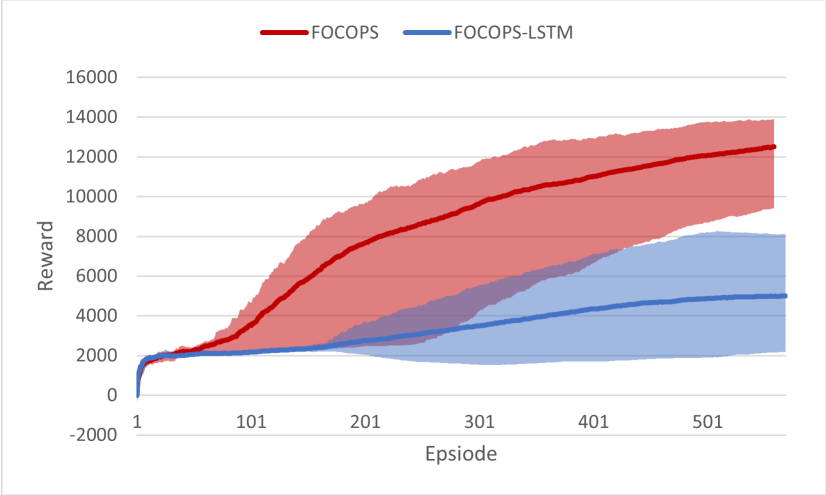
Intriguingly, Figure 19 underscores that the driving behaviour of TD3 closely resembles the suboptimal performance observed in the worst-performing FOCOPS configuration. However, as shown in the results of Research Question 3, TD3 consistently outperforms FOCOPS in terms of average reward. This dichotomy implies that TD3 places a higher emphasis on optimizing reward attainment, often at the expense of safety, whereas FOCOPS, with its constraints, prioritizes safer driving practices.

5.4 Research Question 1

The initial research question, which pertains to the advantages associated with the utilization of a memory-based Deep Reinforcement Learning approach within the distracted driver problem, has been systematically examined and substantiated by our preliminary findings. These preliminary results indicate that the incorporation of an LSTM layer did not yield a substantial improvement in TD3's performance concerning the distracted driver problem. This outcome is primarily attributable to the erroneous presumption that our problem was characterized as a POMDP when, in reality, it aligns more closely with a Block MDP. Within this Block MDP framework, the observations provided to the agent are sufficiently informative to approximate an optimal policy. Consequently, the original findings derived from the preliminary results remain both robust and substantiated. Furthermore, these findings are reinforced by the supplementary evidence presented in Figure 21. This additional data underscores the notion that the LSTM variant either performs equivalently or less effectively than its non-LSTM counterpart in the context of a Block MDP environment



(a)



(b)

Figure 21: (a) Average reward of TD3 and TD3-LSTM over all episodes (b) Reward of FOCOPS and FOCOPS-LSTM over all episodes

5.4.1 Analysis of Research Question 1

The outcomes of Research Question 1 have unveiled a discernible decrease in performance associated with TD3-LSTM and FOCOPS-LSTM when contrasted with TD3 and FOCOPS lacking a LSTM layer. This decline in performance can be ascribed to several underlying factors.

First and foremost, it is crucial to acknowledge that the LSTM layer significantly

amplifies the computational complexity in comparison to a conventional dense layer. Consequently, the training process with a LSTM layer necessitates a substantially prolonged duration to grasp and enhance performance when measured against the swifter progress achievable with a simpler dense layer. While a LSTM layer holds merit in a POMDP environment where the maintenance of a belief regarding the true state remains pivotal, this elongated training period invariably leads to a reduction in the recorded average reward per episode.

The secondary reason for the lesser performance witnessed in the LSTM-enabled counterparts pertains to the inherent nature of the MDP or Block MDP environment. In such environments, the quality of observations provide the algorithm with an ample information, sufficient for approximating an optimal solution. Consequently, the inclusion of a LSTM layer yields no advantageous leverage in steering towards a solution and, paradoxically, might encumber the training process. Furthermore, saturating the algorithm with excess or irrelevant data inputs has the potential to obstruct its learning trajectory. Hence, the selection of inputs bearing the most pertinent information becomes paramount. In our conducted experiments, the designated inputs, encompassing car position, car angle, and the driver's antecedent steering and acceleration inputs, ensure the requisite information essential for fostering effective learning.

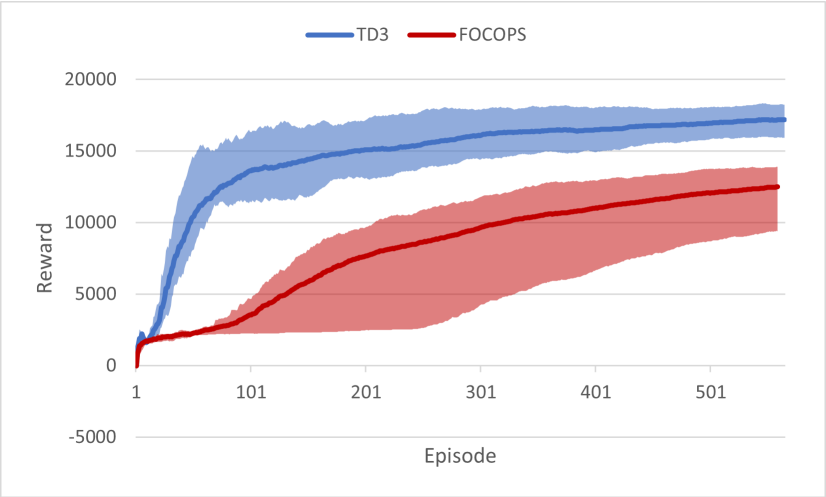
5.5 Research Question 2

The second research question delves into the capacity of safe reinforcement learning methods to prevent unsafe actions, with a particular focus on comparing the performance of FOCOPS and TD3. The outcomes of this investigation are depicted in Figure 22 and Figure 23. For validation, experiments pertaining to research questions 2 and 3 were independently conducted five times. The shaded regions in the figures portray the upper and lower bounds of results from these five experiments, while the solid line traces the average performance across these repetitions for each algorithm.

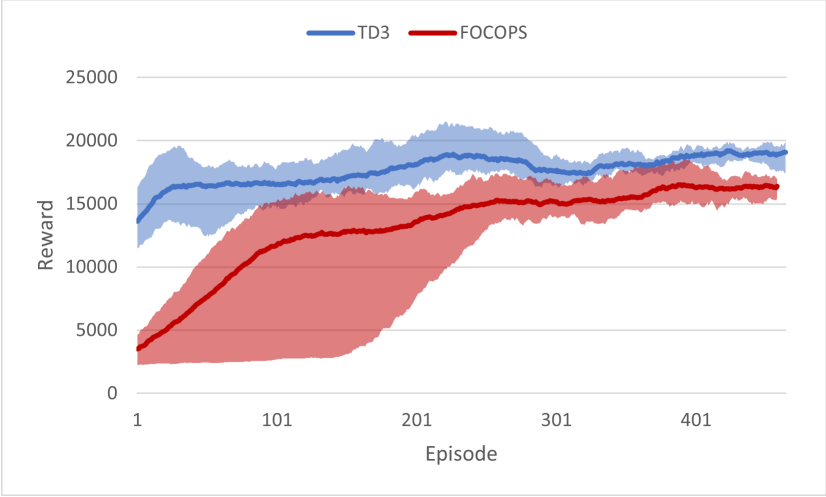
Figure 22 illustrates the average rewards garnered by TD3 and FOCOPS over all episodes, as well as the rolling average reward across 100-episode windows. Evidently, TD3 achieves higher rewards per episode. Nevertheless, when assessing the average costs per episode, TD3's performance mirrors that of FOCOPS, as depicted in Figure 23. This pattern persists when considering their respective LSTM-based variants, showcased in Figure 24. Despite TD3 showcasing analogous average costs in the long run, it does not exhibit an inherent inclination to prioritize cost minimization while simultaneously maximizing rewards. Consequently, TD3 might opt for actions that yield substantial rewards, even if they incur costs. In contrast, while FOCOPS records a lower average reward, it maintains comparable costs. Therefore, FOCOPS endeavors to limit costs, resulting in a comparable maximum cost, despite its lower maximum reward. The disparity in reward outcomes between the two algorithms can be predominantly attributed to differences in sample efficiency.

5.5.1 Analysis of Research Question 2

The findings from Research Question 2 reveal that FOCOPS achieves a lower average reward when compared to TD3. However, in terms of costs, TD3 and FOCOPS demonstrate a similar outcome, largely owing to the incorporation of a cost critic in FOCOPS. Given that TD3 primarily prioritizes maximizing rewards, it may indeed attain higher reward levels, but this often comes at the expense of incurring greater costs. In contrast, FOCOPS must not only optimize rewards but also adhere to the



(a)



(b)

Figure 22: (a) Average reward of TD3 and FOCOPS over all episodes (b) Reward of TD3 and FOCOPS over 100 episode window

constraints defined by the cost critic, which necessitate the vehicle's position to be within 0.5 meters from the centerline. By staying within these specified bounds, the vehicle operates within the safer regions of the road, thereby satisfying the cost critic's requirements. Consequently, although FOCOPS registers a lower average reward compared to TD3, the costs incurred by both algorithms remain at similar levels. This implies that while the overall algorithm performance in terms of reward is diminished, the incorporation of a cost critic contributes to a reduction in undesirable driving

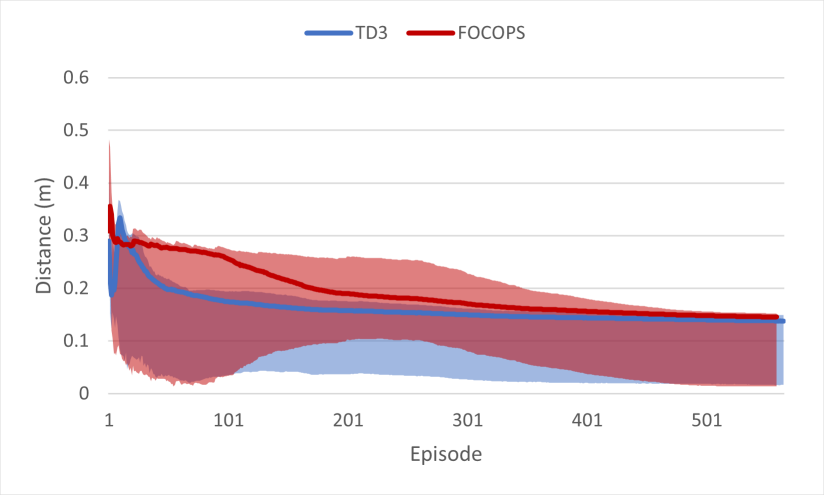


Figure 23: Average distance from the centerline for TD3 and FOCOPS

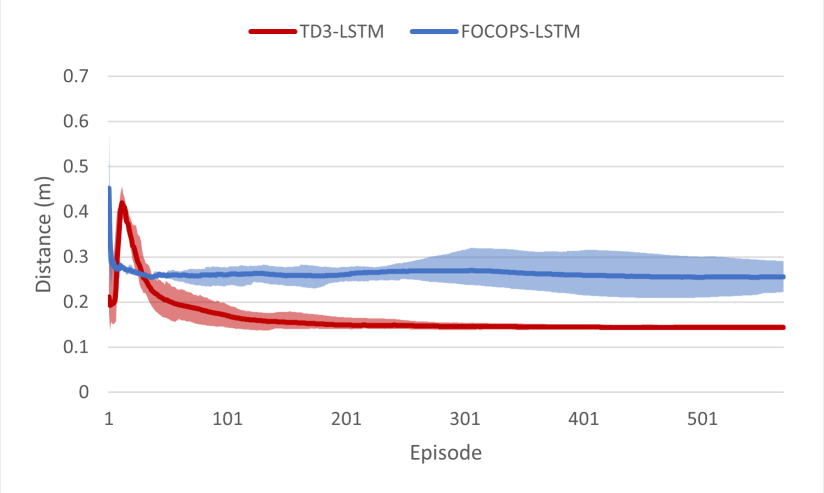


Figure 24: Average distance from the centerline for TD3-LSTM and FOCOPS-LSTM

behaviours, albeit at the expense of total reward.

5.5.2 Learning Methodologies

The observed difference in reward between the two algorithms can be ascribed to their distinctive learning methodologies, with a focus on sample efficiency being particularly pivotal in the context of BMDP environments. This distinction underscores the relevance of the offline and online learning paradigms. TD3 represents an offline

reinforcement learning approach that leverages a replay buffer to store state-action-reward tuples, possibly inclusive of hidden states. During training, random samples are drawn from this buffer to facilitate the algorithm's learning process. Although the replay buffer is of finite size and older tuples are eventually replaced, steps from prior episodes can still contribute to learning. Conversely, FOCOPS adopts an online reinforcement learning framework, where training solely relies on the tuples generated within the current episode. Consequently, FOCOPS exhibits lower sample efficiency. While online learning proves advantageous in scenarios characterized by evolving data patterns, as older and therefore irrelevant tuples are discarded. However, our specific environment does not exhibit new data pattern emergence during training. Consequently, the advantages of online learning diminish, thereby resulting in a sample-inefficient algorithm.

To further underscore this point, a different online algorithm, Proximal Policy Optimization (PPO) (Schulman et al., 2017), is compared to another offline algorithm, Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2015), in Figure 25. A similar graphical trend as observed in Figure 22 is evident, reinforcing the conclusion that the difference in performance between TD3 and FOCOPS is primarily attributable to sample efficiency, a crucial consideration in the context of BMDP environments.

5.6 Research Question 3

The outcomes pertaining to the third research question, which investigates the potential advantages of the proposed FOCOPS-LSTM solution relative to TD3-LSTM, are presented in Figure 26.

Unfortunately, the empirical results do not corroborate our initial hypothesis, as FOCOPS-LSTM does not demonstrate superior performance in comparison to TD3-LSTM. While FOCOPS-LSTM does exhibit lower costs in terms of maintaining proximity to the centerline when contrasted with TD3-LSTM, it concurrently attains a significantly reduced average reward.

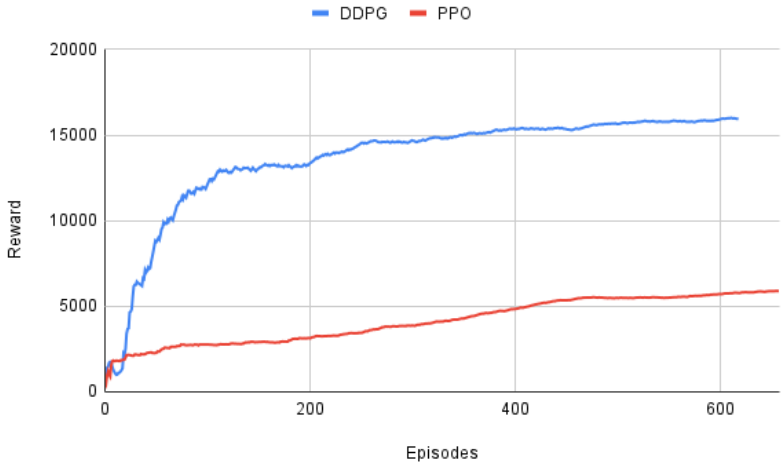
5.6.1 Analysis of Research Question 3

The findings from Research Question 3 have yielded disappointing results, as the envisioned solution of FOCOPS-LSTM failed to align with our initial hypotheses. Given that this proposed solution stemmed from the amalgamation of conclusions drawn from the first two research questions, it is imperative to acknowledge that the limitations encountered also stem from a confluence of these conclusions.

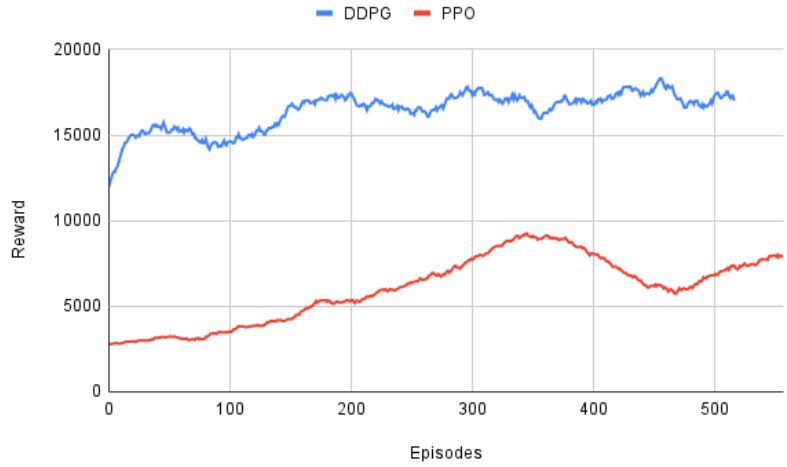
Primarily, the limitations are twofold. Firstly, owing to the inherently online nature of FOCOPS as a reinforcement learning method, its LSTM-equipped counterpart, FOCOPS-LSTM, grapples with analogous difficulties in navigating the Block MDP environment. Secondly, given that the environment conforms to a Block MDP paradigm rather than a POMDP, the inclusion of an LSTM layer does not yield the expected benefits, consequently resulting in a further decline in performance. These combined limitations underscore the challenges encountered in devising an effective solution for the distracted driver problem within this context.

5.6.2 Valuable Insights

Nevertheless, these results provide us with valuable insights. Most notably, they highlight the fundamental distinction between solving POMDPs (Partially Observ-



(a)

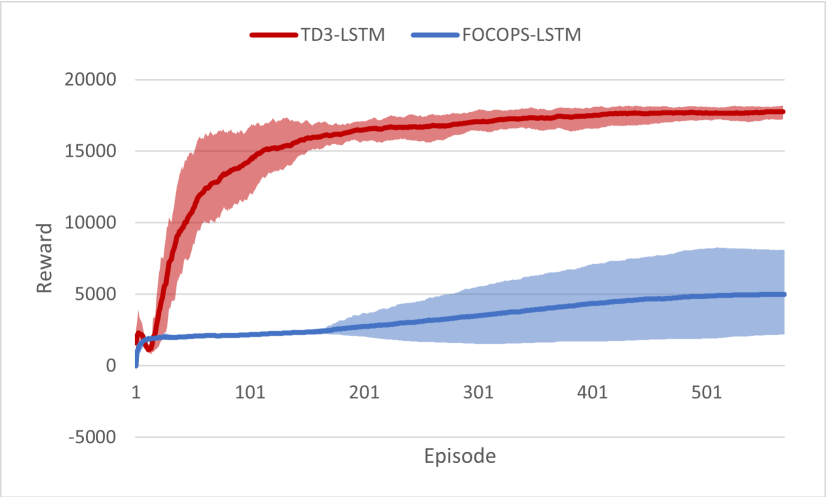


(b)

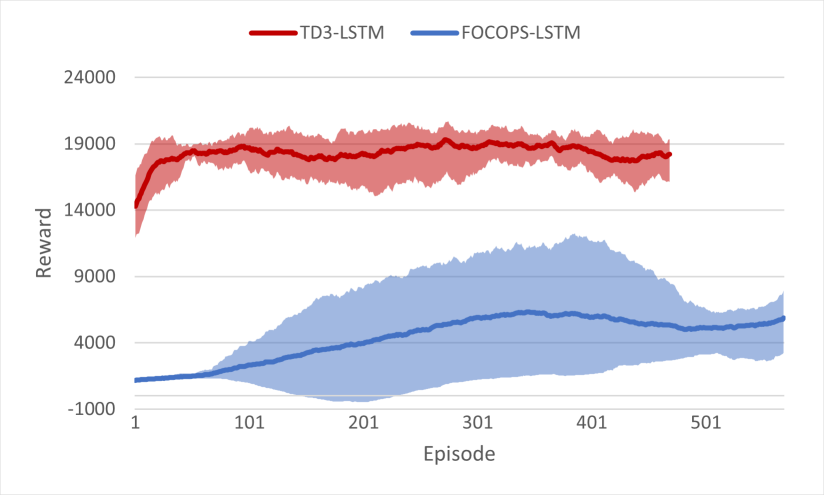
Figure 25: (a) Average reward of DDPG and PPO over all episodes (b) Reward of DDPG and PPO over 100 episode window

able Markov Decision Processes) and BMDPs (Block MDPs), despite the similarity in their formalizations. Both can be represented as tuples involving states, actions, rewards, hidden states, and state-transition functions, along with functions describing how states and actions can result from possible observations. However, the methods employed to tackle these problems differ significantly.

As demonstrated in the preceding experiments, in a POMDP environment, the



(a)



(b)

Figure 26: (a) Average reward of TD3-LSTM and FOCOPS-LSTM over all episodes (b) Reward of TD3-LSTM and FOCOPS-LSTM over 100 episode window

inclusion of an LSTM layer can be advantageous in solving the problem by aiding in handling the partial observability. Conversely, this approach does not yield the same benefits when dealing with a BMDP problem. This delineation underscores the importance of selecting the appropriate problem formalization and corresponding solution methodology based on the underlying characteristics of the environment.

5.7 Additional Experiments

Additionally, a series of supplementary experiments were conducted throughout this thesis, exploring various avenues that, while not directly related to the primary research questions, offer intriguing insights. These experiments encompassed diverse approaches, including data augmentation based on the vehicle crossing cost boundaries, adjustments to input length to simulate a pseudo-memory component, and the incorporation of an unsupervised clustering algorithm into the inputs. The comprehensive results of these supplementary experiments, while not integral to the core thesis, are available in the Section 8.2 of the Appendix for the sake of completeness.

6 Discussion and Future Work

This chapter is dedicated to the discussion of the results presented in the preceding section. While the experiments conducted in this study have yielded valuable insights, it's essential to acknowledge certain limitations inherent in these investigations. Subsequent subsections will delve into these limitations in detail. Moreover, this chapter will outline potential directions for future research, suggesting adaptations and approaches that could be pursued to mitigate the identified limitations and further enhance the understanding of the distracted driver problem in the context of reinforcement learning

6.1 Driver Model

It is important to acknowledge that the driver model used in this study, while serving its purpose, is a simplified representation and does not aim to perfectly mimic a real human driver. While the primary focus of this thesis was not centered on creating a highly realistic driver model, it is a crucial element in comprehensively addressing the distracted driver problem. One notable limitation is that in reality, human drivers are not frequently distracted while driving. If the driver model was more realistic and would only become distracted sparingly, then the agent may not have had ample opportunities to train on the distracted state.

To address this limitation, a more realistic driver model could be developed. This could involve human subjects participating in a highway simulation while engaging in secondary tasks over an extended period. Analyzing their driving behaviour during these simulations could lead to the creation of a more precise and realistic driver model. However, it is worth noting that such an approach would require substantial resources, as it would involve multiple subjects to obtain an average driver model. This presents an intriguing avenue for future research inquiries.

6.1.1 Personalized Assistant

Additionally, it's worth noting that the current agent model is not personalized to each driver's unique behaviour. An interesting adaptation for future research could involve the development of personalized lane-keeping assistants for individual human drivers, based on their driving behaviour data acquired from the simulation. Personalization could be particularly valuable in the context of continuous cooperative lane-keeping assistance.

For example, older drivers might prefer a more defensive driving style, while others may lean towards a more aggressive approach (Sagberg, Selpi, Bianchi Piccinini, & Engström, 2015). By customizing each assistant to align with the specific preferences and driving patterns of the individual human driver, a higher level of trust and satisfaction with the assistant could potentially be achieved.

However, it is essential to acknowledge that these personalized approaches extend beyond the scope of this thesis, primarily due to constraints in resources and complexity. Nevertheless, they represent intriguing avenues for future research and practical implementations in the field.

6.2 Other Algorithmic Approaches

While the proposed solution of FOCOPS-LSTM did not fully align with the initial hypothesis, it did reveal valuable insights. Specifically, it became evident that the

sample efficiency of an offline reinforcement learning algorithm played a significant role in performance, while the constraints imposed by FOCOPS effectively reduced the maximum costs across all episodes. Consequently, a logical and promising direction for future research involves delving deeper into the domain of offline constrained reinforcement learning algorithms. This approach holds the potential to enhance both safety and performance in cooperative driving scenarios, marking a significant step toward the goal of safer autonomous driving.

6.2.1 TD3-ASA

In this context, a recent study by Wang, Zhang, Hou, and Cheng (2023) introduced an algorithm known as Twin Delayed Deep Deterministic Policy Gradient based on Approximate Safe Action (TD3-ASA). TD3-ASA was first developed in an autonomous driving simulation during this thesis. Notably, TD3-ASA alters the policy's action output during the exploration process to approximate a safe action. This safe action is then employed to train a safe policy for deployment. The results of this experiment indicated that TD3-ASA exhibited lower episodic costs compared to TD3 and other safe reinforcement learning methods. It's important to highlight that TD3 still achieved the highest average reward among the tested algorithms.

Considering the promising outcomes observed with TD3-ASA, it suggests that safe, offline reinforcement learning algorithms have the potential to effectively leverage sample efficiency and constraints in autonomous driving scenarios. This finding raises the possibility that such offline constrained reinforcement learning algorithms could be valuable when applied to the distracted driver problem. In future research endeavors, it would be intriguing to conduct a comparative analysis pitting these offline constrained reinforcement learning algorithms against other safe or unsafe reinforcement learning approaches within the distracted driver problem domain. Such a comparison could provide valuable insights into the relative advantages and limitations of different algorithms, contributing to the development of safe and efficient lane-keeping assistants for distracted drivers on highway

6.2.2 BMDP Solvers

The study outcomes indicate that the distracted driver problem in this environment exhibits characteristics more in line with a Block MDP rather than a POMDP. This discovery presents several promising directions for future research, particularly in the exploration of BMDP solvers that could enhance the effectiveness and efficiency of lane-keeping assistants for distracted drivers on highways.

In this context, several noteworthy BMDP solvers warrant further investigation. These algorithms include:

- Block-structured Representation learning with Interleaved Explore Exploit (BRIEE) (X. Zhang et al., 2022): BRIEE is designed to efficiently handle complex environments by leveraging block-structured representations and interleaving exploration and exploitation.
- Upper Confidence Bound driven REPresentation (REP-UCB) (Uehara, Zhang, & Sun, 2021): REP-UCB focuses on scaling function approximation to more intricate environments and strives to achieve a balance between efficient representation learning, exploration, and exploitation.

- Model-Free Feature Learning and Exploration (MOFFLE) (Modi, Chen, Krishnamurthy, Jiang, & Agarwal, 2021): MOFFLE specializes in sample-efficient learning by combining model-free feature learning with exploration strategies.

Comparing the performance of these BMDP solvers with that of constrained reinforcement learning methods, such as TD3-ASA, presents an intriguing and valuable research avenue. Such a comparative analysis would offer insights into the strengths and limitations of various algorithms within the context of the distracted driver problem. Furthermore, it could assist in identifying the most effective approach to developing adaptive and safe lane-keeping assistants. This research could also provide critical insights into optimizing the trade-off between exploration and exploitation while efficiently managing the complexities of real-world driving scenarios.

6.3 Hardware and Software limitations

During the experiments, unexpected errors occurred. To illustrate this, an attentive driver operated the vehicle independently, without any algorithmic assistance or distractions. The driver remained focused throughout the experiment. The expected outcome would have been flawless driving, which was generally observed in most episodes. However, certain episodes exhibited minor discrepancies between TORCS and the script controlling the driver model. As depicted in Figure 27, three episodes are presented. Two of these episodes are identical, while the third episode depicts the vehicle crashing. The precise cause of these anomalies, whether stemming from software or hardware issues, remains unclear. Nonetheless, such errors underscore the importance of conducting multiple runs of experiments to ensure robust and reliable conclusions.

Figure 27: Driving behaviour of TD3

Another constraint in the experimental setup was the availability of resources. Given the inherent randomness in the setup, including factors like neural network initialization and the timing and duration of driver distractions, conducting a substantial

number of experiments was essential for obtaining robust results. However, due to the limitation of having only a single laptop for experimentation, and considering that each complete experiment spanned approximately 9 hours, the capacity to run a large number of experiments was severely restricted. Consequently, not all experiments could be repeated as extensively as desired, thereby constraining the depth and breadth of conclusions that could be drawn from the experiments.

6.4 Different Driving Environments

This thesis predominantly concentrated on the highway environment; however, there exists an intriguing avenue for further exploration, involving the training and testing of the agent in diverse environments. The gym-TORCS wrapper offers the flexibility to select from a range of tracks for the agent's training. Extending the agent's training to encompass various tracks would impart versatility and adaptability to the agent. Although the primary objective centered on lane-keeping on highways, diversifying the training environments would provide a more comprehensive assessment of the solution's potential. Ideally, the agent should demonstrate competence in lane-keeping across diverse scenarios, not solely limited to highways. Otherwise, deploying multiple agents to address each distinct environment would prove impractical. Thorough testing across all types of roadways, including urban, residential, and rural roads, is imperative for the effective deployment of the agent.

6.5 Different Software Environments

Moreover, exploring diverse training environments in the realm of software can be advantageous. While this thesis leveraged TORCS, a widely-adopted driving simulator for AI research, alternative environments present distinct challenges and prospects. Some recommended environments for reinforcement learning encompass Speed Dreams (`http://www.speed-dreams.org/`), MetaDriver (Q. Li et al., 2022), or the creation of custom environments using Bullet-Safety-Gym (Gronauer, 2022). Contrasting results across various road types and software engines would furnish valuable insights into algorithm robustness and generalization. Additionally, varying training environments in terms of software would serve as a safeguard against software bugs, as previously discussed.

In summary, expanding the agent's training to encompass diverse environments and software engines would bolster its adaptability and relevance in real-world driving contexts, rendering it a more comprehensive and dependable lane-keeping assistant.

7 Conclusion

This thesis set out to train an agent for cooperative car control in a highway simulation with a distracted driver, utilizing the TORCS driving simulator. The agent's task was to deduce the driver's distracted state from driving performance data, including position information and driver inputs. Initially, the problem was framed as a POMDP to address the uncertainty surrounding the driver's distraction state. However, it was subsequently revealed that this assumption was incorrect. Both the TD3 and FOCOPS algorithms were employed, along with an LSTM layer, to handle the latent driver state. Surprisingly, the addition of the LSTM layer did not yield the anticipated performance improvement in what was thought to be a POMDP environment. Consequently, the problem was redefined as a Block MDP, indicating that the detailed observations provided to the algorithm revealed enough information about the driver's distracted state to approximate an effective policy.

To the best of our knowledge, this work represents the first attempt to apply offline TD3-LSTM and online FOCOPS-LSTM to the distracted driver problem within a cooperative control framework.

While fully autonomous driving is a long-term goal aimed at enhancing driving safety, the progression toward its achievement is pivotal. Cooperative driving represents a logical intermediate step in this trajectory. Although our proposed solution did not produce the anticipated results, significant insights can still be derived from the findings. The key conclusions are as follows:

- The distracted driver problem was best formalized as a Block MDP, rather than a POMDP.
- The agent and the distracted driver cooperatively controlled a vehicle on a highway in the TORCS driving simulator. The agent acted as a lane-keeping assistant, inferring the driver's distracted state from driving performance.
- TD3 outperformed FOCOPS due to its sample efficiency, while FOCOPS achieved lower costs thanks to its cost critic.
- The inclusion of LSTM extensions to these algorithms did not result in performance improvements in the Block MDP environment. However, they did enhance performance under flicker conditions.
- Future research directions involve exploring algorithms like TD3-ASA and various BMDP solvers to address the distracted driver problem.
- Despite not achieving the expected lane-keeping performance, our approach provided valuable insights into problem formalization.

In conclusion, this thesis tackled the complex task of training a lane-keeping assistant for a distracted driver on a highway using reinforcement learning techniques. Through comprehensive experimentation and analysis, valuable insights were gained into problem formalization and the influence of various algorithms in this domain

In the journey towards safer autonomous driving, each step taken in the development of cooperative driving systems is of paramount importance. While our proposed solution did not achieve the intended level of lane-keeping performance, this study has furnished critical insights into problem formalization and potential pathways for future research.

In essence, this work contributes to the expanding knowledge base within the realm of safe reinforcement learning for cooperative driving scenarios. The outcomes advocate for further exploration of alternative algorithms, such as TD3-ASA and various

BMDP solvers. Additionally, they underscore the significance of precisely defining problem environments to craft more effective and secure lane-keeping systems. As the quest for autonomous driving progresses, these findings assume greater relevance in shaping forthcoming advancements in road safety and driver assistance systems.

References

Abbeel, P., Coates, A., & Ng, A. Y. (2010). Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research*, *29*(13), 1608–1639.

Achiam, J., Held, D., Tamar, A., & Abbeel, P. (2017). Constrained policy optimization. In *International conference on machine learning* (pp. 22–31).

Andriotis, C., & Papakonstantinou, K. (2021). Deep reinforcement learning driven inspection and maintenance planning under incomplete information and constraints. *Reliability Engineering & System Safety*, *212*, 107551.

Ayyasamy, S. (2022). A comprehensive review on advanced driver assistance system. *Journal of Soft Computing Paradigm*, *4*(2), 69–81.

Azizzadenesheli, K., Lazaric, A., & Anandkumar, A. (2016). Reinforcement learning in rich-observation mdps using spectral methods. *arXiv preprint arXiv:1611.03907*.

Beckers, N., Siebert, L. C., Bruijnes, M., Jonker, C., & Abbink, D. (2022). Drivers of partially automated vehicles are blamed for crashes that they cannot reasonably avoid. *Scientific Reports*, *12*(1), 16193.

Benloucif, M. A., Sentouh, C., Floris, J., Simon, P., & Popieul, J.-C. (2019). Online adaptation of the level of haptic authority in a lane keeping system considering the driver's state. *Transportation research part F: traffic psychology and behaviour*, *61*, 107–119.

Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., ... others (2019). Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*.

Bernhard, J., Gieselmann, R., Esterle, K., & Knol, A. (2018). Experience-based heuristic search: Robust motion planning with deep q-learning. In *2018 21st international conference on intelligent transportation systems (itsc)* (pp. 3175–3182).

Bertsekas, D., & Tsitsiklis, J. N. (1996). *Neuro-dynamic programming*. Athena Scientific.

Blaschke, C., Breyer, F., Färber, B., Freyer, J., & Limbacher, R. (2009). Driver distraction based lane-keeping assistance. *Transportation research part F: traffic psychology and behaviour*, *12*(4), 288–299.

Capallera, M., Angelini, L., Meteier, Q., Abou Khaled, O., & Mugellini, E. (2022). Human-vehicle interaction to support driver's situation awareness in automated vehicles: A systematic review. *IEEE Transactions on Intelligent Vehicles*.

Carr, S., Jansen, N., Junges, S., & Topcu, U. (2023). Safe reinforcement learning via shielding under partial observability..

Chai, C., Lu, J., Jiang, X., Shi, X., & Zeng, Z. (2021). An automated machine learning (automl) method for driving distraction detection based on lane-keeping performance. *arXiv preprint arXiv:2103.08311*.

Chen, J., Yuan, B., & Tomizuka, M. (2019). Model-free deep reinforcement learning for urban autonomous driving. In *2019 ieee intelligent trans-*

portation systems conference (itsc) (pp. 2765–2771).

Danesh, M. H., Cai, P., & Hsu, D. (2023). Leader: Learning attention over driving behaviors for planning under uncertainty. In *Conference on robot learning* (pp. 199–211).

Dann, C., Jiang, N., Krishnamurthy, A., Agarwal, A., Langford, J., & Schapire, R. E. (2018). On oracle-efficient pac rl with rich observations. *Advances in neural information processing systems*, *31*.

Detjen, H., Faltaous, S., Pfleging, B., Geisler, S., & Schneegass, S. (2021). How to increase automated vehicles' acceptance through in-vehicle interaction design: A review. *International Journal of Human–Computer Interaction*, *37*(4), 308–330.

Driessens, K., & Džeroski, S. (2004). Integrating guidance into relational reinforcement learning. *Machine Learning*, *57*, 271–304.

Du, S., Krishnamurthy, A., Jiang, N., Agarwal, A., Dudik, M., & Langford, J. (2019). Provably efficient rl with rich observations via latent state decoding. In *International conference on machine learning* (pp. 1665–1674).

Fulton, N., & Platzer, A. (2018). Safe reinforcement learning via formal methods: Toward safe control through proof and learning. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 32).

Garcia, J., & Fernández, F. (2012). Safe exploration of state and action spaces in reinforcement learning. *Journal of Artificial Intelligence Research*, *45*, 515–564.

Garcıa, J., & Fernández, F. (2015). A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, *16*(1), 1437–1480.

Gaskett, C. (2003). Reinforcement learning under circumstances beyond its control.

Geibel, P., & Wysotzki, F. (2005). Risk-sensitive reinforcement learning applied to control under constraints. *Journal of Artificial Intelligence Research*, *24*, 81–108.

Georgeon, O. L., Casado, R. C., & Matignon, L. A. (2015). Modeling biological agents beyond the reinforcement-learning paradigm. *Procedia Computer Science*, *71*, 17–22.

Greenwood, P. M., Lenneman, J. K., & Baldwin, C. L. (2022). Advanced driver assistance systems (adas): demographics, preferred sources of information, and accuracy of adas knowledge. *Transportation research part F: traffic psychology and behaviour*, *86*, 131–150.

Gronauer, S. (2022). Bullet-safety-gym: A framework for constrained reinforcement learning.

Gu, S., Yang, L., Du, Y., Chen, G., Walter, F., Wang, J., ... Knoll, A. (2022). A review of safe reinforcement learning: Methods, theory and applications. *arXiv preprint arXiv:2205.10330*.

Hausknecht, M., & Stone, P. (2015). Deep recurrent q-learning for partially observable mdps. In *2015 aaai fall symposium series.*

Howard, R. A., & Matheson, J. E. (1972). Risk-sensitive markov decision processes. *Management science*, *18*(7), 356–369.

Jansen, J. (2021). *Shared control lane-keeping assistance with a potentially distracted human in the loop-a pomdp approach* (Unpublished master's thesis).

Jansen, N., Könighofer, B., Junges, S., & Bloem, R. (2018). Shielded decision-making in mdps. *arXiv preprint arXiv:1807.06096*.

Kadota, Y., Kurano, M., & Yasuda, M. (2006). Discounted markov decision processes with utility constraints. *Computers & Mathematics with Applications*, *51*(2), 279–284.

Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial intelligence*, *101*(1-2), 99–134.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, *4*, 237–285.

Kashevnik, A., Shchedrin, R., Kaiser, C., & Stocker, A. (2021). Driver distraction detection methods: A literature review and framework. *IEEE Access*, *9*, 60063–60076.

Kashima, H. (2007). Risk-sensitive learning via minimization of empirical conditional value-at-risk. *IEICE TRANSACTIONS on Information and Systems*, *90*(12), 2043–2052.

Kearns, M., & Singh, S. (2002). Near-optimal reinforcement learning in polynomial time. *Machine learning*, *49*, 209–232.

Khan, M. Q., & Lee, S. (2019). A comprehensive survey of driving monitoring and assistance systems. *Sensors*, *19*(11), 2574.

Kong, Q., Zhang, L., & Xu, X. (2021a). Constrained policy optimization algorithm for autonomous driving via reinforcement learning. In *2021 6th international conference on image, vision and computing (icivc)* (pp. 378–383).

Kong, Q., Zhang, L., & Xu, X. (2021b). Lane keeping algorithm for autonomous driving via safe reinforcement learning. In *Knowledge science, engineering and management: 14th international conference, ksem 2021, tokyo, japan, august 14–16, 2021, proceedings, part iii 14* (pp. 439–450).

Koppejan, R., & Whiteson, S. (2011). Neuroevolutionary reinforcement learning for generalized control of simulated helicopters. *Evolutionary intelligence*, *4*, 219–241.

Krishnamurthy, A., Agarwal, A., & Langford, J. (2016). Pac reinforcement learning with rich observations. *Advances in Neural Information Processing Systems*, *29*.

Law, E. L. (2005). Risk-directed exploration in reinforcement learning.

Li, L., Zhao, W., & Wang, C. (2022). Pomdp motion planning algorithm based on multi-modal driving intention. *IEEE Transactions on Intelligent Vehicles*, *8*(2), 1777–1786.

Li, Q., Hou, L., Wang, Z., Wang, W., Zeng, C., Yuan, Q., & Cheng, B. (2021). Drivers' visual-distracted take-over performance model and its application on adaptive adjustment of time budget. *Accident Analysis & Prevention*, *154*, 106099.

Li, Q., Peng, Z., Feng, L., Zhang, Q., Xue, Z., & Zhou, B. (2022). Metadrive:

Composing diverse driving scenarios for generalizable reinforcement learning. *IEEE transactions on pattern analysis and machine intelligence*, *45*(3), 3461–3475.

Liang, Y., Reyes, M. L., & Lee, J. D. (2007). Real-time detection of driver cognitive distraction using support vector machines. *IEEE transactions on intelligent transportation systems*, *8*(2), 340–350.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., ... Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Lin, K., Li, D., Li, Y., Chen, S., Liu, Q., Gao, J., ... Gong, L. (2023). Tag: Teacher-advice mechanism with gaussian process for reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*.

Liu, C., Chu, X., Wu, W., Li, S., He, Z., Zheng, M., ... Li, Z. (2022). Human–machine cooperation research for navigation of maritime autonomous surface ships: A review and consideration. *Ocean Engineering*, *246*, 110555.

Liu, L., Lu, S., Zhong, R., Wu, B., Yao, Y., Zhang, Q., & Shi, W. (2020). Computing systems for autonomous driving: State of the art and challenges. *IEEE Internet of Things Journal*, *8*(8), 6469–6486.

Liu, Y., Halev, A., & Liu, X. (2021). Policy learning with constraints in model-free reinforcement learning: A survey. In *The 30th international joint conference on artificial intelligence (ijcai)*.

Ly, A. O., & Akhloufi, M. (2020). Learning to drive by imitation: An overview of deep behavior cloning methods. *IEEE Transactions on Intelligent Vehicles*, *6*(2), 195–209.

Martín H, J. A., & de Lope, J. (2009). Learning autonomous helicopter flight with evolutionary reinforcement learning. In *Computer aided systems theory-eurocast 2009: 12th international conference, las palmas de gran canaria, spain, february 15-20, 2009, revised selected papers 12* (pp. 75–82).

Meng, L., Gorbet, R., & Kulić, D. (2021). Memory-based deep reinforcement learning for pomdps. In *2021 ieee/rsj international conference on intelligent robots and systems (iros)* (pp. 5619–5626).

Mihatsch, O., & Neuneier, R. (2002). Risk-sensitive reinforcement learning. *Machine learning*, *49*, 267–290.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... others (2015). Human-level control through deep reinforcement learning. *nature*, *518*(7540), 529–533.

Modi, A., Chen, J., Krishnamurthy, A., Jiang, N., & Agarwal, A. (2021). Model-free representation learning and exploration in low-rank mdps. *arXiv preprint arXiv:2102.07035*.

Moldovan, T. M., & Abbeel, P. (2012). Safe exploration in markov decision processes. *arXiv preprint arXiv:1205.4810*.

Monfort, S. S., Reagan, I. J., Cicchino, J. B., Hu, W., Gershon, P., Mehler,

B., & Reimer, B. (2022). Speeding behavior while using adaptive cruise control and lane centering in free flow traffic. *Traffic injury prevention*, *23*(2), 85–90.

Morimura, T., Sugiyama, M., Kashima, H., Hachiya, H., & Tanaka, T. (2012). Parametric return density estimation for reinforcement learning. *arXiv preprint arXiv:1203.3497*.

Ni, T., Eysenbach, B., & Salakhutdinov, R. (2022). Recurrent model-free rl can be a strong baseline for many pomdps. In *International conference on machine learning* (pp. 16691–16723).

Nidamanuri, J., Nibhanupudi, C., Assfalg, R., & Venkataraman, H. (2021). A progressive review: Emerging technologies for adas driven solutions. *IEEE Transactions on Intelligent Vehicles*, *7*(2), 326–341.

Nilim, A., & El Ghaoui, L. (2005). Robust control of markov decision processes with uncertain transition matrices. *Operations Research*, *53*(5), 780–798.

Okuyama, T., Gonsalves, T., & Upadhay, J. (2018). Autonomous driving system based on deep q learnig. In *2018 international conference on intelligent autonomous systems (icoias)* (pp. 201–205).

Ondruš, J., Kolla, E., Vertal', P., & Šarić, Ž. (2020). How do autonomous cars work? *Transportation Research Procedia*, *44*, 226–233.

Oviedo-Trespalacios, O., Tichon, J., & Briant, O. (2021). Is a flick-through enough? a content analysis of advanced driver assistance systems (adas) user manuals. *Plos one*, *16*(6), e0252688.

O'Neill, T., McNeese, N., Barron, A., & Schelble, B. (2022). Human–autonomy teaming: A review and analysis of the empirical literature. *Human factors*, *64*(5), 904–938.

Palade, V., & Deo, A. (2022). Artificial intelligence in cars: How close yet far are we from fully autonomous vehicles? *International Journal on Artificial Intelligence Tools*, *31*(3), 2241005.

Pant, Y. V., Kumaravel, B. T., Shah, A., Kraemer, E., Vazquez-Chanlatte, M., Kulkarni, K., ... Seshia, S. A. (2022). Modeling and influencing human attentiveness in autonomy-to-human perception hand-offs. In *2022 ieee 25th international conference on intelligent transportation systems (itsc)* (pp. 2585–2592).

Paternain, S., Chamon, L., Calvo-Fullana, M., & Ribeiro, A. (2019). Constrained reinforcement learning has zero duality gap. *Advances in Neural Information Processing Systems*, *32*.

Perera, D., Wang, Y.-K., Lin, C.-T., Nguyen, H., & Chai, R. (2022). Improving eeg-based driver distraction classification using brain connectivity estimators. *Sensors*, *22*(16), 6230.

Philip, P., Sagaspe, P., Moore, N., Taillard, J., Charles, A., Guilleminault, C., & Bioulac, B. (2005). Fatigue, sleep restriction and driving performance. *Accident Analysis & Prevention*, *37*(3), 473–478.

Pleines, M., Pallasch, M., Zimmer, F., & Preuss, M. (2023). Memory gym: Partially observable challenges to memory-based agents. In *The eleventh international conference on learning representations*.

Pohl, J., Birk, W., & Westervall, L. (2007). A driver-distraction-based lane-

keeping assistance system. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, *221*(4), 541–552.

Pusse, F., & Klusch, M. (2019). Hybrid online pomdp planning and deep reinforcement learning for safer self-driving cars. In *2019 ieee intelligent vehicles symposium (iv)* (pp. 1013–1020).

Qin, L., Li, Z. R., Chen, Z., Bill, M. A., & Noyce, D. A. (2019). Understanding driver distractions in fatal crashes: An exploratory empirical analysis. *Journal of Safety Research*, *69*, 23–31.

Quintía Vidal, P., Iglesias Rodríguez, R., Rodríguez González, M. Á., & Vázquez Regueiro, C. (2013). Learning on real robots from experience and simple user feedback.

Rajasekhar, M., & Jaswal, A. K. (2015). Autonomous vehicles: The future of automobiles. In *2015 ieee international transportation electrification conference (itec)* (pp. 1–6).

Regan, M. A., Lee, J. D., & Young, K. (2008). *Driver distraction: Theory, effects, and mitigation*. CRC press.

Roozendaal, J., Johansson, E., Winter, J. d., Abbink, D., & Petermeijer, S. (2021). Haptic lane-keeping assistance for truck driving: a test track study. *Human factors*, *63*(8), 1380–1395.

Sagberg, F., Selpi, Bianchi Piccinini, G. F., & Engström, J. (2015). A review of research on driving styles and road safety. *Human factors*, *57*(7), 1248–1275.

Salehinejad, H., Sankar, S., Barfett, J., Colak, E., & Valaee, S. (2017). Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*.

Santara, A., Rudra, S., Buridi, S. A., Kaushik, M., Naik, A., Kaul, B., & Ravindran, B. (2021). Madras: Multi agent driving simulator. *Journal of Artificial Intelligence Research*, *70*, 1517–1555.

Sato, M., Kimura, H., & Kobayashi, S. (2001). Td algorithm for the variance of return and mean-variance reinforcement learning. *Transactions of the Japanese Society for Artificial Intelligence*, *16*(3), 353–362.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Seenivasan, R. D. (2023). *Trends in electrodermal activity, heart rate and temperature during distracted driving among young novice drivers.* (Unpublished master's thesis). University of Waterloo.

Shahini, F., & Zahabi, M. (2022). Effects of levels of automation and non-driving related tasks on driver performance and workload: A review of literature and meta-analysis. *Applied Ergonomics*, *104*, 103824.

Shalev-Shwartz, S., Shammah, S., & Shashua, A. (2016). Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*.

Shen, S., Ausin, M. S., Mostafavi, B., & Chi, M. (2018). Improving learning & reducing time: A constrained action-based reinforcement learning approach. In *Proceedings of the 26th conference on user modeling, adaptation and personalization* (pp. 43–51).

Simão, T. D., Suilen, M., & Jansen, N. (2023). Safe policy improvement for pomdps via finite-state controllers. *arXiv preprint arXiv:2301.04939*.

Singh, B., Kumar, R., & Singh, V. P. (2022). Reinforcement learning in robotic applications: a comprehensive survey. *Artificial Intelligence Review*, 1–46.

Sri Mounika, T., Phanindra, P., Sai Charan, N., Kranthi Kumar Reddy, Y., & Govindu, S. (2022). Driver drowsiness detection using eye aspect ratio (ear), mouth aspect ratio (mar), and driver distraction using head pose estimation. In *Ict systems and sustainability: Proceedings of ict4sd 2021, volume 1* (pp. 619–627).

Stewart, T. (2023). *Overview of motor vehicle traffic crashes in 2021* (Tech. Rep.).

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Tamar, A., Xu, H., & Mannor, S. (2013). Scaling up robust mdps by reinforcement learning. *arXiv preprint arXiv:1306.6189*.

Tan, H. H., & Lim, K. H. (2019). Review of second-order optimization techniques in artificial neural networks backpropagation. In *Iop conference series: materials science and engineering* (Vol. 495, p. 012003).

Tang, J., Singh, A., Goehausen, N., & Abbeel, P. (2010). Parameterized maneuver learning for autonomous helicopter flight. In *2010 ieee international conference on robotics and automation* (pp. 1142–1148).

Tango, F., & Botta, M. (2013). Real-time detection system of driver distraction using machine learning. *IEEE Transactions on Intelligent Transportation Systems*, *14*(2), 894–905.

Thomaz, A. L., & Breazeal, C. (2008). Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artificial Intelligence*, *172*(6-7), 716–737.

Uehara, M., Zhang, X., & Sun, W. (2021). Representation learning for online and offline rl in low-rank mdps. *arXiv preprint arXiv:2110.04652*.

van de Merwe, K., Mallam, S., & Nazir, S. (2022). Agent transparency, situation awareness, mental workload, and operator performance: A systematic literature review. *Human Factors*, 00187208221077804.

Van Houdt, G., Mosquera, C., & Nápoles, G. (2020). A review on the long short-term memory model. *Artificial Intelligence Review*, *53*, 5929–5955.

Wang, X., Zhang, J., Hou, D., & Cheng, Y. (2023). Autonomous driving based on approximate safe action. *IEEE Transactions on Intelligent Transportation Systems*.

Wen, L., Duan, J., Li, S. E., Xu, S., & Peng, H. (2020). Safe reinforcement learning for autonomous vehicles through parallel constrained policy optimization. In *2020 ieee 23rd international conference on intelligent transportation systems (itsc)* (pp. 1–7).

Wierstra, D., Foerster, A., Peters, J., & Schmidhuber, J. (2007). Solving deep memory pomdps with recurrent policy gradients. In *Artificial neural networks–icann 2007: 17th international conference, porto, portugal, september 9-13, 2007, proceedings, part i 17* (pp. 697–706).

Wollmer, M., Blaschke, C., Schindl, T., Schuller, B., Farber, B., Mayer, S., & Trefflich, B. (2011). Online driver distraction detection using long short-term memory. *IEEE Transactions on Intelligent Transportation Systems*, *12*(2), 574–582.

Wymann, B., Espié, E., Guionneau, C., Dimitrakakis, C., Coulom, R., & Sumner, A. (2000). Torcs, the open racing car simulator. *Software available at http://torcs. sourceforge. net*, *4*(6), 2.

Xiong, X., Wang, J., Zhang, F., & Li, K. (2016). Combining deep reinforcement learning and safety based control for autonomous driving. *arXiv preprint arXiv:1612.00147*.

Ye, M., Osman, O. A., Ishak, S., & Hashemi, B. (2017). Detection of driver engagement in secondary tasks from observed naturalistic driving behavior. *Accident Analysis & Prevention*, *106*, 385–391.

Yi, D., Su, J., Hu, L., Liu, C., Quddus, M., Dianati, M., & Chen, W.-H. (2019). Implicit personalization in driving assistance: State-of-the-art and open issues. *IEEE Transactions on Intelligent Vehicles*, *5*(3), 397–413.

Young, K., Regan, M., & Hammer, M. (2007). Driver distraction: A review of the literature. *Distracted driving*, *2007*, 379–405.

Yu, M., Yang, Z., Kolar, M., & Wang, Z. (2019). Convergent policy optimization for safe reinforcement learning. *Advances in Neural Information Processing Systems*, *32*.

Zahabi, M., Razak, A. M. A., Shortz, A. E., Mehta, R. K., & Manser, M. (2020). Evaluating advanced driver-assistance system trainings using driver performance, attention allocation, and neural efficiency measures. *Applied ergonomics*, *84*, 103036.

Zakaria, N., Shapiai, M., Ghani, R., Yasin, M., Ibrahim, M., & Wahid, N. (2023). Lane detection in autonomous vehicles: A systematic review. *IEEE Access*.

Zhang, S., & Abdel-Aty, M. (2022). Drivers' visual distraction detection using facial landmarks and head pose. *Transportation research record*, *2676*(9), 491–501.

Zhang, X., Song, Y., Uehara, M., Wang, M., Agarwal, A., & Sun, W. (2022). Efficient reinforcement learning in block mdps: A model-free representation learning approach. In *International conference on machine learning* (pp. 26517–26547).

Zhang, Y., Vuong, Q., & Ross, K. (2020). First order constrained optimization in policy space. *Advances in Neural Information Processing Systems*, *33*, 15338–15349.

Zhu, Z., & Zhao, H. (2021). A survey of deep rl and il for autonomous driving policy learning. *IEEE Transactions on Intelligent Transportation Systems*, *23*(9), 14043–14065.

8 Appendix

8.1 2D and 3D plots

In Figure 28 and Figures 29, 30, 31, and 32, the data received by the algorithms is visually presented in both 2D and 3D plots. These plots aim to illustrate the potential for data separation within the provided input space. Specifically, these visualizations showcase the distribution of data points in relation to the variables being considered.

In these plots, the x and y axes represent different input variables, such as track position, angle, driver steering, and driver acceleration. Each point on the plot corresponds to a data sample, and the distribution of these points offers insights into the separability of data based on these variables.

For instance, Figure 28 provides a 2D representation of the data distribution, while Figures 29, 30, 31, and 32 present 3D plots that incorporate additional dimensions for a more comprehensive view of the data.

The specified ranges for each variable help define the boundaries within which the data points are distributed. Analyzing these visualizations can aid in understanding how the algorithms can leverage the provided input data to make informed decisions and navigate the driving environment effectively.

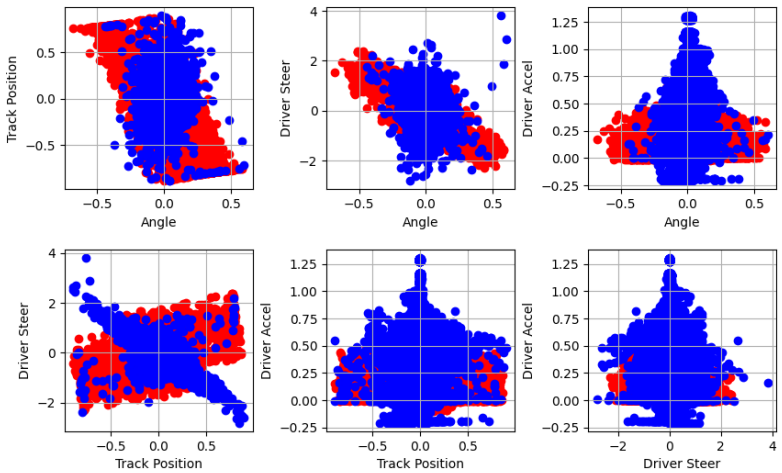


Figure 28: 2D-plots of the inputs

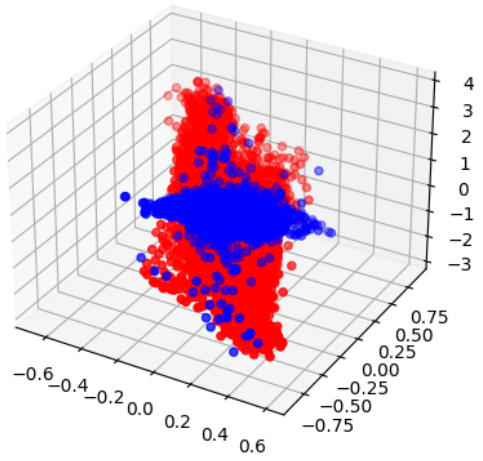


Figure 29: 3D-plots of the inputs angle, track position and driver steering

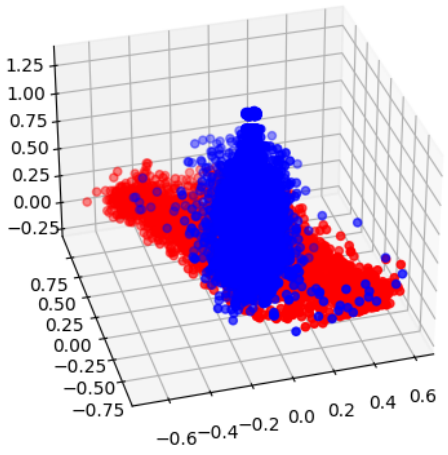


Figure 30: 3D-plots of the inputs angle, track position and driver acceleration

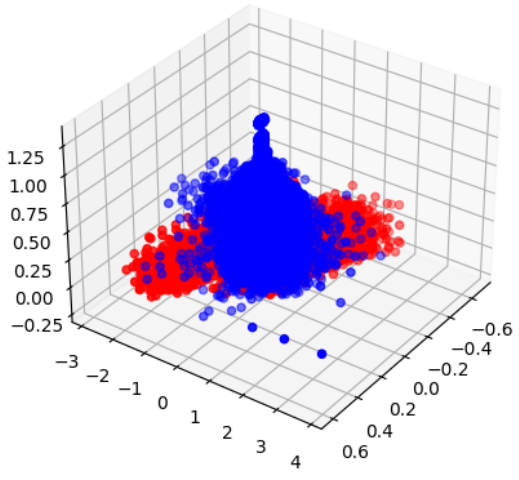


Figure 31: 3D-plots of the inputs angle, driver steering and driver acceleration

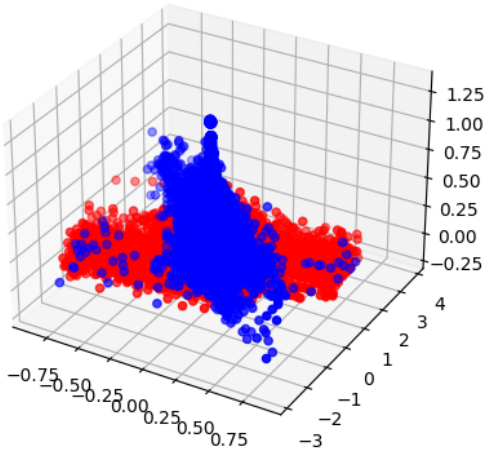


Figure 32: 3D-plots of the inputs track position, driver steering and driver acceleration

8.2 Additional Experiments

In these additional experiments, efforts were made to enhance safety within the unconstrained TD3 algorithm.

8.2.1 Data Augmentation

Data augmentation was implemented as a means of achieving this objective. Instead of uniformly adding each state-action-reward tuple to the memory used for agent training, the data was divided into two categories:

Safe States Augmentation: In this experiment, tuples corresponding to states where the vehicle was closer to the centerline than the 0.7m boundary were added an additional time to the memory. The rationale behind this approach was to increase the likelihood of the agent learning how to act when in a safe state. Improved learning in safe states should, in turn, lead to the agent more effectively remaining in these safe states during operation.

Unsafe States Augmentation: In this experiment, tuples representing states where the vehicle was close to crashing were added an additional time to the memory. The aim here was to boost the chances of the agent learning effective recovery actions when it finds itself in a perilous situation. This increased the likelihood of the agent successfully recovering to a safe state after encountering an unsafe state.

The results presented in Figure 33 demonstrate that the agent trained on the memory with an additional emphasis on safe states exhibits slightly better performance compared to the other scenarios. This outcome suggests that by providing the agent with a better understanding of safe states and effective actions within them, it can achieve higher rewards and, crucially, maintain a safer driving behaviour.

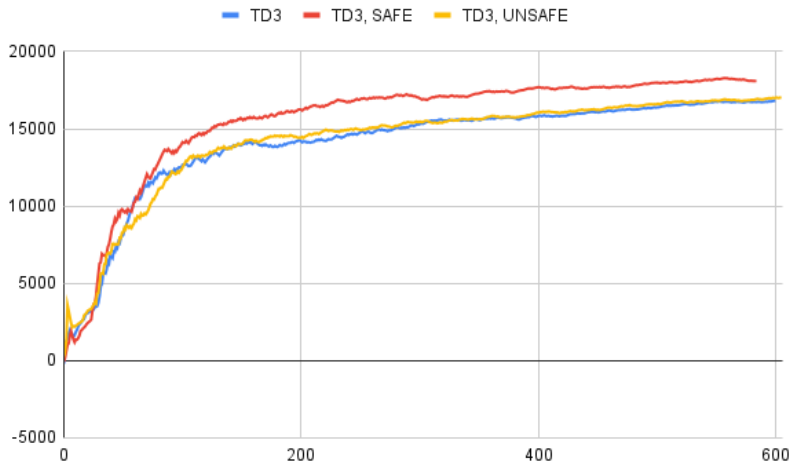
These experiments highlight the potential benefits of tailored data augmentation strategies to reinforce safety in the reinforcement learning process. By selectively emphasizing safe states, agents can be better equipped to navigate challenging driving scenarios and maintain safety during operation.

8.2.2 History Length

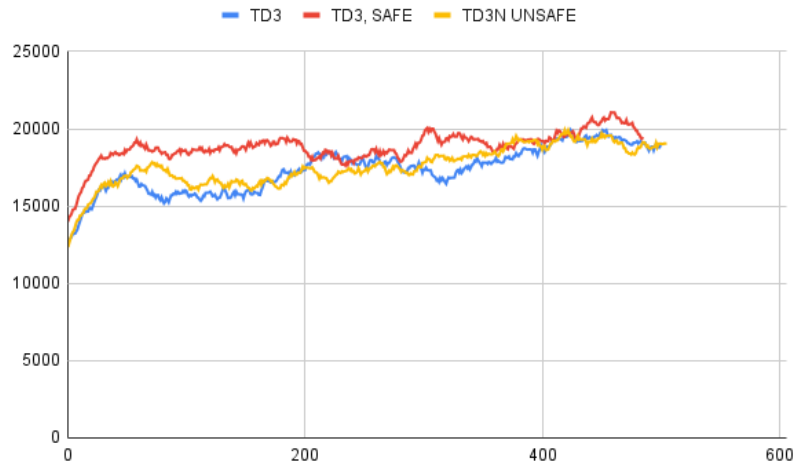
An additional experiment was conducted to investigate the impact of increasing the length of input history on the agent's learning capabilities. In this experiment, the objective was to create a memory of past states that the agent could learn from, allowing it to discern correlations between previous inputs and the current reward. Specifically, TD3 and TD3-LSTM were trained using three different input configurations:

- Single Timestep: In this setting, the agent received inputs from a single timestep, which provided information only about the current state.
- Previous 5 Timesteps: Here, the agent's inputs consisted of data from the previous 5 timesteps, allowing it to consider a short history of past states and actions.
- Previous 10 Timesteps: In this configuration, the agent's inputs encompassed data from the previous 10 timesteps, affording it a more extended history of past states and actions to learn from.

This experiment aimed to determine whether an extended input history would improve the agent's ability to understand the relationship between past actions and current rewards. The results of this experiment can provide insights into the benefits



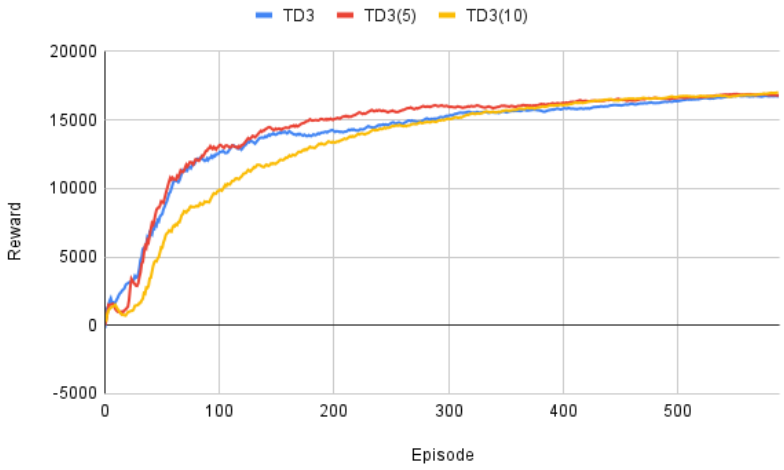
(a)



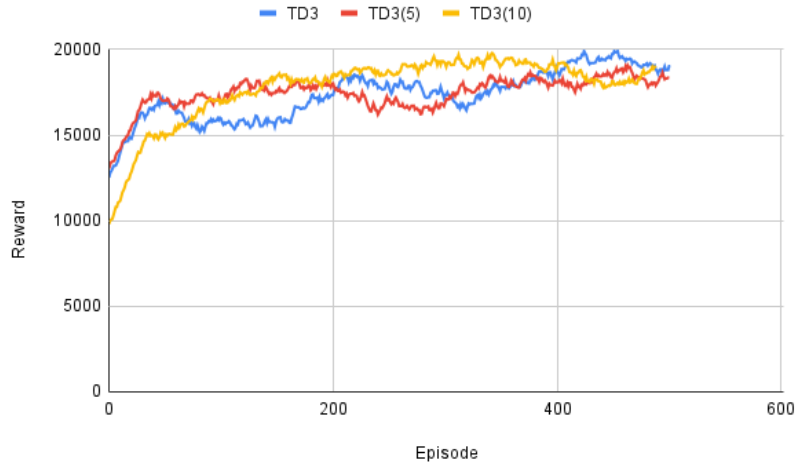
(b)

Figure 33: (a) Average reward of TD3 and TD3 with either safe or unsafe states added an additional time over all episodes (b) Average reward of TD3 and TD3 with either safe or unsafe states added an additional time over 100 episode window

of incorporating longer historical information into the learning process, potentially aiding the agent in making more informed decisions.



(a)

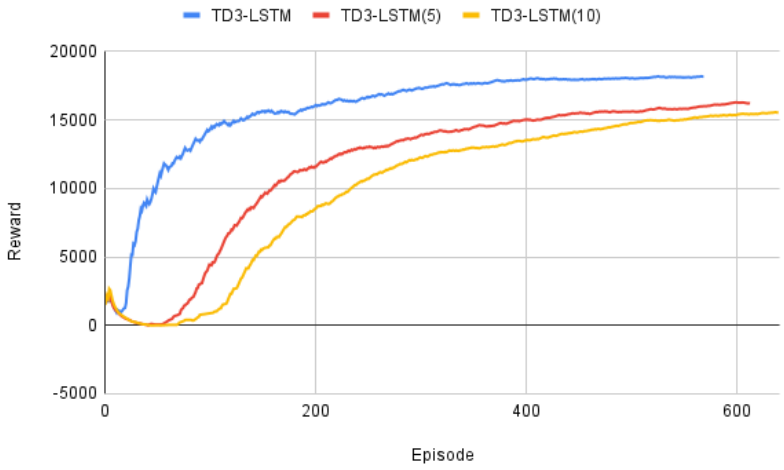


(b)

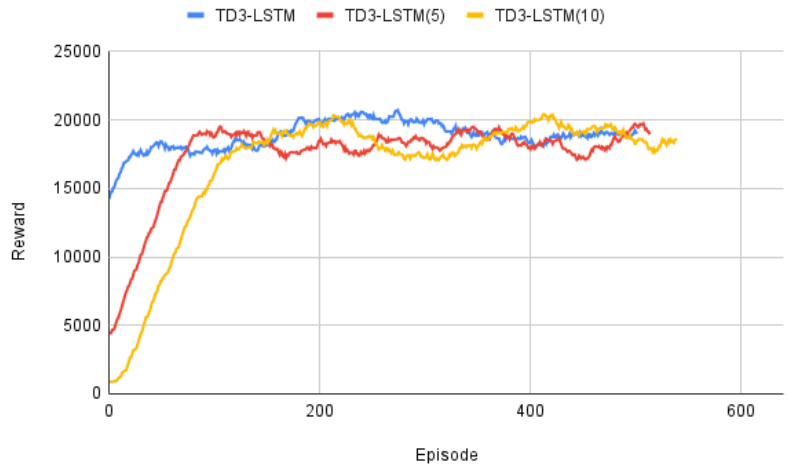
Figure 34: (a) Average reward of TD3, TD3(5) and TD3(10) over all episodes, the number between parentheses indicates the number of previous inputs given to the algorithm (b) Average reward of TD3, TD3(5) and TD3(10) over 100 episode window

8.2.3 Unsupervised Clustering

In an attempt to further improve the agent's learning process, an experiment was conducted involving the use of unsupervised clustering. The underlying idea was



(a)



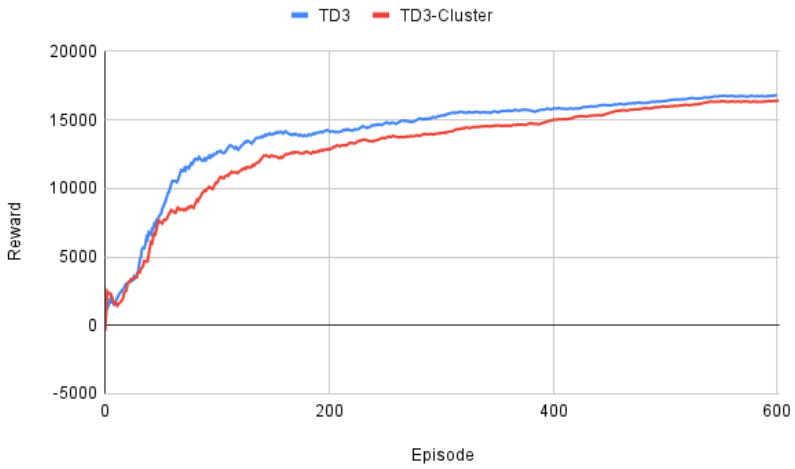
(b)

Figure 35: (a) Average reward of TD3-LSTM, TD3-LSTM(5) and TD3-LSTM(10) over all episodes (b) Average reward of TD3-LSTM, TD3-LSTM(5) and TD3-LSTM(10) over 100 episode window

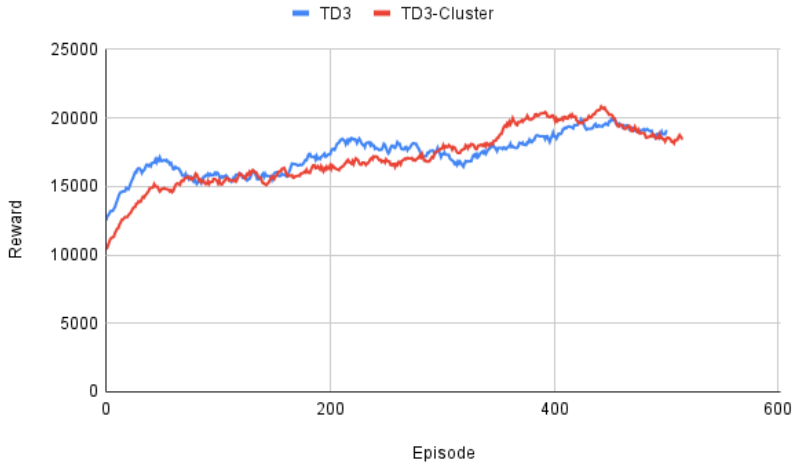
to leverage an unsupervised clustering algorithm, specifically Mini-Batch K-Means, to separate and cluster the input data online. This clustered data would then be incorporated as additional information provided to the agent, effectively transforming the Block MDP (BMDP) into a standard Markov Decision Process (MDP).

The rationale behind this experiment was based on the premise that if there is a

natural separation within the input data, clustering could highlight this separation and potentially enhance the agent's decision-making process. However, the results of this experiment, as illustrated in Figure 36, did not demonstrate any improvement in the agent's performance when using the clustering method. Consequently, this approach was considered non-viable and was not pursued further in the study.



(a)



(b)

Figure 36: (a) Average reward of TD3 and TD3 with a Mini-Batch K-Means clustering method over all episodes (b) Average reward of TD3 and TD3 with a Mini-Batch K-Means clustering method over 100 episode window