Utrecht University

Jokke Mats Jansen

Master's Thesis proposal

Artificial Intelligence
Faculty of Science

Supervisors

First: Dr. Shihan Wang
Second: Dr. Leendert van Maanen

2021

Abstract

Contents

Chapter 1

Introduction

1.1 General context

Fully autonomously driving cars have the potential to rule out human driving error which is at least a contributing factor to most accidents today. Many social and technical obstacles have yet to be overcome until fully autonomous cars become market-ready (Maurer et al., 2016). However, many Advanced driver assistance systems (ADAS) such as adaptive cruise control, lane keeping and changing assistance, and automated collision mitigation are already deployed in modern cars.

The extend to which an ADAS takes control varies. While the potential prevention of human-error caused accidents increases with the elaborateness of intervention by an assistant system, excessive intervention drastically limits the driver's autonomy. A loss of driver autonomy can turn driving into a monotonous and tedious supervisory task. Drivers easily become inattentive and are more prone to distract themselves, for example by looking on their phone. However, as long as assistance systems are not sufficient to handle all situations, a concentrated human will remain necessary to take actions in situations the assistance system fails. Leaving the driver with a pure supervision task can lead to a long transition time for the driver when it is required to retake control of the vehicle (Wang et al., 2020). Being in control means having to concentrate. Therefore, the goal should be to keep the human driver in control as much as possible but to assist when help is really needed. As a result, driving pleasure is enhanced and drivers are prevented from relying too heavily on the assistance systems.

15% of injury crashes in the US were associated with driver distraction in 2018 (NHTSA, 2020). Therefore, it seems reasonable to make the extend of the ADAS's activation dependent on the driver's level of attention. Whenever a driver is inattentive or distracted, an ADAS needs to be particularly sensitive. Yet in what way can an assistance system detect that a driver is distracted?

1.2 Problem overview

There have bee attempts to develop systems that determining the psychological state of a driver in real time while driving. The application of eye tracking technology or analysis of camera footage using machine learning models is conceivable and has led to promising results. However, as promising as promising as these methods are, they are not readily available yet. Furthermore, they are quite intrusive and could be seen as an encroachment on privacy. Thus, the driver's level of attention is essentially unknown. Nevertheless, one can assume that distracted drivers act differently. Among other things, deviations such as increased reaction times and altered steering behavior are likely.

A two-fold problem arises: On the one hand, a lane keeping assistance system has to be able to identify when drivers are distracted by observing their behavior. On the other hand, the system must have the capability to provide meaningful assistance.

Intuitively, it seems reasonable to solve both problems individually; using a model that takes the available data, such as the driver's steering behavior, as input to classify whether the driver is distracted, and another model that assists a distracted driver in steering the car. Both could be trained using example data. However, this supervised approach entails two challenges: First, driving is a sequential decision process. An action influences future actions and driving situations in which decisions have to be made are essentially unique. Second, an activation of the assistance system can affect on how drivers behave. Drivers may adjust to the system. It is not possible to create a dataset that covers these dynamics entirely.

Reinforcement learning (RL) allows a system to learn and represent its behavior by interacting with it rather than learning from past experience. Therefore, RL constitutes a promising method to develop an ADAS or even a fully autonomous driving agent and its application in this area is a very active research area with many successful results (Kiran et al., 2021). Because learning is achieved by exploration rather than from examples, RL is able to perform well in sequential decision making tasks. Moreover, reinforcement learning algorithms can be extended to support learning with a partially observable state (Sutton & Barto, 2018, p. 466). While the agent can perceive the car's environment with sensors, the attention level of the driver is hidden. Nevertheless, only one RL agent is needed to both learn how to assist in driving and to classify when this is desired due to a distracted driver.

The result is a shared control scenario where both the human driver and the agent can actively control (e.g. steer, brake, accelerate) the car simultaneously. Each can indirectly perceive the actions of the other by observing the state of the car. Thereby, on the one hand, the agent is able to analyze the driving behavior of the human and, on the other hand, the human can notice the assistance of the agent and may adapt to it.

Learning in a real-world situation is not feasible in the context of this thesis. Despite the inevitable high safety risk, it would also require an enormous investment of resources, and the complexity of a real-world driving scenario represents an insurmountable obstacle. Instead, the agent learns in a simulation environment with a simulated human driver. The Open Racing Car Simulator (TORCS), a racing car simulator that allows to model various driving situations (Espié et al., 2005) is used as simulation environment. It offers a good balance between realism and resource efficiency and has been utilized in many papers regarding RL-based driving before. An Adaptive Control of Thought-Rational (ACT-R) cognitive model is employed to simulate the human's actions. The model is able to keep the car in its lane, perform lane changes, and avoid collision with other road users. It captures behavioral differences between attentive and inattentive human drivers. Furthermore, a human-subject experiment is performed in the TORCS simulation environment to identify if the agent is able to generalize well enough to be useful for actual human drivers.

1.3 Contributions

The main goal and differentiator of this thesis is to utilize reinforcement learning for a shared-control driving task with unknown attention of the human driver. One of the main challenges is that near real-time decisions of the agent are necessary. This drastically limits the time available for online planning. Accordingly, the implementation needs to be very efficient. Solving the problem using an algorithm that requires discretized states (e.g. steering angle categories) is contrasted with a solution using an algorithm directly supporting continuous states.

1.4 Outline

The rest of the proposal is organised as follows:

Chapter 4 describes the problem in a formal manner using a Partially observable Markov decision process (POMDP).

Chapter 2 summarizes and reviews important literature that serves as the foundation of the thesis.

?? presents the initial research plan for the rest of the thesis, including important milestones and deadlines.

Chapter 2

Literature

This thesis focuses on efficient online POMDP planning. The two most notable fast online POMDP algorithms are DESPOT (Ye et al., 2017) and POMCP (Silver & Veness, 2010). Both apply Monte Carlo tree search to evaluate the quality of candidate policies. At each time point, a simulator of the environment is used to form a search tree from multiple simulations in order to evaluate the resulting hypothetical histories by their mean return, leading to a real action of the agent in the environment and thus to a new real observation (Silver & Veness, 2010). DESPOT addresses and improves upon POMCP's problem of a poor worst-case performance bound (Ye et al., 2017).

Both POMCP and DESPOT can handle continuous state spaces but would have to be modified in order to support continuous action or observation spaces (Sunberg & Kochenderfer, 2018). Sunberg and Kochenderfer provide two online algorithms for POMDP with continuous state, action, and observation spaces: POMCPOW for simulating approximate state trajectories, and PFT-DPW for simulating approximate belief trajectories.

Offline and online approaches can be combined by using an approximate policy computed offline as a default policy (Gelly & Silver, 2007), or by considering a sequence of macro-actions to reduce the size of the search horizon (He et al., 2011). Especially when there is only very little time for online planning, incorporating an offline approximation into an online approach can be useful (Ross et al., 2008).

Lam and Sastry, 2014 provide a framework for using a POMDP to model a Human-in-the-loop (HITL) control system. Their framework serves as foundation for this thesis and is further described in Chapter 4. The framework is used in a case study where an agent assists a potentially drowsy human driver in keeping the car centered in its lane. Whether or not the driver is drowsy remains unknown to the agent. The agent's estimation of the driver's drowsiness is based on the humans actions such as turning the steering wheel and opening or closing its eyes. Intervention by the agent is possible both by alerting the driver with a warning, and by actively steering the car. Any intervention is penalized with the aim to interfere with the driver as little as possible but as much as required in order to keep the car centered when the driver is drowsy. They approximately solve their POMDP problem with an offline randomized point-based value iteration approach. The policy is computed by iteratively sampling a finite set of random points from the agent's belief space. The agent thus interacts randomly with the environment in order to find an approximation of the optimal policy. The employed model of the human's internal state is rather simplistic and based on handcrafted transition probabilities. The state and action spaces are discrete.

Sadigh et al., 2016 evaluate how active probing can be utilized by autonomous vehicles in driving scenarios to reduce their uncertainty about a hidden psychological state of human drivers on the road. Three different scenarios are modeled: First, the agent wants to cross an intersections with other cars driving on the crossed road. The autonomous car can cautiously approach in order to probe the other cars for attentiveness; if they react by reducing their speed, they are likely attentive. Second, the agent drives on a highway with human drivers approaching from behind. The goal is to avoid rear-end collisions, which are especially likely in the case of inattentive human drivers. Active probing can be performed by the agent through braking. If an approaching driver does not slow down, the driver is most likely not paying attention. Third, the agent actively probes for an aggressive or timid driving style of other drivers by nudging into their lane. A human driver is expected to either slow down to allow the agent to switch lanes (timid driving style), or speed up to discourage the agent to switch lanes (agressive). This approach of actively provoking human responses rather than just passively observing leads to a significant improvement in classifying the human drivers' hidden attentiveness. It can potentially also be utilized in a shared control setting. The agent could utilize minor interventions to reduce uncertainty about the human's internal state.

Furthermore, the work of Sadigh et al., 2016 is relevant with regard to how they represent their problem as a POMDP with a continuous state and action space and plan online using Model Predictive Control (MPC). At every time step, the agent uses an embedded human model to make predictions over a finite horizon about the actions a human would take in response to its own actions. The agent knows how a human would act in different psychological states, the state itself, however, is hidden. It is assumed that the human always tries to maximize its reward. The agent chooses a policy that maximizes its reward while accounting for the human's (potential) actions depending on the hidden psychological state the agent believes the human to be in. The real human actions that are observed after the agent executes an action are used to update the agent's belief about the human's internal state. The human model is learned a priori using inverse reinforcement learning (IRL) using demonstrations of human behavior for which the human's internal state is known.

Wang et al., 2020 provide an overview of papers regarding decision making and human driver modeling for driver-vehicle shared control scenarios. Insight about recent developments, different architectures, and remaining challenges is provided. Of particular relevance are the different modes for the communication of authority between human and agent and the cognitive modeling approaches that are discussed.

Chapter 3

Background

3.1 Sequential decision making

Lane keeping of a car is a sequential decision making task. Every steering action that is performed directly influences the choice of the best succeeding steering actions. Markov decision processs (MDPs) are well suited and widely used to model sequential decision making tasks. An MDP is a discrete time framework for a decision maker, the agent, to interact with an environment. At every time step, the environment is in a certain state, fully observable by the agent. The agent interacts with the environment by performing an action that determines the next state of the environment. The underlying assumption, the Markov property, is that the next state of the environment only depends on its current state and the agent's action. The transition to a succeeding state after an action has been performed does not need to be deterministic but can be probabilistic, accounting for randomness in the environment. After performing an action, the agent receives a numerical reward. The agent's goal is to maximize the cumulative reward it receives over time. An action that leads to a high immediate reward is not optimal if another action leads to a higher cumulative reward in the long run. Thus, the agent needs to find an optimal policy that decides the best action to take in every state. In case the state transition probabilities are known to the agent, 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.

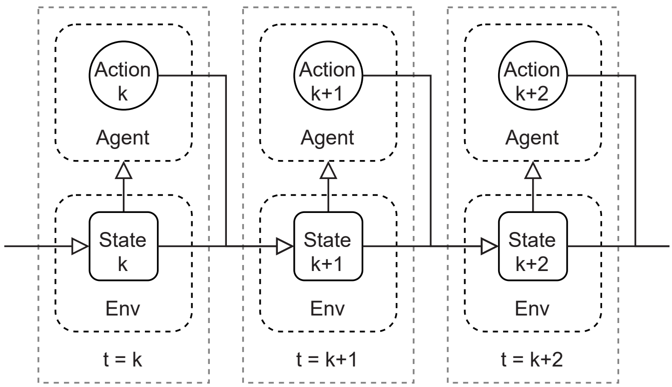


Figure 3.1: Markov decision process (MDP)

Assisting a human driver in the lane keeping task is essentially a sequential decision making task as well. However, the agent that is assisting the human driver does not know about the driver's internal psychological state, and therefore her attention. A distracted driver may steer poorly and needs assistance. But how can the agent tell whether the driver is distracted? Reading the driver's mind is not feasible and even if it were, it would be too invasive for this task. Instead, the agent needs to estimate the driver's internal state in order to act adequately. A POMDP is a generalization of an MDP that allows to plan under uncertainty. Even without observing the full state of the agent's environment, of which the driver is part of, a POMDP allows the agent to estimate the environment's true state using the partial information it observes. A POMDP serves as the foundation of this thesis. The lane keeping assistance problem this thesis aims to solve can be defined as a POMDP. First, a formal definition is needed.

3.2 Partially observable Markov decision process (POMDP)

The POMDP generalizes the MDP for planning under uncertainty. The environment's true state is unknown to the agent. It has to rely on observations with partial information about the environment's true state to choose its actions. Kaelbling et al., 1998 define a POMDP as a tuple (S, A, T, R, Z, O), where:

- S is the set of all possible states $s \in S$ of the environment. A state describes the environment at a time point. It must not be an all-encompassing description but must include all relevant information to make decisions. The state is hidden from the agent. This is the main difference to an MDP.
- A is the set of all possible actions $a \in A$ the agent can perform in the environment.
- $T : S \times A \times S \rightarrow [0, 1]$ defines the conditional state transition probabilities. $T(s, a, s') = Pr(s'|s, a)$ constitutes the probability of transitioning to state s' after performing action a in state s.
- $R : S \times A \rightarrow \mathbb{R}$ is the reward function providing the agent with a reward of $R(s, a)$ after performing action a in state s.
- Z is the set of all possible observations $z \in Z$. Observations are the agent's source of information about the environment, enabling the agent to estimate the environment's state.
- $O : S \times A \times Z \rightarrow [0, 1]$ defines the conditional observation probabilities. $O(s', a, z) = Pr(z|s', a)$ represents the probability of receiving observation z at state s' after performing action a in the previous state.

At any time, the environment is in some state s. Unlike in the case of an MDP, the agent cannot directly observe the environment's state. Instead, the agent receives an observation z that provides partial information about the current state. The agent can use the observations it perceives over time to estimate the true state of the environment and choose actions accordingly. In order to do so, at any time step t, it has to take into account the complete history of actions and observations until t:

$$h_t = \{a_0, z_1, ..., z_{t-1}, a_{t-1}, z_t\} \tag{3.1}$$

Keeping a collection of all past observations and actions is very memory expensive. A less memory demanding alternative is to only keep a probability distribution over the states at every step, called a belief $b \in B$. $B(s, h)$ denotes the probability of being in state s at time t given history h.

$$B(s, h) = Pr(s_t = s | h_t = h) \tag{3.2}$$

The belief is a sufficient statistic for the agent to form a decision about its next action (Smallwood & Sondik, 1973). Thus, only the belief needs to be kept and continually updated whenever an action is performed and a new observation arises. The agent starts with an initial belief b_0 about the initial state of the environment. At every subsequent time step t, the belief can be updated based on the previous belief b_{t-1}, the last action $a_t - 1$ and the current observation z_t. The previous belief can then be discarded as the history it represents is no longer up-to-date.

Since the history that was represented by

The agent has a belief b about the state space S that represents a probability distribution over the underlying states with $b(s_t)$ denoting the probability of being in state s. The agent acts according to its policy $\pi(b) \in A$ mapping beliefs to actions. When an action a is performed, the agent receives a reward $\mathcal{R}(s, u)$ and the environment's hidden state transitions to $s' \in \mathcal{S}$ resulting in an observation $z' \in \mathcal{O}$. The agent then uses this observation to update its belief.

Solving a POMDP consists in finding an optimal policy $\pi*$ that maximizes the belief-value function $V^\pi(b_t) = \mathbb{E}[\sum_{n=t}^{N} \lambda^{(n-t)} R(s_n, u_n)]$ that represents the cumulative reward obtained starting from initial belief b_t following a policy π over some time horizon N using a discount factor $0 \leq \lambda \leq 1$.

3.3 POMDP Solvers

3.3.1 Offline and online solvers

There are two general approaches to solve a POMDP: offline and online. Following the offline approach, a policy depending on all possible future events is computed in advance. A policy that has been derived by offline planning can then be executed very efficiently online. However, offline planning is hard to scale to complex problems as the number of possible future events grows exponentially with the size of the planning horizon. In the online approach, planning and plan execution are intertwined. Rather than computing a policy for all possible events in advance, the belief is updated at every time step by searching for a single best action for the current belief and executing it. Then, the planning process repeats with the new belief. On the one hand, the scalability is greatly increased. On the other hand, sufficiently more complex online computation

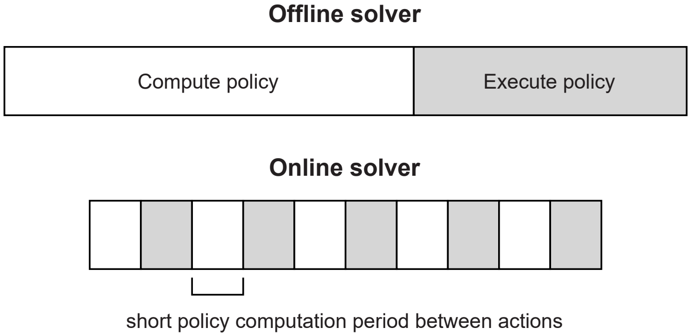


Figure 3.2: Comparison of offline and online solving procedure

than with offline planning is required. The amount of available online planning time at each time step limits the performance. This thesis focuses on the online planning approach.

3.3.2 Curse of dimensionality and curse of history

3.3.3 Monte Carlo tree search solvers

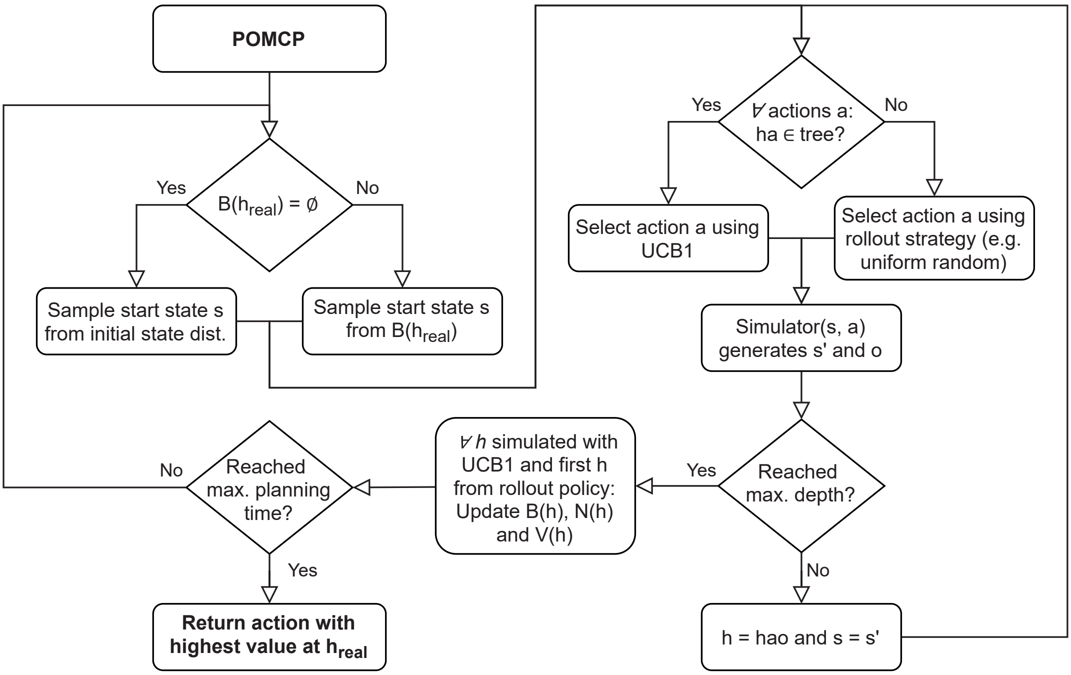
3.3.3.1 Overview

Other solvers have build upon the principle from POMCP of using Monte Carlo tree search for POMDP planning. Notably, there are DESPOT, POMCPOW, ... TODO ... They all have an important requirement in common: The observation probabilities need to be known to the solver. Thereby, particles that are added to the belief can be weighed by how likely their associated observation is at the current belief state after performing a certain action in the prior state. However, the observation probabilities are essentially unknown in our assisted driving scenario.

3.3.3.2 POMCP

Partially observable Monte-Carlo Planning (POMCP) constructs a search tree with nodes representing histories h of actions and observations. At each node, $N(h)$ stores the number of times the represented history h has been encountered, $V(h)$ is the node's value that is approximated by the average return of simulations starting at history h, and $B(h)$ represents the node's belief over the real environment's state. $B(h)$ is a collection of potential states where the likelihood of each state is given by the relative number of times it is included in the collection.

If the belief at the node representing h_{real} is empty, an initial state distribution I is used to sample a start state s for the search. Otherwise, $B(h_{real})$ is utilized. The search tree is then searched in two stages. First, in the case that the search



tree already contains child nodes for all actions at the current history, UCB1 is used for the action selection. Exploration is achieved by increasing the value of rarely-tried actions by an exploration bonus. Second, when the tree is missing a node for a potential action at the current history, a rollout policy is used to select actions. In the most simple case, this means choosing uniformly random over the action space. In either case, the selected action is executed on the start state s, leading to a successive state s', observation o, and reward r. The process is repeated with resulting successive states until a maximum depth of the tree is reached. Afterwards, the beliefs, counts, and values are updated at all nodes for the histories resulting from the UCB1 action selection, and the node for the first history resulting from the rollout policy. The belief is updated by adding the successive states s' from the simulator to the collections $B(ho)$ in the nodes. If the maximum planning time has not yet been reached, another start state is sampled and the whole process repeats. When the time runs out, the action a_{best} with the highest value at the current history h_{real} is returned. After this action is executed in the real environment, with an observation o_{last} the tree can be pruned. Only the nodes from history $h_{real}a_{best}o_{last}$ onward stay relevant as all other histories are rendered impossible.

3.4 Exploration versus exploitation

Chapter 4

Problem formulation

4.1 Lane keeping with a human in the loop

4.2 Driver model

The driver model is simplistic. If the driver is attentive, its actions are optimal. The driver model returns the action that steers the car as close to the center of the lane as possible. In this case, the agent should not interfere. However, if a distracted driver is modeled, the driver just repeats the last action it performed while being attentive. This can have the effect of the driver's action to overshoot with the car diverging from the center of the lane. Following, the agent has to identify distracted driving and counteract.

When the driver model is initialized, it is randomly set to be attentive or distracted. The driver stays in this state for a randomly chosen discrete time period between ten and 60 seconds for an attentive state and between two and six seconds for a distracted state. After the chosen time period, the state of attentiveness switches; a previously attentive driver becomes distracted, and a previously distracted driver becomes attentive. The process repeats until the experiment is over.

4.2.1 Simple driver model

4.2.2 Steering over-correction

4.2.3 Steering over-correction and noise

4.3 Environment

4.3.1 State

The tracks will be round courses. Thus, there is no terminal state if everything goes well. If the car reaches an off-track position, however, the car is reset to be in the initial starting position again.

Name	Measurement	Description
Gear **(constant)**	$\{-1, 0, 1, \ldots, 6\}$	Distance of the car from the start line along the track line. **Neither the human driver nor the agent can directly influence this with their actions.**
RPM **(constant)**	$[0, +\inf)$	Number of engine rotations per minute. **Neither the human driver nor the agent can directly influence this with their actions.**
Speed **(constant)**	$(-\inf, +\inf)$ (km/h)	Speed along the longitudinal axis of the car. **Neither the human driver nor the agent can directly influence this with their actions.**
Side force	$(-\inf, +\inf)$ (km/h)	Speed along the transverse axis of the car. This is directly influenced by the steering actions of both human driver and agent.
Distance from start	$[0, +\inf)$ (m)	Distance of the car from the start line along the track line.
Angle	$[-\pi, +\pi]$ (rad)	Angle between car direction and track axis direction.
Lane position	$(-\inf, +\inf)$	Horizontal distance between the car and the track axis. 0 when the car is on the axis, +1 if the car is on the left edge of the track, and -1 if the car is on the right edge of the track. Greater numbers than +1 or smaller numbers than -1 indicate that the car is off-track.
Driver attention	True / False	Whether the human driver is attentive or distracted.

4.3.2 Actions

Name	Measurement	Description
In our simplified scenario, both the human driver and the agent can not accelerate, brake or switch gears.		
Steering	$[-2, +2]$	The input to the car is generated by combining the agent's action with the human's steering action (see equation 4.1). For the car, -1 means full right (159 degrees) and $+1$ means full left (21 degrees). A value greater than $+1$ or lower than -1 can effectively reverse an opposite action of the human driver.

The human driver and the agent share control of the steering wheel. The speed of the car is fixed and cannot be altered; neither by human driver nor agent. The steering input of the driver $\mathcal{A}_{steer}^{driver}$ and agent $\mathcal{A}_{steer}^{agent}$ are combined to $\mathcal{A}_{steer} \in [-1, +1]$ using equation 4.1.

The agent needs to be able to fully counteract a distracted driver's actions. In the extreme case, while the car is in a curve, a distracted driver could steer into the opposite direction of the trajectory of the lane center. Thus, the car would not only diverge from the lane center but would even get off the road completely. The agent thus needs to reverse the driver's action in order to keep the car centered in the lane and follow the road curve. Therefore, we define the range for the agent's steering action as follows: $\mathcal{A}_{steer}^{agent} \in [-2, +2]$.

$$\mathcal{A}_{\text{steer}} = \min(\,-1,\, \max(\,1,\, (\mathcal{A}_{\text{steer}}^{\text{driver}} + \mathcal{A}_{\text{steer}}^{\text{agent}})\,)\,) \tag{4.1}$$

4.3.3 Observations

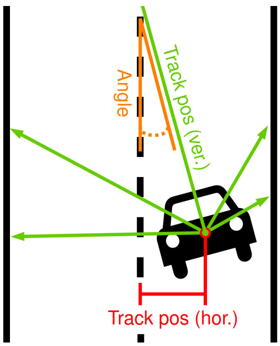


Figure 4.1: Main environment observations

Name	**Measurement**	**Description**
Constant state parameters are not observed as they do not influence learning. **The observations are not noisy.**		
Angle	$[-\pi, +\pi]$ (rad)	Angle between car direction and track axis direction
Side force	$(-\inf, +\inf)$ (km/h)	Speed along the transverse axis of the car. This is directly influenced by the steering actions of both human driver and agent.
Track position (horizontal)	$(-\inf, +\inf)$	Horizontal distance between the car and the track axis. 0 when the car is on the axis, +1 if the car is on the left edge of the track, and −1 if the car is on the right edge of the track. Greater numbers than +1 or smaller numbers than −1 indicate that the car is off-track.
Track position (vertical)	[0,200] (m)	Vector of 5 range finder sensors (of 19 available in TORCS). The range finders serve as lookahead by returning the distance between the car and the track edge in a given forward angle between −90 and +90 degrees with respect to the car axis.
Driver steering (last time step)	$[-1, +1]$	The agent perceives the last input of the human. This is not the action of the human in the next but in the last time step. The agent does not know which action the human is going to choose simultaneous to its own action. −1 means full right (159 degrees) and +1 means full left (21 degrees).

4.3.4 Reward

The overall goal for the agent is to only assist the driver in keeping the car centered in its lane. Therefore, this is the main source of reward for the agent. The more centered the car is at a certain time step, the more reward r is received. However, the agent is supposed to leave the human driver with as much autonomy as possible. Thus, any intervention by the agent is penalized. Minor smooth interventions are generally preferred over large abrupt steering actions. Accordingly, the penalty is (exponentially) dependent on the intensity of the agent's action. The general assumption is that an attentive driver performs better in keeping the car centered than an inattentive driver. The agent has to predict whether a driver is attentive or not in order to choose its actions correctly. An incorrect prediction of the driver's actions will lead to overshooting and thus be negatively reflected in the reward for keeping the car centered. Lastly, the car is never supposed to leave the lane. Consequently, leaving the lane is highly penalized.

$$\mathcal{R} = \mathcal{R}_{\text{center}} - \mathcal{P}_{\text{act intensity}} - \mathcal{P}_{\text{off-lane}}$$

$$\mathcal{R}_{\text{center}} = \begin{cases} r - r * |Pos_{hor}| & \text{if} \quad |Pos_{hor}| \leq 1 \\ 0 & \text{if} \quad \text{off-lane} \end{cases}$$

$$\mathcal{P}_{\text{act intensity}} = |\mathcal{A}_{\text{steer}}|^{p_{\text{int}}}$$

$$\mathcal{P}_{\text{off-lane}} = \begin{cases} p_{\text{off}} & \text{if} \quad |Pos_{hor}| > 1 \\ 0 & \text{if} \quad |Pos_{hor}| \leq 1 \end{cases}$$

Chapter 5

Experimental setup

The task for the agent in the experiment is to keep the car centered in a highway lane. Thus, the track used for the agent's evaluation needs to represent such a scenario. Most tracks readily available in the racing car simulator TORCS are race tracks. These are much wider than common roads and the width often differs in different segments. To ensure a realistic scenario, a one-lane track with a continuous width of 3.5m, which is common for European roads, is used. The track covers a wide array of scenarios. It includes long straight segments, both left and right curves, and multiple curves of alternating directions in a row. By ensuring that all common highway scenarios are covered by the track, a single track is sufficient.

The car used for the experiment does not have a big impact on driving performance, as long as it is consistent during all experiments. To ensure that an action's effect at a particular position are consistent, the speed of the car is constant.

The driver is pulled for an action every 0.1 seconds. The simulation tick rate is 0.002 seconds. When the driver is not pulled, its last action is repeated. It follows, that every action is repeated during 50 simulation ticks. The simulation is not in real time. Therefore, the simulation waits for the agent's planning. If the agent is pulled for an action, the environment does not change until the agent's next action is decided and performed.

5.1 Evaluated scenarios

5.1.1 Driver model

5.1.2 Action space and action selection

5.2 Design decisions

5.2.1 Discretization

5.2.2 Hyperparameter optimization

There are a number of hyperparameters. Most importantly, there is the planning time. This is the time the agent is allowed to search in and expand its search tree in order to find the most likely current state and best course of action. More planning time can result in a wider and/or deeper tree. The search horizon is limited by a discount threshold. If this threshold is reached, the search is stopped and no more actions will be performed for the current trajectory and, if there is enough planning time left, a new state is sampled from the current belief and a new planning trajectory is expanded. Moreover, there is an exploration constant. This value, determined before the start of the experiment, assigns actions that have not been tried before more expected reward and thus favors exploration.

5.2.2.1 Number of searches

5.2.2.2 Exploration constant

5.2.2.3 Discount horizon

5.3 Performance metrics

Due to the randomness involved in the driver model, each experiment run will lead to a different scenario. Therefore, to get credible results, the experiment has to be repeated many times. The average discounted return over all experiment runs serves as performance metric. This result is compared with the average reward of an agent that always performs the optimal reaction to the action of the driver model. The closer the POMCP agent's reward is to the optimal agent's reward, the better was the planning.

Chapter 6

Results

6.1 Lower and upper performance bound

6.2 Grid search for hyperparameters

6.2.1 All actions

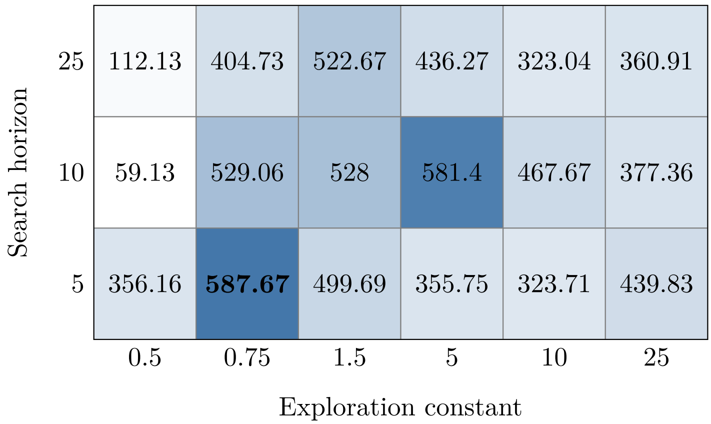


Figure 6.1: Grid search for the configuration leading to the highest average cumulative reward after 20 runs with up to 1000 actions each, using the All actions.

6.2.2 Bounded actions

6.2.3 Preferred actions

6.3 Convergence behavior

6.3.1 Simple driver model

6.3.2 Steering over-correction

6.3.3 Steering over-correction and noise

6.4 Comparison of driving trajectories

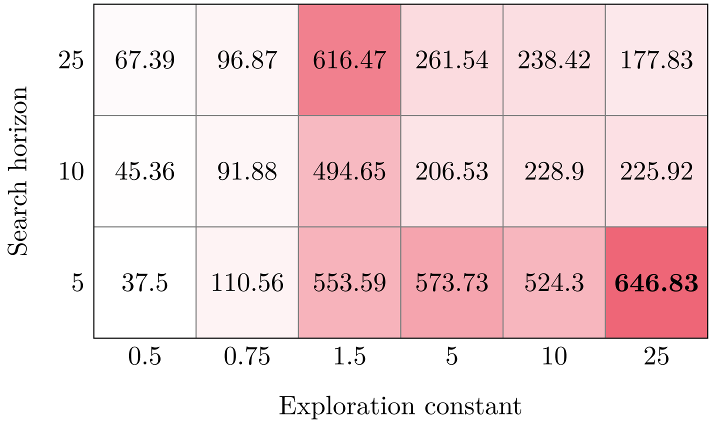


Figure 6.2: Grid search for the configuration leading to the highest average cumulative reward after 20 runs with up to 1000 actions each, using a Bounded actions.

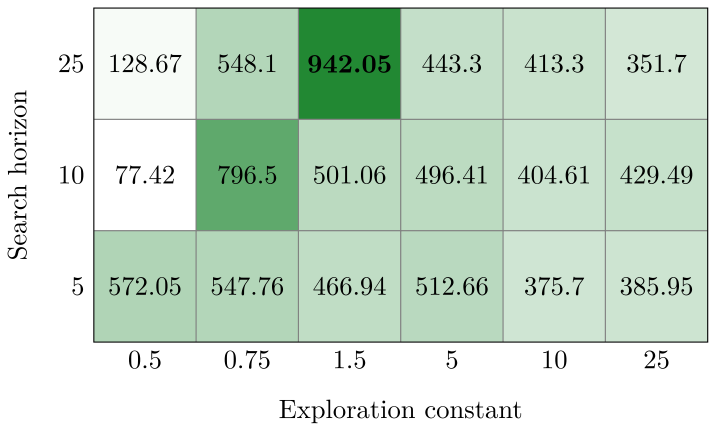


Figure 6.3: Grid search for the configuration leading to the highest average cumulative reward after 20 runs with up to 1000 actions each, using preferred actions.

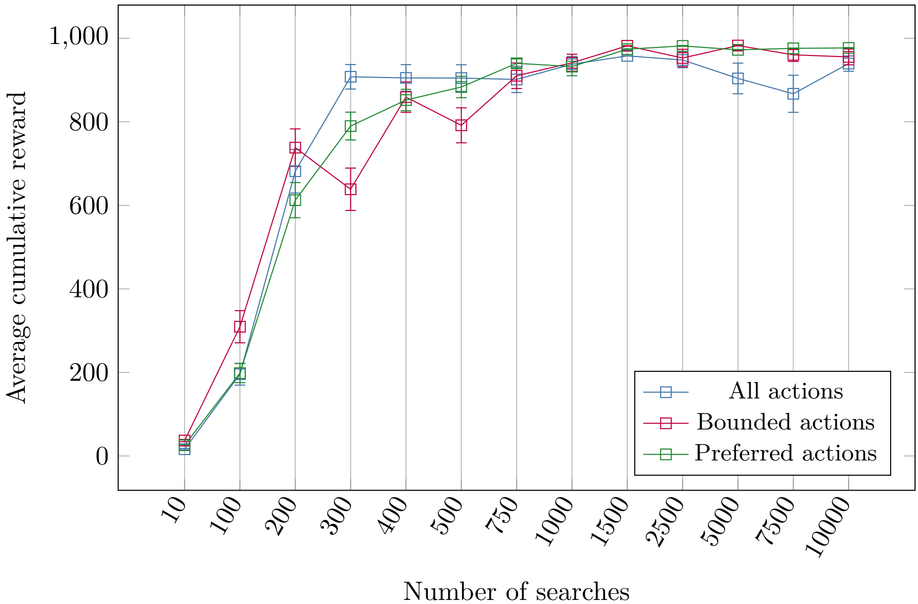


Figure 6.4: Performance comparison of POMCP when utilizing the All actions, a Bounded actions, or preferred actions with a simple driver model. Each point shows the mean cumulative reward from 50 runs with 1000 actions each, if no terminal state is reached earlier.

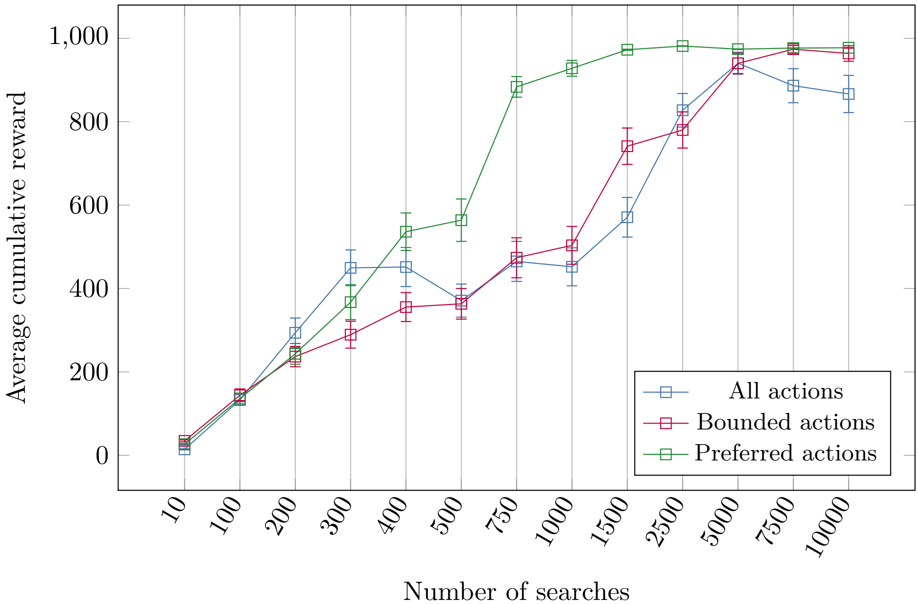


Figure 6.5: Performance comparison of POMCP when utilizing the All actions, a Bounded actions, or preferred actions with a driver model that over-corrects when it regains attention. Each point shows the mean cumulative reward from 50 runs with 1000 actions each, if no terminal state is reached earlier.

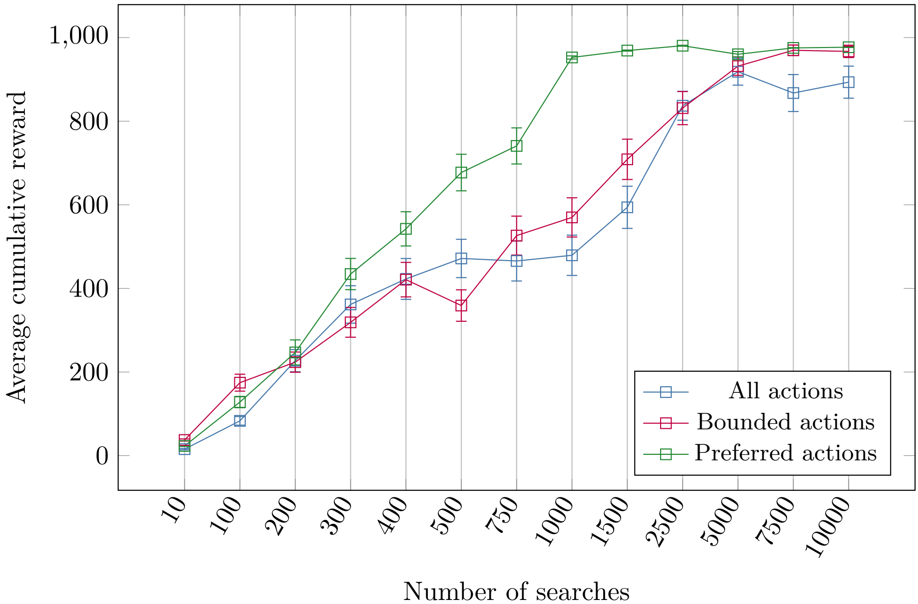


Figure 6.6: Performance comparison of POMCP when utilizing the All actions, a Bounded actions, or preferred actions with a driver model that over-corrects when it regains attention and performs noisy actions. Each point shows the mean cumulative reward from 50 runs with 1000 actions each, if no terminal state is reached earlier.

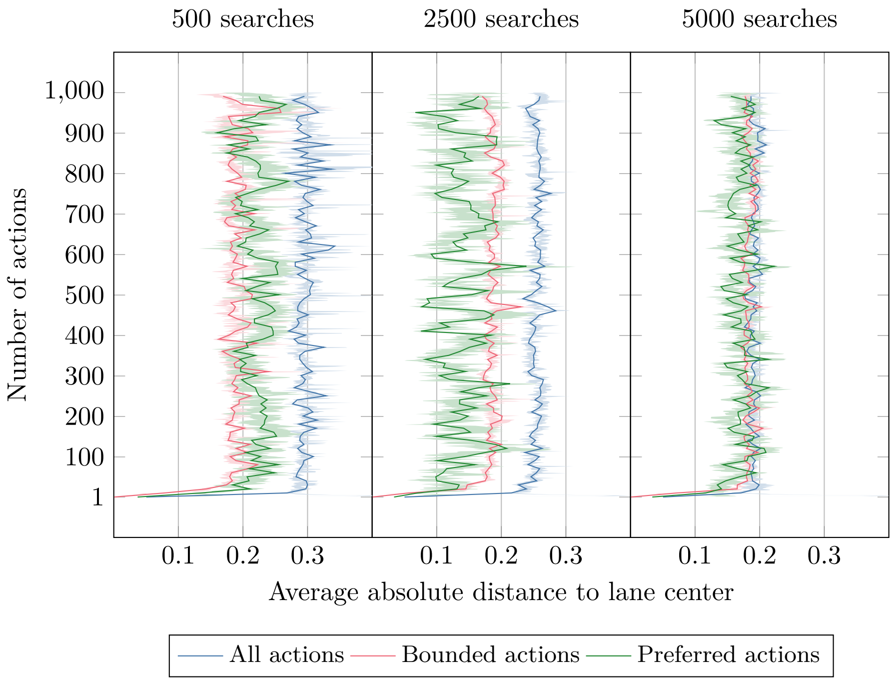


Figure 6.7: Mean lane centeredness. 5000 Over

Chapter 7

Discussion

7.1 Analysis of the results

7.2 Limitations

7.2.1 Long planning time

7.2.2 Action and observation space discretization

7.2.3 Dependency on reliable driver and environment models

7.2.4 Edge cases

Chapter 8

Conclusion and future outlook

8.1 Conclusion

8.2 Road toward application with human drivers

8.2.1 Performance optimization

8.2.2 Integrating realistic driver and environment models

8.2.3 Continuous action and observation space

8.2.4 Using other POMDP solvers

Appendices

Bibliography

Espié, E., Guionneau, C., Wymann, B., Dimitrakakis, C., Coulom, R., & Sumner, A. (2005). Torcs, the open racing car simulator.

Gelly, S., & Silver, D. (2007). Combining online and offline knowledge in uct, Corvalis, Oregon, USA, Association for Computing Machinery. https://doi.org/10.1145/1273496.1273531

He, R., Brunskill, E., & Roy, N. (2011). Efficient planning under uncertainty with macro-actions. *Journal of Artificial Intelligence Research*, vol. 40, 523–570. https://doi.org/10.1613/jair.3171

Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, vol. 101no. 1, 99–134. https://doi.org/https://doi.org/10.1016/S0004-3702(98)00023-X

Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Sallab, A. A. A., Yogamani, S., & Pérez, P. (2021). Deep reinforcement learning for autonomous driving: A survey.

Lam, C., & Sastry, S. S. (2014). A POMDP framework for human-in-the-loop system, In *53rd ieee conference on decision and control*. https://doi.org/10.1109/CDC.2014.7040333

Maurer, M., Gerdes, J., Lenz, B., & Winner, H. (2016). *Autonomous driving. technical, legal and social aspects*. https://doi.org/10.1007/978-3-662-48847-8

NHTSA. (2020). Traffic Safety Facts: Distracted Driving 2018 [Retrieved from https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812926].

Ross, S., Pineau, J., Paquet, S., & Chaib-draa, B. (2008). Online planning algorithms for pomdps. *Journal of Artificial Intelligence Research*, vol. 32, 663–704. https://doi.org/10.1613/jair.2567

Sadigh, D., Sastry, S. S., Seshia, S. A., & Dragan, A. (2016). Information gathering actions over human internal state, In *2016 ieee/rsj international conference on intelligent robots and systems (iros)*. https://doi.org/10.1109/IROS.2016.7759036

Silver, D., & Veness, J. (2010). Monte-carlo planning in large pomdps (J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, & A. Culotta, Eds.). In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, & A. Culotta (Eds.), *Advances in neural information processing systems*, Curran Associates, Inc. https://proceedings.neurips.cc/paper/2010/file/edfbe1afcf9246bb0d40eb4d8027d90f-Paper.pdf

Smallwood, R. D., & Sondik, E. J. (1973). The optimal control of partially observable markov processes over a finite horizon. *Operations Research*, vol. 21no. 5, 1071–1088. https://doi.org/10.1287/opre.21.5.1071

Sunberg, Z., & Kochenderfer, M. (2018). Online algorithms for pomdps with continuous state, action, and observation spaces.

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

Wang, W., Na, X., Cao, D., Gong, J., Xi, J., Xing, Y., & Wang, F. .-Y. (2020). Decision-making in driver-automation shared control: A review and perspectives. *IEEE/CAA Journal of Automatica Sinica*, vol. 7no. 5, 1289–1307. https://doi.org/10.1109/JAS.2020.1003294

Ye, N., Somani, A., Hsu, D., & Lee, W. S. (2017). Despot: Online pomdp planning with regularization. *Journal of Artificial Intelligence Research*, vol. 58, 231–266. https://doi.org/10.1613/jair.5328