TECHNISCHE
UNIVERSITÄT
DARMSTADT

A HYBRID APPROACH TO AUTOMATED DRIVING UNIFYING PREDICTION AND PLANNING

A dissertation submitted to
TECHNISCHE UNIVERSITÄT DARMSTADT
Fachbereich Informatik

in fulfillment of the requirements for the degree of
Doktor-Ingenieur (Dr.-Ing.)

presented by

SASCHA ROSBACH
M.Sc.

Examiner:	Prof. Stefan Roth, Ph.D.
Co-examiner:	Prof. Henryk Michalewski, Ph.D.
Date of Submission:	October 30th, 2023
Date of Defense:	March 22nd, 2024

Darmstadt, 2024

A Hybrid Approach to Automated Driving
Unifying Prediction and Planning

Submitted doctoral thesis by Sascha Rosbach

Examiner: Prof. Stefan Roth, Ph.D.
Co-examiner: Prof. Henryk Michalewski, Ph.D.

Date of Submission: October 30th, 2023
Date of Defense: March 22nd, 2024

Darmstadt, Technische Universität Darmstadt
Jahr der Veröffentlichung der Dissertation auf TUprints: 2024

Bitte zitieren Sie dieses Dokument als:
URN: urn:nbn:de:tuda-tuprints-288413
URL: https://tuprints.ulb.tu-darmstadt.de/28841

Dieses Dokument wird bereitgestellt von tuprints,
E-Publishing-Service der TU Darmstadt
http://tuprints.ulb.tu-darmstadt.de
tuprints@ulb.tu-darmstadt.de

To my parents.

ABSTRACT

Fully automated driving is nearing a stage of large-scale deployment, where the vehicles will interact with many traffic participants within urban traffic environments. The success of the deployments requires reliable decision-making that generalizes over a variety of situations. The conventional modular architecture, encompassing perception, prediction, planning, and control, has been pivotal for fully automated driving, allowing large teams to work simultaneously on the architecture. However, the generalization remains a challenge. This thesis proposes a hybrid approach to automated driving. It draws upon the interpretability intrinsic to traditional modular design and combines this with the generalization capabilities of deep learning.

The first part of this thesis examines the real-world applicability of the direct perception paradigm. The paradigm directly ties perception to control, striving to streamline modular architectures by focusing on essential features to implement the desired driving behaviors rather than explicitly modeling and evaluating the complete environment. The approach employs multi-task learning to predict affordances for driving that directly supply the inputs for lateral and longitudinal controllers. However, the system's operational domain is confined due to rule-based behavior planning, making it infeasible to address unexpected situations. To overcome these limitations, the subsequent work in this thesis builds upon the modular architecture by integrating deep learning. An environmental model and model predictive planner are utilized, leveraging high-resolution action sampling to generate a diverse set of driving policies. These policies have implicit behaviors ranging from lane-changing and emergency braking to merging into time gaps between vehicles, eliminating the need for explicit hierarchical behavior modeling.

The second part of this thesis is concerned with bringing the modular architecture into an offline training loop and aligning the behavior of the model predictive planner with the preferences of human drivers. The first proposed method automates the tedious reward function tuning process that domain experts usually perform manually. The sampled policies of the planner enable maximum entropy inverse reinforcement learning to be tractable within high-dimensional continuous action spaces, utilizing path integral features. The succeeding method uses deep learning to predict situation-dependent reward functions, enabling generalization across diverse driving situations. The network inputs all sampled driving policies to combine environment and vehicle dynamics features and predicts situation-dependent weights of the reward function. Later work proposes policy and temporal attention

mechanisms for the network designed to produce consistent driving behaviors while adapting the reward function for consecutive planning cycles.

The third part of this thesis again focuses on streamlining the modular architecture after tackling the problem of reward function generation. The proposed approach is designed to leverage deep learning-based situation understanding. It focuses on making the explicit future motion prediction of surrounding objects optional. This is achieved by learning from an exhaustively sampling model predictive planner driving in real-world situations. The method unifies prediction and planning by predicting pixel state value sequences of the planning algorithm that implicitly encode driving comfort, reachability, safety, and object interaction.

This thesis provides an important step towards the scalability of automated driving by learning what is difficult to model by hand while preserving interpretability and the interfaces to incorporate explicit reasoning. This hybrid approach allows joint optimization of prediction and planning essential to implement humanlike, assertive, and safe driving in interactive driving environments.

ZUSAMMENFASSUNG

Das vollautomatisierte Fahren nähert sich einer Phase des großflächigen Einsatzes, in der die Fahrzeuge vor die Herausforderung gestellt werden, mit vielen Verkehrsteilnehmern im Stadtverkehr zu interagieren. Eine erfolgreiche Einführung des automatischen Fahrens erfordert ein zuverlässiges Entscheidungsfindungsmodul, das eine Vielzahl von Situationen unterstützt. Die konventionelle modulare Architektur teilt die Wirkkette in Wahrnehmung, Vorhersage, Planung und Regelung. Diese Architektur spielte bisher eine entscheidende Rolle für das automatische Fahren und ermöglichte es mit großen Teams parallel daran zu arbeiten. Die Bewältigung jeglicher Verkehrssituationen und die damit verbundenen Generalisierungsfähigkeiten des Systems bleiben eine Herausforderung. Diese Forschungsarbeit schlägt einen hybriden Ansatz für das automatisierte Fahren vor. Dieser Ansatz vereint die Interpretierbarkeit traditioneller modularer Architekturen für das automatisierte Fahren, und kombiniert diese mit der Generalisierungsfähigkeit tiefer neuronaler Netze.

Im ersten Teil dieser Dissertation wird die Anwendbarkeit des direkten Wahrnehmungsparadigmas in der Praxis untersucht. Dieses Paradigma verbindet die Wahrnehmung direkt mit der Regelung, mit dem Ziel, modulare Architekturen zu simplifizieren. Statt die gesamte Umgebung explizit zu modellieren und zu bewerten, konzentriert sich der Ansatz auf wesentliche Merkmale, die das gewünschte Fahrverhalten beeinflussen. Um Merkmale für das Fahren vorherzusagen, werden Multitask-Lernverfahren genutzt, die direkte Eingaben für Quer- und Längsregler liefern können. Der Einsatzbereich des Systems ist jedoch aufgrund der einfachen regelbasierten Verhaltensplanung begrenzt, was es diesem Ansatz schwer macht, unerwartete Situationen zu addressieren. Aus diesem Grund erweitert und integriert der nachfolgende Teil dieser Dissertation tiefe neuronale Netze in die modulare Architektur. Die hierzu verwendete modulare Architektur nutzt ein Umgebungsmodell und einen modelprädiktiven Planer, welcher durch hochauflösendes Abtasten von Aktionen eine Vielzahl von Fahrstrategien generieren kann. Diese Strategien haben implizite Verhaltensweisen, die von Spurwechseln und Notbremsungen bis hin zum Einfädeln in Zeitlücken zwischen Fahrzeugen reichen. Durch die implizite Verhaltensgenerierung entfällt die Notwendigkeit der expliziten hierarchischen Verhaltensmodellierung.

Der zweite Teil dieser Thesis beschäftigt sich damit, die modulare Architektur in ein Offline-Trainingsverfahren zu integrieren und das Verhalten des modellprädiktiven Planers auf die Präferenzen menschlicher Fahrer abzustimmen. Die erste vorgeschlagene Methode automati-

siert den langwierigen Abstimmungsprozess der Belohnungsfunktion, der in der Regel manuell von Experten durchgeführt werden muss. Die generierten Fahrstrategien des modellprädiktiven Planers ermöglichen es „Maximum Entropy Inverse Reinforcement Learning“ unter der Verwendung von Pfadintegralenmerkmalen in hochdimensionalen kontinuierlichen Aktionsräumen anzuwenden. Die darauffolgende Methode verwendet ein tiefes Lernverfahren, um situationsabhängige Belohnungsfunktionen vorherzusagen, wodurch eine Generalisierung über eine Vielzahl von Fahrsituationen hinweg ermöglicht wird. Die generierten Fahrstrategien dienen bei dieser Methode als Eingabeinformationen für das Netzwerk, um Umgebungs- und Fahrdynamikmerkmale zu kombinieren und mit Hilfe dieser situationsabhängige Gewichte der Belohnungsfunktion vorherzusagen. In einer Erweiterung werden strategische und zeitliche Aufmerksamkeitsmechanismen für das Netzwerk vorgeschlagen, um ein konsistentes Fahrverhalten zu erzeugen, während die Belohnungsfunktion für aufeinanderfolgende Planungszyklen stetig angepasst wird.

Der dritte Teil dieser Dissertation konzentriert sich erneut auf die Optimierung und Vereinfachung der modularen Architektur, nachdem das Problem der Generierung der Belohnungsfunktion behandelt wurde. Der vorgeschlagene Ansatz ist darauf ausgelegt, das Situationsverständnis mit Hilfe eines tiefen neuronalen Netzes zu erlernen. Dabei konzentriert sich der Ansatz darauf, die expliziten Bewegungsvorhersagen von umgebenden Objekten zu optionalisieren. Dies wird erreicht, indem von einer annähernd vollständigen Suche eines modellprädiktiven Planers gelernt wird, der in realen Situationen fährt. Die Methode vereint Vorhersage und Planung durch das Vorhersagen von Pixelzuständen des Planungsalgorithmus, die implizit Fahrkomfort, Erreichbarkeit, Sicherheit und Objektinteraktion encodieren.

Diese Dissertation schlägt eine bedeutende Richtung auf dem Weg zu einer skalierbaren Fahrfunktion ein. Die vorgeschlagenen Ansätze lernen die Inhalte, die nur schwer von Hand zu modellieren sind, und gleichzeitig wird die Interpretierbarkeit und Anwendbarkeit von expliziter Logik aufrechterhalten. Dieser hybride Ansatz ermöglicht eine gemeinsame Optimierung von Vorhersage und Planung, eine entscheidende Kombination, um menschenähnliches, durchsetzungsfähiges und sicheres Fahren in interaktiven Verkehrssituationen umzusetzen.

PUBLICATIONS

Rosbach, **Sascha**, Antonia Breuer, Simon Barthel, Frederik Kanning, and Silviu Homoceanu (2019a). "Towards hybrid automated driving: From direct imitation learning to affordance learning." In: *AAET-Automatisiertes und Vernetztes Fahren: Beiträge zum gleichnamigen Symposium*, pp. 225–245.

Rosbach, **Sascha**, Vinit James, Simon Großjohann, Silviu Homoceanu, and Stefan Roth (2019b). "Driving with style: Inverse reinforcement learning in general-purpose planning for automated driving." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2658–2665.

Rosbach, **Sascha**, Vinit James, Simon Großjohann, Silviu Homoceanu, Xing Li, and Stefan Roth (2020a). "Driving style encoder: Situational reward adaptation for general-purpose planning in automated driving." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6419–6425.

Rosbach, **Sascha**, Xing Li, Simon Großjohann, Silviu Homoceanu, and Stefan Roth (2020b). "Planning on the fast lane: Learning to interact using attention mechanisms in path integral inverse reinforcement learning." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5187–5193.

Großjohann, Simon, Silviu Homoceanu, **Sascha Rosbach**, and Vinit James (2021). "Verfahren und Vorrichtung zum Bereitstellen einer Fahrstrategie für das automatisierte Fahren eines Fahrzeugs." Pat. 102019216232A1. Volkswagen AG. Germany.

Rosbach, **Sascha**, Stefan Leupold, Simon Großjohann, and Stefan Roth (2023). "Pixel state value network for combined prediction and planning in interactive environments." In: *arXiv Preprint*. arXiv:2310.07706 [cs.RO].

Rosbach, **Sascha** and Simon Großjohann (2024). "Verfahren zum automatisierten Führen eines Fahrzeugs sowie Verfahren zum Erzeugen eines hierzu fähigen Modells des Maschinellen Lernens sowie Prozessorschaltung und Fahrzeug." Pat. DE102022131178B3. CARIAD SE. Germany.

ACKNOWLEDGMENTS

First and foremost, I wish to thank my supervisor and mentor, Silviu Homoceanu. He taught me the principles behind academic writing, thinking, and conducting research. I owe my deepest gratitude to my mentor and collaborator Simon Großjohann. He advanced my research by continuously challenging the applicability of the approaches in practice and debating which aspects should be learned or modeled. I want to thank Stefan Roth for his valuable feedback and guidance during my research. I was fortunate for the opportunity to guide Vinit James and Xing Li as my students. Their hard work, feedback, and support enabled valuable experiments. I would like to thank my colleagues in the automated driving department at Volkswagen Group Research, who developed many tools that I was able to use during my research. I am grateful to my colleague Stefan Leupold for his collaboration that enriched this work's depth and quality, particularly by working with me on the real-world application of these approaches. Last but not least, I want to express my appreciation to Natalia, Blair, and my family for their unwavering support throughout this remarkable journey.

DISCLAIMER

The results, opinions and conclusions expressed in this thesis are not necessarily those of Volkswagen AG and Cariad SE.

CONTENTS

LIST OF FIGURES

ACRONYMS

AV	Automated Vehicle
BEV	Bird's Eye View
CNN	Convolutional Neural Network
DARPA	Defense Advanced Research Projects Agency
DNN	Deep Neural Network
ED	Expected Distance
EM	Expectation Maximization
EVD	Expected Value Difference
GAIL	Generative Adversarial Imitation Learning
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
HD	High Definition
IRL	Inverse Reinforcement Learning
LEARCH	Learning to Search
LLD	Log-Likelihood of Demonstration
MDP	Markov Decision Process
MMP	Maximum Margin Planning
MPC	Model Predictive Control
MPP	Model Predictive Planner
OPD	Optimal Policy Distance
OTG	Object Time Gap
PSV	Pixel State Value
PSVN	Pixel State Value Network
RL	Reinforcement Learning
RNN	Recurrent Neural Network
SV	Surrounding Vehicle

1

INTRODUCTION

The mass adoption of automated driving can greatly benefit our society (Kim, 2018). It paves the way for automated on-demand transportation, which can reduce congestion in urban areas, save time, and bring convenience to everyday life. A primary concern on the path toward this realization is the scalability of automated driving systems. Such systems must perceive the environment, understand where objects are moving, and produce safe driving commands, tasks especially demanding given the narrow margin for error when navigating close to other road users. Ultimately, the system has to be capable of driving in unforeseen situations and different geographical locations. A critical open question is whether general-purpose approaches can be developed that do not rely on situation-dependent implementations and produce consistent and humanlike decisions in all driving domains.

Surveys categorize approaches on a broader scope into modular and end-to-end approaches (Schwarting et al., 2018; Yurtsever et al., 2020). Approaches of the first category traditionally use manually engineered predictive models of the environment, vehicle dynamics, and explicit behavior models to plan trajectories. Modeling all intended aspects enables the system to specialize in executing controlled behaviors such as maintaining precise follow-distances to preceding vehicles and adhering to rules such as avoiding driving in oncoming traffic. However, it is challenging to model a sufficiently complete and accurate predictive model of the environment and align the objectives for planning algorithms to reflect humanlike driving. It necessitates large teams working on various subproblems separately, potentially leading to suboptimal solutions due to the lack of unity (Zeng et al., 2019). A fundamental premise of this thesis is to explore whether modeling all these aspects is essential and to investigate the potential for simplification in system design.

In contrast, the second category uses deep learning (LeCun et al., 2015) to train models end-to-end from sensor inputs to driving commands (Tampuu et al., 2020). Two well-known approaches for automated driving are imitation learning and reinforcement learning. The goal of direct imitation learning is to learn by observing the steering angle and acceleration commands of human drivers given large amounts of real-world driving recordings, while the scope of

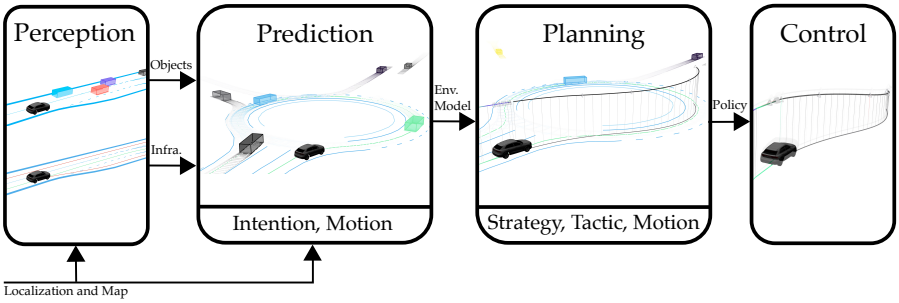


Figure 1.1. Traditional modular system architecture. The perception module recognizes objects and infrastructure with the support of a localization system and reference map. The prediction module forecasts the future motion of objects employing intention prediction, providing the planning module with an environment model. The planner makes decisions on a strategic, tactical, and vehicle motion level using a reward function. The planner generates a policy in the form of a trajectory for vehicle control.

reinforcement learning is to learn how to drive by interacting with the environment. Both methodologies benefit from directly generating driving commands using a learned context understanding that does not require situation-dependent behavioral implementations. Yet, end-to-end approaches lack the interpretability, reliability, and constraints of models, and, as such, can potentially violate vehicle dynamics and safety in the context of automated driving. This thesis aims to combine the benefits and eliminate the downsides of modular and end-to-end approaches by proposing a hybrid approach. A particular focus is on using the modular architecture to leverage prior knowledge and interpretability while using human driving demonstrations and simulated experience to acquire a context understanding of situations.

Before delving into the details of the proposed hybrid approach, it is important to introduce the conventional modular architecture, which serves as a foundation for understanding the subsequent contributions. The modular architecture, as shown in Figure 1.1, separates the whole driving stack into perception, prediction, planning, and control modules (Badue et al., 2021; Yurtsever et al., 2020). The architecture creates abstraction from raw sensory data and uses manually designed interfaces between modules to allow the architecture to leverage prior knowledge. The perception system perceives the environment through sensors and generates a representation of infrastructure and objects. State-of-the-art perception systems utilize deep learning to fuse sensor data from multiple views to represent a bird's eye view (BEV) of the environment (Ma et al., 2022). High-definition (HD) maps are often utilized with localization systems to increase the reliability of the perceived infrastructure and provide semantic information such as traffic rules. The prediction module often uses deep learning to encode information about the infrastructure and object history to forecast the future motion of surrounding vehicles (SVs). The planning module

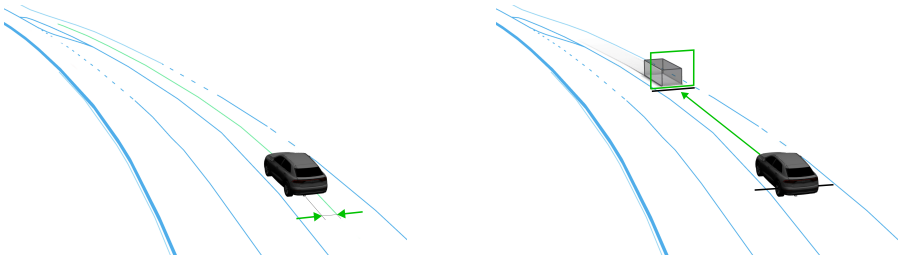


Figure 1.2. Affordances of the direct perception paradigm. The figure shows lateral affordances *(left)* and longitudinal affordances *(right)*, both in green. The road boundaries are illustrated in light blue to help visualize the situation. *Left*: The curvature and the ego offset to the lane center serve as lateral affordances. *Right*: Classifying whether a vehicle is in the same lane and estimating the distance serve as longitudinal affordances.

uses a predictive model of the environment to find a safe and comfortable trajectory by making long-term strategic decisions about the route, mid-term tactical decisions about the behavior, and low-level decisions about the vehicle motion. Domain experts usually manually define the reward function and specify the desired driving style. The planned trajectory is passed to the control module, which generates steering, acceleration, and deceleration commands to achieve the desired behavior, even under various conditions and disturbances.

State-of-the-art fully automated vehicles use modular architectures to drive in urban environments, requiring large teams to work on the modules simultaneously. Despite the complexity, advanced driver assistance systems use this architecture in constrained operational domains, such as highways. The initial approach of this thesis sets the stage for a series of methodologies that will be presented in Chapter 3. This first approach aims to simplify the modular architecture for driver assistance systems by removing the necessity of explicitly modeling the complete environment and focusing on essential features to implement the desired driving behaviors. The first approach, originally proposed by Chen et al. (2015), is evaluated in this thesis for its real-world applicability and potential to serve as the foundation for developing further approaches. The approach uses two deep neural networks (DNNs) that directly predict affordances using end-to-end perception. A simple rule-based behavior model provides interpretability by instructing the model-based controllers to implement these, unlike end-to-end approaches, which do so implicitly. The approach separates the predictions in affordances for lane assistance and adaptive cruise control, as depicted in Figure 1.2. For lane assistance, a DNN predicts the curvature of the road and the divergence from the lane center to control the vehicle back to the center. For cruise control, the DNN classifies if there is a vehicle to follow and, if there is, predicts the distance for longitudinal control. The approach can

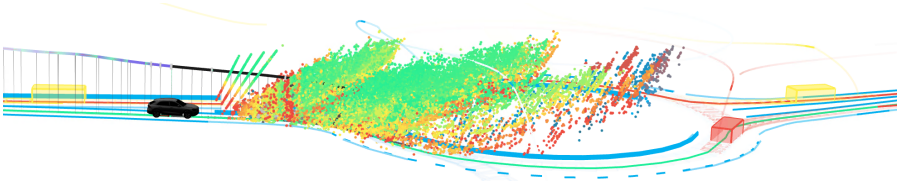


Figure 1.3. Example of a sampling-based planning algorithm approaching a roundabout. The figure depicts the explored state space of the planner and motion predictions for the surrounding vehicles. The states are sampled over a planning horizon of 6.6 seconds, with the color coding indicating the cumulative reward for the states.

effectively focus on the essential elements of driver assistants, such as adhering to a predefined reference path, defined by the lane center, under challenging and diverse road conditions. However, defining a priori and selecting behaviors with a rule-based system for fully automated driving in urban environments is infeasible. Therefore, the other approaches of this thesis leverage behavior and motion planning algorithms that can address unexpected driving situations.

Planning algorithms in automated driving broadly fall into two main categories: variational and search-based methods (Paden et al., 2016). Algorithms of the first category optimize an initial trajectory until a cost function convergences to a local minimum (Gundlach, 2020). These methods excel in well-structured environments like highways and race tracks, where one can derive the behavior of the initial solution without extensively considering moving objects in relation to the ego vehicle dynamics. Designing the features of the cost function in complex environments poses significant challenges and necessitates numerical approximation as these environments are formulated in state space (Kuderer et al., 2015; LaValle, 2006). The algorithms in the second category simulate vehicle motion states in the environment to expand a search graph, aiming for an approximate global optimum directly. However, the need to discretize the search space for time-continuous search leads to discretization errors. As the search space expands exponentially over long planning horizons, it becomes a computationally expensive and challenging task. McNaughton (2011) proposed an approximately exhaustive search algorithm based on dynamic programming, which allows for real-time global search when parallelized on a graphics processing unit (GPU).

This thesis uses a sampling-based planning algorithm that samples densely feasible actions while expanding a search graph. Over the last decade, the computing power of GPU has been steadily increasing, enabling the planner to simulate millions of motion states over planning horizons of ten seconds and more. This results in thousands of feasible trajectories covering the planning horizon in each situation, as depicted in Figure 1.3. A reward function, with a linear combination of weights, assigns the importance among features of motion, infras-

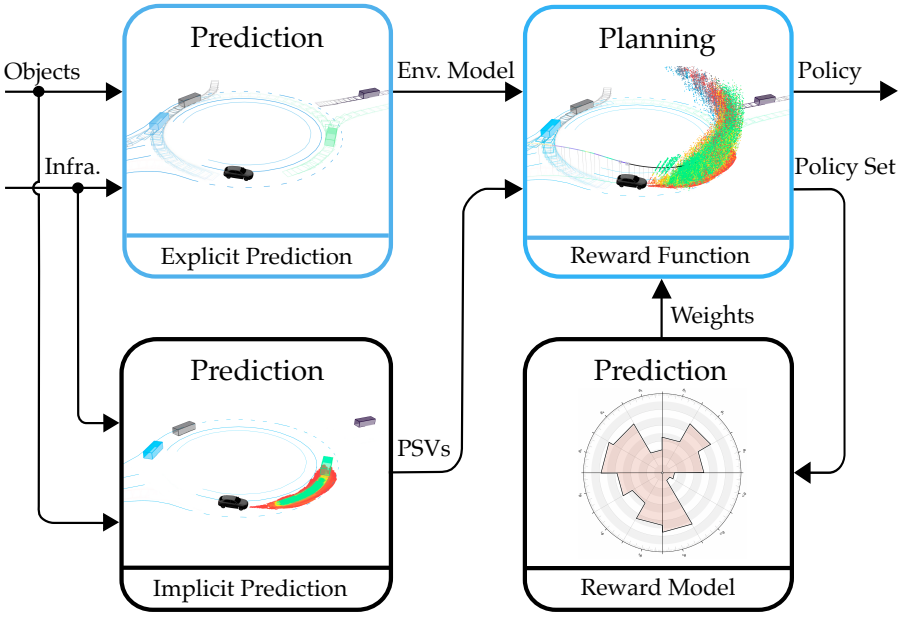


Figure 1.4. Unified architecture of the hybrid approach. The figure gives an overview of the prediction and planning building blocks and their interfaces. The blocks outlined in blue resemble the modular architecture. The implicit prediction block provides pixel state values (PSVs) for the planner. The reward model predicts weights, adapting the importance among features in the reward function.

tructure, and objects to determine which of the trajectories is optimal. However, this requires tedious manual tuning to produce the desired behavior.

The second approach in this thesis focuses on transferring the tuning task to machine learning, a particular type called inverse reinforcement learning (IRL), which evolved around finding reward functions by observing demonstrations of the desired behavior (Ng and Russell, 2000). The IRL approach uses an energy-based model to describe the relationship between the demonstrated behavior and reward, implying the behavior of an experienced driver becomes exponentially more likely with increasing reward. This methodology has been pioneered by Ziebart et al. (2008) and is called maximum entropy IRL. Maximum entropy IRL is often considered intractable over long planning horizons because the distribution requires the calculation of all possible trajectories (Arora and Doshi, 2021; Ziebart et al., 2008). The proposed approach that is part of this work makes the imitation learning methodology tractable for automated driving by using the above mentioned sampling-based planning algorithm that exhaustively searches for feasible trajectories (Rosbach et al., 2019). This combination allows developing novel deep learning approaches, which can, on the one hand, use the planning system's high-dimensional features to improve situational context understanding and, on the other hand, use its rules and heuristics to constrain the final trajectory selection.

Figure 1.4 guides the reader, showing the building blocks of the hybrid architecture, along with their respective inputs and outputs. The prediction and planning blocks of the previously discussed modular architecture are depicted with a blue outline. The next approach presented in this thesis uses the IRL methodology to build a reward model. The approach is concerned with the limited generalization of a single reward function and proposes an approach to perform situation-dependent predictions (Rosbach et al., 2020a). To allow a DNN to understand the driving situation, the method uses the planner's generated policies as input from previous planning cycles. Each policy can be associated with its static and kinematic features of the environment and the sequence of actions that caused it. All sampled policies have different behaviors and concatenated provide a picture of feasible motion in each situation, similar to a detailed textual description composed of sentences using various arguments. Thus, the role of the DNN is to automatically find patterns from the policies to allow a suitable reward function prediction.

The subsequent approach, also a reward model, introduces adaptation mechanisms that control the dynamics of the reward function prediction to provide persistent behavior, such as interactions with other moving vehicles (Rosbach et al., 2020b). Sudden input changes directly trigger transitions in reward functions, an undesirable property in smooth lane following and during lane changes, where it is crucial to consider the overall driving context to ensure persistent goal-directed behaviors. For this reason, this work introduces attention mechanisms for the DNN architecture (Rosbach et al., 2020b). On the one hand, this generates focus on policies among the set having an expert-like behavior similar to eye tracking. On the other hand, it introduces a temporal attention mechanism, which allows for extracting the driving context from a sequence of planning cycles and regulates the adaptation of the reward function.

The last approach in this thesis and pending prediction block of the hybrid architecture in Figure 1.4 makes the explicit future motion prediction optional. The DNN approach implicitly provides the future motion of objects by training the prediction block using the planning algorithm and reward model, thereby unifying prediction and planning (Rosbach et al., 2023). This thesis argues that separating prediction and planning into two distinct modules dramatically increases the system complexity due to the modules' interdependency in interactive driving scenarios, where the planned behavior of the automated vehicle influences the behavior of SVs. This approach aims to discriminate states using a DNN, providing a tight integration with the planning algorithm, going beyond the prediction of situation-dependent linear reward function weights that act on manually defined path integral features to discriminate policies. The DNN implicitly predicts information about driving comfort, kinematic reachability, object interaction,

Figure 1.5. Example of the pixel state value prediction. The vehicle approaches a roundabout while a faster driving vehicle passes the entry. The stacked image sequence depicts the rendered PSVs in 1.1 second time increments over a planning horizon of 6.6 seconds. The green pixels indicate high values, yellow pixels present medium values, and red pixels denote low values. The white pixels are either unreachable or colliding with SVs. Each image has a resolution of one-megapixel.

and safety while observing the whole scene in BEV. The training task is cast as paired image-to-image translation using a conditional generative adversarial network (GAN). The targets encompass two image sequences; one displaying explicit visitations of other SVs, and the other depicting PSVs rendered through an energy-based model derived from the sampled policies. The planner can directly utilize the PSVs in a reward shaping function, which implicitly encodes interaction with the environment. Figure 1.5 shows the PSV target sequence rendered by the planning algorithm that drives in the same situations in Figure 1.3. In this situation, the automated vehicle (AV) yields and plans to drive in the roundabout in the fourth image of the sequence. The color-coding of the pixels corresponds to the value, while white pixels are either unreachable or in collision with SVs. Each of the presented images has a resolution of one-megapixel providing dense discrimination of the environment by the planner's reward model.

The hybrid architecture depicted in Figure 1.4 provides an important step towards scalability in automated driving. It employs the prediction blocks to learn what is difficult to model by hand while retaining interpretability through explicit motion prediction and planning. The unified optimization of prediction and planning via the aligned reward model is essential to implement assertive goal-directed

behaviors. In this hybrid approach, the planner can evaluate feasible driving behaviors in multiple possible futures by simultaneously leveraging explicit and implicit motion forecasting of the behaviors of SVs. This allows for selecting assertive behaviors with additional model-based risk assessment. The next chapter provides the requisite background for this thesis, facilitating a deeper understanding of the intricacies of the proposed contributions.

2

BACKGROUND

Contents

It is important to plan appropriate driving behavior to attain lasting acceptance of fully automated driving. Othman (2021) reviews social and technological determinants contributing to public acceptance. Accordingly, safety and trust in the vehicle are among the most influential factors. An effective way to establish trust and familiarity during a drive is to behave in accordance with the passenger's expectations. But how can an automated vehicle learn to behave humanlike? Numerous studies examine human driving behavior, especially the factors correlated with road safety (Elander et al., 1993; Sagberg et al., 2015; Taubman Ben-Ari and Skvirsky, 2016). The studies often distinguish between driving skill and driving style, which combined determine the resultant behavior.

Evans (1991) describes the ability to respond to complex traffic situations as driving skill. What aspects contribute to these skills can be observed in driver's education (Evans, 1991). Students acquire fundamental skills while learning to operate the vehicle controls and practicing to maintain the vehicle in the lane. In the next progression, students have to sharpen their visual search skills to process complex situations. In a comparative study of experienced and novice drivers, Mourant and Rockwell (1972) observe that experienced drivers are able to make use of their peripheral vision and focus on the future path, while novices show a large spread in fixations (Mourant and Rockwell, 1970). However, the authors point out that acquiring these visual search skills requires a great amount of training.

Other important components that determine the driving behavior are the goals, motives, and intentions of the driver, which contribute to the driving style. Sagberg et al. (2015) describe the driving style as the habitual way of driving. Although it is not directly apparent, the driving style has more influence on road safety as compared to the skill (Evans, 1991). This is due to the fact that the driving style has a direct influence on the driving task difficulty. As such, a chosen driving style may require advanced driving skills to maintain control of the vehicle (Näätänen and Summala, 1976). Traffic psychologists have analyzed human driving behaviors and conceptualized models of the control processes that drivers use to adapt their driving style (Fuller, 2000; Summala, 1988, 2007; Vaa, 2007). Fuller (2011) describes that an important aspect of these models is that drivers continuously evaluate the perceived task demand and their perceived capabilities to control an accepted level of risk or task difficulty. Further, the accepted level may depend on individual differences, such as the emotional state and journey goals. However, modeling these behavioral control aspects is difficult due to the large number of possible situations, behaviors, and actions. This thesis aims to incorporate the modular architecture into a training loop to adopt an appropriate driving style and gain driving skill. The following sections introduce machine learning and planning approaches that aim to imitate these cognitive control processes of human drivers.

2.1 MARKOV DECISION PROCESSES

Machine learning and planning algorithms often use the Markov decision process (MDP) as a mathematical decision-making framework. An MDP (Puterman, 2014) is often described by a 5-tuple $\{\mathcal{S}, \mathcal{A}, T, R, \gamma\}$. The agent can visit a set of states denoted by $\mathcal{S}$. A state represents the vehicle configuration, such as the position in the environment, the velocity, and acceleration. The description can be more detailed by including more derivatives with respect to time or abstract describing that the vehicle is following a specific lane. $\mathcal{A}$ describes the set of actions the agent can perform, such as accelerating, braking, and steering by adjusting the wheel angle. The transition function $T(a, s, s')$ describes how the agent moves over time from state s to the next state s' given the action a. The reward function $R(s, a)$ assigns a reward for applying the action a in state s. The reward function for automated driving encodes the driving style and is very difficult to specify manually. Planning aims to find a policy π, a function, that maps states to actions and maximizes the cumulative reward, where γ discounts future rewards against immediate rewards. An optimization technique used to find an optimal policy is dynamic programming (Bellman, 1966). However, dynamic programming suffers from the curse of dimensionality (Bellman, 1961) when applied to a problem

with high-dimensional state spaces, such as automated driving. The curse of dimensionality refers to the problem of exponential growth of dimensionality with a larger number of state dimensions. This also implies that optimizing driving behavior over a longer time horizon is difficult due to the growing number of states to be evaluated. The next section will introduce methodologies to predict driving behavior directly instead of using planning. Prediction is an important step in the modular automated driving architecture and is often synonymous with forecasting the future motion of SVs. This allows the planning algorithm only to evaluate the ego states relative to predictions because when planning for multiple agents simultaneously, the other agents' states must also be considered. When every agent can visit the same number of states, the state space grows exponentially for the number of agents. By introducing abstraction, deep learning approaches circumvent the explicit evaluation of very large state spaces.

2.2 PREDICTION IN AUTOMATED DRIVING

Deep learning has achieved state-of-the-art results in numerous research areas, including computer vision, natural language processing, and game playing. Many successful deep learning architectures of these research areas have been adopted to predict driving behavior. Chen et al. (2023) survey end-to-end approaches for automated driving on a broad spectrum from imitation learning to reinforcement learning (RL) and point out challenges and opportunities. This section provides an overview of deep learning approaches, focusing on those that predict ego behavior and those aimed at predicting the behavior of SVs. Goodfellow et al. (2016) provide a background in deep learning for further readings about modern practices and research areas.

2.2.1 *Imitation Learning*

Imitation learning algorithms assume experts can demonstrate the desired driving behavior, and recording them using sensor-equipped vehicles is feasible on a large scale. A major advantage of imitation is that drivers participating in collection campaigns do not need domain knowledge in automated driving but can contribute by directly specifying the desired behavior by demonstration. There are two categories of imitation learning algorithms. The first category aims to directly clone the actions of the expert demonstrations, while the second category strives to learn the preferences or goals of the expert instead (Osa et al., 2018).

In 1989, the first behavior cloning approaches were proposed for lane following, where a neural network performs lateral vehicle control (Pomerleau, 1989). The end-to-end approach is typical for behavior cloning, jointly optimizing perception and control, using supervised

learning with a regression or classification loss contrasting the actions of the demonstration and the agent. Conditional imitation learning extends the simple approach by utilizing high-level inputs during training, which also provides influence over the driving policy during deployment (Codevilla et al., 2018). For instance, it can allow the network to execute turns based on navigational inputs. The strengths of behavior cloning lie in its simplicity by allowing implicit feature learning of the environment from raw sensory data while jointly optimizing control. The downside is that the agent is trained only on states the expert policy visited and, when deployed in the real-world, might take actions leading to states that are out of the training distribution, leading to compounding errors that the agent is unable to recover from (Ross and Bagnell, 2010). DAGGER (Dataset Aggregation) by Ross et al. (2011) propose a solution by repeatedly querying a demonstrator on how to recover. Haan et al. (2019) describe another problem of behavior cloning, related to causal confusion, where the agent is unable to differentiate between true and misleading correlations. The authors argue that robustness against distributional shift can be achieved by relying on the true causes of expert actions. Spencer et al. (2021) describe that both problems arise from covariate shift, highlighting the importance of the input features. A stream of literature addresses this problem by augmenting the training dataset. Bojarski et al. (2016) record data with multiple cameras and use them to simulate the appearance of drifting from the lane center by using viewpoint transformations. Tampuu et al. (2020) provide an overview of data augmentation and diversification techniques in their survey of end-to-end driving training methods.

There are also imitation learning methods that first aim to learn the reward function and then use the learned reward function to optimize the agent's behavior. One of these methodologies is called IRL. It is used in the core contributions of this thesis, covered later in detail in this background chapter. Osa et al. (2018) describe that learning the reward function as a description of the expert behavior allows for better generalization since the reward function is regarded as a parsimonious description of the desired behavior. These methodologies are indirect as they necessitate planning or RL to determine the driving behavior. IRL will be reviewed in the following sections after introducing RL and planning approaches for automated driving. Ho and Ermon (2016) propose to directly learn the behavior without explicitly finding reward functions using GANs (Goodfellow et al., 2014). The approach uses a GAN, having two DNNs, one generator network to explore actions, and a discriminator network implicitly learning the reward function. The authors explain that the approach, known as generative adversarial imitation learning (GAIL), has the benefit of exploring the environment in contrast to behavior cloning methods, allowing better generalization in new environments. Exten-

sions of this approach aim to allow the agent to learn from a broader range of experience. Lee et al. (2020) introduce a constraint reward to the discriminator-based reward function to also be able to learn from negative demonstrations. The constraint reward provides positive feedback for positive demonstrations and negative feedback for negative demonstrations, such as collisions. Using collisions allows the agent to also visit unsafe state-action pairs that expert demonstrations do not cover, leading to better generalization. Often, GAIL approaches rely on a binary classification to discriminate the expert from non-expert behavior. Huang et al. (2023) argue that the binary classification often does not encode essential features of the expert behavior, leading to low-quality rewards. The authors propose using a contrastive learning loss to learn a more meaningful representation space, differentiating between good and bad states. Contrastive learning belongs to the group of self-supervised learning methods, which have the ability to learn representations from data without relying on human annotations (Gui et al., 2024). Imitation learning approaches can often utilize pseudo labels, which are derived from demonstrations, to leverage self-supervision.

2.2.2 *Affordance and Multi-Task Learning*

Affordance learning approaches take a step back from end-to-end behavior generation and instead focus on predicting features characterizing the vehicle's state that a controller can use to implement the desired behavior, such as steering the vehicle back to the lane center. This methodology leverages more a priori knowledge to improve the supervision of the learning task and constrain the behavior. The predicted features, being handcrafted, necessitate human annotation. Nonetheless, partial automation can also contribute to creating large-scale datasets akin to those used in imitation learning approaches, for instance, utilizing laser range finders for annotating distances. A benefit of leveraging features is that the vehicle does not have to drive with an optimal policy and can purposefully explore sub-optimal states for their annotation while recording data. The deep learning approaches use multi-task learning to predict multiple features simultaneously (Caruana, 1997; Zhang and Yang, 2021). Chen et al. (2015) propose the direct perception paradigm that uses end-to-end perception that predicts the yaw, distance to lane markings and preceding vehicles, and whether the vehicle is on or between lane markings. Sauer et al. (2018) propose conditional affordance learning as an extension to include directional input to navigate intersections. Both approaches assume that the lateral and longitudinal control can be separated and that behaviors can be defined and selected using rule-based behavior modeling.

A stream of research uses multi-task learning as auxiliary supervision while keeping the main task the behavior prediction (Bansal et al., 2019; Ishihara et al., 2021; Mehta et al., 2018). Chauffeur Net, an approach proposed by Bansal et al. (2019), imitates the expert behavior using imitation losses while using additional losses to make undesired behavior unlikely. The imitation losses include predicting the heading, speed, and waypoints, and the environment losses evaluate drifting off-road or colliding with other objects. The network predicts waypoints that a controller optimizes to find steering and acceleration commands. The authors render BEV input images for the network as an intermediate representation depicting road topology, traffic lights, speed limits, ego, and other agents. The intermediate representation allows the use of simulated and real-world data and facilitates closed-loop tests in simulation before deploying in the vehicle. The approach uses trajectory perturbations of the demonstration, such as offsetting the ego position from the lane center and fitting a new trajectory while keeping the end state, to learn recovering behaviors.

2.2.3 *Multi-Agent Motion Forecasting*

The modular architecture leverages prior knowledge about the environment. Predicting the future motion of SVs is an important aspect that allows the downstream planning modules to evaluate the ego motion relative to the SVs. This subsection presents a concise overview of general methods and output modalities, drawing on insights from several surveys that have addressed trajectory prediction in automated driving (Bharilya and Kumar, 2024; Ding and Zhao, 2023; Huang et al., 2022). Ding and Zhao (2023) describe output modalities of trajectory prediction modules, categorizing them into unimodal and multimodal methods. Unimodal methods yield a deterministic or a stochastic trajectory for each SV. On the other hand, multimodal methods fall into two sub-categories: implicit sampling from a probability density heatmap and explicit mode dividing methods, where the mode often corresponds to different driving intentions. Bharilya and Kumar (2024) categorize methods into conventional, RL, and deep learning-based approaches.

Conventional methods utilize physics, sampling, and probabilistic models, offering greater computational efficiency than deep learning methods. However, they do not capture complex interrelations between vehicle kinematics, infrastructure, and the interaction between agents.

The following Section 2.3 introduces single-agent RL methods, showing how an agent acquires experience through environment interaction. Multi-agent trajectory prediction requires the extension of single-agent RL formulations to account for the interaction among multiple agents. Several surveys provide an overview of multi-agent RL (Gronauer and

Diepold, 2022; Zhang et al., 2021). Notably, multi-agent RL can not simply use a simulator to evaluate the interaction with the environment since the policies of SVs are unknown. Strategies to circumvent this are self-play (Silver et al., 2017) and parameter sharing. Tang (2019) uses self-play to bootstrap the environment using previously trained models that iteratively replace other agents' policies. Konstantinidis et al. (2021) and Gupta et al. (2017) share the parameters of trained policies across agents to implement cooperative policies without explicit communication.

Supervised deep learning approaches make use of large datasets of real-world trajectory recordings. The datasets are typically processed by offboard perception systems to partially automate high-quality labeling (Caesar et al., 2020; Chang et al., 2019; Ettinger et al., 2021). These approaches fall into three categories: sequential, vision-based, and generative models (Bharilya and Kumar, 2024).

Sequential models, dating back to one of the first approaches, extract features from the agents' past motion using a temporal architecture such as recurrent neural networks (RNNs) and aim to predict multimodal trajectories with associated probabilities for each future. Recently, transformer-based architectures (Vaswani et al., 2017) secured high positions in Waymo Open Dataset Challenges. Shi et al. (2022) propose a transformer encoder-decoder for predicting multimodal future motion. The authors use a vectorized representation feeding historical trajectories and road maps as polylines. The decoder jointly optimizes global intention localization and local movement refinement. Improvements to the querying strategies allowed the authors to win in the consecutive year (Shi et al., 2023).

Vision-based models use raw data or depict the input in BEV. Recently, BEV perception approaches gained popularity, fusing information from different sensors and views (Ma et al., 2022). The representation excels in spatial feature extraction using convolutional neural network (CNN) architectures, especially when aiming to reduce the HD map dependency. Gilles et al. (2021) propose an approach generating a grid-based heatmap, representing the possible future position of an agent, capitalizing on the representational advantages for multimodal behavior. The authors propose sampling methods to derive trajectories from the heatmap. Kim et al. (2022) propose an approach that predicts trajectories and time-dependent occupancy grids for dense urban environments to capture group behavior, mitigating the need to represent agent-centric trajectories. Mahjourian et al. (2022) combine occupancy and flow prediction, addressing the shortcomings of occupancy grid prediction by additionally representing the agents' motion and identity.

Generative models such as GANs and variational autoencoders specifically aid in predicting multimodal trajectories. Gao et al. (2022) survey GANs for spatio-temporal data, presenting an overview of GAN

variants. The authors note that using GAN for spatio-temporal data is still in its infancy. Golchoubian et al. (2023) review specifically the domain of pedestrian prediction, showing multiple applications of generative models. Gupta et al. (2018) present an approach using GAN to represent uncertainty in pedestrian prediction and capture socially-accepted motion.

2.3 REINFORCEMENT LEARNING

While the previous sections focused on using supervised learning techniques to predict driving behavior, reinforcement learning aims to learn how to drive by interacting with the environment (Sutton and Barto, 2018). The interaction with the environment is typically modeled using an MDP, where the agent aims to maximize the expected cumulative reward to find the driving policy π. Sutton and Barto (2018) offer a comprehensive overview; the following adopts their notation to provide intuition about the concepts. The state-value function v_π and state-action value function $q_\pi(s,a)$ are given by

$$v_\pi(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s \right], \text{ for all } s \in \mathcal{S}, \tag{2.1}$$

$$q_\pi(s,a) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a \right], \tag{2.2}$$

and provide estimates on how beneficial it is to be positioned in a given state or respectively in a given state s and take a certain action a to maximize cumulative reward. The discount term γ is often used to weigh the future rewards against immediate rewards. Usually, the reward function $R(s,a)$ is defined a priori by the designer. The reward function facilitates the inclusion of prior knowledge regarding the desired behavior. However, it is difficult to find an appropriate level of abstraction. On a high level, one could design a reward function that rewards the agent for staying on the road while punishing the agent for going off the road. When being more specific, one could also reward the agent for keeping the vehicle centered between lanes. However, when driving in curves, keeping an offset from the lane center is often desired to minimize lateral acceleration. Being more specific helps in distinguishing good behavior from bad, however, it may limit generalization.

RL can use models of the environment or learn purely by interacting with the environment and mapping observations of states, such as raw sensory data, to actions. An important aspect to consider is the experience the agent has to gain by interacting. The agent can not only exploit its current knowledge of the world but has to keep exploring better ways of interacting with the environment, known as the

exploration and exploitation tradeoff in RL (Sutton and Barto, 2018). Model-based methodologies are considered more sample efficient because they incorporate important a priori knowledge such as feasible vehicle kinematics or environment features necessary to follow the driving rules.

Kiran et al. (2021) survey deep RL approaches for automated driving. The methods can be categorized into value-based, policy-based, and hybrid approaches, combining both. Value-based methods derive the policy from value estimates by maximizing the cumulative reward, while policy-based methods directly estimate the policy without explicitly expressing value. DNNs serve to approximate functions in RL, such as the policy or state-action value estimates. Numerous RL variants are utilized for continuous state-action spaces, with Leurent (2018) providing a survey on different state-action space representations specifically for automated driving. In automated driving, allowing an agent to acquire experience in the real world is dangerous, and substituting the real-world experience with a simulation poses the challenge of modeling realistic interaction with other SVs requiring multi-agent RL approaches. Learning can be further classified as online and offline RL. While online learning requires constant interaction with the environment, offline learning stores interactions with the environment in a dataset without necessarily knowing the behavior policy of the data collection. Offline RL provides a new perspective on designing scalable approaches that can use large datasets (Levine et al., 2020). Offline RL methods can also be applied to imitation learning when recording experiences from human demonstration.

The following section will briefly overview planning approaches for automated driving. The overview focuses on general-purpose algorithms that generate behaviors in all operational domains, from parking to urban and highway driving. This section also positions the planning algorithm utilized in this thesis within the context of related work.

2.4 PLANNING IN AUTOMATED DRIVING

Much of the literature about practical deployments of planning algorithms for automated driving originated from Defense Advanced Research Projects Agency (DARPA) Challenges such as the Urban Challenge in 2007 (Buehler et al., 2009). Paden et al. (2016) survey planning algorithms for urban environments, categorizing them into variational, graph-search, and incremental search methods. Paden et al. emphasize that the contestants in the DARPA Challenges often combine these methods to use their individual advantages. Planning modules can be decomposed into a hierarchical structure consisting of strategic, tactical, and operational layers. The following paragraphs will briefly

outline the layers, paving the way for a subsequent discussion on the complications that arise from explicit decomposition.

On a strategic level, a route planning module determines the path to a target location using lane-level road networks, also referred to a road graph, with Zheng et al. (2019) reviewing methods for generating these networks. The path is typically computed using algorithms such as Dijkstra's algorithm or more sophisticated variants incorporating additional optimization criteria, as reviewed by Bast et al. (2016) and Delling et al. (2009).

On a tactical level, behavioral planners initiate uniquely defined maneuvers, e. g., lane-changing, lane-following, and emergency-breaking. Initial approaches during the DARPA Urban Challenge use finite-state machines to switch between contexts such as driving in lane and handling intersections (Montemerlo et al., 2008; Urmson et al., 2008). Behavioral planners often have to cope with uncertainty in the situational assessment. On highways, these uncertainties are addressed using dynamic Bayesian networks by estimating hidden state variables such as whether a lane change is beneficial given factors such as the number of lanes and distances to preceding vehicles (Brechtel et al., 2011; Ulbrich and Maurer, 2015).

On an operational level, motion planners generate a feasible reference trajectory for feedback control instructed by the behavioral layer (Huang et al., 2019; Werling et al., 2010; Yuan et al., 2019). However, the hierarchical abstraction between behavior and motion planning introduces modeling challenges, especially in dynamic traffic situations, where maneuvers between moving objects require evaluating spatio-temporal features. These features depend on the vehicle kinematics, which the behavioral layer can not access directly. Therefore, maneuvers may be rendered infeasible due to overestimation or too conservative due to underestimation of the vehicle's capabilities. A stream of research aims to explicitly incorporate the interaction with different agents into planning to not require external prediction by using the mathematical framework of partially observable MDPs, which do not map states directly to actions but instead map observations of states to actions (Kaelbling et al., 1998). However, finding an exact solution for a partially observable MDP is considered intractable (Papadimitriou and Tsitsiklis, 1987), therefore approaches capable of online planning use hierarchical decomposition (Ulbrich and Maurer, 2013) or approximation methods (Galceran et al., 2017; Silver and Veness, 2010). Alternatively, to the decomposition mentioned above, a growing body of research inverts the hierarchy by first generating feasible trajectories and afterward selecting an appropriate behavior. The following will give a brief historical overview of these methods.

At the time of the DARPA Urban Challenge, Dolgov et al. (2008) used an extension of the well-known heuristic graph-search algorithm

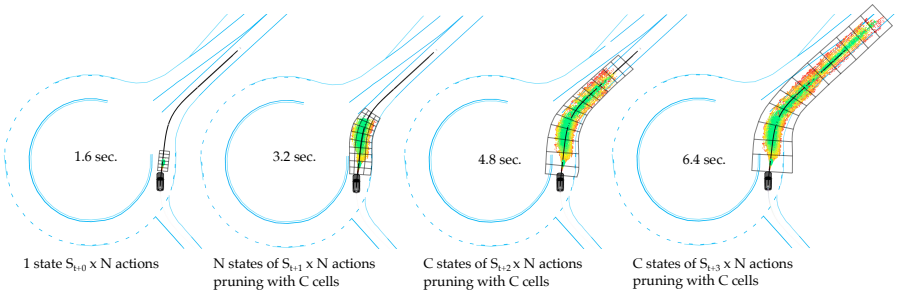


Figure 2.1. Illustration of the search methodology. The AV takes the first exit in a roundabout driving situation. The state space exploration is shown by sampling N actions in four time increments over the planning horizon. A linear state space growth is achieved by pruning the set of sampled states $\mathcal{S}_t$ to at most C possible survivors. The grid serves as a simplified visualization of the cell structure utilized for pruning in high-dimensional space.

A^* (Hart et al., 1968), called hybrid A^*. The authors point out that the main challenge in designing a practical planning algorithm is that the space of controls is continuous. In contrast to A^*, the vehicle's state during the search is 3D continuous while being represented in discrete nodes of the search tree. The search is performed forward-directed by simulating a vehicle model to expand nodes in the tree, providing the kinematic feasibility of the generated paths. The cost-to-go heuristic is used to select which nodes to expand. However, since the paths contained unnatural swerves, the approach uses a second non-linear optimization step to improve the solution.

McNaughton et al. (2011) introduced an exhaustive search algorithm, pointing out that cost-to-go heuristics are difficult to implement in dynamic driving scenarios. In the worst case, all vertex branches have to be expanded to find the desired behavior. The algorithm yields trajectories that respect vehicle kinematics and cover the workspace's spatial and temporal dimensions. To make the algorithm tractable for real-time planning, the authors generate a partial graph before the search having the form of a state lattice (Kelly and Nagy, 2003), which induces a discrete search graph on a continuous state space without considering time and velocity. The state lattice is augmented with time and velocity dimensions to search for trajectories. The graph structure suits itself for parallel edge evaluation on a GPU. A path between lattice vertices is defined as a cubic polynomial spiral in the Frenét coordinate system (Do Carmo, 2016). Under start and end boundary constraints, the parameters of the paths are calculated using closed-form and numerical integration under consideration of a kinematic bicycle model. The search assigns the trajectory's augmented state variables having the least cost to the vertex in case multiple trajectories reach a single lattice vertex.

This thesis uses a similar exhaustive search methodology to generate a set of feasible trajectories for long planning horizons. Figure 2.1 gives

an intuition about the outputs of the planner. The planning horizon is iteratively explored during a forward search using discrete transition times. This produces layers of states that share the same time increment. In contrast to McNaughton (2011), no lattice is used. Instead, in every time increment, a set of actions is sampled from distributions conditioned on the vehicle kinematics. This is done in parallel for every state in that particular time layer. The notion of time layers within an exhaustive parallel search has been described by Heinrich (2018). Like hybrid A^*, the tree nodes expand by simulating a kinematic vehicle model. During the numerical integration of the vehicle dynamics also, the features of the reward function are integrated. Although McNaughton and Heinrich have referenced features concerning motion, infrastructure, and objects within urban traffic environments, the procedures necessary for tuning those reward functions were not addressed in their work (Rosbach et al., 2019).

Planning algorithms that use an inverted planning hierarchy first explore the trajectory candidates and select the trajectory to be executed afterward. The selection is either based on the highest reward or other selection criteria. In practice, it is difficult to base the selection entirely on the reward function since multiple objectives have to be tuned such that the highest reward yields the desired behavior. Motion planning experts often stage the tuning process by concentrating on exploration first, and on selection subsequently. The features of the reward function are also separated accordingly into exploration and selection features (Rosbach et al., 2020a). The exploration features implement motion, infrastructural, and object-related metrics based on the vehicle state or action. The selection features are calculated on the sequence of states or actions after the exploration to distinguish between different behaviors. In this direction, Gu et al. (2016) generate selection features by applying topological analysis to the set of sampled trajectories. In the experiments of this thesis, the tuning of the reward function incorporates both feature types. The next section describes the learning methodologies to automatically find these reward functions.

2.5 INVERSE REINFORCEMENT LEARNING

The goal of IRL involves finding the unknown reward function for a planner or RL agent using demonstrations of the desired behavior. Therefore, it uses an inverse formulation to RL, as indicated by the name. The methodology is suitable for automated driving, as the desired behavior can be cost-efficiently collected by observing human drivers. A comprehensive overview of IRL in the broader scope of imitation learning is given by Osa et al. (2018), and a review of IRL methods, progress, and challenges is presented by Arora and Doshi (2021). The following provides a brief historical overview of IRL al-

gorithms, emphasizing challenges arising in the continuous control domain for automated driving.

In 1964, Kalman (1964) studied the inverse optimal control problem by recovering unknown objectives for linear control systems. Several decades later, Ng and Russell (2000) addressed the problem for an MDP, which is now predominantly referred to by the name of IRL. The authors point out that the problem is "ill-posed" as policies may become optimal for multiple reward functions. Abbeel and Ng (2004) proposed matching the expected feature counts of the policy computed by the learner and empirically calculated expectations of demonstrations. However, multiple policies may accumulate the same expected features. Ratliff et al. (2006) propose maximum margin planning (MMP), a methodology that formulates the learning task as maximum margin structured prediction over the policy space. The optimization includes constraints that limit possible cost function weights for which the demonstrated policies have higher expected rewards than other policies. The algorithm does this by a margin, which scales with a loss distinguishing good from bad policies. A subgradient method is utilized to optimize the convex and non-differentiable objective function. A limitation of the approach is the assumption that the demonstrated behavior has to be optimal and demonstrate better behavior than any alternative policy. In practice, expert demonstrations often include imperfect behavior.

In 2008, Ziebart et al. (2008) propose maximum entropy IRL. The IRL approach applies the maximum entropy principle to resolve the ambiguity introduced by sub-optimal demonstration behavior. The information-theoretic principle of maximum entropy describes the distribution, which makes the fewest assumptions about the demonstrated data (Jaynes, 1957). A policy π with a higher reward is exponentially more preferred in this probabilistic model. The gradient of the log-likelihood ∇L, with respect to the reward function weights $\boldsymbol{\theta}$, is the difference between the demonstration's empirical feature counts $\hat{\boldsymbol{f}}^D$ and the agent's expected feature counts, which are calculated using feature path integrals $\boldsymbol{f}^\pi$. This can be expressed as:

$$\nabla L(\boldsymbol{\theta}) = \hat{\boldsymbol{f}}^D - \sum_{\pi \in \Pi} p(\pi|\boldsymbol{\theta}) \boldsymbol{f}^\pi = \hat{\boldsymbol{f}}^D - \sum_{s \in \mathcal{S}} D_s \boldsymbol{f}_s, \tag{2.3}$$

where the expectation is calculated over all possible policies of a situation, denoted by the policy set Π. This maximum entropy IRL framework is the predominant choice for current IRL approaches since the probabilistic methodology is differentiable and suitable for function approximation using neural networks. Ziebart et al. (2008) propose an algorithm similar to value iteration to calculate expected state visitation frequencies D_s. This allows matching the mean features of the demonstrations $\hat{\boldsymbol{f}}^D$ with the state features $\boldsymbol{f}_s$ visited by the current policy. However, the distribution calculated over all possible

policies is often intractable to compute in MDPs with long planning horizons.

Kuderer et al. (2015) approach the problem of driving style imitation using continuous trajectories. The authors do not model the dynamics as an MDP. Instead, they directly compute the feature gradients of model-based time-continuous trajectory splines. While this approximate method is suitable for trajectory optimization on highways, its extensibility to more complex automated driving domains remains limited. In general, it is tedious to design features manually. The features have to describe the situation sufficiently well to generate a unique optimal solution. In this approach, the feature gradient computation requires differentiable features that are difficult to approximate for environments with complex infrastructure.

The first IRL approach of this thesis aims to address challenging urban driving situations by formulating a maximum entropy IRL approach in combination with a general-purpose planning algorithm for automated driving (Rosbach et al., 2019). The planning approach does not contain behavior implementations and computes feasible trajectories in different operational driving domains. The sampling procedure explores the high-dimensional configuration space and yields a large set of policy rollouts with implicit behaviors, which can be used to approximate the distribution of possible paths for the maximum entropy IRL formulation. The methodology focuses on tuning reward functions for a priori selected driving situation, employing a linear combination of K features f_i, each weighted by θ_i, such that

$$\forall (s,a) \in \mathcal{S} \times \mathcal{A} : R(s,a) = \sum_{i \in K} -\theta_i f_i(s,a). \tag{2.4}$$

In practice, this approach can be used to tune a set of reward functions for different driving styles or situations to overcome the limited generalization capabilities of a single linear reward function.

To this end, Babes et al. (2011) focus on automatically finding a number of linear reward functions using a latent variable model. To find these, the approach integrates the expectation maximization (EM) algorithm by Dempster et al. (1977). During inference, a mixture of reward functions is inferred using Bayes' rule. This process enables training across a variety of driving segments without requiring prior knowledge regarding the segments (Rosbach et al., 2020a). Choi and Kim (2012) suggest employing Dirichlet process mixture models (Neal, 2000) to overcome the limitations of EM, particularly the dependency on a predetermined number of clusters.

Another thread of related work concentrates on finding non-linear reward mappings for each state-action feature. Ratliff et al. (2009) propose Learning to Search (LEARCH), an extension to MMP, which predicts non-linear rewards for a planning algorithm in a grid representation. The reward functions in the form of grid-based cost maps

are used to represent traversability for an off-road planning system. Recently, Wulfmeier et al. (2015) propose a deep IRL approach using DNNs to approximate complex non-linear reward functions. The authors use CNN architectures to learn features from raw input representations and predict grid-based cost maps. Wulfmeier et al. (2017) apply this methodology to autonomous navigation in urban environments. The authors focused on predicting spatial traversability maps that can be combined with a path planner. However, the authors did not consider the continuous control domain, which requires embedding the agent's kinematics. Further in this direction, Drews et al. (2017) demonstrate that online model predictive control (MPC) can optimize cost-map predictions from visual inputs to acquire a direct link between perception and control for high speed autonomous driving on an oval race track.

The subsequent IRL approach of this thesis aims to address the limited generalization of a single reward function. It proposes a deep IRL algorithm that generates situation-dependent reward functions for the above-mentioned sampling-based planning algorithm (Rosbach et al., 2020a). The structure of the approach is similar to the work of Wulfmeier et al. (2017), who use CNNs to encode features of the driving situation. However, in contrast to extracting features and predicting costs on a state basis, the work on IRL in this thesis operates within policy vector space. The sampling-based planning algorithm generates a set of policies for which feature vectors embed static information about the environment and kinematics of the agent. The DNN encodes the driving context based on these policy vectors and maps the context into reward functions for upcoming planning cycles. Instead of predicting a grid-based cost map, a CNN network architecture predicts reward function weight vectors that parameterize feature path integrals in high-dimensional continuous state-action spaces. The reward function weights adapt depending on the encoded situation for each planning cycle of the receding horizon optimization. In contrast to Drews et al. (2017), our planner generates an explicit search space that facilitates the implementation of safety extensions for automated driving. The proposed IRL methodology neglects the temporal dependency among individual planning cycles, with each cycle optimizing its own MDP and transitioning the optimization task abruptly between cycles. This disregard may result in temporal inconsistency despite the temporal dependency of consecutive cycles. The temporal dependency between MDPs has been modeled in previous work regarding actions and objectives. On a level of actions, Sutton et al. (1999) formulate semi-MDPs that introduce the concept of options. Shalev-Shwartz et al. (2016) describe options in the context of automated driving as closed-loop policies guiding actions over a time frame, such as merging into the left or right lane. On a level of objectives, Krishnan et al. (2016) proposed hierarchical IRL. Hierarchical IRL decomposes tasks into sub-tasks, learns their reward functions, and

takes the relationship between individual sub-tasks into account. The authors state that modeling the sub-tasks increases the performance on RL benchmarks regarding convergence and robustness to environmental noise. Šošić et al. (2018) use Bayesian Nonparametric IRL to learn the spatial and temporal context of demonstrations. The final approach of this thesis adopts the concept of subgoal modeling over a sequence of planning cycles to predict reward functions according to learned task dependencies. The DNN uses attention mechanisms to observe context changes and transition reward functions (Rosbach et al., 2020b).

In contrast to model-based IRL approaches, there are also model-free variants for imitation learning. Finn et al. (2016b) address imitation in the continuous control domain without relying on known dynamics. The authors propose a guided cost-learning framework combining policy search with cost-learning from demonstrations. The cost function is trained in the inner loop of policy search. The authors use the maximum entropy principle in which samples of the learned policy solve the partition function. Ho and Ermon (2016) presented a model-free imitation learning approach that directly learns a policy without learning the cost function using GANs by Goodfellow et al. (2014). Finn et al. (2016a) describe the connection between GANs and IRL, which both train energy-based models of policies with maximum likelihood gradient.

2.6 UNIFIED PREDICTION AND PLANNING

A stream of research aims to combine the benefits of modular and end-to-end architectures similar to the approaches presented in this thesis. Hagedorn et al. (2023) survey design choices and architectures aiming to integrate prediction and planning to account for the interdependency between the AV and SVs. The authors contrast prediction and planning for automated driving to motivate design choices. The authors point out that one of the key differences is that planning is goal-conditioned. Planning can be given objectives, long term by the target location and short term by preferences, while prediction has to cope with uncertainty with respect to them. Furthermore, SVs do not necessarily need to follow the predictions, whereas the planned trajectory has to be safe and feasible. Hagedorn et al. (2023) categorize approaches into monolithic end-to-end, interpretable end-to-end, and manual integrations. The monolithic approaches do not explicitly consider SVs, similar to imitation learning approaches using behavior cloning described in Section 2.2.1. Interpretable end-to-end systems include the SV prediction as an auxiliary output. Both approaches implicitly incorporate predictions for the planning task, therefore, they cannot guarantee safety. Integrations that involve the manual design of interactions between AV and SVs demand more engineering

effort; however, they can leverage prior knowledge to find a desired trajectory.

Hagedorn et al. (2023) describe the system design based on the relationship between AV and SVs using four categories by Rhinehart et al. (2021): AV-led planning, SV-led planning, joint planning, and co-led planning. AV-led planning directly infers the AV's behavior independent of the future behavior of SVs. This results in aggressive driving behavior and collisions when SVs do not react to the AV. SV-led planning is used by the sequential processing chain of the modular architecture, predicting the behavior of SVs and reacting to them. Neglecting the dependency of the AV's behavior on the SVs, leads to conservative behavior. Joint planning assumes that all agents' behavior can be controlled by using a joint objective to optimize, assuming that an optimal global outcome exists. However, an SV does not necessarily need to act according to a global optimum, thus leading to unexpected behaviors. Co-led planning does not assume deterministic behavior of SVs and therefore has to consider contingent plans dealing with the uncertainty. Rhinehart et al. (2021) further separate into active and passive contingency planning. Suppose the AV expects multiple SV behaviors, then it implements passive contingency. If the AV can influence the contingent plans of SVs to reduce uncertainty, it is considered active. A research stream not only integrates prediction and planning, but also uses backbone convolutions networks to incorporate perception (Sadat et al., 2020; Zeng et al., 2019, 2020). This thesis focuses on unifying prediction and planning, operating on an intermediate BEV representation, leaving this as a topic for future work. The following reviews approaches that unify prediction and planning, where planning includes model-based feasible trajectory generation.

Zeng et al. (2019) propose a neural motion planner, which can be described as an interpretable end-to-end approach, producing a space-time cost volume. The cost volume evaluates a diverse set of feasible trajectories, capturing uncertainty and multi-modality of possible trajectories. The method is trained using a multi-task objective combining perception, prediction, and planning. The planning loss minimizes a max-margin loss, where the desired behavior is given by demonstrations of the AV, and randomly sampled trajectories present negative and undesirable examples. The authors describe the loss as sparse. However, combining perception and prediction loss improves learning of the intermediate representations. The network is trained with multi-task learning and multiple heads, which can therefore cause inconsistencies (Sadat et al., 2020).

Zeng et al. (2020) propose DSDNet, an extension to the previous work, that uses a deep energy-based model, explicitly modeling interactions with SVs using message passing. In their previous work, planning was independent of prediction and, therefore, can cause inconsistencies. DSDNet aims to model the joint distribution of fu-

ture trajectories of all SVs and conditions the predictions on the cost volume.

Cui et al. (2021) predict scene-level futures and optimize contingency plans. A module scores the probability of each future, and the planner evaluates multiple consistent futures by separately planning for them. The solutions share a nonconservative solution that unfolds into separate plans for the different futures.

Casas et al. (2021) propose an end-to-end model for driving without HD maps by predicting an online map and an occupancy map of dynamic agents. The representation is then used as a cost function for a neural motion planner. The dynamic objects are represented as a probabilistic motion field, where the motion modes are learned unsupervised.

Hu et al. (2023) propose a full-stack driving network from perception, mapping, tracking, prediction, and planning. Prediction is performed agent-centric and also scene-centric using occupancy forecasting. The perception and prediction modules use transformer decoder architectures, and the planner uses an attention-based architecture to predict waypoints. Queries connect the pipeline as a unified interface. The approach is more similar to multi-agent prediction literature, providing a prediction of the ego vehicle trajectory and additionally using the occupancy map to check for collisions.

2.7 QUANTITATIVE EVALUATION

The direct comparison of reward functions does not yield a quantitative measure of the reward learning progress because the optimal policy of an MDP is invariant under certain reward transformations (Ng et al., 1999). Therefore, indirect measures have to be considered that contrast the learned and demonstrated behavior. Nonetheless, tracing reward function weights allows for introspection into the prediction process. In practice, radar charts suit themselves naturally to analyze the variance and correlations among feature categories such as motion and infrastructure. For a limited number of situations, it is feasible to annotate situations and compare predicted with expert tuned weights.

The design of metrics that evaluate the system's behavior is challenging due to the situation dependence. In the evaluations of IRL, the focus is on comparing behaviors with expert reference trajectories. The situation complexity strongly influences the divergence to the expert reference, and therefore situations are categorized to some extent. The metrics themselves are calculated using the explicit state-action and feature space available after each planning cycle of the receding horizon optimization. The metrics can be categorized into online and offline metrics. The online metrics require running the planning algorithm again using the learned reward models to explore trajectories. The offline metric calculations use a buffer of trajectories

originally sampled using a different reward function initialization. Further, a differentiation is made between random and expert-tuned reward function initialization due to their impact on the sampling and, therefore, policy distribution.

The first metric, presented here and described by Rosbach et al. (2019), utilizes the odometry of expert reference trajectories to evaluate the agent's behavior. Given the explicit state-action space, it is feasible to calculate the distance $d(\zeta, \pi)$ between the odometry ζ and a policy π using the following integral:

$$d(\zeta, \pi) = \int_0^H \alpha_t ||\zeta_t - \pi_t|| \, dt, \tag{2.5}$$

where the distance between the odometry ζ and policy π is integrated from the ego state at $t = 0$ over the planning horizon H. The norm uses geometrical properties from the odometry motion states, such as the Euclidean distance in both longitudinal and lateral directions, along with the squared difference in the yaw angle. Additionally, this metric incorporates a discount factor α, which requires manual tuning. The next metric, the optimal policy distance (OPD) describes the agent's behavior. The metric quantifies the distance of the odometry ζ to the optimal policy π^* calculated as $d(\zeta, \pi^*)$ using Equation 2.5. This distance has a nonzero lower bound, which is a result of the discretization errors in the state-action space and the nondeterministic nature of the planner's optimization methodology (Rosbach et al., 2020b). Plotting this metric for individual planning cycles indicates the performance of the selected driving style across various situations. In the evaluations of the papers, distributions of the distance metric summarize the performance of the learned driving style of different approaches either for a whole dataset or segments with annotations of the situations.

To quantify the uncertainty of selecting a policy π, a probability is assigned to each policy using the softmax of the reward function weights $\boldsymbol{\theta}$ and feature path integrals $\boldsymbol{f}^\pi$ as

$$p(\pi|\boldsymbol{\theta}) = \frac{1}{Z} \exp(-\boldsymbol{\theta}^\top \boldsymbol{f}^\pi), \tag{2.6}$$

where the partition function Z necessitates summing over all possible policies in the policy set Π:

$$Z = \sum_{\pi \in \Pi} \exp(-\boldsymbol{\theta}^\top \boldsymbol{f}^\pi). \tag{2.7}$$

The log-likelihood of the demonstration (LLD) evaluates how likely it is that the selected policy reflects the characteristics of the demonstration.

The expected distance (ED) is calculated for the policy set Π generated during a planning cycle using the distance $d(\zeta, \pi)$ and probability of selecting a policy $p(\pi|\boldsymbol{\theta})$ for each policy π as

$$\mathbb{E}[d(\zeta, \Pi)] = \sum_{\pi \in \Pi} p(\pi|\boldsymbol{\theta}) d(\zeta, \pi). \tag{2.8}$$

This metric tracks the learning progress over epochs on the replay buffers.

An alternative to this metric is the expected value difference (EVD). This metric contrasts the expected value of the policy set Π and corresponding set of demonstrations Π^D in a planning cycle as

$$\begin{aligned} &\mathbb{E}[V(\Pi)] - \mathbb{E}[V(\Pi^D)] \\ &= \sum_{\pi \in \Pi} p(\pi|\boldsymbol{\theta}) V(\pi) - \sum_{\pi^D \in \Pi^D} p(\pi^D|\boldsymbol{\theta}) V(\pi^D). \end{aligned} \tag{2.9}$$

There could be multiple demonstrations π^D in a single planning cycle as denoted by the set Π^D. A demonstration π^D is described as the policy among the policy set, which has the least distance to the odometry. In this thesis, a trajectory refers to a sequence of motion states, such as those observed in the odometry of human drivers. A policy, particularly one derived from the planner, can be used to create a sequence of motion states. However, unlike a trajectory, the policy has a transition function and access to the environmental model for calculating reward function features. Projecting the odometry into the policy set using the distance metric enables access to the path integral features and the transition function. An alternative way to generate reward function features for the odometry trajectory is to directly calculate the features using the environment model based on the observed vehicle states and actions. However, this may result in an expert trajectory not being among the possible trajectory candidates of the planner, necessitating the inclusion of the trajectory in the partition function Z. Often, the scale of the weights increases during training, which results in larger EVD over epochs. The problem is mitigated by either scaling the weights or normalizing the EVD by the value of the demonstration. If it is feasible to manually tune a true reward function, the expected value can be calculated using the value under the true reward function. An alternative metric to the EVD is the expected feature difference. During training also the gradient can be considered as presented in Equation 2.3.

In Chapter 3, Section 3.5, the focus shifts from behavior alignment to learning to combine prediction and planning. An important analysis entails whether the agent can avoid other vehicles without the necessity of explicit behavior predictions. To facilitate this, the thesis employs a metric analyzing the gap maintained to other vehicles. The object time gap (OTG) metric computes the time it would take for the following vehicle to reach the position of the preceding vehicle, assuming the preceding vehicle remains stationary. Additionally, OTG is often utilized to categorize behavior as comfortable or uncomfortable when following other vehicles based on predefined thresholds.

3

PAPERS AND CONTRIBUTIONS

Contents

3.1 ADOPTING A HYBRID APPROACH

This section summarizes the paper:

> *Towards Hybrid Automated Driving: From Direct Imitation Learning to Affordance Learning*
> Sascha Rosbach, Antonia Breuer, Simon Barthel, Frederik Kanning, and Silviu Homoceanu

Published in *Proceedings of the Symposium AAET-Automatisiertes und Vernetztes Fahren*, Braunschweig, Germany, February 2019.

3.1.1 *Motivation*

The predominant approaches in automated driving can be broadly categorized into modular and end-to-end approaches. Chen et al. (2015) propose the direct perception paradigm, a simple hybrid that aims to overcome the design complexity of modular approaches while preserving interpretability. The paradigm uses end-to-end perception to predict affordances (Gaver, 1991) such as the distances to the lane center and preceding vehicles, and implements the behaviors using model-based control. The affordances aim to circumvent the need for a complete environment model, concentrating on essential features to assist human drivers in common driving conditions. The method's capabilities are confined to lane keeping and maintaining a safe distance from other vehicles, where the behaviors can be a priori-defined using independent model-based lateral and longitudinal vehicle controllers.

This paper focuses on overcoming the practical hurdles of applying the paradigm in the real world.

End-to-end approaches using behavior cloning are known to suffer from compounding error when only trained with optimal behavior that does not display how to recover when diverging from states of the optimal behavior. In response to this issue, the direct perception approach predicts affordances used as intermediate state representation, allowing model-based control to directly implement recovering behaviors. We observed that testing affordance predictions in isolation is insufficient to identify if a model will perform well in the real world.

The paper describes implementation details for applying the direct perception paradigm in the real world. We establish an efficient labeling process and describe details of the multi-task learning and control approaches. This paper proposes a semi-closed loop simulator using real-world recordings to evaluate the end-to-end perception and lateral control approach before deploying in the vehicle.

3.1.2 *Results*

We recorded data in multi-lane highway driving situations, segments with high curvature, such as exits on highways, and areas with lower speed limits, such as suburban arterial roads. In contrast to data collections for behavioral cloning methods, the data campaigns do not need to demonstrate optimal driving behavior. Instead, the drives show various states that could be encountered while driving in closed-loop that are not optimal, such as diverse offsets to the lane center. The dataset was then manually annotated using a partially automated system. Perspective transformations were applied to diversify the training dataset further.

The supervised learning tasks could be tested on validation and test datasets using standard machine learning metrics such as mean square error. We demonstrate the performance of the lateral and longitudinal modules by contrasting the distance estimates of the network to the lane center and the preceding vehicle with manual annotations. Models that performed well in metric-based evaluations did not necessarily perform well in closed-loop tests in the real-world. We implemented a semi-closed loop simulator for the lateral module and evaluated the time to failure. The simulator uses real-world data recordings and implements feedback via perspective transformation. The tests in the simulator focus on two aspects: firstly, evaluating the robustness of the perception system during drives on highly curved roads or with poor lane markings, and secondly, gauging the agent's capacity to execute recovering actions and realign with the lane center.

3.1.3 *Discussion*

The direct perception paradigm has a clear advantage over the behavior cloning paradigm by using interpretable interfaces between perception and control. The perception system directly predicts a priori-defined affordances that are important to implement the actions. Deep learning is used to generalize while presented with drastically changing appearances of road markings and preceding vehicles. Noisy predictions can lead to intermediate state representation for which the model-based controllers cannot implement the correct behaviors. The proposed semi-closed loop tests allow testing of the system performance using unlabeled data recordings and evaluate the task performance under prediction uncertainty. However, the model-based controller and the simulator based on perspective transformation only focus on lane-centered driving and do not scale to more complex driving situations.

Sauer et al. (2018) propose an extension to the direct perception approach that also considers high-level directional inputs to condition the affordance learning. This way, the authors aim to scale the approach to driving in more complex situations beyond lane-centered driving. On the one hand, the authors condition the affordances on high-level inputs, and on the other hand, the model-based controllers implement different driving modes depending on affordance thresholds. However, it is infeasible to enumerate all driving modes in urban driving situations and implement controllers with a priori-defined behaviors. Furthermore, more complex driving situations such as roundabouts require active behavior planning instead of reactive control, given the behavior of other agents.

Thus, this thesis focuses in the next sections on improving the modular architecture for automated driving that uses active model predictive planning instead of further pursuing affordance learning approaches.

3.1.4 *Contributions*

Silviu Homoceanu introduced me to the ongoing research projects in end-to-end driving and affordance learning, which he led. At the time, I had to familiarize myself with the ongoing research in this direction. I focused half of my working time on researching the possible direction of this thesis, reviewing related work, and the other half on supporting the ongoing research.

Simon Barthel and Frederik Kanning implemented the framework and worked on the vehicle integration under the supervision of Silviu Homoceanu. During the implementations, I was involved in discussions with Simon Barthels and Frederik Kanning. I supported the vehicle integration to become familiar with the framework, prototype

vehicle, and challenges regarding real-world applicability. I structured the paper with Antonia Breuer's help and led the writing process. I wrote the paper with Antonia Breuer and Simon Barthel with inputs from Frederik Kanning.

3.2 AUTOMATING REWARD FUNCTION TUNING

This section summarizes the paper:

> *Driving with Style: Inverse Reinforcement Learning in General-Purpose Planning for Automated Driving*
> Sascha Rosbach, Vinit James, Simon Großjohann, Silviu Homoceanu, and Stefan Roth

Published in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Macau, China, November 2019.

3.2.1 *Motivation*

General-purpose planning algorithms combine behavior and motion planning by sampling high-resolution actions in a forward-directed incremental search. Embedded in an MPC architecture, the planner yields a massive set of policies with implicit behavior after every planning cycle. The optimal behavior depends on the reward function, which encodes the driving style by mapping static and kinematic features into rewards. Typically, the reward function has to be tuned by motion planning experts to produce the desired behavior. However, this process is very tedious and costly, especially when tuning the behavior of a two-dimensional control optimization in a mixture of situations.

In this paper, we concentrate on the automation of the reward function tuning process. We propose a maximum entropy IRL methodology, where human driving demonstrations serve as examples of the desired driving behavior. The gradient, based on expected feature matching, facilitates the training of the approach through gradient descent. The features of the human driving demonstrations can be generated cost-efficiently. The training process does not require the planning algorithm to run within the inner loop of reward learning. During training, a small gradient implies that the learned reward function produces expected features that match those of the demonstrations. However, the partition function for this expectation is often intractable. Our contribution provides details of our sampling-based planning algorithm that approximates the expectation similar to Markov Chain Monte Carlo methods.

3.2.2 *Results*

The experiments were conducted by recording manual driving demonstrations on a set of straight and curvy roads using a prototype vehicle. After that, the proposed offline training methodology has been performed for each curvature type. The evaluation compares the results of automated tuning against the traditional manual tuning process.

Further, the experiments contrast different levels of a priori knowledge about the reward function initialization, namely random and expert initialization.

The evaluation of the training indicates a reduction of both ED and EVD. This suggests that policies having a humanlike driving style become more likely, as compared to other policies. A large reduction of both metrics can be observed across situations.

After training, the learned reward functions have been tested on dedicated test routes and compared in playback against manual human drives. All learned reward functions improve the driving style in terms of OPD on the test segments. Even so, in the case of tuning from random initialization, which indicates more significant improvements. However, expert-initialized reward functions show less variance of OPD on the test segments. The analysis of expected value and distance shows an inverse relationship with a higher rate of change in learned reward functions. Therefore, the learned reward functions introduce a higher degree of bias in the policy evaluation, making the demonstrated behavior unique and preferable.

3.2.3 *Discussion*

Motion planning experts are able to tune reward functions. However, feature and system parameter updates may require re-tuning, which makes the process costly. In fact, the reward function often does not contain more than a manageable set of features and parameters to make manual tuning feasible. Such manual handling and conflicts among the objectives make it difficult to produce a desired behavior using a single reward function alone in a mixture of situations. The automation of the tuning process, on the one hand, opens the possibility of extending the feature set and, on the other hand, allows tuning multiple reward functions that are adjusted specifically for a certain situation. The experiments show that situation-specific tuning exceeds the driving performance of a single manually defined reward function. This provides a step towards scaling over diverse driving situations.

IRL algorithms typically require running an RL or planning algorithm in the inner loop of reward learning to update the policy (Osa et al., 2018). This makes IRL difficult to work with and, generally, training more time-consuming. Our exhaustive search-based planning algorithm allows us to run planning once for all records and trace policy sets for each situation in replay buffers. The policy sets approximate all kinematically feasible actions and includes path integral features of each policy, as well as the projected distances to the demonstration required for offline training. The reward function plays an important part in generating the policy sets, hence also while generating the replay buffers. Therefore, the experiments further examine offline training under different reward function initializations. We show that training

is even possible without prior knowledge of the reward function using random initialization. This is a more difficult starting position than in practice, where a reasonable initialization can be chosen, e. g., using a reward function that performed well on straights as a seed for curvy situations. The results show that both initializations converged. In the direct comparison, the expert initialization produced a lower variance of the OPD on the test track. In the future, we plan to perform small updates of the replay buffers by running the planner again.

3.2.4 *Contributions*

Simon Großjohann introduced me to the existing simulation environment, planning algorithm, and the drawbacks of the pre-existing system. I formulated the approach and learning algorithm with input from Vinit James. I designed the experiments and recorded real-world driving data using an existing prototype vehicle. I implemented the data recording and inference interface for the planning framework. Vinit James implemented a framework to perform training of the models. I wrote the paper with the help of Silviu Homoceanu and Stefan Roth.

3.3 PREDICTING SITUATION-DEPENDENT REWARD FUNCTIONS

This section summarizes the paper:

> *Driving Style Encoder: Situational Reward Adaptation for General-Purpose Planning in Automated Driving*
> Sascha Rosbach, Vinit James, Simon Großjohann, Silviu Homoceanu, Xing Li, and Stefan Roth

Published in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France, June 2020.

3.3.1 *Motivation*

Planning algorithms for automated driving often operate in a spatio-temporal workspace. The algorithms optimize a reward function, which maps features of environment and vehicle kinematics into rewards. In practice, often, a single linear reward function is tuned to cope with anticipated situations. However, a single linear reward function is unable to specify the optimization task across diverse driving situations.

To overcome this limitation, we propose a deep learning approach that predicts a situation-dependent reward function for every MPC planning cycle. Importantly, the mapping between situations and reward functions does not rely on annotations and is automatically learned by a deep IRL approach. The DNN architecture encodes a latent representation of the situation and directly outputs a set of reward function weights. The training methodology builds on the maximum entropy IRL approach of the previous section.

The DNN utilizes policy feature vectors generated by the planner as input to encode the situation's semantics in every planning cycle. These vectors are a concatenation of actions and path integral features. The DNN architecture uses one-dimensional convolutions over the policy feature vectors to learn the latent features of each policy. All latent policy vectors are combined using fully connected layers to encode the driving situation.

The approach allows us to combine the reliability of model predictive planning by evaluating geometric and kinematic models with the generalization capabilities of deep learning, which has the ability to learn how to imitate human behavior across mixtures of situations.

3.3.2 *Results*

We conduct experiments on a route in the city center of Hamburg, Germany, and separate the situations into four groups with different levels of complexity. The first group presents the simplest situations, primarily characterized by lane following at constant velocity. The

second group includes traffic light scenarios, where the task is to stop and start. The third group encompasses a set of sharp turns. The last group combines stopping, starting, and performing a turn. The road networks include multi-lane traffic and large intersections.

The experiments compare the proposed deep IRL approach against latent and linear IRL. Latent IRL uses Expectation-Maximization to automatically generate a set of reward functions and infer a mixture using Bayes' rule. Linear IRL is trained individually and switched with a priori knowledge about each situation during inference. All approaches converge in ED for the utilized feature set, which combines static, kinematic, and high-level selection features. The reward function weight plots indicate nuanced predictions of the deep IRL approach with a high variance within each situation. The driving performance is measured using OPD for each situation. The DNN performs at par with multiple linear reward functions, trained individually for each scenario.

3.3.3 *Discussion*

The situations that combine multiple tasks, such as starting and stopping, introduce conflicts in the optimization objective. The deep IRL addresses the conflicts by adapting the reward function situation-dependently. As expected, the situation combining stopping, starting, and turning is the most difficult and shows the largest variance of the OPD across situations. Both deep and latent IRL predict situation-dependent reward functions. However, deep IRL shows a larger weight variance. The variance of the predicted reward functions of the deep IRL approach is proportional to the situation's complexity. It is noticeable that deep IRL also produces higher variance within the simple lane following situation, where latent IRL shows its predominant parameter pattern. The pattern indicates a strong prior for optimizing lane keeping at a constant velocity. In the lane follow situation, constant linear weights perform best, which produce persistent behavior. The reward function's temporal adaptation has been out of the scope of this paper. The next section introduces an approach to reduce the variance by considering a sequence of planning cycles.

The proposed approach is practical and scales well since it does not require manual annotations to predict situation-dependent reward functions. Furthermore, the data generation for offline learning can be automated to train on large datasets. This study only uses fifteen minutes of driving demonstrations across scenarios, showing that reward prediction for static environments, though with complex infrastructure, can be learned without necessarily having large datasets. However, due to MPC cycles in playback, we generate for fifteen minutes of driving approximately 5000 planning cycles, each having a set of 5000 trajectories.

In this study, we did not take moving objects into account. The approach is not limited to static environments but requires motion predictions of other objects to consider spatio-temporal proximity features in the reward function. Consequently, the dynamic objects increase the situational diversity and make the situation encoding tasks more complex, and therefore, training requires larger datasets.

3.3.4 *Contributions*

I formulated the approach and learning algorithm with input from Vinit James. Vinit James extended the training framework to experiment with different neural network architectures during his master's thesis, which I supervised. I extended the data recording and inference interfaces for the planning framework. I designed the experiments and recorded real-world driving data with the help of Simon Großjohann. Xing Li implemented the reference models with my help. I wrote the paper with the help of Vinit James, Silviu Homoceanu, and Stefan Roth.

3.4 TRANSITIONING CONTEXT-DEPENDENT REWARD FUNCTIONS

This section summarizes the paper:

> *Planning on the Fast Lane: Learning to Interact using Attention Mechanisms in Inverse Reinforcement Learning*
> Sascha Rosbach, Xing Li, Simon Großjohann, Silviu Homoceanu, and Stefan Roth

Published in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, USA, October 2020.

3.4.1 *Motivation*

In MPC, the agent plans for a specified planning horizon and periodically replans to address environmental changes. Ideally, the policy stays optimal for succeeding cycles, resulting in smooth and persistent behavior over an extended planning horizon of multiple planning cycles. By this means, the agent pursues and optimizes its long-term goals. In practice, planned behaviors quickly become sub-optimal, especially in dynamic driving situations involving multiple moving vehicles. To drive optimally, the agent has to know when to switch the driving task and when to be persistent to optimize long-term goals.

This paper utilizes the previous work's deep IRL encoding and the situation-dependent reward function prediction methodology. As an extension, this work refines the network's feature extraction capabilities and enables capturing the driving task dynamics, which is only observable over a sequence of planning cycles. We test the architecture in dynamic driving situations, where the agent drives close to other objects and interacts over different sequence lengths. In contrast to our previous work, the chosen test setting is more difficult. The agent must control the complete driving task by adapting its reward functions. Here, without collision checking in place to select collision-free policies; instead, we always execute the optimal policy.

Two challenges arise on the feature extraction and prediction level. First, the sampled policy set includes non-human-like action sequences and feature vectors that add noise when input to the neural network, especially when using models with long input sequences. Second, situation-dependent reward function predictions have to be persistent and stable depending on the observable driving context. In practice, it is difficult to tell when a new driving situation arises. Situations and corresponding driving tasks blend into each other. A transition dynamic governs the translation from the context to the task sequence and regulates whether the agent must respond quickly or persist.

Previous work in machine translation and action recognition benefited from incorporating attention as a mechanism built into the DNN architecture (Bahdanau et al., 2015). We utilize attention to filter relevant information from the input space and produce a low-dimensional

context vector used in a temporal attention network to predict reward function transitions.

3.4.2 *Results*

We conduct our experiments on an oval course with multiple lanes and track exits. Driving on this track requires the evaluation of routing involving lane-based target proximity costs. The agent's target velocity is set to exceed the velocities of the other vehicles to enforce an aggressive driving style. Many vehicles close to the agent open and close small time gaps, which lead to interaction by following, passing, and merging.

We compare the convergence during training, which measures the context encoding capabilities of our proposed DNN architecture against baseline approaches using EVD. In these complex situations, all deep IRL approaches converge to a similar EVD, whereas linear IRL does not attain a low EVD. We perform inference on a sequential validation dataset and calculate the ED over the training epochs.

Additionally, the trained models are evaluated on a sequential test dataset to compare the sequential prediction performance by measuring the OPD. The temporal attention network performs best on the test and controls the complete driving task in closed-loop inference in the presence of other vehicles. The temporal attention mechanism activates reward functions and generates a mixture over the input reward functions sequence. In contrast to our baseline approaches that always take the mean over the history size, temporal attention can control the variance of the predicted reward function.

3.4.3 *Discussion*

We choose to separate the architecture into two attention mechanisms. Both are implemented to perform a specific task and provide introspection via attention activation. The policy attention mechanism operates on a context encoding level by masking out misleading features and indicating relevant policies. The temporal attention mechanism operates on reward functions and has the task of transitioning between them based on the sequence of context vectors.

In contrast to a baseline method, the policy attention network has eight times fewer parameters while having similar context encoding capacities and prediction performance on the sequential validation dataset. Our policy attention uses a single branch scaled dot product to activate the context vector. The encoding capacities could be enhanced in the future by utilizing multi-head attention similar to the Transformer attention mechanism (Vaswani et al., 2017). Ideas from the Transformer language model can further be used to improve the position-based encoding.

Our temporal attention network uses a two-layered RNN network to model the temporal dependency of the planning cycles. The overarching concept aligns with EM-IRL methodologies, which infer a mixture of reward functions. Here, the mixture is generated by the temporal attention network in the form of an attention weight vector. This mechanism enables controlling the responsiveness of the mapped reward functions for future planning cycles. The highest responsiveness is achieved if the attention is on the last context vector.

We used a semi-supervised attention loss in addition to the maximum entropy gradient, which did not improve nor decrease the results significantly in terms of EVD or ED. However, it allows a principled approach for using self-generated annotations, e. g., the distance of a policy to the demonstration combined with the IRL gradient formulation.

3.4.4 *Contributions*

I formulated the approach and learning algorithm with help from Xing Li. I refactored the training framework to process sequential data, implemented the data recording and integration with the planning framework. Xing Li extended the training framework and experimented with neural network architectures to perform sequential prediction in his master's thesis, which I supervised. Xing Li researched attention mechanisms focusing on the connection to behavior cloning resembling policy attention. In discussion, we came up with the idea of splitting the attention mechanisms. I designed the experiments and modified the simulation environment with the help of Simon Großjohann. I wrote the paper with help from Xing Li, Silviu Homoceanu, and Stefan Roth.

3.5 UNIFYING PREDICTION AND PLANNING

This section summarizes the paper:

> *Pixel State Value Network for Combined Prediction and Planning in Interactive Environments*
> Sascha Rosbach, Stefan Leupold, Simon Großjohann, and Stefan Roth

Available as *arXiv Preprint*, arXiv:2310.07706 [cs.RO], October 2023.

3.5.1 *Motivation*

The traditional modular automated driving architecture considered until now uses a sequential processing chain consisting of perception, prediction, planning, and control. In this architecture, the planning and prediction are often designed separately, meaning that the prediction module lacks awareness of the downstream planning task. Neglecting the interdependency between planning and prediction simplifies the architecture yet may result in conservative driving behavior. The defensiveness is driven by predictions that shape the optimization features of the planner, ensuring a safe distance is kept from these predictions. However, the actions of the AV can influence the behavior of SVs and provide an opportunity to implement assertive behavior. This work aims to unify prediction and planning to allow for interactive and assertive driving.

State-of-the-art deep learning approaches in multi-agent prediction aim to predict the joint future of all traffic participants. The methodologies provide a planner with probabilistic and multimodal predictions of SVs. However, it is difficult to implement the desired driving style based on uncertain motion predictions. This work aims to simplify the architecture by implicitly representing the dynamics of the environment. In contrast to related work, the proposed deep learning methodology predicts time-dependent image sequences encoding joint SV and AV behavior that the planner can directly use to select a driving policy. The deep learning training is posed as a general-purpose image-to-image prediction task using BEV images of the environment as input and two time-dependent image sequences as output. The model predictive planner (MPP) of the previous work is used to render the target images for offline training using a configuration that densely samples feasible kinematics. The first sequence represents value estimates for the AV implicitly encoding kinematic reachability, object dynamics, safety, and driving comfort. The second sequence contains future pixel visitations of other agents to improve the implicit understanding of the environment dynamics in the value prediction. The experiments use the pixel state values as a reward shaping function for the planning algorithm to select a policy from a sampled set.

3.5.2 *Results*

The training dataset was compiled from real-world scenarios recorded on a track in Ingolstadt, Germany. These scenarios encompass arterial and urban roads featuring multi-lane traffic, intersections, and roundabouts. A total of 34 drivers participated in the data collection campaign, during which 24 hours of driving data were recorded. The drivers were categorized into three groups based on their level of driving experience: chauffeurs, experienced, and novice. The test dataset was generated through simulation on a test track in Wolfsburg, Germany, reflecting similar road types as those recorded in the real-world scenarios.

The dataset for offline training was generated by replaying real-world situations and rendering both input and output images. These images, with a resolution of one megapixel, provide dense discriminative feedback on the AV's reachable positions from a BEV, encompassing both spatial and temporal dimensions. While replaying the situation, the AV's starting position is adjusted to augment the dataset. For instance, the AV is moved closer to preceding vehicles, slightly offset from the lane center, or its starting direction is altered to enforce recovery behavior from suboptimal states. This paper provides qualitative and quantitative assessments of the pixel state value network (PSVN). The qualitative examples demonstrate that the DNN successfully learns dense sampling and policy evaluation from the MPP algorithm. The quantitative evaluation compares the behavior of the learned reward shaping function to a reward function utilizing features of a prediction module. The evaluation examines both behaviors using OTG on the test dataset, utilizing a confusion matrix for analysis. It reveals that the learned behavior does not exhibit reckless driving, having learned where objects are moving to avoid potential collisions. Compared to the reference reward function, the learned behavior is more assertive, opting for smaller time gaps and making more progress along the route.

3.5.3 *Discussion*

Predicting image sequences instead of trajectories facilitates the representation of arbitrary distributions in pixel state space, effectively capturing the interaction among multiple agents. This representation, focused on the downstream planning task, leverages high-resolution images to depict desired behavior and identify pixels to avoid due to collision or reachability constraints. This formulation enables the use of general-purpose image-to-image translation approaches, eliminating the need for specialized designs like graph neural networks to model joint behavior interaction. Additionally, rendering image sequences as channels of a single output image allows for the fusion of

multiple prediction tasks, such as predicting explicit pixel visitations of other agents and pixel state values.

While utilizing images offers high-resolution spatial discrimination, the approach currently employs only six temporal layers in each sequence. As a result, it does not discriminate actions as effectively as the feature path integral of the reward function within the planning algorithm. An objective of future work could be to increment the number of temporal layers to 30 so that the time intervals of each layer align with the vehicle model integration time, which is currently set to 0.2 seconds. The behavior selected entirely by the reward shaping function lacks the consistency seen in the model-based reward function, indicating a deficiency in jerk awareness. However, merging both the reward shaping term and state-action rewards appears promising.

Currently, the MPP algorithm employs a single linear reward function to render the target image sequences. The situation-dependent reward function methodologies discussed in the previous section can be applied during rendering to enhance situation-dependent discrimination further. The reward shaping term can also be viewed as a new path integral feature for utilization in deep path integral IRL. Combining the reward shaping term and the path integral reward function would necessitate learning the weights from human demonstration.

The rendering process employs a configuration of the MPP that densely samples high-resolution actions and utilizes a larger state-action space than previous approaches, as this planner configuration is not required to run on the target vehicle platform at deployment. While runtime-optimized planners may focus on refining solutions around previous exploration cycles of the receding horizon, this approach enables leveraging the learned exploration that can simultaneously evaluate multiple adjacent lanes in its rendering configuration. Using the PSVN as an admissible heuristic could reduce the hardware demands for vehicle deployment and accelerate the search.

Certain simplifications were accepted in the experiments. The trajectories of the SVs were derived from the prediction module of the conventional processing chain, not from real-world recordings. Consequently, the explicit future pixel visitations in the target sequence in the training dataset embodied multimodality, which would not be present if real-world trajectories were utilized. Future work aims to utilize ground-truth trajectories for the training dataset and further implement the interfaces to open-source benchmark datasets.

The exploratory starts used in the training dataset augmentations are a simple way to diversify the behavior and render state values that show responses to suboptimal initial states such as too close proximities to preceding vehicles. However, altering the initial position of the ego could introduce non-realistic scenarios that fail to mirror accurate interactions. A promising extension to this work would be to explore the domain of combined paired and unpaired image-to-image

translation (Tripathy et al., 2019), thereby relaxing the constraint that presumes proper interaction can always be rendered in target images.

3.5.4 *Contributions*

I formulated the deep learning approach and implemented the training framework. I implemented the extension to the MPP to render the inputs and outputs of the image-to-image translation task. Simon Großjohann helped in discussions and provided inputs to speedup the GPU kernels. Stefan M. Leupold encouraged to focus on image sequences and experiment with interactive situations. He helped me to perform evaluations against ground-truth predictions. I wrote the paper with Stefan M. Leupold and help from Stefan Roth.

4

DISCUSSION

Contents

This thesis introduces a hybrid approach that synergistically fuses the advantages of model predictive planning and deep learning while concurrently mitigating the inherent limitations of each standalone approach. Specifically, model predictive planning contributes to reliability and interpretability by evaluating geometric and kinematic models. Additionally, deep learning facilitates continuous improvement of prediction and planning by learning from human driving demonstrations and simulated experiences. This thesis takes five incremental steps toward scalability in automated driving. The initial step examines a simplistic hybrid approach combining end-to-end perception and model-based control in the direct perception paradigm. The next step automates the tedious and costly work of reward function tuning. The subsequent step generalizes model predictive planning by predicting situation-dependent reward functions using deep learning. The subsequent refinement of the neural network adapts reward functions situation-dependently across planning cycles using attention mechanisms to produce consistent driving behaviors. The unifying step combines prediction and planning by directly learning the expected cumulative rewards to evaluate trajectories in interactive environments, making explicit object motion prediction optional.

4.1 SUMMARY OF CONTRIBUTIONS

Section 3.1 examines the direct perception paradigm, which resides between the end-to-end and modular approaches, embodying a simplistic hybrid methodology. Two multi-task DNNs are employed, one for lateral control to predict the curvature and distance to the lane center, and another for longitudinal control, classifying the presence

of a preceding vehicle and, if present, determining its distance. The study describes implementation details and focuses on the real-world application. Given its inability to scale for complex driving scenarios, this concept was set aside in favor of a modular architecture utilizing model predictive planning.

Section 3.2 proposes a path integral maximum entropy IRL approach that automatically finds a tuned reward function for general-purpose planning algorithms using exhaustive search (Rosbach et al., 2019). The approach utilizes human driving demonstrations to train offline without running the planning algorithm in the inner loop of reward learning. Path integrals of the planner allow the IRL approach to be tractable for high-dimensional state spaces. The experiments show that training yields situation-dependent reward functions for a priori-defined situations. These reward functions enhance the driving style, surpassing the level achieved by expert-tuned reward functions.

Section 3.3 proposes a DNN to predict situation-dependent reward functions for the planner (Rosbach et al., 2020a). The deep learning approach encodes the driving situation by inputting the planner's sampled driving policies. These include path integral features and observations of the actions that produced them. In the experiments, we compare reward function predictions by our proposed neural network against multiple linear and clustered reward functions. Evaluations on complex urban road networks show that the predictions perform on the level of multiple linear reward functions tuned with prior knowledge of the situation.

Section 3.4 proposes a refined deep learning architecture to interact with moving vehicles and produce persistent decisions over a sequence of planning cycles (Rosbach et al., 2020b). The DNN architecture embeds information of the driving context from a sequence of planning cycles and predicts smooth reward transitions using two attention mechanisms: First, a policy attention mechanism that masks out non-expert-like behavior in policy space and generates context vectors. Second, a temporal attention mechanism observes sequential patterns from the context vector sequence and predicts reward functions. The experiments show that the DNN architectures with attention mechanisms outperform our baseline deep-learning approaches while having fewer learnable parameters. The approach, while having the autonomy to select any trajectory within the explored set, successfully assigns the highest value to trajectories that avoid collisions, even in the presence of moving vehicles. This selection demonstrates its inherent capability to interact safely without explicit restrictive model-based supervision on the selection.

Section 3.5 proposes a deep learning approach that predicts PSVs for the MPP to evaluate trajectories. The PSVs implicitly encode reachability, driving comfort, object dynamics, and safety. The network takes BEV images as input and predicts the PSVs in an image sequence. In

the experiments, the implicit object handling is contrasted with the modular prediction and planning architecture. The PSVN can implicitly predict the future motion of objects and provides state values for diverse road types, including roundabouts, thus eliminating the need for an external prediction module.

The hybrid decision-making approaches are still in the early stages of development. The next section covers the potential limitations of the approaches, which will be revisited in proposals for future work in Section 4.3.

4.2 POTENTIAL LIMITATIONS

The modular architecture has strengths and weaknesses. Since the planning module is located at the end of the processing chain, as displayed in Figure 1.1, the planner must receive accurate information about the perceived static and dynamic environment. In contrast to end-to-end approaches, the deep IRL approaches rely on explicit representations for planning and take as input sampled driving policies, which aggregate all information from the prior modules. Hence, during inference, the errors in perception, localization, and prediction may directly propagate. Also, our approach relies on an accurate estimate of the ego-motion state to perform initialization during playback, which is required to sample actions from this state. Errors in the initial state estimate could lead to skewed demonstrations.

Path integrals allow the deep IRL approach to operate in a high-dimensional state-action space. These integrals describe the observations in the static and dynamic environment for each policy. The feature extraction in the DNN is performed on the sampled policy set. There is no direct spatial relationship between policies on the input level. Therefore, the networks utilize one-dimensional convolutions to encode each policy's latent features. We expect performance improvements if higher dimensional convolutions are applied to encode the spatial and temporal dimensions of the workspace.

The path integral state action feature vectors are mapped into rewards using a linear reward function for all explored states. Each state has unique features but utilizes the same reward function weights during a planning cycle. Thus, the reward function complexity is directly linked to the expressiveness of the underlying features to acquire a unique solution. A basic set of features describing the AV's actions and the interaction with infrastructure and moving objects can be defined and provide interpretability. However, manually defining nuanced situation-dependent features is difficult. Our approach aims to increase the complexity by parameterizing a large set of basic features. Learning features and temporal decay could also increase representational complexity and address this limitation.

The PSVN is designed to address certain limitations inherent in the path integral deep IRL approaches. Unlike previous methods that were constrained to extract features using one-dimensional convolutions on features, actions, and states of trajectories, this approach can learn spatial patterns from high-resolution images. The PSVN offers a novel stream of information to the planning algorithm, implicitly encoding the environmental dynamics. This can complement explicit motion predictions to mitigate error propagation from the prediction module.

Unlike the work by Zeng et al. (2019) that uses max-margin loss to train a cost volume using human demonstrations directly, this work is trained indirectly in two steps. Instead, this approach relies on a feature-based reward function that is defined and tuned before rendering the target images. Retraining with different demonstrations would result in a different reward function and requires re-rendering. Directly optimizing with max-margin loss is more intuitive but only provides a sparse training signal.

4.3 FUTURE WORK

In recent years, advances have been made in self-supervised representation learning (Gui et al., 2024; Jing and Tian, 2021). In contrast to supervised learning, self-supervised learning does not rely on labeling and instead utilizes naturally occurring supervision of data. This supervision can be used for representation learning. The architecture can be fine-tuned in a subsequent training step to perform the target task. This can, for example, lead to state-of-the-art results on a set of computer vision benchmark tasks (He et al., 2022; Oquab et al., 2024). This thesis takes an initial step in that direction by using a distance metric to the human demonstration as a pseudo label for training the policy attention mechanism. Novel DNN architectures and self-supervised learning techniques can be derived by transforming other geometrical and temporal properties of the state space and corresponding feature space. Combined with different input modalities, this can further improve the network's policy and temporal context encoding capabilities. Pre-trained DNN from the perception stack could also provide inputs to the reward function prediction network.

The proposed IRL architecture in this thesis relies on RNN layers to take into account the sequential context switches. In the preceding years, the focus has shifted away from RNNs for sequence understanding to Transformer architectures (Vaswani et al., 2017). Transformer variants have established themselves as the leading architecture in the fields of computer vision and natural language processing (Li et al., 2023; Lin et al., 2022). It is a promising direction of research to derive transformer-based architectures for stable reward function adaptation. Instead of modeling the reward function transitions using a temporal reward function attention vector it seems promising to de-

rive an approach using temporal regularization (Thodoroff et al., 2018) for situation-dependent reward prediction using a transformer-based architecture.

Planning using a sampling algorithm that evaluates high-resolution actions can be computationally expensive. Especially if multiple processes share the same compute resources, it can be desirable to bias the sampling and thereby reduce the execution time and computational cost. In future work, situation-dependent reward function predictions could be utilized to learn a driving policy using RL algorithms. This driving policy could bias the state-dependent action distributions during sampling.

This thesis does not further investigate the bootstrap capabilities of utilizing the reward shaping function based on PSVN as a new feature for the path integral IRL approaches, which could then be utilized to render enhanced target images for PSVN through a refined reward function. This, in turn, could lead to a better reward shaping function. Through this mechanism, the pixel state value could also be further refined by human demonstrations.

Currently, this work employs two inference architectures: the deep IRL network and PSVN. In the future, these architectures could be merged into a single network. Initially, deep IRL could enhance the target image rendering of PSVN. Once PSVN is trained, this network can be the backbone for the reward function parameter prediction. It carries the advantage of learning about environmental dynamics and operating independently from external predictions. An added advantage is that the BEV inputs to the network are intuitively understandable, unlike the policy set used as input in other architectures.

The BEV representation can also be leveraged for infrastructure-based augmentations, for instance, dropping out the lane center or boundaries to enhance the network's robustness against errors propagated from the perception system. A promising future direction could be to directly utilize latent features from BEV perception as network input. This approach might alleviate the reliance on HD maps during inference while still employing HD maps for target image generation. Ultimately, image-to-image translation aims to understand the driving situation and yield similar outputs as if all relevant information from HD maps were accessible.

As of now the PSVN performs image-to-image translation using a conditional GAN approach by Isola et al. (2017). In recent years, the approaches for image generation have shifted from GANs to diffusion models, which have shown breakthrough performance (Dhariwal and Nichol, 2021). Diffusion models are a class of deep generative models that iteratively add noise to data in a forward process and then learn to remove noise to restore the data (Sohl-Dickstein et al., 2015). Yang et al. (2024) provide a survey of diffusion models and summarize the benefits and limitations in contrast to GANs. While training GANs

is often instable and difficult, they hold the advantage in terms of inference speed. This becomes particularly noticeable when compared to most diffusion methodologies, which typically rely on multiple time-consuming denoising steps. Diffusion models are a promising future direction for PSVN, as their stability would facilitate training at scale. However, for the vehicle deployment in closed-loop, the model must be capable of inferring at a rate significantly faster than five hertz, especially when considering planning cycles at this frequency. Parmar et al. (2024) present an approach for image-to-image translation that combines both methodologies by fine-tuning a one-step diffusion model using adversarial learning methods. This approach allows utilizing pre-trained one-step diffusion models (Sauer et al., 2023) and performs efficient inference during deployment, and therefore could be considered as an underlying methodology for the PSVN.

A promising next step could be to reformulate the prediction task from image-to-image translation to image-to-video generation. For now the PSVN uses a modified U-Net (Ronneberger et al., 2015) architecture that produces a single output image. In this setup, the channel dimension is used to represent time. Bar-Tal et al. (2024) propose a diffusion model for video generation, utilizing a space-time U-Net architecture. This architecture allows the authors to generate videos at full frame rates with coherent motion. Fine-tuning such pre-trained models for PSV generation hold the potential to significantly improve the temporal consistency of the predictions.

Incorporating new information sources within the modular architecture enhances system dependability by facilitating hybrid operational modes. The PSVN can be deployed alongside explicit collision checks using the external prediction module. Leveraging the external prediction module for shorter prediction horizons can be a viable strategy. When deep IRL selects inappropriate behavior, expert-tuned parameters can be a backup for driving policy selection, ensuring a safer and more reliable operation.

4.4 CONCLUSION

The scalability of automated driving systems remains a major concern on the way toward full autonomy. Fully self-driving vehicles are tasked with generalizing in unforeseen driving situations while aligning with passengers' expectations. This thesis combines model predictive planning and deep learning in a hybrid approach, harnessing the advantages of reliability and interpretability while continuously refining prediction and planning through feedback from human driving demonstrations and simulated experiences to generalize across diverse situations. This thesis addresses prevailing challenges in RL and IRL related to search in large high-dimensional state-action spaces by formulating efficient offline learning methodologies using

model predictive planning. The approaches concentrate on predicting situation-dependent reward function weights and PSVs for a MPP to address challenging interactive situations in urban driving environments. Overall, the results demonstrate important steps towards the scalability of automated driving by learning, which is difficult to model.

A

APPENDIX

A.1 TOWARDS HYBRID AUTOMATED DRIVING: FROM DIRECT IMITATION LEARNING TO AFFORDANCE LEARNING

Sascha Rosbach, Antonia Breuer, Simon Barthel, Frederik Kanning, and Silviu Homoceanu
AAET-Automatisiertes und Vernetztes Fahren, Braunschweig, Germany, February 2019.

Abstract

Current state-of-the-art autonomous driving systems adopt the mediated perception approach (Chen et al., 2015), in which the driving decision is made based on an understanding of the scene, derived from the current and past sensor inputs. This scene interpretation includes the detection and recognition of surrounding objects, such as cars and pedestrians, as well as the location of the ego-vehicle in the scene with reference to a map.

Next to the mediated perception approach, so called end-to-end automated driving has been proposed within recent years (Bojarski et al., 2016). These purely data driven automated driving approaches directly imitate human driving behavior with the help of neural networks. The neural networks aim to implicitly learn relevant features which are required to predict vehicle control outputs, such as the steering wheel angle and acceleration.

In this work, we analyze the affordance learning approach presented by Chen et al. (2015). Affordance learning can be seen as a simplistic hybrid approach, adopting features of both the mediated perception as well as the end-to-end driving approaches. The end-to-end perception system estimates the distance to the lane center, the curvature of the lane, and the distance to the preceding vehicle. Based on this simplistic environmental representation, a classical model-based controller is used to compute lateral and longitudinal vehicle controls

Our contribution is twofold. First, we propose a framework for behavior testing of lateral control on real-world image data by utilization of homographic transformations. The introduced playback simulator for behavior testing allows a fast evaluation of the system behavior

once the deep learning models achieved convergence within training. A failure to perform dedicated playback scenarios disqualifies a model from actual real-world testing in the vehicle. We observed that training convergence, validation and testing on labeled datasets are not a satisfactory indicator for model selection. We propose to focus on the time to failure within the playback simulators which enables an empirical analysis of the lateral vehicle control stability. As a result, unlabeled data can be used to assess the performance of the affordance estimators with particular emphasis on the influence of compounding prediction errors. Second, we introduce a point-cloud feature representation that enables convolutional neural networks to distinguish between static and dynamic objects without the need of recurrent architectures.

Copyright notice

Towards Hybrid Automated Driving: From Direct Imitation Learning to Affordance Learning

Sascha Rosbach[1], Antonia Breuer [1], Simon Barthel[2], Frederik Kanning[2], Silviu Homoceanu[1]

[1]Volkswagen AG, {sascha.rosbach, antonia.breuer, silviu.homoceanu}@volkswagen.de
[2]Ameno GmbH, {simon.barthel, frederik.kanning}@ameno.de

Abstract

Current state-of-the-art autonomous driving systems adopt the *mediated perception* approach [1], in which the driving decision is made based on an understanding of the scene, derived from the current and past sensor inputs. This scene interpretation includes the detection and recognition of surrounding objects, such as cars and pedestrians, as well as the location of the ego-vehicle in the scene with reference to a map.

Next to the *mediated perception* approach, so called *end-to-end* automated driving has been proposed within recent years [2]. These purely data driven automated driving approaches directly imitate human driving behavior with the help of neural networks. The neural networks aim to implicitly learn relevant features which are required to predict vehicle control outputs, such as the steering wheel angle and acceleration.

In this work, we analyze the *affordance learning* approach presented by Chen et al. [1]. *Affordance learning* can be seen as a simplistic hybrid approach, adopting features of both the mediated perception as well as the end-to-end driving approaches. The end-to-end perception system estimates the distance to the lane center, the curvature of the lane, and the distance to the preceding vehicle. Based on this simplistic environmental representation, a classical model-based controller is used to compute lateral and longitudinal vehicle controls.

Our contribution is twofold. First, we propose a framework for behavior testing of lateral control on real-world image data by utilization of homographic

transformations. The introduced playback simulator for behavior testing allows a fast evaluation of the system behavior once the deep learning models achieved convergence within training. A failure to perform dedicated playback scenarios disqualifies a model from actual real-world testing in the vehicle. We observed that training convergence, validation and testing on labeled datasets are not a satisfactory indicator for model selection. We propose to focus on the time to failure within the playback simulators which enables an empirical analysis of the lateral vehicle control stability. As a result, unlabeled data can be used to assess the performance of the affordance estimators with particular emphasis on the influence of compounding prediction errors. Second, we introduce a point-cloud feature representation that enables convolutional neural networks to distinguish between static and dynamic objects without the need of recurrent architectures.

1 Introduction

Advanced longitudinal cruise control and lateral steering pilots have great potential of automating manual highway drives and thereby decreasing traffic accidents caused by the lack of drivers attention. We aim for a lateral and longitudinal vehicle control system at SAE Level 2 [3] with a modular and low-complexity system design. Specifically, we require system introspection for humans while using deep learning to be able to handle drastically changing road conditions.

Within the 90s, behavior cloning approaches based on end-to-end neural networks were analyzed within the ALVINN (Autonomous Land Vehicle In a Neural Network) project. The neural vision enabled the ALVINN prototype vehicle to drive, with individually trained neural networks, in a variety of situations such as single-lane dirt roads, single-lane paved roads and two lane highways [4]. By now, one is able to clone the behavior of human driving on very diverse roads by training a single deep neural network [1]. Although, the trainability and simplicity of end-to-end approaches is astonishing, behavior cloning strongly depends on the training dataset behavior. As such, the focus of engineering shifts towards dataset diversification in order to reduce biases of arbitrary image features, picked up by the network. In addition, behavior cloning of expert human drivers lacks recovering behavior, which is necessary once false vehicle control predictions bring the system off the optimal path. As a consequence, input space transformations are often used to directly influence the system behavior. Compared to end-to-end

behavior cloning, we reduce the learning task to a state estimation and use model-based feedback control to stabilize the steering on the lane center. The strict separation of behavior and state estimation allows the dynamics of the system to be independent of the driving demonstrations, while being interpretable by humans. For lateral vehicle control, we estimate a distance-based target vector from image data and use feed-back control to steer the vehicle back to the center lane. Our longitudinal system regresses the distance towards the preceding vehicle and controls the speed-dependent distance. In this work, we extend the hybrid solution, called *affordance learning*, first introduced by Chen et al. [1]. Our evaluation based on a perspective transformation allows testing on unlabeled data and is our deep learning model selection routine for real-world vehicle testing.

This publication is structured as follows: In Section 2 we discuss work related to our approach. In the following Sections 3.1 and 3.3, the lateral and longitudinal controller are introduced. Here, we give a brief system overview followed by an introspection into our approach. The conducted experiments, the labeling procedure, and the evaluation are discussed in Section 4. At the end of this work we give an overview and an outlook into possible future work in Section 5.

2 Related Work

Pomerleau et al. [5] studied end-to-end neural network driving functions for lateral vehicle control. Their experiments were conducted in the NAVLAB autonomous navigation test vehicle in the context of the ALVINN (Autonomous Land Vehicle In a Neural Network) project. The end-to-end approach offers an intuitive approach for sensor fusion of video data and laser range finders. Further, the neural network generates salient feature representations for specific driving conditions. Similar driving performance can be achieved by traditional image and pattern recognition techniques. Here, a priori appearance models are exploited, but they often fail to generalize to drastically changing road appearances. Additionally, extensive manual tuning must be performed. The approach itself is very data hungry and bound to a consistent vehicle sensor setup during data collection. The team developed a simulator for road data generation and trained their network partially on real-world and simulated images. This enhanced training-set diversity and reduced the risk of catastrophic forgetting. They emphasize

that driving demonstrations need to include recovering behavior of false driving predictions.

Pomerleau et al. [6] continued research on the RALPH (Rapidly Adapting Lateral Position Handler). They applied a connectivist approach for the lateral offset relative to the lane center. Learned controllers such as RAPLH have several shortcomings, such as the need to be retrained to new road topologies. Secondly, similar to their previous work, this system requires driver demonstrations for optimal driving and recovering behavior.

Jochem et al. [7] extended the capabilities of the NAVLAB vehicle by tactical driving maneuvers such as lane changes and intersection navigation. They transform the image to increase the data diversity and address the recovery problems of direct imitation approaches mentioned above. Domain specific neural networks are trained for different driving situations, such as two-lane and one-lane driving. Which of the networks is used is based on a rule-based evaluation of the current driving situation. In this approach, the modelling efforts of the mediated perception approach are traded against heavy data engineering.

With their DAVE-2 system, Nvidia [2] trains a lateral controller in an end-to-end fashion with a single neural network. Although their system does not require training of multiple domain specific networks, it heavily relies on data augmentation. They try to automate the data augmentation process by using three cameras mounted behind the wind shield. During training, they use images of each of the cameras and augment the applied steering wheel angle depending on the camera's placement behind the wind shield.

Toromanoff et al. [8] present a simulator based on perspective transformations that provides metrics for lateral end-to-end evaluation. Once the neural net has beaten the simulator based on real-world videos, the final model is capable of driving more than 99% of the situations on real roads autonomously.

Chen et al. [1] decomposed the driving tasks into machine learning based affordance prediction and learned control which can be seen as a simplistic hybrid approach. The end-to-end perception system estimates the distance to the lane center, the curvature of the lane, and the distance to the preceding vehicle. Based on this simplistic environmental representation, a classical model-based controller is used to compute lateral and longitudinal vehicle controls.

This work expands the work of Chen et al. by describing the implementation and labeling process in detail. We analyze the performance of our implementation directly in the vehicle, introduce a behavior testing framework and add a critical evaluation of the more theoretic presentation done by Chen et al..

3 System Architecture

The system architecture proposed in the following section focuses on the prediction of driving affordance and vehicle state instead of the control behavior. Simplifying the neural network in this way, allows us to strongly reduce network complexity, and introduces a higher degree of introspection. The target affordance signals and states of our networks itself support an intuitive evaluation based on standard machine learning metrics. The vehicle states predicted by our network are subsequently used as an input for a simple controller.

3.1 Lateral Control

Our desired driving style for lateral vehicle control is characterized by a close maintenance of proximity to the lane center. Further, the control model must be able to compute steering wheel angles based on non-sequential data from current sensor input. However, maintaining the position on the lane center is not trivial and requires context of the vehicles movement vector. A standard lane correction maneuver is characterized by the following elements:

1. Due to a slight error of the applied steering wheel angle, the vehicle diverges from the lane center.
2. The driver steers back towards the lane center.
3. Once the vehicle vector approaches the lane center again, the driver may steer straight.
4. Before reaching the lane center, the driver must steer to the opposite direction in order to realign the vehicle vector with the lane, and not overshoot the lane center.

In particular, note that single camera image in situation (1) and situation (3) can be extremely similar - the distance to the lane center may even be equal. Only the context that in situation (1) the vehicle vector diverges from the lane center as opposed to situation (3) makes the decision of steering towards the lane center valid. This circumstance makes an end-to-end system operating on non-sequential data difficult. Therefore, we want to predict vehicle states that can be derived from a single image and use a sequence of those states to model vehicle behaviour.

The vehicle state that we want to predict consist of two attributes: (1) The expected divergence from the lane center Δ_c and (2) the curvature of the road κ. In an hypothetical error-free scenario with no disrupting effects, κ alone would be sufficient to implement a perfect lane following assistant. This is due to the fact that κ can be directly correlated to a wheel angle α_κ that has to be applied in order to follow the lane. However, due to prediction inaccuracies, input as well as output latencies, and further physical disturbances, the vehicle will diverge from the lane center with no chance of recovery.

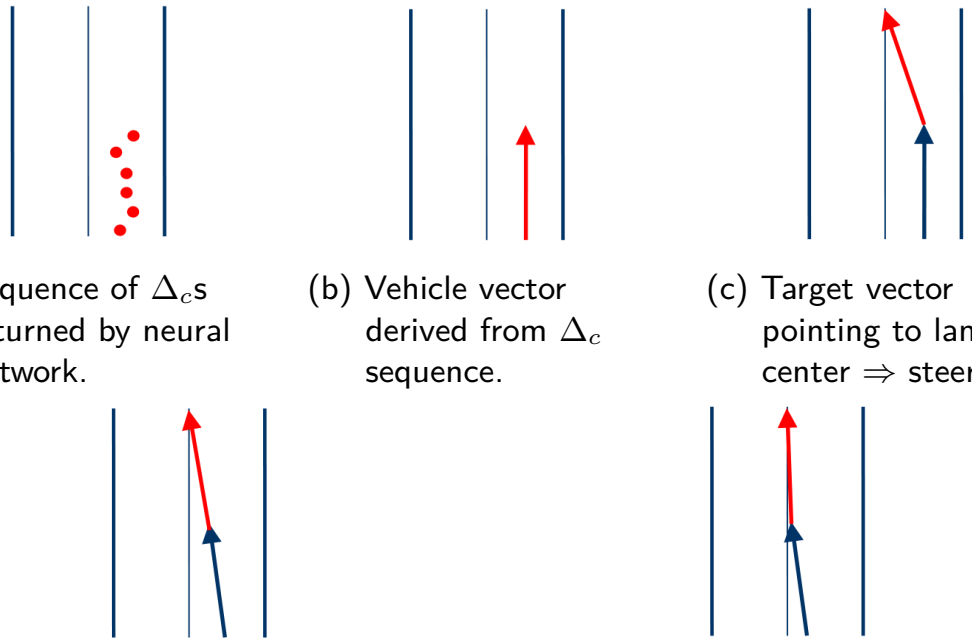
In this case, we explicitly do *not* want to predict a modified wheel angle with an applied recovery maneuver, but continue to predict the original curvature κ. The purpose of κ is therefore the provision of a wheel angle baseline that follows the lane center under perfect conditions.

To recover from divergence from the lane center, we use Δ_c values as input for a simple controller that generates a recovery wheel angle. Since we do not consider the lateral controller as the main contribution of this paper, we will only give a coarse overview of its functionality displayed in Figure 1.

The controller $C : \mathbb{R}^n \to \mathbb{R}$ receives a sequence of divergences from the lane center Δ_c^n which is then used to determine a vehicle vector using least-squares method. Then, a target vector is calculated that points back to the lane center within t_r seconds (tuning parameter). The recovery wheel angle is then calculated by the difference between vehicle angle and target angle. Figures 1c to 1e visualize the recovery maneuver, including a counter steering maneuver to prevent overshooting the lane center.

The final wheel angle applied to the vehicle is then received by adding up intermediate results as described in Equation 1.

$$\alpha_{final} = \alpha_\kappa + C(\Delta_c^n) \tag{1}$$



(a) Sequence of Δ_cs returned by neural network.

(b) Vehicle vector derived from Δ_c sequence.

(c) Target vector pointing to lane center $\Rightarrow$ steer left.

(d) Vehicle aligned with target vector. Pointing to the lane center $\Rightarrow$ steer straight.

(e) Vehicle near lane center. Avoid overshooting by steering right as implied by target vector.

Figure 1: Example of a recovery maneuver using a sequence of Δ_cs.

(a) Sample image from CAM2 without perspective transformation

(b) Image (a) shifted by 0.5m using perspective transformation.

Figure 2: Example an applied perspective transformation using a homography.

3.2 Training details

The input for the neural network origins from two cameras CAM1 and CAM2. CAM1 has a 60° non-distorted lens which is located behind the rear mirror and it sends 20 images per second in a high resolution (1920×1280). CAM2 has a 190° fisheye distortion lens which is located above the shock absorber sending 20 images per second in lower resolution (1280×800).

Due to the high view angle and the fisheye distortion lens, the view distance of CAM2 is very limited. Thus, vehicles that are farther away than 20m-30m away are not perceivable. On the other hand the sight at close proximity to the vehicle is exceptionally well. In contrast to CAM2, CAM1 has opposing attributes: The closest point that can be observed by the camera is about 5m in front of the vehicle but objects further away are well perceivable.

For both cameras a homography H_C was calibrated in order to augment produced images with a perspective transformation. The homography is used to map an image from pixel coordinates into vehicle coordinates where arbitrary translations and rotations can be applied in vehicle coordinates using an affine transformation matrix A. Afterwards, the transformed image can be mapped back to the original pixel space using the inverse homography H_C^{-1}.

In short, the augmentation of an image frame is applied by performing a perspective transformation using the transformation matrix $(H_C A)H_C^{-1}$. A sample of an applied perspective transformation using the described homography can be seen in Figure 2.

Additionally, the distances to the left and right boundary of the current lane were annotated as described in Section 4. These annotations were used to determine the value for the distance to the lane center Δ_c for the training process. For κ, note that an exact value of κ is not needed but only a value that correlates to κ. We therefore used the correlating wheel

angle α_κ here. In order to filter correction maneuvers performed by the driver we take the median wheel angle within the next second.

The network shown in Figure 3 was trained with training example batches that were randomly picked among the training set. Although the training set already contains a diversity of distances to the lane center, for each training example a random perspective transformation was applied shifting the lateral offset by a uniform random number between -0.25m and $+$0.25m and a yaw angle between -2.5° to $+$2.5°. These augmentation was consistently applied to both camera images and the applied lateral shift was added to the Δ_c value in the recording. The image size of both cameras was scaled down to 300$\times$150 pixel grayscale images.

The network architecture defines two independent convolutional stacks which are flattened and concatenated. After two dense layers the network then splits up to two individuals dense branches to allow both train targets to store individual weights. The train targets were trained interleaved in batches of 100 for the first target and 100 batches for the second target. The train set consists of a rather small amount of 85,000 frames which approximately corresponds to a drive of 70 minutes.

3.3 Longitudinal Control

The purpose of the longitudinal controller is the regulation of the vehicles acceleration, such that it will automatically accelerate up to a certain speed limit and decelerate when obstacles are detected. Analogous to the lateral controlling problem, it is also not possible to determine a suitable vehicle speed based on non-sequential data. Therefore, we employ a similar strategy to the longitudinal control and predict a state that is perceivable in the context of a single frame, and use a sequence of such states to predict the driving behaviour.

In Figure 4 an overview of the longitudinal control process of our proposed method is given. The state predicted by the neural network consists of two target features: whether there is a vehicle in the same lane in front of the vehicle, and the distance to this vehicle. As long as our trained network does not detect a vehicle in front of the ego vehicle for M consecutive frames, we use the cruise control to define the applied speed, as pictured on the right side of Figure 4. Otherwise we switch to distance control mode, in which the distance to the vehicle ahead and the corresponding relative speed determine the applied speed. Here, the distance to the front

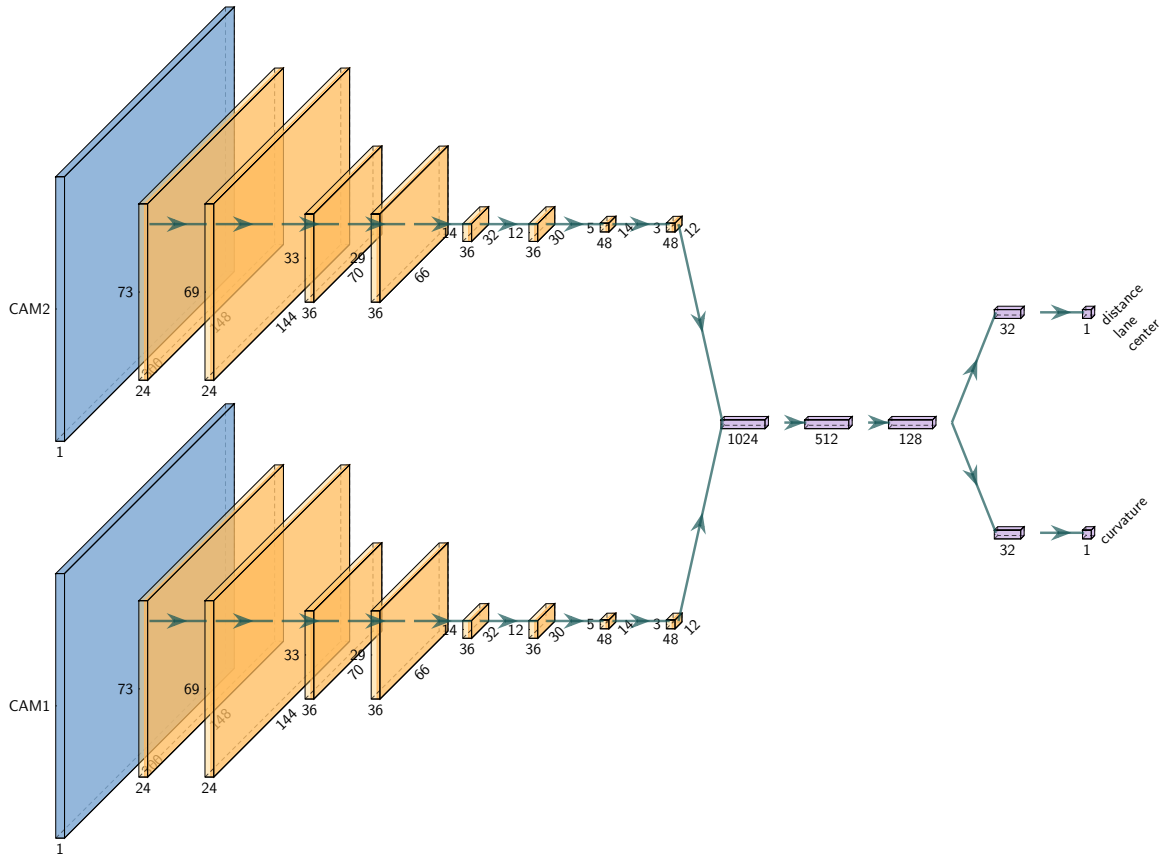


Figure 3: Multi-target neural network architecture for lane center and curvature prediction.

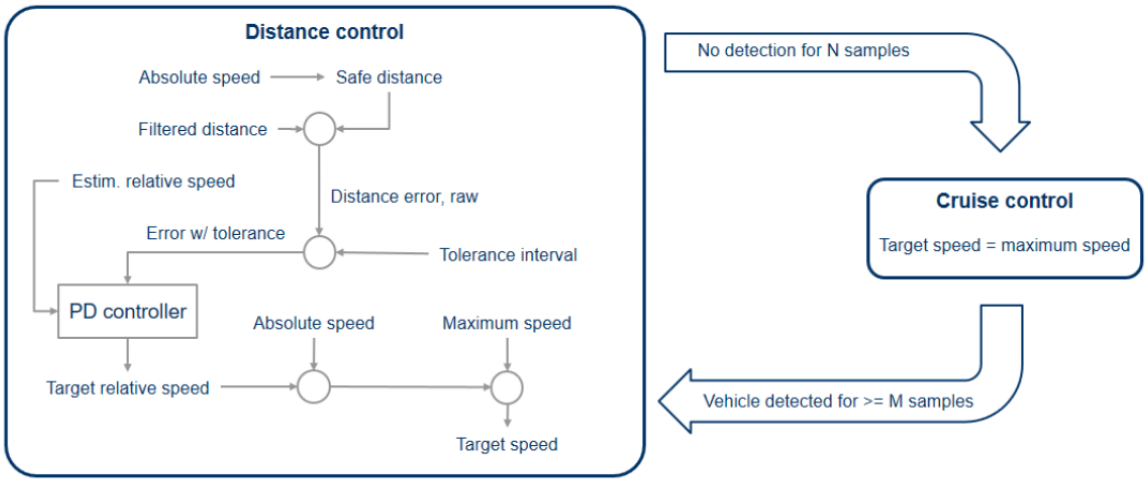


Figure 4: A overview of the longitudinal control process used in the proposed system. If our neural network detects a vehicle in front of the ego vehicle, the cruise control system is used. Otherwise, the distance control procedure visualized on the left is applied based on the detected distance to the front vehicle.

vehicle computed by the neural network is filtered by a Kalman filter and compared to the safe distance, yielding the distance error. We also use the Kalman filter to derive the relative speed from the change of distance. The PD controller applies this distance error, corrected by a tolerance interval, together with the relative speed to compute the target relative speed. The tolerance interval makes the controlled system more robust against prediction errors from the network, it however introduces a slow oscillation to the control loop. In a last step, the target speed of the ego vehicle is computed from this relative speed, the absolute speed, and the maximum speed.

The neural network infers the distance to the vehicle ahead from a LIDAR point cloud, and also determines whether a vehicle is present at all. We preprocess the raw data to make it more suitable for neural network training, and to emphasize relevant characteristics. First, we project the points along two axes to obtain two-dimensional representations with fixed size. Then, to enable the network to discern static and moving objects, we superimpose projections of the three most recent frames using two different methods. Consequently, these preprocessing steps result in a total of four matrices, which are then fed into the first convolutional layers of the network.

To project the point cloud along one axis, we first define a fixed-size grid with finite extent on the other axes. Then, we determine the minimum

value along the projected axis for each grid cell. Here, empty cells are defined as positive infinity. The matrix of minimum values is the result of the projection. Using this method, we compute a bird's eye view of the environment in front of the vehicle with 20m width, 100m length, and 0.2m by 0.25m cell size. We also compute another projection along the longitudinal axis with 20m width, 1.6m height, and 0.2m by 0.2m cell size.

To obtain the final network inputs, we superimpose the projections of the three most recent frames with one out of two methods. The first method simply recomputes the minimum for each cell over the three frames. Therefore static objects, such as lane markings, will appear multiple times and slightly shifted as long as the ego vehicle is moving. By contrast, objects with lower speed relative to the vehicle, such as other traffic participants, will appear less blurred. The second superimposition method aims to stitch together a consistent view of the static environment over the last three frames. To this end, we apply a shift and rotation to each frame to revert the ego vehicle movement. This transformation is inferred from the ego speed and steering angle between frames. As a consequence, static objects appear sharp, while objects with no relative movement are blurred.

In total, we have two different superimposition results along the longitudinal axis, and another two superimposition results of the bird's eye view. The neural network processes each of the resulting matrices in a separate convolutional stack. The output layers of the convolutional stacks are first concatenated and then connected to two separate stacks of dense layers, where one dense stack represents the classification function (i.e., vehicle detection) and the other stack represents the regression function (i.e., distance to vehicle ahead). In Figure 5, we show the full network architecture. We train both target values of the network simultaneously, however we do not apply the loss function simultaneously when no vehicle is present and hence no meaningful distance can be trained.

4 Experiments

In this work, we are focused on experiments that evaluate the real-world applicability of our system architecture. Since our lateral and longitudinal state estimates utilize deep learning, it is important to access the performance on real-world sensor data. The system design allows the evaluation

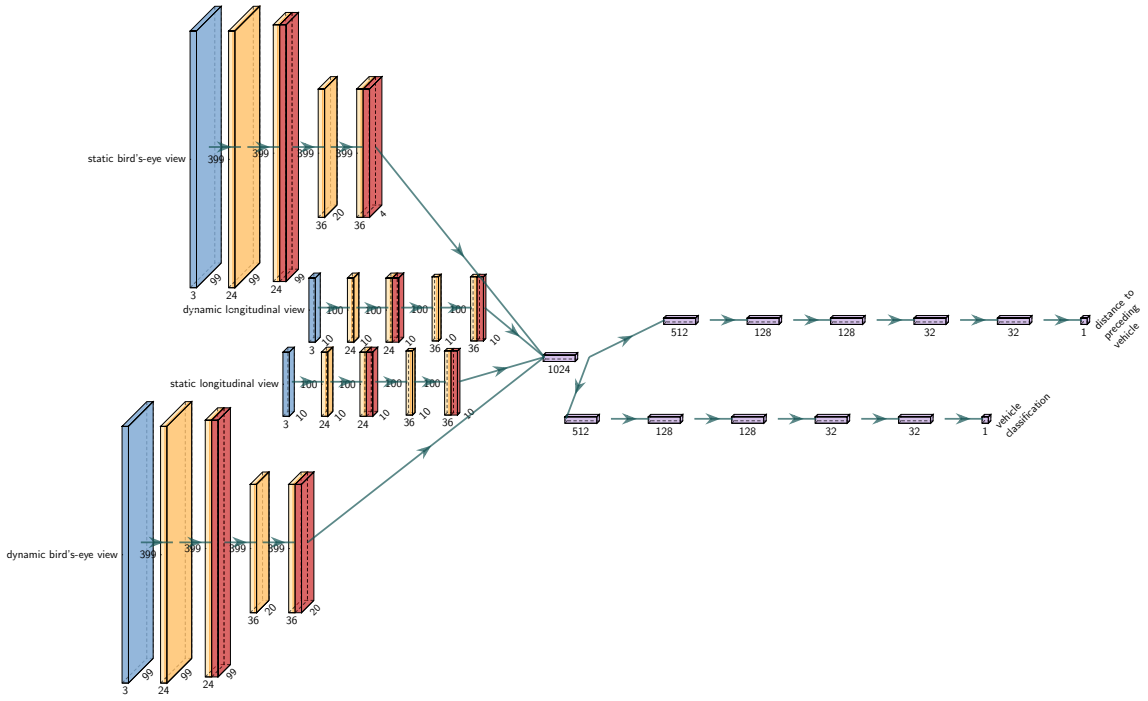


Figure 5: Multi-target architecture for longitudinal affordance signal prediction. Architecture combines regression task for distance estimation to preceding vehicle with the classification of a preceding vehicle presence within the current lane.

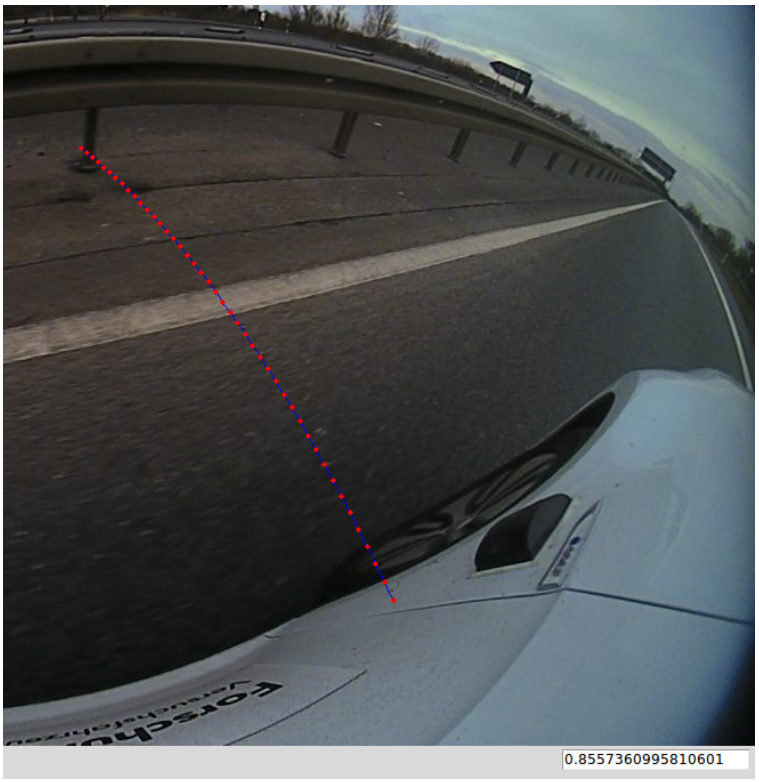


Figure 6: To label the distance to the lane boundary, a visualisation of the distance is projected into the camera image. A human labeller selected the first dot of a *"ruler"* touching the lane marking, thus defining an estimate of the distance to the lane marking in an intuitive way. The projection of the ruler into the image has been hard coded by hand measurements, under the assumption that the camera position is fixed.

of the control state estimates on non-sequential data by standard machine learning metrics. Due to architectural separation of lateral and longitudinal control, it is valid to conduct individual testing on a modular basis. The modular tests as well as the supervised model training require an efficient data labelling procedure which is described in Section 4.1. In order to accelerate the system behavior testing, and preempt unsatisfactory models we introduce a playback simulator which serves as necessary criterion for lateral in-vehicle control testing. The lateral playback simulator is based on a holographic transformation which imitates closed loop behavior on raw sensor data.

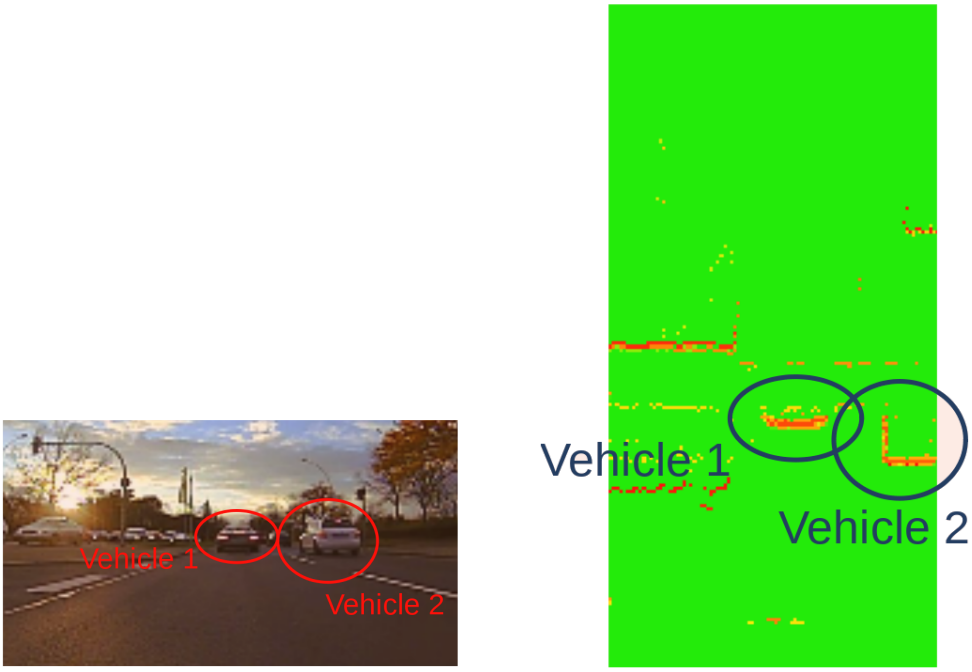
4.1 Data Collection

In the following the data collection and labeling process for both the lateral and longitudinal control part will be explained in detail. The network computing the lateral control, described in Section 3.1, uses the distance to the lane center and the curvature of the lane center as target features. For the training data, the distance to the lane center was derived from the distance to the left lane marking, which is manually labeled. The distance to the left lane marking has been labelled using the tool shown in Figure 6. Here, a ruler-like measurement is projected into the camera image with fixed distances between the ruler points. In the label tool, the labeller then selected the first point touching a lane marking, thus estimating the distance to the lane marking. During the data collection process only lanes with a fixed width have been driven, allowing to infer the distance to the lane center from the labeled distance to the left lane marking. This assumption is only important for the training of the lateral control network. During the inference stage, no assumptions on the properties of the driven lane need to be made, since the network learns the distance to the lane center based on a surround view of the vehicle. To complement the hand-labeled training data, the future curvature of the lane can be directly derived from the recorded yaw and steering wheel angle of the car.

As described in Section 3.3, the information whether or not someone is in the same lane, and the distance to a possible front vehicle is used as target features for the longitudinal control. The first task is a classification task, which can be directly labeled using the front view image. For the latter regression task, the target feature is the distance to the preceding car in the driven lane. The labeling procedure is simplified by the labelling interface shown in Figure 10. Here, the labeller selects the laser scan points, visualized in a bird's eye view, corresponding to the front vehicle using the front camera image as guidance. Given the corresponding laser scan points, the distance can be directly inferred by the average return time of the laser reflections.

4.2 Evaluation

Our system architecture consists of a lateral and longitudinal subsystems that can be evaluated individually by standard machine learning metrics. The distance and curvature estimates might include noisy predictions given drastically changing environment situations such as weather, and road



(a) front view camera image (b) laser scan points

Figure 7: To label the training data for the longitudinal control, the points corresponding to the front vehicle were selected in a bird's eye view projection of the laser scan points, visualized in (b). The camera image of the front view camera, as shown in (a), was provided to the labeler to guide the labeling process.

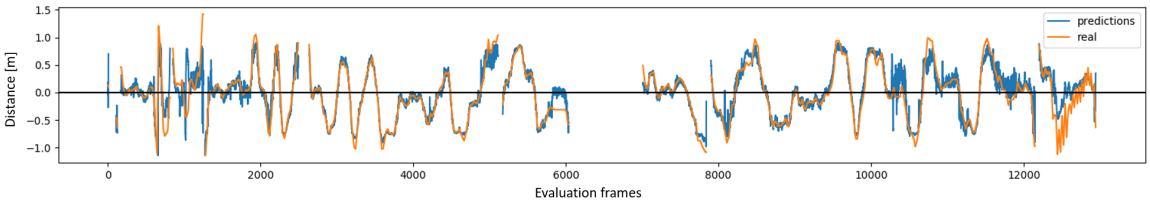


Figure 8: Evaluation of the lateral model on a test recording. The plot shows two graphs. The orange graph shows the annotated values, the blue graph the predicted values. The y axis represents the distance to the lane center of the manual human driving.

conditions. Therefore we propose a lateral playback simulator for semi-closed loop testing on unlabeled image recordings.

4.2.1 Distance Metric Evaluation of Controller Inputs

The separation of our system architecture into state estimates for lateral and longitudinal control allows modular testing of the individual components on dedicated validation datasets.

Lateral Module Evaluation Figure 8 shows a trend of predicted and annotated lane diversions for each frame in a test recording that was recorded on a motorway. The test drive mostly contains annotated diversions from the lane center between -1.0m to +1.0m which were achieved by driving between the left and right boundary of the lane. Most errors occur before a lane change, e.g. at frame 1200, frame 5000 or 8000. At frame 6000 there is a constant error caused by a confusing construction site marking that was misinterpreted as the lane boundary. Also the performance from frame 10,000 drops significantly due to missing lane markings at that section of the street. However, even with completely missing lane markings the tendency is still correct. We are confident that the accuracy can be significantly improved by a loner training on more diverse training data.

Longitudinal Model Evaluation A trend of predicted and annotated distances to the front vehicle can be seen in Figure 9. The trend shows a front vehicle that is increasing distance until frame 1200 before coming to a stop. The ego vehicle then reduces the distance to the front vehicle until the front vehicle re-accelerates at frame 1600. The ego vehicle then follows

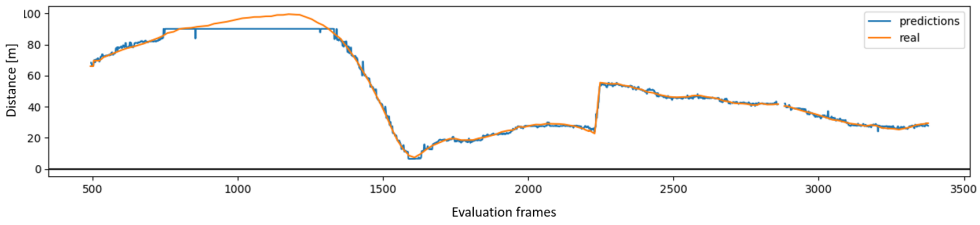


Figure 9: Evaluation of the longitudinal error based on a test recording. The orange graph shows the annotated values, the blue graph the predicted values. The y axis represents the distance to the vehicle in front.

the front vehicle until frame 2300 where it decides to change lanes. From that point on, the preceding vehicle is detected resulting in the jump of the front vehicle distance.

For a majority of time instances, the predicted distance is very close to the annotated vehicle distance. It is however noticeable that the prediction caps at about 85m which is not sufficient for driving velocities above 40–50 km/h. We are confident however, that this flaw can be removed by a careful reapplication of the training on a larger dataset.

4.2.2 Lateral System Behavior Evaluation

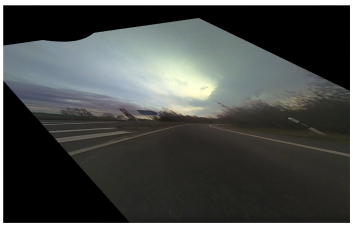
The metric-based evaluation requires a large dataset including labeled distances towards the lane center. Our playback simulator does not require any labels and can be used for extended testing of pre-selected models that performed well on metric tests.

Playback Simulator Even though the affordance perception approach allows an evaluation of deep learning models with traditional machine learning metrics, the margin for failure between a metrics-based test and a closed loop test in a real world scenarios is still large. As an intermediate step, we therefore tested our lateral model and controller in a simulation that manipulates real recordings based on the model/controller outputs.

To achieve this, the simulator maintains a vehicle model that is able to calculate a trajectory based on wheel angle and velocity-pairs. In this context, given a sequence of wheel angle and velocity-pairs, the vehicle model returns a lateral and longitudinal offset as well as a yaw angle for each pair.

(a) Lane following in time step t.

(b) Recovering behavior in time step t+1.



(c) Imminent system failure in time step t+5.

Figure 10: The images (a)-(c) display time steps within a playback of a data recording. While images (a) and (b) still show lane following and recovering behavior, image (c) already displays imminent system failure within the next consecutive time steps.

In the simulator this model is used to calculate a lateral offset between the driven trajectory and the trajectory that was driven by the model. To determine the lateral offset of the model to the recorded trajectory, the velocity values provided to the vehicle model are the recorded velocity values. The wheel angles are the difference between the recorded wheel angle, and the wheel angle returned by the model/controller chain. The lateral offset is then used to apply a perspective transformation on the camera images to simulate the divergence from the recorded trajectory. The perspective shifts applied to the camera input in the simulator can be seen in Figure 10.

This approach allows a number of sanity checks that the model must be able to pass before a real-world test should be performed:

- For normal drives where the driver was following the lane center, the simulator must not diverge from the lane center even when occurring lanes with bad lane markings or high steering angles.
- In drives where the driver was oscillating between the left and right border of the lane, the simulator should correct those maneuvers and not follow the trajectory of the recording.
- When manually setting a lateral offset during a simulation, the simulator must find its way back to the lane center.

Only if these tests pass, a real-world test is sensible.

5 Conclusion

In this paper, we propose an automated driving architecture that utilizes deep convolutional networks to compute affordance signals for lane-centered longitudinal and lateral control. We relate our system architecture towards previous research in deep driving, classical active steering, and cruise control systems. We describe the intuitive and efficient data collection that motivates our practical data-driven control input estimates. The data driven estimates for model-based control include: Firstly, a lateral ConvNet which computes features based on front view camera images and estimates the distance towards the lane center. Secondly, a longitudinal ConvNet which detects preceding vehicles within the lane and estimates the distance based on laser range finder data. Our approach leverages the representation learning benefits of end-to-end deep learning architectures while providing introspection during the adjustment of the model-based control behavior. One of these representation learning benefits is the ability to adapt to diverse driving environments encountered by drastically changing road conditions, and appearance diversity of preceding vehicles during detection. Due to the prior definition of affordance metrics, the system allows a modular testing and introspection layer, which deep driving approaches lack to provide. We perform a modular evaluation of the distance regression tasks based on standard machine learning performance metrics. Since prediction noise and false estimates influence the behavior of the system, traditional machine learning performance metrics on their own do not provide a sufficient validation of the system in real driving conditions. As a result, we introduce a system test of the closed loop perception and model-based control system performance on unlabeled data. Our real-world driving experiments show that our closed loop system test on unlabeled data provides a necessary condition for successful deployment in real-world.

For future work we plan to integrate the controller back into the neural network using a sequence of latent states in the common dense stack. Since the dimensionality is comparably low at that point, a recurrent network can be applied efficiently to a sequence of latent states. An initial training can be performed using the simple controller that was proposed in this paper.

Literatur

[1] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, ser. ICCV '15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 2722–2730. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2015.312

[2] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *CoRR*, vol. abs/1604.07316, 2016. [Online]. Available: http://arxiv.org/abs/1604.07316

[3] Society of Automotive Engineers. (2014, Jan.) SAE J3016: Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems. [Online]. Available: http://standards.sae.org/j1772_201001/

[4] D. A. Pomerleau, "Neural Network Vision for Robot Driving," in *Intelligent Unmanned Ground Vehicles*. Springer, 1997, pp. 53–72. [Online]. Available: https://doi.org/10.1007/978-1-4615-6325-9_4

[5] D. A. . Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Advances in neural information processing systems*, 1989, pp. 305–313. [Online]. Available: https://papers.nips.cc/paper/95-alvinn-an-autonomous-land-vehicle-in-a-neural-network.pdf

[6] D. Pomerleau and T. Jochem, "Rapidly adapting machine vision for automated vehicle steering," *IEEE Expert*, vol. 11, no. 2, pp. 19–27, Apr. 1996. [Online]. Available: https://doi.org/10.1109/64.491277

[7] T. Jochem and D. A. Pomerleau, "Life in the Fast Lane: The Evolution of an Adaptive Vehicle Control System," *AI Magazine*, vol. 17, no. 2, p. 41, 1996. [Online]. Available: https://doi.org/10.1609/aimag.v17i2.1221

[8] M. Toromanoff, E. Wirbel, F. Wilhelm, C. Vejarano, X. Perrotton, and F. Moutarde, "End to End Vehicle Lateral Control Using a Single Fisheye Camera," in *I2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2018)*, Madrid, Spain, Oct. 2018, 7 pages paper accepted at IROS 2018. [Online]. Available: https://hal-mines-paristech.archives-ouvertes.fr/hal-01861697

A.2 DRIVING WITH STYLE: INVERSE REINFORCEMENT LEARNING IN GENERAL-PURPOSE PLANNING FOR AUTOMATED DRIVING

Sascha Rosbach, Vinit James, Simon Großjohann, Silviu Homoceanu, and Stefan Roth
IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, November 2019.

Abstract

Behavior and motion planning play an important role in automated driving. Traditionally, behavior planners instruct local motion planners with predefined behaviors. Due to the high scene complexity in urban environments, unpredictable situations may occur in which behavior planners fail to match predefined behavior templates. Recently, general-purpose planners have been introduced, combining behavior and local motion planning. These general-purpose planners allow behavior-aware motion planning given a single reward function. However, two challenges arise: First, this function has to map a complex feature space into rewards. Second, the reward function has to be manually tuned by an expert. Manually tuning this reward function becomes a tedious task. In this paper, we propose an approach that relies on human driving demonstrations to automatically tune reward functions. This study offers important insights into the driving style optimization of general-purpose planners with maximum entropy inverse reinforcement learning. We evaluate our approach based on the expected value difference between learned and demonstrated policies. Furthermore, we compare the similarity of human driven trajectories with optimal policies of our planner under learned and expert-tuned reward functions. Our experiments show that we are able to learn reward functions exceeding the level of manual expert tuning without prior domain knowledge.

Copyright notice

Driving with Style: Inverse Reinforcement Learning in General-Purpose Planning for Automated Driving

Sascha Rosbach[1,2], Vinit James[1], Simon Großjohann[1], Silviu Homoceanu[1] and Stefan Roth[2]

***Abstract*— Behavior and motion planning play an important role in automated driving. Traditionally, behavior planners instruct local motion planners with predefined behaviors. Due to the high scene complexity in urban environments, unpredictable situations may occur in which behavior planners fail to match predefined behavior templates. Recently, general-purpose planners have been introduced, combining behavior and local motion planning. These general-purpose planners allow behavior-aware motion planning given a single reward function. However, two challenges arise: First, this function has to map a complex feature space into rewards. Second, the reward function has to be manually tuned by an expert. Manually tuning this reward function becomes a tedious task. In this paper, we propose an approach that relies on human driving demonstrations to automatically tune reward functions. This study offers important insights into the driving style optimization of general-purpose planners with maximum entropy inverse reinforcement learning. We evaluate our approach based on the expected value difference between learned and demonstrated policies. Furthermore, we compare the similarity of human driven trajectories with optimal policies of our planner under learned and expert-tuned reward functions. Our experiments show that we are able to learn reward functions exceeding the level of manual expert tuning without prior domain knowledge.**

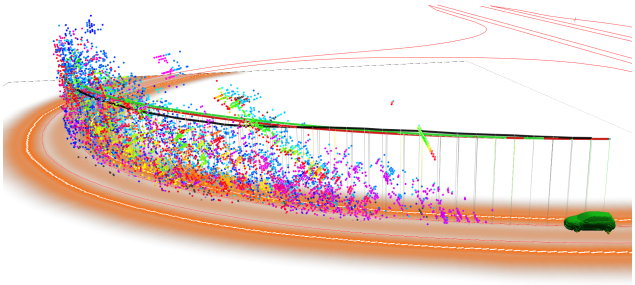


Fig. 1: This figure illustrates our general-purpose planner for automated driving. The color coding of the visualized state space indicates the state-action values. The z-axis corresponds to the velocity, while the groundplane depicts a subset of spatial features such as distance transformed lane centers and road boundaries. There are three color coded policies, black denotes the optimal policy of the planner, red the odometry of a human demonstration, and green the projection of the demonstration into the state space.

I. Introduction

The trajectory planner in highly automated vehicles must be able to generate comfortable and safe trajectories in all traffic situations. As a consequence, the planner must avoid collisions, monitor traffic rules, and minimize the risk of unexpected events. General-purpose planners fulfill these functional requirements by optimization of a complex reward function. However, the specification of such a reward function involves tedious manual tuning by motion planning experts. Tuning is especially tedious if the reward function has to encode a humanlike driving style for all possible scenarios. In this paper, we are concerned with the automation of the reward function tuning process.

Unlike a strict hierarchical planning system, our planner integrates behavior and local motion planning. The integration is achieved by a high-resolution sampling with continuous actions [1]. Our planner, shown in Fig. 1, derives its actions from a vehicle transition model. This model is used to integrate features of the environment, which are then used to formulate a linear reward function. During every planning cycle of a model predictive control (MPC), the planning algorithm generates a graph representation of the high-dimensional state space. At the end of every planning cycle, the algorithm yields a large set of driving policies with multiple implicit behaviors, e.g., lane following, lane changes, swerving and emergency stops. The final driving policy has the highest reward value while satisfying model-based constraints. The reward function, therefore, influences the driving style of all policies without compromising safety.

Human driving demonstrations enable the application of inverse reinforcement learning (IRL) for finding the underlying reward functions, i.e., a linear combination of the reward weights. In this work, we utilize this methodology to automate the reward function tuning of our planner. Due to the planner's exploration of a large set of actions, we are able to project demonstrated actions into our graph representation. Thereby, the demonstrations and associated features are efficiently captured. As a result, the learning algorithm enables the imitation of the demonstrated driving style. Most related work in IRL utilizes the state visitation frequency to calculate the gradient in *maximum entropy* IRL. However, the calculation of the state visitation is generally intractable in this high-dimensional state space. We utilize our graph representation to approximate the required empirical feature expectations to allow *maximum entropy* IRL.

[1]The authors are with the Volkswagen AG, 38440 Wolfsburg, Germany {sascha.rosbach, vinit.james, simon.grossjohann, silviu.homoceanu}@volkswagen.de

[2]The authors are with the Visual Inference Lab, Department of Computer Science, Technische Universität Darmstadt, 64289 Darmstadt stefan.roth@visinf.tu-darmstadt.de

The main contributions of this paper are threefold: First, we formulate an IRL approach which integrates maximum entropy IRL with a model-predictive general-purpose planner. This formulation allows us to encode a humanlike driving style in a linear reward function. Second, we demonstrate the superiority of our automated reward learning approach over manual reward tuning by motion planning experts. We draw this conclusion on the basis of comparisons over various performance metrics as well as real world tests. Third, our automated tuning process allows us to generate multiple reward functions that are optimized for different driving environments and thereby extends the generalization capability of a linear reward function.

II. Related Work

The majority of active planning systems for automated driving are built on the mediated perception paradigm. This paradigm divides the automated driving architecture into sub-systems to create abstractions from raw sensory data. The general architecture includes a perception module, a system to predict the intention of other traffic participants, and a trajectory planning system. The planning system is usually decomposed in a hierarchical structure to reduce the complexity of the decision making task [2]–[4]. On a strategic level, a route planning module provides navigational information. On a tactic and behavioral level, a behavioral planner derives the maneuver, e.g., lane-change, lane following, and emergency-breaking [5]. On an operational level, a local motion planner provides a reference trajectory for feedback control [6]. However, these hierarchical planning architectures suffer from uncertain behavior planning due to insufficient knowledge about motion constraints. As a result, a maneuver may either be infeasible due to over-estimation or discarded due to under-estimation of the vehicle capabilities. Furthermore, behavior planning becomes difficult in complex and unforeseen driving situations in which the behavior fails to match predefined admissibility templates. Starting with the work of McNaughton, attention has been drawn to parallel real-time planning [1]. This approach enables sampling of a large set of actions that respect kinematic constraints. Thereby a sequence of sampled actions can represent complex maneuvers. This kind of general-purpose planner uses a single reward function, which can be adapted online by a behavioral planner without the drawbacks of a hierarchical approach. However, it is tedious to manually specify and maintain a set of tuned reward functions. The process required to compose and tune reward functions is outside the scope of McNaughton's work. We adopt the general-purpose planning paradigm in our approach and focus on the required tuning process.

Reward functions play an essential role in general-purpose planning. The rewards encode the driving style and influence the policy selection. Recently, literature has been published on the feature space of these reward functions. Heinrich et al. [7] propose a model-based strategy to include sensor coverage of the relevant environment to optimize the vehicle's future pose. Gu et al. [8] derive tactical features from the large set of sampled policies. So far, however, there has been little discussion about the automated reward function tuning process of a general-purpose planner.

Previous work has investigated the utilization of machine learning in hierarchical planning approaches to predict tactical behaviors [5]. Aside of behavior prediction, a large and growing body of literature focuses on finding rewards for behavior planning in hierarchical architectures [9], rewards associated with spatial traversability [10], and rewards for single-task behavior optimization of local trajectory planners [11]. The IRL approach plays an import role in finding the underlying reward function of human demonstrations for trajectory planning [12]. Similar to this work, several studies have investigated IRL in high-dimensional planning problems with long planning horizons. Shiarlis et al. [13] demonstrate maximum margin IRL within a randomly-exploring random tree (RRT*). Byravan et al. [14] focus on a graph-based planning representation for robot manipulation, similar to our planning problem formulation. Compared to previous work in IRL, our approach integrates IRL directly into the graph construction and allows the application of *maximum entropy* IRL for long planning horizons *without* increasing the planning cycle time.

Compared to supervised learning approaches such as direct imitation and reward learning, reinforcement learning solves the planning problem through learning by experience and interaction with the environment. Benefits of reinforcement learning are especially notable in the presence of many traffic participants. Intention prediction of other traffic participants can be directly learned by multi-agent interactions. Learned behavior may include complex negotiation of multiple driving participants [15]. Much of the current literature focuses on simulated driving experience and faces challenges moving from simulation to real-world driving, especially in urban scenarios. Another challenge includes the formulation of functional safety within this approach. Shalev-Shwartz et al. [16] describe a safe reinforcement learning approach that uses a hierarchical options graph for decision making where each node within the graph implements a policy function. In this approach, driving policies are learned whereas trajectory planning is not learned and bound by hard constraints.

Most of the current work in IRL utilizes the *maximum entropy* principle by Ziebart et al. [17] that allows training of a probabilistic model by gradient descent. The gradient calculation depends on the state visitation frequency, which is often calculated by an algorithm similar to backward value iteration in reinforcement learning. Due to the curse of dimensionality, this algorithm is intractable for driving style optimization in high-dimensional continuous spaces. Our work extends previous work by embedding IRL into a general-purpose planner with an efficient graph representation of the state space. The design of the planner effectively enables the driving style imitation without a learning task decomposition. As a result, we utilize the benefits of a model-based general-purpose planner and reward learning to achieve nuanced driving style adaptations.

III. Preliminaries

The interaction of the agent with the environment is often formulated as a Markov Decision Process (MDP) consisting of a 5-tuple $\{\mathcal{S}, \mathcal{A}, T, R, \gamma\}$, where $\mathcal{S}$ denotes the set of states, and $\mathcal{A}$ describes the set of actions. A continuous action a is integrated over time t using the transition function $T(s, a, s')$ for $s, s' \in \mathcal{S}, a \in \mathcal{A}$. The reward function R assigns a reward to every action $\mathcal{A}$ in state $\mathcal{S}$. The reward is discounted by γ over time t.

In this work, a model of the environment M returns a feature vector $\boldsymbol{f}$ and the resultant state s' after the execution of action a in state s. The reward function R is given by a linear combination of K feature values f_i with weights θ_i such that $\forall (s, a) \in \mathcal{S} \times \mathcal{A} : R(s, a) = \sum_{i \in K} -\theta_i f_i(s, a)$. A policy π is a sequence of time-continuous transitions T. The feature path integral f_i^π for a policy π is defined by $f_i^\pi = -\int_t \gamma_t f_i(s_t, a_t)\, dt$. The path integral is approximated by the iterative execution of sampled state-action sets $\mathcal{A}_s$ in the environment model M. The value V^π of a policy π is the integral of discounted rewards during continuous transitions $V^\pi = \int_t \gamma_t R(s_t, a_t)\, dt$. An optimal policy π^* has maximum cumulative value, where $\pi^* = \arg\max_\pi V^\pi$. A human demonstration ζ is given by a vehicle odometry record. A projection of the odometry record ζ into the state-action space allows us to formulate a demonstration as policy π^D. For every planning cycle, we consider a set of demonstrations Π^D, which are geometrically close to the odometry record ζ. The planning algorithm returns a finite set of policies Π with different driving characteristics. The final driving policy π^S is selected and satisfies model-based constraints.

IV. Methodology

The planning system in Fig. 2 uses MPC to address continuous updates of the environment model. A periodic trigger initiates the perception system for which the planner returns a selected policy. In the following, we give an overview of the general-purpose planner. We use the nomenclature of reinforcement learning to underline the influence of reward learning in the context of search-based planning. Furthermore, we propose a *path integral maximum entropy* IRL formulation for high-dimensional reward optimization.

A. General-Purpose Planner for Automated Driving

Our planning algorithm for automated driving in all driving situations is based on [7], [18]. The planner is initialized at state s_0, either by the environment model or in subsequent plans by the previous policy, and designed to perform an exhaustive forward search of actions to yield a set of policies Π. The set Π implicitly includes multiple behaviors, e.g., lane following, lane changes, swerving, and emergency stops [1]. Fig. 2 visualizes the functional flow of the planning architecture during inference and training.

Algo. 1 formally describes our search-based planning approach. The planner generates trajectories for a specified planning horizon H. Trajectories for the time horizon H are iteratively constructed by planning for discrete transition lengths. The planner uses the parallelism of a graphics processing unit (GPU) to sample for all states $s \in \mathcal{S}_t$ a discrete number of continuous actions $\mathcal{A}_s$, composed of accelerations and wheel angles. The sampling distribution for each state is based on feasible vehicle dynamics. The actions itself are represented by time-continuous polynomial functions, where order and coefficients are derived from actor-friendly continuity constraints. This results in longitudinal actions described by velocity profiles up to fifth order, and lateral actions described by wheel angle profiles up to third order.

Algorithm 1: General-Purpose Planner

Input: planning horizon H, model M, reward function R, reward discount factor γ
Output: policies Π, planning solution π^S

function $SearchAlgorithm(H, M, R, \gamma)$
  for t *in* H **do**
    $\mathcal{S}_t \leftarrow$ get set of states
    forall $s \in \mathcal{S}_t$ **do**
      $\mathcal{A}_s \leftarrow$ sample set of actions
      forall $a \in \mathcal{A}_s$ **do**
        execute action a in $M(s, a)$
        observe resultant state s'
        observe resultant transition T
        observe resultant features $\boldsymbol{f}(s, a)$
        construct labels $\boldsymbol{c}(T)$
        $R(s, a) \leftarrow \sum_{i \in K} -\theta_i f_i(s, a)$
        $V(s') \leftarrow V(s) + \gamma_t R(s, a)$
    $\mathcal{S}_{t+1} \leftarrow$ prune $\mathcal{S}_t$
  $\Pi \leftarrow$ get policies in $\mathcal{S}, \mathcal{A}$
  $\pi^S \leftarrow$ select model-based from Π

The search algorithm calls the model of the environment M for all states $s \in \mathcal{S}_t$ to observe the resultant state s', transition T, and features $\boldsymbol{f}$ for each state-action tuple. The feature vector $\boldsymbol{f}$ is generated by integrating the time-continuous actions in the environment model. A labelling function assigns categorical labels to transitions, e.g., a label associated with collision. A pruning operation limits the set of states $\mathcal{S}_{t+1}$ for the next transition step $t + 1 \in H$. Pruning is performed based on the value $V(s)$, label c, and properties of the reachable set $\mathcal{S}_t$ to terminate redundant states with low value $V(s)$. This operation is required, first to limit the exponential growth of the state space, and second to yield a policy set Π with maximum behavior diversity. The algorithm is similar to parallel breadth first search and forward value iteration. The final driving policy π^S is selected based on the policy value $V(\pi)$ and model-based constraints.

B. Inverse Reinforcement Learning

The driving style of a general-purpose motion planner is directly influenced by the reward function weights $\boldsymbol{\theta}$. The goal of IRL is to find these reward function weights $\boldsymbol{\theta}$ that enable the optimal policy π^* to be at least as good

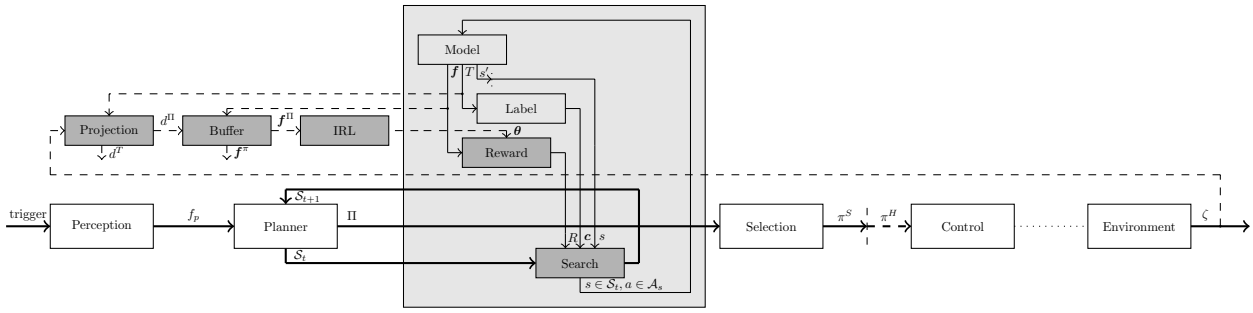


Fig. 2: Functional flow block diagram: The left input of a block corresponds to the output of the previous function. The inputs on top of the blocks denote intermediate outputs of previous functions. A thick solid line indicates the main flow from the environment perception f_p to the driven trajectory ζ. The vehicle control architecture is outside the scope of this work. In this work, we focus on the dark grey blocks of the architecture that influence the reward learning. Dashed connections between the blocks indicate the information flow during the training procedure. During data collection, we record the environment as well as the odometry ζ of the hidden driving policy of a human π^H.

as the demonstrated policy π^D, i.e., $V(\pi^*) \geq V(\pi^D)$. Thereby, the planner indirectly imitates the behavior of a demonstration [19]. However, learning a reward function given an optimal policy is ambiguous since many reward functions may lead to the same optimal policy [20]. Early work in reward learning for A* planning and dynamic programming approaches utilized structured maximum-margin classification [21], yet this approach suffers from drawbacks in the case of imperfect demonstrations [17]. Over the past decade, most research in IRL has focused on maximizing the entropy of the distribution on state-actions under the learned policy, which is known as *maximum entropy* IRL. This problem formulation solves the ambiguity of imperfect demonstrations by recovering a distribution over potential reward functions while avoiding any bias [17]. Ziebart et al. [17] propose a state visitation calculation, similar to backward value iteration in reinforcement learning, to compute the gradient of the entropy. The gradient calculation is adopted by most of the recent work in IRL for low-dimensional, discrete action spaces, which is inadequate for driving style optimizations. Our desired driving style requires high-resolution sampling of time-continuous actions, which produces a high-dimensional state space representation. In the following, we describe our intuitive approach, which combines search-based planning with *maximum entropy* IRL.

C. Path Integral Maximum Entropy IRL

In our IRL formulation, we maximize the log-likelihood L of expert behavior in the policy set Π by finding the reward function weights $\boldsymbol{\theta}$ that best describe human demonstrations $\pi^D \in \Pi^D$ within a planning cycle, which is given by

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \sum_{\pi^D \in \Pi^D} \ln p(\pi^D|\boldsymbol{\theta}) \quad (1)$$

$$= \arg\max_{\boldsymbol{\theta}} \sum_{\pi^D \in \Pi^D} \ln \frac{1}{Z} \exp(-\boldsymbol{\theta} \boldsymbol{f}^{\pi^D}), \quad (2)$$

where the partition function is defined by $Z = \sum_{\pi \in \Pi} \exp(-\boldsymbol{\theta} \boldsymbol{f}^{\pi})$.

Similar to Aghasadeghi et al. [22], we optimize under the constraint of matching the feature path integrals $\boldsymbol{f}^{\pi}$ of the demonstration and feature expectations of the explored policies,

$$\forall i \in 1, ..., k : \sum_{\pi \in \Pi} p(\pi|\theta) f_i^{\pi} = \frac{1}{m} \sum_{\pi^D \in \Pi^D} f_i^{\pi^D} = \hat{f}_i^{\Pi^D}, \quad (3)$$

where $\hat{f}_i^{\Pi^D}$ references the empirical mean of feature i calculated over m demonstrations in Π^D. The constraint in Eq. 3 is used to solve the non-linear optimization in Eq. 2.

The gradient of the log-likelihood can be derived as,

$$\nabla L(\boldsymbol{\theta}) = \sum_{\pi \in \Pi} p(\pi|\boldsymbol{\theta}) \boldsymbol{f}^{\pi} - \hat{\boldsymbol{f}}^{\Pi^D}, \quad (4)$$

and allows for gradient descent optimization.

The calculation of the partition function Z in Eq. 2 is often intractable due to the exponential growth of the state-action space over the planning horizon. The parallelism of the action sampling of the search-based planner allows us to explore a high-resolution state representation S_t for each discrete planning horizon increment t. A pruning operation terminates redundant states having sub-optimal behavior in the reachable set S_t, which is denoted by a lower value $V(s)$. Therefore, the pruning operation ensures multi-behavior exploration of the reachable set S_t that is evaluated with a single reward function. Thereby our sample-based planning methodology allows us to approximate the partition function similar to Markov chain Monte Carlo methods.

Once we obtain the new reward function, the configuration of the planner is updated. Hence, policies that have similar features as the human demonstration acquire a higher value assignment. This implies that they are more likely to be chosen as driving policy.

V. Experiments

We assess the performance of *path integral maximum entropy* IRL in urban automated driving. We focus on a base feature set for static environments, similar to the manual

tuning process of a motion planning expert. After this process more abstract reward features are tuned relative to the base features.

A. Data Collection and Simulation

Our experiments are conducted on a prototype vehicle, which uses a mediated perception architecture to produce feature maps as illustrated in Fig. 1. We recorded data in static environments and disabled object recognition and intention prediction. The data recordings include features of the perception system as well as odometry recordings of the human driver's actions. The training of our algorithm is performed during playbacks of the recorded data. After every planning cycle of the MPC, the position of the vehicle is reset to the odometry recording of the human demonstration.

B. Projection of Demonstration in State Space

The system overview in Fig. 2 includes a projection function that transfers the actions of a manual drive into the state-action space of the planning algorithm. The projection metric d is calculated during the graph construction between odometry ζ and continuous transitions $T(s, a, s')$ of all policies π in the set Π:

$$d(\zeta, \pi) = \int_t \alpha_t ||\zeta_t - \pi_t|| \, dt. \tag{5}$$

The norm is based on geometrical properties of the state space, e.g., the Euclidean distance in longitudinal and lateral direction as well as the squared difference in the yaw angle. Further, the metric includes a discount factor α_t over the planning horizon. The policy π^D has the least discounted distance towards the odometry record. There are multiple benefits of using the projection metric: First, the projected trajectory includes all constraints of the planner. If the metric surpasses a threshold limit, the human demonstrator does not operate in the actor's limits of the vehicle and therefore can not be used as a valid demonstration. Second, the projection metric allows for an intuitive evaluation of the driving style based on the geometrical proximity to the odometry. Third, we may augment the number of demonstrations by loosening the constraint of the policy π^D to have least discounted distance towards the odometry. Thereby, multiple planner policies qualify as demonstration $\pi^D \subseteq \Pi^D$.

C. Reward Feature Representation

In this work, the reward function $R(s, a)$ is given by a linear combination of K reward features. The features describe motion and infrastructural rewards of driving. The discount factor γ is manually defined by a motion planning expert and is not optimized at this stage of the work. Our perception system in Fig. 2 provides normalized feature maps with spatial information of the environment. The feature path integral $\boldsymbol{f}^\pi$ of a policy π is created by transitioning through the feature map representation of the environment. We concentrate on a base reward set consisting of $K = 12$ features, which are listed in the legend of Fig. 4a. Heinrich et al. formally described a sub-set of our features [18]. Seven of our feature values describe the motion characteristics of the policies, which are given by derivatives of the lateral and longitudinal actions. They include the difference between the target and policy velocity, and the acceleration and jerk values of actions. The target in the velocity may change depending on the situation, e.g., in a situation with a traffic light the target velocity may reduce to zero. Furthermore, the end direction feature is an important attribute for lateral behavior that specifies the angle towards the driving direction at the end of the policy. The creeping feature suppresses very slow longitudinal movement in situations, where a full stop is more desired. Infrastructural features include proximity measures to the lane center and curbs, cost potentials for lanes, and direction. Furthermore, we specify a feature for conflict areas, e.g., stopping at a zebra crossing.

D. Implementation Details

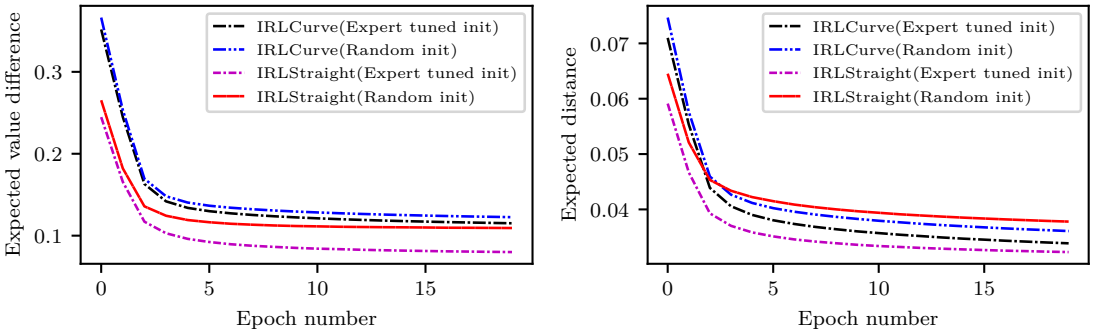
During the playback of a human demonstration, the path integral feature vectors $\boldsymbol{f}^\Pi$ of the policy set Π are approximated for every planning cycle and stored within a replay buffer. By including our projection metric in the action sampling procedure, we associate each policy π with the distance to the odometry of the human demonstration. During training, we query demonstrations Π^D, which are policies with a low projection metric value, from our replay buffer where $\pi^D \subseteq \Pi^D \subseteq \Pi$. Hence, the replay buffer contains features of demonstrations for each planning cycle denoted as $\boldsymbol{f}^{\Pi^D} \subseteq \boldsymbol{f}^\Pi$. Fig. 2 describes the information flow from the odometry record of the demonstration to the feature query from the replay buffer. Due to actor constraints of the automated vehicle's actions, the planning cycles without demonstrations are not considered for training. We utilize experience replay similar to reinforcement learning and update on samples or mini-batches of experience, by drawing randomly from the buffered policies. This process allows us to efficiently use previous experience, which can be trained on multiple times. Further, stability is provided by not altering the representation of the expert demonstrations within the graph representation.

VI. Evaluation

We aim to evaluate the utility of our automated reward function optimization in comparison to manual expert tuning. First, we analyze our driving style imitation in terms of value convergence towards the human demonstration. Second, we compare the driving style of our policies under random, learned, and expert-tuned reward functions against human driving on dedicated test route segments.

A. Training Evaluation

We analyze the convergence for different training initializations and road segment types, namely straight and curvy. Due to the linear combination of reward weights, one expects a segment-specific preference of the reward function. As a reference, a motion planning expert generated a tuned reward function for general driving. We perform two drives per training segment, one with a random and one with an



(a) Difference between expected value of human driving demonstration and expected value of planner policies under learned reward functions.

(b) Expected distance of planner policies towards the human driving demonstration under learned reward functions.

Fig. 3: Illustration of training and validation metrics for multiple segments and training initializations. Convergence of maximum entropy IRL over training epochs. Validation of the training by indicating the reduction of the expected distance towards the human demonstration. The probability is calculated independently for every planning cycle of the MPC, whereas the policy set includes an average of approx. 4000 policies.

expert-tuned reward function. The policies to be considered as human demonstrations are chosen based on our projection metric and therefore depend on our chosen reward function initialization. The expert initialization yields demonstrations with a mean projection error 7% lower as compared to random initialization. During every planning cycle on the segments, we trace the policies of the planner in replay buffers. We generate four tuned reward functions which are referred to in Fig. 4a by training on our replay buffers.

The convergence of the training algorithm is measured by the expected value difference (EVD) over training epochs between learned and demonstrated policies. The EVD is calculated for every planning cycle and averaged over the segment. The EVD is given by

$$\mathbb{E}[V(\Pi)] - \mathbb{E}[V(\Pi^D)] \tag{6}$$

$$= \sum_{\pi \in \Pi} p(\pi|\boldsymbol{\theta}) V(\pi) - \sum_{\pi^D \in \Pi^D} p(\pi^D|\boldsymbol{\theta}) V(\pi^D). \tag{7}$$

The performance of the random and expert-tuned reward functions is given by the EVD at epoch zero. The initial and final EVD differences between the straight and curvy segment is 30% and 19% respectively. A preference of the straight segments by both reward functions is visible in the initial EVD difference Fig. 3. The learned reward functions show a large EVD reduction of 67% for curvy and 63% for straight segments at the end of training.

We can interpret the training results in the following way:

(a) The projection metric depends on the quality of the reward function.
(b) Improved reward functions lead to improved action sampling and therefore produce better demonstrations.
(c) Learning reward functions without prior knowledge is possible, e.g. generating a replay buffer with a randomly initialized reward function and training with a random initialization.
(d) Unsuitable reward functions improve more significantly during training.

Hence, continuously updating the policies in the replay buffer generated from an updated reward function should lead to faster convergence.

The desired driving style is given by the actions of a human driving demonstration. Therefore, the projection error in Eq. 5, which we use to select driving demonstrations, extends itself as a direct validation metric of the actions. Due to our goal of optimizing the likelihood of human driving behavior within the policy set, we calculate the expected distance (ED) in the policy set, given by

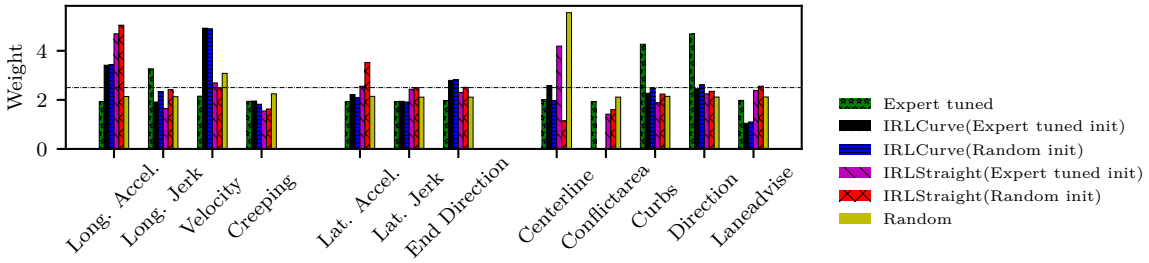
$$\mathbb{E}[d(\zeta, \Pi)] = \sum_{\pi \in \Pi} p(\pi|\boldsymbol{\theta}) d(\zeta, \pi). \tag{8}$$

The learned reward functions in Fig. 3b show a large ED reduction of 54% for curvy and 44% for straight segments at the end of training. The ED reduction trends have high similarity to the above mentioned EVD trend and therefore this validates the premise of a high correlation between value and distance to the demonstration. An improved expected distance ensures a high likelihood of selecting policies which are similar to humanlike driving demonstrations.

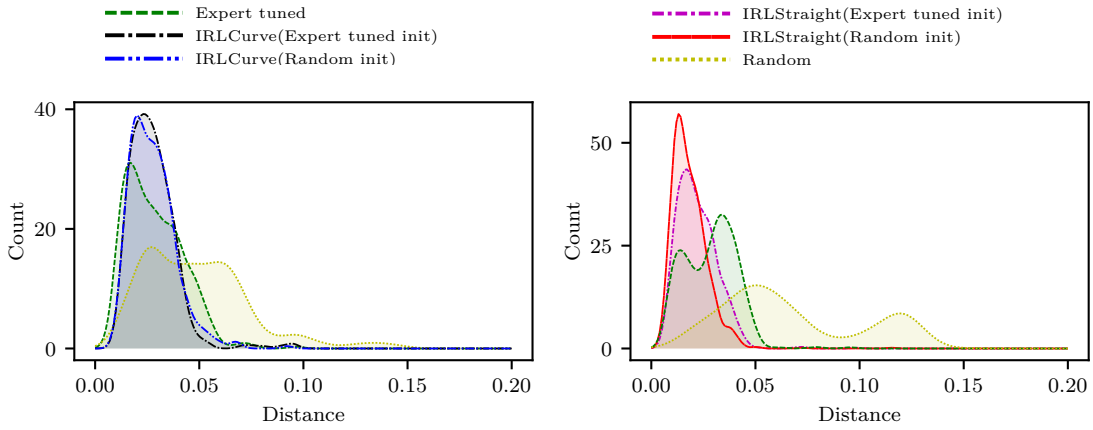
B. Driving Style Evaluation

In this part of the evaluation, we compare the driving style of the random, learned, and expert-tuned reward functions shown in Fig. 4a to manual human driving. The parameters of the reward functions allow for introspection and reasoning about the segment-specific preference. The reward weight is inversely proportional to the preference of that feature value in the policy. Learned reward functions are of two types:

(a) IRL with random initialization, hereby referred as IRL(random). Both the training trajectory set and the learning task are randomly initialized.
(b) IRL with expert initialization, hereby referred as IRL(expert). Both the training trajectory set and the learning task are initialized by expert tuning.

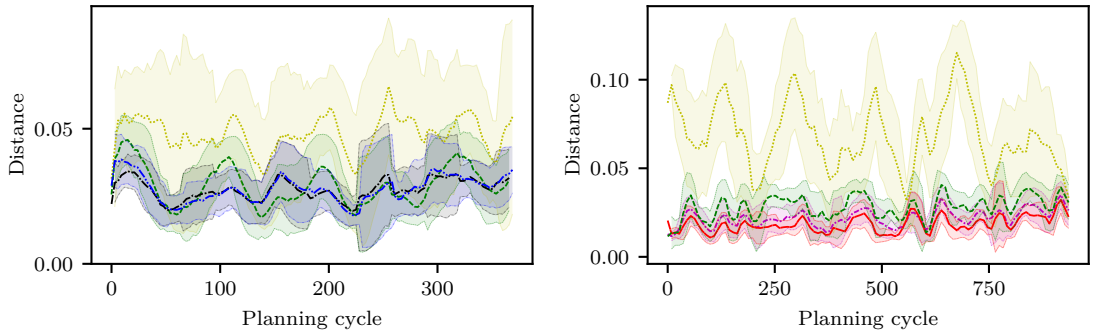


(a) Feature weights of tested reward functions.



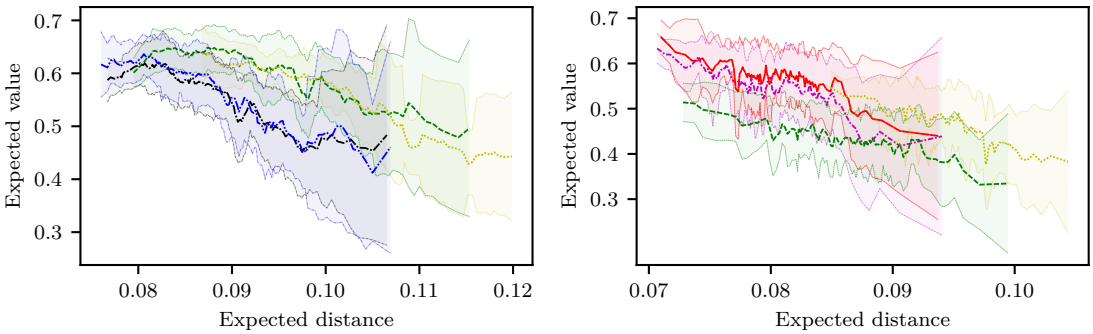
(b) Distances of the optimal policies on a curvy test segment.

(c) Distances of the optimal policies on a straight test segment.



(d) Distances of the optimal policy over planning cycles on a curvy test segment.

(e) Distances of the optimal policy over planning cycles on a straight test segment.



(f) Expected value and distance under the expert tuned reward function of every planning cycle on a curvy test segment.

(g) Expected value and distance under the expert tuned reward function of every planning cycle on a straight test segment.

Fig. 4: The tests contrast the driving style of random, learned, and expert-tuned reward functions. The graphs present the results of independent playbacks on a dedicated test track. The probability is calculated independently for every planning cycle of the MPC, whereas the policy set includes on average 4000 policies.

Using these reward functions, we run our planning algorithm on dedicated test route segments to verify the generalized performance of the optimal policies. We carry out multiple drives over the test segments to generate representative statistics. Fig. 4b and Fig. 4c present the projection metric distribution, which is the distance of the optimal policy to the odometry of a manual human drive for every planning cycle. We fit a Gaussian distribution over the histogram with 200 bins of size 0.001 with 944 planning cycles for the straight and 369 planning cycles for the curvy segment. The learned reward functions improve the driving style on all segments even in the case of random initialization. Our evaluation metric, which is the mean distance of the optimal policy to the odometry, decreases for IRLStraight(random) by 73% and for IRLCurve(random) by 43%. In case of expert-tuned initialization, IRLStraight(expert) decreased by 22% and IRLCurve(expert) by 4%. The strong learning outcome in the straight segment can be attributed to the easier learning task as compared to the curvy segment. Even though the expert-tuned reward functions do not improve substantially in terms of mean distance, they show a lower variance in distance of the optimal policy to the odometry over planning cycles after training as is shown in Fig. 4d and Fig. 4e. Here we indicate variance in the distance of the optimal policy over planning cycles by one standard deviation. The variance reduction of learned reward function depicts higher stability over planning cycles. Hence, we are able to encode the human driving style through IRL without applying prior domain knowledge as done by motion planning experts.

Fig. 4f and Fig. 4g present the expected value of our evaluated reward functions r^A under the expert-tuned reward function r^E, given by $\mathbb{E}[V(\Pi)] = \sum_{\pi \in \Pi} p(V^\pi(r^A))V^\pi(r^E)$. The overall trend indicates an inverse relationship between expected value and expected distance. The learned reward functions have lower expected distance as compared to expert tuned and random reward functions, while having a higher rate of value reduction with increasing expected distance. This ensures that the learned reward functions induce a high degree of bias in the policy evaluation such that the humanlike demonstrated behavior is preferred.

VII. Conclusion and Future Work

We utilize *path integral maximum entropy* IRL to learn reward functions of a general-purpose planning algorithm. Our method integrates well with model-based planning algorithms and allows for automated tuning of the reward function encoding the humanlike driving style. This integration makes maximum entropy IRL tractable for high dimensional state spaces. The automated tuning process allows us to learn reward functions for specific driving situations. Our experiments show that learned reward functions improve the driving style exceeding the level of manual expert-tuned reward functions. Furthermore, our approach does not require prior knowledge except the defined features of the linear reward function. In the future, we plan to extend our IRL approach to update the reward function dynamically.

References

[1] M. McNaughton, "Parallel Algorithms for Real-time Motion Planning," Ph.D. dissertation, Carnegie Mellon University, 2011.

[2] C. Katrakazas, M. Quddus, W.-H. Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transportation Res. Part C: Emerging Technologies*, vol. 60, pp. 416–442, 2015.

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

[4] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and Decision-Making for Autonomous Vehicles," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 1, no. 1, pp. 187–210, 2018.

[5] S. Ulbrich and M. Maurer, "Towards Tactical Lane Change Behavior Planning for Automated Vehicles," in *Proc. IEEE Int. Conf. Intell. Transp. Syst. (ITSC)*, 2015.

[6] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenet frame," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2010.

[7] S. Heinrich, J. Stubbemann, and R. Rojas, "Optimizing a driving strategy by its sensor coverage of relevant environment information," in *IEEE Intell. Vehicles Symp.*, 2016, pp. 441–446.

[8] T. Gu, J. M. Dolan, and J.-W. Lee, "Automated tactical maneuver discovery, reasoning and trajectory planning for autonomous driving," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Syst. (IROS)*, Daejeon, South Korea, 2016.

[9] P. Abbeel, D. Dolgov, A. Y. Ng, and S. Thrun, "Apprenticeship learning for motion planning with application to parking lot navigation," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Syst. (IROS)*, 2008.

[10] M. Wulfmeier, D. Rao, D. Z. Wang, P. Ondruska, and I. Posner, "Large-scale cost function learning for path planning using deep inverse reinforcement learning," *Int. J. Robotics Research*, vol. 36, no. 10, pp. 1073–1087, 2017.

[11] M. Kuderer, S. Gulati, and W. Burgard, "Learning driving styles for autonomous vehicles from demonstration," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2015.

[12] S. Arora and P. Doshi, "A survey of inverse reinforcement learning: Challenges, methods and progress," in *arXiv Preprint arXiv:1806.06877*, 2018.

[13] K. Shiarlis, J. Messias, and S. Whiteson, "Inverse Reinforcement Learning from Failure," in *Proc. Int. Conf. Autonomous Agents Multi-Agent Syst.*, 2016.

[14] A. Byravan, M. Monfort, B. Ziebart, B. Boots, and D. Fox, "Graph-Based Inverse Optimal Control for Robot Manipulation," in *Proc. Int. Joint Conf. Artificial Intell. (IJCAI)*, vol. 15, 2015.

[15] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," in *Learning, Inference and Control of Multi-Agent Syst. Workshop (NIPS)*, 2016.

[16] ——, "On a Formal Model of Safe and Scalable Self-driving Cars," in *arXiv Preprint arXiv:1708.06374*, 2017.

[17] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum Entropy Inverse Reinforcement Learning." in *Proc. Nat. Conf. Artificial Intell. (AAAI)*, vol. 8, 2008.

[18] S. Heinrich, "Planning Universal On-Road Driving Strategies for Automated Vehicles," Ph.D. dissertation, Freie Universität Berlin, 2018.

[19] A. Y. Ng and S. J. Russell, "Algorithms for Inverse Reinforcement Learning," in *Proc. Int. Conf. Machine Learning (ICML)*, 2000.

[20] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proc. Int. Conf. Machine Learning (ICML)*. ACM, 2004.

[21] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "Maximum margin planning," in *Proc. Int. Conf. Machine Learning (ICML)*, 2006.

[22] N. Aghasadeghi and T. Bretl, "Maximum entropy inverse reinforcement learning in continuous state spaces with path integrals," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Syst. (IROS)*. IEEE, 2011.

A.3 DRIVING STYLE ENCODER: SITUATIONAL REWARD ADAPTATION FOR GENERAL-PURPOSE PLANNING IN AUTOMATED DRIVING

Sascha Rosbach, Vinit James, Simon Großjohann, Silviu Homoceanu, Xing Li, and Stefan Roth
IEEE International Conference on Robotics and Automation (ICRA), Paris, France, June 2020.

Abstract

General-purpose planning algorithms for automated driving combine mission, behavior, and local motion planning. Such planning algorithms map features of the environment and driving kinematics into complex reward functions. To achieve this, planning experts often rely on linear reward functions. The specification and tuning of these reward functions is a tedious process and requires significant experience. Moreover, a manually designed linear reward function does not generalize across different driving situations. In this work, we propose a deep learning approach based on inverse reinforcement learning that generates situation-dependent reward functions. Our neural network provides a mapping between features and actions of sampled driving policies of a model predictive control-based planner and predicts reward functions for upcoming planning cycles. In our evaluation, we compare the driving style of reward functions predicted by our deep network against clustered and linear reward functions. Our proposed deep learning approach outperforms clustered linear reward functions and is at par with linear reward functions with a priori knowledge about the situation.

Driving Style Encoder: Situational Reward Adaptation for General-Purpose Planning in Automated Driving

Sascha Rosbach[1,2], Vinit James[1], Simon Großjohann[1], Silviu Homoceanu[1], Xing Li[1] and Stefan Roth[2]

***Abstract*— General-purpose planning algorithms for automated driving combine mission, behavior, and local motion planning. Such planning algorithms map features of the environment and driving kinematics into complex reward functions. To achieve this, planning experts often rely on linear reward functions. The specification and tuning of these reward functions is a tedious process and requires significant experience. Moreover, a manually designed linear reward function does not generalize across different driving situations. In this work, we propose a deep learning approach based on inverse reinforcement learning that generates situation-dependent reward functions. Our neural network provides a mapping between features and actions of sampled driving policies of a model-predictive control-based planner and predicts reward functions for upcoming planning cycles. In our evaluation, we compare the driving style of reward functions predicted by our deep network against clustered and linear reward functions. Our proposed deep learning approach outperforms clustered linear reward functions and is at par with linear reward functions with a-priori knowledge about the situation.**

I. Introduction

Automated driving in urban environments requires intelligent decision making that scales over a variety of traffic situations. A scalable approach needs to be able to address both structured and un-structured traffic as experienced in urban environments. In model-based planning, a semantic description of the environment is encoded in the form of static and kinematic features. The planning system has to translate this semantic description of the environment into safe and human-acceptable actions. The underlying planning algorithm often relies on manually-tuned linear reward functions that encode the relevance of the predefined features. Tuning such reward functions is a tedious task and is usually performed by motion planning experts. As a result, the reward function is often considered to be a static external signal that does not depend on the driving situation [1]. Manual tuning of the reward function becomes infeasible if a planning system is applied at scale and has to adopt a variety of driving styles. In this paper, we propose a deep learning approach in which a neural network dynamically predicts reward functions based on constantly changing static and kinematic features of the environment.

Human driving demonstrations enable the application of inverse reinforcement learning (IRL) for finding the underlying reward functions [2], [3]. In this work, we utilize this methodology to learn situation-dependent reward functions for our planner [4]–[6]. Our planner samples a large set of actions generating distributions of path integral (PI) features and actions. Unlike related work in deep inverse reinforcement learning (DIRL) that focuses on spatial reward functions [7], [8], our approach generates driving styles by incorporating vehicle kinematics. This is done by integrating our deep learning approach into a model-predictive control (MPC) planning algorithm. Sampled driving policies of the MPC are used as inputs to our neural network. The network learns a representation of the driving situation by matching distributions of features and actions to reward functions based on the maximum entropy principle.

The main contributions of this paper are threefold: First, we formulate a deep IRL methodology that predicts situation-dependent reward functions based on PI features and actions. Second, we propose a neural network architecture that utilizes one-dimensional convolutions over PI features and actions of sampled driving policies to learn a representation of the statics and kinematics of the situation. Third, we show the feasibility of our proposed approach in real-world traffic environments and compare our method to linear and latent maximum entropy IRL.

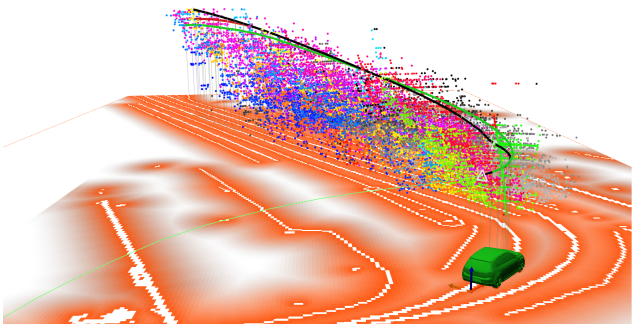


Fig. 1: This figure illustrates our planner for automated driving that samples policies for our path integral (PI) IRL formulation. The visualized state space is color-coded based on the state-action values. The z-axis corresponds to the velocity, whereas the ground plane depicts the spatial feature maps such as distance transformed lane centers and road boundaries. Three color-coded policies are visualized, namely the optimal policy (black), the odometry of a human demonstration (green), and the projection of the demonstration into the state space (red).

[1]The authors are with the Volkswagen AG, 38440 Wolfsburg, Germany {sascha.rosbach, vinit.james, simon.grossjohann, silviu.homoceanu, xing.li}@volkswagen.de

[2]The authors are with the Visual Inference Lab, Department of Computer Science, Technische Universität Darmstadt, 64289 Darmstadt, Germany stefan.roth@visinf.tu-darmstadt.de

II. Related Work

As of now, the decision-making system in automated vehicles is often decomposed into a hierarchical structure to reduce the complexity of the model-based system. This hierarchical decomposition, however, yields uncertainty due to insufficient knowledge of the subsequent level. In particular, these systems are prone to failure in unforeseen driving situations where simplified behavior models fail to match predefined templates [6]. General-purpose planning algorithms for automated driving aim to reduce planning-task decompositions to enable a scalable architecture that does not rely on behavior implementations. Starting with the work of McNaughton [9], attention has been drawn to parallel real-time planning that combines behavior planning and local motion planning. This approach enables sampling of a large set of actions that respect kinematic constraints. Thereby a sequence of sampled actions can represent complex maneuvers. This kind of general-purpose planner uses a linear reward function to map a complex feature space into rewards. Manual tuning of such a linear reward function is a tedious task performed by motion-planning experts. In our previous work, we automated the tuning process using IRL [5]. We found that inferred reward functions for situations defined a-priori surpass the performance of an expert-tuned reward function. However, the tuned linear reward functions do not generalize well over different situations as the objectives change continuously, e.g., the importance of keeping to the lane center in straight road segments while allowing deviations in curvy parts. In this work, we utilize the general-purpose planning approach and focus on situation-dependent reward function prediction.

A naive solution to the aforementioned problem is to use IRL with a-priori knowledge of situations and recover a set of situation-dependent reward functions. However, the complexity is limited to the number of identifiable scenarios that have to be mapped during inference. Prior work in IRL employs the latent maximum-entropy principle to condition reward functions on the situation [10]. Expectation Maximization (EM) is often used in combination with IRL to learn a prior over the situation [11]. Training can be performed on a mixture of different driving segments without a-priori knowledge about the situation. Hence, multiple reward functions are derived and a mixture is inferred during planning using Bayes' rule. Dirichlet process mixture models have been used to address some drawbacks of EM, specifically the reliance on a specified number of clusters [12], [13]. Recent work in IRL utilizes deep neural networks to extend the model capacity of mixture models. Wulfmeier et al. [7] proposed to learn spatial traversability by deep IRL using deep convolutional neural networks, which generates a direct mapping from raw sensory data to reward maps. This allows for encoding a situation into the reward function for a planning algorithm that operates on a grid-based representation. However, their IRL formulation relies on state-visitation frequency updates that do not scale to high-dimensional state-action spaces.

In contrast to our model-based IRL formulation, prior work has also addressed high-dimensional state-spaces with model-free approaches. Finn et al. [14] proposed a sample-based deep IRL approach that learns the cost in the inner loop of a model-free policy optimization. This approach is suitable for problems with unknown system dynamics. Recent work also includes the idea of combining generative adversarial neural networks and imitation learning to clone the behavior of the demonstrated policy without finding the underlying reward functions [15].

In our work, we rely on a defined environment and vehicle transition model to explicitly incorporate traffic rules and provide safety [16]. We propose to couple the reliability of a model-based planning system with the generalization ability of deep inverse reinforcement learning. Our search-based planning algorithm operates in large and continuous state space and produces policies with multiple behaviors. This enables us to approximate the partition function required for the IRL gradient formulation with high accuracy [5]. Our IRL formulation uses a neural network to reason about static and kinematic features of the environment, thereby predicting situation-dependent driving styles.

III. Preliminaries

The interaction of the agent with the environment is often formulated as a Markov Decision Process (MDP), consisting of a 5-tuple $\{\mathcal{S}, \mathcal{A}, T, R, \gamma\}$, where $\mathcal{S}$ denotes the set of states and $\mathcal{A}$ describes the set of actions. The reward function R is discounted by γ and assigns a reward for every action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$. Our planning algorithm relies on an environment model M and an underlying vehicle transition function T. The model consists of static features of the environment that are derived from perception and localization, as well as kinematic features derived from the vehicle transition function. The planner generates a policy set Π by sampling actions a from distributions conditioned on vehicle dynamics for each state s. The features for each continuous action a are integrated in the policy generation process. The reward function R is given by a linear combination of K static and kinematic feature values f_i with weights θ_i such that $R(s,a) = \sum_{i \in K} -\theta_i f_i(s,a)$. The value V^π of a policy π is the integral of discounted rewards during continuous transitions, $V^\pi = \int_t \gamma_t R(s_t, a_t)\, dt$. The PI feature f_i^π for a policy π is defined by $f_i^\pi = \int_t \gamma_t f_i(s_t, a_t)\, dt$. The demonstrations for our IRL formulation are based on odometry records ζ of human driving. During the training in our simulation, we project the odometry record ζ into the state-action space and are thereby able to formulate a demonstration from the policy set such that $\pi^D \in \Pi$.

A. Maximum entropy IRL

The goal of IRL is to find the reward function weights $\boldsymbol{\theta}$ that enable the optimal policy π^* to be at least as good as the demonstrated policy π^D [17]. Thereby, the planner indirectly imitates the behavior of a demonstration [2]. In PI IRL, a probabilistic model is formulated that yields a probability distribution over policies, $p(\pi|\boldsymbol{\theta})$ [18], [19]. The model is optimized such that the expected PI feature values

$\mathbb{E}_{p(\pi|\boldsymbol{\theta})}[\boldsymbol{f}^{\pi}]$ of the policy set Π match the empirical feature values $\hat{\boldsymbol{f}}^{\Pi^D}$ of the demonstrations for each planning cycle of the MPC. Ziebart et al. [20] propose to maximize the entropy of the distribution to solve the ambiguity introduced by imperfect demonstrations, giving rise to the policy distribution

$$p(\pi|\boldsymbol{\theta}) = \frac{1}{Z}\exp(-\boldsymbol{\theta}^{\top}\boldsymbol{f}^{\pi}). \tag{1}$$

The calculation of the partition function $Z = \sum_{\pi\in\Pi}\exp(-\boldsymbol{\theta}^{\top}\boldsymbol{f}^{\pi})$ is often intractable due to the exponential growth of the state-action space. Our planning algorithm approximates the partition function similar to Markov chain Monte Carlo methods. Maximizing the entropy of the distribution over policies subject to the feature constraints from demonstrated policies implies that the log-likelihood $L(\boldsymbol{\theta})$ of the observed policies under the maximum entropy distribution is maximized. From this hypothesis, the log-likelihood gradient is obtained as

$$\nabla L(\boldsymbol{\theta}) = \sum_{\pi\in\Pi} p(\pi|\boldsymbol{\theta})\boldsymbol{f}^{\pi} - \hat{\boldsymbol{f}}^{\Pi^D}. \tag{2}$$

Due to the dependency of the reward function on the driving situation, the probabilistic model $p(\pi|\boldsymbol{\theta})$ that recovers a single reward function for the demonstrated trajectories does not scale.

IV. Path Integral Latent IRL

Instead of a single linear reward function, we consider that there are N different reward functions, each corresponding to situation-dependent behavior. As extension of our previous work, we incorporate latent variables in our PI IRL formulation [5], which allows us to infer a mixture of these reward functions during planning. In the following derivation, we consider a single demonstration π^D for every planning cycle. We adopt the formulation of Babes et al. [11] that utilizes Expectation Maximization (EM) in IRL, where $\boldsymbol{\beta}_c^{\pi^D}$ is the probability that a demonstration π^D belongs to a cluster c, and $\boldsymbol{\psi}(c)$ is the the estimated prior probability of a cluster c. The EM algorithm iteratively alternates between an estimation step (E-step) and a maximization step (M-step). Within the E-step, we compute the probability of a demonstration belonging to a cluster as

$$\boldsymbol{\beta}_c^{t,\pi^D} = \frac{p(\pi^D|c,\boldsymbol{\theta}^t,\boldsymbol{\psi}^t)\boldsymbol{\psi}^t(c)}{\sum_{c\in C} p(\pi^D|c,\boldsymbol{\theta}^t,\boldsymbol{\psi}^t)\boldsymbol{\psi}^t(c)}, \tag{3}$$

exploiting the constraint $\sum_{c\in C}\boldsymbol{\beta}_c^{\pi^D} = 1$. In Eq. 3, $p(\pi^D|c,\boldsymbol{\theta},\boldsymbol{\psi})$ reduces to $p(\pi^D|\boldsymbol{\theta}_c)$.

Within the M-step, we compute for each iteration t the prior probability $\boldsymbol{\psi}(c)$ as

$$\boldsymbol{\psi}^{t+1}(c) = \frac{1}{B}\sum_{\pi^D\in\Pi^B}\boldsymbol{\beta}_c^{t,\pi^D}, \tag{4}$$

where B is the number of demonstrations in a batch Π^B. Furthermore, we compute the reward functions $\boldsymbol{\theta}_c$ for each cluster as

$$\boldsymbol{\theta}_c^{t+1} = \arg\max_{\boldsymbol{\theta}} \sum_{\pi^D\in\Pi^B}\boldsymbol{\beta}_c^{t,\pi^D}\Big[\ln p(\pi^D|\boldsymbol{\theta}_c)\Big]. \tag{5}$$

We obtain the log-likelihood gradient for a cluster c as

$$\nabla_{\boldsymbol{\theta}_c} L(\boldsymbol{\theta}_c) = \boldsymbol{\beta}_c^{\pi^D}\Big(\sum_{\pi\in\Pi} p(\pi|\boldsymbol{\theta}_c)\boldsymbol{f}^{\pi} - \boldsymbol{f}^{\pi^D}\Big). \tag{6}$$

In contrast to the gradient formulation of IRL that finds a single reward function for the demonstrated policies, latent IRL finds a number of reward functions. The probability that a demonstration belongs to cluster $\boldsymbol{\beta}_c^{\pi}$ influences the update of the gradient. During training, $\boldsymbol{\beta}_c^{\pi}$ is inferred using Bayes' rule on the basis of the PI features of the demonstration. Due to the absence of features of the demonstration during online planning, we assume that the optimal solution on the basis of the last predicted reward function allows us to infer a situation-dependent reward function for the next planning cycle. In order to scale over a large number of situations, we next formulate an alternative approach based on deep learning.

V. Path Integral Deep IRL

We propose a deep learning approach for PI maximum entropy IRL that approximates a complex mapping between the situation and reward function. Our neural network predicts the reward function weights $\boldsymbol{\theta}_{k+1}$ based on the actions $\boldsymbol{a}_k^{\Pi}$ and PI features $\boldsymbol{f}_k^{\Pi}$ of the policy set Π_k at MPC cycle k, given by $\boldsymbol{\theta}_{k+1} \approx g(\boldsymbol{\Theta}, \boldsymbol{f}_k, \boldsymbol{a}_k)$.

The IRL problem can be formulated in the context of Bayesian inference as MAP estimation, which entails maximizing the joint posterior distribution of observing expert demonstrations Π^D. We calculate the maximum entropy probability based on the linear reward weights $\boldsymbol{\theta}$, which are inferred by the network with parameters $\boldsymbol{\Theta}$ as

$$L(\boldsymbol{\theta}) = L(g(\boldsymbol{\Theta},\boldsymbol{f},\boldsymbol{a})) = \sum_{\pi^D\in\Pi^D}\ln p(\pi^D|g(\boldsymbol{\Theta},\boldsymbol{f},\boldsymbol{a})). \tag{7}$$

Maximization leads to the optimal weights

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \sum_{\pi^D\in\Pi^D}\ln p(\pi^D|g(\boldsymbol{\Theta},\boldsymbol{f},\boldsymbol{a})). \tag{8}$$

The gradient calculation from PI maximum entropy IRL from Eq. 2 lends itself naturally towards training deep neural networks, where the gradient of the log-likelihood $L(\boldsymbol{\theta})$ can be calculated in terms of $\boldsymbol{\Theta}$ as

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{\Theta}} &= \frac{\partial L}{\partial \boldsymbol{\theta}}\cdot\frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\Theta}} \\ &= \Big[\sum_{\pi\in\Pi} p(\pi|\boldsymbol{\theta})\boldsymbol{f}^{\pi} - \hat{\boldsymbol{f}}^{\Pi^D}\Big]\cdot\frac{\partial}{\partial\Theta}g(\boldsymbol{\Theta},\boldsymbol{f},\boldsymbol{a}).\end{aligned} \tag{9}$$

The gradient is separated into the maximum entropy gradient in terms of $\boldsymbol{\theta}$ and the gradient of $\boldsymbol{\theta}$ w.r.t. the network parameters $\boldsymbol{\Theta}$, which can be directly obtained via backpropagation in the deep neural network.

A. Training methodology

Algo. 1 describes the training algorithm used for PI maximum entropy deep IRL. IRL training is often very time consuming since the MDP has to be solved with the current reward function in the inner loop of reward learning. In

Algorithm 1: Deep IRL Training

Input: odometry of human demonstation ζ, model M, planning horizon H, reward discount factor γ
Output: network weights Θ^*

function $DeepIRLTraining(\Pi^D, M, H, \gamma)$
  $\boldsymbol{\theta_0} \leftarrow$ init reward weights
  $\Pi, \Pi^D \leftarrow$ Planning$(\boldsymbol{\theta_0}, M, H, \gamma, \zeta)$
  $\boldsymbol{\Theta_0} \leftarrow$ init network weights
  for *e in Epochs* **do**
    for *b in Batch* **do**
      $\boldsymbol{\theta_b} \leftarrow$ network forward pass$(\boldsymbol{\Theta_b}, \boldsymbol{f_b}, \boldsymbol{a_b})$
      $\frac{\partial \mathcal{L}_b}{\partial \boldsymbol{\theta_b}} = \sum_{\pi \in \Pi} p(\pi|\boldsymbol{\theta_b}) \boldsymbol{f_b^\pi} - \hat{\boldsymbol{f}}_{\boldsymbol{b}}^{\Pi^D}$
      $\frac{\partial \mathcal{L}_b}{\partial \boldsymbol{\Theta_b}} \leftarrow$ network backprop$(\boldsymbol{\Theta_b}, \boldsymbol{f_b}, \boldsymbol{a_b}, \frac{\partial \mathcal{L}_b}{\partial \boldsymbol{\theta_b}})$
      $\boldsymbol{\Theta_{b+1}} \leftarrow$ update weights$(\boldsymbol{\Theta_b}, \frac{\partial \mathcal{L}_b}{\partial \boldsymbol{\Theta_b}})$

our IRL formulation, we do not solve the MDP within the inner loop; instead we run our planning algorithm prior to training with a randomly initialized reward function $\boldsymbol{\theta}_0$ over training segments. Thereby, we generate a buffer of planning cycles consisting of a set of policy sets Π with corresponding PI features f^Π and actions a^Π. Due to the high-resolution sampling of actions, we ensure that there are policies that are geometrically close to human-recorded odometry and resemble human driving styles. Therefore the task of IRL is to find the unknown reward function that increases the likelihood of these trajectories to be considered as optimal policies. During the policy generation, we utilize a weighted Euclidean distance towards human driven trajectories ζ to generate demonstrations Π^D for each policy set Π. Similar to the reward weights $\boldsymbol{\theta}$, the network parameters $\boldsymbol{\Theta}$ are initialized with random values before starting the training. The training loop is run for a predefined number of epochs, ensuring that the convergence metric, the expected value difference (EVD), reaches the desired threshold value. For each epoch the training dataset is shuffled and divided into batches to perform mini-batch gradient decent.

B. Neural network architecture

In this work a deep architecture is proposed, which maps the PI features f^Π and actions a^Π to linear reward weights $\boldsymbol{\theta}$. The input to the network is a set of 15 PI features f^Π and 8 actions a^Π for every policy of a planning cycle. The continuous actions are reduced to steering angle and accelerations at discrete control points. The network architecture comprises of one-dimensional convolutions and average pooling to extract latent variables of policies describing the high-dimensional input space. The architecture uses one-dimensional convolutional layers to learn the causal relationship between sampled actions and resultant features of every policy. There is no inherent relationship among policies in the input space therefore no information is gained by performing higher dimensional convolutions. The architecture consists of convolutional building blocks, each comprising of two convolutional layers followed by an average pooling layer which performs dimension reduction. There are five such blocks in succession followed by fully-connected layers. A series of eight fully-connected layers at the end learn inter-policy relationships from the resultant latent variables of the convolutional stack. A vector of linear reward weights with the same dimension as the no. of features, i.e. 15, is the output of the network. All activation functions in the network are chosen as Rectified Linear Units, except for the output layer where no activation function is used.

C. Inference during MPC planning cycles

During real-time planning in an automated drive, the MPC re-plans in discrete time-steps k. After receiving the features and actions of the latest planning cycle, the neural network infers the reward weights. To enable smooth transitions of the reward functions, we utilize a predefined history size h to calculate the empirical mean of weights $\hat{\boldsymbol{\theta}}$. The weights hence obtained are used to continuously re-parameterize the planning algorithm for the subsequent planning cycle.

VI. Experiments

In our experiments, we use our proposed PI deep IRL (PIDIRL), as well as PI latent IRL (PILIRL) and PI linear IRL (PIIRL) in real urban driving situations. We focus our experiments on situations that introduce conflicts in driving objectives so as to highlight the importance of situation-dependent reward functions.

A. Data collection and simulation

Our experiments are conducted on a prototype vehicle, which uses a mediated perception architecture to produce feature maps of the environment. These feature maps are depicted in the ground-plane in Fig. 1. The MPC planning algorithm uses the feature maps to get state features and computes state-values as shown in as color-coded point clouds in Fig. 1. We gathered data on a set of streets in Hamburg, Germany located around Messe and Congress, see Fig. 3a. We concentrate on static infrastructure and objects while disabling intention prediction of dynamic objects to test our reward functions in different location-dependent situations on this track. Three drives over this course are segmented into train and test tracks, each of which is subdivided into four situations. Situation 1 resembles sharp curves, situation 2 stopping, starting and turns at traffic lights, situation 3 stopping and starting at a traffic lights, and situation 4 resembles lane following. The training of our algorithm is performed in a playback simulation of the recorded data as depicted in Fig. 1. After every planning cycle of the MPC, the position of the vehicle is reset to the odometry recording of the human demonstration

B. Reward feature representation

We concentrate on a reward set consisting of $K = 15$ manually engineered features, which are listed in the table of Fig. 3g. Infrastructural features are derived by a data fusion between objects and street network [21]. The kinematic characteristics of the policies are given by derivatives of the lateral and longitudinal actions as well as features related to behavior, e.g. lane change delay. In addition selection features are designed so as to evaluate policies based on more nuanced attributes such as maneuvers space, progress and policy end direction toward the road.

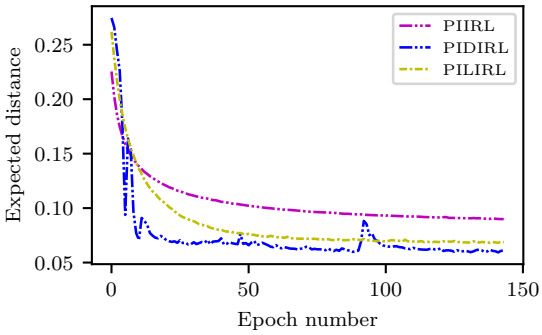


Fig. 2: Expected distance over epochs of training.

VII. Evaluation

We compare the results of the situation-dependent reward functions from PI deep IRL (PIDIRL) against PI latent IRL (PILIRL), and PI linear IRL (PIIRL) with and without a-priori knowledge about the situation. First, we compare the model training based on distance convergence toward the human demonstration. Second, we compare the driving style of our policies under learned reward functions against human driving in different driving situations. Third, we analyze the reward weights inferred on the different test segments.

A. Training evaluation

We analyze the convergence of our model training based on a distance metric $d(\zeta, \pi)$ towards the human driven demonstration ζ. This distance metric is an integral of the Euclidean distance in longitudinal and lateral direction as well as the squared difference in the yaw angle over the policy as described in detail in [5]. Due to our goal of optimizing the likelihood of human driving behavior within the policy set Π, we measure our training convergence based on the expected distance (ED), given by $\mathbb{E}[d(\zeta, \Pi)] = \sum_{\pi \in \Pi} p(\pi|\boldsymbol{\theta}) d(\zeta, \pi)$. We plot ED for each IRL approach over an equal amount of training epochs. In Fig. 2 the ED for PIDIRL, PILIRL and PIIRL are plotted for 143 epochs. The PIDIRL shows a large ED reduction 78% over the entire training dataset. The starting ED is high due to the randomly initialized network, but a high reduction of 74% is achieved in only 22 epochs. PILIRL also has a high initial ED due to higher variance in weight initialization in order to find multiple different clusters, it finally shows a reduction of 74%. Due to the limited model complexity, PIIRL shows lowest ED reduction of 60%, here trained over the entire training dataset without a-apriori knowledge about the situations. The slight variance in the PIDIRL plot, particular in epoch 90, can be explained due to high learning rate and small batch size. From the statics above we can deduce that PIDIRL has the highest likelihood of selecting policies with humanlike driving behavior across different driving situations.

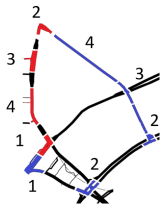
B. Driving style evaluation

In this part of the evaluation, we compare the driving style of PIDIRL, PILIRL, and PIIRL models against manual human driving over different test situations. We generate multiple PIIRL models with a-prior knowledge for each driving situations so as to compare the situation-dependent performance of the proposed PIDIRL. We plot the distance of the optimal policy obtained from theses models to the odometry of the human demonstration for different driving situations. Our test segments contain 187 planning cycles for segment 1, 232 for segment 2, 217 for segment 3, and 224 for segment 4. In Fig. 3c-f, we fit a Gaussian distribution over the segment specific distance histogram with 200 bins of size 0.0025. From our experiments we deduce that our proposed PIDIRL method produces policies with 21% lower mean distance as compared to PILIRL in seg. 1 and in seg. 2. In seg. 3 the mean distances from both of these methods are comparable. Moreover, PIDIRL shows 13% lower mean distance as compared to the segment specific PIIRL for seg. 3 and comparable results for the other segments. In seg. 4 PIDIRL produces policies with higher mean distances as compared to PIIRL and PILIRL. This trend can be described by the dominant prior of PILIRL and overfitting of PIIRL for straight segments. This poses as a disadvantage for both of theses methods as compared to our proposed PIDIRL which has better generalization capabilities. This generalization can be seen in the Fig. 3b, where PIDIRL has a similar mean distance over all segments. The reward function weights obtained from these models over the test situations are shown in Fig. 3h-k, where the weights are inversely proportional to the preference of that feature value in the policy. The reward weights are plotted in logarithmic scale. In these plots, it can be seen that PIDIRL produces nuanced reward functions for every planning cycle in the segment. The variance of the reward function produced by PIDIRL is proportional to the situation complexity. In simple situations, as shown in Fig. 3b at seg. 4, having a nuanced reward functions prediction mechanism like PIDIRL producing high variance as shown in Fig. 3k could lead to lower performance as compared to simpler reward functions namely PILIRL and PIIRL.

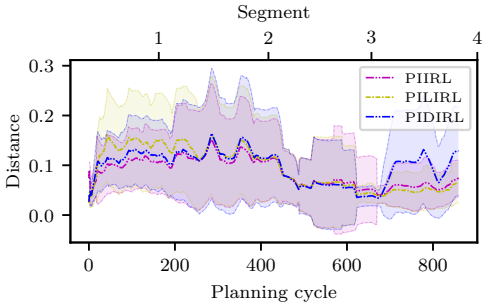
VIII. Conclusion and Future Work

We utilize path integral (PI) maximum entropy deep IRL to learn situation-dependent reward functions of a general-purpose planning algorithm. We propose a method to couple the reliability of a model-based planning system with the generalization capability of deep inverse reinforcement learning. Our experiments show that reward function predictions by our proposed neural network architecture are at par with multiple PI linear IRL reward function model that are trained with a-priori knowledge about the situation. In addition to this, we show that our deep IRL methodology has better generalization capabilities as compared to PI latent IRL which uses behavioral clustering of demonstrations. In future we plan to experiment with different network architectures in our proposed deep IRL paradigm so as to reduce the variance in reward functions prediction in simple driving situations.

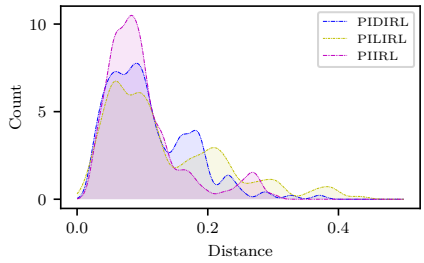
Label	Description
1	Sharp Turn
2	Stop, Start, Turn
3	Stop, Start
4	Lane follow



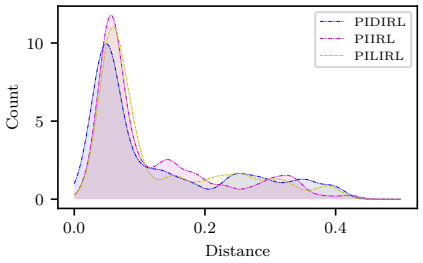
(a) The map depicts the training (blue) and test (red) segments of the selected Route in Hamburg City Center.



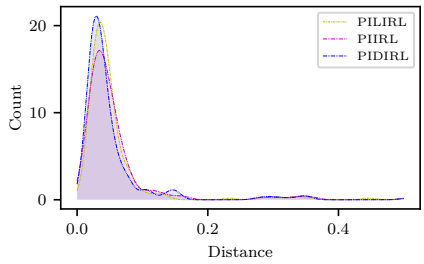
(b) Distance of the opt. policies on different driving segments.



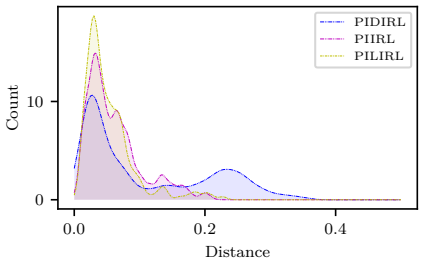
(c) Distribution of the opt. policy distances in seg. 1.



(d) Distribution of the opt. policy distances in seg. 2.



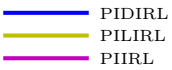
(e) Distribution of the opt. policy distances in seg. 3.

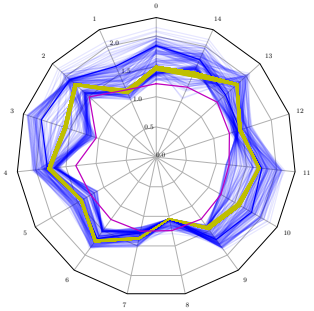


(f) Distribution of the opt. policy distances in seg. 4.

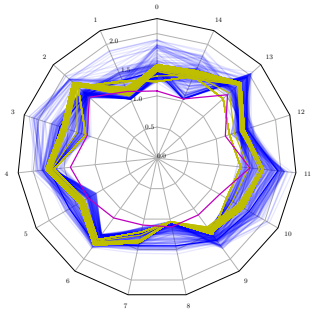
	Label	Feature
Kinematics	0	Long. Acc.
	1	Long. Jerk
	2	Long. Velocity
	3	Lat. Acc.
	4	Lat. Jerk
	9	Lat. Overshooting
	10	Lane Change Delay
Statics	5	Centerline
	6	Direction
	7	Proximity
	8	Curbs
Selection	11	State Class
	12	Manuver Space
	13	End Direction
	14	Min. Progress

(g) The table lists the considered PI features. The parameters are plotted in logarithmic scale.

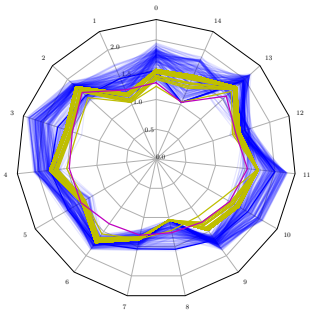




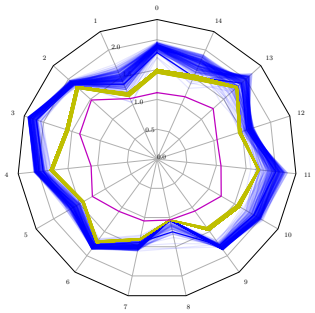
(h) Inf. reward weights on seg. 1.



(i) Inf. reward weights on seg. 2.



(j) Inf. reward weights on seg. 3.



(k) Inf. reward weights on seg. 4.

Fig. 3: *(a)* Tests conducted on a selected Route in Hamburg City center. *(b-f)* Graphs compares the performance of PIDIRL with PILIRL, and PIIRL over different driving segments. *(g)* Table of reward function features. *(h-k)* Graphs show the reward function predictions of our methods over driving segments.

REFERENCES

[1] S. Levine, "Reinforcement learning and control as probabilistic inference: Tutorial and review," *arXiv preprint arXiv:1805.00909*, 2018.

[2] A. Y. Ng and S. J. Russell, "Algorithms for Inverse Reinforcement Learning," in *Proc. Int. Conf. Machine Learning (ICML)*, 2000, pp. 663–670.

[3] M. Kuderer, S. Gulati, and W. Burgard, "Learning driving styles for autonomous vehicles from demonstration," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2015, pp. 2641–2646.

[4] S. Heinrich, A. Zoufahl, and R. Rojas, "Real-time trajectory optimization under motion uncertainty using a GPU," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots and Syst. (IROS)*, 2015, pp. 3572–3577.

[5] S. Rosbach, V. James, S. Großjohann, S. Homoceanu, and S. Roth, "Driving with style: Inverse reinforcement learning in general-purpose planning for automated driving," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots and Syst. (IROS)*, 2019, pp. 2658–2665.

[6] S. Heinrich, "Planning Universal On-Road Driving Strategies for Automated Vehicles," Ph.D. dissertation, Freie Universität Berlin, 2018.

[7] M. Wulfmeier, P. Ondruska, and I. Posner, "Maximum Entropy Deep Inverse Reinforcement Learning," *arXiv:1507.04888 [cs]*, p. 1507.04888, 2015.

[8] M. Wulfmeier, D. Rao, D. Z. Wang, P. Ondruska, and I. Posner, "Large-scale cost function learning for path planning using deep inverse reinforcement learning," in *Int. J. Robotics Research*, vol. 36, no. 10, pp. 1073–1087, 2017.

[9] M. McNaughton, "Parallel Algorithms for Real-time Motion Planning," Ph.D. dissertation, Carnegie Mellon University, 2011.

[10] S. Wang, D. Schuurmans, and Y. Zhao, "The Latent Maximum Entropy Principle," in *Proc. IEEE Int. Symp. on Inf. Theory ISIT*, 2002. pp. 131-.

[11] M. Babes, V. Marivate, K. Subramanian, and M. L. Littman, "Apprenticeship learning about multiple intentions," in *Proc. Int. Conf. Machine Learning (ICML)*, 2011, pp. 897–904.

[12] M. Shimosaka, K. Nishi, J. Sato, and H. Kataoka, "Predicting driving behavior using inverse reinforcement learning with multiple reward functions towards environmental diversity," in *IEEE Intell. Vehicles Symp.*, 2015, pp. 567–572.

[13] J. Choi and K.-E. Kim, "Nonparametric Bayesian inverse reinforcement learning for multiple reward functions," in *Adv. in Neural Inform. Process. Syst.*, 2012, pp. 305–313.

[14] C. Finn, S. Levine, and P. Abbeel, "Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization," in *J. of Machine Learning Research*, pp. 49–58, 2016

[15] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Adv. in Neural Inform. Process. Syst.*, 2016, pp. 4565–4573.

[16] C. M. Hruschka, D. Töpfer, and S. Zug, "Risk assessment for integral safety in automated driving," in *Int. Conf. on Intell. Autonomous Systems (ICoIAS)*, 2019, pp. 102–109.

[17] S. Arora and P. Doshi, "A survey of inverse reinforcement learning: Challenges, methods and progress," in *arXiv Preprint arXiv:1806.06877*, 2018.

[18] N. Aghasadeghi and T. Bretl, "Maximum entropy inverse reinforcement learning in continuous state spaces with path integrals," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots and Syst. (IROS)*., 2011. pp. 1561–1566.

[19] E. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning," in *J. of Machine Learning Research*, vol. 11, no., pp. 3137–3181, 2010.

[20] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum Entropy Inverse Reinforcement Learning." in *Proc. Nat. Conf. Artificial Intell. (AAAI)*, vol. 8, 2008. pp. 1433–1438.

[21] K. Homeier and L. Wolf, "RoadGraph: High level sensor data fusion between objects and street network," in *Proc. IEEE Int. Conf. Intell. Transp. Syst. (ITSC)*, 2011, pp. 1380–1385.

A.4 PLANNING ON THE FAST LANE: LEARNING TO INTERACT USING ATTENTION MECHANISMS IN INVERSE REINFORCEMENT LEARNING

Sascha Rosbach, Xing Li, Simon Großjohann, Silviu Homoceanu, and Stefan Roth
IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, USA, October 2020.

Abstract

General-purpose trajectory planning algorithms for automated driving utilize complex reward functions to perform a combined optimization of strategic, behavioral, and kinematic features. The specification and tuning of a single reward function is a tedious task and does not generalize over a large set of traffic situations. Deep learning approaches based on path integral inverse reinforcement learning have been successfully applied to predict local situation-dependent reward functions using features of a set of sampled driving policies. Sample-based trajectory planning algorithms are able to approximate a spatio-temporal subspace of feasible driving policies that can be used to encode the context of a situation. However, the interaction with dynamic objects requires an extended planning horizon, which depends on sequential context modeling. In this work, we are concerned with the sequential reward prediction over an extended time horizon. We present a neural network architecture that uses a policy attention mechanism to generate a low-dimensional context vector by concentrating on trajectories with a human-like driving style. Apart from this, we propose a temporal attention mechanism to identify context switches and allow for stable adaptation of rewards. We evaluate our results on complex simulated driving situations, including other moving vehicles. Our evaluation shows that our policy attention mechanism learns to focus on collision-free policies in the configuration space. Furthermore, the temporal attention mechanism learns persistent interaction with other vehicles over an extended planning horizon.

Copyright notice

Planning on the fast lane: Learning to interact using attention mechanisms in path integral inverse reinforcement learning

Sascha Rosbach[1,2], Xing Li[1], Simon Großjohann[1], Silviu Homoceanu[1] and Stefan Roth[2]

***Abstract*— General-purpose trajectory planning algorithms for automated driving utilize complex reward functions to perform a combined optimization of strategic, behavioral, and kinematic features. The specification and tuning of a single reward function is a tedious task and does not generalize over a large set of traffic situations. Deep learning approaches based on path integral inverse reinforcement learning have been successfully applied to predict local situation-dependent reward functions using features of a set of sampled driving policies. Sample-based trajectory planning algorithms are able to approximate a spatio-temporal subspace of feasible driving policies that can be used to encode the context of a situation. However, the interaction with dynamic objects requires an extended planning horizon, which depends on sequential context modeling. In this work, we are concerned with the sequential reward prediction over an extended time horizon. We present a neural network architecture that uses a policy attention mechanism to generate a low-dimensional context vector by concentrating on trajectories with a human-like driving style. Apart from this, we propose a temporal attention mechanism to identify context switches and allow for stable adaptation of rewards. We evaluate our results on complex simulated driving situations, including other moving vehicles. Our evaluation shows that our policy attention mechanism learns to focus on collision-free policies in the configuration space. Furthermore, the temporal attention mechanism learns persistent interaction with other vehicles over an extended planning horizon.**

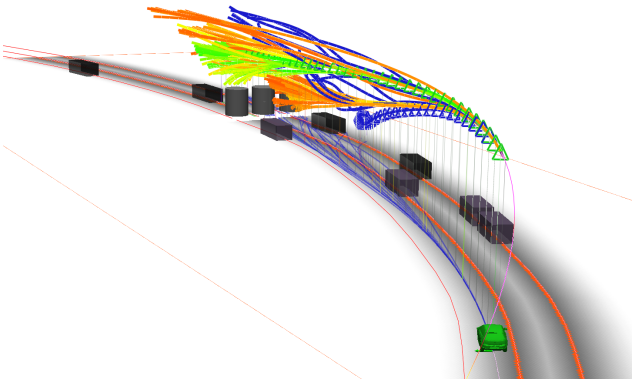


Fig. 1: Illustration of our planner for automated driving, which samples policies for our deep inverse reinforcement learning approach. The z-axis corresponds to the velocity, whereas the ground plane depicts spatial feature maps such as distances from the lane centers. A subset of policies is visualized, where the green triangle shows the optimal policy, and the blue triangles high-light the highest policy attention. The color gradient corresponds to the policy value. Blue policies show high attention activation. The cylindric objects represent a stop barrier.

I. Introduction

To drive in complex environments, automated vehicles plan in spatio-temporal workspaces. Sampling-based planning algorithms explore this workspace by sampling kinematically feasible actions. Encoding features of dynamic objects is challenging because interaction occurs over an extended planning horizon. Planning algorithms often rely on object predictions to derive features. During persistent maneuvers such as lane changes, automated vehicles mediate between a set of rewards from kinematics, infrastructure, behavior, and mission. A single reward function is often unable to evaluate a large set of heterogeneous driving situations. In this work, we focus on situation-dependent reward predictions using inverse reinforcement learning (IRL) that enables persistent behavior over an extended time horizon.

However, two challenges arise regarding the spatial and temporal dimensions: First, sampling a set of feasible driving policies often includes non-human-like trajectories that distort the assessment of the situational driving context. Second, sequence-based reward prediction requires an efficient context encoding over an extended time horizon. We propose a trajectory attention network that focuses on human-like trajectories to encode the driving context. Furthermore, we use this context vector in a sequence model to predict a temporal reward function attention vector. This temporal attention vector allows for stable reward transitions for upcoming planning cycles of a model-predictive control-based planner.

We evaluate the behavior of our approach in complex simulated driving situations over an oval course, including multiple lanes. The agent has to reach checkpoints, stop at stop signs, and has to interact by passing other vehicles that drive at lower velocities. We compare the reward predictions of our neural network architecture against baseline approaches using the expected value difference (EVD), expected distance (ED), and optimal policy distance (OPD) to the demonstrations. Our experiments show that we are able to produce stationary reward functions if the driving task does not change while at the same time addressing situation-

[1]The authors are with the Volkswagen AG, 38440 Wolfsburg, Germany {sascha.rosbach, xing.li, simon.grossjohann, silviu.homoceanu}@volkswagen.de

[2]The authors are with the Visual Inference Lab, Department of Computer Science, Technische Universität Darmstadt, 64289 Darmstadt, Germany stefan.roth@visinf.tu-darmstadt.de

dependent task switches with rapid response by giving the highest weight to the reward prediction of the last planning cycle.

II. Related Work

General-purpose planning algorithms combine mission, behavior, and local motion planning. These planning algorithms generate a set of driving policies in all traffic situations [1]. The policies are generated by sampling high-resolution actions based on action distributions that are derived from vehicle kinematics. A sequence of sampled actions can produce driving policies with complex implicit maneuvers, e.g., double lane-changes and merges in the time gap between two vehicles. The action sampling is achieved through massive parallelism on modern GPUs. In contrast to classical hierarchical planning systems, these approaches do not decompose the decision-making based on behavior templates [2]. Thus, the planning paradigm does not suffer from uncertain behavior planning that is often introduced due to insufficient knowledge about the underlying motion constraints. However, general-purpose planning systems require a reward function that evaluates the policy set in terms of kinematic and environment features in all driving situations. Specification and tuning of such a reward function is a tedious process that requires expert domain knowledge. Motion planning experts often rely on linear reward functions, which do not generalize over a large set of driving situations. The generalization of linear reward functions can be addressed by introducing an additional selection function of the final driving policy based on the generated policy set. During the selection, clustering and reasoning techniques can be used to discover maneuver patterns and evaluate the final policy [3]. We adopt the methodology of a sample-based general-purpose planning algorithm and focus on predicting local situation-dependent reward functions to scale over a large set of driving situations. In contrast to previous work, we do not use collision checking and features derived by post-sampling on the policy set [4]. Instead, we challenge the deep learning approach to predict situation-dependent reward functions and thereby control the overall driving task. Therefore, the interaction with infrastructure and dynamic vehicles is based on learned context representations.

In our previous work, we proposed a deep learning approach that predicts situation-dependent reward functions for such a sample-based planning algorithm. These planning algorithms operate in a model-predictive framework to address updates of the environment [4], [5]. The deep learning approach based on IRL uses features and actions of sampled driving policies to predict a set of linear reward function weights. The closed loop from sampled driving policies to reward function allows for dynamic updates of the reward weights over discrete planning cycles. However, continuous reward function switches may result in non-stationary behavior over an extended planning horizon. The authors found that the variance of the reward function prediction itself is proportional to the situational changes. In this work, we concentrate on persistent interaction with other moving vehicles over an extended time horizon, which can only be achieved if temporally consistent reward functions are predicted.

Planning and reinforcement learning algorithms for automated driving often solve a Markov-Decision Process (MDP) to find an optimal action sequence. The actions in automated driving are often represented as a tuple of wheel angle and acceleration. Sutton et al. introduced a temporal abstraction to such primitive actions in semi-MDPs, which are referred to as options [6]. Options are closed-loop policies for taking actions over a period of time, e.g., stay on a lane, change a lane to the left or right [7]. Similar to the temporal driving abstraction in reinforcement learning that has been presented by Shalev et al. [7], we utilize temporal abstraction in IRL. Previous work has investigated this hierarchical abstraction in IRL in terms of sub-task and sub-goal modeling using Mixture Models [8], [9]. In contrast to this work, we utilize sequential deep learning models to determine task transitions automatically.

In order to interact with dynamic objects, the planning algorithm operates on a spatio-temporal space, where a subspace is sampled based on kinematic feasibility. Path integral features for a policy are approximated during the action-sampling procedure and describe features of individual policies. In previous work, we focused on one dimensional (1D) convolutional neural network (CNN) architectures that generate a latent representation of trajectories [4]. The situation-dependent context description is encoded in fully-connected layers using latent trajectory features of the 1D-CNN block. The architecture's parameters largely depend on the policy set's size, which causes slow inference in recurrent models. The size of the policy set used to understand the spatio-temporal scene can be significantly reduced by concentrating on relevant policies with a human-like driving style. In this work, we use a policy attention mechanism to achieve this dimension reduction using a situational context vector.

Attention networks have gained significant interest in computer vision, natural language processing, and imitation learning [10]–[12]. Sharma et al. propose an attention-based model for action recognition in videos, which selectively focuses on parts of the video frames [13]. Fukui et al. use an attention branch to allow for visual explanation and improved performance in image recognition [14]. We utilize the visual explanation capabilities of an attention mask to explain which of the sampled driving policies are most relevant in every planning cycle. Wang et al. use an attention mechanism to learn unsupervised object segmentation [12]. They leverage the availability of affordable eye-tracking from human gazes to annotate objects. Similar to this work, we use odometry records as affordable labels to add supervised conditions on our situational context vector. Thereby, high attention on trajectories yields a proxy for closeness to expert demonstrations.

III. Preliminaries

Planning is often formulated as an MDP consisting of a 5-tuple $\mathcal{S}, \mathcal{A}, T, R, \gamma$, where $\mathcal{S}$ denotes the set of states,

and $\mathcal{A}$ describes the set of actions. In the domain of continuous control, an action a is integrated over time t using a transition function $T(s, a, s')$ for $s, s' \in \mathcal{S}, a \in \mathcal{A}$. Every action a in state s is evaluated using a reward function R that is discounted by γ over time t. The reward function uses features that are computed using an environment model and a vehicle transition model. The planner explores the subspace of feasible policies Π by sampling actions from a distribution conditioned on vehicle dynamics for each state s. The reward function is a linear combination of k static and kinematic features f_i with weight θ_i such that $\forall(s, a) \in \mathcal{S} \times \mathcal{A} : R(s, a) = \sum_{i \in K} -\theta_i f_i(s, a)$. The value of a policy V^π is the integral of discounted rewards during continuous transitions. The feature path integral f_i^π for a policy π is defined by $f_i^\pi = \int_t \gamma_t f_i(s_t, a_t)\, dt$. We project odometry records ζ of expert demonstrations into the state-action space to formulate a demonstration policy based on a Euclidean distance metric ensuring $\pi^D \in \Pi$. To extend the temporal planning horizon, a sequence of a-priori unknown reward functions can be defined as $R_{seq} = [R_{seq}^{(1)}, ..., R_{seq}^{(k)}]$. Similar to options in a semi-MDP, which are a generalization of primitive actions, a task can be decomposed into a sequence of subtasks, which depends on a preceding sequence $R_{seq}^{(k-1)}$. Thereby planning can be described in an MDP within a set of MDPs $\mathcal{M} = [M^{(1)}, ..., M^{(k)}]$, each having different reward functions $R^{(k)}$.

A. Maximum entropy PI deep IRL

IRL allows finding the reward function weights $\boldsymbol{\theta}$ that enable the optimal policy π^* to be at least as good as the demonstrated policy π^D [15]. The behavior of a demonstration is thereby indirectly imitated by the planning algorithm [16]. In path integral (PI) IRL, we formulate a probabilistic model that yields a probability distribution over policies, $p(\pi|\boldsymbol{\theta})$ [17], [18]. For each planning cycle, we optimize under the constraint of matching the expected PI feature values $\mathbb{E}_{p(\pi|\boldsymbol{\theta})}[\boldsymbol{f}^\pi]$ of the policy set Π and the empirical feature values $\hat{\boldsymbol{f}}^{\Pi^D}$ of the demonstrations. Imperfect demonstrations introduce ambiguities in the optimization problem, which Ziebart et al. [19] propose to solve by maximizing the entropy of the distribution. The policy distribution is given by

$$p(\pi|\boldsymbol{\theta}) = \frac{1}{Z} \exp(-\boldsymbol{\theta}^\top \boldsymbol{f}^\pi). \tag{1}$$

Due to the exponential growth of the state-action space it is often intractable to compute the partition function $Z = \sum_{\pi \in \Pi} \exp(-\boldsymbol{\theta}^\top \boldsymbol{f}^\pi)$. We approximate the partition function by sampling driving policies similar to Markov chain Monte Carlo methods. Maximizing the entropy of the distribution over policies subject to the feature constraints from demonstrated policies implies that the log-likelihood $L(\boldsymbol{\theta})$ of the observed policies under the maximum entropy distribution is maximized. In previous work, we formulated a deep learning approach for PI maximum entropy IRL, which approximates a complex mapping between PI features $\boldsymbol{f}_k^\Pi$, actions $\boldsymbol{a}_k^\Pi$ and reward function weights $\boldsymbol{\theta}_{k+1}$ at MPC cycles k, given by $\boldsymbol{\theta}_{k+1} \approx g(\boldsymbol{\Theta}, \boldsymbol{f}_k, \boldsymbol{a}_k)$.

The IRL problem can be formulated in the context of Bayesian inference as maximum a posteriori estimation, which entails maximizing the joint posterior distribution of observing expert demonstrations Π^D. We calculate the maximum entropy probability based on the linear reward weights $\boldsymbol{\theta}$, which are inferred by the network with parameters $\boldsymbol{\Theta}$ as

$$L(\boldsymbol{\theta}) = L(g(\boldsymbol{\Theta}, \boldsymbol{f}, \boldsymbol{a})) = \sum_{\pi^D \in \Pi^D} \ln p(\pi^D | g(\boldsymbol{\Theta}, \boldsymbol{f}, \boldsymbol{a})). \tag{2}$$

The gradient for the log-likelihood $L(\boldsymbol{\theta})$ can be calculated in terms of $\boldsymbol{\Theta}$ as

$$\begin{aligned} \frac{\partial L}{\partial \boldsymbol{\Theta}} &= \frac{\partial L}{\partial \boldsymbol{\theta}} \cdot \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\Theta}} \\ &= \Big[\sum_{\pi \in \Pi} p(\pi|\boldsymbol{\theta}) \boldsymbol{f}^\pi - \hat{\boldsymbol{f}}^{\Pi^D} \Big] \cdot \frac{\partial}{\partial \Theta} g(\boldsymbol{\Theta}, \boldsymbol{f}, \boldsymbol{a}). \end{aligned} \tag{3}$$

The gradient is separated into the maximum entropy gradient in terms of $\boldsymbol{\theta}$ and the gradient of $\boldsymbol{\theta}$ w.r.t. the network parameters $\boldsymbol{\Theta}$, which can be directly obtained via backpropagation in the deep neural network.

B. Open-loop reward learning

Training IRL algorithms is often time-consuming. The MDP has to be solved with respect to the current reward function in the inner loop of reward learning. We reduce the time constraint by running our planning algorithm prior to training with a randomly initialized reward function $\boldsymbol{\theta}_0$. This allows us to generate a buffer of policy sets with corresponding features and actions. Sampling high-resolution actions allows us to project odometry records ζ in the state-action space. We use a weighted Euclidean distance metric calculation in the sampling procedure to evaluate distances of policies to the odometry of the expert trajectories [5]. The weighted Euclidean metric was evaluated against sequence alignment methods such as dynamic time warping and found to be sufficient to select demonstration policies. In addition to the path integral features $\boldsymbol{f}^\pi$ of a policy, we use features of the policies at time-equidistant control points $\boldsymbol{c}^\pi$. The feature sequence includes the lateral offsets with reference to the ego position, yaw, and progress along the route. The progress is calculated using Dijkstra's algorithm on the road-network after receiving the target destination. The training algorithm is run for a predefined number of epochs, ensuring that the convergence metrics, the EVD and ED, reach the desired threshold value. For each epoch, the training dataset is shuffled and divided into batches to perform mini-batch gradient descent.

IV. Neural Network Architecture

We propose a deep learning architecture for PI deep IRL. This architecture uses inputs of PI features f^Π, actions a^Π, and a sequence of spatio-temporal features in the form of policy control points c^Π. Our deep IRL architecture is

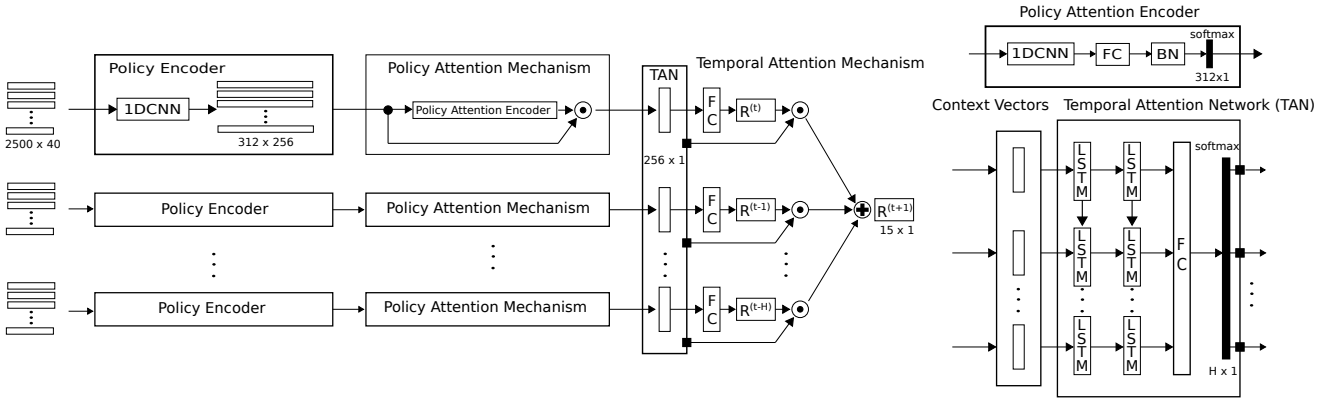


Fig. 2: Neural network architectures for situation-dependent reward prediction. Policy temporal attention architecture consisting of policy attention and temporal attention mechanism. Inputs are a set of planning cycles, each having a set of policies. Policy encoder generates a latent representation of individual policies. Policy attention mechanism produces a low-dimensional context vector, which is forwarded to the temporal attention network (TAN). Policy temporal attention mechanism predicts a mixture reward function given a history of context vectors.

separated into a policy attention mechanism and a temporal attention mechanism, as shown in Fig. 2.

A. Policy Attention

The policy attention mechanism generates a 1D context vector of the situation. We feed the policy sets into a policy encoder, which relies on 1DCNN layers to generate latent features of individual policies. The combined policy encoder and policy attention mechanism are referred to as policy attention CNN (PACNN). A policy attention encoder uses combinations of 1D convolutions, average pooling, and fully-connected layers to compute a policy attention vector. Our attention vector is based on the soft attention mechanism [10]. We perform a softmax operation over the output of the attention encoder network to generate a 1D vector. The attention vector essentially filters non-human-like trajectories from the policy encoder. We combine the maximum entropy IRL gradient with a semi-supervised attention loss [12]. The semi-supervised loss is based on the mean absolute distance towards the expert demonstration. To compute the loss, we sort the policies in ascending order of progress along the route. Sorting enables a consistent relationship between attention loss and the sampled policy set distribution. The output of spatial attention is multiplied by a learned scalar [20]. The scalar learns cues in the local neighborhood and gradually assigns more weight to non-local evidence. The maximum entropy gradient is calculated based on the policy set of the input distribution [4]. We use 1D average upsampling of the attention vector to match the dimensionality of policy sets. This allows us to visualize the trajectory attention during inference.

B. Temporal attention

In a second training step, we use context vectors of our PACNN networks and the corresponding situation-dependent reward functions to predict the reward functions for the next planning cycle at time $t+1$. We do so by taking a sequential history size H of context vectors and reward functions into account. The temporal attention network consists of a two-layered recurrent long short-term memory (LSTM) network and a fully-connected network of four layers. The output is a 1D weight vector computed by a softmax activation function. The final reward function is a mixture of situation-dependent reward functions $R^{(t+1)} = \sum_{h \in H} w_h R^{(h)}$. In contrast to the PACNN network, the temporal attention network PTACNN learns to predict by training with the maximum entropy gradient of the future planning cycle at time $t+1$. This architecture allows for long sequence lengths and fast inference during the prediction due to a low dimensional context vector. The overall idea is similar to expectation-maximization (EM) IRL, which uses a mixture of clustered reward functions to infer a situation-dependent reward function given features of the demonstrations [21]. In contrast to the mixture model, we infer a mixture of sequential reward functions based on a latent context description of the situations.

V. Experiments

We conducted our experiments on complex simulated scenarios. The driving situations on our oval course are designed in a way that a lap completion requires continuous task predictions. The oval map includes multiple lanes, as depicted in Fig. 1. Four checkpoints provide a proxy for the target locations on the course; checkpoints are toggled from inner to outer lanes to enforce mission-oriented lane-changes. There are multiple exits on the oval, which make the mission evaluation a requirement. On two locations of the oval, stop signs span over all lanes to assess stopping, starting, and making progress along the route. At most 15 vehicles are spawned at random at a distance of 200 m from the ego vehicle. The vehicles drive with constant velocity if they do not interact with other vehicles or infrastructure.

The spawning velocity is selected at random in the range of 25 - 35 kph. The agent's target velocity is set to 70 kph, which requires constant mediation between strategic, behavioral, and motion-related reward features. Due to the large velocity difference between the agent and other vehicles, it is expected that the agent drives aggressively. In the presence of 15 other vehicles, this implies that the agent has to learn how to merge into small gaps between vehicles and pass without colliding.

A. Data collection and simulation

We collect expert driving demonstrations by recording the optimal policies of an expert-tuned planning algorithm. The expert-tuned planner uses a manually tuned reward function and a model-based trajectory selection. Similar to Gu et al. [3], the expert-tuned planner uses topological clustering and additional features that are computed on the policy set to derive the final driving policy. A crucial input for the selection is the progress value of policies along the route. This feature gets the vehicle moving and influences mission-oriented lane-changes. Once the odometry of the expert-tuned planning algorithm is recorded, the model-based selection, and its additional features are disabled. This is done to test if learned context vectors are able to encode latent features of the policy set, which allow the indirect expert-planner imitation. During data collection, the odometry of the expert-tuned optimal policies are recorded. We utilize the same data collection principle, as in [4], [5]. The odometry records are projected into the state space to formulate geometrically close demonstrations π^D for the training, validation, and test dataset. For our training datasets, we do not assume prior knowledge of the reward function, therefore solve the MDP using a random reward function. For our tests on sequential datasets, we collect policy sets using the expert-tuned planner. By projecting the odometry of the expert optimal policies in state space during testing, we achieve a proxy for perfect imitation.

B. Reward feature representation

The reward function features are computed during the action sampling procedure and describe vehicle motion, infrastructure, and time-dependent distances to objects. We consider 15 manually engineered features. Infrastructural features are derived from street networks [22]. Derivatives of lateral and longitudinal actions describe the vehicle kinematics. Lane change dynamics are described by lane change delay and lateral overshooting. The lane change delay punishes performing lane changes at the end of the planning horizon. Spatio-temporal proximity is calculated from object motion predictions.

C. Baseline approaches

All of the baseline approaches use the path integral IRL training methodology and produce a linear combination of reward weights. We consider a linear IRL (LIRL) approach and two non-recurrent deep IRL neural network architectures as baseline methods. The neural networks generate a latent context representation of the input policy distribution. The 1DCNN architecture uses fully-connected layers to encode the context from latent policy features; we refer to this architecture as 1DCNN [4]. An alternative architecture uses 1D convolutions over latent features to decrease the neural network parameters. This architecture is referred to as Bi1DCNN.

TABLE I: Overview of average test performance based on ED, and OPD. Tests are conducted on a test dataset, recorded by an expert-tuned planning algorithm. Number of trainable variables of the neural networks are listed and split for the PTACNN networks into the policy (left) and temporal (right) attention network parameters.

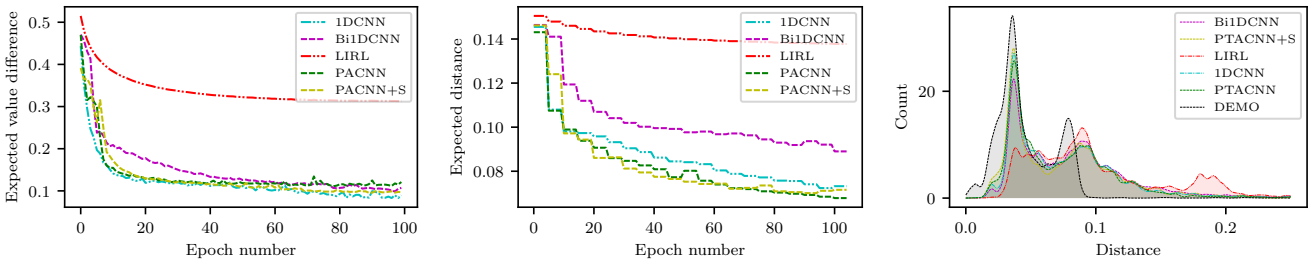
Approach	Trainable variables	$\overline{ED}$	$\overline{OPD}$
LIRL	15	0.121	0.116
Bi1DCNN	18.7 M	0.105	0.096
1DCNN	22.6 M	0.094	0.088
PTACNN	2.94 M + 1.61 M = 4.55 M	**0.092**	**0.086**
PTACNN+S	2.94 M + 1.61 M = 4.55 M	**0.091**	**0.081**

VI. Evaluation

We evaluate the performance of our proposed spatio-temporal attention networks against our baseline approaches. First, we evaluate the context encoding capabilities of the approaches. Here, we concentrate on the convergence of our PACNN network against neural-networks without such an attention mechanism. The convergence is analyzed in terms of EVD and ED on training and validation datasets over training epochs. Second, we compare the sequential prediction performance in terms of the optimal policy distance (OPD) to the expert-demonstration on a playback test dataset. Furthermore, our supplementary video displays the closed-loop reward function prediction and driving performance in challenging driving situations.

A. Comparison with expert-demonstrations

Fig. 3 depicts the training, validation, and test results of our evaluated methods. All methods in the convergence plot are trained using the maximum entropy gradient of the trajectory input distribution. During validation and testing, we calculate the EVD, ED, OPD based on the inferred reward function using a history size $H = 10$ for all methods. This means that all methods except the PTACNN use a mean of inferred reward weights over the history size. We configured the planning algorithm so that it yields approximately 2.500 policies during each planning cycle. Fig. 3a represents the convergence of our training, which is measured by EVD over epochs [5]. In the EVD calculation, the value is normalized by the value of the demonstration, since the weights may increase their range over the training epochs. We abort training after achieving a high ED and EVD reduction and observe the weight distributions over the epochs. In our training dataset, we use one hour of driving demonstrations, which provide approximately 17.000 planning cycles and an equal amount of expert-demonstrations π^D. We split our evaluation dataset into validation and hold out test dataset.



(a) Training: Convergence on a non-sequential training dataset based on EVD.

(b) Validation: Convergence on a sequential validation dataset based on ED.

(c) Test: Distance of policy to the demonstration on a sequential test dataset.

Fig. 3: Training and test results of our proposed methods in contrast to baseline approaches. *(a)* Convergence on a non-sequential training dataset based on EVD. *(b)* Convergence on a non-sequential validation dataset based on ED. *(c)* Distance distribution of optimal policies to the expert demonstrations on a sequential test dataset. The lower bound for the distance distribution is given the distance of the demonstration (DEMO). During the sequential prediction all deep learning approaches use a history size $H = 10$.

PACNN+S and PTACNN+S have been trained using an additional semi-supervised loss based on the mean absolute policy distance to the demonstration. We calculate the EVD every epoch and perform validation every fifth epoch.

All deep IRL methods converge to a similar EVD, in contrast to LIRL, which is unable to fit a single reward function yielding low EVD. Bi1DCNN converges after 100 epochs of training with an ED of 0.1. PACNN, PACNN+S, and 1DCNN converge at a close ED proximity at a value of 0.07. Using a semi-supervised loss in addition to the maximum entropy gradient did not improve nor decrease the training results in terms of EVD significantly. All deep IRL approaches show a similar peak in the OPD distribution, akin to the demonstration, as depicted in Fig. 3c. The distance distribution of the demonstration is the lower bound for the OPD. The nonzero lower bound is caused by discretization errors of the state-action space and the planner's nondeterministic optimization methodology and can be regarded as a proxy for perfect imitation. In addition to the distribution, we summarize the test results in Table I. PTACNN+S is trained in a second stage using the context vector and reward predictions of PACNN+S. The generalization of a single reward function is not achieved, as shown in the ED reduction and OPD on the test set. The performance of 1DCNN and Bi1DCNN models on the validation set is proportional to the trainable variables after latent feature extraction using 1DCNNs. 1DCNN uses fully-connected layers to learn a context representation. In contrast to PACNN, Bi1DCNN learns a set of filters over latent variables of policies. The attention networks stand out, having fewer parameters and a low-dimensional context vector while yielding similar performance as compared to larger neural network architectures. PACNN uses eight times less trainable variables than 1DCNN and six times less trainable variables than Bi1DCNN. PTACNN performs best on the test dataset, yet the evaluation of persistent reward predictions using temporal attention requires a closed-loop inference.

B. *Visualization of attention mechanisms*

During inference, the attention mechanisms can be visualized, as depicted in Fig. 1. The blue trajectories show the trajectories with the highest policy attention activation. A color gradient is assigned to the policy value that ranges from green (high) to red (low). Only a subset of all feasible policies is visualized. Our video shows the driving performance during closed-loop inference of our proposed method. In the video, we display two additional figures. A radar chart depicts the predicted reward function weights, and a bar chart shows the temporal attention activation. PTACNN is able to control the complete driving task and interacts with other vehicles without relying on model-based collision checking.

VII. Conclusion and Future Work

In this work, we propose a deep network architecture that is able to predict situation-dependent reward functions for a sample-based planning algorithm. Our architecture uses a temporal attention mechanism to predict reward functions over an extended planning horizon. This is achieved by generating a low dimensional context vector of the driving situation from features and actions of sampled-driving policies. Our experiments show that our attention mechanisms outperform our baseline deep learning approaches during comparisons against expert-demonstrations. In closed-loop inference, our approach is able to control the complete driving task in challenging situations while only learning from one hour of driving demonstrations. In future, we plan to train the algorithm on a large scale dataset and use it in combination with model-based constraints in real-world driving situations. Furthermore, we want to integrate raw sensory data into the deep inverse reinforcement learning approach so as to learn relevant features of the environment automatically.

REFERENCES

[1] M. McNaughton, "Parallel Algorithms for Real-time Motion Planning," Ph.D. dissertation, Carnegie Mellon University, 2011.

[2] S. Heinrich, "Planning Universal On-Road Driving Strategies for Automated Vehicles," Ph.D. dissertation, Freie Universität Berlin, 2018.

[3] T. Gu, J. M. Dolan, and J.-W. Lee, "Automated tactical maneuver discovery, reasoning and trajectory planning for autonomous driving," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Syst. (IROS)*, Daejeon, South Korea, 2016.

[4] S. Rosbach, V. James, S. Großjohann, S. Homoceanu, X. Li, and S. Roth, "Driving style encoder: Situational reward adaptation for general-purpose planning in automated driving," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, Paris, France, 2020.

[5] S. Rosbach, V. James, S. Großjohann, S. Homoceanu, and S. Roth, "Driving with style: Inverse reinforcement learning in general-purpose planning for automated driving," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Syst. (IROS)*, Macau, China, Nov 2019, pp. 2658–2665.

[6] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[7] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," in *Learning, Inference and Control of Multi-Agent Syst. Workshop (NIPS)*, 2016.

[8] S. Krishnan, A. Garg, R. Liaw, L. Miller, F. T. Pokorny, and K. Goldberg, "Hirl: Hierarchical inverse reinforcement learning for long-horizon tasks with delayed rewards," *arXiv preprint arXiv:1604.06508*, 2016.

[9] A. Šošić, A. M. Zoubir, E. Rueckert, J. Peters, and H. Koeppl, "Inverse reinforcement learning via nonparametric spatio-temporal subgoal modeling," *The Journal of Machine Learning Research*, vol. 19, no. 1, pp. 2777–2821, 2018.

[10] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Int. Conf. Learning Representations ICLR*, Y. Bengio and Y. LeCun, Eds., San Diego, USA, 2015. [Online]. Available: http://arxiv.org/abs/1409.0473

[11] Y. Duan, M. Andrychowicz, B. Stadie, O. J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, "One-shot imitation learning," in *Adv. in Neural Inform. Process. Syst.*, 2017, pp. 1087–1098.

[12] W. Wang, H. Song, S. Zhao, J. Shen, S. Zhao, S. C. Hoi, and H. Ling, "Learning unsupervised video object segmentation through visual attention," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 3064–3074.

[13] S. Sharma, R. Kiros, and R. Salakhutdinov, "Action recognition using visual attention," *arXiv preprint arXiv:1511.04119*, 2015.

[14] H. Fukui, T. Hirakawa, T. Yamashita, and H. Fujiyoshi, "Attention branch network: Learning of attention mechanism for visual explanation," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 10 705–10 714.

[15] S. Arora and P. Doshi, "A survey of inverse reinforcement learning: Challenges, methods and progress," in *arXiv Preprint arXiv:1806.06877*, 2018.

[16] A. Y. Ng and S. J. Russell, "Algorithms for Inverse Reinforcement Learning," in *Proc. Int. Conf. Machine Learning (ICML)*, 2000.

[17] N. Aghasadeghi and T. Bretl, "Maximum entropy inverse reinforcement learning in continuous state spaces with path integrals," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Syst. (IROS)*. IEEE, 2011.

[18] E. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning," in *Int. J. Machine Learning Research*, vol. 11, no. Nov, 2010, pp. 3137–3181.

[19] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum Entropy Inverse Reinforcement Learning." in *Proc. Nat. Conf. Artificial Intell. (AAAI)*, vol. 8, 2008.

[20] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," *arXiv preprint arXiv:1805.08318*, 2018.

[21] M. Babes, V. Marivate, K. Subramanian, and M. L. Littman, "Apprenticeship learning about multiple intentions," in *Proc. Int. Conf. Machine Learning (ICML)*, 2011, pp. 897–904.

[22] K. Homeier and L. Wolf, "RoadGraph: High level sensor data fusion between objects and street network," in *Proc. IEEE Int. Conf. Intell. Transp. Syst. (ITSC)*, 2011, pp. 1380–1385.

A.5 PIXEL STATE VALUE NETWORK FOR COMBINED PREDICTION AND PLANNING IN INTERACTIVE ENVIRONMENTS

Sascha Rosbach, Stefan Leupold, Simon Großjohann, and Stefan Roth
Available as *arXiv Preprint*, arXiv:2310.07706 [cs.RO], October 2023.

Abstract

Automated vehicles operating in urban environments have to reliably interact with other traffic participants. Planning algorithms often utilize separate prediction modules forecasting probabilistic, multimodal, and interactive behaviors of objects. Designing prediction and planning as two separate modules introduces significant challenges, particularly due to the interdependence of these modules. This work proposes a deep learning methodology to combine prediction and planning. A conditional GAN with the U-Net architecture is trained to predict two high-resolution image sequences. The sequences represent explicit motion predictions, mainly used to train context understanding, and pixel state values suitable for planning encoding kinematic reachability, object dynamics, safety, and driving comfort. The model can be trained offline on target images rendered by a sampling-based model predictive planner, leveraging real-world driving data. Our results demonstrate intuitive behavior in complex situations, such as lane changes amidst conflicting objectives.

Pixel State Value Network for Combined Prediction and Planning in Interactive Environments

Sascha Rosbach[1,2], Stefan M. Leupold[1], Simon Großjohann[1] and Stefan Roth[2]

***Abstract*— Automated vehicles operating in urban environments have to reliably interact with other traffic participants. Planning algorithms often utilize separate prediction modules forecasting probabilistic, multi-modal, and interactive behaviors of objects. Designing prediction and planning as two separate modules introduces significant challenges, particularly due to the interdependence of these modules. This work proposes a deep learning methodology to combine prediction and planning. A conditional GAN with the U-Net architecture is trained to predict two high-resolution image sequences. The sequences represent explicit motion predictions, mainly used to train context understanding, and pixel state values suitable for planning encoding kinematic reachability, object dynamics, safety, and driving comfort. The model can be trained offline on target images rendered by a sampling-based model-predictive planner, leveraging real-world driving data. Our results demonstrate intuitive behavior in complex situations, such as lane changes amidst conflicting objectives.**

I. Introduction

State-of-the-art automated driving functions utilize a sequential processing chain consisting of perception, prediction, planning, and control. In this chain, the interdependence of prediction and planning presents a major challenge. It is often neglected that the actions of the automated vehicle influence the behavior of other traffic participants. This work concentrates on the joint prediction of both object behavior (other agents) and ego behavior (the automated vehicle).

Due to inherent uncertainty in real-world driving situations, it would be required to provide a planner with probabilistic multi-modal predictions that jointly provide a coherent future. However, implementing the desired driving style based on uncertain motion predictions is difficult and an error-prone task. In contrast to related work in motion prediction, we do not predict trajectories but, instead, focus on predicting two time-dependent image sequences implicitly encoding joint object and ego behavior. High-resolution image sequences allow us to represent arbitrary distributions over a pixel state space. The first sequence contains future pixel visitations of other agents. We use it as a training target to force the network to form a coherent world view but we do not utilize it for planning. The second sequence encodes the value of a position for the automated vehicle in space and time and can be directly used by a planner. The deep learning task is posed as paired image-to-image prediction task [1] and uses four bird's eye view images as input that depict information about road infrastructure and the current state of objects and ego vehicle in the environment. The architecture is depicted in Fig. 1.

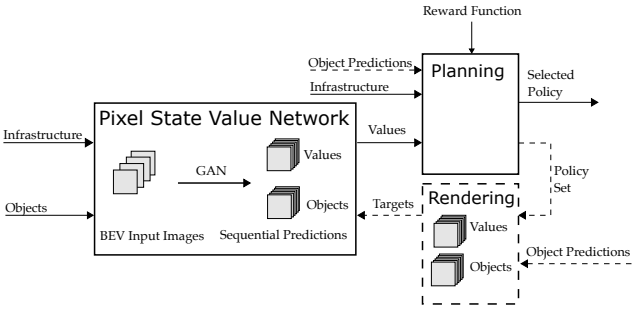


Fig. 1: Shows the proposed architecture comprised of pixel state value network (PSVN), planning, and rendering module. The rendering module uses policy sets and object predictions to generate targets for offline training of the PSVN. The PSVN infers pixel state values for the planning module. Object predictions and the rendering module (dashed lines) are not required during inference.

In this work, we propose a novel imitation learning methodology that is able to scale in terms of experience and resolution of the state evaluation. The approach can be divided into two stages: The first stage is concerned with rendering image pairs and the second stage focuses on training the network offline. In the first stage, we create our dataset by replaying 24 hours of real-world driving data and use our 'exhaustive' model-predictive planner to render megapixel-sized value images using inputs from perception, localization, and predictions from queried object motion. In the second stage, we train a conditional GAN offline on the dataset (approx. 700 k images).

We perform a qualitative evaluation of the image-to-image translation in challenging scenarios. Further, we perform a quantitative evaluation of collision avoidance and desired trajectory selection by contrasting the implicit object handling via predicted pixel state values against a traditional processing chain relying on explicit object predictions.

II. Related Work

Traditionally, planning algorithms rely on prediction modules providing the future motion of traffic participants in the form of trajectories. State-of-the art motion forecasting approaches perform multi-modal predictions of agents using deep learning yielding a set of trajectories [2]–[5]. A number

[1]The authors are with CARIAD SE, 38440 Wolfsburg, Germany {sascha.rosbach, stefan.leupold, simon.grossjohann}@cariad.technology

[2]The authors are with the Visual Inference Lab, Department of Computer Science, Technische Universität Darmstadt, 64289 Darmstadt, Germany stefan.roth@visinf.tu-darmstadt.de

of public benchmark datasets [6]–[9] have been proposed, setting the primary focus on the task of trajectory prediction with performance measured against ground truth. Planning algorithms use predictions to check for collisions [10] and derive features such as proximities and time gaps over the planning horizon [11]. These features are used in the reward function to minimize risk and optimize the desired behavior. However, deriving such features based on uncertain behavior predictions is difficult. In this work, we train a deep model that directly provides state values for planning in dynamic environments to produce the desired driving behavior. This circumvents the problem of generating interactive explicit object predictions by learning through observation of these features computed on recorded behaviors.

Inverse Reinforcement Learning (IRL) is a methodology that concentrates on finding the unknown reward function for planning and Reinforcement Learning (RL) agents. Maximum Entropy Deep IRL (MEDIRL) predicts a reward map encoding the spatial traversability for path planning [12]. Lee et al. propose an extension to MEDIRL for trajectory planning by predicting multiple time-dependent maps [13]. This allows to anticipate the behavior of moving objects implicitly without manually specifying a cost-function representation. However, reward maps generated by MEDIRL suffer from noise and artifacts due to utilized pixel state visitation frequency matching between demonstrations and model. This fundamental problem gets more severe when demonstrations are distributed on multiple target maps. Lee et al. propose a penalty loss for unvisited pixels to overcome these shortcomings [13]. However, the dimensionality of the observable space is limited, because the higher the spatio-temporal resolution, the less feedback received from the expert demonstrations. Wulfmeier et al. suggest first using human priors such as handcrafted reward maps to pretrain the model and then fine-tune using MEDIRL to address this issue [14]. Our work addresses these problems and proposes a new methodology to discriminate states in a high-resolution spatio-temporal pixel space. Furthermore, we address the shortcoming of IRL being time-consuming to train because learning a representation for the reward function requires planning in the inner loop of reward learning. This gives our approach characteristics of value equivalence learning in Model-based Reinforcement Learning [15]–[18]. However, our aim is to utilize learned state values to perform reward function shaping and later tune the influence using path integral IRL [19].

A stream of literature aims to learn a holistic end-to-end model for driving by predicting a spatio-temporal cost volume for planning [20], [21]. The authors train the approach using real-world driving data and define a max-margin planning loss function using the ground-truth ego trajectory as positive example and 100 randomly sampled trajectories as negative examples [21]. This allows them to discriminate good from bad trajectories. We do not utilize demonstrations of the ego vehicle but, instead, 'exhaustively' sample kinematically feasible actions leading to trajectories having different behaviors (approx. 14 k trajectories per situation) and utilize the cost function of our planner to discriminate trajectories and also give our model the possibility to learn about reachability and collision avoidance.

III. Methodology

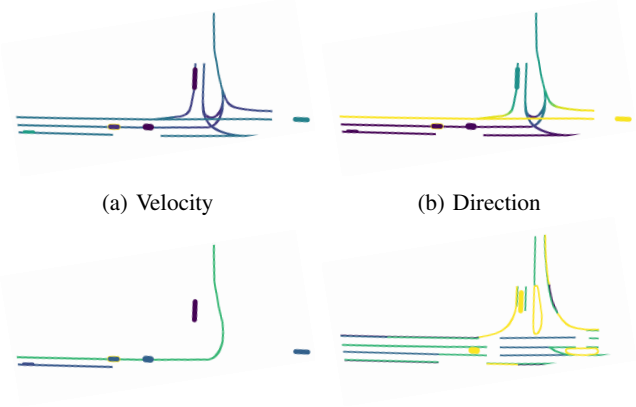
In this work, we propose a deep learning methodology for the combined prediction of both object behavior (other agents) and ego behavior (the automated vehicle). The training is formulated as a paired image-to-image translation task based on bird's eye view images of the environment. We first explain the image generation process that uses an exhaustive model-predictive planner at its core which is able to generate megapixel-sized target images. Furthermore, we propose a conditional GAN [1] using the U-Net architecture [22] to translate the image inputs to a sequence of images covering space and time. Last, we outline a hybrid mode of operation that combines value inference and sample-based planning.

A. Image Pair Generation

Our approach allows using real-world data recordings. As of now, we chose to insert a layer of abstraction from raw sensory data. We assume access to a perception, a localization, and a road graph module to create bird's eye view input images. All images are rendered with one-megapixel resolution. The input consists of four images with a single channel each, stacked together as can be seen in Fig. 2. The first layer contains road infrastructure and velocity information for all objects including the ego vehicle. Road infrastructure is drawn with 10 pixel wide lines depicting the centerlines. Its color values correspond to speed limits and precomputed maximal velocities depending on road curvature and acceptable lateral acceleration. The ego vehicle is highlighted by an outline in the first three layers. The second layer depicts the direction of centerlines and all objects relative to the ego vehicle's orientation. The third layer contains accelerations of all objects and centerlines that are reachable by the ego vehicle. Centerlines are enumerated by color and the navigational target lane is highlighted. The fourth layer contains static vehicles and obstacles such as road boundaries classified as, e.g., dashed, solid, or curb. The deep neural network receives square images of size 512×512. We resize the images without distortion by always rotating onto the diagonal to maintain the highest possible resolution in a square image.

Target image generation can be separated into two algorithms: planning and rendering. Planning is performed at approximately 5 Hz frequency on inputs of real-world driving recordings. Based on reprocessed data using perception, localization, and tracking, the planning algorithm generates a set of policies Π and their corresponding values V^{Π}. These are used by the rendering algorithm to produce images that approximate a value function over pixel coordinates.

1) Planning algorithm: The planning algorithm is based on massively parallel search and runs on a graphics processing unit (GPU) [11], [23]. Its pseudo code is given in Algo. 1. Implementation details of the planning algorithm have been published in our previous work on path integral



(a) Velocity (b) Direction

(c) Target Lane (d) Boundaries

Fig. 2: Depictions of the input representation for the neural network: *(a)* Velocities of objects and speed limits of centerlines. *(b)* Directions of objects and centerlines. *(c)* Accelerations of objects and target lane. *(d)* Static objects and boundaries.

IRL [19], [24], [25]. The algorithm starts by sampling a discrete number of continuous actions $\mathcal{A}_s$ (combining wheel angles and accelerations) for all states $s \in \mathcal{S}_t$. For each action, a transition is set up that uses fourth-order time-continuous polynomial functions for velocity and wheel angle profiles. The sampling distribution for each state is conditioned on feasible vehicle dynamics and the coefficients of the polynomials are derived from actor-friendly continuity constraints. The planner can generate diverse driving behavior by combining transitions of one to two seconds over the planning horizon of six seconds and more. This allows the policy set Π to includes multiple behaviors, e.g., lane following, lane changes, swerving, and emergency stops. To address the curse of dimensionality, a pruning operation removes redundant states that are evaluated based on state similarity, kinematic, and infrastructure based features. Our work makes use of a reward function with $K = 25$ handcrafted features that can be categorized into motion, infrastructure, and object related components [19], [24], [25]. The planning algorithm yields a policy set Π and computes a value $V^\pi = \int_t \gamma_t R(s_t, a_t)\, dt$ for each $\pi \in \Pi$. We increase the state action space during the target image generation phase compared to deployment. This allows generating a large set of approx. 14 k policies (with a planning horizon of 6.6 sec. and vehicle model integration steps of 0.2 sec. this provides each situation with approx. 450 k states) that can be utilized to render dense pixel state value images.

2) Rendering algorithm: The rendering algorithm generates a sequence of images by drawing each of the planner's policies. The color of the pixels is derived from the policy value V^π and discriminates pixels on the interval of [0,1] in each situation. The red, yellow, green coloring in Fig. 3 can be seen as an example. The images have a resolution of one megapixel and the same aspect ratio as the input images. The scaling and aspect ratio are determined by optimizing a convex hull around all object positions at time zero, the ego position, and a safety corridor around the reachable set of the last planning cycle. This provides situation adjusted, high-resolution pixel state value readings during inference.

A sequence of images has six temporal layers l for a planning horizon of 6.6 seconds with intervals T_l as follows:

$$T_l = \ ](l-1) \times 1.1, l \times 1.1],\ l \in \{1, \ldots, 6\}. \quad (1)$$

Before rendering, we calculate the probability P of selecting a policy π using the Maximum Entropy principle [26] based on the value V^π as $P(V^\pi) = Z^{-1}\exp(-\beta V^\pi)$, where the partition function is defined by $Z = \sum_{\pi \in \Pi} \exp(-\beta V^\pi)$ and the temperature parameter β. The planner acts as an approximation method to the partition function similar to Markov chain Monte Carlo methods [19]. We discard collisions in the calculation of the distribution to focus on the kinematically feasible, non-colliding set for which the features of the reward function provide a coherent ranking. We manually tune the temperature β to balance the planner solution bias and uncertainty of the distribution in the reachable set.

Each pixel in each layer image receives the maximal value of any policy passing through that pixel at any time. The resulting function for each pixel's color value can be written as

$$V(p_x, p_y, l) = \max_{\pi \in \Pi, t \in T_l} P_\pi(p_x, p_y, t) \quad (2)$$

$$p_x \in \{1, \ldots, \text{width}\},\ p_y \in \{1, \ldots, \text{height}\}. \quad (3)$$

We calculate pixel coordinates from Cartesian coordinates of the policy every 0.2 seconds, which provides start and end points for a Bresenham [27] line drawing algorithm connecting the pixels. The rendering algorithm is explained in Algo. 2.

B. Image-to-Image Prediction

We propose a conditional Generative Adversarial Network (GAN) training methodology to perform image-to-image prediction. The deep neural network architecture utilized is based on an Encoder-Decoder U-Net architecture as in Pix2Pix [1]. The network is modified to take input images of size $512 \times 512 \times 4$ and produce images of size $512 \times 512 \times 12$ in contrast to the original architecture using a spatial resolution of 256×256. Target images are scaled to the interval $[0, 1]$ with a cut-off at 0.1 for the lowest possible reachable values. The generator is trained using hard sigmoid activation functions to assign values of zero to pixels out of the time-dependent non-colliding reachable set and values of one to the best pixels of the set.

The capacity of the generator is increased by using an additional down- and upsampling layer, each having 512 filters resulting in a total of 67 M parameters. The discriminator has an additional downsampling layer with 512 filters with a total of 7 M parameters. We resize the 512×512 square output images again to the target aspect ratio, resulting in a resolution of one megapixel. The neural network must adapt to various scaling factors due to the planning algorithm's

Algorithm 1: Planning Algorithm

Input: planning horizon H, model M, reward function R, discount γ
Output: policy set Π

function planning(H, M, R, γ)
  $\mathcal{S}_0 \leftarrow$ initial state
  for $t \in H$ **do**
    $\mathcal{S}_t \leftarrow$ get set of states at time t
    forall $s \in \mathcal{S}_t$ **do**
      $\mathcal{A}_s \leftarrow$ sample set of actions
      forall $a \in \mathcal{A}_s$ **do**
        $s' \leftarrow$ integrate transition $T(s, a)$
        $f(s, a) \leftarrow$ observe features
        $R(s, a) = -\theta^\top f(s, a)$
        $V(s') \leftarrow V(s) + \gamma R(s', a)$
        $\mathcal{S}_{t'} \leftarrow \mathcal{S}_{t'} \cup \{s'\}$
    $\mathcal{S}_{t'} \leftarrow$ prune $\mathcal{S}_{t'}$
  $\Pi \leftarrow$ get policies in $\mathcal{S}, \mathcal{A}$
  return policies Π

Algorithm 2: Rendering Algorithm

Input: policies Π with access to states S and transitions T
Output: value images I

function rendering(Π)
  $I \leftarrow$ init value image sequence
  forall π *in* Π **do**
    $p^\pi \leftarrow$ get probability of policy $P(V^\pi)$
    for $T \in \pi$ **do**
      integrate transition T with integation step Δt
      foreach integration step from s_a to s_b **do**
        $c_a \leftarrow$ get pixel coordinate of $I(s_a)$
        $c_b \leftarrow$ get pixel coordinate of $I(s_b)$
        max bresenham from c_a to c_b with value $P(V^\pi)$
  return I

situation-dependent choices regarding the meters-to-pixel scale and image aspect ratio.

C. Planning with the PSVN

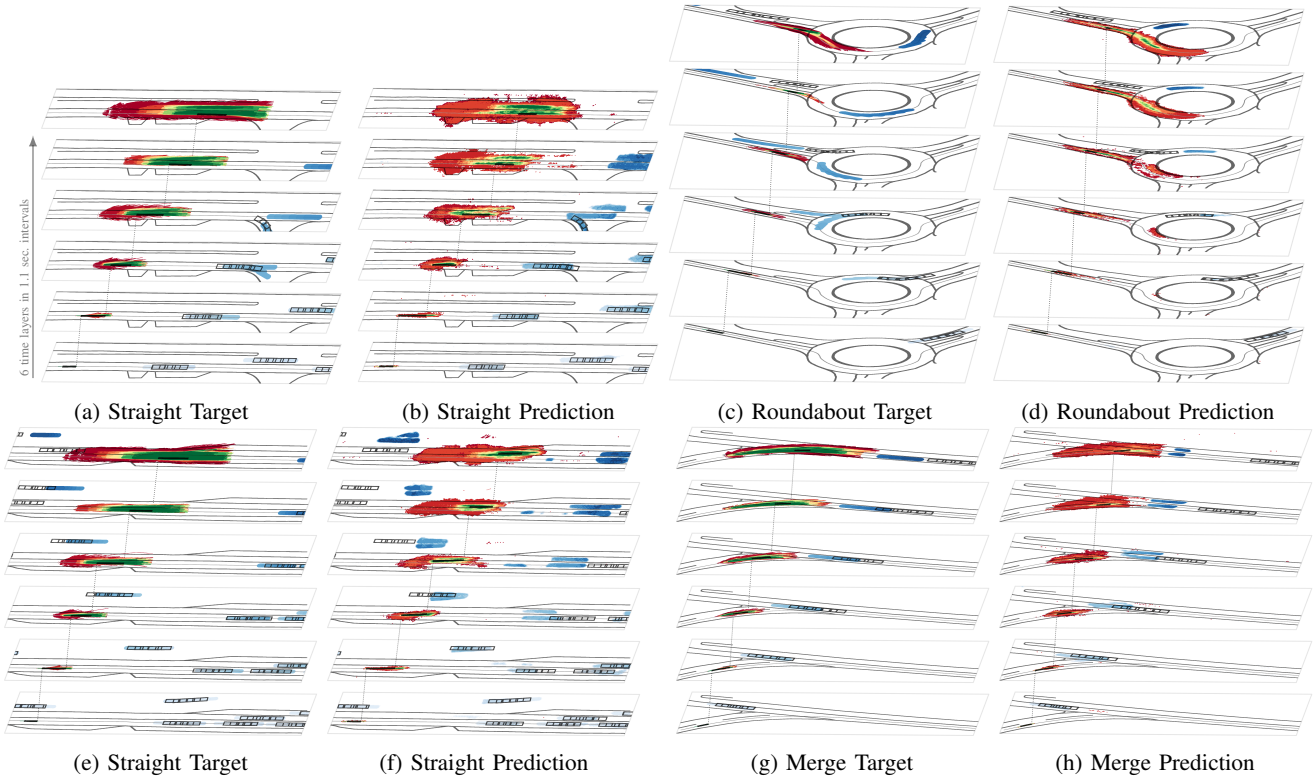
When using the PSVN the infrastructure and objects are provided in bird's eye view representation as depicted in Fig. 1. The PSVN directly provides pixel state values $V(p_x, p_y, l)$ in a sequence of images encoding values for planning, replacing explicit object predictions that are usually required to plan in interactive environments. Recall that pixel values are represented in the scaled interval $[0, 1]$, where a value of 0 represents unreachable or colliding pixel states and 1 pixels being part of the optimal policy having received the $\max_{\pi \in \Pi, t \in T_l} P_\pi(p_x, p_y, t)$ pixel update. This work uses the pixel values to perform reward shaping for our planning algorithm listed in Algo. 1 to improve the optimal policy selection [28]. Therefore supplementing the missing behavioral information in features $f(s, a)$ of the reward function that only depend on s and action a with the PSVN values encoding the preference of following a desired behavior in a situation. The new reward function of the planner is given by $R' = R + F$, where the reward shaping function F is obtained by reading and integrating pixel values encountered on the continuous transition from state s to s' using $log(V(p_x, p_y, l))$. The continuous transitions are integrated using a vehicle model, and at every intermediate step, the Cartesian state coordinates are converted into pixel coordinates. The new reward function R' combines prior knowledge such as vehicle kinematics and distances to infrastructure in R with situation-dependent abstract behavioral information given by the reward shaping function F such as how the ego vehicle has to behave with respect to other objects.

IV. Experiments

We show experiments that exhibit the combined prediction and planning capabilities of the proposed network in interactive driving environments. Our experiments are divided into a qualitative analysis of the predicted image sequences and a quantitative evaluation of the policy selection. The quantitative evaluation contrasts the performance of a traditional architecture separating prediction and planning with the proposed approach combining PSVN with planning. The training dataset contains 24 hours of real-world driving data from 35 different drivers on a predetermined route. The driving data includes mostly urban situations with a short rural road section. Following the route requires multiple lane changes, left and right turns at traffic lights, and roundabouts covering each exit possibility. The datasets are filtered discarding situations having less than one other vehicle involved. In the following experiments, we perform a simplification of the implementation by using an object prediction module as described in the traditional sequential chain to render the training dataset targets. This allows us to disregard perception and tracking uncertainties at this stage of this work. Furthermore, this simplifies the quantitative evaluation allowing a direct comparison of the separated prediction and planning architecture with the proposed approach combining PSVN with planning. We augment the data by shifting the ego vehicle forward, closer to preceding vehicles, increasing the ego state velocity, and randomizing the position and orientation relative to the centerline. All other input information remains unmodified. The training dataset is rebalanced based on the aspect ratio of the target images, which provide a proxy to different situations, e.g., squared images in roundabouts and rectangular in straights. Inputs for tests are generated in a simulation in separate geographical locations compared to the training dataset covering similar situations. The test dataset contains a roundabout and multi-lane road segments. Lane changes are required to follow the route and adhere to user inputs such as preference for adjacent lanes. Setting the preference for neighboring lanes in the presence of other moving vehicles allows us to generate difficult scenarios with conflicting objectives between route navigation and object interaction.

A. Qualitative Examples

First, we show qualitative examples of the combined prediction of object motion and ego vehicle behavior in interactive urban traffic environments. The images contrast the model's prediction with rendered target images. Both prediction and target are depicted including overlays of ground-truth object positions (black rectangles), infrastructure, and the selected trajectory in black passing through all time layers. We show a few selected situations with



(a) Straight Target (b) Straight Prediction (c) Roundabout Target (d) Roundabout Prediction

(e) Straight Target (f) Straight Prediction (g) Merge Target (h) Merge Prediction

Fig. 3: Displays targets *(a,c,e,g)* and predictions *(b,d,f,h)* for straights, roundabout and merge situations. Motion predictions of objects are displayed in blue color. State values are depicted in red, yellow, and green colors. Multiple overlays are displayed in grey (boundaries, target lane, ground-truth objects). The trajectories passing through time layers are drawn in black. The trajectories in *(a,c,e,g)* are selected based on the predicted state value images and in *(b,d,f,h)* are based on the trajectory values.

parallel and merging lanes, as well as a roundabout, which require particular emphasis on object motion prediction and timing of the interactive behavior. Figs. 3a and 3b display an often encountered situation with two parallel lanes, where the ego vehicle has to follow a preceding vehicle. The preceding vehicle might take the exit or continue heading in the same direction as the ego vehicle. Both prediction and target provide multi-modal behavior in state values and object motion and both choose to follow the vehicle in the current lane instead of an unnecessary lane change. Figs. 3e and 3f depict a similar situation with both lanes occupied by preceding vehicles. The network correctly distinguishes objects based on their driving direction and predicts the oncoming traffic. However, the timing of the explicit motion predictions do not match the ground truth; specifically, the network estimates objects to move slower. The optimal policy selected sorely on predicted values performs a lane change, which is feasible in this situation to gain more progress along the route. Figs. 3g and 3h depict a merging situation, where the ego vehicle has to judge the velocity and direction of the moving vehicle in order to determine whether to move behind or in front of the vehicle. Our network is able to generate a similar motion prediction as the target until 3.3 sec., as depicted in the third time-dependent image. Thereafter, the proposed network generates two hypotheses for the motion prediction and positions itself behind. Figs. 3c and 3d demonstrate a roundabout with the object taking the first exit. The predicted state values focus around the center-line of the roundabout, displaying that arbitrary geometries can potentially be predicted. The solution candidate of the prediction shows preference for not entering the roundabout until more information is revealed of the objects' intention.

B. Quantitative Evaluation

In our quantitative evaluation, we contrast the policy selection of the planner using the training setup with object predictions available against the inference setup only using the PSVN as input. After sampling a policy set for each situation while neglecting object related features, we select two policies by computing the $\max_{\pi \in \Pi} V^{\pi}$. The first policy is selected by using the reward function $R' = R$ that includes explicit object prediction features (the traditional chain). The second policy is selected by solely using the reward shaping function $R' = F$ (the output of the PSVN). We purposefully neglect all state and action rewards $R(s, a)$ of R' to analyze the encoding capabilities of the inferred pixel state values and focus on object interaction. Metrics that compare against ground truth trajectories are problematic for evaluation of

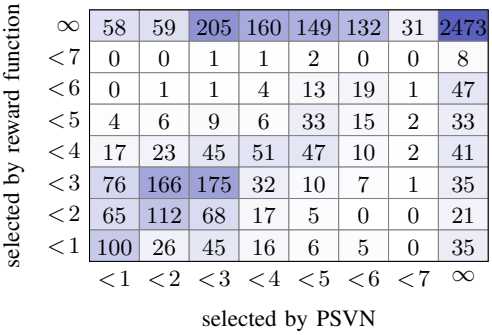
interactive behavior as in most situations multiple behaviors are valid. Therefore we propose to use a different evaluation metric that is termed object time gap (OTG). It is defined for a policy π as

$$\mathrm{OTG}(\pi) = \min_{\substack{obj \in O \\ t_{obj} \in [0,H] \\ t_\pi \in [0,H]}} |t_\pi - t_{obj}| \; s.t. \left\| \begin{pmatrix} x_{t_\pi} - x_{t_{obj}} \\ y_{t_\pi} - y_{t_{obj}} \end{pmatrix} \right\|^2 < 2,$$

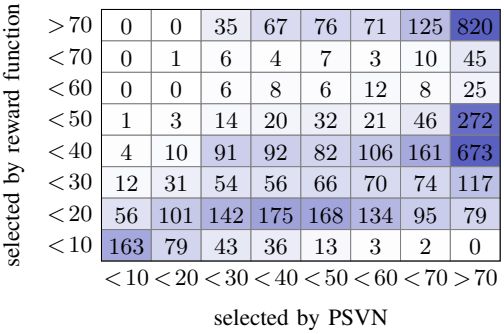
where O is the set of all objects in that scenario, x and y are the Cartesian coordinates of an object or the ego vehicle at that time t. An object time gap of 0 sec. represents the ego vehicle and an object reaching the same place at the same time (collision). A 3 sec. time gap means that one object and the ego vehicle reach the same place with a 3 sec. time difference. If no path overlap exists, the object time gap results in a value of ∞. Fig. 4a shows in the upper right corner that in most scenarios both the trajectory selected by the planner reward function and the trajectory selected by the predicted values never overlap with objects. It can be seen in our supplementary video that there are always vehicles present in the situations. This means that the selected policies do not display reckless behaviors such as driving into vehicles on parallel lanes. Many scenarios fall into the diagonal where both selected trajectories display correlating OTG. Another accumulation can be seen on the first row, where the reward function based selection correctly avoids objects but the predicted value map produces more aggressive behavior. The second confusion matrix (see Fig. 4b) displays the travel distance of a trajectory relative to the initial position. Values are accumulated slightly below the diagonal. It visualizes that in many scenarios the prediction based selection results in a further progressing trajectory in comparison to the reward function based selection. This shows that trajectories selected by our prediction value successfully leave large enough object time gaps or completely avoid collisions while not trivially standing still.

V. Limitations and Future Work

As of now, this work assumes access to an a-priori defined reward function with features encoding information about ego motion, infrastructure, and dynamics of the environment. The reward function for planning is a linear weighing of the relevance of these features. We aim to increase the quality of rendered target images by using situation-dependent reward function parameters [24], [25]. We see a lot of potential in substituting the segmentation focused network architecture with a recurrent architecture that is more capable of capturing dynamics. We have shown that our training methodology is able to generate high-resolution situation-dependent state values and motion predictions for a diverse set of environments. We see a lot of potential when using our methodology to pretrain a network to generate an encoding of the situation and fine-tune the network for other tasks such as predicting actions or reward functions. We proposed using PSVN for reward shaping, however aim to incorporate predicted pixel



(a) Object Time Gap [s]



(b) Progress of Trajectory [m]

Fig. 4: Confusion matrices contrasting optimal policy of planner using the reward function and PSVN for 4732 test scenarios that each contain objects.

state values as admissible heuristic to accelerate search and reduce the GPU memory requirements for the set of states and actions when deploying in vehicle.

VI. Conclusion

This work proposes a methodology for combined pixel state value and object motion prediction. A conditional GAN is trained to predict both a sequence of value and object motion images given a bird's eye view of the environment. The proposed planning and rendering algorithms make the generation of one megapixel-sized pixel state value image sequences tractable. This allows training the generative model with large image pair datasets. Our training algorithm uses consumer grade hardware and does not require interaction with the environment, which allows us to utilize real-world driving recordings. The results demonstrate confident multi-modal pixel state value and object predictions in situations requiring moderate timing capabilities amidst a large set of moving objects and a promising outlook for complex interactive situations such as roundabouts and merging situations.

Acknowledgments

This research was supported by Federal Ministry for Economic Affairs and Climate Action in the national research project RUMBA under grant number 19A20007L.

REFERENCES

[1] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks," *arXiv:1611.07004 [cs]*, Nov. 2018.

[2] W. Luo, C. Park, A. Cornman, B. Sapp, and D. Anguelov, "JFP: Joint future prediction with interactive multi-agent modeling for autonomous driving," in *Conference on Robot Learning (CoRL)*. PMLR, 2023, pp. 1457–1467.

[3] B. Varadarajan, A. Hefny, A. Srivastava, K. S. Refaat, N. Nayakanti, A. Cornman, K. Chen, B. Douillard, C. P. Lam, D. Anguelov *et al.*, "Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 7814–7821.

[4] S. Shi, L. Jiang, D. Dai, and B. Schiele, "MTR-A: 1st place solution for 2022 Waymo open dataset challenge–motion prediction," *arXiv preprint arXiv:2209.10033*, 2022.

[5] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde, "Thomas: Trajectory heatmap output with learned multi-agent sampling," *arXiv preprint arXiv:2110.06607*, 2021.

[6] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan *et al.*, "Argoverse: 3d tracking and forecasting with rich maps," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 8748–8757.

[7] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. K. Pontes *et al.*, "Argoverse 2: Next generation datasets for self-driving perception and forecasting," *arXiv preprint arXiv:2301.00493*, 2023.

[8] S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. R. Qi, Y. Zhou *et al.*, "Large scale interactive motion forecasting for autonomous driving: The Waymo open motion dataset," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*, 2021, pp. 9710–9719.

[9] W. Zhan, L. Sun, D. Wang, H. Shi, A. Clausse, M. Naumann, J. Kummerle, H. Konigshof, C. Stiller, A. de La Fortelle *et al.*, "Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps," *arXiv preprint arXiv:1910.03088*, 2019.

[10] S. M. LaValle, *Planning algorithms*. Cambridge University Press, 2006.

[11] M. McNaughton, *Parallel algorithms for real-time motion planning*. Carnegie Mellon University, 2011.

[12] M. Wulfmeier, D. Rao, D. Z. Wang, P. Ondruska, and I. Posner, "Large-scale cost function learning for path planning using deep inverse reinforcement learning," *IJRR*, vol. 36, no. 10, pp. 1073–1087, Sep. 2017.

[13] K. Lee, D. Isele, E. A. Theodorou, and S. Bae, "Spatiotemporal costmap inference for MPC via deep inverse reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3194–3201, 2022.

[14] M. Wulfmeier, D. Rao, and I. Posner, "Incorporating human domain knowledge into large scale cost function learning," *arXiv preprint arXiv:1612.04318*, 2016.

[15] C. Grimm, A. Barreto, S. Singh, and D. Silver, "The value equivalence principle for model-based reinforcement learning," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 5541–5552, 2020.

[16] C. Grimm, A. Barreto, G. Farquhar, D. Silver, and S. Singh, "Proper value equivalence," *Advances in Neural Information Processing Systems*, vol. 34, pp. 7773–7786, 2021.

[17] J. Oh, S. Singh, and H. Lee, "Value prediction network," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[18] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel *et al.*, "Mastering Atari, Go, chess and Shogi by planning with a learned model," *Nature*, vol. 588, no. 7839, pp. 604–609, 2020.

[19] S. Rosbach, V. James, S. Großjohann, S. Homoceanu, and S. Roth, "Driving with Style: Inverse Reinforcement Learning in General-Purpose Planning for Automated Driving," *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2658–2665, Nov. 2019.

[20] W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun, "End-to-end interpretable neural motion planner," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 8660–8669.

[21] W. Zeng, S. Wang, R. Liao, Y. Chen, B. Yang, and R. Urtasun, "DSDNet: Deep structured self-driving network," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXI 16*. Springer, 2020, pp. 156–172.

[22] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer, 2015, pp. 234–241.

[23] S. Heinrich, *Planning universal on-road driving strategies for automated vehicles*. Springer, 2018, vol. 119.

[24] S. Rosbach, V. James, S. Großjohann, S. Homoceanu, X. Li, and S. Roth, "Driving style encoder: Situational reward adaptation for general-purpose planning in automated driving," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. Paris, France: IEEE, May 2020, pp. 6419–6425.

[25] S. Rosbach, X. Li, S. Großjohann, S. Homoceanu, and S. Roth, "Planning on the fast lane: Learning to interact using attention mechanisms in path integral inverse reinforcement learning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5187–5193.

[26] B. D. Ziebart, A. L. Maas, J. Bagnell, and A. Dey, "Maximum entropy inverse reinforcement learning," in *AAAI*, 2008.

[27] J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Systems Journal*, vol. 4, no. 1, pp. 25–30, 1965.

[28] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Icml*, vol. 99, 1999, pp. 278–287.

BIBLIOGRAPHY

Abbeel, Pieter and Andrew Ng (2004). "Apprenticeship learning via inverse reinforcement learning." In: *Proceedings of the International Conference on Machine Learning (ICML)*, p. 1.

Arora, Saurabh and Prashant Doshi (2021). "A survey of inverse reinforcement learning: Challenges, methods and progress." In: *Journal of Artifical Intelligence* 297, p. 103500.

Babes, Monica, Vukosi Marivate, Kaushik Subramanian, and Michael Littman (2011). "Apprenticeship learning about multiple intentions." In: *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 897–904.

Badue, Claudine, Rânik Guidolini, Raphael Carneiro, Pedro Azevedo, Vinicius Cardoso, Avelino Forechi, Luan Jesus, Rodrigo Berriel, Thiago Paixao, Filipe Mutz, Lucas de Paula Veronese, Thiago Oliveira-Santos, and Alberto De Souza (2021). "Self-driving cars: A survey." In: *Expert Systems with Applications* 165, p. 113816.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2015). "Neural machine translation by jointly learning to align and translate." In: *International Conference on Learning Representations (ICLR)*.

Bansal, Mayank, Alex Krizhevsky, and Abhijit Ogale (2019). "ChauffeurNet: Learning to drive by imitating the best and synthesizing the worst." In: *Proceedings of the Robotics: Science and Systems Conference (RSS)*.

Bar-Tal, Omer, Hila Chefer, Omer Tov, Charles Herrmann, Roni Paiss, Shiran Zada, Ariel Ephrat, Junhwa Hur, Guanghui Liu, Amit Raj, Yuanzhen Li, Michael Rubinstein, Tomer Michaeli, Oliver Wang, Deqing Sun, Tali Dekel, and Inbar Mosseri (2024). "Lumiere: A space-time diffusion model for video generation." In: *arXiv Preprint*. arXiv:2401.12945 [cs.CV].

Bast, Hannah, Daniel Delling, Andrew Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato Werneck (2016). "Route planning in transportation networks." In: *Algorithm Engineering - Selected Results and Surveys*. Vol. 9220. Springer, pp. 19–80.

Bellman, Richard (1961). *Adaptive control processes: A guided tour*. Princeton University Press.

– (1966). "Dynamic programming." In: *Science* 153.3731, pp. 34–37.

Bharilya, Vibha and Neetesh Kumar (2024). "Machine learning for autonomous vehicle's trajectory prediction: A comprehensive survey, challenges, and future research directions." In: *Vehicular Communications* 46, p. 100733.

Bojarski, Mariusz, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba (2016). "End-to-end learning for self-driving cars." In: *arXiv Preprint*. arXiv:1604.07316 [cs.CV].

Brechtel, Sebastian, Tobias Gindele, and Rüdiger Dillmann (2011). "Probabilistic MDP-behavior planning for cars." In: *IEEE Transactions on Intelligent Transportation Systems*, pp. 1537–1542.

Buehler, Martin, Karl Iagnemma, and Sanjiv Singh (2009). *The DARPA urban challenge: Autonomous vehicles in city traffic*. Vol. 56. Springer.

Caesar, Holger, Varun Bankiti, Alex Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom (2020). "Nuscenes: A multimodal dataset for autonomous driving." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11621–11631.

Caruana, Rich (1997). "Multitask learning." In: *Machine Learning* 28, pp. 41–75.

Casas, Sergio, Abbas Sadat, and Raquel Urtasun (2021). "MP3: A unified model to map, perceive, predict and plan." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14403–14412.

Chang, Ming-Fang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays (2019). "Argoverse: 3D tracking and forecasting with rich maps." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8748–8757.

Chen, Chenyi, Ari Seff, Alain Kornhauser, and Jianxiong Xiao (2015). "Deepdriving: Learning affordance for direct perception in autonomous driving." In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 2722–2730.

Chen, Li, Penghao Wu, Kashyap Chitta, Bernhard Jaeger, Andreas Geiger, and Hongyang Li (2023). "End-to-end autonomous driving: Challenges and frontiers." In: *arXiv Preprint*. arXiv:2306.16927 [cs.RO].

Choi, Jaedeug and Kee-Eung Kim (2012). "Nonparametric bayesian inverse reinforcement learning for multiple reward functions." In: *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 305–313.

Codevilla, Felipe, Matthias Müller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy (2018). "End-to-end driving via conditional imitation learning." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4693–4700.

Cui, Alexander, Sergio Casas, Abbas Sadat, Renjie Liao, and Raquel Urtasun (2021). "Lookout: Diverse multi-future prediction and planning for self-driving." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16107–16116.

Delling, Daniel, Peter Sanders, Dominik Schultes, and Dorothea Wagner (2009). "Engineering route planning algorithms." In: *Algorithmics of Large and Complex Networks*. Springer, pp. 117–139.

Dempster, Arthur, Nan Laird, and Donald Rubin (1977). "Maximum likelihood from incomplete data via the EM algorithm." In: *Journal of the Royal Statistical Society (Series B)* 39.1, pp. 1–22.

Dhariwal, Prafulla and Alexander Nichol (2021). "Diffusion models beat gans on image synthesis." In: *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 8780–8794.

Ding, Zhezhang and Huijing Zhao (2023). "Incorporating driving knowledge in deep learning based vehicle trajectory prediction: A survey." In: *IEEE Transactions on Intelligent Vehicles* 8.8, pp. 3996–4015.

Do Carmo, Manfredo (2016). *Differential geometry of curves and surfaces: revised and updated second edition*. Courier Dover Publications.

Dolgov, Dmitri, Sebastian Thrun, Michael Montemerlo, and James Diebel (2008). "Practical search techniques in path planning for autonomous driving." In: *Ann Arbor* 1001.48105, pp. 18–80.

Drews, Paul, Grady Williams, Brian Goldfain, Evangelos Theodorou, and James Rehg (2017). "Aggressive deep driving: Combining convolutional neural networks and model predictive control." In: *Conference on Robot Learning (CoRL)*, pp. 133–142.

Elander, James, Robert West, and Davina French (1993). "Behavioral correlates of individual differences in road-traffic crash risk: An examination of methods and findings." In: *Psychological Bulletin* 113.2, p. 279.

Ettinger, Scott, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles Qi, Yin Zhou, Zoey Yang, Aurelien Chouard, Pei Sun, Jiquan Ngiam, Vijay Vasudevan, Alexander McCauley, Jonathon Shlens, and Dragomir Anguelov (2021). "Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9710–9719.

Evans, Leonard (1991). *Traffic safety and the driver*. Science Serving Society.

Finn, Chelsea, Paul Christiano, Pieter Abbeel, and Sergey Levine (2016a). "A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models." In: *arXiv Preprint*. arXiv:1611.03852 [cs.LG].

Finn, Chelsea, Sergey Levine, and Pieter Abbeel (2016b). "Guided cost learning: Deep inverse optimal control via policy optimization." In: *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 49–58.

Fuller, Ray (2000). "The task-capability interface model of the driving process." In: *Recherche-Transports-Sécurité* 66, pp. 47–57.

– (2011). "Driver control theory: From task difficulty homeostasis to risk allostasis." In: *Handbook of Traffic Psychology*. Elsevier, pp. 13–26.

Galceran, Enric, Alexander Cunningham, Ryan Eustice, and Edwin Olson (2017). "Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction: Theory and experiment." In: *Autonomous Robots* 41, pp. 1367–1382.

Gao, Nan, Hao Xue, Wei Shao, Sichen Zhao, Kyle Qin, Arian Prabowo, Mohammad Rahaman, and Flora Salim (2022). "Generative adversarial networks for spatio-temporal data: A survey." In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 13, pp. 1–25.

Gaver, William W (1991). "Technology affordances." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 79–84.

Gilles, Thomas, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde (2021). "HOME: Heatmap output for future motion estimation." In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pp. 500–507.

Golchoubian, Mahsa, Moojan Ghafurian, Kerstin Dautenhahn, and Nasser Azad (2023). "Pedestrian trajectory prediction in pedestrian-vehicle mixed environments: A systematic review." In: *IEEE Transactions on Intelligent Transportation Systems* 24.11, pp. 11544–11567.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep learning*. MIT press.

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). "Generative adversarial nets." In: *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2672–2680.

Gronauer, Sven and Klaus Diepold (2022). "Multi-agent deep reinforcement learning: A survey." In: *Artificial Intelligence Review* 55.2, pp. 895–943.

Gu, Tianyu, John Dolan, and Jin-Woo Lee (2016). "Automated tactical maneuver discovery, reasoning and trajectory planning for autonomous driving." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5474–5480.

Gui, Jie, Tuo Chen, Jing Zhang, Qiong Cao, Zhenan Sun, Hao Luo, and Dacheng Tao (2024). "A survey on self-supervised learning: Algorithms, applications, and future trends." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Gundlach, Ingmar (2020). "Zeitoptimale Trajektorienplanung für automatisiertes Fahren bis in den fahrdynamischen Grenzbereich." PhD thesis. Technische Universität Darmstadt.

Gupta, Agrim, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi (2018). "Social gan: Socially acceptable trajectories with generative adversarial networks." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2255–2264.

Gupta, Jayesh, Maxim Egorov, and Mykel Kochenderfer (2017). "Cooperative multi-agent control using deep reinforcement learning." In: *Autonomous Agents & Multiagent Systems Workshops*, pp. 66–83.

Haan, Pim de, Dinesh Jayaraman, and Sergey Levine (2019). "Causal confusion in imitation learning." In: *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 11698–11709.

Hagedorn, Steffen, Marcel Hallgarten, Martin Stoll, and Alexandru Condurache (2023). "Rethinking integration of prediction and planning in deep learning-based automated driving systems: A review." In: *arXiv Preprint*. arXiv:2308.05731 [cs.RO].

Hart, Peter, Nils Nilsson, and Bertram Raphael (1968). "A formal basis for the heuristic determination of minimum cost paths." In: *IEEE Transactions on Systems Science and Cybernetics* 4.2, pp. 100–107.

He, Kaiming, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick (2022). "Masked autoencoders are scalable vision learners." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16000–16009.

Heinrich, Steffen (2018). "Planning universal on-road driving strategies for automated vehicles." PhD thesis. Freie Universität Berlin.

Ho, Jonathan and Stefano Ermon (2016). "Generative adversarial imitation learning." In: *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 4565–4573.

Hu, Yihan, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, Lewei Lu, Xiaosong Jia, Qiang Liu, Jifeng Dai, Yu Qiao, and Hongyang Li (2023). "Planning-oriented autonomous driving." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17853–17862.

Huang, Jialei, Zhao-Heng Yin, Yingdong Hu, and Yang Gao (2023). "Policy Contrastive Imitation Learning." In: *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 14007–14022.

Huang, Yanjun, Jiatong Du, Ziru Yang, Zewei Zhou, Lin Zhang, and Hong Chen (2022). "A survey on trajectory-prediction methods for autonomous driving." In: *IEEE Transactions on Intelligent Vehicles* 7.3, pp. 652–674.

Huang, Yanjun, Hong Wang, Amir Khajepour, Haitao Ding, Kang Yuan, and Yechen Qin (2019). "A novel local motion planning framework for autonomous vehicles based on resistance network and model predictive control." In: *IEEE Transactions on Vehicular Technology* 69.1, pp. 55–66.

Ishihara, Keishi, Anssi Kanervisto, Jun Miura, and Ville Hautamaki (2021). "Multi-task learning with attention for end-to-end autonomous driving." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2902–2911.

Isola, Phillip, Jun-Yan Zhu, Tinghui Zhou, and Alexei Efros (2017). "Image-to-image translation with conditional adversarial networks." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1125–1134.

Jaynes, Edwin (1957). "Information theory and statistical mechanics." In: *The Physical Review* 106, pp. 620–630.

Jing, Longlong and Yingli Tian (2021). "Self-supervised visual feature learning with deep neural networks: A survey." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.11, pp. 4037–4058.

Kaelbling, Leslie Pack, Michael Littman, and Anthony Cassandra (1998). "Planning and acting in partially observable stochastic domains." In: *Artificial Intelligence* 101.1-2, pp. 99–134.

Kalman, Rudolf (1964). "When is a linear control system optimal?" In: *ASME Journal of Basic Engineering* 86.1, pp. 51–60.

Kelly, Alonzo and Bryan Nagy (2003). "Reactive nonholonomic trajectory generation via parametric optimal control." In: *International Journal of Robotics Research* 22.7-8, pp. 583–601.

Kim, Jinkyu, Reza Mahjourian, Scott Ettinger, Mayank Bansal, Brandyn White, Ben Sapp, and Dragomir Anguelov (2022). "StopNet: Scalable

trajectory and occupancy prediction for urban autonomous driving." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8957–8963.

Kim, Tschangho (2018). "Automated autonomous vehicles: Prospects and impacts on society." In: *Journal of Transportation Technologies* 8, pp. 137–150.

Kiran, Ravi, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad Al Sallab, Senthil Yogamani, and Patrick Pérez (2021). "Deep reinforcement learning for autonomous driving: A survey." In: *IEEE Transactions on Intelligent Transportation Systems* 23.6, pp. 4909–4926.

Konstantinidis, Fabian, Moritz Sackmann, Oliver De Candido, Ulrich Hofmann, Jörn Thielecke, and Wolfgang Utschick (2021). "Parameter sharing reinforcement learning for modeling multi-agent driving behavior in roundabout scenarios." In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1974–1981.

Krishnan, Sanjay, Animesh Garg, Richard Liaw, Lauren Miller, Florian Pokorny, and Ken Goldberg (2016). "HIRL: Hierarchical inverse reinforcement learning for long-horizon tasks with delayed rewards." In: *arXiv Preprint*. arXiv:1604.06508 [cs.RO].

Kuderer, Markus, Shilpa Gulati, and Wolfram Burgard (2015). "Learning driving styles for autonomous vehicles from demonstration." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2641–2646.

LaValle, Steven (2006). *Planning algorithms*. Cambridge University Press.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). "Deep learning." In: *Nature* 521.7553, pp. 436–444.

Lee, Gunmin, Dohyeong Kim, Wooseok Oh, Kyungjae Lee, and Songhwai Oh (2020). "MixGAIL: Autonomous driving using demonstrations with mixed qualities." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5425–5430.

Leurent, Edouard (2018). "A survey of state-action representations for autonomous driving." In: *HAL Preprint*. HAL: hal-01908175.

Levine, Sergey, Aviral Kumar, George Tucker, and Justin Fu (2020). "Offline reinforcement learning: Tutorial, review, and perspectives on open problems." In: *arXiv Preprint*. arXiv:2005.01643 [cs.LG].

Li, Wenzhe, Hao Luo, Zichuan Lin, Chongjie Zhang, Zongqing Lu, and Deheng Ye (2023). "A survey on transformers in reinforcement learning." In: *Transactions on Machine Learning Research*.

Lin, Tianyang, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu (2022). "A survey of transformers." In: *AI Open* 3, pp. 111–132.

Ma, Yuexin, Tai Wang, Xuyang Bai, Huitong Yang, Yuenan Hou, Yaming Wang, Yu Qiao, Ruigang Yang, Dinesh Manocha, and Xinge Zhu (2022). "Vision-centric BEV perception: A survey." In: *arXiv Preprint*. arXiv:2208.02797 [cs.CV].

Mahjourian, Reza, Jinkyu Kim, Yuning Chai, Mingxing Tan, Ben Sapp, and Dragomir Anguelov (2022). "Occupancy flow fields for motion forecasting in autonomous driving." In: *IEEE Robotics and Automation Letters (RA-L)* 7.2, pp. 5639–5646.

McNaughton, Matthew (2011). "Parallel algorithms for real-time motion planning." PhD thesis. Carnegie Mellon University.

McNaughton, Matthew, Chris Urmson, John Dolan, and Jin-Woo Lee (2011). "Motion planning for autonomous driving with a conformal spatiotemporal lattice." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4889–4895.

Mehta, Ashish, Adithya Subramanian, and Anbumani Subramanian (2018). "Learning end-to-end autonomous driving using guided auxiliary supervision." In: *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP)*, pp. 1–8.

Montemerlo, Michael, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, Doug Johnston, Stefan Klumpp, Dirk Langer, Anthony Levandowski, Jesse Levinson, Julien Marcil, David Orenstein, Johannes Paefgen, Isaac Penny, Anna Petrovskaya, Mike Pflueger, Ganymed Stanek, David Stavens, Antone Vogt, and Sebastian Thrun (2008). "Junior: The Stanford entry in the urban challenge." In: *Journal of Field Robotics* 25.9, pp. 569–597.

Mourant, Ronald and Thomas Rockwell (1970). "Visual information seeking of novice drivers." In: *International Automotive Safety Conference Compendium*.

– (1972). "Strategies of visual search by novice and experienced drivers." In: *Human Factors* 14.4, pp. 325–335.

Näätänen, Risto and Heikki Summala (1976). "Road-user behaviour and traffic accidents." In: *Publication of: North-Holland Publishing Company*.

Neal, Radford (2000). "Markov chain sampling methods for Dirichlet process mixture models." In: *Journal of Computational and Graphical Statistics* 9.2, pp. 249–265.

Ng, Andrew, Daishi Harada, and Stuart Russell (1999). "Policy invariance under reward transformations: Theory and application to

reward shaping." In: *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 278–287.

Ng, Andrew and Stuart Russell (2000). "Algorithms for inverse reinforcement learning." In: *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 663–670.

Oquab, Maxime, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski (2024). "DINOv2: Learning robust visual features without supervision." In: *Transactions on Machine Learning Research*.

Osa, Takayuki, Joni Pajarinen, Gerhard Neumann, Andrew Bagnell, Pieter Abbeel, and Jan Peters (2018). "An algorithmic perspective on imitation learning." In: *Foundations and Trends® in Robotics* 7.1-2, pp. 1–179.

Othman, Kareem (2021). "Public acceptance and perception of autonomous vehicles: A comprehensive review." In: *AI and Ethics* 1.3, pp. 1–33.

Paden, Brian, Michal Cáp, Sze Yong, Dmitry Yershov, and Emilio Frazzoli (2016). "A survey of motion planning and control techniques for self-driving urban vehicles." In: *IEEE Transactions on Intelligent Vehicles* 1.1, pp. 33–55.

Papadimitriou, Christos and John Tsitsiklis (1987). "The complexity of Markov decision processes." In: *Mathematics of Operations Research* 12.3, pp. 441–450.

Parmar, Gaurav, Taesung Park, Srinivasa Narasimhan, and Jun-Yan Zhu (2024). "One-step image translation with text-to-image models." In: *arXiv Preprint*. arXiv:2403.12036 [cs.CV].

Pomerleau, Dean (1989). "Alvinn: An autonomous land vehicle in a neural network." In: *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 305–313.

Puterman, Martin (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Ratliff, Nathan, Andrew Bagnell, and Martin Zinkevich (2006). "Maximum margin planning." In: *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 729–736.

Ratliff, Nathan, David Silver, and Andrew Bagnell (2009). "Learning to search: Functional gradient techniques for imitation learning." In: *Autonomous Robots* 27.1, pp. 25–53.

Rhinehart, Nicholas, Jeff He, Charles Packer, Matthew Wright, Rowan McAllister, Joseph Gonzalez, and Sergey Levine (2021). "Contingencies from observations: Tractable contingency planning with learned behavior models." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13663–13669.

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015). "U-Net: Convolutional networks for biomedical image segmentation." In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pp. 234–241.

Rosbach, Sascha, Vinit James, Simon Großjohann, Silviu Homoceanu, Xing Li, and Stefan Roth (2020a). "Driving style encoder: Situational reward adaptation for general-purpose planning in automated driving." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6419–6425.

Rosbach, Sascha, Vinit James, Simon Großjohann, Silviu Homoceanu, and Stefan Roth (2019). "Driving with style: Inverse reinforcement learning in general-purpose planning for automated driving." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2658–2665.

Rosbach, Sascha, Stefan Leupold, Simon Großjohann, and Stefan Roth (2023). "Pixel state value network for combined prediction and planning in interactive environments." In: *arXiv Preprint*. arXiv:2310.07706 [cs.RO].

Rosbach, Sascha, Xing Li, Simon Großjohann, Silviu Homoceanu, and Stefan Roth (2020b). "Planning on the fast lane: Learning to interact using attention mechanisms in path integral inverse reinforcement learning." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5187–5193.

Ross, Stéphane and Drew Bagnell (2010). "Efficient reductions for imitation learning." In: *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 661–668.

Ross, Stéphane, Geoffrey Gordon, and Drew Bagnell (2011). "A reduction of imitation learning and structured prediction to no-regret online learning." In: *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 627–635.

Sadat, Abbas, Sergio Casas, Mengye Ren, Xinyu Wu, Pranaab Dhawan, and Raquel Urtasun (2020). "Perceive, predict, and plan: Safe motion planning through interpretable semantic representations." In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 414–430.

Sagberg, Fridulv, Selpi, Giulio Piccinini, and Johan Engström (2015). "A review of research on driving styles and road safety." In: *Human Factors* 57.7, pp. 1248–1275.

Sauer, Axel, Dominik Lorenz, Andreas Blattmann, and Robin Rombach (2023). "Adversarial diffusion distillation." In: *arXiv Preprint*. arXiv:2311.17042 [cs.CV].

Sauer, Axel, Nikolay Savinov, and Andreas Geiger (2018). "Conditional affordance learning for driving in urban environments." In: *Conference on Robot Learning (CoRL)*, pp. 237–252.

Schwarting, Wilko, Javier Alonso-Mora, and Daniela Rus (2018). "Planning and decision-making for autonomous vehicles." In: *Annual Review of Control, Robotics, and Autonomous Systems* 1, pp. 187–210.

Shalev-Shwartz, Shai, Shaked Shammah, and Amnon Shashua (2016). "Safe, multi-agent, reinforcement learning for autonomous driving." In: *Neural Information Processing Systems (NeurIPS) Workshop on Learning, Inference and Control of Multi-Agent Systems*. arXiv: 1610.03295 [cs.AI].

Shi, Shaoshuai, Li Jiang, Dengxin Dai, and Bernt Schiele (2022). "Motion transformer with global intention localization and local movement refinement." In: *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 6531–6543.

– (2023). "MTR++: Multi-agent motion prediction with symmetric scene modeling and guided intention querying." In: *arXiv Preprint*. arXiv:2306.17770 [cs.CV].

Silver, David, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis (2017). "Mastering chess and shogi by self-play with a general reinforcement learning algorithm." In: *arXiv Preprint*. arXiv:1712.01815 [cs.AI].

Silver, David and Joel Veness (2010). "Monte-Carlo planning in large POMDPs." In: *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2164–2172.

Sohl-Dickstein, Jascha, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli (2015). "Deep unsupervised learning using nonequilibrium thermodynamics." In: *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 2256–2265.

Šošić, Adrian, Abdelhak M Zoubir, Elmar Rueckert, Jan Peters, and Heinz Koeppl (2018). "Inverse reinforcement learning via nonparametric spatio-temporal subgoal modeling." In: *Journal of Machine Learning Research* 19.1, pp. 2777–2821.

Spencer, Jonathan, Sanjiban Choudhury, Arun Venkatraman, Brian Ziebart, and Andrew Bagnell (2021). "Feedback in imitation learning: The three regimes of covariate shift." In: *arXiv Preprint*. arXiv: 2102.02872 [cs.LG].

Summala, Heikki (1988). "Risk control is not risk adjustment: The zero-risk theory of driver behaviour and its implications." In: *Ergonomics* 31.4, pp. 491–506.

– (2007). "Towards understanding motivational and emotional factors in driver behaviour: Comfort through satisficing." In: *Modelling Driver Behaviour in Automotive Environments*. Springer, pp. 189–207.

Sutton, Richard and Andrew Barto (2018). *Reinforcement learning: An introduction*. Vol. 2. MIT Press.

Sutton, Richard, Doina Precup, and Satinder Singh (1999). "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning." In: *Artificial Intelligence* 112.1-2, pp. 181–211.

Tampuu, Ardi, Tambet Matiisen, Maksym Semikin, Dmytro Fishman, and Naveed Muhammad (2020). "A survey of end-to-end driving: Architectures and training methods." In: *IEEE Transactions on Neural Networks and Learning Systems* 33.4, pp. 1364–1384.

Tang, Yichuan (2019). "Towards learning multi-agent negotiations via self-play." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pp. 2427–2435.

Taubman Ben-Ari, Orit and Vera Skvirsky (2016). "The multidimensional driving style inventory a decade later: Review of the literature and re-evaluation of the scale." In: *Accident Analysis & Prevention* 93, pp. 179–188.

Thodoroff, Pierre, Audrey Durand, Joelle Pineau, and Doina Precup (2018). "Temporal regularization for markov decision process." In: *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1784–1794.

Tripathy, Soumya, Juho Kannala, and Esa Rahtu (2019). "Learning image-to-image translation using paired and unpaired training samples." In: *Proceedings of the Asian Conference on Computer Vision (ACCV)*, pp. 51–66.

Ulbrich, Simon and Markus Maurer (2013). "Probabilistic online POMDP decision making for lane changes in fully automated driving." In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2063–2067.

– (2015). "Situation assessment in tactical lane change behavior planning for automated vehicles." In: *IEEE Transactions on Intelligent Transportation Systems*, pp. 975–981.

Urmson, Chris, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, M. N. Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, Michele Gittleman, Sam Harbaugh, Martial Hebert, Thomas Howard, Sascha Kolski, Alonzo Kelly, Maxim Likhachev, Matt McNaughton, Nick Miller, Kevin Peterson, Brian Pilnick, Raj Rajkumar, Paul Rybski, Bryan Salesky, Young-Woo Seo, Sanjiv Singh, Jarrod Snider, Anthony Stentz, William Whittaker, Ziv Wolkowicki, Jason Ziglar, Hong Bae, Thomas Brown, Daniel Demitrish, Bakhtiar Litkouhi, Jim Nickolaou, Varsha Sadekar, Wende Zhang, Joshua Struble, Michael Taylor, Michael Darms, and Dave Ferguson (2008). "Autonomous driving in urban environments: Boss and the urban challenge." In: *Journal of Field Robotics* 25.8, pp. 425–466.

Vaa, Truls (2007). "Modelling driver behaviour on basis of emotions and feelings: Intelligent transport systems and behavioural adaptations." In: *Modelling Driver Behaviour in Automotive Environments*. Springer, pp. 208–232.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). "Attention is all you need." In: *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 5998–6008.

Werling, Moritz, Julius Ziegler, Sören Kammel, and Sebastian Thrun (2010). "Optimal trajectory generation for dynamic street scenarios in a frenet frame." In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 987–993.

Wulfmeier, Markus, Peter Ondruska, and Ingmar Posner (2015). "Maximum entropy deep inverse reinforcement learning." In: *arXiv Preprint*. arXiv:1507.04888 [cs.LG].

Wulfmeier, Markus, Dushyant Rao, Dominic Zeng Wang, Peter Ondruska, and Ingmar Posner (2017). "Large-scale cost function learning for path planning using deep inverse reinforcement learning." In: *International Journal of Robotics Research* 36.10, pp. 1073–1087.

Yang, Ling, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang (2024). "Diffusion models: A comprehensive survey of methods and applications." In: *ACM Computing Surveys* 56.4, pp. 1–39.

Yuan, Kang, Hong Shu, Yanjun Huang, Yubiao Zhang, Amir Khajepour, and Lin Zhang (2019). "Mixed local motion planning and tracking control framework for autonomous vehicles based on model pre-

dictive control." In: *IET Intelligent Transport Systems* 13.6, pp. 950–959.

Yurtsever, Ekim, Jacob Lambert, Alexander Carballo, and Kazuya Takeda (2020). "A survey of autonomous driving: Common practices and emerging technologies." In: *IEEE Access* 8. pp. 58443–58469.

Zeng, Wenyuan, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun (2019). "End-to-end interpretable neural motion planner." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8660–8669.

Zeng, Wenyuan, Shenlong Wang, Renjie Liao, Yun Chen, Bin Yang, and Raquel Urtasun (2020). "DSDNET: Deep structured self-driving network." In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 156–172.

Zhang, Kaiqing, Zhuoran Yang, and Tamer Başar (2021). "Multi-agent reinforcement learning: A selective overview of theories and algorithms." In: *Handbook of Reinforcement Learning and Control*. Springer, pp. 321–384.

Zhang, Yu and Qiang Yang (2021). "A survey on multi-task learning." In: *IEEE Transactions on Knowledge and Data Engineering* 34.12, pp. 5586–5609.

Zheng, Ling, Bijun Li, Bo Yang, Huashan Song, and Zhi Lu (2019). "Lane-level road network generation techniques for lane-level maps of autonomous vehicles: A survey." In: *Sustainability* 11.16, p. 4511.

Ziebart, Brian, Andrew Maas, Andrew Bagnell, and Anind Dey (2008). "Maximum entropy inverse reinforcement learning." In: *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 1433–1438.