THE UNIVERSITY
of EDINBURGH

Interpretable and Verifiable Planning and Prediction for Autonomous Vehicles

Cillian Brewitt

Doctor of Philosophy
Institute of Perception, Action and Behaviour
School of Informatics
University of Edinburgh
2023

Abstract

Autonomous driving (AD) has gained much attention in recent years due to its many potential benefits such as improving safety and increasing efficiency. However, AD is a difficult problem with challenges such as handling interactions with other vehicles and predicting the future behaviour of human drivers. This often takes place in complicated urban environments where information is missing due to occlusions. AD methods must also be accurate and effective while still being efficient enough to run in real time.

In this thesis, several novel AD methods are presented which contribute towards solving some of the problems of AD. In particular, the focus is on planning, prediction and goal recognition (GR) methods which are interpretable by humans and formally verifiable. Interpretability can increase user trust of AD systems and aid with debugging issues with such systems. Having the ability to formally verify propositions made about AD methods can help ensure safety and compliance with regulations.

The first novel method is Interpretable Goal-based Prediction and Planning (IGP2) which integrates GR through inverse planning with Monte Carlo tree search (MCTS) to achieve a full planning and prediction system. IGP2 is evaluated in several urban driving scenarios and is shown to successfully recognise other vehicle's goals and improve driving efficiency. The second method is Goal Recognition with Interpretable Trees (GRIT). GRIT makes use of learned decision trees trained to infer a probability distribution over the goals of other vehicles. An evaluation across two vehicle trajectory datasets shows that the inference process of GRIT is fast, accurate, interpretable and verifiable. The third method is Goal Recognition with Interpretable Trees under Occlusion (OGRIT). Similarly to GRIT, OGRIT makes use of learned decision trees for GR. Through an evaluation across two vehicle trajectory datasets with significant occlusions, OGRIT is also shown to handle information missing due to occlusions and can make inferences across multiple scenarios using the same learned models, while still remaining fast, accurate, interpretable and verifiable. This thesis contributes three novel methods which work towards allowing autonomous vehicles to accurately and efficiently infer the goals of other vehicles in complex, partially occluded urban environments, and then predict their future behaviour and plan accordingly.

Lay Summary

Autonomous driving (e.g. self-driving vehicles) has seen increasing attention in recent years as it has many benefits such as improving safety and efficiency of vehicles. However, autonomous driving is very difficult as autonomous vehicles need to interact with human drivers and pedestrians and predict what they will do next. These interactions can be especially complicated in urban settings, where many parts of the surrounding area could be hidden from view by obstacles. Autonomous driving methods must also make accurate predictions and effective decisions very fast so that vehicles can quickly react to their surroundings.

In this thesis, three new autonomous driving methods are presented, which address the problems mentioned above. In particular, the focus is on methods for planning the future actions of an autonomous vehicle, predicting the behaviour of other road users, and recognising the goals of other road users. We focus is on methods which can be easily understood and interpreted by humans. We also focus on methods which can be verified to always follow specified rules relating to their input and output.

The first novel method is Interpretable Goal-based Prediction and Planning (IGP2), which is a full planning and prediction system. IGP2 first infers the goals of other vehicles, by planning from their point of view. IGP2 then generates a plan for the ego vehicle, which takes into account the goals of the other vehicles. An evaluation in several different simulated situations shows that IGP2 can accurately recognise the goals of other vehicles and improve driving efficiency. The second novel method is Goal Recognition with Interpretable Trees (GRIT). GRIT recognises the goals of other vehicles, using models which have a tree-like structure. These models are trained on data from real vehicles. An evaluation using two datasets shows that GRIT is fast, accurate, interpretable by humans, and verifiable. The third method is Goal Recognition with Interpretable Trees under Occlusion (OGRIT). Similarly to GRIT, OGRIT makes use of models which have a tree-like structure. However, OGRIT also has the ability to handle situations where information is occluded from the point of view of the ego vehicle. In addition to this, the same OGRIT models can be applied across multiple different scenarios, while still being fast, accurate and verifiable.

This thesis contributes several novel methods for autonomous driving. These methods allow autonomous vehicles to accurately predict the behaviour of other vehicles and plan accordingly, in a manner which is fast, human interpretable, can generalise, and handles situations where the surrounding are only partially visible.

Acknowledgements

This thesis and the work described in it would not have been possible without the support I receive from many people. First, I would like to thank my advisor, Stefano Albrecht. He accepted me into this program and has enthusiastically provided generous amounts of guidance and feedback on my research directions and writing throughout my studies. I have learned much from him about the academic research process. I would also like to acknowledge the support given by my secondary advisor Kobi Gal in providing feedback about my research and plans.

I would like to thank all the members of the Autonomous Agents Research Group at Edinburgh University for their help and support. Much of the work described in this thesis has been achieved in close collaboration with Balint Gyevnar, Samuel Garcin and Massimiliano Tamborski. Their contributions have invaluable in achieving the results we have achieved. Many other members of the group have also been valuable in providing feedback about my writing. It has been amazing to have a supportive community like this throughout my studies.

I would also like to thank Nick Hoernle and Giorgos Papoudakis for the many discussions we had together during lunchtimes. The pandemic restrictions that took place during our studies have been especially challenging, and I really appreciate having peers to regularly meet with and mutually support each other.

I thank Marco Pavone and Chris Xiaoxuan Lu for taking the time out of their busy schedules to review my thesis and act as examiners for my viva. It is an honour to have such accomplished academics take part in this process.

I thank my family for their support throughout the entire process. I hugely appreciate the regular calls and visits over the last few years. They have been very generous with advice about life and other forms of support throughout this process.

Finally, I would like to express my gratitude to Kritika, whose love and support has been essential through the highs and lows of my studies.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Cillian Brewitt*)

Table of Contents

Chapter 1

Introduction

Autonomous driving is a difficult problem, but has enormous potential benefits. Approximately 1.35 million people are killed in road traffic crashes each year [1], many of which are caused by human error. Autonomous vehicles (AVs) have the potential to significantly reduce this figure by improving safety and reducing human error [2]. AVs could also significantly reduce the cost of urban mobility, and reduce the need for individual car ownership. Another benefit is improved efficiency in many areas. AVs could improve fuel efficiency through smoother acceleration, platooning, and coordination with other AVs. Autonomous taxi services which operate continually could reduce the need for car parks in cities, opening up urban space for other uses [3].

Since the DARPA Grand Challenges for autonomous vehicles from 2004-2007 [4, 5], there has constantly been growing interest and progress in this field. AVs serve as an important real-world test bed for artificial intelligence (AI) and robotics methods. Building AV systems requires a diverse range of hardware and software technology from many different fields. On the software side, the problem is often broken down into sub-problems such as perception, prediction, and planning [6]. The task of perception involves using raw data from various sensors such as cameras, LIDAR and radar to infer the state of the vehicle's surroundings [7, 8]. Prediction refers to predicting the future states of other road users. Planning systems and control systems must make decisions and act by controlling aspects of the vehicle such as steering, acceleration and braking. One of the greatest remaining challenges for prediction and planning in AVs is interaction with human driven vehicles. Human driving behaviour can be difficult to model, and there is a diverse long tail of uncommon scenarios that can arise while driving. Planning and prediction methods which are interpretable allow people to reason about how methods will generalise to scenarios which they have not yet been tested on.

It is also important to have methods for validating planning and prediction methods, as autonomous driving is a safety critical task.

In contrast to the modular approach, a different approach to autonomous driving is end-to-end learning [9, 10, 11]. When using this approach, a machine learning system is trained to act directly based on sensory input. This approach requires vast amounts of data, which can be collected through simulation or vehicles operating in the real world.

In recent years, there has been increasing interest in *interpretable* or *explainable* AI. For systems that make decisions which affect people's lives significantly, or are safety critical, such as autonomous driving, there is a growing recognition of the importance of having autonomous systems with interpretable decision-making processes [12]. Regulations which codify the "right to an explanation" for some types of automated decision-making have already been created [13], and regulators may create similar rules for AVs. Another benefit of interpretability is that users may have greater trust in interpretable systems [14], leading to faster adoption of AVs. There are still many situations in which AVs make mistakes that human drivers would be unlikely to make, and interpretability allows systems to be debugged more easily.

As autonomous driving is a safety critical task, it is important to have the ability to validate that planning and prediction systems will act as intended when deployed. However, validating planning and prediction systems empirically could require billions of miles of real or simulated driving [15]. An alternative approach is to use *verifiable* planning and prediction methods. Verifiability refers to the ability to prove that certain statements about a system will always hold true, with the aim of guaranteeing safety under all possible situations [16, 17].

Despite the recent progress in AVs, there are still open problems. It is desirable for planning and prediction methods to be accurate, fast to compute, interpretable, verifiable, generalisable, and handle occlusions. Existing methods fail at one or more of these objectives. For example, one of the earliest approaches for planning was finite state machines [18, 19, 20], which are both fast to compute and highly interpretable. However, these require plans to be handcrafted for predefined scenarios, resulting in poor generalisation to new situations. Deep neural networks have been widely applied to perception, prediction, decision-making, and end-to-end learning, often achieving high accuracy [21]. However, these models typically have millions of learned parameters, which makes them impossible for humans to directly interpret. One approach to interpretable prediction infers the goals of other agents by assuming that they are approximately rational and then performing inverse planning [22, 23]. Such methods

can give accurate predictions and allow intuitive explanations to be extracted, but the inverse planning process is computationally expensive.

1.1 Aims and Objectives

The objective of this thesis is to study and develop planning and prediction methods for AVs. More specifically, the aim is to develop methods for AVs which are fast, accurate, interpretable, verifiable, generalisable, and handle occlusions. These desiderata are defined as follows:

- **Fast:** Planning and prediction methods must have the ability to make decisions and inferences in real time, in order to be useful for AVs. We define this as making decisions or inferences with a latency of 100 milliseconds or lower.
- **Accurate:** In order for the plans and predictions to be useful, they must be accurate. The proposed methods should have comparable or better accuracy than related state-of-the-art methods.
- **Interpretable:** The proposed methods should produce plans and predictions in a manner that can be easily interpreted by humans. As defined in Section 2.2, methods should have algorithmic transparency or simulatability.
- **Verifiable:** It should be possible to formally prove that the outputs of methods will always obey constraints which are specified using propositional logic.
- **Generalisable:** The proposed methods should have the ability to make plans or predictions across multiple scenarios using the same models in each scenario.
- **Handle Occlusions:** The proposed methods should have the ability to function and make plans and predictions in partially observable scenarios where information is missing due to occlusions.

1.2 Contributions

In this thesis, the contributions are a literature review of related work, along with several novel planning and prediction methods for AVs. In particular, the focus is on goal recognition (GR), which involves inferring the long-term goals of agents. The main contributions of this thesis are:

- A literature review of research related to planning and prediction for AVs, interpretable AI, and formal verification. From this review, it was found that existing planning and prediction methods for AVs do not satisfy all the objectives of being fast, accurate, interpretable, verifiable, generalisable, and handle occlusions. The literature review is presented in Chapter 2.

- An AV planning and prediction method, named Interpretable Goal-based Prediction and Planning (IGP2) [24]. IGP2 makes use of inverse planning to perform GR used for multi-modal trajectory prediction. An integration of these predictions with Monte Carlo Tree Search (MCTS) is used to obtain optimised plans for the ego vehicle, giving a full planning and prediction system. An evaluation of IGP2 across several simulated urban driving scenarios and two open-world towns is presented, and it is shown that IGP2 achieves accurate goal recognition, efficient driving, and interpretable plans and predictions. This method is described in Chapter 4.

- An AV goal recognition method, named Goal Recognition with Interpretable Trees (GRIT) [25]. GRIT can make fast inferences due to its straightforward inference process, which makes use of learned decision trees (DTs). GRIT is evaluated across four different urban driving scenarios using data from two different vehicle trajectory datasets, and it is shown that GRIT achieves comparable accuracy to a neural network baseline. It is also shown that properties of the learned DTs can be formally verified using an off-the-shelf satisfiability modulo theories (SMT) solver. Even if verification fails, the solver provides a counterexample which can teach us more about the learned model. It is also shown that inferences made by GRIT can be easily interpreted by humans thanks to the interpretable input features used and shallow DT depth. This method is described in Chapter 5.

- An AV goal recognition method which can handle occlusions, named Goal Recognition with Interpretable Trees under Occlusion (OGRIT) [26]. OGRIT can handle occlusions and generalise across multiple scenarios, while still being fast, accurate, interpretable and verifiable. Two new datasets of occluded regions are also introduced, named the inDO and rounDO datasets, along with an open-source tool that can be used to detect occluded regions in other datasets. An evaluation of OGRIT on the inDO and rounDO datasets is presented, and OGRIT is shown to have accuracy close to that of a neural network based GR method, while still

being fast, accurate, interpretable and verifiable. This method is described in Chapter 6.

1.3 Limiting Assumptions

Throughout this thesis, some limiting assumptions are made in order to simplify the problem. For each of the GR methods presented in this thesis, a set of goals are generated for each non-ego vehicle, for example taking a certain junction exit or going straight on at a junction. The assumption is made that the actual goal of the non-ego vehicle belongs to this set, but in rare cases vehicles may have other goals, for example parking at the side of the road.

For the method named IGP2 described in Chapter 4, the assumption is also made that vehicles only perform manoeuvres and macro actions from a known library. It is also assumed that vehicles are rational and plan to minimise a known cost function while driving to their goal.

In the method named GRIT presented in Chapter 5, it is assumed that the environment is fully observable by the ego vehicles, though this assumption is relaxed in Chapter 4 and Chapter 5. Another limitation of GRIT is that new models must be trained for each scenario the method is applied to, so the method cannot be applied to previously unseen scenarios where there is no training data available.

Some of these limitations are addressed by the method named OGRIT which is presented in Chapter 6. OGRIT can handle missing information due to occlusions, and can use the same learned models across multiple scenarios. However, occlusions from the point of view of non-ego vehicles are not taken into account. In addition to this, the scene is modelled in 2D when detecting occlusions, and the height of obstacles is not taken into account.

All methods described in this thesis focus solely on planning and prediction, and make the assumption that perception of the surrounding scene has already been performed by a separate module. The simplifying assumption is made that the output of the perception module is perfectly accurate. However, it is important to note that in a real deployment, the perception output may be noisy or incorrect.

1.4 Publications

Research presented in the following publications make up parts of this thesis:

- *S. V. Albrecht, C. Brewitt, J. Wilhelm, B. Gyevnar, F. Eiras, M. Dobre, and S. Ramamoorthy, "Interpretable goal-based prediction and planning for autonomous driving," in IEEE International Conference on Robotics and Automation, 2021.*

  My contributions to this publication included contributing to the design, implementation and evaluation of the method. Wilhelm and Gyevnar also contributed to the design, implementation and evaluation of the method. Eiras, Dobre, Ramamoorthy and Albrecht also contributed to the design of the method.

- *C. Brewitt, B. Gyevnar, S. Garcin, and S. V. Albrecht, "GRIT: Fast, interpretable, and verifiable goal recognition with learned decision trees for autonomous driving," in IEEE/RSJ International Conference on Intelligent Robots and Systems, 2021.*

  My contributions to this publication included the design, implementation, evaluation of the method, and writing the main text of the publication. Gyevnar and Garcin contributed to the implementation and evaluation of baseline methods, and gave advice and feedback on the text of the publication. Albrecht contributed to the design of the method, and gave advice and feedback on the text of the publication.

- *C. Brewitt, M. Tamborski, S. V. Albrecht, Verifiable Goal Recognition for Autonomous Driving with Occlusions, NeurIPS Workshop on Machine Learning for Autonomous Driving, 2022.*

  This publication is also under review for the International Conference on Intelligent Robots and Systems (IROS) 2023. My contributions to this publication include the design, implementation, evaluation of the method, and writing the majority of the text of the publication. Tamborski contributed to the implementation and evaluation of baseline methods, and writing some text of the publication. Albrecht contributed to the design of the method, and gave advice and feedback on the text of the publication.

Chapter 2

Related Work

There have been ever-increasing amounts of research in methods for autonomous driving in recent years. In this chapter, it is described how the research presented in this thesis relates to existing research. In Section 2.1, an overview is given of autonomous driving methods, with a focus on planning methods in Section 2.1.1 and prediction methods in Section 2.1.2. An overview is also given of work in the field of interpretable AI in Section 2.2, and discuss its application in autonomous driving. In Section 2.3 it is discussed how formal verification has been applied to autonomous driving methods.

2.1 Autonomous Driving

There are many approaches to autonomous driving. Some methods take a modular approach, where the whole task is subdivided into easier subtasks which are solved by separate modules, as shown in Fig. 2.1. The task of perception is usually given its own module. Perception takes raw data from sensors as input, and extracts information about the vehicle's surroundings [7, 8]. Extracted information can include aspects such as the local road layout, road signs, road markings and the positions and velocities of other vehicles and pedestrians. This information can then be passed to other modules which perform tasks such as motion planning and prediction. After a trajectory has been planned, a low level control module can then be used to track this trajectory by controlling steering, braking and acceleration. The methods presented in this thesis follow the modular approach. Planning and prediction is the main focus of this thesis, and existing approaches for solving these tasks will be reviewed in the following subsections.

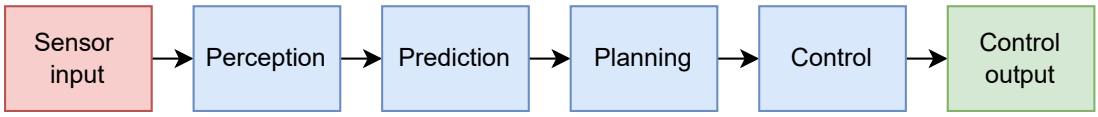


Figure 2.1: Modular approach to autonomous driving using sequential prediction and planning. The red block represents input, blue blocks represent intermediate steps, and the green block represents output.

2.1.1 Planning and Decision-Making

The aim of planning is usually to select a path, trajectory or immediate action which aims to bring the vehicle to a goal location, while minimising a cost function, sometimes subject to constraints. Most planning methods for AVs can be categorised as either sequential planning, behaviour aware planning, or end-to-end planning [27]. Sequential planning uses a pipeline approach, where perception and prediction are performed first before passing their output to a planning module. This adds more modularity to systems and can simplify their design. However, there are some situations where the actions of the ego vehicle can affect the actions of other road users, and sequential planning fails to take this into account. Behaviour aware planning methods couple planning and prediction together. These methods can more accurately model interaction between the ego vehicle and other road users, but tend to be more complex than sequential planning methods. End to end planning involves directly training machine learning models to make decisions when given raw sensor input. These have the potential to learn intermediate representations that achieve better performance than hand designed representations. However, the intermediate representations learned by end-to-end methods may be uninterpretable.

2.1.1.1 Sequential Planning

Some sequential planning methods use lattice planning to plan over a discretised action space while checking for collisions [28, 29]. Other approaches used random sampling-based planning [30], such as rapidly exploring random trees (RRT) [31, 32] or deterministic sampling-based planning, such as Fast Marching Tree (FMT*) [33, 34]. These methods can explore large state spaces, although with a large computational cost. Another planning approach is to use constrained optimisation or model predictive control [35, 36, 37, 38, 39, 40, 41] to find an optimal trajectory, while satisfying constraints such as collision avoidance. These methods converge only to local optima, but this is often still sufficient for deployment. Sequential planning methods such as these

can be efficient and simplify design of the system. However, such methods do not take into account how the actions of the ego vehicle affect the actions of other road users. In contrast, the planning method presented in Chapter 4 is a form of behaviour aware planning, which takes into account interaction between the ego vehicle and other vehicles.

2.1.1.2 Behaviour-Aware Planning

Behaviour aware planning methods can be seen as combined planning and prediction methods. Such methods can model interaction more accurately than sequential planning methods, and can be much more interpretable than end-to-end planning methods. One of the earliest approaches are finite state machines, where plans are handcrafted for a set of predefined scenarios [18, 19, 20]. This approach is highly interpretable, but cannot generalise to scenarios outside the predefined set. Another approach is to use decision trees to select manoeuvres or paths from a set of candidates [42, 43].

Sadigh et al. [44] modelled interaction as a dynamical system which was solved using MPC, and modelled how humans react to the ego vehicle's actions by assuming they are optimising a known loss function. However, this approach may fail and lead to dangerous situations if the human model is inaccurate. One solution is to model a joint distribution over the actions of the ego vehicle and other agents [45, 46]. During and Pascheka [47] discretise the action space and evaluate all possible interactions. However, this has exponential complexity and quickly becomes intractable as the number of agents increases. This can be made more tractable by performing a game tree search, where some branches can be pruned [48]. Another way to make the problem more tractable is to use Monte Carlo Tree Search (MCTS) to prioritise exploring more promising interaction sequences [49], while using a simple lane following model for other vehicles. Similarly to this, MCTS is used as part of the method described in Chapter 4, but MCTS is integrated with a more accurate motion prediction method based around goal recognition. By assuming that followers are completely dominated by leaders, complexity in the number of agents can be reduced from exponential to quadratic [50]. Complexity can be further reduced to linear by modelling interaction as a Stackleberg game [51].

A different approach is to model the problem as a Markov Decision Process (MDP). Actions can then be selected by simulating forward and choosing the actions leading to the lowest cost [52]. A similar approach is to model the problem as a POMDP [53, 54, 55, 56], where the hidden variables can represent the intentions of other road

users, potentially hidden objects, or observation uncertainty. Solving these POMDPs exactly is usually intractable, so approximate solvers are generally used. Finally, another approach is to train machine learning models to make decisions. Vallon et al. [57] trained a support vector machine from human data to make lane change decisions based on features giving information about surrounding vehicles. However, such methods may require large amounts of training data, in contrast to the MCTS based planning method described in Chapter 4, which does not require any training data.

2.1.1.3 End-to-End Learning

For end-to-end autonomous driving, models are trained to perform some element of planning given raw sensor data such as LIDAR or images from cameras as input. There are two general approaches to end-to-end learning: Imitation learning (IL), where a model is trained from human example via supervised learning, and reinforcement learning (RL), where an agent learns to act based on its own experiences, usually in simulation. Some imitation learning methods involve training a neural network to map directly from a camera image or LIDAR to a steering angle [9, 10, 11]. Bojarski et al. [58] showed that convolutional neural networks (ConvNets) trained in such a way can learn to extract features from images representing road markings and other vehicles. Some other methods do not select actions directly, but instead used ConvNets to propose paths and identify drivable areas given camera images [59, 60]. Xu et al. [61] trained a model to select discrete actions such as go straight, turn left, or turn right based on video input. One drawback of IL methods is that the human demonstrations used as training data may not cover the entire state space. Data relating to situations such as near-collisions may be missing, as collecting such data would be dangerous. One solution to this is to train reinforcement learning agents in a simulated environment. Wolf et al. [62] trained a Deep Q-Network in simulation to select discrete actions, while other methods such as Deep Deterministic Policy Gradient [63] can be trained for continuous action spaces. One drawback of training in simulation is that models trained on rendered images may not transfer to real life images. One way to address this issue is to train a generative neural network to generate realistic images to be used while training [64]. An advantage of end-to-end planning methods over modular approaches is that their internal components are optimised to maximise the overall system performance, rather than intermediate objectives chosen for their human interpretability. However, interpretability is often desirable, even if it means sacrificing some performance by other metrics. Deep neural networks such as those used for end-to-end learning are

notoriously difficult to interpret, and can often behave in unexpected ways if used in situations not represented in their training data. In contrast to such end-to-end learning methods, the methods presented in this thesis are easily interpretable by humans, and the methods described in Chapter 5 and Chapter 6 are verifiable using formal verification.

2.1.2 Motion Prediction

Considering all physically possible actions of other agents can lead to an explosion of uncertainty, for which there is no safe plan. This is known as the freezing robot problem [45]. To solve this problem, behavioural models of other agents are needed. One of the simplest models assumes that agents will continue to travel at a constant velocity [65]. A more realistic model is the Intelligent Driver Model (IDM) [66, 67], which models distance keeping for following vehicles by assuming they act as if using adaptive cruise control. However, IDM does not take into account manoeuvres such as turning and lane changes.

Some methods such as the method presented in Chapter 4 output a probability distribution over trajectories, rather than a single trajectory. Hoermann et al. [66] used a particle filter to estimate distributions over IDM parameters, and then propagated forward to obtain a distribution over trajectories. Another approach modelled trajectory distributions as Gaussian processes [45].

In Section 2.1.2.1, methods which make use of deep neural networks for motion prediction are discussed. In Section 2.1.2.2 methods which assume vehicles follow a distinct set of manoeuvres are reviewed. In Section 2.1.2.3 an overview is given of how goal recognition has been applied to autonomous driving. Finally, in Section 2.1.2.4 a description is given of how inverse reinforcement learning methods have been used to infer cost functions for road users.

2.1.2.1 Deep Neural Networks

In recent years, there have been a growing number of methods which use deep neural networks (DNNs) to model distributions over trajectories given the scene history [68, 69, 70, 71, 72, 73, 74]. Although such methods can achieve a high accuracy, neural networks are black box models that are not easily interpretable. In addition to this, neural networks often have millions of learned parameters, making verification intractable [75]. One approach is to train a variational autoencoder (VAE) to generate trajectories [76, 77]. Another approach trains DNNs to output the parameters of Gaussian Mixture Models,

to represent multimodal distributions over trajectories [68, 49]. Such methods which use DNNs can achieve high accuracy up to short time horizons, but are inherently difficult to interpret and verify. In contrast to this, the methods presented in Chapter 5 and Chapter 6 are shown to be interpretable and verifiable.

2.1.2.2 Manoeuvres

A common approach in agent modelling is to assume that agents are following one of a distinct set of behaviours [78, 79]. When such methods are applied to autonomous driving [80, 81, 55, 82], the behaviours can represent manoeuvres such as lane following, changing lane, or turning. These approaches usually infer distributions over possible current manoeuvres for vehicles, so that the rest of the manoeuvre can be predicted. Manoeuvres are used in the method described in Chapter 4, where the assumption is made that other vehicles are using manoeuvres from a known set, and then planning is performed over the same set of manoeuvres.

2.1.2.3 Goal Recognition

Another general approach known as goal recognition or intent recognition is to first infer distributions over the goals or intentions of other agents, before predicting their trajectories. Some methods use this approach to infer whether a vehicle intends to yield while the ego vehicle is merging using a simple Bayesian model [80] or probabilistic graphical model [52]. Other methods dynamically generate a set of possible goals based on the local road layout, such as taking an exit at a junction, or reaching the visible end of a lane [83, 56]. This can allow accurate long term predictions to be made. One approach to goal recognition is through inverse planning [22, 23], where optimal plans are computed from the perspective of the other agent. Hanna et al. presented a goal recognition method named GOFI [84] which uses inverse planning to infer occluded factors. GOFI builds directly on the inverse planning based method presented in Chapter 4. Recent work by Antonello et al. [85] presented a trajectory prediction method which is a fusion of neural networks and inverse planning. Inverse planning-based methods can be accurate, interpretable and handle occlusion. However, these methods are typically slow and unverifiable due to the complexity of inverse planning. In Chapter 4, it is shown how inverse planning-based goal recognition can be integrated with an MCTS planner to create a full planning and prediction system for autonomous driving.

2.1.2.4 Inverse Reinforcement Learning

Many prediction methods rely on the assumption that other agents are rational and are attempting to minimise a known cost function. However, specifying a cost function that accurately reflects human preferences can be very difficult. Inverse Reinforcement Learning (IRL) methods can be used to learn cost function parameters from human driving data. Maximum entropy IRL [86] was used to learn a cost function represented by a weighted sum of features, through principles of matching feature expectations while maximising entropy. This approach was later extended as maximum entropy deep IRL [87], which represented the cost function with a ConvNet. Deep IRL can learn spatial relationships in the data, but loses interpretability. In addition, Finn et al. [88] found that IRL can be prone to overfitting unless domain specific regularisation techniques are used. Sadigh et al. [89] presented an alternative approach to IRL which does not require human demonstrations, but instead asks humans to show preferences between two trajectories. In contrast to the cost functions learned through data using IRL, in Chapter 4 a cost function with hand designed parameters is used, which kept the method simpler and did not require any training data. Methods such as IRL could potentially be used to automate this process, but that is outside the scope of this thesis.

2.2 Interpretable AI

There are many motivations for having interpretable models and algorithms. One motivation is trust, or faith in a model's performance and robustness. Increased user trust in AVs may significantly increase their adoption [90, 91, 92]. Although evaluation metrics can be used to empirically evaluate a model, such metrics may not accurately reflect real life objectives. Many real life objectives such as safety, legality and ethics are difficult to represent as objective functions or evaluation metrics. In the EU, a regulation giving the "right to an explanation" has been passed, and requires that automated decisions should be contestable [13]. Interpretability is also important in situations where the deployment environment may differ from data used for training and evaluation [93]. For example, AVs may be deployed to geographic locations different from where the system was evaluated, where people follow different driving styles and social conventions. Additionally, previously untested scenarios may occur. Interpretable models allow people to reason about how systems will generalise to unseen scenarios and data distributions which differ from those used in training and evaluation.

There is not a single widely agreed upon of definition of interpretability, however it generally refers to having an understanding of the mechanisms of a model. Lipton [12] identified several different types of interpretability, including simulatability, decomposability, and algorithmic transparency. Simulatability is the perhaps the strictest form of interpretability, and refers to whether a human can reasonably examine the entire model at once, and manually compute its output. An example of such a model is a shallow decision tree [94]. For decision trees, the time to compute the output, given by the path to a leaf node, grows more slowly than the total size of the model, given by the number of nodes. As the size of a decision tree grows large, it may lose simulatability. Decomposability refers to the ability to understand what the parameters of a model represent. For example, in a decision tree used in autonomous driving to make a decision whether to overtake [43], a node may have an intuitive interpretation such as "the speed of the vehicle ahead is less than 5 metres per second". The methods presented in Chapter 5 and Chapter 6 are decomposable in this manner, due to their usage of decision trees with interpretable features. Linear regression models also have this type of interpretability, where weights represent the importance of different features. Algorithmic transparency refers to whether the learning algorithm itself is well understood. For example, linear models can be shown to have a convex loss surface and are guaranteed to find a unique global optimum. Other methods such as heuristics used to train DNNs can be shown empirically to achieve good performance, but lack a formal theory of how they achieve such results. The system presented in Chapter 4, named IGP2, has algorithmic transparency due to its use of high level manoeuvres to interpret vehicle trajectories. Gyevnar et al. [14] have extended this work and present a method of generating natural language explanations for decisions made by IGP2.

One method of achieving some weak interpretability without sacrificing any performance is post-hoc interpretation, where an opaque model is used to make decisions, and then an explanation is generated afterwards. An example of this is to train a deep neural network to generate text explanations of an agent's decisions, using human explanations as training data [95, 96, 97]. This method can generate plausible sounding explanations, but such explanations may not faithfully represent how the agent actually arrived at its decisions. Another form of post-hoc interpretation is through surrogate models. Simple surrogate models which are simulatable can be used to assign feature importance for an input of a black box model [98, 99, 100]. However, such methods ignore how the black boxes work internally, and may not lead to sound interpretations. Another approach is to produce saliency maps which visualise the parts of an image that an agent is focusing

on, based on the magnitude of the gradient of the model output with respect to each pixel in the input image [101]. However, this approach does not explain why those parts of an image led the agent to makes its decisions.

2.2.1 Decision Trees

Decision trees (DTs) are tree-structured models which are traversed iteratively from a root node to a leaf node by making a decision at each node based on a feature from a set of input features. An output value is specified at each leaf node. As discussed in Section 2.2, DTs are easily interpretable by humans if they are not too deep and if the features are interpretable. DTs can be constructed by hand by an expert, or can be trained from data to perform classification or regression. The methods presented in Chapter 5 and Chapter 6 make use of DTs to infer the likelihood of long term goals for vehicles.

2.2.1.1 Decision Tree Training

There are several methods of training decision trees directly from data. Trees are usually trained in a greedy top down manner. ID3 [102] can be used to train DTs for classification based on categorical features. At each node, the feature which maximises information gain is selected. C4.5 [103] is an extension of ID3 which allows for scalar features. Classification and Regression Trees (CART) [104] is similar to C4.5, but can also perform regression. At each node, CART finds a splitting value for each scalar feature which minimises mean squared error. As the depth of a DT grows, the expected number of training samples present at each node decreases exponentially, which can lead to severe overfitting. Trees are iteratively grown up to a specified depth, after which they can be pruned to help reduce overfitting. In the work presented in 5, DTs are trained using CART. CART is then built on in 6 to introduce a novel DT training algorithm which can handle missing data.

2.2.1.2 Handling Missing Data with Decision Trees

There are many existing methods which allow DT inference with missing data [105]. Some methods compute the expected inference based on the distributions over the values of missing features [106, 102]. However, such methods introduce more complexity into the DT inference process. This reduces their interpretability, and may make such methods unverifiable. Another approach is lazy decision trees [107], where a new DT is trained for each inference made. However, this is computationally expensive and is not

suitable when inferences are needed in real time. As part of the work in Chapter 6, a novel DT training algorithm is introduced, which is designed to handle missing feature values while resulting in DTs which have fast, interpretable and verifiable inference.

2.2.1.3 Knowledge Distillation

In many domains, DNNs can achieve higher accuracy than DTs, likely due to their more powerful representational capabilities of DNNs. Another reason for this could be that DTs can easily overfit [108]. However, the better accuracy of DNNs comes at the expense of interpretability. One method of improving this trade off is to use knowledge distillation to train DTs to model the input-output mapping of a DNN. DTs trained in such a way often achieve better performance than DTs trained directly from data. The idea of model distillation [109] was presented as a way to compress the knowledge of a large and cumbersome DNN into a smaller, more efficient DNN. Rather than training directly on the training data, the smaller DNN is trained on the outputs of the larger DNN, which contain more information than the training labels. A similar approach can be used to train DTs by replacing the smaller DNN with a DT. Craven and Shavlik [110] presented a method for training DTs while using a neural network as an oracle to generate samples from a distribution similar to the training data. An alternative method is to use CART to train a DT on the outputs given by a DNN on existing training data [108]. Another approach trained a soft decision tree, where logistic regression over all over the features is performed at each node [111]. This increases the complexity of the model, reducing its interpretability. Several methods have distilled DNNs into gradient tree boosting models [112, 113, 114], which is claimed to improve interpretability over a DNN. However, such models rely on combining large numbers of DTs, which leads to poor simulatability as discussed in Section 2.2.

2.3 Formal Verification

Autonomous driving is a safety critical application, and it is important to have the ability to verify properties of planning and prediction methods. One approach is to empirically evaluate methods, however this can require billions of miles of real or simulated driving [15]. Another approach is to formally verify that certain properties of planning and prediction methods will always hold true under all conditions [16, 17].

DTs can easily be verified due to the simplicity of their inference process. Bastani et

al. [94] presented a method named VIPER which uses DTs to represent reinforcement learning policies, and used an off-the-shelf satisfiability modulo theories (SMT) solver to verify properties of the policy. VIPER first learns a neural network policy using reinforcement learning, and then uses imitation learning to learn a DT policy with similar behaviour. More recently, Schmidt et al. [115] used a similar approach to train a DT policy for autonomous driving. One drawback of such approaches is that a formal model of the environment is required, which may not accurately represent the real world environment. In contrast to this, in our goal recognition methods described in Chapters 5 and 6 we formally verify properties of our methods without needing any environment model. To the best of my knowledge, there is no existing related work which applies verification to GR or motion prediction for AVs.

As discussed in Section 2.1, many decision-making and motion prediction methods for AVs make use of DNNs. There has been significant work in developing methods for formally verifying neural networks [116]. However, these are typically small fully connected networks, ranging from hundreds of parameters [116] to hundreds of thousands of parameters [117]. DNNs such as those mentioned in Section 2.1.2.1 typically have millions of learned parameters and more complicated network architectures such as recurrent neural networks [76] or graph neural networks [118], making verification intractable.

2.4 Related Work and Desiderata

In this section, we include an analysis of how related planning and prediction methods perform when considering the desiderate defined in Section 1.1. In Table 2.1, we show whether each of the methods satisfy or do not satisfy each of the desiderata. For brevity, we do not included every single related work. We instead include a representative state-of-the-art method from each of the families of methods discussed in this chapter. As can be seen in table Table 2.1, none of the existing works surveyed satisfy all the stated desiderata.

Table 2.1: Comparison of planning and prediction methods for autonomous driving. It is desirable for methods to be fast, accurate, interpretable, verifiable, handle occlusions, and generalise across multiple scenarios.

Method	Fast	Accurate	Interp.	Verif.	Occl.	General.
Lattice Planning [28]	✓	✓	✓	×	×	✓
MPC [37]	×	✓	✓	×	×	✓
PILOT [38]	×	✓	×	×	×	✓
FSM [18]	✓	×	✓	✓	×	×
SafeVIPER [115]	✓	✓	✓	✓	×	×
MCTS [49]	×	✓	✓	×	×	✓
Dave-2 [11]	✓	×	×	×	✓	✓
DQN [62]	×	✓	×	×	✓	✓
MPDM [82]	×	✓	✓	×	×	×
Flash [85]	×	✓	×	×	×	✓
GOFI [84]	×	✓	✓	×	✓	×
Explainable DNN [97]	×	✓	✓	×	✓	✓
PRECOG [76]	✓	✓	×	×	✓	✓

Chapter 3

Problem Definition

In this thesis, we present several novel planning and prediction methods for autonomous driving, with a particular focus on goal recognition. This chapter presents a formal definition of the planning and goal recognition problems for autonomous driving. The notation introduced in this chapter is then used in following chapters, which introduce our novel methods.

3.1 Planning Problem

Let I be the set of vehicles in the local neighbourhood of the ego vehicle (including itself). At time t, each vehicle $i \in I$ is in a local state $s_t^i \in \mathcal{S}^i$, receives a local observation $o_t^i \in O^i$, and can choose an action $a_t^i \in \mathcal{A}^i$. We write $s_t \in \mathcal{S} = \times_i \mathcal{S}^i$ for the joint state and $s_{a:b}$ for the tuple $(s_a, ..., s_b)$, and similarly for $o_t \in O, a_t \in \mathcal{A}$. Observations depend on the joint state via $p(o_t^i|s_t)$, and actions depend on the observations via $p(a_t^i|o_{1:t}^i)$. In our system, a local state contains a vehicle's pose, velocity, and acceleration (we use the terms velocity and speed interchangeably); an observation contains the poses and velocities of nearby vehicles; and an action controls the vehicle's steering and acceleration. The probability of a sequence of joint states $s_{1:n}$ is given by

$$p(s_{1:n}) = \prod_{t=1}^{n-1} \int_O \int_{\mathcal{A}} p(o_t|s_t) p(a_t|o_{1:t}) p(s_{t+1}|s_t, a_t)\, do_t\, da_t \tag{3.1}$$

where $p(s_{t+1}|s_t, a_t)$ defines the joint vehicle dynamics, and we assume independent local observations and actions, $p(o_t|s_t) = \prod_i p(o_t^i|s_t)$ and $p(a_t|o_{1:t}) = \prod_i p(a_t^i|o_{1:t}^i)$. Vehicles react to other vehicles via their observations $o_{1:n}^i$.

We define the planning problem as finding an optimal policy π^* which selects the

actions for the ego vehicle, ε, to achieve a specified goal, G^ε, while optimising the driving trajectory via a defined reward function. Here, a policy is a function $\pi : (O^\varepsilon)^* \mapsto \mathcal{A}^\varepsilon$ which maps an observation sequence $o^\varepsilon_{1:n}$ to an action a^ε_t. A goal can be any subset of local states, $G^\varepsilon \subset \mathcal{S}^\varepsilon$. In this case, we focus on goals that specify target locations and "stopping goals" which specify a target location and zero velocity. Formally, define

$$\Omega_n = \left\{ s_{1:n} \,\middle|\, s^\varepsilon_n \in G^\varepsilon \wedge \forall m < n : s^\varepsilon_m \notin G^\varepsilon \right\} \tag{3.2}$$

where $s^\varepsilon_n \in G^\varepsilon$ means that s^ε_n satisfies G^ε. The second condition in (4.2) ensures that $\sum_{n=1}^{\infty} \int_{\Omega_n} p(s_{1:n}) ds_{1:n} \leq 1$ for any policy π, which is needed for soundness of the sum in (4.3). The problem is to find π^* such that

$$\pi^* \in \arg\max_{\pi} \sum_{n=1}^{\infty} \int_{\Omega_n} p(s_{1:n}) R^\varepsilon(s_{1:n})\, ds_{1:n} \tag{3.3}$$

where $R^i(s_{1:n})$ is vehicle i's reward for $s_{1:n}$. We define R^i as a weighted sum of reward elements based on trajectory execution time, longitudinal and lateral jerk, path curvature, and safety distance to leading vehicle.

3.2 Goal Recognition Problem

Expending on the preliminaries established in Section 3.1, the observations from the point of view of the ego vehicle between times a and b are denoted by $o^\varepsilon_{a:b}$. We assume a set of possible goals $G^i_t = \{g^{i,1}_t, g^{i,2}_t, ...\}$ for each vehicle i at time t, where a goal is a subset of states $g^{i,k}_t \subset \mathcal{S}^i$ such as a target location. We define the problem of goal recognition under occlusion as the task of inferring the probability distribution over goals based on past observations, $P(g^{i,k}_t | o^\varepsilon_{1:t}, \phi)$, where ϕ represents static scene information such as the road layout and static obstacles.

Chapter 4

Interpretable Goal-based Prediction and Planning

4.1 Introduction

An important problem in autonomous driving is predicting the intentions and future trajectories of other vehicles [6]. The problem is made significantly harder by the requirement to make accurate predictions in real time based on partially missing observations involving complicated multi-agent interaction.

One approach to make trajectory prediction tractable in autonomous driving is to make the assumption that vehicles use distinct manoeuvres from a finite set, for example turn, lane-change, lane-follow and stop [80, 119, 82, 55, 120, 81]. After observing the trajectory driven by a vehicle, the current manoeuvre of that vehicle is detected using a classifier. These methods have a significant limitation, as they can only detect the manoeuvre that the vehicle is currently executing. If a planner makes use of such predictions, it can only plan over timescales around the length of detected manoeuvres.

A different approach is to first define a set of possible *goals* (for example junction exits) for each non-ego vehicle, and then generate a trajectory from the vehicle's current observed state to each goal [23, 83, 56]. Methods using this approach can generate predictions over long durations, but cannot make predictions with high confidence unless the vehicle closely follows the generated trajectory.

Many recent trajectory prediction methods for autonomous driving have made use of deep learning [121, 76, 68, 69, 70, 77, 122]. There are now many large datasets available with data gathered by vehicles using sensors such as radar, video and LIDAR, and these datasets have been extensively used to train prediction models. Despite this,

is it still a difficult problem to make reliable predictions several seconds into the future, partly because of the highly coupled nature of traffic behaviour. One of the largest limitations of deep learning methods is that interpretable predictions are difficult to extract in a manner that can be easily integrated with hierarchical planning methods, all through there has been some recent progress in this area [123].

In order to predict the future manoeuvres of a vehicle, it is necessary to understand the reasons behind its past manoeuvres, as this can provide insight into its intended goals [78]. Knowing the goals of other vehicles can aid in predicting their future actions and trajectories, which is important for long-term planning. Our research, with examples given in Figure 4.2, shows how this type of reasoning can help address the issue of overly-conservative autonomous driving [124]. Additionally, by basing predictions on the interpretation of observed trajectories in terms of high-level manoeuvres, it is possible to intuitively understand and analyse the system, as well as justify its decisions. These concepts of interpretation and explanation are crucial [125] as we work to increase the trustworthiness of autonomous systems [126].

To this end, we propose *Interpretable Goal-based Prediction and Planning* (IGP2) which leverages the computational advantages of using a finite space of manoeuvres, but extends the approach to planning and prediction of *sequences* (i.e., plans) of manoeuvres. We achieve this via a novel integration of rational inverse planning [22, 127] to recognise the goals of other vehicles, with Monte Carlo Tree Search (MCTS) [128] to plan optimal manoeuvres for the ego vehicle. Inverse planning and MCTS utilise a shared set of defined manoeuvres to construct plans which are explainable by means of *rationality* principles, i.e. plans are optimal with respect to given metrics. We evaluate IGP2 in simulations of diverse urban driving scenarios, showing that (1) the system robustly recognises the goals of other vehicles, even if significant parts of a vehicle's trajectory are occluded, (2) goal recognition enables our vehicle to exploit opportunities to improve driving efficiency as measured by driving time compared to other prediction baselines, and (3) we are able to extract intuitive explanations for the predictions to justify the system's decisions.

We introduce *Interpretable Goal-based Prediction and Planning* (IGP2) as a solution that combines the computational benefits of using a finite set of manoeuvres with the ability to plan and predict *sequences* of manoeuvres. This is achieved by integrating rational inverse planning [22, 127] for recognising the goals of other vehicles with Monte Carlo Tree Search (MCTS) [128] for planning optimal manoeuvres for the ego vehicle. Both inverse planning and MCTS use a common set of defined manoeuvres

to create plans that can be explained through rationality principles, meaning they are optimal according to certain metrics. We tested IGP2 in simulations of various urban driving situations and found that (1) it robustly recognizes the goals of other vehicles even when parts of their trajectory are occluded, (2) it enables the vehicle to drive more efficiently, and (3) it can provide intuitive explanations for the predictions to justify the decisions made by the system.

To summarise our contributions:

- A combined goal recognition and multi-modal trajectory prediction method which makes use of rational inverse planning.
- Combining goal recognition with MCTS planning to search for optimal plans for the ego vehicle.
- Evaluation in simulated urban driving environments demonstrating the effectiveness of the system in accurately identifying the goals of other vehicles, increasing driving efficiency, and allowing interpretation of the predictions and plans of the ego vehicle.

4.2 Preliminaries and Problem Definition

Let I be the set of vehicles in the local neighbourhood of the ego vehicle (including itself). At time t, each vehicle $i \in I$ is in a local state $s_t^i \in \mathcal{S}^i$, receives a local observation $o_t^i \in O^i$, and can choose an action $a_t^i \in \mathcal{A}^i$. We write $s_t \in \mathcal{S} = \times_i \mathcal{S}^i$ for the joint state and $s_{a:b}$ for the tuple $(s_a, ..., s_b)$, and similarly for $o_t \in O, a_t \in \mathcal{A}$. Observations depend on the joint state via $p(o_t^i|s_t)$, and actions depend on the observations via $p(a_t^i|o_{1:t}^i)$. In our system, a local state contains a vehicle's pose, velocity, and acceleration (we use the terms velocity and speed interchangeably); an observation contains the poses and velocities of nearby vehicles; and an action controls the vehicle's steering and acceleration. The probability of a sequence of joint states $s_{1:n}$ is given by

$$p(s_{1:n}) = \prod_{t=1}^{n-1} \int_O \int_{\mathcal{A}} p(o_t|s_t) p(a_t|o_{1:t}) p(s_{t+1}|s_t, a_t)\, do_t\, da_t \tag{4.1}$$

where $p(s_{t+1}|s_t, a_t)$ defines the joint vehicle dynamics, and we assume independent local observations and actions, $p(o_t|s_t) = \prod_i p(o_t^i|s_t)$ and $p(a_t|o_{1:t}) = \prod_i p(a_t^i|o_{1:t}^i)$. Vehicles react to other vehicles via their observations $o_{1:n}^i$.

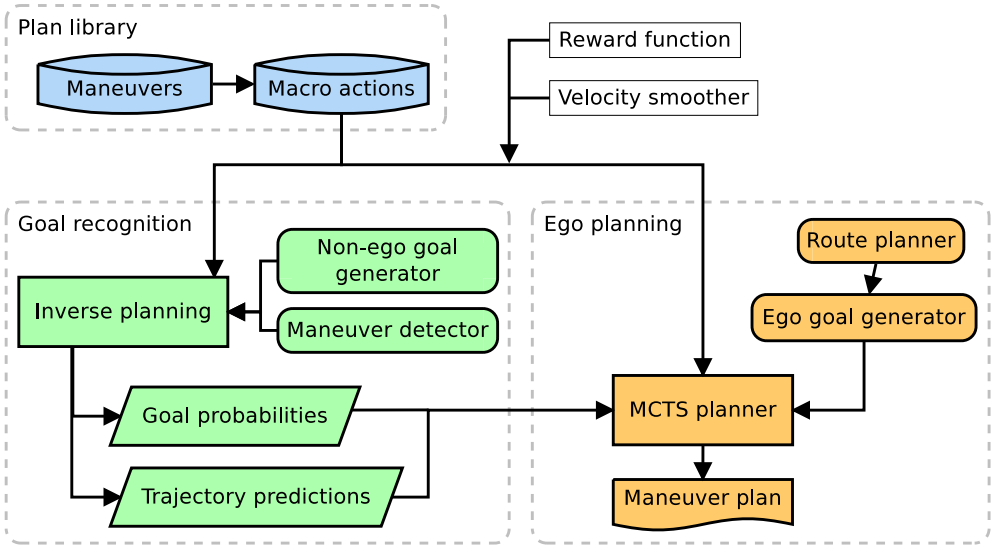


Figure 4.1: Overview of the IGP2 system.

We define the planning problem as finding an optimal policy π^* which selects the actions for the ego vehicle, ε, to achieve a specified goal, G^ε, while optimising the driving trajectory via a defined reward function. Here, a policy is a function $\pi : (O^\varepsilon)^* \mapsto \mathcal{A}^\varepsilon$ which maps an observation sequence $o^\varepsilon_{1:n}$ to an action a^ε_t. A goal can be any subset of local states, $G^\varepsilon \subset \mathcal{S}^\varepsilon$. In this case, we focus on goals that specify target locations and "stopping goals" which specify a target location and zero velocity. Formally, define

$$\Omega_n = \left\{ s_{1:n} \,\middle|\, s^\varepsilon_n \in G^\varepsilon \wedge \forall m < n : s^\varepsilon_m \notin G^\varepsilon \right\} \tag{4.2}$$

where $s^\varepsilon_n \in G^\varepsilon$ means that s^ε_n satisfies G^ε. The second condition in (4.2) ensures that $\sum_{n=1}^{\infty} \int_{\Omega_n} p(s_{1:n}) ds_{1:n} \leq 1$ for any policy π, which is needed for soundness of the sum in (4.3). The problem is to find π^* such that

$$\pi^* \in \arg\max_{\pi} \sum_{n=1}^{\infty} \int_{\Omega_n} p(s_{1:n}) R^\varepsilon(s_{1:n})\, ds_{1:n} \tag{4.3}$$

where $R^i(s_{1:n})$ is vehicle i's reward for $s_{1:n}$. We define R^i as a weighted sum of reward elements based on trajectory execution time, longitudinal and lateral jerk, path curvature, and safety distance to leading vehicle.

Macro action:	Manoeuvre sequence (manoeuvre parameters in brackets):
Continue	*lane-follow* (end of visible lane)
Continue next exit	*lane-follow* (next exit point)
Change left/right	*lane-follow* (until target lane clear), *lane-change-left/right*
Exit left/right	*lane-follow* (exit point), *give-way* (relevant lanes), *turn-left/right*
Stop	*lane-follow* (close to stopping point), *stop*

Table 4.1: The set of macro actions used by our system. Each macro action is composed of one or more manoeuvres which have their parameters automatically set.

Macro action:	Additional applicability condition:
Continue	—
Continue next exit	Must be in roundabout and not in outer-lane
Change left/right	There is a lane to the left/right
Exit left/right	Exit point on same lane ahead of car and in correct direction
Stop	There is a stopping goal ahead of the car on the current lane

Table 4.2: Additional applicability conditions for each macro action used in our system.

4.3 IGP2: Interpretable Goal-based Prediction and Planning

Our methodology is based on two assumptions: (1) each vehicle has an unknown goal from a set of potential goals, and (2) each vehicle acts according to a plan constructed from a finite set of predefined manoeuvres.

The diagram in Figure 4.1 illustrates the components of our proposed system called IGP2. In summary, IGP2 approximates the optimal ego policy π^* for the ego vehicle as follows: First by potential goals for each non-ego vehicle are identified, and then plans are generated for each goal. The probabilities of each goal and the predicted trajectories of each non-ego vehicle are then used in a simulation process using Monte Carlo Tree Search (MCTS) algorithm to determine the optimal plan over manoeuvres for the ego vehicle. To make the process more efficient and limit the search depth required, both the inverse planning and the MCTS use a shared set of macro actions that concatenate manoeuvres using contextual information. The system's components are further explained in the following sections.

4.3.1 Manoeuvres

We make the assumption that at any given time, each vehicle is executing a manoeuvre from the following list: *lane-follow, turn-left/right, lane-change-left/right, stop, give-way*. Applicability conditions and termination conditions are specified for each manoeuvre

ω. As an example, the lane-change-left macro action, can only be executed if there is a lane to the left of the vehicle with the same driving direction, and the manoeuvre terminates when the vehicle reaches the new lane and is correctly aligned with it. Some manoeuvres have adjustable parameters, for example the "follow-lane" manoeuvre has a parameter that determines after what distance it should terminate.

If applicable, a manoeuvre specifies a local trajectory $\hat{s}^i_{1:n}$ to be followed by the vehicle, which includes a reference path in the global coordinate frame and target velocities along the path. For convenience in exposition, we assume that $\hat{s}^i$ uses the same representation and indexing as s^i, but in general this does not have to be the case (for example, $\hat{s}$ may be indexed by longitudinal position rather than time, which can be interpolated to time indices). In our system, the reference path is generated via a Bezier spline function fitted to a set of points extracted from the road topology, and target velocities are set using domain heuristics similar to [129].

If a manoeuvre is applicable, it specifies a local trajectory $\hat{s}^i_{1:n}$ which should be followed by the vehicle. This trajectory includes a path specified in the global coordinate frame, along with target velocities at each point along the path. To generate the reference path, we fit a Bezier spline function to a set of points based on the road layout, and then set the target velocities using domain heuristics similar to [129].

4.3.2 Macro Actions

Macro actions specify common sequences of manoeuvres and automatically set the free parameters (if any) in manoeuvres based on context information such as road layout. Table 4.1 specifies the macro actions used in our system. The applicability condition of a macro action is given by the applicability condition of the first manoeuvre in the macro action, as well as optional additional conditions, which are specified in Table 4.2. The termination condition of a macro action is given by the termination condition of the last manoeuvre in the macro action.

4.3.3 Velocity Smoothing

To obtain a feasible trajectory across manoeuvres for vehicle i, we define a velocity smoothing operation which optimises the target velocities in a given trajectory $\hat{s}^i_{1:n}$. Let $\hat{x}_t$ be the longitudinal position on the reference path at $\hat{s}^i_t$ and $\hat{v}_t$ its target velocity, for $1 \leq t \leq n$. We define $\kappa : x \rightarrow v$ as the piecewise linear interpolation of target velocities between points $\hat{x}_t$. Given the time elapsed between two time steps, Δt; the maximum

velocity and acceleration, v_{max}/a_{max}; and setting $x_1 = \hat{x}_1, v_1 = \hat{v}_1$, we define velocity smoothing as

$$\begin{aligned} \min_{x_{2:n}, v_{2:n}} \quad & \sum_{t=1}^{n} ||v_t - \kappa(x_t)||_2 + \lambda \sum_{t=1}^{n-1} ||v_{t+1} - v_t||_2 \\ & \text{s.t. } x_{t+1} = x_t + v_t \Delta t \\ & 0 < v_t < v_{\max},\ v_t \leq \kappa(x_t) \\ & |v_{t+1} - v_t| < a_{\max} \Delta t \end{aligned} \tag{4.4}$$

where $\lambda > 0$ is the weight given to the acceleration part of the optimisation objective. Eq. (5.5) is a nonlinear non-convex optimisation problem which can be solved, e.g., using a primal-dual interior point method (we use IPOPT [130]). From the solution of the problem, $(x_{2:n}, v_{2:n})$, we interpolate to obtain the achievable velocities at the original points $\hat{x}_t$.

4.3.4 Goal Recognition

We assume that each non-ego vehicle i seeks to reach one of a finite number of possible goals $G^i \in \mathcal{G}^i$, using plans constructed from our defined macro actions. We use the framework of rational inverse planning [22, 127] to compute a Bayesian posterior distribution over i's goals at time t

$$p(G^i | s_{1:t}) \propto L(s_{1:t} | G^i) p(G^i) \tag{4.5}$$

where $L(s_{1:t}|G^i)$ is the likelihood of i's observed trajectory assuming its goal is G^i, and $p(G^i)$ specifies the prior probability of G^i.

The likelihood is a function of the reward difference between two plans: the reward $\hat{r}$ of the optimal trajectory from i's initial observed state s_1^i to goal G^i after velocity smoothing, and the reward $\bar{r}$ of the trajectory which follows the observed trajectory until time t and then continues optimally to goal G^i, with smoothing applied only to the trajectory after t. The likelihood is defined as

$$L(s_{1:t} | G^i) = \exp(\beta(\bar{r} - \hat{r})) \tag{4.6}$$

where β is a scaling parameter (we use $\beta = 1$). This likelihood definition assumes that vehicles drive approximately *rationally* (i.e., optimally) to achieve their goals while allowing for some deviation. If a goal is infeasible, we set its probability to zero.

Algorithm 1 shows the pseudocode for our goal recognition algorithm, with further

details in below subsections.

4.3.4.1 Goal Generation

A heuristic function is used to generate a set of possible goals $\mathcal{G}^i$ for vehicle i based on its location and context information such as road layout. In our system, we include goals for the visible end of the current road and connecting roads (bounded by the ego vehicle's view region). In addition to such static goals, it is also possible to add dynamic goals which depend on current traffic. For example, in the dense merging scenario shown in Figure 4.2d, stopping goals are dynamically added to model a vehicle's intention to allow the ego vehicle to merge in front.

4.3.4.2 Manoeuvre Detection

Manoeuvre detection is used to detect the current executed manoeuvre of a vehicle (at time t), allowing inverse planning to complete the manoeuvre before planning onward. We assume a module which computes probabilities over current manoeuvres, $p(\omega^i)$, for each vehicle i. One option is Bayesian changepoint detection (e.g. [131]). The details of manoeuvre detection are outside the scope of our paper, and in our experiments we use a simulated detector (cf. Sec 4.4.2). As different current manoeuvres may hint at different goals, we perform inverse planning for each possible current manoeuvre for which $p(\omega^i) > 0$. Thus, each current manoeuvre produces its associated posterior probabilities over goals, denoted by $p(G^i \,|\, s_{1:t}, \omega^i)$.

4.3.4.3 Inverse Planning

Inverse planning is done using A* search [132] over macro actions. A* starts after completing the current manoeuvre ω^i which produces the initial trajectory $\hat{s}_{1:\tau}$. Each search node q corresponds to a state $s \in \mathcal{S}$, with initial node at state $\hat{s}_\tau$, and macro actions are filtered by their applicability conditions applied to s. A* chooses the next macro action leading to a node q' which has the lowest estimated total cost[1] to goal G^i, given by $f(q') = l(q') + h(q')$. The cost $l(q')$ to reach the node q' is given by the driving time from i's location in the initial search node to its location in q', following the trajectories returned by the macro actions leading to q'. A* uses the assumption that all other vehicles not planned for use a constant-velocity lane-following model after

[1] Here we use the term "cost" in keeping with standard A* terminology and to differentiate from the reward function defined in Sec. 4.2.

Algorithm 1 Goal recognition algorithm

Input: vehicle i, current manoeuvre ω^i, observations $s_{1:t}$
Returns: goal probabilities $p(G^i|s_{1:t},\omega^i)$

1: Generate possible goals $G^i \in \mathcal{G}^i$ from state s_t^i
2: Set prior probabilities $p(G^i)$ (e.g. uniform)
3: **for all** $G^i \in \mathcal{G}^i$ **do**
4: $\quad \hat{s}^i_{1:n} \leftarrow$ A*(ω^i) from $\hat{s}^i_1 = s^i_1$ to G^i
5: $\quad$ Apply velocity smoothing to $\hat{s}^i_{1:n}$
6: $\quad \hat{r} \leftarrow$ reward $R^i(\hat{s}^i_{1:n})$
7: $\quad \bar{s}^i_{1:m} \leftarrow$ A*(ω^i) from $\bar{s}^i_t$ to G^i, with $\bar{s}^i_{1:t} = s^i_{1:t}$
8: $\quad$ Apply velocity smoothing to $\bar{s}^i_{t+1:m}$
9: $\quad \bar{r} \leftarrow$ reward $R^i(\bar{s}^i_{1:m})$
10: $\quad L(s_{1:t}|G^i,\omega^i) \leftarrow \exp(\beta(\bar{r}-\hat{r}))$
11: Return $p(G^i|s_{1:t},\omega^i) \propto L(s_{1:t}|G^i,\omega^i)\,p(G^i)$

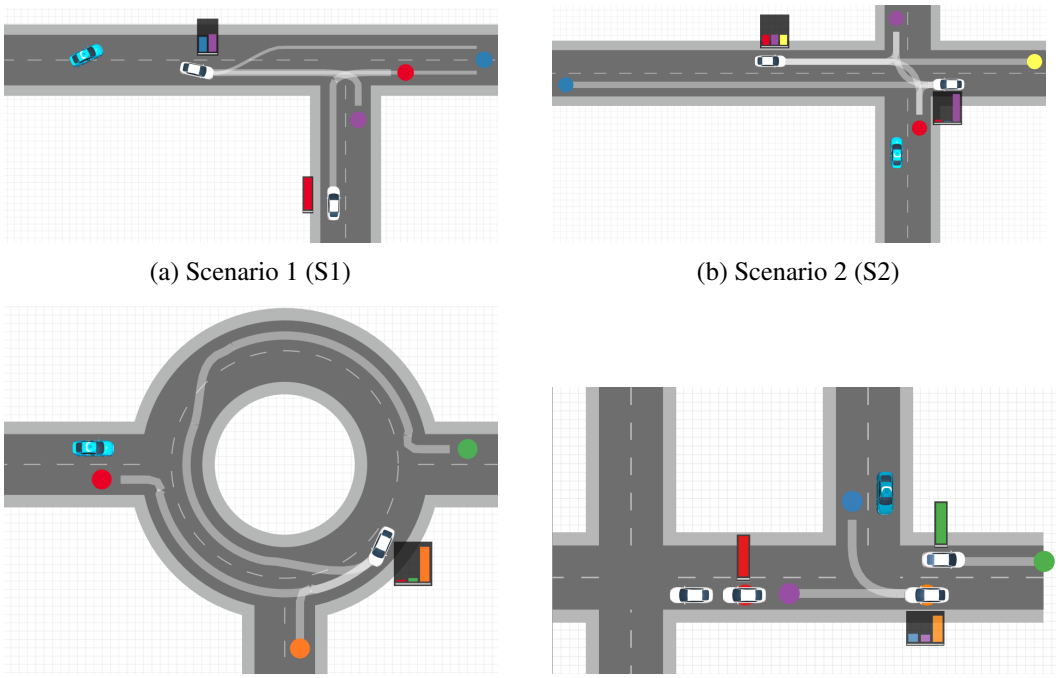
their observed trajectories. We do not check for collisions during inverse planning. The cost heuristic $h(q')$ to estimate remaining cost from q' to goal G^i is given by the driving time from i's location in q' to goal via straight line at speed limit. This definition of $h(q')$ is admissible as per A* theory, which ensures that the search returns an optimal plan. After the optimal plan is found, we extract the complete trajectory $\hat{s}^i_{1:n}$ from the manoeuvres in the plan and initial segment $\hat{s}_{1:\tau}$.

4.3.4.4 Trajectory Prediction

The method performs multi-modal trajectory prediction, as predictions are made for multiple possible goals. Our system also predicts multiple plausible trajectories for a given vehicle and goal. This is required because there are situations in which different trajectories may be (near-optimal) but may lead to different predictions which could require different behaviour on the part of the ego vehicle. We run A* search for a fixed amount of time and let it compute a set of plans with associated rewards (up to some fixed number of plans). Any time A* search finds a node that reaches the goal, the corresponding plan is added to the set of plans. Given a set of smoothed trajectories $\{\hat{s}^{i,k}_{1:n}|\omega^i,G^i\}_{k=1..K}$ to goal G^i with initial manoeuvre ω^i and associated reward $r_k = R^i(\hat{s}^{i,k}_{1:n})$, we compute a distribution over the trajectories via a Boltzmann distribution

$$p(\hat{s}^{i,k}_{1:n}) \propto \exp(\gamma r_k) \tag{4.7}$$

where γ is a scaling parameter (we use $\gamma = 1$). Similar to Eq. (4.6), Eq. (4.7) encodes the assumption that trajectories which are closer to optimal are more likely.



(a) Scenario 1 (S1)

(b) Scenario 2 (S2)

(c) Scenario 3 (S3)

(d) Scenario 4 (S4)

Figure 4.2: **IGP2 in 4 test scenarios.** Ego vehicle shown in blue. Bar plots show goal probabilities for non-ego vehicles. For each goal, up to two of the most probable predicted trajectories to goal are shown with thickness proportional to probability. (a) **S1**: Ego's goal is blue goal. Vehicle V_1 is on the ego's road, V_1 changes from left to right lane, biasing the ego prediction towards the belief that V_1 will exit, since a lane change would be irrational if V_1's goal was to go east. As exiting will require a significant slowdown, the ego decides to switch lanes to avoid being slowed down too. (b) **S2**: Ego's goal is blue goal. Vehicle V_1 is approaching the junction from the east and vehicle V_2 from the west. As V_1 approaches the junction, slows down and waits to take a turn, the ego's belief that V_1 will turn right increases significantly, since it would be irrational to stop if the goal was to turn left or go straight. Since the ego recognised V_1's goal is to go north, it predicts that V_1 will wait until V_2 has passed, giving the ego an opportunity to enter the road. (c) **S3**: Ego's goal is green goal. As V_1 changes from the inside to the outside lane of the roundabout and decreases its speed, it significantly biases the ego prediction towards the belief that V_1 will take the south exit, since that is the rational course of action for that goal. This encourages the ego to enter the roundabout while V_1 is still in roundabout. (d) **S4**: Ego's goal is purple goal. With two vehicles stopped at the junction at a traffic light, vehicle V_1 is approaching them from behind, and vehicle V_2 is crossing in the opposite direction. When V_1 reaches zero velocity, the goal generation function adds a stopping goal (orange) for V_1 in its current position, shifting the goal distribution towards it since stopping is not rational for the north/west goals. The interpretation is that V_1 wants the ego to merge in front of V_1, which the ego then does.

Algorithm 2 Monte Carlo Tree Search algorithm

Returns: optimal manoeuvre for ego vehicle ε in state s_t

Perform K simulations:

Search node $q.s \leftarrow s_t$ (*root* node)
Search depth $d \leftarrow 0$
for all $i \in I \setminus \{\varepsilon\}$ **do**
    Sample current manoeuvre $\omega^i \sim p(\omega^i)$
    Sample goal $G^i \sim p(G^i \mid s_{1:t}, \omega^i)$
    Sample trajectory $\hat{s}^i_{1:n} \in \{\hat{s}^{i,k}_{1:n} \mid \omega^i, G^i\}$ with $p(\hat{s}^{i,k}_{1:n})$
while $d < d_{max}$ **do**
    Select macro action μ for ε applicable in $q.s$
    $\hat{s}_{\tau:\mathfrak{l}} \leftarrow$ Simulate μ until it terminates, with non-ego vehicles following their sampled trajectories $\hat{s}^i_{1:n}$
    $r \leftarrow \emptyset$
    if ego vehicle collides during $\hat{s}_{\tau:\mathfrak{l}}$ **then**
        $r \leftarrow r_{coll}$
    else if $\hat{s}^{\varepsilon}_{\mathfrak{l}}$ achieves ego goal G^{ε} **then**
        $r \leftarrow R^{\varepsilon}(\hat{s}_{t:n})$
    else if $d = d_{max} - 1$ **then**
        $r \leftarrow r_{term}$
    if $r \neq \emptyset$ **then**
        Use (4.8) to backprop r along search branches (q, μ, q') that generated the simulation
        Start next simulation
    $q'.s = \hat{s}_{\mathfrak{l}}$; $q \leftarrow q'$; $d \leftarrow d + 1$

Return manoeuvre for ε in s_t, $\mu \in \arg\max_{\mu} Q(root, \mu)$

4.3.5 Ego Vehicle Planning

To compute an optimal plan for the ego vehicle, we use the goal probabilities and predicted trajectories to inform a Monte Carlo Tree Search (MCTS) algorithm [128] (see Algorithm 2).

The algorithm performs a number of closed-loop simulations $\hat{s}_{t:n}$, starting in the current state $\hat{s}_t = s_t$ down to some fixed search depth or until a goal state is reached. At the start of each simulation, for each non-ego vehicle, we first sample a current manoeuvre, then goal, and then trajectory for the vehicle using the associated probabilities (cf. Section 4.3.4). Each node q in the search tree corresponds to a state $s \in \mathcal{S}$ and macro actions are filtered by their applicability conditions applied to s. After selecting a macro action μ using some exploration technique (we use UCB1 [133]), the state in the current search node is forward-simulated based on the trajectory generated by the macro action μ and the sampled trajectories of non-ego vehicles, resulting in a partial trajectory $\hat{s}_{\tau:\mathfrak{l}}$ and new search node q' with state $\hat{s}_{\mathfrak{l}}$. Forward-simulation of trajectories uses a combination of proportional control and adaptive cruise control (based on IDM [134]) to control a vehicle's acceleration and steering. Termination conditions of manoeuvres

are monitored in each time step based on the vehicle's observations. Collision checking is performed on $\hat{s}_{\tau:\mathfrak{l}}$ to check whether the ego vehicle collided, in which case we set the reward to $r \leftarrow r_{coll}$ which is back-propagated using (4.8), where r_{coll} is a method parameter. Otherwise, if the new state $\hat{s}_{\mathfrak{l}}$ achieves the ego goal $G^{\mathfrak{e}}$, we compute the reward for back-propagation as $r \leftarrow R^{\mathfrak{e}}(\hat{s}_{t:n})$. If the search reached its maximum depth d_{max} without colliding or achieving the goal, we set $r \leftarrow r_{term}$ which can be a constant or based on heuristic reward estimates similar to A* search.

The reward r is back-propagated through search branches (q,μ,q') that generated the simulation, using a 1-step off-policy update function (similar to Q-learning [135])

$$Q(q,\mu) \leftarrow Q(q,\mu) + \begin{cases} \delta^{-1}[r - Q(q,\mu)] \text{ if } q \text{ leaf node, else} \\ \delta^{-1}[\max_{\mu'} Q(q',\mu') - Q(q,\mu)] \end{cases} \tag{4.8}$$

where δ is the number of times that macro action μ has been selected in q. After the simulations are completed, the algorithm selects the best macro action for execution in s_t from the root node, $\arg\max_{\mu} Q(root,\mu)$.

4.3.5.1 Closed-Loop Simulation

Closed-loop simulation uses a combination of proportional control and adaptive cruise control (ACC). Two independent proportional controllers control the acceleration and steering of the vehicle. If there is another vehicle close ahead of the controlled vehicle, control is given to ACC which keeps the vehicle at a safe distance to the leading vehicle (our ACC is based on IDM [134]). No velocity smoothing is applied since the combination of proportional/ACC control achieves approximately smooth control. Termination conditions in manoeuvres are monitored in each time step based on the vehicle's observations.

4.3.5.2 Open-Loop Simulation

Open-loop simulation works in the same way as in A* search (see Sec. 4.3.4.3), by setting the vehicle's position and velocity directly as specified in trajectory. Hence, there is no automatic distance keeping in open-loop control. Velocity smoothing is applied to the trajectory to improve realism of the prediction. Termination conditions in manoeuvres such as "wait until oncoming traffic is clear", e.g. as used in *give-way* manoeuvre, are realised by waiting until traffic is *predicted* to be clear assuming that non-controlled vehicles use a constant-velocity lane-following model.

4.4 Evaluation

In this section we present an evaluation of IGP2 across several urban driving scenarios, and show that: (1) Our goal recognition method can accurately infer the goals of other vehicles; (2) Driving efficiency, measured as driving time, is improved by robust goal recognition; (3) Decisions made by the system can be justified by extracting intuitive explanations. A video showing IGP2 in operation in these scenarios is available at: `https://www.five.ai/igp2`.

4.4.1 Scenarios

We use two sets of scenario instances. For in-depth analysis of goal recognition and planning, we use four defined local interaction scenarios, shown in Figure 4.2. For each of these scenarios, we generate 100 instances with randomly offset initial longitudinal positions ($\sim[-10, +10]$ meters) and initial speed sampled from range $[5, 10]$ m/s for each vehicle, including ego vehicle. Here the ego vehicle observes the whole scenario. To further assess IGP2's ability to complete full routes with random traffic, we use two random town layouts shown in Figure 4.3. Each town spans an area of 0.16 square kilometres and consists of roads, crossings, and roundabouts with 2–4 lanes each. Each junction has one defined priority road. The ego vehicle's observation radius in towns is 50 meters. Non-ego vehicles are spawned within 25 meters outside the ego observation radius, with random road, lane, speed, and goal. The total number of non-ego vehicles within the ego radius and spawning radius is kept at 8 to maintain a consistent medium-to-high level of traffic. In each town, we generate 10 instances by choosing random routes for the ego vehicle to complete. The ego vehicle's goal is continually updated to be the outermost point on the route within the ego observation radius. In all simulations, the non-ego vehicles use manual heuristics to select from the manoeuvres in Section 4.3.1 to reach their goals. All vehicles use independent proportional controllers for acceleration and steering, and IDM [134] for automatic distance-keeping. Vehicle motion is simulated using a kinematic bicycle model.

4.4.2 Algorithms & Parameters

We compare the following algorithms in scenarios S1–S4. **IGP2:** full system using goal recognition and MCTS. **IGP2-MAP:** like IGP2, but MCTS uses only the most probable goal and trajectory for each vehicle. **CVel:** MCTS without goal recognition,

replaced by constant-velocity lane-following prediction after completion of current manoeuvre. **CVel-Avg:** like CVel, but uses velocity averaged over the past 2 seconds. **Cons:** like CVel, but using a conservative *give-way* manoeuvre which always waits until all oncoming vehicles on priority lanes have passed. In the town scenarios we focus on IGP2 and Cons, and additionally compare to **SH-CVel** which works similarly to MPDM [55]: it simulates each macro action followed by a default *Continue* macro action, using CVel prediction for non-ego vehicles, then choosing the macro action with maximum estimated reward. (SH stands for "short horizon" as the search depth is effectively limited to 1.)

We simulate noisy manoeuvre detection (cf. Sec. 4.3.4.2) by giving 0.9 probability to the current executed manoeuvre of the non-ego vehicle and the rest uniformly to other manoeuvres. Prior probabilities over non-ego goals are uniform. A* computes up to two predicted trajectories for each non-ego vehicle and goal. MCTS is run at a frequency of 1 Hz, performs $K = 30$ simulations with a maximum search depth of $d_{max} = 5$, and uses $r_{coll} = r_{term} = -1$. We set $\lambda = 10$ for velocity smoothing (cf. Eq. (5.5)).

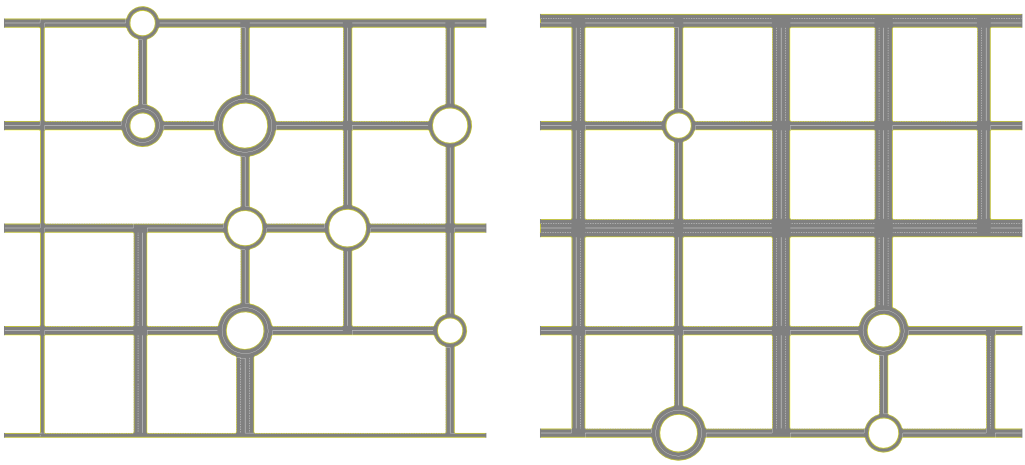


Figure 4.3: Town 1 and Town 2 layouts.

4.4.3 Results

4.4.3.1 Goal probabilities

Figure 4.4 shows the average probability over time assigned to the true goal in scenarios S1–S4. In all tested scenario instances, we observe that the probability increases with growing evidence and at different rates depending on random scenario initialisation. Snapshots of goal probabilities (shown as bar plots) associated with the non-ego's most probable current manoeuvre can be seen in Figure 4.2. We also tested the method's

robustness to missing segments in the observed trajectory of a vehicle. In scenarios S1 and S3 we removed the entire *lane-change* manoeuvre from the observed trajectory (but keeping the short lane-follow segment before the lane change). To deal with occlusion, we applied A* search before the beginning of each missing segment to reach the beginning of the next observed segment, thereby "filling the gaps" in the trajectory. Afterwards, we applied velocity smoothing to the reconstructed trajectory. The results are shown as dashed lines in Figure 4.4, showing that even under significant occlusion, the method is able to correctly recognise a vehicle's goal.

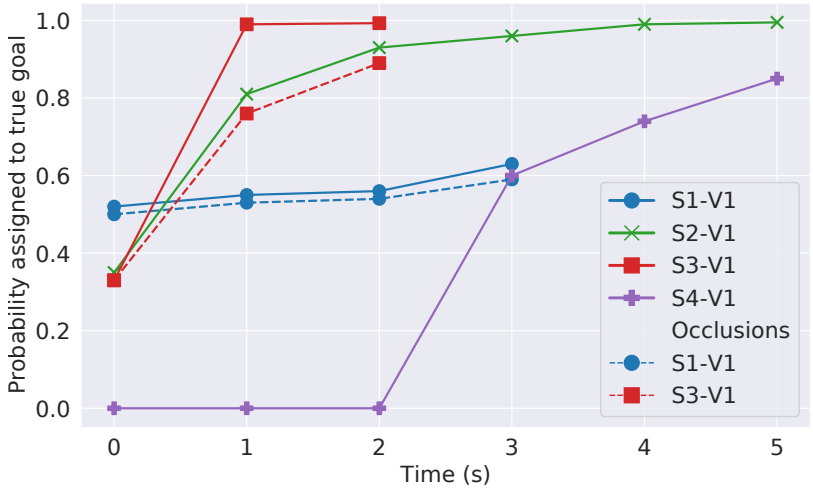


Figure 4.4: Average probability given to true goal of selected vehicles in scenarios S1–S4. Note: lines for S1/S3 are shorter than indicated in Tab. 4.3 since possible vehicle goals change after exit points are reached and we only show lines for initial possible goals.

4.4.3.2 Driving times

Table 4.3 shows the average driving times required of each algorithm in scenarios S1–S4. Goal recognition enabled IGP2 and IGP2-MAP to reduce their driving times. **(S1)** All algorithms change lanes to avoid being slowed down by V1, leading to same driving times, however IGP2 and IGP2-MAP initiate the lane change before all other algorithms by recognising V1's intended goal. **(S2)** Cons waits for V_1 to clear the lane, which in turn must wait for V_2 to pass. IGP2 and IGP2-MAP anticipate this behaviour, allowing them to enter the road earlier. CVel and CVel-Avg wait for V_1 to reach near-zero velocity. **(S3)** IGP2 and IGP2-MAP are able to enter early as they recognise V_1's goal to exit the roundabout, while CVel, CVel-Avg, and Cons wait for V_1 to exit. **(S4)** Cons waits until V_1 decides to close the gap after which the ego can enter the road. IGP2 and IGP2-MAP recognise V_1's goal and merge in front.

IGP2-MAP achieved shorter driving times than IGP2 on some scenario instances

	S1	S2	S3	S4
IGP2	$5.97 \pm .02$	$7.24 \pm .05$	$8.54 \pm .05$	$10.83 \pm .03$
IGP2-MAP	$5.99 \pm .02$	$7.23 \pm .05$	$8.36 \pm .06$	$10.40 \pm .03$
CVel	$6.04 \pm .03$	$9.80 \pm .17$	$10.49 \pm .09$	$12.83 \pm .03$
CVel-Avg	$6.01 \pm .02$	$11.31 \pm .17$	$10.49 \pm .09$	$13.59 \pm .02$
Cons	$6.01 \pm .02$	$12.89 \pm .03$	$10.90 \pm .04$	$16.78 \pm .02$

Table 4.3: Average driving time (seconds) required to complete scenario instances from S1–S4, with standard error.

(such as S3 and S4). This is because IGP2-MAP commits to the most-likely goal and trajectory of other vehicles, while IGP2 also considers residual uncertainty about goals and trajectories, which may lead MCTS to select more cautious actions in some situations. The limitation of IGP2-MAP can be seen when simulating unexpected (irrational) behaviours in other vehicles. To test this, we compared IGP2 and IGP2-MAP on instances from S3 and S4 which were modified such that V1, after slowing down, suddenly accelerates and continues straight (rather than exiting as in S3, or stopping as in S4). In these cases we observed a 2-3% collision rate for IGP2-MAP (in all collisions, V1 collided into the ego) while IGP2 produced no collisions. These results show that IGP2 exhibits safer driving than IGP2-MAP by accounting for uncertainty over goals and trajectories.

Figure 4.5 shows the driving times of IGP2 and Cons for the routes in the two towns. Both algorithms completed all the routes. Goal recognition allowed IGP2 to reduce its driving times substantially by exploiting multiple opportunities for proactive lane changes and road/junction entries. In contrast, Cons exhibited more conservative driving and often waited considerably longer at junctions or before taking a turn until traffic cleared up. SH-CVel was unable to complete any of the given routes, as its short planning horizon often caused it to take a wrong turn (thus failing the instance).

4.4.3.3 Interpretability

We are able to extract intuitive explanations for the predictions and decisions made by IGP2. The explanations are given in the caption of Figure 4.2.

4.4.3.4 Scalability

The goal recognition method can easily scale to make inferences about the goals of many non-ego vehicles. In the open-world experiments described in Section 4.4.1, inferences were made for 8 different non-ego vehicles. As the goals of each vehicle are

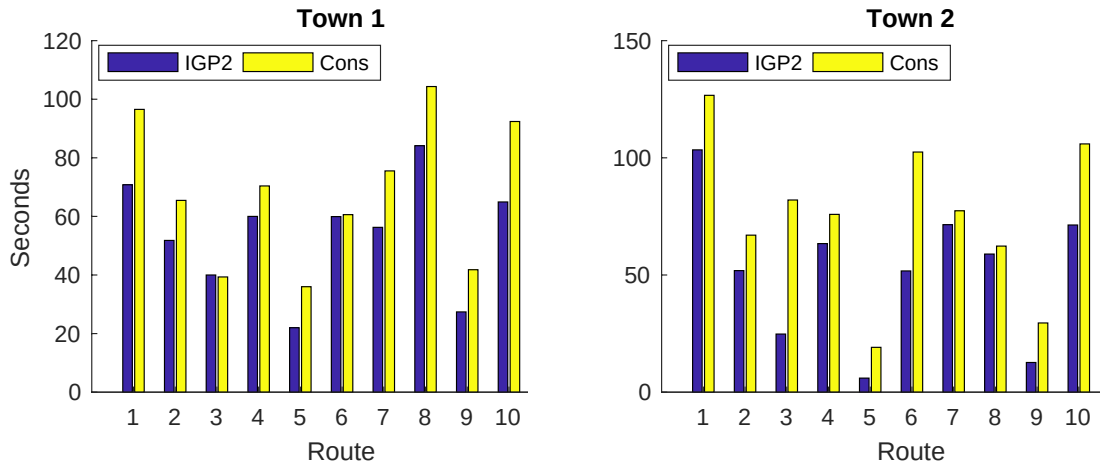


Figure 4.5: Driving times (seconds) of IGP2 and Cons for 10 routes in Town 1 and Town 2.

independent, inferences for each vehicle can be made in parallel. The method can be further parallelised by computing the optimal plans for each goal from the initial and current position of each non-ego vehicle in parallel.

4.5 Limitations

While IGP2 was shown to be effective in certain scenarios, it is crucial to acknowledge the limitations of this approach to provide a comprehensive understanding of its applicability in real-world driving situations. IGP2 makes several assumptions that may impact its performance and accuracy, and our evaluation also has some limitations.

One of the primary limitations of our evaluation is its reliance on simulations, which may only represent an idealized version of real-world driver behaviour. The discrepancy between the simulated behaviour and actual driver behaviour may lead to greater inaccuracies in the method's predictions when applied to real-world scenarios.

Moreover, IGP2 assumes that vehicles only perform maneuvers and macro actions from a predefined library. The predefined library may be incomplete or not sufficiently diverse, leading to situations where the actual actions taken by vehicles are not accurately captured and predicted by IGP2.

Another assumption made by IGP2 is that non-ego vehicles are approximately rational and select actions to minimise a predefined cost function. However, this assumption may not accurately reflect the preferences of actual agents in real-world driving scenarios. Drivers may have different objectives, driving styles, or preferences, which may not be accurately captured by the predefined cost function.

Another limiting assumption of IGP2 is that perception of the surrounding scene is

perfect. However, in real-world scenarios there may be significant noise and inaccuracy in perception. Such inaccuracies may come from sensor noise or model generalisation failures.

4.6 Conclusion

In this chapter, we presented an autonomous driving system named IGP2. This system uses rational inverse planning to recognise the goals of other vehicles and uses this to inform an integrated planning and prediction system that can operate over long-term horizons. We evaluated IGP2 across several urban driving scenarios and showed that IGP2 can accurately identify the goals of other vehicles, leading to improved driving efficiency. Through these scenarios, we also demonstrated how predictions generated by IGP2 can be intuitively interpreted, allowing the decisions made by the system to be explained. IGP2 is designed with a modular architecture, which would allow modules such as the planner to be replaced with other standard techniques, for example POMDP planners [136]. IGP2 could also be generalised to other robotics domains in which there is interaction with other robots or humans. One future direction that could be explored is to handle goal recognition when there are objects occluded from the view of the ego vehicle but visible to a non-ego vehicle. Another avenue for future research could be to account for the irrational biases of humans [137, 71]

Chapter 5

Goal Recognition with Interpretable Trees

5.1 Introduction

To safely navigate through busy city traffic, autonomous vehicles (AVs) must be able to predict the future trajectories of other road users, and an effective method of doing this is to first recognise their goals. For example, the goal of a vehicle could be to take a certain exit at a junction, as shown in Figure 5.1. There are several desirable properties for goal recognition methods: these methods must be **fast**, as AVs must make decisions in real time and quickly react to new information; and predictions must be **accurate** to be useful for planning and navigation. It is also desirable for goal recognition methods to be **interpretable** by humans. Regulations which codify the “right to an explanation" for some types of automated decision have already been created [13], and regulators may create similar rules for AVs.

Prediction accuracy is typically measured empirically based on statistical averages, but no guarantees can be given about inferences made [138]. Autonomous driving is a safety-critical task, and it is important to have ways of validating that prediction methods will act as intended when deployed. The amount of data required to empirically validate the safety of an autonomous vehicle is enormous, on the order of billions of miles [15]. An alternative approach to safety validation is to formally **verify** models of the system to guarantee safety under all possible conditions [16, 17].

Verification and interpretation are not possible for some prediction methods, such as those that make use of deep neural networks (DNNs) [76, 77, 68, 69, 70, 71, 72, 73], due to their complexity and large number of parameters [75]. Another approach to

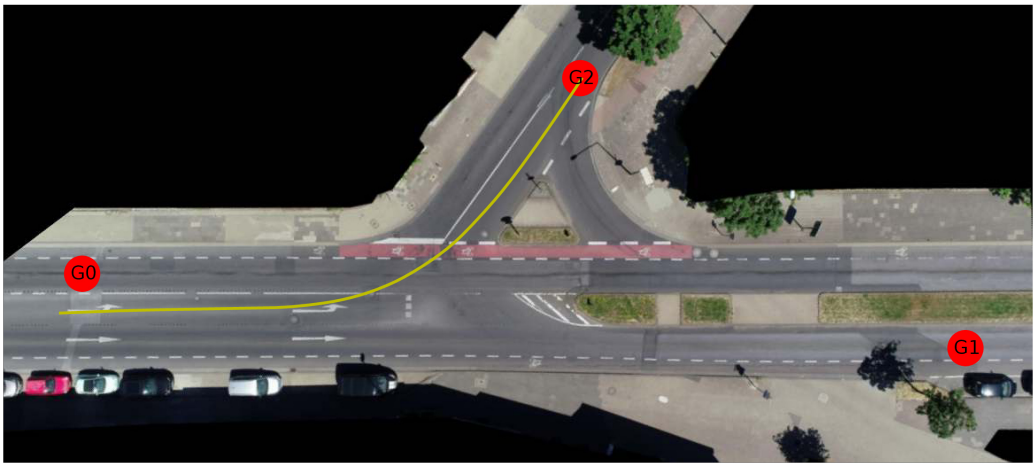


Figure 5.1: The “Heckstrasse" junction from the inD dataset [139]. Goal locations are shown by red circles. An example vehicle trajectory is shown by the yellow line.

prediction is to first perform goal recognition, and use this to inform trajectory prediction. One method of goal recognition is planning to a set of possible goal locations from the perspective of the agent for which we are performing goal recognition on [83, 56, 86, 140, 84], as described in Chapter 4. Such goal recognition methods can be used to generate accurate long term trajectory predictions which are explainable through rationality principles. However, the planning process is computationally complex, making it difficult to run in real time and intractable to verify.

There has been significant previous work on verification of autonomous driving policies [141, 15]. However, to the best of our knowledge, no existing verifiable prediction methods for AVs. Recent work has shown that decision trees can produce models which are more easily verified and interpreted than deep neural networks in domains other than prediction for autonomous vehicles. Bastani et al. [94] used decision trees to represent a reinforcement learning policy, and showed that certain properties of these decision trees can be efficiently verified using off-the-shelf satisfiability modulo theories (SMT) solvers. Liu et al. [108] used knowledge distillation to obtain an interpretable decision tree from a less interpretable deep neural network for classification. These works motivate our approach to use decision trees for an interpretable and verifiable goal recognition method.

In this case, we present *Goal Recognition with Interpretable Trees* (GRIT), a vehicle goal recognition method which satisfies the objectives of being fast, accurate, interpretable and verifiable. At the core of this method, we use decision trees which are trained from vehicle trajectory data. Decision trees are computationally efficient and highly structured, which allows GRIT to be fast at inference time and interpretable to

humans. We show how properties of GRIT can be verified automatically by mapping the learned trees into propositional logic and using an SMT solver [142]. We evaluate GRIT across four scenarios from two vehicle trajectory datasets [139, 143] and show that it achieves comparable accuracy to deep learning baselines and performs inference fast enough to run in real time. We demonstrate by example how the trained decision trees are human interpretable. We verified several properties of the models, for example verifying that the probability of a goal is above a threshold if a vehicle is in the correct lane for that goal. If verification fails, the SMT solver provides a counterexample which can teach us about the way in which the model works, facilitating inspection and debugging. To the best of our knowledge, GRIT is the first goal recognition method for autonomous vehicles which has been shown to be verifiable.

5.2 GRIT: Goal Recognition with Interpretable Trees

Our method aims to infer a probability distribution over goals for a vehicle based on past observations, using models trained from vehicle trajectory data. We use the name GRIT (Goal Recognition with Interpretable Trees) to refer to the entire training and inference process. GRIT computes a Bayesian posterior probability distribution over goals

$$P(g^i|s_{1:t},\phi) = \frac{L(s_{1:t}|g^i,\phi)P(g^i|\phi)}{\sum_{g'\in g_t^i} L(s_{1:t}|g',\phi)P(g'|\phi)} \tag{5.1}$$

and represents the likelihood $L(s_{1:t}|g^i,\phi)$ of a trajectory $s_{1:t}$ given goal g^i using decision trees learned from vehicle trajectory data prior to deployment. We assume full observability, in which case the full trajectory $s_{1:t}$ can be determined from the observations $o_{1:t}$

An overview of GRIT's inference process is shown in Figure 5.2. As input during inference, GRIT takes the past observed trajectories of local vehicles $s_{1:t}$ and static scene information ϕ. As output, GRIT gives a probability distribution over possible goals for a vehicle. The first step in the inference process is to generate a set of possible goals for the vehicle. Next, a feature vector is extracted for each goal based on the past trajectories of all observed vehicles and static scene information. Decision trees are then used to infer a likelihood for each goal, before inferring a posterior probability distribution over goals via Eq. (6.1).

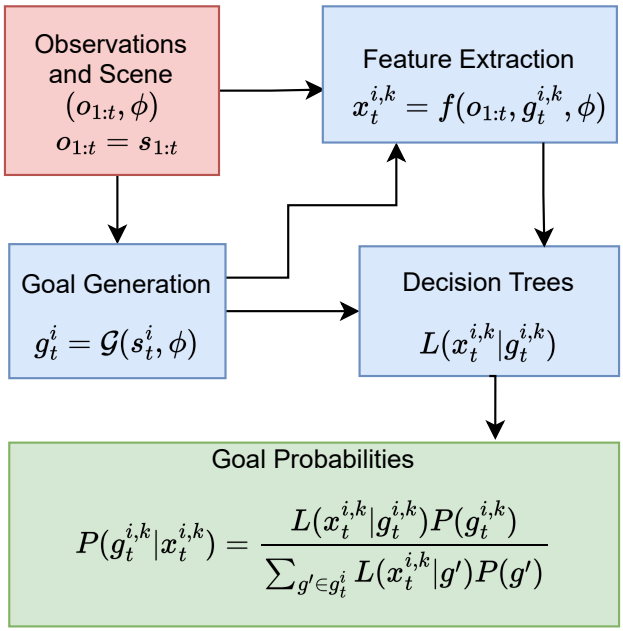


Figure 5.2: Diagram of the overall GRIT inference system.

5.2.1 Goal Generation

In order to perform goal recognition for vehicle i at time t, we first generate a set of possible goals $g_t^i = \mathcal{G}(s_t^i, \phi)$ from the vehicle's current state and static scene information. We assume a module $\mathcal{G}$ which gives us the possible goals. For example, goals could be extracted heuristically using road layout and vehicle state, as we do in our experiments.

5.2.2 Feature Extraction

For each vehicle i, for each possible goal $g_t^{i,k}$ at time t we extract a feature vector $x_t^{i,k} = f(s_{1:t}, g_t^{i,k}, \phi)$ which will be used by the decision trees. These features can have binary or scalar values. We extracted the following features for each vehicle i at each time t, which were chosen to be easily interpretable: Length of path to goal $\in \mathbb{R}_0^+$; In correct lane for goal $\in \{0, 1\}$; Current speed $\in \mathbb{R}_0^+$; Current acceleration $\in \mathbb{R}$; Angle in lane $\in [-\pi, \pi)$; Distance to vehicle in front $\in \mathbb{R}_0^+$; Speed of vehicle in front $\in \mathbb{R}_0^+$; Oncoming vehicle distance $\in \mathbb{R}_0^+$; Speed of oncoming vehicle $\in \mathbb{R}_0^+$.

5.2.3 Goal Types

Depending on a vehicle's position on the road relative to a goal location, the actions that vehicle must take to reach that goal can be quite different. For example, consider a vehicle with goal G1 in the scenario shown in Figure 5.1. If the vehicle is

approaching from the west, then it simply needs to continue straight on to reach its goal. However, if the vehicle is coming from the north, it needs to enter the T-junction and cross several lanes of traffic. We address this by considering a set of goal types, such as *straight_on* or *turn_left*. We could in principle train one tree for each goal that handles all goal types, but this would make the tree more complicated and thus less interpretable. We split the model up into separate trees for the different goal types to reduce the complexity of the model and improve interpretability. For each vehicle i at time t, each possible goal location is assigned with a goal type $\tau_t^{i,k}$ from the set $\mathcal{T} = \{straight\_on, turn\_left, turn\_right, u\_turn\}$. We assume these goal types are automatically assigned by the goal generation module.

5.2.4 Decision Trees

For each possible goal location/goal type pair, we train a decision tree which takes the feature values as input and outputs the likelihood $L(x_t^{i,k}|g_t^{i,k})$ of the features given the goal and goal type. These likelihoods are combined with priors $P(g)$ to obtain a categorical posterior distribution over goals, as shown in Equation (6.1). To obtain the output likelihood of the decision tree, we traverse the tree starting from the root based on the decision rule at each node until a leaf is reached. As shown in Figure 5.6, each edge in the tree is assigned a weight. The likelihood value at each leaf node is calculated from the product of the initial likelihood of 0.5 with the weights on each edge leading to that leaf.

5.2.5 Decision Tree Training

We train each decision tree using the CART algorithm [104]. When using CART, decision trees are expanded in an iterative manner starting from the root, greedily choosing the decision rule which maximises a certain criterion, in our case information gain. Cost complexity pruning [144] is used to regularise the trees, and to aid interpretability the depth of the trees is limited.

Each decision tree is trained using the set of sampled vehicle states from the training set for which the relevant goal g is reachable, while having the relevant goal type. These make up the dataset $D = \{(x_1, y_1), ..., (x_N, y_N)\}$, where x_j is the set of features, and y_j is the corresponding ground truth goal. A likelihood value is assigned to each node based on several sample counts: the total number of samples with goal g, $N_g = |\{j|y_j = g\}|$, the number of samples without goal g, $N_{\bar{g}} = N - N_g$, the number of samples at node

n with goal g, $N_{ng} = |\{j|x_j \in R_n \wedge \ y_j = g\}|$, and the number of samples at node n without goal g, $N_{n\bar{g}} = N_n - N_{ng}$. Each of these counts is regularised using additive (Laplace) smoothing with hyperparameter α. In many cases, there is an imbalance of samples between N_{ng} and $N_{n\bar{g}}$. To correct for this, we weight samples using the weights $w_g = N/N_g$ and $w_{\bar{g}} = N/N_{\bar{g}}$ so that the total weight given to samples with true goal g and $\bar{g}$ is equal. The likelihood assigned to node n of the tree is then given by:

$$L_n = \frac{w_g N_{ng}}{w_g N_{ng} + w_{\bar{g}} N_{n\bar{g}}} \tag{5.2}$$

5.2.6 Verification

One limitation of current prediction methods is the inability to guarantee safety through formal verification. GRIT can easily be verified due to the computational simplicity of decision tree inference, and the tree representation, which can be mapped into propositional logic. For example, we can verify that under certain conditions, certain nonsensical predictions will not be made.

There are some limitations to this approach to verification. Our approach can verify whether propositions made about the input-output mapping of GRIT will always hold true. However, our method does not force the model to obey these propositions during the training phase. In addition, we do not verify how a larger autonomous driving system would function when integrated with GRIT.

In order to perform verification, we first represented the model using propositional logic, and then verify a proposition Ψ by proving that $\neg\Psi$ is unsatisfiable. We used the Z3 SMT solver [142] to perform the verification. In the event that verification of Ψ fails, the solver provides a counterexample, which can be useful to understand why the model makes certain predictions.

The decision trees can be represented using propositional logic with equality/inequality constraints by taking the conjunction of the statements given below. A Boolean variable N_n is created for each node in the decision tree. The value of the variable is true if the node is reached, and false otherwise. The variable for root nodes N_{root} is always true. The decision rule at node n is D_n. D_n can represent a Boolean feature x_j directly, or an inequality constraint on a scalar feature $c_n > x_j$. For each non-leaf node n, the child node variables follow the constraints $N_{ntruechild} = N_n \wedge D_n$, and $N_{nfalsechild} = N_n \wedge \neg D_n$. If a leaf node is reached, then the likelihood output by the tree for goal g is the likelihood L_n at that node:

$$N_{leaf} \implies (L(x|g) = L_{leaf}) \tag{5.3}$$

$$P(g|x) = \frac{L(x|g)P(g)}{\sum_{g,\tau)'} L(x|g')P(g')} \tag{5.4}$$

Some of the feature values can differ for different goals, such as “in correct lane" and “path to goal length". However, other features such current speed and acceleration are constrained to be the same regardless of the goal.

5.3 Evaluation

We evaluated GRIT and several baselines in four scenarios from two vehicle trajectory datasets. We show that: (1) GRIT has similar or better accuracy than the baselines; (2) GRIT inference is fast enough to run in real time; (3) the GRIT inference process is interpretable by humans; (4) properties of GRIT inference can be formally verified. A video showing GRIT is available at: `https://www.five.ai/grit`.

5.3.1 Datasets

We evaluate GRIT and the baselines in the inD dataset [139] and the rounD dataset [143]. Both of these datasets consist of vehicle trajectories recorded at several different junctions and roundabouts, along with local road layout maps which are provided in the Lanelet2 [145] format.

We trained and evaluated the models on three scenarios from the inD dataset, shown in Figure 5.3. These included “Heckstrasse", a T-junction; “Bendplatz", a marked crossroad with separate lanes for exiting; and “Frankenberg", an unmarked crossroad [139]. We used one roundabout scenario, “Neuweiler" from the rounD dataset [143]. Each scenario had a number of continuous recordings, with a typical duration of 20 minutes. For each scenario in the inD dataset, we randomly selected one recording for testing, one recording for validation (used for hyperparameter selection), and used the rest of the recordings for training. Due to the larger number of recordings in the rounD dataset, two recordings were randomly selected for validation and testing. The same split was used for all tested methods.

We manually annotated each of the scenarios with goal locations, as can be seen in Figure 5.1. These included junction/roundabout exits and visible lane ends. We determined the ground truth goal of each vehicle by finding the first goal for which the trajectory passes within a defined distance (1.5m in the Heckstrasse example). Vehicle trajectories for which none of the predefined goals were reached, were discarded. To

create training data and test data for GRIT, each trajectory was first trimmed up to the point where the goal was reached. Following this 11 evenly timed samples were taken from each trajectory. Each sample contained the state history of the vehicle of interest up to that point in time, along with the state history of other vehicles from the point at which the vehicle of interest was first observed.

5.3.2 Baselines

In this section, we describe the baseline methods with which we compared GRIT. To the best of our knowledge, there are no GR methods which have published results on the inD or rounD datasets other than those presented in this thesis. To validate this, we used Google Scholar to search through all works which cite the inD or rounD datasets and mention "goal recognition" or "intent recognition".

5.3.2.1 GRIT-no-DT

As a first baseline, we have included an ablation of GRIT in which the decision trees have been removed. This amounts to generating the set of possible goals based on the current vehicle state and road layout, and then re-normalising the prior probabilities for these goals to obtain the posterior goal distribution. In some scenarios, the prior probability of some goals is much higher than others, and simply always predicting the most common goal could achieve a high accuracy. This baseline gives context to the results achieved by the other methods by acting as a floor showing what a very simple method can achieve.

5.3.2.2 IGP2

In contrast to the learning based approaches, we also included the inverse planning based goal recognition used as part of IGP2, which is described in Chapter 4. This method finds the optimal plan to each possible goal from both the vehicle's current position, and first observed position. The goal likelihood is then calculated based on the cost difference between these two plans, with a larger difference leading to lower likelihood. Similarly to GRIT, a Bayesian posterior probability distribution over goals is then calculated. The predictions made by IGP2 are highly interpretable, however the planning process used by IGP2 is computationally complex, which makes inference slow and verification infeasible.

Table 5.1: IGP2 reward weights and free parameters of macro-actions and manoeuvres used for each dataset.

Parameters	**InD dataset**	**RounD dataset**
time to goal reward weight	0	0.01
angular velocity reward weight	0	0.01
heading reward weight	1000	10
acceleration reward weight	0	0.01
give way distance	10	10
give way lane angle threshold	$\pi/6$	$\pi/6$
give way turn target threshold	1	1
manoeuvre point spacing	0.25	0.25
manoeuvre max speed	speed limit	speed limit
manoeuvre min speed	3	3
switch lane target switch length	20	10
switch lane minimum switch length	5	5

The goal recognition module of IGP2 is implemented as described in Chapter 4, using all macro-actions and manoeuvres to predict trajectories, except *Stop*. We run the velocity smoothing module with parameters $\lambda = 10$, timestep $\Delta t = 0.1$, $a_{max} = 5$ and v_{max} set to the episode speed limit. To improve convergence during velocity smoothing, we are using the objective function:

$$\min_{x_{2:n}, v_{2:n}} \quad \sum_{t=1}^{n} (v_t - \kappa(x_t))^2 + \lambda \sum_{t=1}^{n-1} (v_{t+1} - v_t)^2 \tag{5.5}$$

The reward terms are normalised between 0 and 1 according to their distributions across both datasets, with values falling beyond three standard deviations of the distribution being clipped. The reward weights and the free parameters of the macro-actions and manoeuvres are reported in table 5.1.

5.3.2.3 LSTM

As another baseline, we trained a recurrent neural network architecture based on Long Short-Term Memory (LSTM) [146] for each scenario individually to directly predict goal probabilities. The input to the LSTM is a raw state sequence $s^i_{1:t}$ of vehicle i and the target for each time step is the true goal G^i_t. Our model architecture is built from a single-layer LSTM. The hidden unit of each cell has size 64. The outputs of the LSTM at each time step is pushed through a fully connected (FC) network with one hidden layer of dimension 725 and ReLU activation. The weights of the FC layer are initialised using normally distributed Glorot initialisation [147]. We minimise the mean total loss

cross-entropy using the Adam optimiser [148] with a learning rate of 5×10^{-4}. We train for 1,000 epochs with early stopping and using a batch size of 10 trajectories. Our FC layer is regularised with dropout with $p = 0.2$. We schedule the learning rate to decrease by a factor of 0.5 if the best validation loss did not improve for 10 consecutive epochs.

5.3.2.4 GR-ESP

We implemented a goal recognition method based on trajectories sampled from the multi-agent deep generative model (called ESP) of the PRECOG system [76]. We call this baseline Goal Recognition with ESP (GR-ESP). PRECOG is a deep learning based model which was shown to make robust, goal-aware planning decisions when conditioned on goal-positions and was able to accurately estimate the likeliest future trajectory an agent could take to safely reach a goal. Given K trajectory samples drawn from ESP for a vehicle, we define the probability of a goal G as the normalised count of trajectories whose endpoints are closest to G. The generated trajectories of ESP are fixed-length, therefore we perform repeated sampling to obtain trajectories with a suitable length.

GR-ESP predicts from $t = 0$ the joint future state of length T (4 seconds) of all vehicles given a tuple $\{s_{-\tau:0}, \chi\}$. The first element is the past joint state with length τ, and $\chi \in \mathbb{R}^{w \times h \times 8}$ is composed of feature-maps that represent the road surface, road markings, and vehicles at $t = 0$. We downsample our data set to 10 fps, then discretise it to time-steps using a moving window with a step-size of 0.5 seconds (5 frames) and a length of $\tau + T$ seconds, in our experiments $\tau = 2$. Trajectories in the window that are shorter than $\tau + T$ seconds are padded using a straight-ahead constant-velocity assumption. We train GR-ESP for 40,000 steps using the original hyper-parameters and sample $K = 100$ trajectories per time-step. During testing, if no goals are reached on the first sampling, then we repeatedly generate new trajectories up to R additional times. At each repeated generation, we condition on the final τ seconds of the previously generated future trajectory. We set $R = 2$ for the rounD scenario and $R = 1$ otherwise.

5.3.3 GRIT Implementation

We manually annotated each scenario with goal locations and define the possible goals $\mathcal{G}(s_t^i, \phi)$ for vehicle i at time t as the set of goal locations reachable from s_t^i under traffic rules allowed by the local road layout ϕ. Reachability checking is performed using functionality built into the Lanelet2 library [145] which uses Dijkstra's shortest path

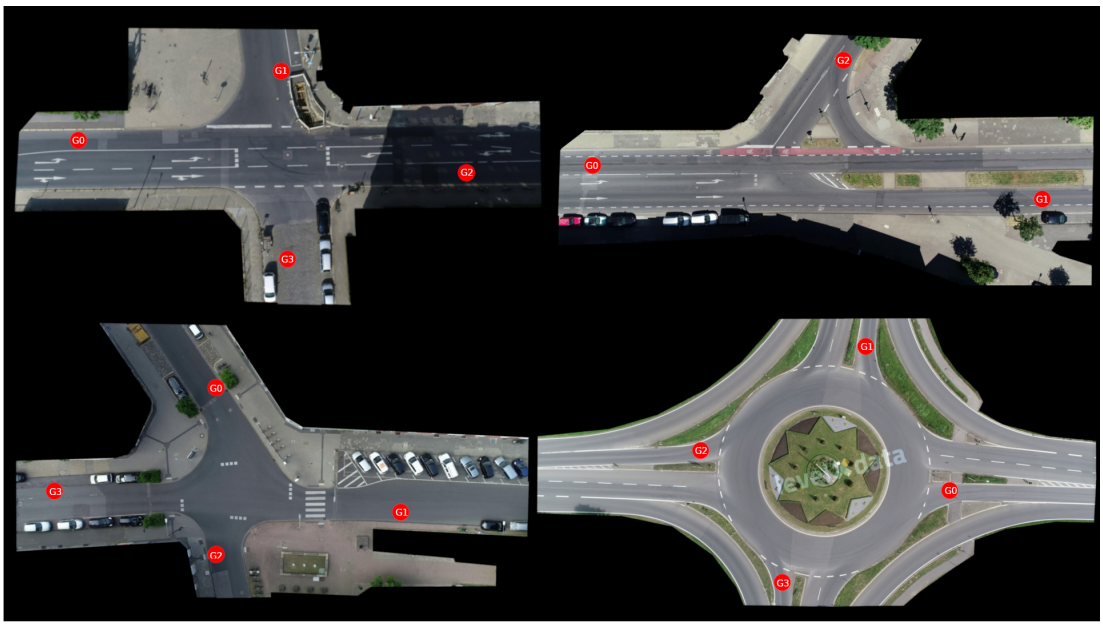


Figure 5.3: All scenarios used from the inD [139] and rounD [143] datasets. Frankenberg (top left), Heckstrasse (top right), Bendplatz (bottom left) and Neuweiler (bottom right). The red dots show goal locations.

algorithm.

The prior probabilities for each goal/goal type pair were estimated from their frequency in the training dataset, with Laplace smoothing. The maximum depth of the decision trees was limited to 7. The parameters for Laplace smoothing and cost complexity pruning [144] were chosen by grid search, with a separate set of parameters selected for each scenario.

5.3.4 Accuracy and Entropy

One evaluation metric used was **accuracy** – the fraction of test samples for which the true goal was assigned the highest probability. Another metric which was examined is **normalised entropy**, which is the entropy of the posterior goal distribution divided by the entropy of a uniform distribution. This gives a measure of the uncertainty of the model about the vehicle's goal.

The evolution of accuracy and entropy as the fraction of the trajectory observed increases is shown in Figure 5.4. Across all methods except GRIT-no-DT, accuracy increases as the fraction of the trajectory observed increases. Similarly, normalised entropy tends to decrease as more of the trajectory is observed, showing how the models become more certain about a vehicle's goal as more observations are made. The LSTM model achieved the highest accuracy overall. In the Heckstrasse and Bendplatz

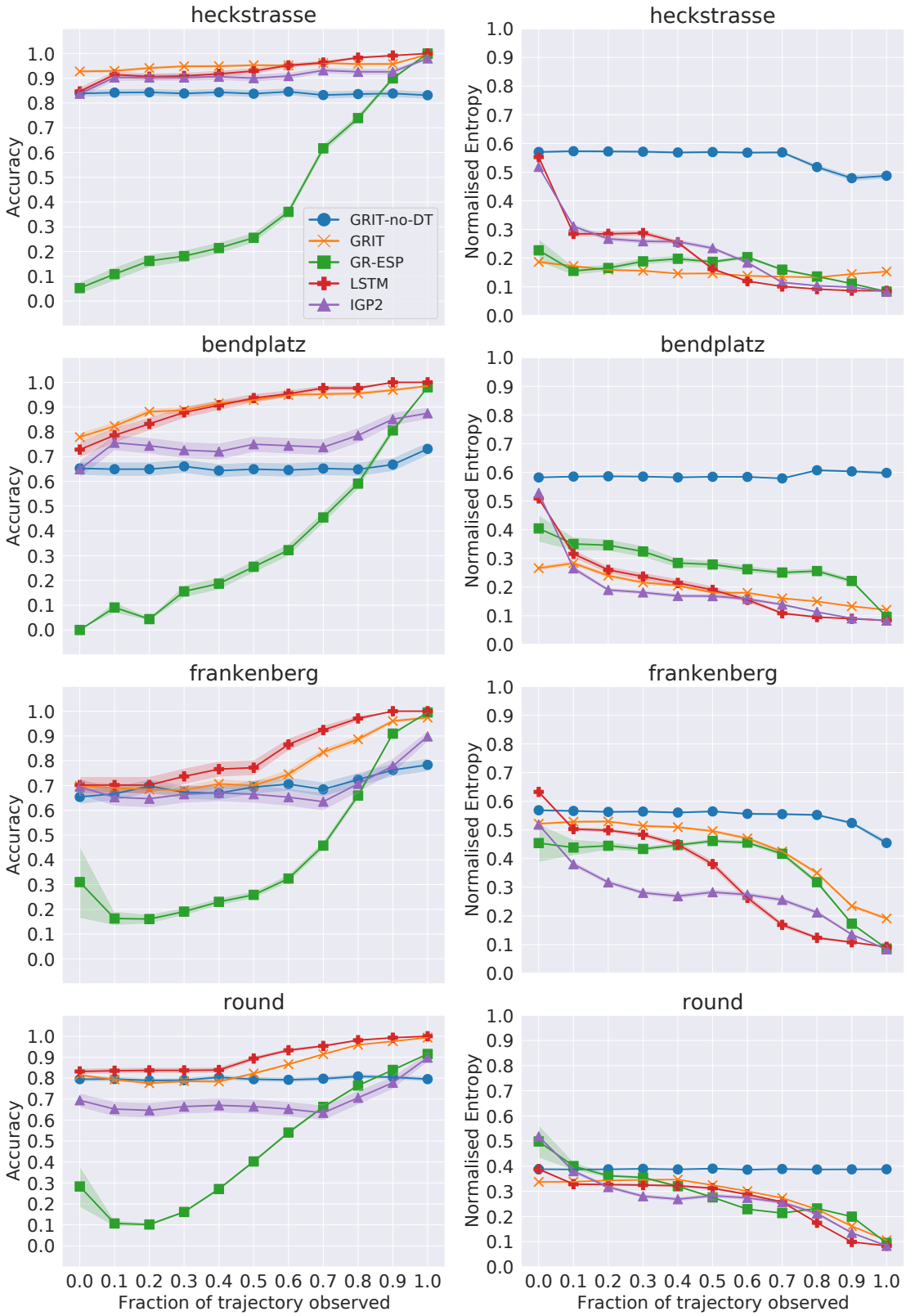


Figure 5.4: Goal recognition accuracy (left) and normalised entropy (right) for each method on each scenario. The shaded areas show standard error.

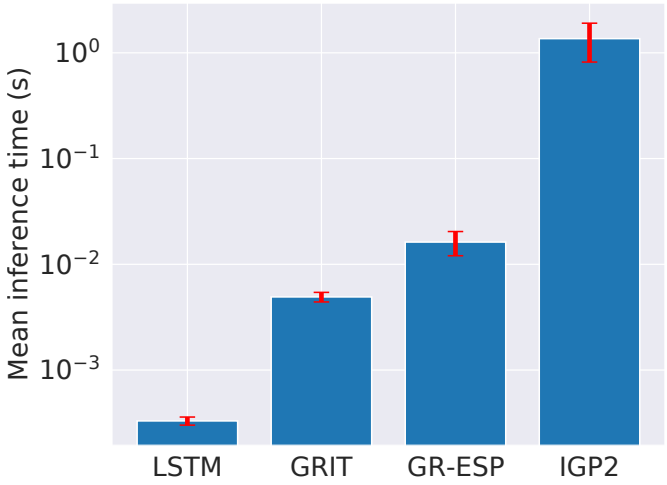


Figure 5.5: Mean inference time per vehicle in seconds on log-scale with standard error.

scenarios, GRIT achieved a similar accuracy to the LSTM, however GRIT had slightly lower accuracy than the LSTM in Frankenberg and rounD. The LSTM model could be extracting more information from the raw trajectory than is represented in the tree features used by GRIT, leading to higher accuracy than GRIT in some cases. As expected, the accuracy of GRIT-no-DT is lower than that of full GRIT and LSTM, as it does not have access to observations other than the set of reachable goals. IGP2 also achieves lower accuracy than GRIT and LSTM. One reason for this is that the inverse planning used in IGP2 sometimes fails to find a plan to the true goal, in which case this goal is given zero probability [24]. The average fraction of samples over time for which no plan was found to the true goal in each scenario was 0.013 for Heckstrasse, 0.070 for Frankenberg, 0.101 for Bendplatz, and 0.010 for rounD. Another reason for the lower accuracy could be that the goal priors used for IGP2 are less fine-grained than those used for GRIT, as IGP2 uses a prior probability for each goal, rather than each goal/goal type pair. GR-ESP achieves significantly lower accuracy than other methods, except when the fraction of trajectory observed is very close to one.

5.3.5 Inference Time

A comparison of mean inference times for each method is shown in Figure 5.5. All methods other than IGP2 are fast enough to run in real time. The fastest method is LSTM, followed by GRIT. The majority of time during GRIT inferences is taken up by the feature extraction. GR-ESP is slower than LSTM and GRIT, due to its repeated sampling process. IGP2 is by far the slowest method due to its computationally intensive planning process.

All experiments were carried out on a server with two AMD EPYC 7502 CPUs and eight Nvidia GeForce RTX 2080 Ti GPUs. The GPUs were used for the neural networks in both the LSTM and GR-ESP baselines, and all other computation was performed on the CPUs.

GRIT is highly scalable, as several aspects of the method can be parallelised. If there are multiple nearby vehicles, the goal inferences for each vehicle are independent and can be computed in parallel. In addition to this, the feature extraction and likelihood calculation for each goal can be parallelised. In our implementation of GRIT, we did not parallelise the method to maintain simplicity, but for practical applications, parallelisation could easily be implemented.

5.3.6 Interpretability

We found that the trees learned by GRIT were human-interpretable, with an average depth of 6.19. For example, take the trained decision tree shown in Figure 5.6. If the top-left leaf node with likelihood 0.291 is reached, an explanation with weights for each factor can easily be extracted: “Goal G1 *straight_on* has a likelihood of 0.291 because the vehicle is in the correct lane (weight 1.97) and the vehicle’s angle in lane is greater than 0.05 radians to the left (weight 0.30)". Examining the scenario in Figure 5.1 shows that this interpretation makes sense. Although a vehicle which reached the 0.291 likelihood leaf node in Figure 5.6 is in the correct lane to go straight on, the fact that it is angled to the left suggests that it will turn left rather than going straight on, leading to a low likelihood for the *straight_on* goal.

5.3.7 Verification

Using the method described in Section 5.2.6, we were able to verify several properties of our learned trees. In failed verification attempts, the method provided a counter-example to explain why our original intuition was incorrect, and this knowledge could be useful for improving the models. Here we give examples for the Heckstrasse scenario, shown in Figure 5.1. We were also able to prove similar properties in other scenarios. Such verification is currently not possible with the deep learning models – we never quite know what these methods will predict. The verification process was relatively fast, taking an average of 65.2 milliseconds to verify a proposition for each tree.

For formal definitions of each proposition, we use the following notation. The entire set of features are represented by x_t^g, where g is an identifier for a goal, and t is a certain

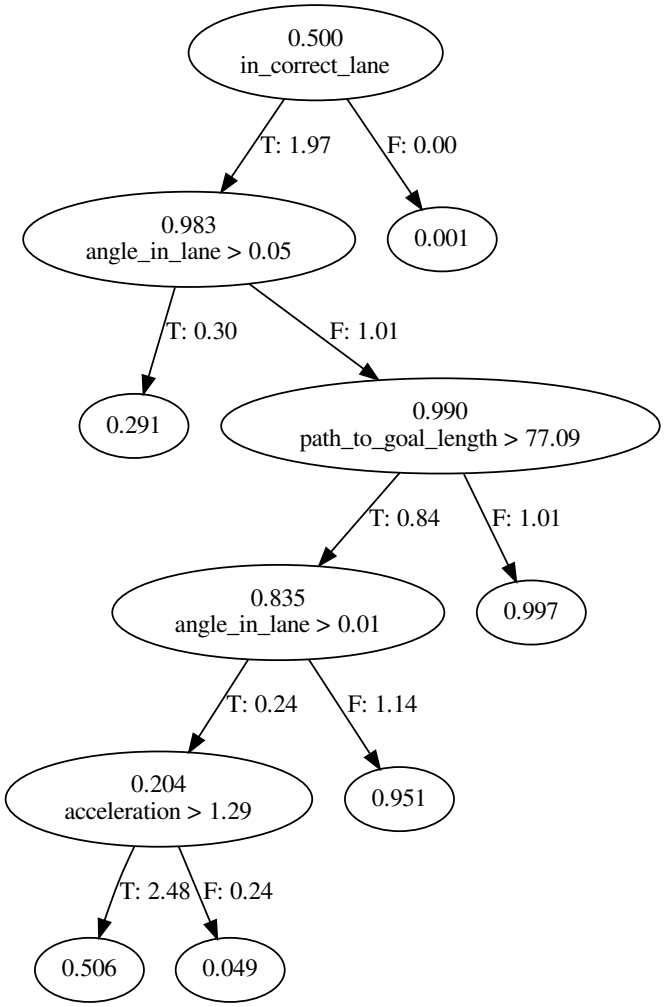


Figure 5.6: Trained decision tree for the Heckstrasse scenario goal G1, goal type *straight_on*. Multiplicative weights are assigned to each True (T) and False (F) edge. The first value shown in each node is the cumulative product of the initial likelihood 0.5 and the weights of edges traversed so far, representing the likelihood $L(x|g)$. Beneath this, the decision rule for each node is shown.

timestep or instance of the scene. Individual feature values are represented by $x_t^{g,f}$, where $f \in \mathcal{F}$ is an identifier for a feature. The identifiers used are: path to goal length: $path$, in correct lane: l, speed: spd, vehicle in front speed: fs, vehicle in front distance: fd, oncoming vehicle distance: ond.

5.3.7.1 Predict goal corresponding to lane

As can be seen from Figure 5.1, if a vehicle is coming from the west, there are two possible lanes: one marked as the lane for exiting left, and one marked as the lane for continuing straight on. One reasonable expectation of a goal recognition model would be that if a vehicle is in the correct lane for a goal, then that goal will be assigned the

highest probability. We successfully verified the proposition “If the vehicle is in the correct lane for G2 (*turn_left*), then G2 is assigned the highest probability", which was represented as follows:

$$x_t^{G1,l} \wedge \neg x_t^{G2,l} \implies P(G1|x_t^{G1}) > P(G2|x_t^{G2})$$

Verification failed for the equivalent proposition for G1. However, the solver provided the feature values shown in Table 5.2 as a counterexample, which can still teach us about the way in which the model works. Despite the fact that the vehicle is in the lane for G1 (*straight_on*), the vehicle is angled to the left in its lane, while going at a slow speed and decelerating, and is still far from the junction entry (path_to_goal_length 78.09 meters). This together is reasonable evidence towards G2 being the true goal, and in such a case assigning the higher probability to G2 is correct. We also attempted to verify a relaxed version of this proposition, in which we verify that the probability of G1 is always above a certain lower bound. We successfully verified that if the vehicle is in the correct lane for G1, then the probability assigned to G1 is always greater than 0.2.

5.3.7.2 Goal distribution entropy

As a vehicle travels closer towards its goal, a reasonable expectation is that we should become more certain about what its goal is – that is, the entropy of the distribution over goals should decrease, or at least stay constant. If there are just two possible goals, then it is equivalent to show that if one goal has higher probability than the other, then the probability of the most probable goal will not decrease as the length of the path to the goal decreases. We represented the proposition as follows:

$$\begin{aligned}
&\bigwedge_{g\in\{G1,G2\}} ((x_{t1}^{g,path} > x_{t2}^{g,path}) \bigwedge_{\substack{f\in\mathcal{F},\\ f\neq path}} (x_{t1}^{g,f} = x_{t2}^{g,f})) \\
&\implies ((P(G1|x_{t1}^{G1}) < P(G2|x_{t1}^{G2}) \implies P(G2|x_{t2}^{G2}) \\
&\geq P(G2|x_{t1}^{G2})) \wedge (P(G1|x_{t1}^{G1}) > P(G2|x_{t1}^{G2}) \\
&\implies P(G1|x_{t2}^{G1}) \geq P(G1|x_{t1}^{G1})))
\end{aligned}$$

We attempted to verify this for the situation where a vehicle is approaching from the east in the Heckstrasse scenario. However, in this case, the verification failed. If a vehicle is in a state such as that shown in Table 5.2, the model predicts with high certainty that a vehicle is going to turn at the junction. The prediction is biased towards

Table 5.2: Generated counterexample to the proposition: “If the vehicle is in the correct lane for G1 (*straight_on*), then G1 is assigned the highest probability". Despite the fact that the vehicle is in the lane for G1 (*straight_on*), there is some evidence of G2 being the true goal: the vehicle is angled to the left in its lane, while going at a slow speed and decelerating, and is still far from the junction entry.

Features	**Goal 1**	**Goal 2**
path_to_goal_length	78.09	57.09
in_correct_lane	True	False
speed	0	0
acceleration	-1	-1
angle_in_lane	0.03125	0.03125
vehicle_in_front_dist	32.87	32.87
vehicle_in_front_speed	0	0
oncoming_vehicle_dist	None	None
goal likelihood	0.04874	0.9242
goal probability	0.2014	0.7985

the goal G2, *turn_left* due to the vehicle’s angle in the lane, although the vehicle is still quite distant from the junction entry. However, if the vehicle continues further along the road and has still not switched lane, the uncertainty over goals can actually increase. In this situation it makes sense for uncertainty to increase, because the vehicle took actions that were irrational for the goal it originally started moving towards, showing that our original intuition was incorrect.

5.3.7.3 Verification across all scenarios

The verification cases mentioned above apply to specific situations in a scenario, however it is also possible to verify some propositions more broadly across all scenarios. One such proposition is that changing a single input feature while leaving all other features unchanged will have a certain effect on the goal likelihood. More specifically, we ran verification for the proposition “If a vehicle is in the correct lane for a goal then that goal should have the same or higher likelihood than if the vehicle is not in the correct lane, if all other features remain unchanged". This proposition was represented as:

$$x_{t1}^{g,l} \wedge \neg x_{t2}^{g,l} \bigwedge_{\substack{f \in \mathcal{F}, \\ f \neq lane}} x_{t1}^{g,f} = x_{t2}^{g,f} \implies L(x_{t1}^{g}|g) \geq L(x_{t2}^{g}|g)$$

In total there were 47 goal/goal type pair across all scenarios, each having a separate decision tree learned by GRIT. Verification was successful for all but 4 of these

goal/goal type pairs. Upon inspection of the counterexamples generated in those cases, the models were still giving reasonable likelihoods given the features. For example, in one counterexample a vehicle is in the incorrect lane to turn, but is travelling at a low speed. In such a situation the vehicle could have slowed down in order to turn, so assigning a high likelihood to turning makes sense.

5.3.7.4 Stopping for oncoming vehicles

It is also possible to verify predictions made by the model in more complicated situations, such as stopping for oncoming vehicles. Verification was performed across all scenarios for the proposition "If a vehicle (V1) has stopped, and there is no stopped vehicle in front of V1, and there is an oncoming vehicle (V2) in a lane which V1 must cross to reach certain goal, then that goal will have the same or higher likelihood for V1 than if there was no oncoming vehicle V2, all other features being unchanged". This proposition was represented as:

$$x_{t1}^{g,spd} < 1 \wedge x_{t1}^{g,fs} = 20 \wedge x_{t1}^{g,fd} = 100 \wedge x_{t1}^{g,ond} = 100 \wedge$$
$$x_{t2}^{g,ond} = 20 \bigwedge_{\substack{f \in \mathcal{F}, \\ f \neq ond}} x_{t1}^{g,f} = x_{t2}^{g,f} \implies L(x_{t2}^{g}|g) \geq L(x_{t1}^{g}|g)$$

For this proposition, verification was successful for 44/47 of the total goal/goal type pairs.

5.4 Limitations

It is important to acknowledge the limitations of our method and evaluation. We evaluated GRIT across four different static scenarios. For each scenario, we trained separate DTs, which could only applied to the same scenario on which they were trained. For this reason, GRIT cannot be applied to scenarios for which we do not already have training data.

Another limitation of GRIT is that it assumes full observability. However, in real-world driving, there is often information hidden due to occlusions. This could cause some of the features used by GRIT to have missing values, and in such cases GRIT would fail to make inferences. In addition to this, GRIT also assumes that there is no noisy in perceptions, when in reality perception of the surrounding scene may be unreliable.

One limitation of our evaluation, is that we only evaluated GRIT across four different scenarios, with possible goal locations manually labelled. Although GRIT achieved good performance across these scenarios, the four scenarios do not capture the full range of situations that may be encountered in real-world driving.

5.5 Conclusions

We presented GRIT, a goal recognition method for autonomous vehicles which makes use of decision trees trained from vehicle trajectories. We have shown empirically in four scenarios from two vehicle trajectory datasets that GRIT achieves high goal recognition accuracy and fast inference times, and that the learned tree models are both interpretable and verifiable. To the best of our knowledge, GRIT is the first verifiable goal recognition method for autonomous vehicles.

In this work, we trained and tested GRIT on several specific "fixed-frame" scenarios. Future work could extend GRIT for open-world driving, by training one decision tree for each goal type across scenarios and dynamically generating possible goal locations. Recent work [108] has shown that decision trees trained by distilling knowledge from deep neural networks can achieve higher accuracy than those trained from scratch. To further improve the accuracy of GRIT, future work could investigate knowledge distillation from deep neural networks to decision trees. Another extension of GRIT could be to handle occlusion, as is explored in Chapter 6.

Chapter 6

Goal Recognition under Occlusion

6.1 Introduction

In order to navigate through complex urban environments, autonomous vehicles (AV) must have the ability to predict the future trajectories of other road users. One method of doing this is to first perform goal recognition (GR) to infer the goals of other vehicles, such as taking certain junction exits or roundabout exits. If the goals of road users are known, this can facilitate prediction of their trajectories over longer horizons [76, 149, 121]. For example, a planner can be used to plan multiple possible trajectories to an inferred goal, which can then be used for trajectory prediction, as shown in Chapter 4.

GR is a difficult task, as there are many criteria that GR methods must fulfill to be applied successfully to autonomous driving. GR methods must be able to handle **missing information** due to occlusions. It is also important that GR methods have the ability to **generalise** across many scenarios. Inferences made by GR methods must be **accurate** in order to be useful for planning, and they must be **fast** so that they can run in real time. GR methods should ideally be **interpretable**, which can improve user trust and add debuggability to the method. As autonomous driving is a safety-critical domain, it is also important that the inference process is **verifiable**. This can be achieved by formally proving that certain statements made about the method will hold true under all possible conditions [16, 17, 115]. Existing GR methods for AVs fail to satisfy all the requirements of being fast, accurate, interpretable, verifiable, generalisable and able to handle occlusions.

As described in Chapter 5, GRIT makes use of decision trees trained on vehicle trajectory data to perform GR for AVs. GRIT is shown to be fast, and reasonably

accurate when compared with deep learning and inverse planning based methods. The inference process of GRIT is also highly interpretable due to the shallow depth of the trees and interpretable input features used. Properties of the trained GRIT models can also be formally verified due to the simplicity of DT inference. However, GRIT makes the assumption that all vehicles in the local area are fully observable when, in reality, some of the DT input features can have missing values due to occlusions. In addition to this, the GRIT models are trained specifically for a given fixed scenario and do not readily generalise across different scenarios, limiting its applicability in real-world driving.

We present a novel GR method named *Goal Recognition with Interpretable Trees under Occlusion* (OGRIT). OGRIT uses decision trees (DTs) trained with vehicle trajectory data in order to infer the likelihoods of goals. To handle missing data, we introduce indicator features which show when certain DT input features are missing. These indicator features are added to the DT using a novel training algorithm which uses a one level look-ahead. One DT is trained for each goal type, such as exiting left at a junction, and the same DT can be used when a goal of that type is encountered across many scenarios. We evaluate OGRIT across four scenarios from two different vehicle trajectory datasets, inD [139] and rounD [143]. We show that OGRIT can handle occlusions and generalise across multiple scenarios, while still being fast, accurate, and interpretable. We also formally verify that certain propositions about predictions made by OGRIT will always hold true. For example, we verify that if a certain input feature is missing due to occlusions, the inferred goal distribution entropy will be greater than or equal than the entropy if the feature is not missing, all other features being equal.

As the inD and rounD datasets are not already annotated with occluded regions, we developed a tool to extract occluded regions in 2D trajectory data. We release the tool and the extracted occlusion datasets, which we name inDO and rounDO. Along with the original inD and rounD datasets, inDO and rounDO can serve as a benchmark for AV systems designed to handle occlusions.

In summary, our main contributions are:

- The inDO and rounDO datasets[1] of occluded regions, which act as annotations for two existing vehicle trajectory datasets.
- OGRIT[2], a GR method that handles occlusions and is generalisable, fast, accurate,

[1] **inDO and rounDO datasets:** `https://datashare.ed.ac.uk/handle/10283/4480`
[2] **OGRIT code:** `https://github.com/uoe-agents/OGRIT`

interpretable and verifiable.

- An evaluation of OGRIT using the inDO and rounDO datasets.

6.2 OGRIT: Goal Recognition with Interpretable Trees under Occlusion

Our method, OGRIT, can infer a probability distribution over possible goals for a vehicle in a partially observable environment with data missing due to occlusions. As shown in Eq. (6.1), OGRIT computes the Bayesian posterior probability of each goal $P(g_t^{i,k}|o_{1:t},\phi)$, given a likelihood $L(o_{1:t}|g_t^{i,k},\phi)$ and prior probability $P(g_t^{i,k}|\phi)$ for each goal:

$$P(g_t^{i,k}|o_{1:t},\phi) = \frac{L(o_{1:t}|g_t^{i,k},\phi)P(g_t^{i,k}|\phi)}{\sum_{g' \in g_t^{i,k}} L(o_{1:t}|g',\phi)P(g'|\phi)} \tag{6.1}$$

Similarly to GRIT, OGRIT computes the likelihoods $L(o_{1:t}|g_t^{i,k},\phi)$ via decision trees trained from vehicle trajectory data. However, the decision trees used for OGRIT use a novel training algorithm described in Section 6.2.6 which allows them to handle missing features, while still being highly efficient, interpretable and verifiable.

An overview of the inference process can be seen in Fig. 6.2. As input, OGRIT takes the static scene information ϕ and the observation history $o_{1:t}$. As output, OGRIT gives the posterior distribution over goals $P(g_t^{i,k}|o_{1:t},\phi)$ for vehicle i. First, a set of possible goals is generated based on the current state of the vehicle and the road layout. Next, a set of base features is extracted for each goal (Section 6.2.2), for example, whether i is in the correct lane for $g_t^{i,k}$. A set of occluded regions is detected, which represent the sections of the roads in the local area which are occluded from the ego vehicle's point of view due to objects such as buildings and other vehicles. Using the observations along with the occluded regions, a set of indicator features is extracted. Each indicator feature indicates whether certain base features have missing values due to occlusions. The base features and indicator features are concatenated together to obtain the full set of features. These features are given as input to the associated decision tree for each goal, which outputs the likelihood for that goal. Using the goal likelihoods and prior probabilities, the Bayesian posterior probability for each goal is then computed using Eq. (6.1). The following subsections will detail each of these components.

6.2.1 Goal Generation

For each observable vehicle i at time t, we generate a set of possible goals G_t^i using a goal generator function $\mathcal{G}(s_t^i, \phi)$. The possible goals are generated by starting from the current lane of vehicle i and performing a breadth first search across the directed graph of lane connections until the nearest junction exits, roundabout exits, or visible lane ends are found, and added to the set of possible goals. We assume that the ego vehicle has access to HD road maps and so can generate goals in occluded locations. Each goal $g_t^{i,k} \in G_t^i$ is assigned a goal type $\tau_t^{i,k} \in \mathcal{T}$, and in this work we use $\mathcal{T} = \{straight\text{-}on, cross\text{-}road, exit\text{-}left, enter\text{-}left, exit\text{-}right, enter\text{-}right, exit\text{-}roundabout\}$. We use *enter* to signify entering a higher-priority road and *exit* to signify leaving a higher-priority road. For each goal type, we train one decision tree. Training DTs in this manner allows OGRIT to make inferences for previously unseen scenarios, as each decision tree for a certain goal type can be reused when new goals of that type are encountered.

6.2.2 Feature Extraction

At each point in time t, for each possible goal $g_t^{i,k}$ of each observable vehicle i, a set of interpretable features is extracted. We refer to these as the base features, represented by $w_t^{i,k} = f(o_{1:t}, g_t^{i,k}, \phi)$. These features can have binary or scalar values, and contain information extracted from the observation history of vehicle i and other vehicles nearby, and the static scene information. The base features we use include the following, which refer to vehicle i: *speed; acceleration; angle-in-lane; heading-change-1s; incorrect lane; path-to-goal-length; junction-heading-change; roundabout-exit-number; distance-to-vehicle-in-front; speed-of-vehicle-in-front; distance-from-oncoming-vehicle; speed-of-oncoming-vehicle*. Some of the base features may have missing values due to occlusions, and we represent the set of potentially missing features with $\mathcal{M}$. We use $\mathcal{B}$ to represent the set of features that will never have missing values, and so the set of base features is $\mathcal{M} \cup \mathcal{B}$. For example, if vehicle i is observable, then *angle-in-lane* should never have a missing value, but the vehicle in front of vehicle i may be occluded, causing *speed-of-vehicle-in-front* to have a missing value.

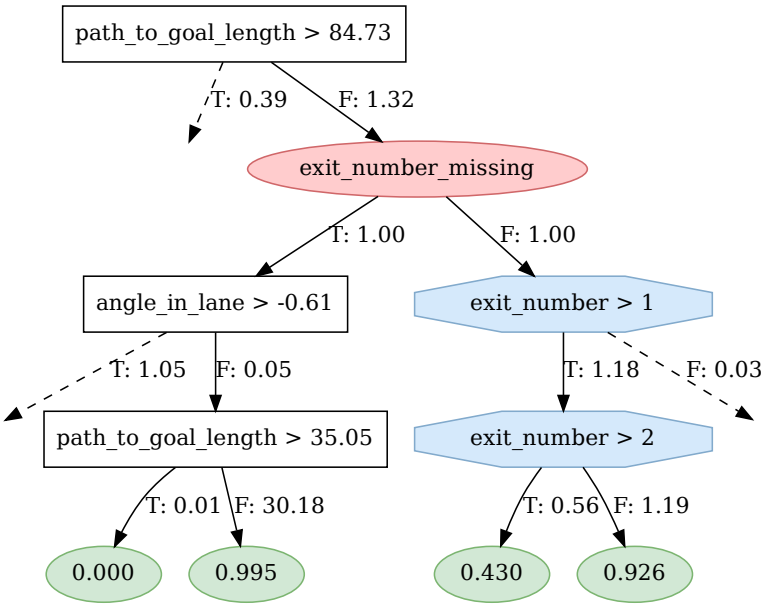


Figure 6.1: An illustrative OGRIT tree for the *exit-roundabout* goal type. Dashed edges correspond to truncated subtrees. The red oval node represents an indicator feature. The blue octagonal nodes represent potentially missing features (see Section 6.2.4). The goal likelihood at each leaf node is calculated using the multiplicative weights at each edge and an initial likelihood of 0.5.

6.2.3 Occlusion Detection

At each time t, we extract the set of regions $\mathcal{H}_t$ which are occluded from the point of view of the ego vehicle using the occlusion detector function $h(o_t, \phi)$. For occlusion detection, we use a two-dimensional top-down representation of the scene, with obstacles such as cars and buildings represented by polygons, as shown in Fig. 6.3. For each obstacle u, we find a pair of line segments l_1 and l_2 from the centre of the ego vehicle to vertices v_1 and v_2 of u, such that l_1 and l_2 yield the greatest angle between them. We then extend l_1 and l_2 so that their total length from the ego vehicle is 100 meters, obtaining new endpoints v_3 and v_4. We declare the area confined by v_1, v_2, v_3, v_4 to be occluded by u. All areas more than 100 meters from the ego vehicle are considered occluded. Once we have the occlusions for all obstacles, we find the intersection of each lane and those occlusions. We consider a vehicle to be occluded only if its entire boundary is inside an occluded region.

6.2.4 Handling Missing Features

The set of potentially missing features $\mathcal{M}$ comprises of: *roundabout-exit-number; speed; acceleration; heading-change-1s; distance-to-vehicle-in-front; speed-of-vehicle-*

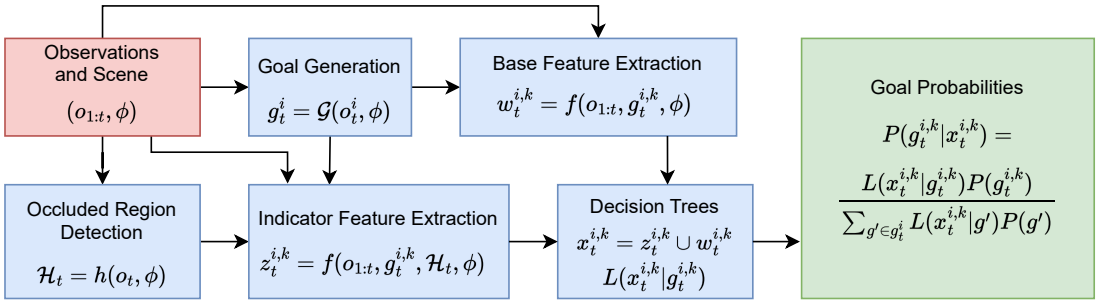


Figure 6.2: Diagram of the OGRIT inference system. Red represents input and green represents output.

in-front; distance-from-oncoming-vehicle; speed-of-oncoming-vehicle. We also use a set of indicator features $\mathcal{C}$ which indicate which features are currently missing. The indicator features are binary features which are true if the corresponding base features are missing. The function $ind : \mathcal{M} \mapsto \mathcal{C}$ gives the indicator feature for a potentially missing base feature. At each point in time t, for each observable vehicle i and possible goal $g_t^{i,k}$, the set of indicator feature values $z_t^{i,k} = f(o_{1:t}, g_t^{i,k}, \mathcal{H}_t, \phi)$ is extracted. The set of all features used for training and inference is given by $\mathcal{L} = \mathcal{B} \cup \mathcal{M} \cup \mathcal{C}$. These features are used in learned DTs as shown in Fig. 6.1

6.2.5 Decision Trees

For each goal type $\tau \in \mathcal{T}$, we train a single DT to be used across multiple scenarios. Each decision tree takes the set of features $x_t^{i,k} = z_t^{i,k} \cup w_t^{i,k}$ as input, and outputs a goal likelihood $L(x_t^{i,k}|g_t^{i,k})$. These likelihood values are combined with prior probabilities for each goal to obtain posterior goal probabilities as shown in Eq. (6.1). For simplicity, we use uniform prior probabilities. A weight is assigned to each edge in the decision tree, as shown in Fig. 6.1. The likelihood output at each leaf node of the decision tree is calculated by starting with an initial likelihood of 0.5 at the root node, and then taking the product of this with weights of the edges traversed. Each node n in a decision tree contains a condition which is an inequality on a scalar feature $x_l > c_n$, or the value of a binary feature, equivalent to $x_l > 0.5$. In order to handle missing feature values, potentially missing features $l \in \mathcal{M}$ are only included in decision nodes if the indicator feature $x_{ind(l)}$ is known to be false.

6.2.6 Decision Tree Training

Similar to DT training methods such as CART [104], ID3 [102] and C4.5 [150], the DTs used by OGRIT are trained in a top down manner, starting from the root node and iteratively expanding the tree by adding more decision nodes. However, the OGRIT DT training algorithm introduces some novel elements in order to handle features with missing values. The three main novelties are: 1) Adding indicator features which indicate that base features are missing; 2) Look-ahead by one level when considering indicator features during training; 3) Add a term to the cost function when looking ahead to penalise the additional complexity.

Pseudocode for DT training is shown in Algorithm 3. Decision nodes are added recursively as shown in Algorithm 4 until a termination condition is met, including reaching the maximum depth, the number of samples reaching that node fall below a threshold, or reaching an impurity (entropy) of zero. We use $impuritydecrease(c,l,D_n)$ to refer to the decrease in impurity when adding the decision rule $x_l > c$ to node n of the decision tree, where D_n is the set of training samples reaching node n. We prune the tree using cost complexity pruning (CCP) [104]. Decision nodes are iteratively pruned to minimise the cost function shown in Eq. (6.2), where q_n is the impurity at node n, T is the set of leaf nodes, and λ is a parameter penalising the complexity of the tree:

$$C(T) = \sum_{n \in T} q_n + \lambda |T| \tag{6.2}$$

To ensure that no nodes are reached where the relevant feature has a missing value, we only allow decision rules relating to potentially missing features to be added where the relevant indicator feature is found to be false in an ancestor node. When using DT training algorithms such as CART, nodes are expanded one at a time and the decision rule which achieves the maximum impurity decrease is chosen, without taking subsequent child decision nodes into account. However, if we naively tried to add decision rules in this way with indicator features and potentially missing features, it may give unsatisfactory results. There may be cases where adding a potentially missing feature leads to a large impurity decrease, but adding the corresponding indicator feature does not lead to any impurity decrease. As the indicator feature must be added before the potentially missing feature can be added, in such a case the potentially missing feature would never be added, despite its effectiveness. To overcome this problem we look-ahead by one level when considering indicator features and missing features, rather than using the greedy approach of only considering one decision node at a time. We

consider a parent which uses the indicator feature and a child which uses the potentially missing feature. As this operation considers adding two nodes to the decision tree rather the usual one node, there should be an additional complexity penalty added to the cost function relative to when we consider adding only one decision node. The same λ complexity penalty used during CCP can be used as a penalty for adding two decision nodes at once, as shown in line 11 of Algorithm 4.

Algorithm 3 Train Decision Tree

Input: dataset D
Returns: decision tree T

1: True indicator features $\mathcal{C}_t \leftarrow \emptyset$
2: False indicator features $\mathcal{C}_f \leftarrow \emptyset$
3: $T \leftarrow$ decision tree with root node n
4: $recursivelygrowtree(n, D, \mathcal{C}_t, \mathcal{C}_f)$
5: $T \leftarrow prune(T, \lambda)$
6: Return T

Algorithm 4 Recursively Grow Tree

Input: leaf node n on tree T, dataset D_n, true indicator features $\mathcal{C}_t$, false indicator features $\mathcal{C}_f$

1: **if** $\neg terminate(D_n)$ **then**
2: $\quad \Delta q_n \leftarrow 0$
3: $\quad$ **for all** $l | l \in \mathcal{L} \setminus \mathcal{M} \vee ind(l) \in \mathcal{C}_f$ **do**
4: $\quad\quad c_n^l \leftarrow \arg\max_c(impuritydecrease(c, l, D_n))$
5: $\quad\quad \Delta q \leftarrow -impuritydecrease(c_n^l, l, D_n)$
6: $\quad\quad$ **if** $\Delta q < \Delta q_n$ **then**
7: $\quad\quad\quad c_n, q_n, l_n \leftarrow c_n^l, q, l$
8: $\quad$ **for all** $l | l \in \mathcal{M} \wedge ind(l) \notin \mathcal{C}_t \cup \mathcal{C}_f$ **do**
9: $\quad\quad D_{nf} \leftarrow \{(x_j, y_j) | \neg x_{j,ind(l)} \wedge (x_j, y_j) \in D_n\}$
10: $\quad\quad c_{nf} \leftarrow \arg\max_c(impuritydecrease(c, l, D_{nf}))$
11: $\quad\quad \Delta q \leftarrow -(impuritydecrease(ind(l), D_n) + impuritydecrease(c_n^l, l, D_{nf}) - \lambda)$
12: $\quad\quad$ **if** $\Delta q < \Delta q_n$ **then**
13: $\quad\quad\quad c_n, q_n, l_n \leftarrow 0.5, q, ind(l)$
14: $\quad$ **if** $\Delta q_n < 0$ **then**
15: $\quad\quad D_{nt} \leftarrow \{(x_j, y_j) | x_{j,l_n} > c_n \wedge (x_j, y_j) \in D_n\}$
16: $\quad\quad D_{nf} \leftarrow D_n \setminus D_{nt}$
17: $\quad\quad$ **if** $l_n \in \mathcal{C}$ **then**
18: $\quad\quad\quad \mathcal{C}_t' \leftarrow \mathcal{C}_t \cup \{l_n\}$
19: $\quad\quad\quad \mathcal{C}_f' \leftarrow \mathcal{C}_f \cup \{l_n\}$
20: $\quad\quad$ **else**
21: $\quad\quad\quad \mathcal{C}_t' \leftarrow \mathcal{C}_t$
22: $\quad\quad\quad \mathcal{C}_f' \leftarrow \mathcal{C}_f$
23: $\quad\quad recursivelygrowtree(truechild(n), D_{nt}, \mathcal{C}_t', \mathcal{C}_f)$
24: $\quad\quad recursivelygrowtree(falsechild(n), D_{nf}, \mathcal{C}_t, \mathcal{C}_f')$

We train one DT for each goal type, and each decision tree is trained using a set of samples for which that goal type is a possible goal. These make up the dataset

$D = \{(x_1, y_1), ..., (x_N, y_N)\}$, where x_j is the set of features input to the DT, and y_j is a Boolean value that indicates whether the possible goal is the true goal for the vehicle. $x_{j,l}$ represents the value of feature $l \in \mathcal{L}$ for sample j. We use $D_n \subseteq D$ to represent the set of samples which reach node n of the DT. The likelihood value assigned to each node is calculated using several sample counts, each of which is regularised using Laplace smoothing to obtain pseudo-counts. These include the number of samples for which the possible goal is the true goal, $N_G = |\{j|y_j\}|$, the number of samples which reach node n and the possible goal is the true goal, $N_{nG} = |\{j|(x_j, y_j) \in D_n \wedge\ y_j\}|$, and the number of samples at node n where the possible goal is not the true goal, $N_{n\bar{G}} = |D_n| - N_{nG}$. In some cases there may be a large imbalance between the number of samples where the possible goal is and is not the true goal. To compensate this, we weight the samples where the possible goal is the true goal by $w_G = N/N_G$, and weight the samples where the possible goal is not the true goal by $w_{\bar{G}} = N/N_{\bar{G}}$. The likelihood value at each node is then calculated as shown in Eq. (6.3):

$$L_n = \frac{w_G N_{nG}}{w_G N_{nG} + w_{\bar{G}} N_{n\bar{G}}} \tag{6.3}$$

6.3 Implementation Details

6.3.1 Indicator Feature Implementation

For a target vehicle i on which we are performing goal recognition, the *exit_number* feature is missing if in the first frame in which i is visible to the ego vehicle, vehicle i is already in the roundabout or it is occluded w.r.t. the ego when it enters the roundabout.

The *distance-to-vehicle-in-front* and *speed-of-vehicle-in-front* features are missing if the vehicle in front of i could potentially be occluded w.r.t. the ego vehicle. If there is a visible vehicle in front of i and no occluded areas in between them, then the feature is not missing. It is also not missing if there are no occlusions within 30 meters in front of i, limited to occlusions in the lanes that i would take if its goal was the $g_t^{i,k}$ under consideration. Instead, the feature is considered missing if there is no visible vehicle in front, but there is an occluded area within 30 meters from i that is large enough to contain a hidden vehicle, or if there is a vehicle in front, but there is also an occluded area in between then two vehicles.

The *distance-from-oncoming-vehicle* and *speed-of-oncoming-vehicle* features are missing if there could be an oncoming vehicle hidden due the occlusions w.r.t the ego

vehicle. Specifically, we find the points on the different lanes in the junction that the target vehicle i will cross (assuming its goal is $g_t^{i,k}$), and check if there could be an occluded area that is closer to each of these points than any other visible vehicle in these lanes. If that's the case, then there could be a hidden oncoming vehicle, that is closer to the path i will take than any of those vehicles we can see, which could reveal useful information about the target vehicle's goal.

The *speed*, *acceleration*, and *heading-change-1s* features are missing if the target vehicle i was occluded w.r.t. the ego vehicle in the previous second. To accurately estimate the speed, acceleration, and heading change of a vehicle, its recent motion over time must be observed, and so these features have missing values if the i was recently occluded.

6.3.2 Decision Tree Training

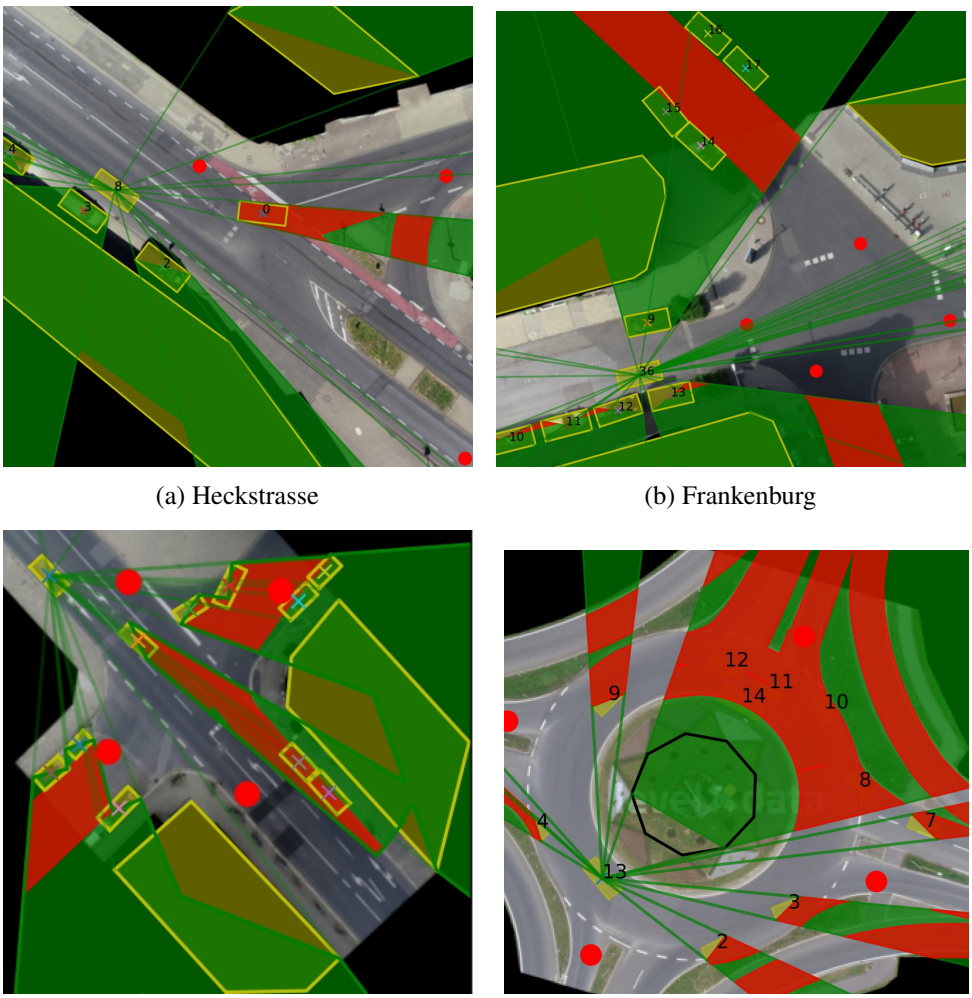
While training the decision trees, we use the following hyperparameter values. The cost complexity pruning parameter $\lambda = 0.0001$ was selected by grid search to maximise the true goal probability of OGRIT on the validation set. The maximum decision tree depth is set to 7, to ensure that the learned trees are interpretable. To help avoid overfitting, the minimum number of training samples allowed at each leaf node is set to 10. For Laplace smoothing of sample counts, a value of $\alpha = 1$ is used. We use uniform prior probabilities for the goal distributions.

6.4 Evaluation

We compare the performance of OGRIT to several baselines across four scenarios from two vehicle trajectory datasets. We show that OGRIT can achieve fast, accurate and interpretable inference under occlusion. We run formal verification for several propositions about inferences made by OGRIT.

6.4.1 Datasets

To train and evaluate our models, we use two datasets: the *inDO* dataset and the *rounDO* dataset. We generate these datasets by using the occlusion detector described in Section 6.2.3 to extract occluded regions in the inD dataset [139] and rounD dataset [143]. The inD and rounD datasets consist of vehicle trajectories extracted from videos taken by drones hovering above junctions or roundabouts. We use three scenarios from the



(a) Heckstrasse (b) Frankenburg

(c) Bendplatz (d) Neuweiler

Figure 6.3: Detected occlusions in each scenario. Vehicles and buildings are yellow, occluded regions are green and occluded lanes are red. Goal locations are shown by red dots.

inD dataset, named Heckstrasse, Bendplatz and Frankenburg, and one scenario from the rounD dataset, named Neuweiler. The data for each scenario is divided into continuous recordings. In each scenario in the inDO dataset, we hold out one randomly chosen recording for test, one recording for validation, and use the remaining recordings for training. Due to the larger number of recordings in the rounDO dataset, we hold out three recordings for validation and test. During each recording, we extract samples at one second intervals. At each interval, we consider each vehicle as an ego vehicle, and extract samples for each of the other target vehicles that are observable by the ego vehicle up to the point when each target vehicle reaches its goal.

6.4.2 Baselines

We compare OGRIT to several baselines, described in the following subsections. To the best of our knowledge, there are no GR methods which have published results on the inD or rounD datasets other than those presented in this thesis. To validate this, we used Google Scholar to search through all works which cite the inD or rounD datasets and mention "goal recognition" or "intent recognition".

6.4.2.1 OGRIT-Oracle

A version of OGRIT that has access to all occluded information. This method acts as an upper bound to the performance of OGRIT. For this baseline, the same hyperparameters were used as OGRIT, described in Section 6.3.2. During training and evaluation, the values of all features missing due to occlusions were provided to the model. The training process was also modified so that potentially missing features could be added to trees even if the corresponding indicator feature had not been added in an ancestor node.

6.4.2.2 GRIT

The GRIT method from Chapter 5, which only uses features from the set $\mathcal{B}$ which are never missing due to occlusions. This method has specialised DTs trained for each scenario. Similarly to OGRIT, we restrict the maximum tree depth to 7 for fair comparison. A different cost complexity pruning parameter λ was selected by grid search for each scenario. We do not allow potentially missing features to be added to the DTs during training, as GRIT has no way of handling features with missing values.

6.4.2.3 LSTM

Long Short-Term Memory (LSTM) neural network [146]. As input, the LSTM takes the sequence of observations for the target vehicle, along with a mask sequence which indicates when the target vehicle was occluded. The position and heading are imputed with default values during timesteps where the target vehicle was occluded. As output, the LSTM gives a probability distribution over goals. We trained a separate LSTM model for each scenario.

As input, the LSTM takes the state sequence $s^i_{1:t}$ for the target vehicle i up until the current timestep t, but during timesteps where i was occluded w.r.t the ego, the state is replaced with a default value of zero. A mask sequence which indicates when i was occluded is also given as additional input. We use a single LSTM layer, where each

hidden unit has cell size of 64. The outputs of the LSTM layer are passed through a fully connected layer with 725 hidden units.

6.4.2.4 IGP2

In addition to learning-based GR methods, we also use the inverse planning-based GR method from IGP2 described in Chapter 4. This method functions by using a planner to find the optimal plan for a goal from both the initially observed and current position of the target vehicle. The cost difference between the two plans is then used to calculate the goal likelihood. IGP2 can handle occlusions in observed vehicle trajectories by using a planner to fill sections of the trajectory missing due to occlusions, as detailed in [24]. We use same manoeuvres and macro actions as in original work, except for *Stop*.

We used the goal recognition module of IGP2, implemented as described in [24]. We used the parameter values of $v_{max} = speedlimit$, $a_{max} = 5$, and $\Delta_t = 0.1$ for the velocity smoother. Similarly to [25], we modify the velocity smoother loss function to improve convergence by using sum of squares instead of L2 norm.

We used the following parameter values across both datasets: give way distance of 15; give way lane angle threshold of $\pi/6$; give way turn target threshold of 1; manoeuvre point spacing of 0.25; manoeuvre max speed of 10; manoeuvre min speed of 3; switch lane minimum switch length. For the inDO dataset we used the following parameter values: time to goal reward weight of 0; angular velocity reward weight of 0; heading reward weight of 1000; acceleration reward weight of 0; switch lane target switch length of 20. For the rounDO dataset we used the following parameter values: time to goal reward weight of 0.01; angular velocity reward weight of 0.01; heading reward weight of 10; acceleration reward weight of 0.01; switch lane target switch length of 10.

6.4.3 Goal Recognition Accuracy

The mean probability assigned to the ground truth goal by OGRIT and the baselines is shown in Fig. 6.4. In all scenarios, OGRIT achieves higher accuracy than GRIT. OGRIT also uses the same learned DTs to generalise across all scenarios, while GRIT requires specialised DTs for each scenario. The accuracy of OGRIT is close to that of the oracle, despite not having access to occluded information. In all scenarios other than Heckstrasse, OGRIT achieves better performance than IGP2. This could be due to IGP2's strict assumption that vehicles use manoeuvres from a predefined library. In addition, OGRIT can use information from the initial observed vehicle state to make

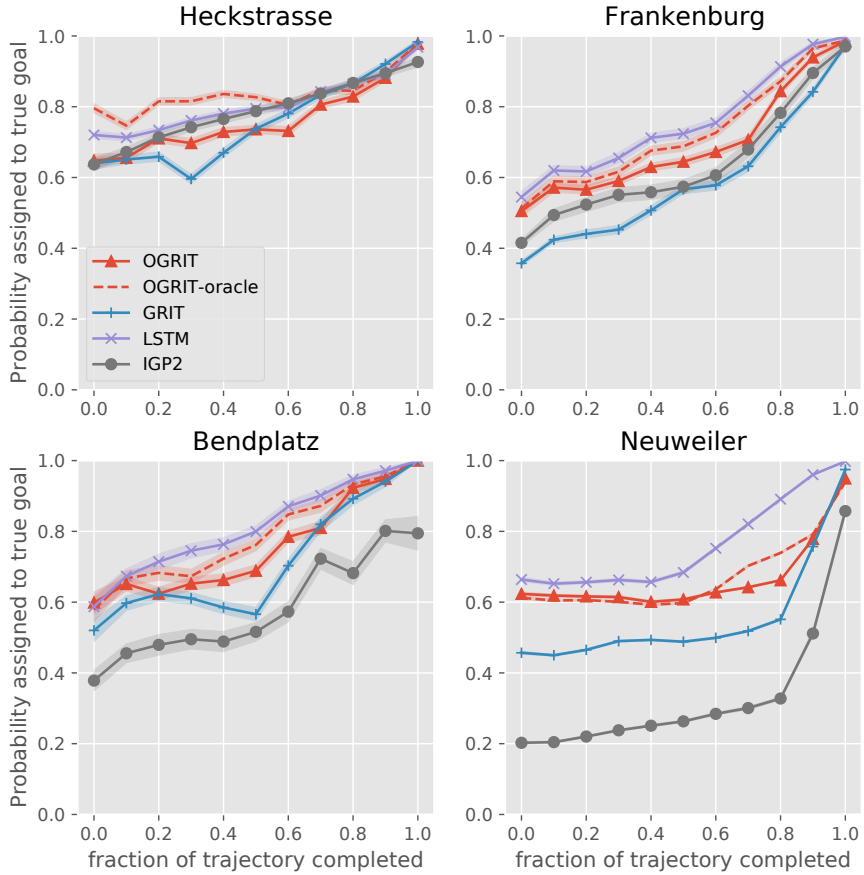


Figure 6.4: A comparison of mean probability assigned to true goal by OGRIT and several baselines (see Section 6.4.2). Fraction of trajectory completed is the fraction of time passed between the first observation of a vehicle and reaching its goal. Shaded areas show standard error. OGRIT-oracle acts as an upper bound for OGRIT.

inferences about goal likelihood. However, IGP2 only takes into account the vehicle's actions after its initial state. The LSTM achieves the best overall performance, which may be because it learns directly from raw trajectories rather than the handcrafted features used by OGRIT. However, the LSTM is not interpretable or verifiable due to its large number of learned parameters.

The largest performance difference between OGRIT and GRIT is found in the Neuweiler scenario, which contains a roundabout with four exits. In this scenario we found that the potentially missing feature *exit-number* has a large impact on the inferences made. The *exit-number* for a certain *exit-roundabout* goal $g_t^{i,k}$ represents the number of roundabout exits that would be passed by the vehicle i between entering the roundabout and reaching $g_t^{i,k}$. Knowledge of the *exit-number* is important for goal recognition, as vehicles rarely exit at the same place that they entered and rarely take the first exit due to the alternative slip roads seen in Fig. 6.3d.

6.4.4 Inference Speed

We measure the average time taken for OGRIT to compute the posterior probability over goals, using an AMD EPYC 7502 CPU. This includes the entire inference process shown in Fig. 6.2. The average inference time of OGRIT was 26 ms. This is fast enough to make inferences in real time to be used as input to a prediction and planning module. For comparison, GRIT had a faster inference time of 4.9 ms, due to the lack of occlusion detection and indicator feature extraction.

6.4.5 Interpretability

We found the that DTs learned by OGRIT are easily interpretable. For example, consider the DT for *exit-roundabout* goals shown in Fig. 6.1. If the leftmost leaf node is reached during inference, we can infer that: "The *exit-roundabout* goal has a likelihood of 0.0003, because the path to goal length is less than 84.73 metres (weight 1.32), the angle in lane is greater than than 0.61 radians to the right (weight 0.05), and the path to goal length is greater than 35.05 metres (weight 0.01)". The low likelihood makes sense in this case because the vehicle has turned harshly to the right, while still over 35 metres from the relevant roundabout exit.

6.4.6 Verification

To perform formal verification the trained model and a proposition to be verified Ψ are first represented using propositional logic. Next, we can verify that Ψ will always hold true by using a satisfiability modulo theories (SMT) solver to prove that $\neg\Psi$ is unsatisfiable. In our case, we use the Z3 solver [142]. In the case that verification fails, the SMT solver provides a counterexample which can teach us more about the way in which our model works. We ran verification for three propositions on the trained OGRIT model. We use x_t^g to represent the set of all feature values for goal g at timestep or scene instance t. The value of feature $l \in \mathcal{L}$ is represented by $x_t^{g,l}$. We use feature identifiers: *onm* (*oncoming-vehicle-missing*), *xn* (*exit-number*), *xnm* (*exit-number-missing*), *pth* (*path-to-goal-length*), *an* (*angle-in-lane*).

Goal distribution entropy with vehicle in front occluded: If certain feature values are missing, a reasonable expectation is that the model should be more uncertain about the goals of vehicles than if the feature values are not missing. In other words, the entropy of the goal distribution should be higher if the feature values are missing. In

the case that there are two goals, a decrease in the probability of the most probable goal is equivalent to an increase in entropy. For the case that a vehicle is at a junction where there are *straight-on* and *exit-left* goals possible, we successfully verify the following proposition: "If the *oncoming-vehicle-missing* indicator feature is true, then the entropy of the goal distribution should be greater than or equal to the case where the *oncoming-vehicle-missing* indicator feature is false, all other features being equal":

$$\bigwedge_{g\in\{G1,G2\}} ((x_{t1}^{g,onm} \wedge \neg x_{t2}^{g,onm}) \bigwedge_{\substack{l\in\mathcal{L},\\ l\neq onm}} (x_{t1}^{g,l} = x_{t2}^{g,l})) \implies ((P(G1|x_{t1}^{G1}) < P(G2|x_{t1}^{G2}) \implies$$

$$P(G2|x_{t2}^{G2}) \geq P(G2|x_{t1}^{G2})) \wedge (P(G1|x_{t1}^{G1}) > P(G2|x_{t1}^{G2}) \implies P(G1|x_{t2}^{G1}) \geq P(G1|x_{t1}^{G1})))$$

Goal probability with oncoming vehicles occluded: If a target vehicle is stopped at a junction entrance, one explanation for stopping may be that its goal is *enter-right*, and there is an oncoming vehicle which would block its way while turning. Even if oncoming vehicles are occluded, it would still be reasonable to assign a high probability to the *enter-right* goal, as there could be an oncoming vehicle hidden from view. However, if there are no occlusions and the ego vehicle can see that there are no oncoming vehicles, then stopping would be irrational for the *enter-right* goal and *enter-right* should be given a low probability. We run verification the proposition: "If a vehicle is stopped at a junction, angled straight ahead in its lane, and oncoming vehicles are occluded, then *enter-right* should have higher probability than if oncoming vehicles are not occluded and there is no oncoming vehicle", represented as the following, where $G1$ is the *enter-right* goal:

$$x_{t1}^{G1,onm} \wedge \neg x_{t2}^{G1,onm} \implies P(G1|x_{t2}^{G1}) \geq P(G1|x_{t1}^{G1})$$

In this case verification fails and the solver provides a counter example where there is a stopped vehicle 8.5 metres in front of the target vehicle. In such a case it would be rational for a vehicle with an *enter-right* goal to stop, even if there are no oncoming vehicles.

Exit roundabout goal likelihood with roundabout exit number occluded: When a vehicle is in a roundabout, it is a reasonable expectation that the goal of exiting to the same road from which a vehicle entered should have a lower likelihood than other exits. In the Neuweiler scenario, this corresponds to taking the fourth exit relative to the entry point. If the exit number for goal g is missing due to occlusions, then the *exit-number*

for g could have any value and it would be reasonable to expect that the goal likelihood should be higher than if the *exit-number* is known to be four. We verified following proposition: "If the exit number for goal g is known to be four, then the likelihood of g should be lower than or equal to the likelihood of g if the *exit-number* for g is missing due to occlusions, if *path-to-goal-length* is 50m, and *angle-in-lane* is zero, all other features being equal":

$$x_{t1}^{g,pth} = 50 \wedge x_{t1}^{g,an} = 0 \wedge x_{t2}^{g,xn} = 4 \wedge x_{t1}^{g,xnm} \wedge \neg x_{t2}^{g,xnm}$$

$$\bigwedge_{\substack{l \in \mathcal{L}, \\ l \neq xnm}} x_{t1}^{g,l} = x_{t2}^{g,l} \implies L(x_{t1}^{g}|g) \geq L(x_{t2}^{g}|g)$$

6.5 Limitations

Our method and evaluation have some limitations, which are important to address. We only evaluated OGRIT across four different static scenarios from two datasets. These four scenarios do not cover the full range of situations that may be encountered during real-world driving. In addition to this, the datasets used only contained 2D information. Because of this, the height of objects could not be taken into consideration when detecting which objects were occluded.

Another limitation of OGRIT is that it only takes into account occlusions from the point of view of the ego vehicle. It is also possible that road users could be occluded from the view of non-ego vehicles, which could influence their behaviour. Taking account of such occlusions could give additional information useful for goal recognition.

One further limiting assumption of OGRIT is that there is a separate module performing perception, and the output of this model is perfectly reliable, as long as information is not occluded. However, in real scenarios there may be innacuracies in perception due to sensor noise or failure or perception models to generalise.

6.6 Conclusion

We presented a novel autonomous vehicle goal recognition method named OGRIT. We showed that OGRIT can handle missing data due to occlusions and make inferences across multiple scenarios using the same model, while still resulting in fast, accurate, interpretable and verifiable inference.

Future directions could include considering occlusions from the point of view of the target vehicle in addition to the ego vehicle. OGRIT also assumes a predefined set of goal types, which may be incomplete. This could be addressed in future work through anomaly detection, by declaring the goal "unknown" if the likelihood for all generated goals is low. Future work could also work towards taking 3D information into account when detecting occlusions. A similar approach could be used as the method detailed in Section 6.2.3, but instead of using polygons, polyhedrons could be used to represent obsticals and occluded regions.

Chapter 7

Conclusion

This thesis presented a study of interpretable and verifiable methods for planning and prediction in autonomous vehicles (AVs), with a specific emphasis on goal recognition (GR) techniques. Existing planning and prediction methods have many shortcomings, and several novel methods have been introduced which address some of these problems. The contributions introduced by each of these methods are summarised in Section 7.1. The limitations of these methods are also discussed in Section 7.2, and possible directions for future work are mentioned in Section 7.3.

7.1 Summary of Contributions

In this thesis, several contributions to the field of planning and prediction for autonomous driving were provided. In Chapter 4, a GR method was introduced that can be used for multi-modal trajectory prediction, by making use of rational inverse planning. It was also shown how this GR method can be integrated with Monte Carlo Tree Search (MCTS) planning to obtain optimised plans for the ego vehicle, giving the IGP2 autonomous driving system. An evaluation of IGP2 across several urban driving scenarios was presented, and it was shown that IGP2 can achieve accurate goal recognition, efficient driving, and interpretable plans and predictions.

In Chapter 5 an alternative GR method named GRIT was introduced which can make fast inferences due to its straightforward inference process which makes use of learned DTs trained using vehicle trajectory data. GRIT was evaluated across four different urban driving scenarios using data from two different vehicle trajectory datasets and it was shown that GRIT achieves comparable accuracy to a neural network baseline. It was also shown that properties of the learned DTs can be formally verified using an off-

the-shelf satisfiability modulo theories (SMT) solver. It was also shown that inferences made by GRIT can be easily interpreted by humans thanks to the interpretable features used and shallow DT depth.

In Chapter 6, a novel GR method named OGRIT was presented which builds on GRIT and can handle occlusions and generalise across multiple scenarios, while still being fast, accurate, intereptable and verifiable. In busy urban environments, there is often information missing due to occlusions caused by obstacles such as buildings or other vehicles. Two new datasets of occluded regions were introduced, the inDO and rounDO datasets, which can be used to evaluate how well autonomous driving methods can handle occlusions, along with an open source tool that can be used to detect occluded regions in existing datasets. An evaluation of OGRIT on the inDO and rounDO datasets was also presented and it was shown that OGRIT comes close in accuracy to a neural network based GR method, while still being interpretable and verifiable.

7.2 Limitations

The methods presented in this thesis have some limitations that are important to discuss. In all three of the GR methods presented in this thesis, a set of possible goals are generated for each non-ego vehicle based on local road layout, for example at junction exits or visible lane ends. These are expected to be sufficient in the vast majority of cases, but during real-world driving it is possible the actual goal of a vehicle may not in the set of generate goals. In addition to this, the IGP2 method described in Chapter 4 makes several other assumptions. This method was evaluated within simulations which only represent an idealised version of real-world driver behaviour. When performing GR, the assumption was made that vehicles only perform manoeuvres and macro actions from a predefined library, which may be incomplete. It was also assumed that non-ego vehicles were approximately rational and were selecting actions to minimise a predefined cost function. It is possible that the predefined cost function would not accurately reflect the preference of actual agents in real-world driving.

The GRIT method presented in Chapter 5 had several limitations. GRIT was evaluated across four different static scenarios, and required separate DTs to be trained for each scenario. Because of this, it cannot be applied to new scenarios for which there is no available training data. GRIT also made the assumption that the environment is fully observable, but in real-world driving there is often information missing due to

occlusions. Some of these limitations were addressed by the OGRIT method presented in Chapter 6. OGRIT generalised across multiple scenarios using the same learned DTs, and handled data missing due to occlusions. However, OGRIT still has some limitations. OGRIT was only evaluated across a limited set of four different scenarios, which does not cover all situations that may be encountered during real world driving. The source datasets used to evaluate OGRIT only contain 2D information about objects in the scene, and so the height of objects was not taken into account when occlusions were detected. In addition, OGRIT only took into account occlusions from the point of view of the ego vehicle, and not occlusions from the point of view of non-ego vehicles.

Each of these methods also has the limitation of assuming that there is no noise or inaccurate values in perception. The methods assume that there is a separate module performing perception, which is perfectly reliable. However, in reality the perception output could be noisy and unreliable.

7.3 Future Work

There are many avenues by which future work could extend the work presented in this thesis, and address the limitations mentioned in Section 7.2. In each of the GR methods presented in this thesis, it was possible that the generated set of possible goals did not include the actual goal of the vehicle on which we are performing GR. Future work could address this problem by using anomaly detection, which would declared the goal to be unknown if the likelihood of all generated goals was low. In such a case, the ego vehicle could proceed with additional caution. In the IGP2 method described in Chapter 4, the assumption was made that drivers are behaving rationally. However, human drivers tend to have predictable irrational biases [137, 71], and future work could account for these. Another future direction to the extend IGP2 could be to use multi-agent MCTS to allow AVs to plan cooperatively.

There are also several ways in which future work could build on the research relating the the GRIT and OGRIT methods presented in Chapter 5 and Chapter 6. These methods were evaluated using recorded vehicle trajectories from four different fixed frame scenarios. It would be interesting to evaluate how well these methods perform on more varied data from open-world driving. Future work could also evaluate how well these GR methods perform when integrated with a larger planning and prediction system. For example, these OGRIT could be used to replace the inverse planning based GR module in the IGP2 planning and prediction system. Although GRIT and OGRIT

achieved a high accuracy, both of these methods still achieved slightly lower accuracy than an LSTM neural network in most scenarios. Another direction that future work could take would be to investigate methods of improving the accuracy of OGRIT. Recent work has shown that the accuracy of DTs can be improved through knowledge distillation from neural networks [108]. A similar approach could be tried to improve the accuracy of OGRIT by distilling knowledge from an LSTM neural network.

Another important direction for future work would be research into how to best to display the decisions and inferences made by autonomous driving systems to the passengers of vehicles. There has already been some research conducted to develop methods to generate natural language explanations [14] for the decisions made by IGP2. Similarly, future research could investigate methods of generating natural language explanations for inferences made by GRIT and OGRIT. Presenting visualisations of the inference process could also be an alternative avenue for improving interpretability. In addition to this, a quantitative comparison of different explanation methods could be valuable. For example, a user study could be conducted where users are surveyed about the interpretability of different explanations [151].

Another direction for future research would be to address noisy and unreliable perception. IGP2 currently assumes that perception output has no noise. Future work could improve this by simulating noisy perceptions during each MCTS rollout, which could results in a plan that is robust to perception noise. For GRIT and OGRIT, perception noise could be handled by assuming distributions over feature values, and then computing the expectation of the DT output.

Appendix A

Chapter 5 Appendix

A.1 Learned Decision Trees

In Fig.A.1 to A.7, we show the decision trees that were trained for each goal type for OGRIT. Red oval nodes represent an indicator feature from the set $\mathcal{C}$, blue octagonal nodes represent potentially missing features from the set $\mathcal{M}$, and green oval nodes represent leaf nodes. The rectangular nodes represent decision nodes with base features which are never missing, from the set $\mathcal{B}$. The goal likelihood at each leaf node is calculated using the multiplicative weights at each edge and an initial likelihood of 0.5. In some cases nodes with indicator features are included without the related potentially missing features being included. This can happen if simply adding the indicator node on its own leads to an impurity decrease. In some cases simply checking whether a feature could be missing due to occlusions can give some useful information about the scene. For example, occlusions may be more likely if there is a large number vehicles present, so the indicator feature could act as a proxy for checking the number of vehicles present.

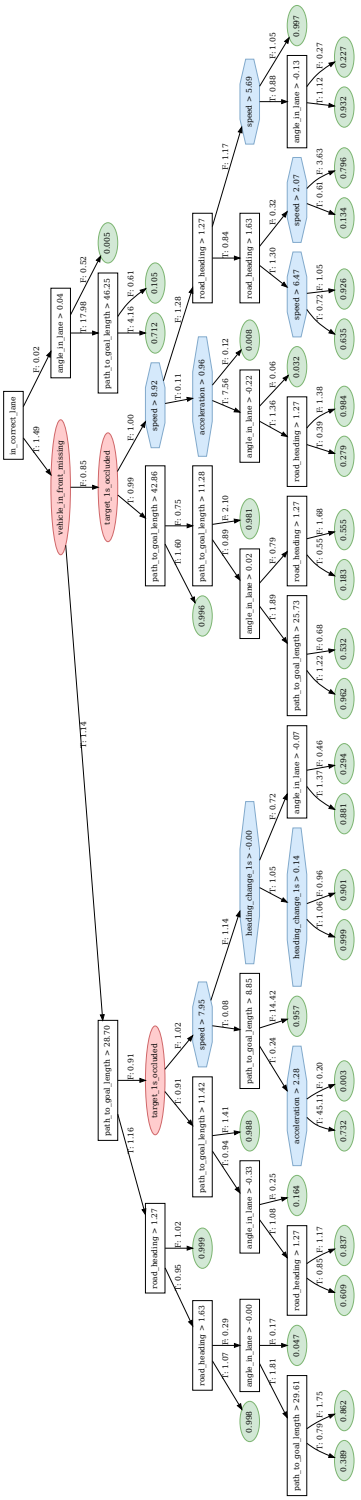


Figure A.1: Learned decision tree for the *exit-left* goal type.

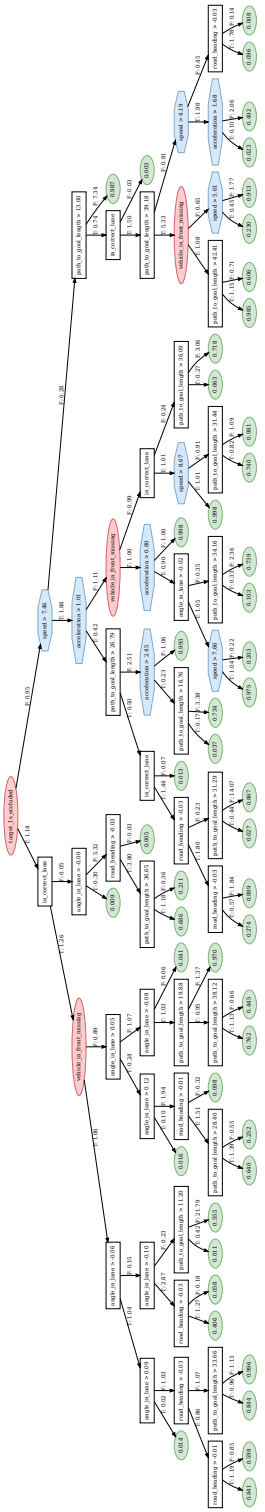


Figure A.2: Learned decision tree for the *straight-on* goal type.

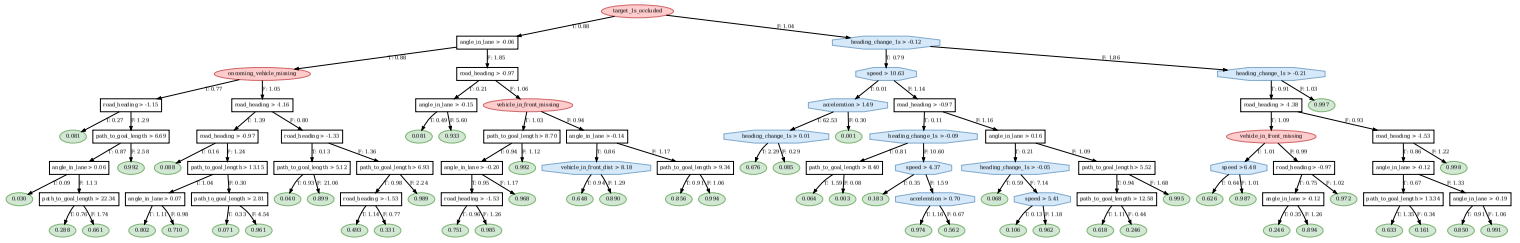


Figure A.3: Learned decision tree for the *exit-right* goal type.

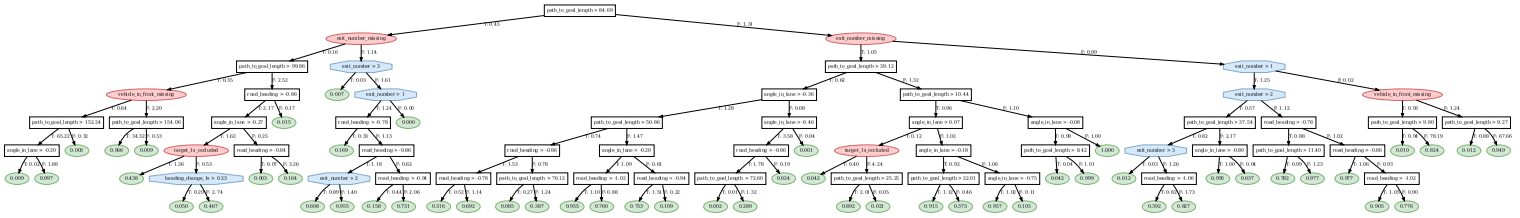


Figure A.4: Learned decision tree for the *exit-roundabout* goal type.

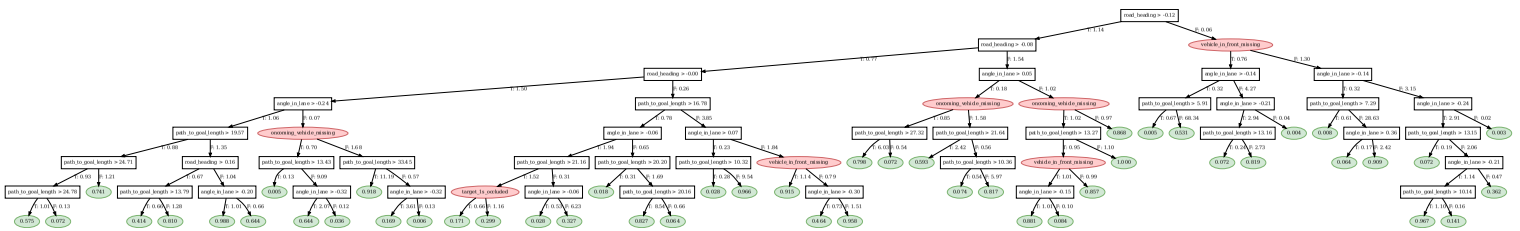


Figure A.5: Learned decision tree for the *cross-road* goal type.

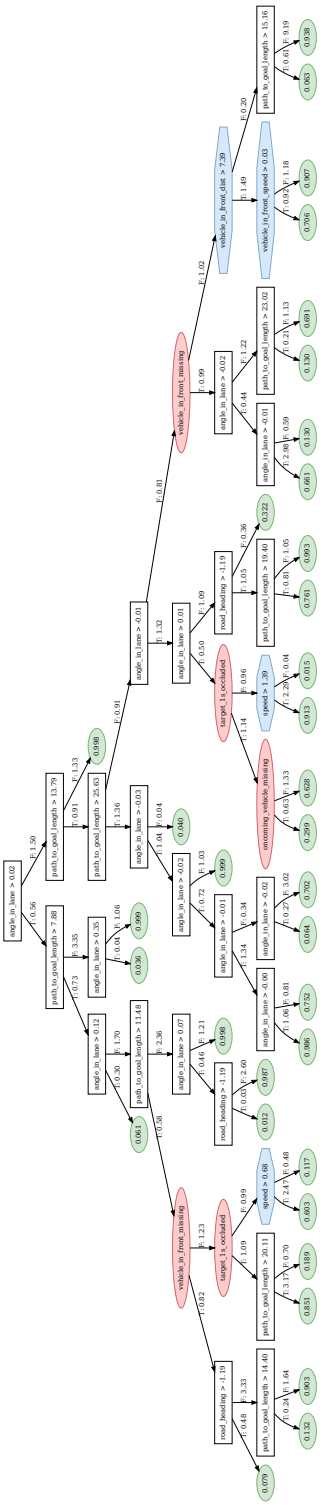


Figure A.6: Learned decision tree for the *enter-right* goal type.

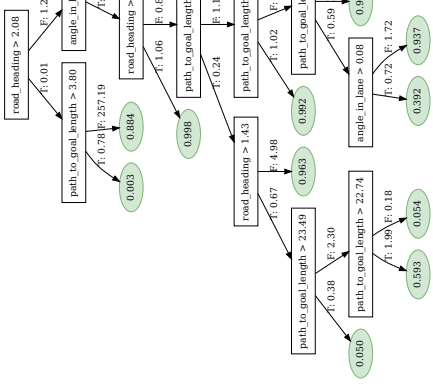


Figure A.7: Learned decision tree for the *enter-left* goal type.

Bibliography

[1] WHO. Global Status Report on Road Safety 2018. Technical Report 978-92-4-156568-4, 2018.

[2] Daniel J. Fagnant and Kara Kockelman. Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations. *Transportation Research Part A: Policy and Practice*, 77:167–181, 2015.

[3] Soheil Sohrabi, Haneen Khreis, and Dominique Lord. Impacts of Autonomous Vehicles on Public Health: A Conceptual Model and Policy Recommendations. *Sustainable Cities and Society*, 63:102457, 2020.

[4] Martin Buehler, Iagnemma Karl, and Sanjiv Singh. The 2005 DARPA Grand Challenge – The Great Robot Race. *Industrial Robot: An International Journal*, 35(6), 2008.

[5] Martin Buehler, Iagnemma Karl, and Sanjiv Singh. The DARPA Urban Challenge - Autonomous Vehicles in City Traffic. *Springer Tracts in Advanced Robotics*, 56, 2009.

[6] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:187–210, 2018. Publisher: Annual Reviews.

[7] Guillaume Bresson, Zayed Alsayed, Li Yu, and Sébastien Glaser. Simultaneous Localization and Mapping: A Survey of Current Trends in Autonomous Driving. *IEEE Transactions on Intelligent Vehicles*, 2(3):194–220, 2017.

[8] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. Visual SLAM algorithms: A survey from 2010 to 2016. *IPSJ Transactions on Computer Vision and Applications*, 9, 2017. Publisher: IPSJ Transactions on Computer Vision and Applications.

[9] Dean a Pomerleau. ALVINN: an autonomous land vehicle in a neural network (Technical Report CMU-CS-89-107). *Advances in Neural Information Processing Systems*, pages 305–313, 1989.

[10] Yann LeCun, Urs Muller, Jan Ben, Eric Cosatto, and Beat Flepp. Off-road obstacle avoidance through end-to-end learning. *Advances in Neural Information Processing Systems*, pages 739–746, 2005.

[11] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to End Learning for Self-Driving Cars, April 2016. arXiv:1604.07316 [cs].

[12] Zachary C. Lipton. The Mythos of Model Interpretability: In Machine Learning, the Concept of Interpretability is Both Important and Slippery. *Queue*, 16(3):31–57, June 2018. Place: New York, NY, USA Publisher: Association for Computing Machinery.

[13] Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision making and a "right to explanation". *AI Magazine*, 38(3):50–57, 2017.

[14] Balint Gyevnar, Massimiliano Tamborski, Cheng Wang, Christopher G. Lucas, Shay B. Cohen, and Stefano V. Albrecht. A Human-Centric Method for Generating Causal Explanations in Natural Language for Autonomous Vehicle Motion Planning. *IJCAI Workshop on Artificial Intelligence for Autonomous Driving*, 2022.

[15] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. On a Formal Model of Safe and Scalable Self-driving Cars, October 2018. arXiv:1708.06374 [cs, stat].

[16] R.W. Butler and G.B. Finelli. The infeasibility of quantifying the reliability of life-critical real-time software. *IEEE Transactions on Software Engineering*, 19(1):3–12, 1993.

[17] Matt Luckcuck, Marie Farrell, Louise A. Dennis, Clare Dixon, and Michael Fisher. Formal Specification and Verification of Autonomous Robotic Systems. *ACM Computing Surveys*, 52(5):1–41, October 2019. Publisher: Association for Computing Machinery (ACM).

[18] Michael Montemerlo, 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. Junior: The stanford entry in the urban challenge. *Springer Tracts in Advanced Robotics*, 56(October 2005):91–123, 2009.

[19] John Leonard, Jonathan How, Seth Teller, Mitch Berger, Stefan Campbell, Gaston Fiore, Luke Fletcher, Emilio Frazzoli, Albert Huang, Sertac Karaman, Olivier Koch, Yoshiaki Kuwata, David Moore, Edwin Olson, Steve Peters, Justin Teo, Robert Truax, Matthew Walter, David Barrett, Alexander Epstein, Keoni Maheloni, Katy Moyer, Troy Jones, Ryan Buckley, Matthew Antone, Robert Galejs, Siddhartha Krishnamurthy, and Jonathan Williams. A perception-driven autonomous urban vehicle. *Springer Tracts in Advanced Robotics*, 56(November 2007):163–230, 2009.

[20] Chris Urmson, 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 M. 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. Autonomous driving in Urban environments: Boss and the Urban Challenge. *Springer Tracts in Advanced Robotics*, 56:1–59, 2009.

[21] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21918.

[22] M. Ramírez and H. Geffner. Probabilistic plan recognition using off-the-shelf classical planners. In *24th AAAI Conference on Artificial Intelligence*, pages 1121–1126, 2010.

[23] Brian D Ziebart, Nathan Ratliff, Garratt Gallagher, Christoph Mertz, Kevin Peterson, J Andrew Bagnell, Martial Hebert, Anind K Dey, and Siddhartha Srinivasa. Planning-based prediction for pedestrians. In *Intelligent Robots and Systems*, pages 3931–3936, 2009.

[24] Stefano V Albrecht, Cillian Brewitt, John Wilhelm, Balint Gyevnar, Francisco Eiras, Mihai Dobre, and Subramanian Ramamoorthy. Interpretable Goal-based Prediction and Planning for Autonomous Driving. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[25] Cillian Brewitt, Balint Gyevnar, and Stefano V. Albrecht. GRIT: Verifiable Goal Recognition for Autonomous Driving using Decision Trees. *arXiv:2103.06113 [cs]*, March 2021. arXiv: 2103.06113.

[26] Cillian Brewitt, Massimiliano Tamborski, and Stefano V. Albrecht. Verifiable Goal Recognition for Autonomous Driving with Occlusions, June 2022. arXiv:2206.14163 [cs].

[27] Wilko Schwarting, Javier Alonso-Mora, Liam Paull, Sertac Karaman, and Daniela Rus. Safe Nonlinear Trajectory Generation for Parallel Autonomy with a Dynamic Vehicle Model. *IEEE Transactions on Intelligent Transportation Systems*, 19(9):2994–3008, 2018. Publisher: IEEE.

[28] Dave Ferguson, Thomas M. Howard, and Maxim Likhachev. Motion planning in urban environments. *Springer Tracts in Advanced Robotics*, 56:61–89, 2009.

[29] Mihail Pivtoraiko, Ross A. Knepper, and Alonzo Kelly. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 26(3):308–333, 2009.

[30] Brian Ichter, James Harrison, and Marco Pavone. Learning Sampling Distributions for Robot Motion Planning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7087–7094, May 2018. ISSN: 2577-087X.

[31] Sertac Karaman and Emilio Frazzoli. Incremental sampling-based algorithms for optimal motion planning. *Robotics: Science and Systems*, 6:267–274, 2011.

[32] Steven M Lavalle and James J Kuffner. Randomized kinodynamic planning. *Int. J. Robot. Res.*, 20:378–400, 2001.

[33] Lucas Janson, Edward Schmerling, Ashley Clark, and Marco Pavone. Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *The International Journal of Robotics Research*, 34(7):883–921, June 2015. Publisher: SAGE Publications Ltd STM.

[34] Lucas Janson, Brian Ichter, and Marco Pavone. Deterministic sampling-based motion planning: Optimality, complexity, and performance. *The International Journal of Robotics Research*, 37(1):46–61, January 2018. Publisher: SAGE Publications Ltd STM.

[35] Alexander Liniger, Alexander Domahidi, and Manfred Morari. Optimization-based autonomous racing of 1:43 scale RC cars. *Optimal Control Applications and Methods*, 36(5):628–647, 2015.

[36] Paolo Falcone, Francesco Borrelli, Jahan Asgari, Hongtei Eric Tseng, and Davor Hrovat. Predictive active steering control for autonomous vehicle systems. *IEEE Transactions on Control Systems Technology*, 15(3):566–580, 2007.

[37] Francisco Eiras, Majd Hawasly, Stefano V. Albrecht, and Subramanian Ramamoorthy. A Two-Stage Optimization-based Motion Planner for Safe Urban Driving, June 2021. arXiv:2002.02215 [cs, math].

[38] Henry Pulver, Francisco Eiras, Ludovico Carozza, Majd Hawasly, Stefano V. Albrecht, and Subramanian Ramamoorthy. PILOT: Efficient Planning by Imitation Learning and Optimisation for Safe Autonomous Driving, July 2021. arXiv:2011.00509 [cs].

[39] Riccardo Bonalli, Abhishek Cauligi, Andrew Bylard, and Marco Pavone. GuSTO: Guaranteed Sequential Trajectory optimization via Sequential Convex Programming. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6741–6747, May 2019. ISSN: 2577-087X.

[40] Zhijie Zhu, Edward Schmerling, and Marco Pavone. A convex optimization approach to smooth trajectories for motion planning with car-like robots. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 835–842, December 2015.

[41] Sumeet Singh, Anirudha Majumdar, Jean-Jacques Slotine, and Marco Pavone. Robust online motion planning via contraction theory and convex optimization.

In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5883–5890, May 2017.

[42] Laurene Claussmann, Ashwin Carvalho, and Georg Schildbach. A path planner for autonomous driving on highways using a human mimicry approach with Binary Decision Diagrams. *2015 European Control Conference, ECC 2015*, pages 2976–2982, 2015. Publisher: EUCA.

[43] Isaac Miller, Mark Campbell, Dan Huttenlocher, Aaron Nathan, Frank Robert Kline, Pete Moran, Noah Zych, Brian Schimpf, Sergei Lupashin, Ephrahim Garcia, Jason Catlin, Mike Kurdziel, and Hikaru Fujishima. Team Cornell's Skynet: Robust perception and planning in an urban environment. *Springer Tracts in Advanced Robotics*, 56:257–304, 2009.

[44] Dorsa Sadigh, Shankar Sastry, Sanjit A. Seshia, and Anca D. Dragan. Planning for autonomous cars that leverage effects on human actions. *Robotics: Science and Systems*, 12, 2016.

[45] Pete Trautman, Jeremy Ma, Richard M. Murray, and Andreas Krause. Robot navigation in dense human crowds: Statistical models and experimental studies of human-robot cooperation. *International Journal of Robotics Research*, 34(3):335–356, 2015.

[46] Henrik Kretzschmar, Markus Spies, Christoph Sprunk, and Wolfram Burgard. Socially compliant mobile robot navigation via inverse reinforcement learning. *The International Journal of Robotics Research*, 35(11):1289–1307, 2016. _eprint: https://doi.org/10.1177/0278364915619772.

[47] Michael Düring and Patrick Pascheka. Cooperative decentralized decision making for conflict resolution among autonomous agents. *INISTA 2014 - IEEE International Symposium on Innovations in Intelligent Systems and Applications, Proceedings*, pages 154–161, 2014. Publisher: IEEE.

[48] Mohammad Bahram, Andreas Lawitzky, Jasper Friedrichs, Michael Aeberhard, and Dirk Wollherr. A Game-Theoretic Approach to Replanning-Aware Interactive Scene Prediction and Planning. *IEEE Transactions on Vehicular Technology*, 65(6):3981–3992, 2016. Publisher: IEEE.

[49] David Lenz, Tobias Kessler, and Alois Knoll. Tactical cooperative planning for autonomous highway driving using Monte-Carlo Tree Search. *IEEE Intelligent Vehicles Symposium, Proceedings*, 2016-Augus(Iv):447–453, 2016. Publisher: IEEE.

[50] Wilko Schwarting and Patrick Pascheka. Recursive conflict resolution for cooperative motion planning in dynamic highway traffic. *2014 17th IEEE International Conference on Intelligent Transportation Systems, ITSC 2014*, pages 1039–1044, 2014. Publisher: IEEE.

[51] Nan Li, Dave W. Oyler, Mengxuan Zhang, Yildiray Yildiz, Ilya Kolmanovsky, and Anouck R. Girard. Game theoretic modeling of driver and vehicle interactions for verification and validation of autonomous vehicle control systems. *IEEE Transactions on Control Systems Technology*, 26(5):1782–1797, 2018.

[52] Junqing Wei, John M. Dolan, and Bakhtiar Litkouhi. Autonomous vehicle social behavior for highway entrance ramp management. *IEEE Intelligent Vehicles Symposium, Proceedings*, (Iv):201–207, 2013.

[53] Wei Liu, Seong Woo Kim, Scott Pendleton, and Marcelo H. Ang. Situation-aware decision making for autonomous driving on urban road using online POMDP. *IEEE Intelligent Vehicles Symposium, Proceedings*, 2015-Augus(Iv):1126–1133, 2015. Publisher: IEEE.

[54] Sebastian Brechtel, Tobias Gindele, and Rudiger Dillmann. Probabilistic decision-making under uncertainty for autonomous driving using continuous POMDPs. *2014 17th IEEE International Conference on Intelligent Transportation Systems, ITSC 2014*, pages 392–399, 2014.

[55] E. Galceran, A.G. Cunningham, R.M. Eustice, and E. Olson. Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction: Theory and experiment. *Autonomous Robots*, 41(6):1367–1382, 2017. Publisher: Springer.

[56] Tirthankar Bandyopadhyay, Kok Sung Won, Emilio Frazzoli, David Hsu, Wee Sun Lee, and Daniela Rus. Intention-aware motion planning. In *Algorithmic Foundations of Robotics X*, pages 475–491. Springer, 2013.

[57] Charlott Vallon, Ziya Ercan, Ashwin Carvalho, and Francesco Borrelli. A machine learning approach for personalized autonomous lane change initiation and control. *IEEE Intelligent Vehicles Symposium, Proceedings*, (Iv):1590–1595, 2017. Publisher: IEEE.

[58] Mariusz Bojarski, Philip Yeres, Anna Choromanska, Krzysztof Choromanski, Bernhard Firner, Lawrence Jackel, and Urs Muller. Explaining How a Deep Neural Network Trained with End-to-End Learning Steers a Car. pages 1–8, 2017.

[59] Luca Caltagirone, Mauro Bellone, Lennart Svensson, and Mattias Wahde. LIDAR-based driving path generation using fully convolutional neural networks. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2018-March:1–6, 2018.

[60] Dan Barnes, Will Maddern, and Ingmar Posner. Find your own way: Weakly-supervised segmentation of path proposals for urban autonomy. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 203–210, 2017.

[61] Huazhe Xu, Yang Gao, Fisher Yu, and Trevor Darrell. End-to-end learning of driving models from large-scale video datasets. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua:3530–3538, 2017.

[62] Peter Wolf, Christian Hubschneider, Michael Weber, Andre Bauer, Jonathan Hartl, Fabian Durr, and J. Marius Zollner. Learning how to drive in a real world simulation with deep Q-Networks. *IEEE Intelligent Vehicles Symposium, Proceedings*, (Iv):244–250, 2017.

[63] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016.

[64] Xinlei Pan, Yurong You, Ziyan Wang, and Cewu Lu. Virtual to real reinforcement learning for autonomous driving. *British Machine Vision Conference 2017, BMVC 2017*, 2017.

[65] Stéphanie Lefèvre, Ashwin Carvalho, Yiqi Gao, H. Eric Tseng, and Francesco Borrelli. Driver models for personalised driving assistance. *Vehicle System Dynamics*, 53(12):1705–1720, 2015.

[66] Stefan Hoermann, Daniel Stumper, and Klaus Dietmayer. Probabilistic long-Term prediction for autonomous vehicles. *IEEE Intelligent Vehicles Symposium, Proceedings*, (Iv):237–243, 2017. Publisher: IEEE.

[67] Niclas Evestedt, Erik Ward, John Folkesson, and Daniel Axehill. Interaction aware trajectory planning for merge scenarios in congested traffic situations. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pages 465–472, 2016. Publisher: IEEE.

[68] Yuning Chai, Benjamin Sappm, Mayank Bansal, and Dragomir Anguelov. Multi-Path: Multiple Probabilistic Anchor Trajectory Hypotheses for Behavior Prediction. In *Conference on Robot Learning*, 2019.

[69] Yifei Xu, Tianyang Zhao, Chris Baker, Yibiao Zhao, and Ying Nian Wu. Learning trajectory prediction with continuous inverse optimal control via Langevin sampling of energy-based models. *arXiv preprint arXiv:1904.05453*, 2019.

[70] Sergio Casas, Wenjie Luo, and Raquel Urtasun. IntentNet: Learning to Predict Intention from Raw Sensor Data. In *Conference on Robot Learning*, pages 947–956, 2018.

[71] Yeping Hu, Liting Sun, and Masayoshi Tomizuka. Generic prediction architecture considering both rational and irrational driving behaviors. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3539–3546. IEEE, 2019.

[72] Jiachen Li, Hengbo Ma, and Masayoshi Tomizuka. Conditional Generative Neural System for Probabilistic Trajectory Prediction. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2019, Macau, SAR, China, November 3-8, 2019*, pages 6150–6156. IEEE, 2019.

[73] S. Mozaffari, O. Y. Al-Jarrah, M. Dianati, P. Jennings, and A. Mouzakitis. Deep Learning-Based Vehicle Behavior Prediction for Autonomous Driving Applications: A Review. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–15, 2020.

[74] Boris Ivanovic and Marco Pavone. The Trajectron: Probabilistic Multi-Agent Trajectory Modeling With Dynamic Spatiotemporal Graphs. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2375–2384, Seoul, Korea (South), October 2019. IEEE.

[75] Edward W. Ayers, Francisco Eiras, Majd Hawasly, and Iain Whiteside. PaRoT: A Practical Framework for Robust Deep Neural Network Training. In Ritchie Lee, Susmit Jha, Anastasia Mavridou, and Dimitra Giannakopoulou, editors, *NASA Formal Methods*, pages 63–84, Cham, 2020. Springer International Publishing.

[76] Nicholas Rhinehart, Rowan McAllister, Kris Kitani, and Sergey Levine. PRECOG: Prediction conditioned on goals in visual multi-agent settings. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2821–2830, 2019.

[77] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. DESIRE: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 336–345, 2017.

[78] Stefano V. Albrecht and Peter Stone. Autonomous Agents Modelling Other Agents: A Comprehensive Survey and Open Problems. *Artificial Intelligence*, 258:66–95, 2018. Publisher: Elsevier.

[79] Stefano V. Albrecht, Jacob W. Crandall, and Subramanian Ramamoorthy. Belief and Truth in Hypothesised Behaviours. *Artificial Intelligence*, 235:63–94, 2016. Publisher: Elsevier.

[80] Chiyu Dong, John M Dolan, and Bakhtiar Litkouhi. Smooth behavioral estimation for ramp merging control in autonomous driving. In *IEEE Intelligent Vehicles Symposium*, pages 1692–1697. IEEE, 2018.

[81] Weilong Song, Guangming Xiong, and Huiyan Chen. Intention-aware autonomous driving decision-making in an uncontrolled intersection. *Mathematical Problems in Engineering*, 2016, 2016. Publisher: Hindawi.

[82] Bingyu Zhou, Wilko Schwarting, Daniela Rus, and Javier Alonso-Mora. Joint multi-policy behavior estimation and receding-horizon trajectory planning for

automated urban driving. In *IEEE International Conference on Robotics and Automation*, pages 2388–2394. IEEE, 2018.

[83] Jason Hardy and Mark Campbell. Contingency planning over probabilistic obstacle predictions for autonomous road vehicles. *IEEE Transactions on Robotics*, 29(4):913–929, 2013. Publisher: IEEE.

[84] Josiah P. Hanna, Arrasy Rahman, Elliot Fosong, Francisco Eiras, Mihai Dobre, John Redford, Subramanian Ramamoorthy, and Stefano V Albrecht. Interpretable Goal Recognition in the Presence of Occluded Factors for Autonomous Vehicles. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7044–7051, United States, December 2021. Institute of Electrical and Electronics Engineers (IEEE).

[85] Morris Antonello, Mihai Dobre, Stefano V. Albrecht, John Redford, and Subramanian Ramamoorthy. Flash: Fast and Light Motion Prediction for Autonomous Driving with Bayesian Inverse Planning and Learned Motion Profiles, August 2022. arXiv:2203.08251 [cs].

[86] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. *Proceedings of the National Conference on Artificial Intelligence*, 3:1433–1438, 2008.

[87] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Maximum Entropy Deep Inverse Reinforcement Learning. 2015.

[88] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization, May 2016. arXiv:1603.00448 [cs].

[89] Dorsa Sadigh, Anca D. Dragan, Shankar Sastry, and Sanjit A. Seshia. Active preference-based learning of reward functions. *Robotics: Science and Systems*, 13, 2017.

[90] Daniel Omeiza, Helena Webb, Marina Jirotka, and Lars Kunze. Explanations in Autonomous Driving: A Survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(8):10142–10162, August 2022.

[91] W Kim and T Kelly-Baker. Users' Trust in and Concerns about Automated Driving Systems. Technical report, AAA Foundation for Traffic Safety, 2021.

[92] Shahin Atakishiyev, Mohammad Salameh, Hengshuai Yao, and Randy Goebel. Explainable artificial intelligence for autonomous driving: An overview and guide for future research directions, April 2022. arXiv:2112.11561 [cs].

[93] Rasheed Hussain and Sherali Zeadally. Autonomous Cars: Research Results, Issues, and Future Challenges. *IEEE Communications Surveys & Tutorials*, 21(2):1275–1313, 2019.

[94] Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. Verifiable Reinforcement Learning via Policy Extraction. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[95] S Krening, B Harrison, K Feigh, and C Isbell. Learning from Explanations using Sentiment and Advice in RL. *on Cognitive and . . .*, (December):1–12, 2016.

[96] Hédi Ben-Younes, Éloi Zablocki, Patrick Pérez, and Matthieu Cord. Driving Behavior Explanation with Multi-level Fusion. *Pattern Recognition*, 123:108421, March 2022. arXiv:2012.04983 [cs].

[97] Jinkyu Kim, Anna Rohrbach, Trevor Darrell, John Canny, and Zeynep Akata. Textual Explanations for Self-Driving Vehicles. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, volume 11206, pages 577–593. Springer International Publishing, Cham, 2018. Series Title: Lecture Notes in Computer Science.

[98] Grégoire Montavon, Sebastian Bach, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining NonLinear Classification Decisions with Deep Taylor Decomposition. *Pattern Recognition*, 65:211–222, May 2017. arXiv:1512.02479 [cs, stat].

[99] Scott Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions, November 2017. arXiv:1705.07874 [cs, stat].

[100] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier, August 2016. arXiv:1602.04938 [cs, stat].

[101] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Frcitas. Dueling Network Architectures for Deep Reinforcement

Learning. *33rd International Conference on Machine Learning, ICML 2016*, 4(9):2939–2947, 2016.

[102] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, March 1986.

[103] J. R. Quinlan. C4. 5 programs for machine learning. Elsevier. 2014.

[104] Leo Breiman. *Classification and regression trees*, volume 1. 1st editio edition, 1984. Issue: 1 Publication Title: Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery.

[105] Sachin Gavankar and Sudhirkumar Sawarkar. Decision Tree: Review of Techniques for Missing Values at Training, Testing and Compatibility. In *2015 3rd International Conference on Artificial Intelligence, Modelling and Simulation (AIMS)*, pages 122–126, December 2015.

[106] Pasha Khosravi, Antonio Vergari, YooJung Choi, Yitao Liang, and Guy Van den Broeck. Handling Missing Data in Decision Trees: A Probabilistic Approach. *arXiv:2006.16341 [cs, stat]*, June 2020. arXiv: 2006.16341.

[107] Jerome Friedman, Ron Kohavi, and Yeogirl Yun. Lazy Decision Trees. *Proc. of the AAAI*, 1, September 1997.

[108] Xuan Liu, Xiaoguang Wang, and Stan Matwin. Improving the interpretability of deep neural networks with knowledge distillation. *IEEE International Conference on Data Mining Workshops, ICDMW*, 2018-Novem:905–912, 2019. Publisher: IEEE.

[109] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. pages 1–9, 2015.

[110] Mark W Craven and Jude W Shavlik. Extracting tree-structured representations of trained neural networks. *Advances in Neural Information Processing Systems*, 8:24–30, 1996.

[111] Nicholas Frosst and Geo rey Hinton. Distilling a neural network into a soft decision tree. *CEUR Workshop Proceedings*, 2071, 2018.

[112] Zhengping Che, Sanjay Purushotham, Robinder Khemani, and Yan Liu. Distilling Knowledge from Deep Networks with Applications to Healthcare Domain. pages 1–13, 2015.

[113] Zhengping Che, Sanjay Purushotham, Robinder Khemani, and Yan Liu. Interpretable Deep Models for ICU Outcome Prediction. *AMIA ... Annual Symposium proceedings. AMIA Symposium*, 2016:371–380, 2016.

[114] Yin Lou, Rich Caruana, and Johannes Gehrke. Intelligible models for classification and regression. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 150–158, 2012.

[115] Lukas M. Schmidt, Georgios Kontes, Axel Plinge, and Christopher Mutschler. Can You Trust Your Autonomous Car? Interpretable and Verifiably Safe Reinforcement Learning. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 171–178, July 2021.

[116] Changliu Liu, Tomer Arnon, Christopher Lazarus, Christopher Strong, Clark Barrett, and Mykel J. Kochenderfer. Algorithms for Verifying Deep Neural Networks, October 2020. arXiv:1903.06758 [cs, stat].

[117] Guy Katz, Clark Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In Rupak Majumdar and Viktor Kunčak, editors, *Computer Aided Verification*, pages 97–117, Cham, 2017. Springer International Publishing.

[118] Xin Li, Xiaowen Ying, and Mooi Choo Chuah. GRIP: Graph-based Interaction-aware Trajectory Prediction. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3960–3966, 2019.

[119] Constantin Hubmann, Jens Schulz, Marvin Becker, Daniel Althoff, and Christoph Stiller. Automated driving in uncertain environments: Planning with interaction and uncertain maneuver prediction. *IEEE Transactions on Intelligent Vehicles*, 3(1):5–17, 2018. Publisher: IEEE.

[120] Constantin Hubmann, Marvin Becker, Daniel Althoff, David Lenz, and Christoph Stiller. Decision making for autonomous driving considering interaction and uncertain prediction of surrounding vehicles. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1671–1678. IEEE, 2017.

[121] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Benjamin Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai, Cordelia Schmid, Congcong Li, and Dragomir Anguelov. TNT: Target-driven trajectory prediction. In *Conference on Robot Learning*, 2020.

[122] Markus Wulfmeier, Dominic Zeng Wang, and Ingmar Posner. Watch this: Scalable cost-function learning for path planning in urban environments. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2089–2095. IEEE, 2016.

[123] Abbas Sadat, Sergio Casas, Mengye Ren, Xinyu Wu, Pranaab Dhawan, and Raquel Urtasun. Perceive, predict, and plan: Safe motion planning through interpretable semantic representations. In *European Conference on Computer Vision*, pages 414–430. Springer, 2020.

[124] Jack Stewart. Why People Keep Rear-Ending Self-Driving Cars, 2020. Published: WIRED magazine.

[125] Matthew Gadd, Daniele De Martini, Letizia Marchegiani, Paul Newman, and Lars Kunze. Sense–Assess–eXplain (SAX): Building Trust in Autonomous Vehicles in Challenging Real-World Driving Scenarios. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 150–155. IEEE, 2020.

[126] P. Koopman, R. Hierons, S. Khastgir, J. A. Clark, M. Fisher, R. Alexander, K. Eder, P. Thomas, G. Barrett, Philip H. S. Torr, A. Blake, S. Ramamoorthy, and J. McDermid. Certification of Highly Automated Vehicles for Use on UK Roads: Creating An Industry-Wide Framework for Safety. 2019.

[127] C.L. Baker, R. Saxe, and J.B. Tenenbaum. Action Understanding as Inverse Planning. *Cognition*, 113(3):329–349, 2009. Publisher: Elsevier.

[128] C. Browne, E. Powley, D. Whitehouse, S. Lucas, P.I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012. Publisher: IEEE.

[129] Citlalli Gámez Serna and Yassine Ruichek. Dynamic speed adaptation for path tracking based on curvature information and speed limits. *Sensors*, 17(1383), 2017. Publisher: Multidisciplinary Digital Publishing Institute.

[130] Andreas Wächter and Lorenz Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006. Publisher: Springer.

[131] Scott Niekum, Sarah Osentoski, Christopher Atkeson, and Andrew Barto. Online Bayesian changepoint detection for articulated motion models. In *IEEE International Conference on Robotics and Automation*. IEEE, 2015.

[132] P.E. Hart, N.J. Nilsson, and B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. In *IEEE Transactions on Systems Science and Cybernetics*, volume 4, pages 100–107, July 1968. Issue: 2.

[133] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002. Publisher: Springer.

[134] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical Review E*, 62(2):1805, 2000. Publisher: APS.

[135] Christopher Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992. Publisher: Springer.

[136] Adhiraj Somani, Nan Ye, David Hsu, and Wee Sun Lee. DESPOT: Online POMDP planning with regularization. In *Advances in Neural Information Processing Systems*, pages 1772–1780, 2013.

[137] Minae Kwon, Erdem Biyik, Aditi Talati, Karan Bhasin, Dylan P. Losey, and Dorsa Sadigh. When Humans Aren't Optimal: Robots that Collaborate with Risk-Aware Humans. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction*, 2020.

[138] Michael Fisher, Viviana Mascardi, Kristin Yvonne Rozier, Bernd-Holger Schlingloff, Michael Winikoff, and Neil Yorke-Smith. Towards a framework for certification of reliable autonomous systems. *Autonomous Agents and Multi-Agent Systems*, 35(1):8, December 2020.

[139] Julian Bock, Robert Krajewski, Tobias Moers, Steffen Runde, Lennart Vater, and Lutz Eckstein. The inD Dataset: A Drone Dataset of Naturalistic Road User Trajectories at German Intersections. In *arXiv preprint arXiv:1911.07602*, 2019.

[140] Hatem Darweesh, Eijiro Takeuchi, and Kazuya Takeda. Estimating the Probabilities of Surrounding Vehicles' Intentions and Trajectories using a Behavior Planner. *IJAE*, pages 299–308, 2019.

[141] Anton Zita, Sahar Mohajerani, and Martin Fabian. Application of formal verification to the lane change module of an autonomous vehicle. In *13th IEEE Conference on Automation Science and Engineering, CASE 2017, Xi'an, China, August 20-23, 2017*, pages 932–937. IEEE, 2017.

[142] Leonardo de Moura and Nikolaj Bjørner. Z3: An Efficient SMT Solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science, pages 337–340, Berlin, Heidelberg, 2008. Springer.

[143] R. Krajewski, T. Moers, J. Bock, L. Vater, and L. Eckstein. The rounD Dataset: A Drone Dataset of Road User Trajectories at Roundabouts in Germany. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, September 2020.

[144] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition, 2009.

[145] Fabian Poggenhans, Jan Hendrik Pauls, Johannes Janosovits, Stefan Orf, Maximilian Naumann, Florian Kuhnt, and Matthias Mayr. Lanelet2: A high-definition map framework for the future of automated driving. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2018-Novem:1672–1679, 2018.

[146] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, November 1997. Place: Cambridge, MA, USA Publisher: MIT Press.

[147] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, March 2010. ISSN: 1938-7228.

[148] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, January 2017. arXiv:1412.6980 [cs].

[149] Karttikeya Mangalam, Harshayu Girase, Shreyas Agarwal, Kuan-Hui Lee, Ehsan Adeli, Jitendra Malik, and Adrien Gaidon. It Is Not the Journey but the Destination: Endpoint Conditioned Trajectory Prediction. Technical Report arXiv:2004.02025, arXiv, July 2020. arXiv:2004.02025 [cs] type: article.

[150] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[151] Andrew Silva, Matthew Gombolay, Taylor Killian, Ivan Jimenez, and Sung-Hyun Son. Optimization Methods for Interpretable Differentiable Decision Trees Applied to Reinforcement Learning. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1855–1865. PMLR, August 2020.