Verifiable Goal Recognition for Autonomous Driving with Occlusions

Cillian Brewitt* cillian.brewitt@ed.ac.uk
Massimiliano Tamborski* m.tamborski@sms.ed.ac.uk
Stefano V. Albrecht*† s.albrecht@ed.ac.uk

*School of Informatics, University of Edinburgh, UK
†Five AI Ltd., UK

Abstract

Goal recognition (GR) allows the future behaviour of vehicles to be more accurately predicted. GR involves inferring the goals of other vehicles, such as a certain junction exit. In autonomous driving, vehicles can encounter many different scenarios and the environment is partially observable due to occlusions. We present a novel GR method named Goal Recognition with Interpretable Trees under Occlusion (OGRIT). We demonstrate that OGRIT can handle missing data due to occlusions and make inferences across multiple scenarios using the same learned decision trees, while still being fast, accurate, interpretable and verifiable. We also present the inDO and rounDO datasets of occluded regions used to evaluate OGRIT.

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 [1, 2, 3]. For example, a planner can be used to plan multiple possible trajectories to an inferred goal, which can then be used for trajectory prediction [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 [5, 6, 7]. Existing GR methods for AVs fail to satisfy all of the requirements of being fast, accurate, interpretable, verifiable, generalisable and able to handle occlusions.

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 [8] and rounD [9]. We show that OGRIT can handle occlusions and generalise across multiple scenarios, while still being fast, accurate, and interpretable. We also formally verify that

Machine Learning for Autonomous Driving Workshop at the 36th Conference on Neural Information Processing Systems (NeurIPS 2022), New Orleans, USA.

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

2 Related Work

Many existing AV prediction methods make use of deep neural networks [1, 10, 11, 12, 13, 14, 15, 16]. 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 [17]. Another approach to prediction is to first perform GR through inverse planning [18, 19, 20, 21]. A recent method named IGP2 [4] uses an interpretable inverse planning approach to GR, which can handle occlusions by using a planner to fill gaps in trajectories. Agents are assumed to be rational and unobserved sections of trajectories are imputed with the optimal plan. A similar GR method named GOFI [22] uses inverse planning to infer occluded factors. 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.

A recent method named GRIT [23] 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.

There are many existing methods which allow DT inference with missing data [24]. Some methods compute the expected inference based on the distributions over the values of missing features [25, 26]. 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 [27], 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 our work, we instead introduce a novel DT training algorithm designed to handle missing feature values while resulting in trained DTs which have fast, interpretable and verifiable inference.

3 Problem Definition

Let $\mathcal{I}$ represent the local set of vehicles surrounding the ego vehicle under our control. The state of vehicle $i \in \mathcal{I}$ at time t is represented by $s_t^i \in \mathcal{S}^i$, and includes information about the pose, velocity and acceleration of the vehicle. The state of all vehicles at time t is given by $s_t = \{s_t^i | i \in \mathcal{I}\} \in \mathcal{S}$. At each time t, the ego vehicle receives an observation $o_t \subseteq s_t$ which contains the state of a subset of vehicles. The state of some vehicles may be missing from o_t due to occlusions. The observations

[1] **inDO and rounDO datasets:** `https://datashare.ed.ac.uk/handle/10283/4480`

[2] **OGRIT code:** `https://github.com/uoe-agents/OGRIT`

between times a and b are denoted by $o_{a:b} = \{o_a, o_{a+1}, ..., o_b\}$. We assume a set of possible goals $G_t^i = \{g_t^{i,1}, g_t^{i,2}, ...\}$ for each vehicle i at time t, where a goal is a subset of states $g_t^{i,k} \subset \mathcal{S}$ 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, $P(g_t^{i,k}|o_{1:t}, \phi)$, where ϕ represents static scene information such as the road layout and static obstacles.

4 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 missing data due to occlusions. As shown in Eq. (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:

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

Similar to the existing GRIT method [23], 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 4.6 which allows them to handle missing features, while still being highly efficient, interpretable and verifiable.

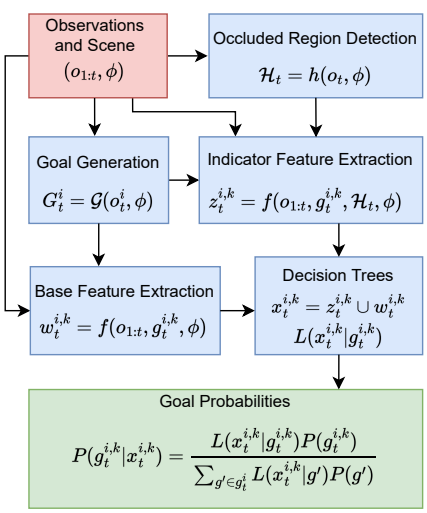
An overview of the inference process can be seen in Fig. 1a. As input, OGRIT takes the static scene information ϕ and the observation history $o_{1:t}$. As output, OGRIT gives the posterior distribution over goals $P(g_t^{i,k}|o_{1:t}, \phi)$ for vehicle i. First, a set of possible goals is generated based on the current state of the vehicle and the road layout. Next, a set of base features is extracted for each goal (Section 4.2), 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. (1). The following subsections will detail each of these components.

4.1 Goal Generation

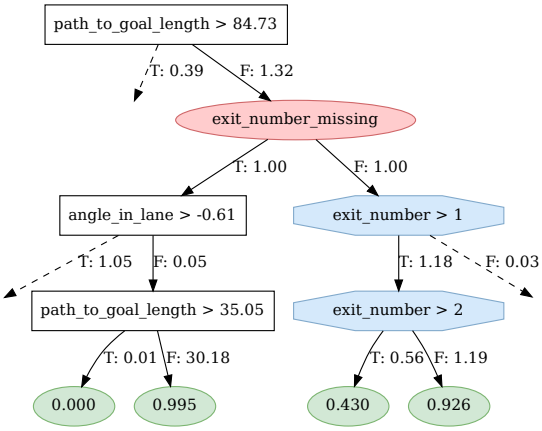
For each observable vehicle i at time t, we generate a set of possible goals G_t^i using a goal generator function $\mathcal{G}(s_t^i, \phi)$. The possible goals are generated by starting from the current lane of vehicle i and performing a breadth first search across the directed graph of lane connections 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.

4.2 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; in-correct lane; path-to-goal-length; junction-heading-change; roundabout-exit-number;*



(a) Red represents input and green is output.



(b) Dashed edges point to truncated subtrees. Red oval node represents an indicator feature. Blue octagonal nodes represent potentially missing features (see Section 4.4). The goal likelihood at each leaf node is calculated using the multiplicative weights at each edge and an initial likelihood of 0.5.

Figure 1: (a) OGRIT inference system and (b) illustrative learned DT for *exit-roundabout* goal type.

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.

4.3 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. 2. 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.

4.4 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-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. 1b

4.5 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. (1). For simplicity, we use uniform prior probabilities. A weight is assigned to each edge in the decision tree, as shown in Fig. 1b. 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.

4.6 Decision Tree Training

Similar to DT training methods such as CART [28], ID3 [26] and C4.5 [29], 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 1. Decision nodes are added recursively as shown in Algorithm 2 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) [28]. Decision nodes are iteratively pruned to minimise the cost function shown in Eq. (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:

$$C(T) = \sum_{n \in T} q_n + \lambda |T| \tag{2}$$

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 2.

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

Algorithm 1 Train Decision Tree

Input: dataset D

Returns: decision tree T

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

Algorithm 2 Recursively Grow Tree

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

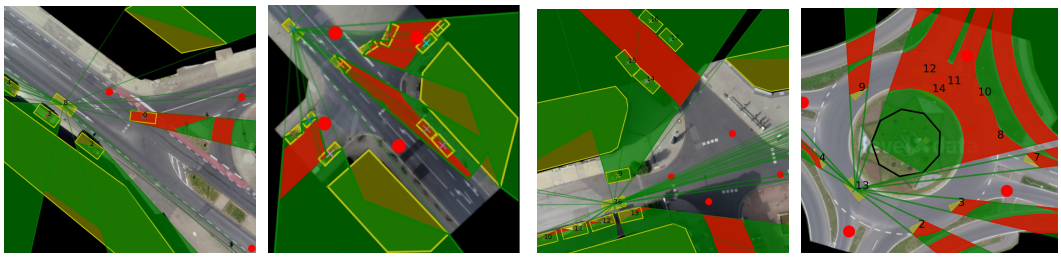
if $\neg terminate(D_n)$ **then**
  $\Delta q_n \leftarrow 0$
  for all $l | l \in \mathcal{L} \setminus \mathcal{M} \vee ind(l) \in \mathcal{C}_f$ **do**
    $c_n^l \leftarrow \arg\max_c(impuritydecrease(c, l, D_n))$
    $\Delta q \leftarrow -impuritydecrease(c_n^l, l, D_n)$
    if $\Delta q < \Delta q_n$ **then**
      $c_n, q_n, l_n \leftarrow c_n^l, q, l$
  for all $l | l \in \mathcal{M} \wedge ind(l) \notin \mathcal{C}_t \cup \mathcal{C}_f$ **do**
    $D_{nf} \leftarrow \{(x_j, y_j) | \neg x_{j,ind(l)} \wedge (x_j, y_j) \in D_n\}$
    $c_{nf} \leftarrow \arg\max_c(impuritydecrease(c, l, D_{nf}))$
    $\Delta q \leftarrow -(impuritydecrease(ind(l), D_n) + impuritydecrease(c_n^l, l, D_{nf}) - \lambda)$
    if $\Delta q < \Delta q_n$ **then**
      $c_n, q_n, l_n \leftarrow 0.5, q, ind(l)$
  if $\Delta q_n < 0$ **then**
    $D_{nt} \leftarrow \{(x_j, y_j) | x_{j,l_n} > c_n \wedge (x_j, y_j) \in D_n\}$
    $D_{nf} \leftarrow D_n \setminus D_{nt}$
    if $l_n \in \mathcal{C}$ **then**
      $\mathcal{C}_t' \leftarrow \mathcal{C}_t \cup \{l_n\}$
      $\mathcal{C}_f' \leftarrow \mathcal{C}_f \cup \{l_n\}$
    else
      $\mathcal{C}_t' \leftarrow \mathcal{C}_t$
      $\mathcal{C}_f' \leftarrow \mathcal{C}_f$
    $recursivelygrowtree(truechild(n), D_{nt}, \mathcal{C}_t', \mathcal{C}_f)$
    $recursivelygrowtree(falsechild(n), D_{nf}, \mathcal{C}_t, \mathcal{C}_f')$

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. (3):

$$L_n = \frac{w_G N_{nG}}{w_G N_{nG} + w_{\bar{G}} N_{n\bar{G}}} \tag{3}$$

5 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.



(a) Heckstrasse (b) Bendplatz (c) Frankenburg (d) Neuweiler

Figure 2: 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.

5.1 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 4.3 to extract occluded regions in the inD dataset [8] and rounD dataset [9]. 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 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.

5.2 Baselines

We compare OGRIT to several baselines, with implementation details given in Appendix A.3.

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.

GRIT: The existing GRIT [23] method, 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.

LSTM: Long Short-Term Memory (LSTM) neural network [30]. 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.

IGP2: In addition to learning-based GR methods, we also use the inverse planning-based GR method from IGP2 [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 [4]. We use same maneuvers and macro actions as in original work, except for *Stop*.

5.3 Goal Recognition Accuracy

The mean probability assigned to the ground truth goal by OGRIT and the baselines is shown in Fig. 3. 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 maneuvers from a predefined library. In addition, OGRIT can use information from the initial observed vehicle state to make inferences about goal likelihood.

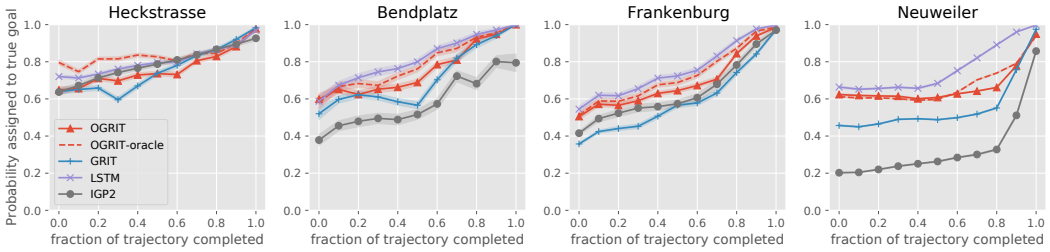


Figure 3: A comparison of mean probability assigned to true goal by OGRIT and several baselines (see Section 5.2). Fraction of trajectory completed is the fraction of time passed between the first observation of a vehicle and reaching its goal. Shaded areas show standard error. OGRIT-oracle acts as an upper bound for OGRIT.

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. 2d.

5.4 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. 1a. 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.

5.5 Interpretability

We found the that DTs learned by OGRIT are easily interpretable. For example, consider the DT for *exit-roundabout* goals shown in Fig. 1b. 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.

5.6 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 [31]. 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 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.

References

[1] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine, "PRECOG: Prediction conditioned on goals in visual multi-agent settings," in *IEEE International Conference on Computer Vision*, 2019.

[2] K. Mangalam, H. Girase, S. Agarwal, K.-H. Lee, E. Adeli, J. Malik, and A. Gaidon, "It Is Not the Journey But the Destination: Endpoint Conditioned Trajectory Prediction," in *European Conference on Computer Vision*, 2020.

[3] H. Zhao, J. Gao, T. Lan, C. Sun, B. Sapp, B. Varadarajan, Y. Shen, Y. Shen, Y. Chai, C. Schmid, C. Li, and D. Anguelov, "Tnt: Target-driven trajectory prediction," in *Conference on Robotic Learning*, 2020.

[4] 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.

[5] R. Butler and G. Finelli, "The infeasibility of quantifying the reliability of life-critical real-time software," *IEEE Transactions on Software Engineering*, 1993.

[6] M. Luckcuck, M. Farrell, L. A. Dennis, C. Dixon, and M. Fisher, "Formal Specification and Verification of Autonomous Robotic Systems," *ACM Computing Surveys*, 2019.

[7] L. M. Schmidt, G. Kontes, A. Plinge, and C. Mutschler, "Can You Trust Your Autonomous Car? Interpretable and Verifiably Safe Reinforcement Learning," in *IEEE Intelligent Vehicles Symposium*, 2021.

[8] J. Bock, R. Krajewski, T. Moers, S. Runde, L. Vater, and L. Eckstein, "The ind dataset: A drone dataset of naturalistic road user trajectories at german intersections," in *IEEE Intelligent Vehicles Symposium*, 2020.

[9] 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 *IEEE International Conference on Intelligent Transportation Systems*, 2020.

[10] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. S. Torr, and M. Chandraker, "DESIRE: Distant future prediction in dynamic scenes with interacting agents," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[11] Y. Chai, B. Sappm, M. Bansal, and D. Anguelov, "MultiPath: Multiple Probabilistic Anchor Trajectory Hypotheses for Behavior Prediction," in *Conference on Robot Learning*, 2019.

[12] Y. Xu, T. Zhao, C. Baker, Y. Zhao, and Y. N. Wu, "Learning trajectory prediction with continuous inverse optimal control via Langevin sampling of energy-based models," *arXiv preprint arXiv:1904.05453*, 2019.

[13] S. Casas, W. Luo, and R. Urtasun, "IntentNet: Learning to Predict Intention from Raw Sensor Data," in *Conference on Robot Learning*, 2018.

[14] Y. Hu, L. Sun, and M. Tomizuka, "Generic Prediction Architecture Considering both Rational and Irrational Driving Behaviors," in *IEEE Intelligent Transportation Systems Conference*, 2019.

[15] J. Li, H. Ma, and M. Tomizuka, "Conditional Generative Neural System for Probabilistic Trajectory Prediction," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019.

[16] 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*, 2020.

[17] E. W. Ayers, F. Eiras, M. Hawasly, and I. Whiteside, "PaRoT: A Practical Framework for Robust Deep Neural Network Training," in *NASA Formal Methods*, 2020.

[18] J. Hardy and M. Campbell, "Contingency Planning Over Probabilistic Obstacle Predictions for Autonomous Road Vehicles," *IEEE Transactions on Robotics*, 2013.

[19] T. Bandyopadhyay, K. S. Won, E. Frazzoli, D. Hsu, W. S. Lee, and D. Rus, "Intention-aware motion planning," in *Algorithmic Foundations of Robotics X*, 2013.

[20] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," *Proceedings of the National Conference on Artificial Intelligence*, 2008.

[21] H. Darweesh, E. Takeuchi, and K. Takeda, "Estimating the Probabilities of Surrounding Vehicles' Intentions and Trajectories using a Behavior Planner," *International Journal of Automotive Engineering*, 2019.

[22] J. Hanna, A. Rahman, E. Fosong, F. Eiras, M. Dobre, J. Redford, S. Ramamoorthy, and S. Albrecht, "Interpretable goal recognition in the presence of occluded factors for autonomous vehicles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2021.

[23] 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.

[24] S. Gavankar and S. Sawarkar, "Decision Tree: Review of Techniques for Missing Values at Training, Testing and Compatibility," in *International Conference on Artificial Intelligence, Modelling and Simulation*, 2015.

[25] P. Khosravi, A. Vergari, Y. Choi, Y. Liang, and G. V. den Broeck, "Handling missing data in decision trees: A probabilistic approach," in *ICML Workshop on the Art of Learning with Missing Values*, 2020.

[26] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, 1986.

[27] J. Friedman, R. Kohavi, and Y. Yun, "Lazy Decision Trees," *Association for the Advancement of Artificial Intelligence*, 1997.

[28] L. Breiman, *Classification and regression trees*. Chapman and Hall, 1984.

[29] J. R. Quinlan, *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., 1993.

[30] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, 1997.

[31] L. de Moura and N. Bjørner, "Z3: An Efficient SMT Solver," in *Tools and Algorithms for the Construction and Analysis of Systems*, 2008.

Appendix

A Method Details

A.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.

A.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.

A.3 Baseline Implementation Details

A.3.1 OGRIT-Oracle

For this baseline, the same hyperparameters were used as OGRIT, described in Appendix A.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.

A.3.2 GRIT

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.

A.3.3 LSTM

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.

A.4 IGP2

We used the goal recognition module of IGP2, implemented as described in [4]. We used the parameter values of $v_{max} = speedlimit$, $a_{max} = 5$, and $\Delta_t = 0.1$ for the velocity smoother. Similarly to [23], 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; maneuver point spacing of 0.25; maneuver max speed of 10; maneuver 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.

B Learned Decision Trees

In Fig.4 to 10, we show the decision trees that were trained for each goal type for OGRIT. In the shown trees, the red oval nodes represent an indicator feature from the set $\mathcal{C}$, the blue octagonal nodes represent potentially missing features from the set $\mathcal{M}$, and the 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.

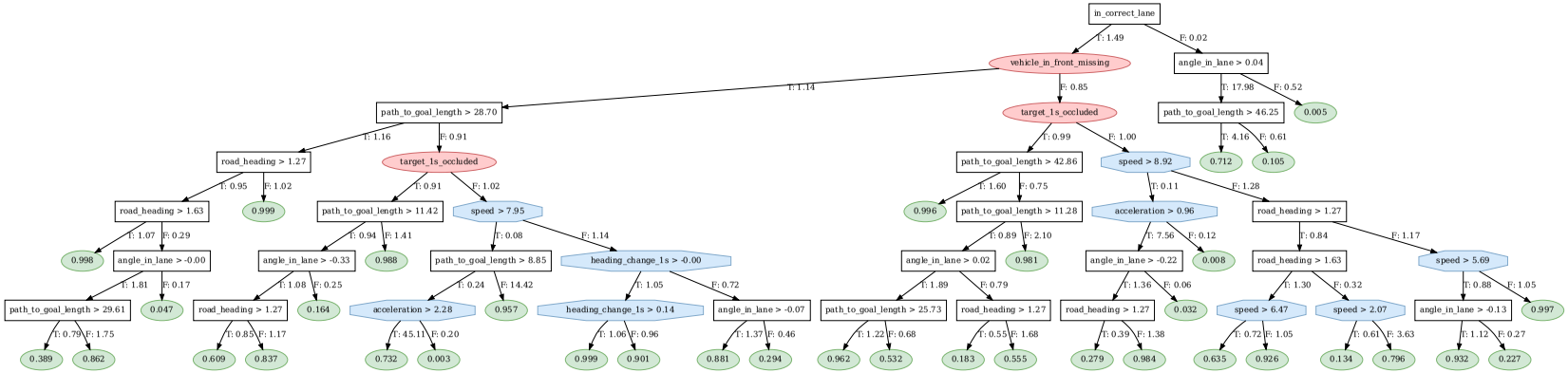


Figure 4: Learned decision tree for the $exit\text{-}left$ goal type.

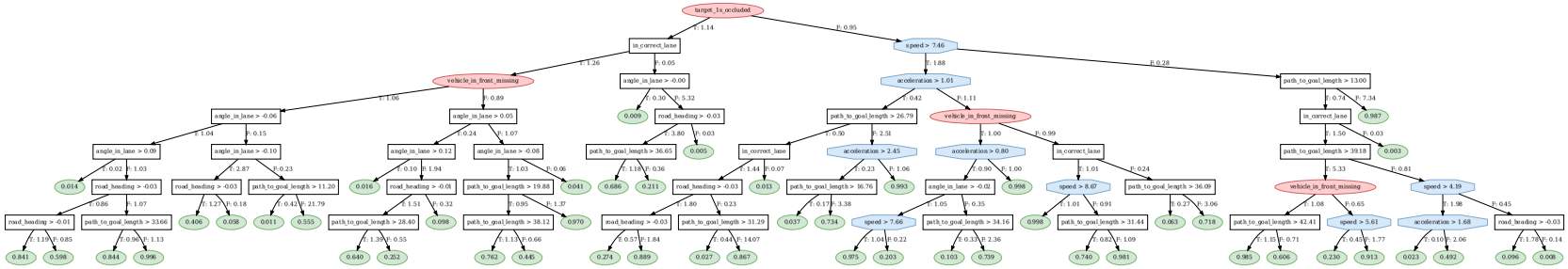


Figure 5: Learned decision tree for the $straight\text{-}on$ goal type.

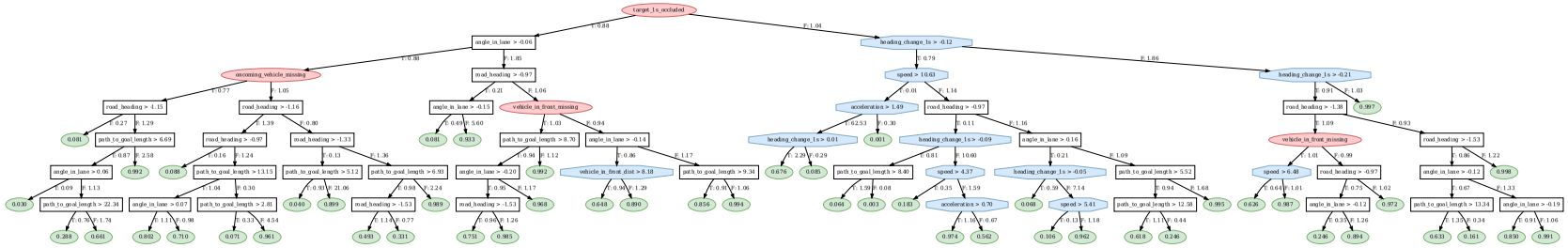


Figure 6: Learned decision tree for the *exit-right* goal type.

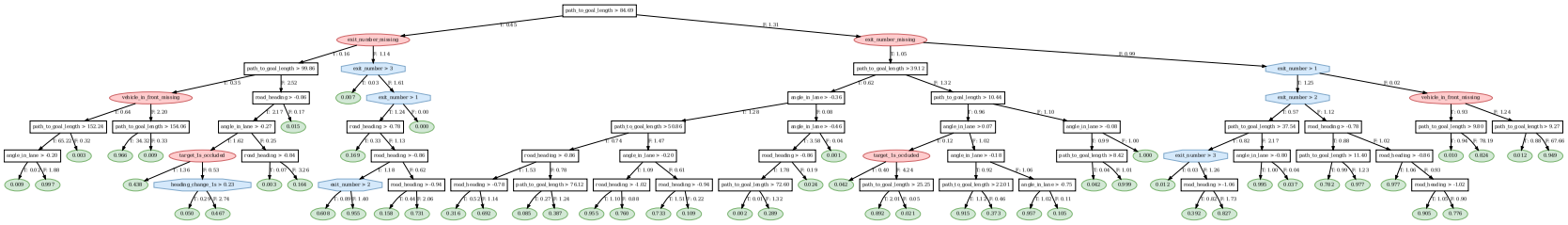


Figure 7: Learned decision tree for the *exit-roundabout* goal type.

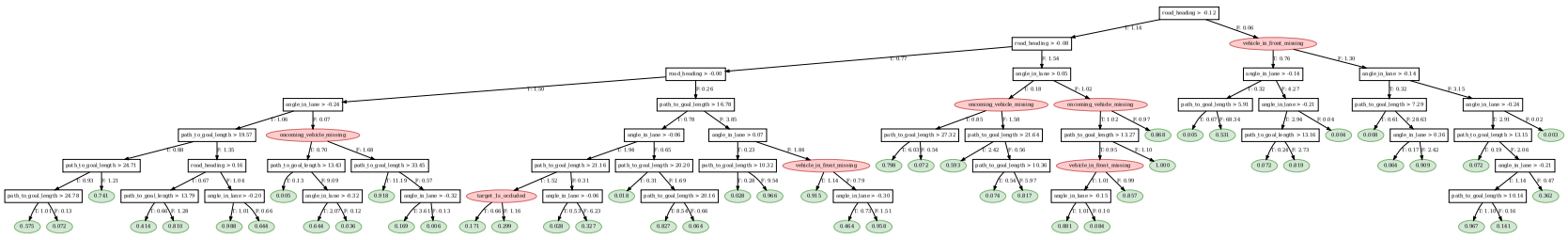


Figure 8: Learned decision tree for the *cross-road* goal type.

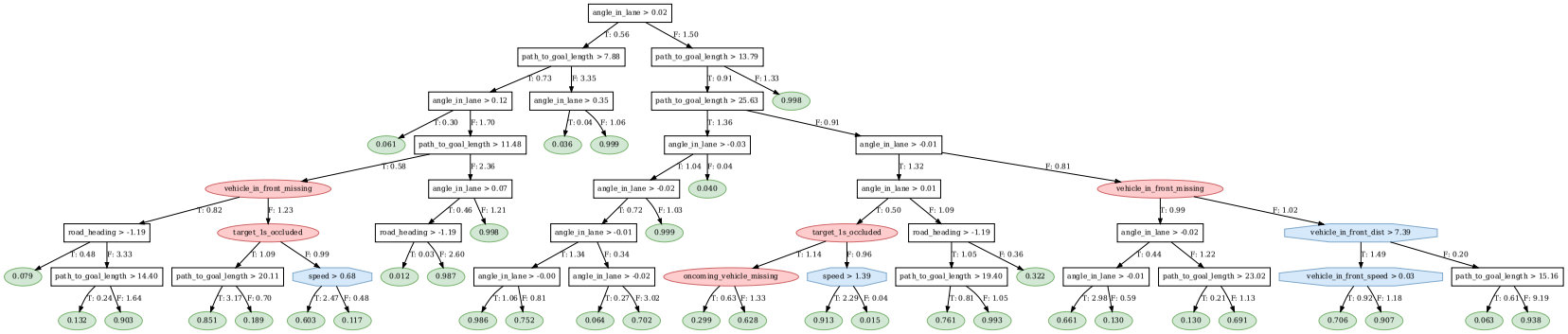


Figure 9: Learned decision tree for the *enter-right* goal type.

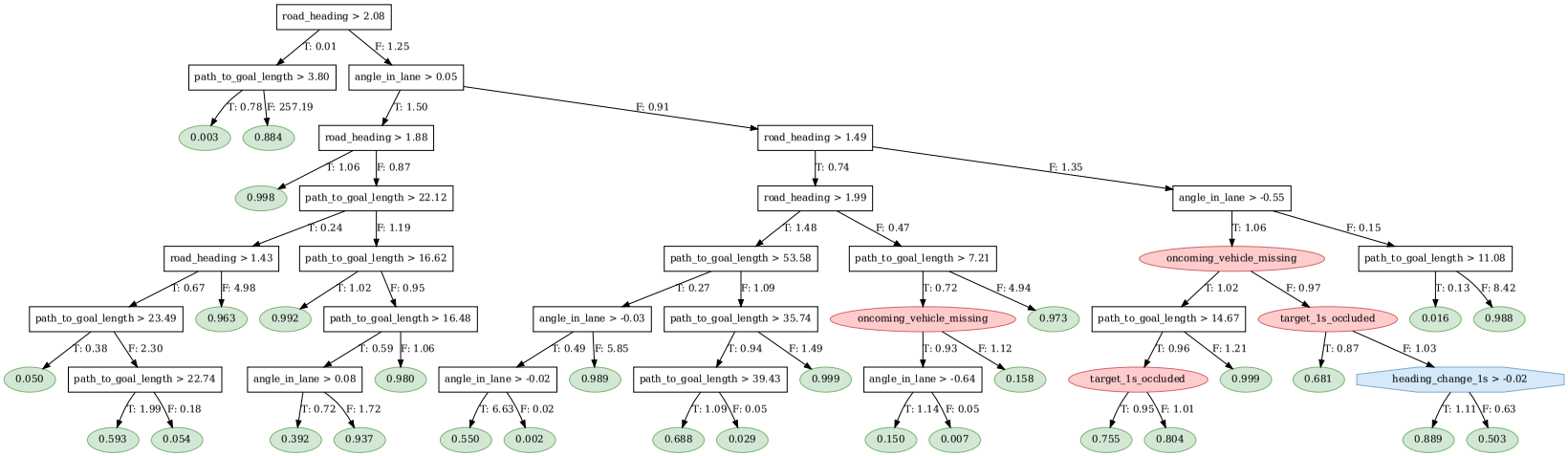


Figure 10: Learned decision tree for the *enter-left* goal type.