Test-Driven Inverse Reinforcement Learning Using Scenario-Based Testing

Johannes Fischer[1,3], Moritz Werling[2], Martin Lauer[1], Christoph Stiller[1]

***Abstract*—Automated vehicles require carefully designed cost functions, which are challenging to specify due to the complexity of the behavior they need to cover. Inverse reinforcement learning is a principled methodology for deriving cost functions, but it requires high-quality expert demonstrations, which are expensive to obtain. Recently, scenario-based testing has emerged as a promising approach for validation of driving behavior. In this paper, we introduce a novel methodology that circumvents the need for costly expert driving demonstrations by harnessing scenario-based testing. Our Test-Driven Inverse Reinforcement Learning approach leverages Bayesian inference, utilizing the outcomes of scenario tests as observations to infer cost functions. We rigorously evaluate our method on simulated and real-world scenarios and demonstrate its ability to learn cost functions that successfully pass the respective scenario tests. We also show that the learned cost function generalizes well by also passing scenario tests from an unseen validation set and illustrate that few scenario tests are sufficient to learn meaningful cost functions. This innovative framework not only streamlines the cost function specification process but also offers a cost-effective and practical solution for advancing automated driving systems.**

I. Introduction

Automated vehicles will play an important role in future society. They are expected to not only reduce the number of road fatalities, but also increase transportation availability and carbon footprint [1]. To successfully implement automated vehicles driving on public roads, it is crucial to ensure that they behave in a safe and predictable manner. Motion planning for automated vehicles is typically formulated in terms of a cost function that is optimized while at the same time respecting constraints of the environment [2].

An increasingly popular approach for validating automated driving solutions is provided by Scenario-Based Testing (SBT) [3], [4]. In SBT, a set of scenarios with test conditions is used to verify that an automated system behaves as expected. These tests can be automatically evaluated in a continuous-integration pipeline of the system's development cycle. Then a human expert tunes the cost function parameters until the system passes the tests. The test principle of this approach is illustrated in Fig. 1.

Alternatively, Inverse Reinforcement Learning (IRL) can be used to infer a cost function from expert demonstrations [5], [6]. Some works also take planning constraints into account when inferring the cost function [7]–[9]. Other approaches query a human expert for trajectory preferences at run time instead of using prerecorded demonstrations [10]–[14]. Negative feedback is employed into the cost function learning process by modeling physical corrections by the human [15], [16] or by using examples of failed demonstrations [17]–[19]. These approaches all have the downside of requiring a human expert to tune the cost function, provide demonstrations, correct the robot, or answer preference queries, which is time-consuming and expensive.

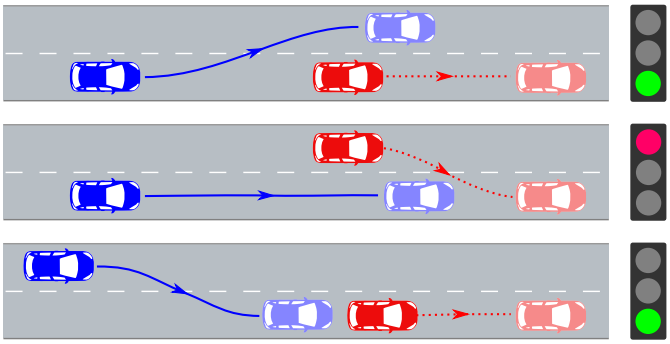


Figure 1: Three exemplary scenario tests for verifying the behavior of an automated vehicle (blue). In the middle scenario, the automated vehicle does not maintain an adequate safety distance, hence the scenario test fails.

Therefore, we propose Test-Driven Inverse Reinforcement Learning (TDIRL), which makes use of existing scenario tests that are often readily available since they are used for verifying and validating the driving behavior in scenario-based testing environments. The idea is to use the scenario test results as feedback for learning the cost function. Thus, there is no additional human effort required to tune the cost function. The contributions of our work are as follows:

- We propose a novel approach for automatically learning a cost function from scenario-based testing outcomes.
- We formulate the approach as a Bayesian inference problem with test outcomes as observations.
- We evaluate our approach in two different environments and show that the learned cost function generalizes well to tests from an unseen validation set.

In the Reinforcement Learning (RL) literature, the objective is formulated as maximizing a reward function instead of minimizing a cost function. For a unified presentation, we will formulate our considerations in terms of cost functions

[1]Institute of Measurement and Control Systems, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
[2]BMWGroup, Unterschleissheim, Germany
[3]Corresponding author: johannes.fischer@kit.edu

throughout this paper. We also want to highlight, that the term *test* in this work does not refer to a hold-out test set as it is typically used to evaluate machine learning models. Instead, it denotes a *scenario test* that is used to verify the behavior of an automated system. This work uses such tests for training cost function parameters as well as for validating the resulting motion planning algorithm. To make this distinction even more clear, we mostly use the term *scenario test*.

II. Related Work

Maximum entropy inverse reinforcement learning is a principled approach to learning cost functions from demonstrations. It selects the cost function that maximizes the policy's entropy while matching the demonstration's feature expectations [6]. In contrast to our approach, which only requires defining scenario-based tests, inverse reinforcement learning approaches typically require collecting numerous high-quality expert driving demonstrations. Furthermore, care has to be taken to collect only high-quality demonstrations whereas our method can use real traffic situations together with scenario-based tests that define the desired behavior. Other works incorporate failure demonstrations in addition to successful demonstrations [17]–[19]. Their objective not only makes the cost function reproduce the expert's behavior but also makes the behavior dissimilar to the failure demonstrations.

Active learning strategies have been proposed to reduce the number of demonstrations by asking a human expert for demonstrations or cost labels in the states where it provides the most information [14], [20]. Similarly, a large body of work investigates preference-based IRL, where the human expert is asked to provide preferences between two trajectories [10]–[12]. The cost inference problem is modeled in a Bayesian framework where the likelihood models the uncertain human preference selection. This has the downside that the human expert needs to be available during the learning process. We also use a Bayesian framework for cost inference, but use the results of automated scenario-based testing as feedback in the likelihood for learning the cost function. Palan *et al.* combined learning from demonstrations and preferences by using demonstrations to infer a prior over the cost function which is then refined by querying the human expert for preferences [13]. Our method can also be combined with a prior learned from demonstrations, as described in Section IV-B3. Another possible source of feedback investigated in the literature is physical human-robot interaction [15], [16], [21]. This also has the downside of requiring a human expert to correct the robot's behavior. Moreover, it is difficult to provide physical corrections in simulation environments.

III. Fundamentals

A. Motion Planning with Optimal Control

Optimal control (OC) is a method frequently used for motion planning in robotic systems [22]. Denoting the system's state by $\mathbf{x}$ and the control input by $\mathbf{u}$, the system dynamics can be described as $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$. The goal of OC is to find a trajectory $\mathbf{x}^*(t)$ and $\mathbf{u}^*(t)$ starting from an initial state $\mathbf{x}_0$ that optimizes an objective function, while satisfying the system dynamics and constraints. The OC problem is often solved by discretizing the time horizon into N time steps which results in a finite dimensional optimization problem

$$\begin{aligned} \min_{\mathbf{x},\mathbf{u}} \quad & \sum_{k=0}^{N-1} J(\mathbf{x}_{k+1}, \mathbf{u}_k) \\ \text{s.t.} \quad & \mathbf{x}_{k+1} = F(\mathbf{x}_k, \mathbf{u}_k), \\ & g(\mathbf{x}_{k+1}, \mathbf{u}_k) \leq 0, \quad \forall k \in \{0, ..., N-1\}. \end{aligned} \tag{1}$$

Here the trajectory is given by the state and control input sequences $\tau = (\mathbf{x}_0, \mathbf{u}_0, \ldots, \mathbf{x}_{N-1}, \mathbf{u}_{N-1}, \mathbf{x}_N)$. The objective is the step cost function J and the discrete-time system dynamics are given by F. They are approximations of the continuous time objective and system dynamics obtained by numerical integration. The constraints g can be used to enforce safety as well as state and input bounds [22].

Model-Predictive Control (MPC) refers to the technique of repeatedly solving the OC problem, whenever a new state measurement $\mathbf{x}$ is available, while only applying the first control input $\mathbf{u}_0$ to the system [23]. Reinforcement learning (RL) algorithms are tailored to the related problem of making decisions in an environment where the system dynamics are unknown and probabilistic [24]. They learn a probabilistic policy $\pi(\mathbf{u}|\mathbf{x})$ by interacting with the environment and observing the resulting cost.

B. Bayesian Inverse Reinforcement Learning

Inverse reinforcement learning (IRL) is the problem of inferring parameters θ of a cost function $\mathcal{J}_\theta$ from expert demonstrations $\mathcal{D}$. However, there are infinitely many cost functions that are consistent with the demonstrations. Maximum entropy IRL solves this ambiguity by choosing the policy with the highest entropy out of all policies that can explain the demonstrations [6].

In contrast, Bayesian IRL (BIRL) solves this ambiguity by modeling the cost function as a random variable and introducing a prior over the cost function [25]. The likelihood $P(\mathcal{D}|\theta)$ is still modeled according to the principle of maximum entropy, while the prior $P(\theta)$ can incorporate additional information on the cost function. The posterior over cost function parameters is then given by

$$P(\theta|\mathcal{D}) = \frac{P(\mathcal{D}|\theta)P(\theta)}{P(\mathcal{D})} \tag{2}$$

$$\propto \exp\left(-\sum_{\tau\in\mathcal{D}} \mathcal{J}_\theta(\tau)\right) P(\theta). \tag{3}$$

IV. Technical approach

Motion planning algorithms for robotic systems typically use a cost function that balances multiple objectives and constraints to ensure safety. Scenario-based testing is a promising approach to validate that the resulting behavior is as desired in certain situations.

To overcome the challenges of hand-tuning the parameters and of obtaining high-quality human demonstrations for inverse reinforcement learning, we propose to use the scenario-based testing outcomes as feedback for the learning process. Our Bayesian approach to Test-Driven Inverse Reinforcement Learning (TDIRL) allows to tune all parameters of the motion planning algorithm.

A. Scenario-based Testing

In the following we describe the aspects of scenario-based testing (SBT) that are necessary for this work. We assume the scenario tests are defined in certain predefined scenarios with multiple tests per scenario. For a detailed description of SBT we refer the reader to prior work [3].

1) Scenario Database: A scenario $S = (\mathbf{x}_0, t_{\text{end}}, \mathcal{O})$ is defined by the vehicle's initial state $\mathbf{x}_0$, including the environment's static objects, the end time t_{end}, and the future trajectories of all dynamic objects $\mathcal{O} = \{o_1, \dots, o_n\}$, that is, $o_i = (\mathbf{x}_0^{(i)}, \dots, \mathbf{x}_{t_{\text{end}}}^{(i)})$ is the future trajectory of object i.

This is based on the assumption that the trajectories of dynamic objects are independent of the vehicles' actions, which is not true in general. However, the scenario tests are typically formulated to validate that the agent behaves in a certain way in a given situation. Nevertheless, our approach can be equally applied to scenarios that do not define the future trajectories of dynamic objects but instead simulate them in the motion planning algorithm.

For a given set of parameters θ of the motion planning algorithm $\mathcal{M}_\theta$, the vehicle's executed trajectory $\tau_S = \mathcal{M}_\theta(S)$ is solely defined by the scenario S, independent of the test that is evaluated on τ_S. It is obtained by rolling out the motion planning algorithm for multiple time steps until reaching the scenario end time t_{end}.

2) Scenario Test Database: The scenario tests we consider are binary tests on the agent's executed trajectory. They are formulated to check if a specific trajectory feature is below or above a certain threshold. For example, a scenario test could check if the vehicle's maximum acceleration is below a certain threshold $a_{\max}$ or if the vehicle's final position is close to the goal position.

Such scenario tests can always be formulated in terms of a trajectory test feature f in the form $f(\tau) \geq 0$. Building on the previous example, a test on the maximum acceleration

$$\max_{t \in [0,T]} \|a(t)\| \overset{!}{\leq} a_{\max}$$

can be encoded in the test feature $f(\tau) = a_{\max} - \max_{t\in[0,T]} \|a(t)\|$.

Hence, a scenario test $T = (S, f)$ consists of a scenario S and a test feature f that is evaluated on the trajectory $\tau_S = \mathcal{M}_\theta(S)$ produced by the motion planning algorithm The scenario test database consists of a set of such tests $\mathcal{T} = \{T_1, \dots, T_m\}$. Typically, multiple tests are formulated for the same scenario. For instance, the same trajectory can be tested for safety, comfort, and final goal distance at the same time.

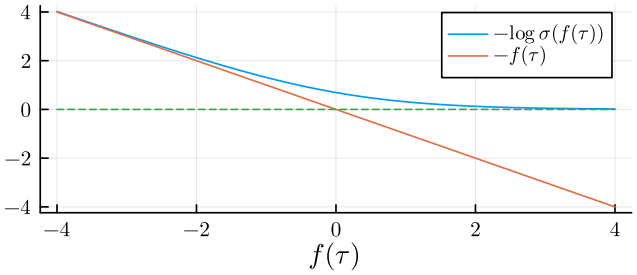


Figure 2: Negative log likelihood of the logit model.

B. Bayesian Approach

We formulate the problem of learning the parameters θ of the motion planning algorithm as a Bayesian inference problem. Starting from Eq. (2), the posterior distribution given the scenario test database $\mathcal{T}$ is $P(\theta|\mathcal{T}) \propto P(\mathcal{T}|\theta)P(\theta)$.

1) Scenario Test Likelihood: Specifying scenario tests is not a straightforward task. It can easily happen that too many tests are defined such that there is no single set of parameters θ that satisfies all tests. To account for this imperfect specification of tests by a human designer, we use the sigmoid function σ to model the success likelihood of a test $T = (S, f)$ with a logistic model as

$$P(T|\theta) = \sigma(f(\mathcal{M}_\theta(S))). \tag{4}$$

This implies that the likelihood for the test success increases logistically with the value of the test feature f.

Assuming independence of scenario test outcomes, the log likelihood for the whole test database $\mathcal{T}$ is given as

$$\log P(\mathcal{T}|\theta) = \sum_{(S,f)\in\mathcal{T}} \log\sigma(f(\mathcal{M}_\theta(S))). \tag{5}$$

This corresponds to the maximum entropy inverse reinforcement learning likelihood in Eq. (3) with a cost of $\mathcal{J}(\tau_S) = -\log\sigma(f(\tau_S))$ for a trajectory $\tau_S = \mathcal{M}_\theta(S)$. This is reasonable since the negative log likelihood of the logistic distribution is monotonically decreasing but almost zero for $x > 0$, as Fig. 2 illustrates. In other words, it is beneficial for the cost if the test features become larger until they are positive. But there is no benefit for further exceeding the test feature with values $f(\tau_S) \gg 0$. If the cost just corresponded to the test features itself, this could lead to some test features not being fulfilled but others exceeding the test threshold and thus compensating the failing tests, as illustrated in the orange curve in Fig. 2. In contrast, using the log likelihood of the logistic model does not incentivize gaining an unnecessary large margin to the scenario test boundary.

2) Adaptive Test Weighting: It is possible that the global optimum of the posterior distribution does not satisfy all scenario tests, even though there is a set of parameters θ that satisfies all scenario tests. For instance, the scenario test features f, which are the unnormalized logits of the logistic model, could be on vastly different orders of magnitude, such that the overall log likelihood $\log P(\mathcal{T}|\theta)$ of the scenario tests

$\mathcal{T}$ is larger if one test is satisfied by a greater margin while other tests fail. To resolve this issue, we use weighted scenario tests $T = (S, f, w)$ with weights $w > 0$ that are initialized uniformly and tuned adaptively: In each iteration of the Markov-Chain Monte Carlo method described in Section IV-B4, we increase the weights of all scenario tests that fail by setting $w \leftarrow \delta \cdot w$ with $\delta > 1$. Then all test weights are normalized such that $\sum_{\mathcal{T}} w = 1$. Those weights are used to optimize the weighted log likelihood

$$\log P(\mathcal{T}|\theta) = \sum_{(S,f,w)\in\mathcal{T}} w \cdot \log \sigma(f(\mathcal{M}_\theta(S))). \tag{6}$$

Thus, the cost for failing a scenario test that is currently failing is increased compared to already successful tests.

3) Learning from Tests and Demonstrations: It is also possible to specify a posterior respecting a set of tests $\mathcal{T}$ and human demonstrations $\mathcal{D}$. We assume that the tests are defined independently of the collection of demonstrations such that the posterior is given by $P(\theta|\mathcal{T}, \mathcal{D}) \propto P(\mathcal{T}|\theta)P(\mathcal{D}|\theta)P(\theta)$. The likelihood of the demonstrations is given by the principle of a maximum entropy trajectory distribution according to Eq. (3). Care needs to be taken to ensure that the scenario tests capture the same target behavior as the demonstrations. Otherwise, the modes of the test likelihood and demonstration likelihood will not align, resulting in a bad optimization problem, for instance, if all demonstrations respect a specific maximum velocity, but the scenario tests require a higher velocity to pass one test.

4) Maximizing the Posterior Likelihood: We use Markov-Chain Monte Carlo (MCMC) methods in a form of policy walk [25] to maximize the posterior likelihood: By sampling the motion planner parameters θ from the posterior distribution $P(\theta|\mathcal{T})$, we are implicitly sampling in policy space. The sampled motion planner is used to recompute the test likelihood. In this work, we employ an adaptive Metropolis-Hastings algorithm to obtain samples from the posterior [26] which, combined with simulated annealing, converge in probability to the mode of the posterior distribution [27]. Simulated annealing uses a temperature ρ to narrow the distribution which is decreased in each iteration. Pseudocode of the algorithm is provided in Algorithm 1.

Algorithm 1 Pseudocode of TDIRL

Require: Scenario tests $\mathcal{T}$, initial parameters θ_0
Require: $\delta > 1, \rho > 0, \sigma^2 > 0, \gamma < 1$
  $\theta \leftarrow \theta_0$ ▷ Parameters
  $w_i \leftarrow 1$ for $i = 1, \dots, m$ ▷ Test weights
  $z \leftarrow \log P(\mathcal{T}|\theta)$ ▷ Log likelihood Eq. (6)
  while not all tests pass **do**
    $V \leftarrow (\theta\sigma)^2$ ▷ Variance of proposal dist.
    $\tilde{\theta} \leftarrow \mathcal{W}_V(\,\cdot\,|\theta)$ ▷ Sample from proposal dist.
    $\tilde{z} \leftarrow \log P(\mathcal{T}|\tilde{\theta})$ ▷ Log likelihood Eq. (6)
    $p_A \leftarrow \min\left(1, \exp\left(\frac{\tilde{z}-z}{\rho}\right) \cdot \frac{\mathcal{W}_V(\theta|\tilde{\theta})}{\mathcal{W}_V(\tilde{\theta}|\theta)}\right)$
      ▷ Adaptive Metropolis-Hastings with annealing
    if accept with prob. p_A **then**
      $\theta \leftarrow \tilde{\theta}, z \leftarrow \tilde{z}$
    end if
    $w_i \leftarrow \delta \cdot w_i$ for failing tests T_i ▷ Section IV-B2
    $w_i \leftarrow w_i / \sum_j w_j$ for $i = 1, \dots, m$ ▷ Normalize
    $\rho \leftarrow \gamma \cdot \rho$ ▷ Decrease annealing temperature
    $\sigma^2 \leftarrow \gamma \cdot \sigma^2$ ▷ Decrease proposal variance
  end while
  return θ ▷ Motion planning parameters

C. Motion Planning Algorithms for TDIRL

So far our treatment was agnostic to the type of motion planning algorithm. The presented principles can be used with any planning algorithm $\mathcal{M}_\theta$ with tunable parameters θ. The agents's trajectory $\tau_S = \mathcal{M}_\theta(S)$ for a given scenario S and parameters θ is the result of repeatedly computing the next control input u and applying it to the system for one time step. Depending on the planner, computing the next control input could require sampling trajectories, solving an optimization problem, or querying a reinforcement learning policy that was re-trained with the parameters θ. In our experiments we focus on motion planning for automated vehicles using optimal control approaches as described in Section III-A. The unknown parameters θ can include the weights of the cost function features, parameters for computing the cost function features, and parameters in the constraints g.

V. Experiment Setup

A. Automated Driving Environments

We evaluate our TDIRL algorithm in two different car following environments. The first environment employs simulated scenarios whereas the second is based on a real-world driving dataset. The motion planning OC problem formulations for both environments are implemented as described in Section III-A using the CasADi framework [28].

1) Description of Simulated Car Following Scenarios: In the simulated scenarios, we consider an ego vehicle following a leading vehicle on a one-lane road. The automated vehicle's state $\mathbf{x} = (x, v, a)$ consists of the longitudinal position x, velocity v, and acceleration a. The system dynamics is given by a triple integrator with the jerk j as the control input $\mathbf{u}$. We obtain concrete scenarios by randomly sampling the initial states $\mathbf{x}_0$ of the ego vehicle and the leading vehicle, and the desired velocity. A smooth and realistic motion for the leading vehicle is generated by sampling its snap profile (derivative of jerk).

The employed cost function is a weighted sum of features, which are functions of the state and control input. The features are listed in Table I and described in more detail in the following and the weights are the parameters $\theta \in \mathbb{R}^4$.

The safety distance feature quadratically penalizes if the distance d to the leading vehicle is smaller than the safety distance d_{safe} but does not penalize if it is larger than the safety distance. This is achieved by using the softplus rectifier function $\varphi(x) = \log(1 + \exp(x))$ as a smooth approximation

Feature	Simulated car following	INTERACTION dataset
Safety distance	$(\varphi(d_{\text{safe}} - d))^2$	$(\varphi(d'_{\text{safe}} - d))^2$
Desired velocity	$(\exp(v - v_{\text{des}}) - 1)^2$	$(v - v_{\text{lead}})^2$
Acceleration	a^2	a^2
Jerk	j^2	j^2
Velocity	-	$v_{\max} - v$

Table I: Cost function features for both environments.

	Parameter values				Test success rate	
	$\theta_1 \cdot 10^3$	θ_2	θ_3	θ_4	Training	Validation
Initial	5.15	5.41	1.71	1.96	47.5%	55%
Learned	4.67	2.15	0.152	0.2	100%	100%

Table II: Parameter values and scenario test success rates for the simulated car following scenario on the training and validation datasets.

of $\max(0, x)$. We use the responsibility-sensitive safety (RSS) model [29] to compute the safety distance d_{safe} based on the current scene.

Deviations from the desired velocity v_{des} are penalized by the desired velocity feature. However, exceeding the desired velocity is penalized to a greater extent than driving slower than the desired velocity. Furthermore, acceleration and jerk are penalized quadratically.

2) Description of INTERACTION Dataset Scenarios: The INTERACTION dataset is a collection of real-world driving data recorded with drones in different traffic situations [30]. For generating concrete scenarios, we restrict ourselves to car following situations in the location `CHN_Merging_ZS` of the dataset. In each scenario, one of the recorded vehicles is considered as the main vehicle. Its leading vehicle is determined from the recording and its motion is obtained from the dataset.

We consider only the longitudinal motion of the vehicles along their recorded path. Since the dataset does not contain jerk measurements, we model the system as a double integrator with the acceleration a as the control input $\mathbf{u}$ and the longitudinal position x and velocity v as the state $\mathbf{x} = (x, v)$.

The cost function features are similar to the ones used in the simulated car following scenario as listed in Table I. One difference is the computation of the safety distance d'_{safe}. Since human drivers often violate the RSS safety distance [31], we treat the safety distance as an additional parameter that is tuned by our method to fulfill safety-related tests.

For real-world data, we do not know the desired velocity of the ego vehicle. For this reason, we adopt a different approach to encode the target velocity in the objective. There is one term that penalizes deviations from the leading vehicle's velocity v_{lead} since in dense traffic, the ego vehicle's velocity is often close to the leading vehicle's velocity. To encourage the vehicle to close the gap to the leading vehicle, there is a linear velocity cost term that penalizes driving slower than the maximum velocity $v_{\max}$ and rewards driving faster. Again, acceleration and jerk are penalized quadratically.

The parameters $\theta \in \mathbb{R}^6$ consist of weights for the five terms in the cost function, in the same order as the features in Table I, plus the safety distance parameter d'_{safe} as the last component. This illustrates that the proposed method can also be used to tune parameters that are not weights of the cost function features but other parameters of the OC problem.

B. MCMC Implementation

Instead of using a prior distribution, we incorporate knowledge on the parameters into the proposal distribution $\mathcal{W}_V(\cdot|\theta)$ with mean θ and variance V to get high quality samples. The parameters are sampled independently of each other, so we describe only the marginal proposal distributions. Since we know that all parameters are positive, we use a Weibull proposal distribution, which has a nonnegative support. We use a scale parameter $\lambda = 1$ and shape parameter of $k = 3.602$, which results in almost zero skewness. To sample more relevant parameters, we shift this proposal distribution such that its mean is the latest sample θ that was accepted by the Metropolis-Hastings criterion. To converge to the mode of the posterior distribution, the standard deviation of the proposal distribution is exponentially decreased with each iteration. Furthermore, we scale the standard deviation with θ, to account for different parameters potentially requiring different orders of magnitude.

VI. Results and Evaluation

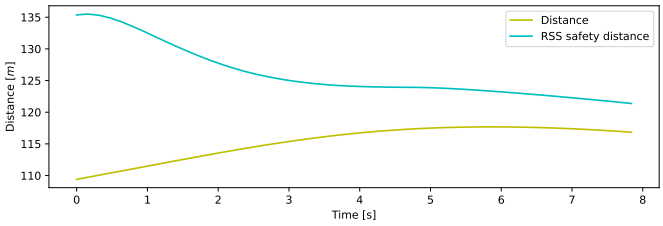
A. Evaluation of Simulated Car Following Scenarios

To evaluate our method in the simulated environment, we manually tuned the cost function parameters to $\theta = (0.01, 4, 0.403, 0.009)$, which reflected reasonable driving behavior. We use these solely to generate a set of 40 expert trajectories, which will be used to determine an initial guess θ_0 for the cost function parameters and to automatically generate scenario tests for the quantitative evaluation in Section VI-A2. The expert trajectories are split evenly into a training and a validation dataset of scenarios.

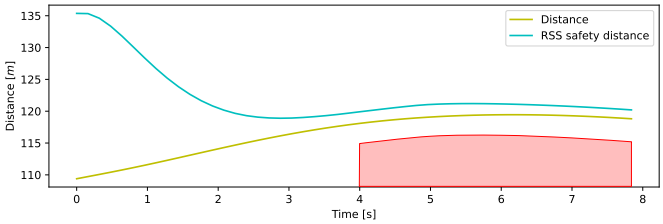
We determine the initial guess θ_0 by computing the feature expectations μ of the expert trajectories from the training dataset and setting θ_0 as the component-wise inverse of μ, making all cost function terms contribute to the total cost within the same order of magnitude.

1) Qualitative Evaluation: To illustrate the immense flexibility for creating scenario tests, we define two tests for the simulated car following scenario, based on observing the resulting behavior of the motion planning algorithm with the initial parameters θ_0.

The first test is in a scenario where the safety distance according to RSS is initially violated, illustrated in Fig. 3a. We formulate the scenario test that the safety distance violation is resolved within the first half of the $8\,\text{s}$ trajectory up to a tolerance of $5\,\text{m}$. This gives the vehicle sufficient time to

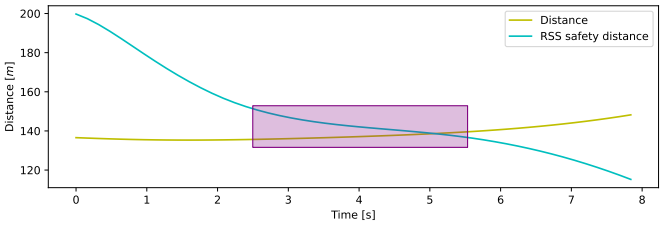


(a) Scenario with inital parameters.

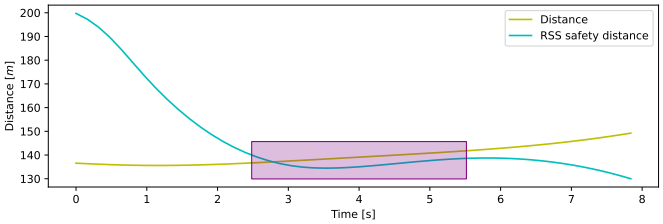


(b) Scenario with learned parameters. The test feature encodes that the distance must not be within the red region, which is the RSS safety distance with a $5\,\mathrm{m}$ tolerance.

Figure 3: Results for the first test scenario where the RSS safety distance is initially heavily violated.



(a) Result with inital parameters.



(b) Result with learned parameters.

Figure 4: Safety distance on a validation scenario. With the learned parameters, the RSS safety distance is fulfilled already after $3\,\mathrm{s}$ instead of after $5\,\mathrm{s}$ as with the initial parameters (highlighted in purple).

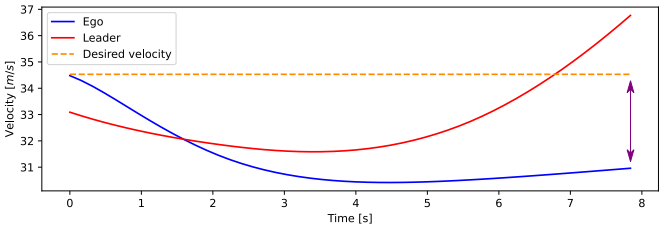
brake to achieve the desired safety distance. With the vehicle distance $d(\mathbf{x}, \mathbf{x}^{\text{lead}})$ and the RSS safety distance $d_{\text{RSS}}(\mathbf{x}, \mathbf{x}^{\text{lead}})$ to the leading vehicle, this test can be formulated with the test feature

$$f(\tau) = \min_{t \in [4\,\mathrm{s}, 8\,\mathrm{s}]} d(\mathbf{x}_t, \mathbf{x}_t^{\text{lead}}) - d_{\text{RSS}}(\mathbf{x}_t, \mathbf{x}_t^{\text{lead}}) + 5\,\mathrm{m}.$$

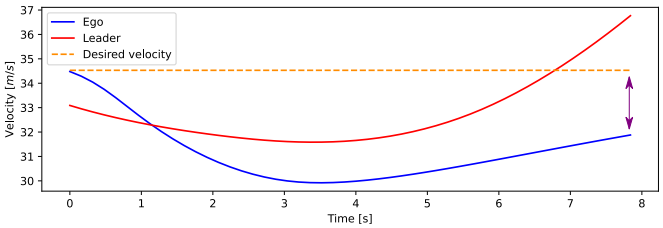
The second test is in a scenario where the leading vehicle is far enough away, but the initial velocity of the ego vehicle is below its target velocity. For this scenario we formulate the test that at the end of the trajectory, the ego vehicle's velocity is within $1\,\frac{\mathrm{m}}{\mathrm{s}}$ of its target velocity. Note that the first test could be easily fulfilled by braking hard. However, the second test requires the ego vehicle to accelerate. To fulfill both tests, the cost function needs to balance the trade-off between maintaining a safe distance and reaching the target velocity, even though the tests are in two different scenarios.

After optimizing the parameters with only those two tests, the resulting weights fulfill both tests. Fig. 3b displays the resulting trajectory in the first test scenario using the learned parameters. The red region illustrates the test feature: The distance between the vehicles in the second half of the trajectory must not be within the red region, which is the RSS safety distance minus a $5\,\mathrm{m}$ tolerance margin.

To show the generalization capabilities of the learned parameters, Figs. 4 and 5 illustrate the effect of using the parameters trained on only these two scenario tests in a third validation scenario. Compared to the behavior with the initial parameters, Fig. 4b shows that the safety distance violation is resolved much earlier by braking. At the same time, Fig. 5b demonstrates that the vehicle also reaches a final velocity that is closer to its desired velocity.



(a) Result with inital parameters.



(b) Result with learned parameters.

Figure 5: Velocity on a validation scenario. The final velocity error is smaller with the learned parameters (highlighted in purple).

2) Quantitative Evaluation: To quantitatively evaluate the performance of our method, we generate scenario tests automatically based on the simulated expert trajectories. For every scenario in the expert trajectories we create one distance-based test and one velocity-based test. For the distance-based test, the final distance to the leading vehicle must be at least as large as the expert trajectory's final distance, up to a tolerance of $1\,\mathrm{m}$. For the velocity-based test, the final velocity must at least match the expert trajectory's final velocity, up to a tol-

erance of $0.5\,\frac{\mathrm{m}}{\mathrm{s}}$. The allowances are added to account for the randomness of the Markov-Chain Monte Carlo method which renders it infeasible to exactly find the expert's equilibrium point between keeping a safe distance and reaching the target velocity.

The parameters θ learned using these scenrio tests are shown in Table II, alongside the scenario test success rate on the training dataset as well as the validation dataset. The test success rate is computed as the fraction of tests that are fulfilled, independent of the test weights or modeled likelihood. For comparison, we also show the scenario test success rate on both datasets using the initial parameters θ_0, which are determined as described in Section VI-A.

B. Evaluation of INTERACTION Dataset Scenarios

It is challenging to evaluate the performance of our method on real data: If we manually specify scenario tests, it would be easy to achieve a high test success rate by specifying the scenario tests for the validation set accordingly. At the same time, failures on manually specified scenario tests from the validation set do not necessarily imply that the method is unsuitable. Instead, it could be the result of impossible requirements in the validation set.

As a consequence, we chose to automatically generate tests based on the human-driven trajectories in the dataset, similar to Section VI-A2. However, since every human driver has a different driving style, it is unlikely that there exists one set of parameters θ such that all generated tests are fulfilled. For this reason, we investigate how well parameters learned from only a very small number of scenario tests generalize to tests on 15 validation scenarios from the INTERACTION dataset. We only use three scenarios from the dataset and generate distance- and velocity-based tests in the same way as described previously in Section VI-A2.

With the parameters θ learned from these in total only six tests, the tests used for training are all satisfied. Additionally, 24 of the 30 automatically generated scenario tests from the validation dataset are fulfilled, resulting in a validation success rate of 80%. Due to the automatic test generation it is unlikely that there exists a parameter set that satisfies all tests. In summary, the automatically generated scenario tests defined on only three training scenarios generalize well to the automatically generated tests in the validation scenarios from the real driving dataset.

VII. Conclusions and Future Work

This work presents Test-Driven Inverse Reinforcement Learning, a method for learning motion planning parameters from scenario-based testing specifications. Our method uses the testing outcomes as feedback in a Bayesian inference problem and learns parameters such that the resulting behavior passes the scenario tests. TDIRL is not limited to the cost function parameters, but also applicable to other parameters of the motion planning algorithm, such as constraint thresholds.

Our evaluation illustrates that TDIRL can be used to learn cost functions from only few test specifications. It is able to generalize well to scenario tests from an unseen validation dataset. Furthermore, it can be used to impose additional safety on real driving data instead of simply imitating the human driver's behavior, as we have shown by defining safety distance tests on the INTERACTION dataset scenarios.

With the growing popularity of scenario-based testing for validation of automated driving functionality, the necessary infrastructure and pipeline are often already available. This allows to integrate the procedure into a continuous integration development pipeline to tune the motion planning parameters without additional human input. Whenever a change in the motion planning algorithm results in a failure of a scenario test, the method can automatically update the algorithm parameters until all tests are passed again. Future work will explore the method with a trajectory planning algorithm developed for a real test vehicle.

Acknowledgements

This work has been supported by the BMWGroup, Germany. The authors also thank the German Research Foundation (DFG) for funding within the priority program "SPP 1835 – Cooperative Interacting Automobiles (CoIn-Car)".

References

[1] D. J. Fagnant and K. Kockelman, "Preparing a nation for autonomous vehicles: Opportunities, barriers and policy recommendations," *Transportation Research Part A: Policy and Practice*, vol. 77, pp. 167–181, Jul. 2015, ISSN: 0965-8564.

[2] S. M. LaValle, *Planning Algorithms*. Cambridge: Cambridge University Press, 2006, ISBN: 978-0-511-54687-7 978-0-521-86205-9.

[3] C. Neurohr, L. Westhofen, T. Henning, T. De Graaff, E. Mohlmann, and E. Bode, "Fundamental Considerations around Scenario-Based Testing for Automated Driving," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, Las Vegas, NV, USA: IEEE, Oct. 2020, pp. 121–127, ISBN: 978-1-72816-673-5.

[4] H. Araujo, M. R. Mousavi, and M. Varshosaz, "Testing, Validation, and Verification of Robotic and Autonomous Systems: A Systematic Review," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 2, pp. 1–61, Apr. 2023, ISSN: 1049-331X, 1557-7392.

[5] A. Y. Ng and S. J. Russell, "Algorithms for Inverse Reinforcement Learning," in *Proceedings of the Seventeenth International Conference on Machine Learning*, ser. ICML '00, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 663–670, ISBN: 1-55860-707-2.

[6] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum Entropy Inverse Reinforcement Learning," in *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, Chicago, Illinois: AAAI Press, 2008, pp. 1433–1438, ISBN: 978-1-57735-368-3.

[7] J. Fischer, C. Eyberg, M. Werling, and M. Lauer, "Sampling-based Inverse Reinforcement Learning Algorithms with Safety Constraints," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Prague, Czech Republic, Sep. 2021, pp. 791–798.

[8] G. Kalweit, M. Huegle, M. Werling, and J. Boedecker, "Deep inverse q-learning with constraints," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 14 291–14 302.

[9] A. Jamgochian, E. Buehrle, J. Fischer, and M. J. Kochenderfer, "SHAIL: Safety-Aware Hierarchical Adversarial Imitation Learning for Autonomous Driving in Urban Environments," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, May 2023, pp. 1530–1536.

[10] E. Biyik, N. Huynh, M. Kochenderfer, and D. Sadigh, "Active Preference-Based Gaussian Process Regression for Reward Learning," in *Robotics: Science and Systems XVI*, Robotics: Science and Systems Foundation, Jul. 2020, ISBN: 978-0-9923747-6-1.

[11] R. Cosner, M. Tucker, A. Taylor, K. Li, T. Molnar, W. Ubelacker, *et al.*, "Safety-Aware Preference-Based Learning for Safety-Critical Control," in *Proceedings of The 4th Annual Learning for Dynamics and Control Conference*, PMLR, May 2022, pp. 1020–1033.

[12] D. Sadigh, A. Dragan, S. Sastry, and S. Seshia, "Active Preference-Based Learning of Reward Functions," in *Robotics: Science and Systems XIII*, Robotics: Science and Systems Foundation, Jul. 2017, ISBN: 978-0-9923747-3-0.

[13] M. Palan, G. Shevchuk, N. Charles Landolfi, and D. Sadigh, "Learning Reward Functions by Integrating Human Demonstrations and Preferences," in *Robotics: Science and Systems XV*, Robotics: Science and Systems Foundation, Jun. 2019, ISBN: 978-0-9923747-5-4.

[14] S. Reddy, A. D. Dragan, S. Levine, S. Legg, and J. Leike, "Learning Human Objectives by Evaluating Hypothetical Behavior," in *Proceedings of the 37th International Conference on Machine Learning*, 2020, p. 10.

[15] A. Bajcsy, D. P. Losey, M. K. O'Malley, and A. D. Dragan, "Learning Robot Objectives from Physical Human Interaction," in *Conference on Robot Learning*, 2017, p. 10.

[16] M. Li, A. Canberk, D. P. Losey, and D. Sadigh, "Learning human objectives from sequences of physical corrections," in *IEEE International Conference on Robotics and Automation, ICRA 2021, Xi'an, China, May 30 - June 5, 2021*, IEEE, 2021, pp. 2877–2883.

[17] K. Shiarlis, J. Messias, and S. Whiteson, "Inverse reinforcement learning from failure," in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, ser. AAMAS '16, Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2016, pp. 1060–1068, ISBN: 978-1-4503-4239-1.

[18] X. Xie, C. Li, C. Zhang, Y. Zhu, and S.-C. Zhu, "Learning Virtual Grasp with Failed Demonstrations via Bayesian Inverse Reinforcement Learning," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Macau, China: IEEE, Nov. 2019, pp. 1812–1817, ISBN: 978-1-72814-004-9.

[19] P. Nagarajan, "Inverse Reinforcement Learning via Ranked and Failed Demonstrations," p. 6,

[20] M. Lopes, F. Melo, and L. Montesano, "Active learning for reward estimation in inverse reinforcement learning," in *Machine Learning and Knowledge Discovery in Databases*, W. Buntine, M. Grobelnik, D. Mladenić, and J. Shawe-Taylor, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 31–46, ISBN: 978-3-642-04174-7.

[21] A. Bajcsy, D. P. Losey, M. K. O'Malley, and A. D. Dragan, "Learning from Physical Human Corrections, One Feature at a Time," in *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, Chicago IL USA: ACM, Feb. 2018, pp. 141–149, ISBN: 978-1-4503-4953-6.

[22] O. Föllinger and G. Roppenecker, *Optimale Regelung und Steuerung: mit 7 Tabellen und 16 Übungsaufgaben mit genauer Darstellung des Lösungsweges*, 3., verb. Aufl, ser. Methoden der Regelungs- und Automatisierungstechnik. München Wien: Oldenbourg, 1994, ISBN: 978-3-486-23116-8.

[23] C. E. García, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—A survey," *Automatica*, vol. 25, no. 3, pp. 335–348, May 1989, ISSN: 0005-1098.

[24] R. S. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, Second edition, ser. Adaptive Computation and Machine Learning. Cambridge, MA London: The MIT Press, 2018, ISBN: 978-0-262-03924-6.

[25] D. Ramachandran and E. Amir, "Bayesian Inverse Reinforcement Learning," in *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, ser. IJCAI'07, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007, pp. 2586–2591.

[26] H. Haario, E. Saksman, and J. Tamminen, "An Adaptive Metropolis Algorithm," *Bernoulli*, vol. 7, no. 2, p. 223, Apr. 2001, ISSN: 13507265. JSTOR: 3318737.

[27] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan, "An introduction to MCMC for machine learning," *Machine Learning*, vol. 50, pp. 5–43, 2003.

[28] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, Mar. 2019, ISSN: 1867-2949, 1867-2957.

[29] S. Shalev-Shwartz, S. Shammah, and A. Shashua, *On a Formal Model of Safe and Scalable Self-driving Cars*, Oct. 2018. arXiv: 1708.06374 [cs, stat].

[30] W. Zhan, L. Sun, D. Wang, H. Shi, A. Clausse, M. Naumann, *et al.*, "INTERACTION Dataset: An INTERnational, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps," *arXiv:1910.03088 [cs, eess]*, 2019. arXiv: 1910.03088 [cs, eess].

[31] M. Naumann, F. Wirth, F. Oboril, K.-.-U. Scholl, M. S. Elli, I. Alvarez, *et al.*, "On Responsibility Sensitive Safety in Car-following Situations - A Parameter Analysis on German Highways," in *2021 IEEE Intelligent Vehicles Symposium (IV)*, Jul. 2021, pp. 83–90.