Modeling Driver Behavior From Demonstrations in Dynamic Environments Using Spatiotemporal Lattices

David Sierra González[1], Ozgur Erkent[1], Víctor Romero-Cano[2], Jilles Dibangoye[1], and Christian Laugier[1]

Abstract— **One of the most challenging tasks in the development of path planners for intelligent vehicles is the design of the cost function that models the desired behavior of the vehicle. While this task has been traditionally accomplished by hand-tuning the model parameters, recent approaches propose to learn the model automatically from demonstrated driving data using Inverse Reinforcement Learning (IRL). To determine if the model has correctly captured the demonstrated behavior, most IRL methods require obtaining a policy by solving the forward control problem repetitively. Calculating the full policy is a costly task in continuous or large domains and thus often approximated by finding a single trajectory using traditional path-planning techniques. In this work, we propose to find such a trajectory using a conformal spatiotemporal state lattice, which offers two main advantages. First, by conforming the lattice to the environment, the search is focused only on feasible motions for the robot, saving computational power. And second, by considering time as part of the state, the trajectory is optimized with respect to the motion of the dynamic obstacles in the scene. As a consequence, the resulting trajectory can be used for the model assessment. We show how the proposed IRL framework can successfully handle highly dynamic environments by modeling the highway tactical driving task from demonstrated driving data gathered with an instrumented vehicle.**

I. Introduction

In reinforcement learning, the goal is to find a policy that maximizes the accumulated long-term reward [1], [2]. Consequently, the behavior of an agent for a given task is implicitly encoded by the reward function, i.e. given the reward function and the agent's dynamics, the optimal policy is determined. Reward shaping techniques focus on the design of reward functions that can guide and scale up the search process of an exploring agent [3], [4]. However, specifying an appropriate reward function for complex, high-dimensional robotic problems can be extremely challenging, and reward shaping techniques can lead to bugs in which the reinforcement learning algorithm exploits the reward function in ways not anticipated by the designer [5].

Instead of relying on hand-engineered reward functions, the desired behavior of an agent in a given task can be specified by providing expert demonstrations of the task [6]. While it is possible to extract policies directly from the demonstrations using imitation learning techniques, the resulting policies tend to perform poorly when the agent diverges too much from the demonstrations [7]. It is commonly accepted that in reinforcement learning "the reward function, rather than the policy or the value function, is the most succinct, robust, and transferable definition of a task" [8], [9]. In consequence, many imitation learning approaches improve generalization by first learning the reward function using Inverse Reinforcement Learning (IRL) [9], [10], and then using it optimize the policy [8]. This has led to some impressive results in real-world robotics problems such as autonomous helicopter flight control [11], human activity forecasting [12], human-like parking lot navigation [13] and robotic manipulation tasks [14].

The design of cost functions to model the behavior of drivers is a common topic in the Intelligent Vehicles research domain (note that costs can be seen as negative rewards). Traditionally, the process involves the design of each component of the model using background knowledge [15], as well as finding a balance between components that appropriately captures the desired behavior. This last step is typically addressed by hand-tuning the model parameters [15], [16], [17], which becomes an increasingly difficult task as the complexity of the model grows. Moreover, different scenarios usually require different behavioral models, making the hand-tuning a recurring operation. To avoid these problems, recent efforts learn the model automatically from demonstrated driving data using IRL for applications as diverse as outdoor autonomous navigation [18], risk anticipation on residential roads [19], highway maneuver prediction [20] and communicative human-aware autonomous driving [21].

Traditional IRL approaches require solving the forward control problem to obtain a policy in the inner loop of an iterative optimization algorithm. A similarity metric between the demonstrations and the policy is then computed and used to determine if an appropriate model has been found. In continuous domains, such as motion planning for autonomous vehicles, finding the optimal policy is intractable. Existing approximations propose to solve instead a local control problem [22], construct a graph-based representation of the state-space [23], [24] or find instead a single trajectory using path-planning techniques such as A* [25], spline optimization [26], Rapidly-exploring Random Trees [27], or Gaussian processes [28]. In dynamic environments, even finding a single trajectory can become a challenging task. As a consequence, in the particular case of driver behavior modeling most existing approaches focus on static environments [19], [29], [27].

In this work, we propose to integrate the IRL paradigm with a motion planner based on conformal spatiotemporal

This work was supported by Toyota Motor Europe.

[1]The authors are with Inria Rhône-Alpes, Grenoble, France {david.sierra-gonzalez, ozgur.erkent, jilles.dibangoye, christian.laugier}@inria.fr

[2] Víctor Romero-Cano is with Universidad Autónoma de Occidente, Cali, Colombia varomero@uao.edu.co.

state lattices [30]. State lattices provide a search space of dynamically feasible actions that can additionally be conformed to the structured environment of public roadways [31]. This has the immediate advantages of limiting the size of the search space and producing trajectories that satisfy the kinematic constraints of the vehicle. By using the spatiotemporal variant, which includes time in the state-space, we sacrifice the optimality of the solution in exchange for increased awareness about the surrounding traffic. We hypothesize that this planning variant resembles more closely the behavior of human drivers and that in consequence, the resulting trajectories constitute a better evaluation metric for the model. The proposed approach is experimentally validated by successfully modeling the highway driving task from suboptimal driving demonstrations gathered with an instrumented vehicle.

The rest of the paper is organized as follows: Section II provides the necessary background about IRL and state lattices; Section III details the proposed modeling framework; Section IV describes our experimental setup and shows the qualitative and quantitative experimental results. Lastly, Section V concludes.

II. Background

In this section, we briefly describe the Markov Decision Process (MDP) formulation of IRL, the problematic of applying traditional IRL approaches in dynamic environments, and the path planning approach based on spatiotemporal state lattices that we use to circumvent that problem.

A. Inverse Reinforcement Learning

Generally speaking, the goal of IRL is to undercover the preferences behind exhibited behaviors [9]. The task is typically framed in the MDP framework. An MDP is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, \{P_{sa}\}, \mathcal{C}, \gamma \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces respectively, $P_{sa}(\cdot)$ is the state transition probability upon taking action a in state s, $\mathcal{C} : \mathcal{S} \mapsto \mathbb{R}$ is the cost function, and $\gamma \in [0, 1)$ is the discount factor. For real-world driving tasks, the state and action spaces are very large, i.e. $\mathcal{S} \doteq \mathbb{R}^n$ and $\mathcal{A} \doteq \{g | g : \mathcal{S} \mapsto \mathcal{S}\}$. In this work, we make use of a state-lattice to compactly represent the underlying state and action spaces by exploiting the environment and non-holonomic motion constraints of the vehicle. We provide further details in subsection II-B.

A policy is defined as any map $\pi : \mathcal{S} \to \mathcal{A}$. The value function for a policy π, can be evaluated at any arbitrary state $s \in \mathcal{S}$ as follows:

$$V^{\pi}(s) = E[\sum_{t=0}^{\infty} \gamma^t \mathcal{C}(S_t) \mid \pi, S_0 = s] \tag{1}$$

Solving an MDP implies finding the policy π that minimizes V^{π}, that is, for any arbitrary state $s \in \mathcal{S}$ we have that:

$$\begin{aligned} V^*(s) &= \min_{\pi} \; V^{\pi}(s) \\ &= \mathcal{C}(s) + \min_{a \in \mathcal{A}} \; E_{s' \sim P_{sa}(\cdot)}[V^*(s')] \end{aligned} \tag{2}$$

where notation $s' \sim P_{sa}(\cdot)$ means the expectation is with respect to s' distributed according to $P_{sa}(\cdot)$.

IRL addresses the case where the cost function $\mathcal{C}$ is not known, but instead we have access to a set of expert, possibly suboptimal, demonstrated trajectories $\mathcal{D} = \{\xi^{[1]}, \xi^{[2]}, \ldots, \xi^{[M]}\}$. A trajectory is defined as a sequence of states of a vehicle $\xi = \{s_1, \ldots, s_T\}$. The majority of IRL algorithms assume that the cost function can be fully specified as a linear combination of features:

$$\mathcal{C}(s) = \mathbf{w}^T \mathbf{f}(s) \tag{3}$$

where $\mathbf{w} = (w_1, w_2, \ldots, w_K)$ is the unknown weight vector and $\mathbf{f}(s) = (f_1(s), \ldots, f_K(s))$ is the feature vector that parameterizes state s, both of dimension K. We shall use short-hand notation $\mathbf{f}_{\xi}$ to denote the sum of features along any arbitrary trajectory ξ:

$$\mathbf{f}_{\xi} \doteq \left[\sum_{t=1}^{T} \mathbf{f}(s_t) | \xi = (s_1, \ldots, s_T) \right] \tag{4}$$

The goal of IRL is then to find the weight parameters $\mathbf{w}$ for which the optimal policy, obtained by solving the corresponding planning problem, would achieve similar trajectories to those demonstrated according to a given statistic. Abbeel and Ng propose to use the feature expectations as the similarity metric [8]. Unfortunately, this is an ill-posed problem, as many different weights can make the demonstrated behavior optimal. An additional complication is that, in real-world applications, the demonstrations are typically suboptimal.

The Maximum Entropy IRL framework addresses both problems by modeling expert behavior as a distribution over trajectories and applying the principle of Maximum Entropy to select the one that does not exhibit any additional preferences beyond matching feature expectations [32]. Under this model, the probability of a trajectory is proportional to the negative exponential of the costs encountered along the trajectory:

$$P_{\mathbf{w}}(\xi) = \frac{1}{Z(\mathbf{w})} \exp(-\mathbf{w}^T \mathbf{f}_{\xi}) \tag{5}$$

Let $L(\mathbf{w})$ be the likelihood function of $\mathbf{w}$ given observed trajectories. To find the parameter vector $\mathbf{w}^*$ that optimizes $L(\mathbf{w})$, one can rely on its gradient:

$$\nabla L(\mathbf{w}) = \tilde{\mathbf{f}} - \mathbb{E}_{\xi \sim P_{\mathbf{w}}(\cdot)}[\mathbf{f}_{\xi}] = \tilde{\mathbf{f}} - \sum_{s \in \mathcal{S}} \rho_{\mathbf{w}}(s) \mathbf{f}(s) \tag{6}$$

where $\tilde{\mathbf{f}} \doteq \frac{1}{M} \sum_{i=1}^{M} \mathbf{f}_{\xi^{[i]}}$ represents the average sum of features of the demonstrations and $\rho_{\mathbf{w}}$ denotes the expected state visitation frequencies under parameter vector $\mathbf{w}$.

To obtain the expectation in Eq. 6, the calculation of the full policy is required, which is not tractable in continuous or large domains. A common workaround is to compute, at each optimization step, a single trajectory instead. This can be done for instance using A* [25] or RRTs [27]. In all these cases, convergence guarantees are sacrificed for tractability since the likelihood function is no longer convex. In dynamic

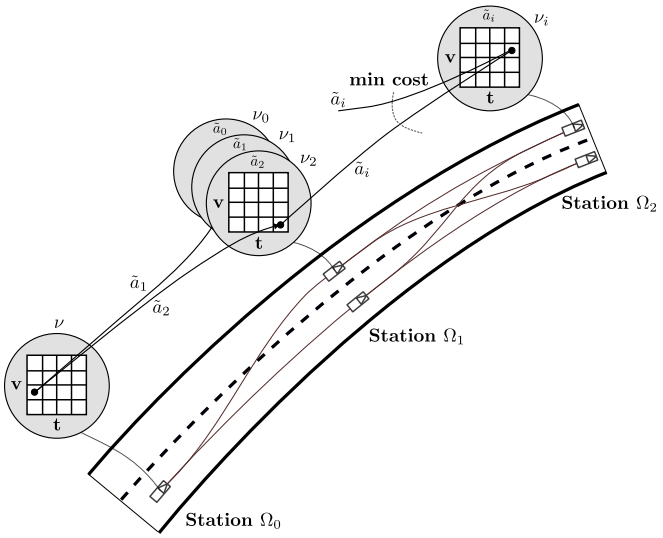


Fig. 1: Structure of the spatiotemporal state lattice. Road-aligned states are sampled at regular longitudinal intervals on each lane and connected with feasible paths using parametric optimal control. The spatiotemporal planner searches for the best trajectory in the lattice and keeps the problem tractable by pruning the search space [30].

environments, finding a trajectory that minimizes the cost can be particularly challenging. In this paper, we propose to use a conformal spatiotemporal lattice, which is described next.

B. Conformal Spatiotemporal State Lattices

State lattices provide a concise representation of continuous state-spaces using a discrete search graph of dynamically attainable states [33]. They are a very convenient approach for path planning in structured environments as they can be conformed to the environment by applying a state-based sampling strategy [31]. Sampled states can then be connected using an inverse path generator [34]. A path, denoted τ, is defined as a continuous curve through a 4-dimensional state-space $[x, y, \theta, \kappa]$ connecting two endpoint states; x represents the longitudinal position along the road, y the lateral position, θ the heading, and κ the curvature or rate of change of the heading. In Fig. 1 we show the paths that connect sampled feasible vehicle states along the road.

The state lattice graph is built over the states of the MDP, which is an idea already explored in the RL literature [35], [36], [24]. The state space of the underlying MDP is 7-dimensional. Each state s is a tuple $(x, y, \theta, \kappa, t, v, \tilde{a})$, where t denotes time, v velocity, and $\tilde{a}$ acceleration. The nodes of the graph are samples from the state space. The edges correspond to the actions. An edge (or local trajectory) represents the traversal of the path between two nodes with a given acceleration value, i.e. $e_j : [t_j^s, t_j^e] \mapsto \mathbb{R}^6$ with $j \in \{1, 2, \ldots, E\}$, and where the number of edges E is determined by the number of paths in the lattice and the number of acceleration actions available. Within this lattice framework, a full trajectory ξ is a sequence of edges from a start to a goal state. Due to the different lengths and durations of the edges, the lattice graph represents a deterministic Semi-Markov Decision Process (SMDP). We will refer to this deterministic SMDP as a graph-MDP. The cost of traversing an edge e_j is then given by

$$c(e_j) \doteq \int_{t_j^s}^{t_j^e} \gamma^{t-t_j^s} \mathcal{C}(s_t = e_j(t)) dt \tag{7}$$

In practice, this integral can be approximated by sampling states along the edge.

To handle highly-dynamic environments, we have considered time as part of the state-space. This renders the exact solution of graph-MDP using traditional dynamic programming methods intractable. Instead, we solve the path planning problem using the approach presented by McNaughton et al. [30], in which a pruning strategy is applied to keep the size of the state-space within feasible limits. We provide here a summary of the approach and refer the reader to the original publication for a detailed description.

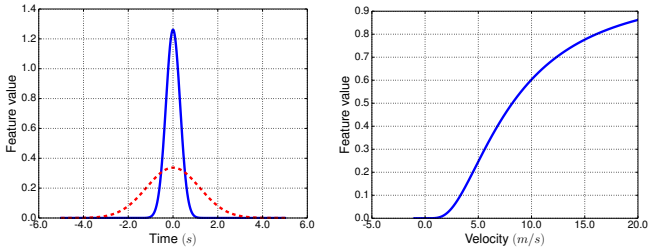
The key idea of this approach in order to attain tractability is to specify the time-dependent components of the graph (that is, the velocity and time components of the nodes) online as the search proceeds. In other words, the search graph is dynamically built during the search. Let us define a vertex ν as the set of all nodes with the same $[x, y, \theta, \kappa, \tilde{a}]$ components. Additionally, a station Ω is defined as the set of all vertices whose states' x component is equal. For all vertices, a discrete range of threshold times and velocities is specified, determining a grid-like structure (see Fig. 1). For each vertex, only a single node is allowed to exist per cell, which efficiently prunes the search space.

The search for the best trajectory is based on maintaining the minimum cost $\hat{c} : \mathcal{S} \mapsto \mathbb{R}$ that requires traveling from the start node s_0 to any other node in the graph. Initially, this value is set to infinity for all nodes other than the start node. The search begins from the start node and proceeds recursively along the stations, as specified in Algorithm 1. For each node in a given station, the outgoing edges (which are determined by the paths of the lattice and the allowed

Algorithm 1: Spatiotemporal trajectory search

input : $\mathbf{w}$, T, $\tilde{\mathbf{a}}$, s_0
output: Best trajectory found ξ^*

```
SpatiotemporalSearch
    foreach station Ω_i do
        foreach vertex ν_j do
            foreach node s_k do
                if ĉ(s_k) ≠ ∞ then
                    foreach ã ∈ ã do
                        Construct edge e
                        Determine edge end node s_end
                        Evaluate c(e) using (7)
                        Get node s_old at cell of arrival
                        if ĉ(s_k) + c(e) < ĉ(s_old) then
                            s_old ← s_end
    Find final node s* = arg min_s ĉ(s) s.t. s[t] ≥ T
    return backtrace trajectory ξ*
```



(a) TTC and TH features (b) Desired velocity deviation feature

Fig. 2: Continuous features selected for the time-to-collision, time-headway, and desired velocity deviation signals.

accelerations $\tilde{\mathbf{a}}$) are created and evaluated. The acceleration followed during each edge determines the vertex to which the end node is associated. The time and velocity components of the end node determine the cell within the vertex to which it is assigned. If another node (s_{old} in Algorithm 1) had already been assigned to the same cell, the one with the lowest accumulated cost between the two is kept. The search continues until the final station is reached. In general, one of the challenges of searching with this spatiotemporal lattice is selecting the final node. For our IRL learning purpose this becomes a trivial task: we select the node with the lowest accumulated cost and with an associated time beyond the threshold time T, which is given by the duration of the corresponding demonstrated trajectory. We detail this in the next section. The best trajectory ξ^* is found by backtracking until the initial node.

III. Method

In this section, we detail how to put together the MaxEnt IRL framework with the spatiotemporal lattice planner in order to learn driver behaviors from demonstrations in dynamic environments. In the first place, we will list the features that we have selected to compose the cost function. Then, we describe the proposed algorithm.

A. Feature selection

In this paper, we evaluate our approach on highway driving data gathered with an instrumented vehicle. The goal is to find the model that best captures the demonstrated behavior. The capacity of the model to capture any behavior will always be constrained by the composing features and the shape of the function. The following features have been selected using background knowledge:

a) Lane: this is a binary mutually-exclusive feature aimed to model the preference of the driver to stay on a given lane. Typically, a driver will remain on the right-most lane unless an overtake needs to be performed.

b) Time-to-collision (TTC): this is an indicator of highly dangerous states. It is defined as the remaining time until a collision occurs if two vehicles continue on the same course and at the same speed. We model it with two Gaussian distributions: a narrow one that aims to capture the high cost associated to low TTC values, and a wider one that represents mildly dangerous states. This feature is considered both for the front and the back in the lane of the corresponding state of the vehicle, accounting for a total of four features. The one-dimensional Gaussians used are shown in Fig. 2.

c) Time-headway (TH): the time-headway indicates potentially dangerous situations. It is defined as the time elapsed between the back of the lead vehicle passing a point and the front of the following vehicle passing the same point. It can be seen as the equivalent of distance when driving at high speeds. We model it similarly as the TTC.

d) Deviation from desired speed: this feature models the cost that is paid when the vehicle is forced to deviate from the desired speed, which we set to the speed limit. We model it with the cumulative distribution function of the inverse-gamma. For low deviations, the feature is not activated and no cost is paid. Beyond a given threshold, the value of the feature rises sharply and with it the associated cost. This should force the vehicle to overtake if it is safe.

e) Acceleration: we consider the absolute value of the acceleration as a feature to discourage strong accelerations or decelerations.

f) Distance: the distance traveled along the road is considered as a feature with a negative weight value in order to encourage the vehicle to make progress.

Algorithm 2: IRL with Spatiotemporal State Lattices for Driver Modeling in Dynamic Environments

input : $\mathcal{D}$, $\mathbf{w}_0$, δ, $\tilde{\mathbf{a}}$
output: optimal weights $\mathbf{w}^*$

$\mathbf{w} \leftarrow \mathbf{w}_0$
while *not converged* **do**
  $\Xi_{\tilde{\mathbf{f}}} \leftarrow 0$, $\Xi_{\mathbf{f}_{\xi^*}} \leftarrow 0$
  foreach $\xi_i \in \mathcal{D}$ **do**
    Determine road waypoints
    Construct state lattice (Fig. 1)
    $\Xi_{\tilde{\mathbf{f}}} \leftarrow \Xi_{\tilde{\mathbf{f}}} + \mathbf{f}_{\xi^{[i]}}$
    $\xi^* \leftarrow \text{SpatiotemporalSearch}(\mathbf{w}, T(\xi^{[i]}), \tilde{\mathbf{a}}, s_0(\xi^{[i]}))$
    $\Xi_{\mathbf{f}_{\xi^*}} \leftarrow \Xi_{\mathbf{f}_{\xi^*}} + \mathbf{f}_{\xi^*}$
  $\nabla_{\mathbf{w}} \leftarrow \frac{1}{M}(\Xi_{\tilde{\mathbf{f}}} - \Xi_{\mathbf{f}_{\xi^*}})$
  $\mathbf{w} \leftarrow \mathbf{w} - \delta\nabla_{\mathbf{w}}$
return $\mathbf{w}$

B. Inverse Reinforcement Learning using spatiotemporal state lattices

As we have introduced in subsection II-A, we approximate the feature expectations in Eq. 6 by the features associated to a single trajectory found using the spatiotemporal planner. This enables us to calculate the approximated gradient and perform gradient-based optimization.

The whole process is described in Algorithm 2. We set the initial weights $\mathbf{w}_0$ as an input to the algorithm, as they can be set using background knowledge. Additionally, a set of discrete valid acceleration actions $\tilde{\mathbf{a}}$ and the optimization step-size δ need to be specified. Until convergence, the algorithm repeatedly iterates over the demonstrated trajectories $\mathcal{D}$ to optimize the weight vector $\mathbf{w}$. For each demonstration,

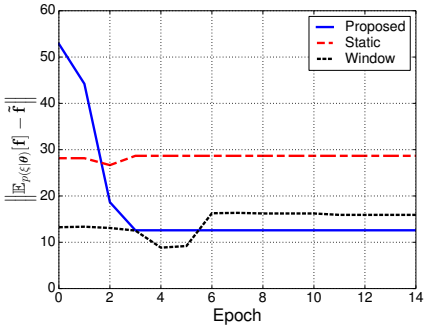


Fig. 3: Convergence of the different learning algorithms considered.

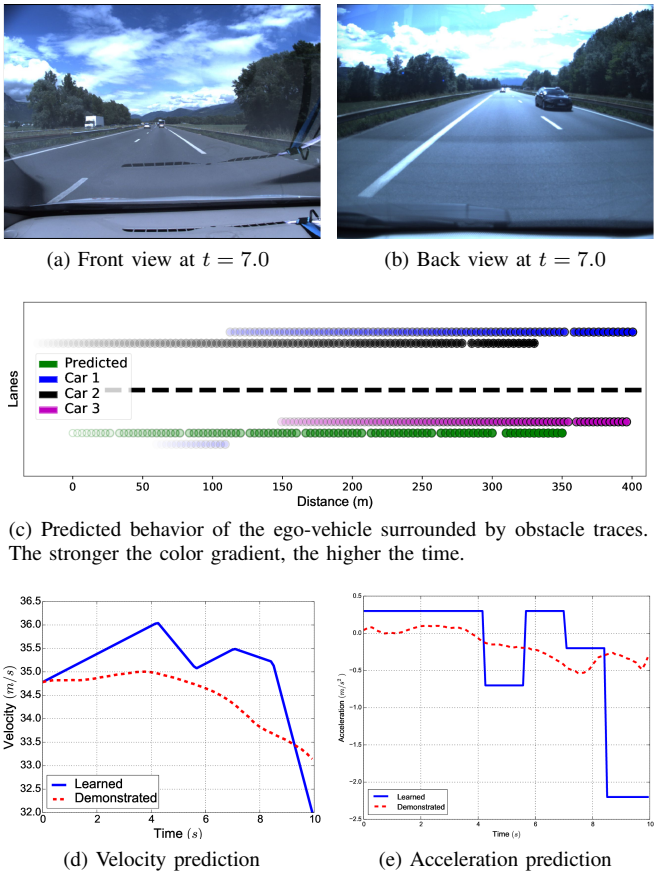
a state lattice is deployed adapted to the shape of the road. Starting from the initial state of the demonstrated trajectory, and with the threshold time set to the last timestamp in the demonstration, we obtain a trajectory ξ^* with Algorithm 1. This trajectory is obtained in an obstacle-aware manner and, in consequence, it can be used in the evaluation of the current estimate of the model. The sum of features of the demonstrations and of the planned trajectories are aggregated in the container variables $\Xi_{\tilde{\mathbf{f}}}$ and $\Xi_{\mathbf{f}_{\xi^*}}$ respectively. After iterating over each epoch, the average gradient is calculated and used to update the current estimate of the weights $\mathbf{w}$.

IV. Experimental evaluation

We evaluate our driver modeling approach based on IRL with spatiotemporal state lattices using demonstrated driving data gathered on a French two-lane highway. The goal is to capture the demonstrated behavior in the model. We first present the dataset used and provide details about the tested lattice configuration; then we discuss the evaluation metrics and competing approaches against we compare; finally, we present the qualitative and quantitative results obtained.

A. Dataset and lattice configuration

As training data, we use $170s$ of annotated highway driving data. It consists of a variety of traffic scenes with an average duration of $10s$, including lane changes in both directions, getting stuck behind slow traffic and free driving. The tracking of the obstacles was performed with a grid-based tracker [37], but only the lane in which they circulate was annotated, i.e. there is no lateral in-lane position information. This is also true for the ego-vehicle performing the demonstrations. This motivated us to setup the lattice as seen in Fig. 1, with the nodes of the lattice always at the center of the lanes. The longitudinal separation between stations was set to $50m$, the allowed accelerations were 9 equidistant discrete values in the range [-2, 2] m/s^2. The threshold values that determine the composition of the grid of subnodes were set to 6 equidistant values in the interval [25, 38] m/s for the velocity, and the range between $0.5s$ and the maximum timestamp in the demonstration sampled every $0.5s$ for the time dimension. This defines a much finer grid than the one proposed in the original paper [30].



(a) Front view at $t = 7.0$

(b) Back view at $t = 7.0$

(c) Predicted behavior of the ego-vehicle surrounded by obstacle traces. The stronger the color gradient, the higher the time.

(d) Velocity prediction

(e) Acceleration prediction

Fig. 4: Prediction of behavior based on the optimized model.

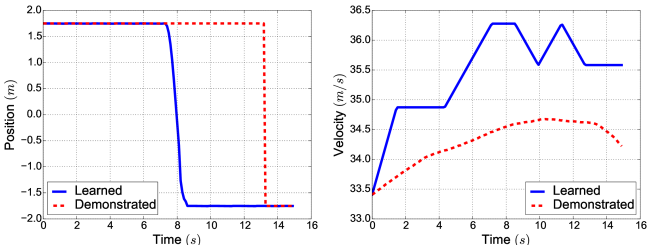
B. Metrics and competing approaches

The metric that we use is the modified Hausdorff distance (MHD), which is a generalized distance metric between point sequences and typically used for this kind of evaluation [12], [38], [29]. Given the lack of a ground truth cost function, we follow previous work and additionally provide the values of task-specific metrics obtained by solving the planning problem with the optimized model. We consider average speed, average acceleration and the total time spent on each lane. Regarding the competing approaches, we consider solving the planning problem by examining only the state of the environment at the beginning of the demonstration, and a slightly more refined approach that uses a sliding window over each demonstration. The optimization takes place at this time window and the environment is assumed static as in the first competing approach. To compensate for this assumption, the optimization is performed on overlapping time windows. This approach has been used in the context of social robotic navigation [39], [40]. We set the length of the time window to $5s$, with overlapping of $2.5s$ between optimization cycles.

C. Qualitative results

In the first place, we show the convergence of the algorithms in Fig. 3. The proposed approach converges smoothly

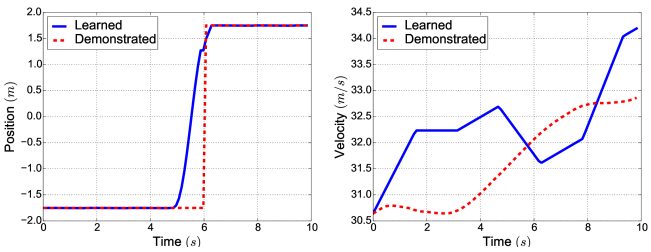
(a) Back view at $t = 8.0$ (b) Back view at $t = 13.0$



(c) Position prediction (d) Velocity prediction

Fig. 5: Prediction of behavior for a right-merge scene.

(a) Front view at $t = 5.0$ (b) Back view at $t = 5.0$



(c) Position prediction (d) Velocity prediction

Fig. 6: Prediction of overtaking behavior.

TABLE I: Performance statistics

METHOD	MHD	Avg speed (m/s)	Avg accel. (m/s^2)	t left (s)	t right (s)
Static	0.031	30.67	-0.026	36.46	108.98
Window	0.145	25.04	-1.024	65.83	79.61
Proposed	**0.016**	**32.88**	**0.045**	**59.86**	**85.58**
Ground truth	-	31.93	0.019	51.55	93.89

from an initial high deviation between the empirical features and the approximated expected sum of features. The initial low value for the window approach is probably due to the shorter length of the "windowed" demonstrations. As it can be seen, the norm of the proposed approach does not converge to zero. This is due to the high suboptimality of the demonstrations and also to their long distances. One of the features considered in our model is the deviation from the desired velocity, which we assumed to be the speed limit. We can see for instance in Fig. 5d, that the demonstrated velocity does not attempt to reach the speed limit of $36.1m/s$, despite not having any obstacles in front (not seen in the figure).

The results shown in Figs. 4-6 correspond to roughly $150s$ of demonstrated driving data that were used for testing the resulting optimized model. We used the spatiotemporal state lattice planner to produce a trajectory based on the optimized model and we compare it against the demonstrated data. Fig. 4 shows the results for a scene in which the ego-vehicle approaches slow traffic ahead and it is not safe to perform a lane change due to the presence of a vehicle on lane 2. The predicted behavior is a deceleration in order to reduce the relative velocity with the traffic ahead, which matches with the demonstrated behavior. Fig. 5, consists of a merge to the right lane after an overtake. The model correctly captures that, in the absence of obstacles, the demonstrations suggested driving on the right-most lane. In a similar fashion, Fig. 6 shows that the predicted behavior when approaching slow traffic ahead with no vehicles blocking the left lane is an overtake. In this case the lane change trajectory is preferred because it allows an acceleration towards the speed limit, which in turn maximizes the traveled distance.

D. Qualitative results

Table I shows the performance metrics obtained on the roughly $150s$ of evaluation driving data. The proposed approach seems to clearly have captured the demonstrated behavior better than the "static" and "window" approaches: it obtains a lower average MHD and resembles the ground truth more closely in the task-specific metrics.

V. Conclusions

We have presented a driver modeling algorithm based on a combination of Maximum Entropy IRL and a spatiotemporal state lattice motion planner. The proposed approach is suitable for learning behavior models from demonstrations in highly dynamic environments. We validated our proposal using demonstrated suboptimal highway driving data gathered with an instrumented vehicle. Despite having been trained with general driving data containing all kinds of situations such as merges to the right, overtakes, getting stuck behind slow traffic and free driving, the model successfully captured the main characteristics of the demonstrated behavior. We show this result in both quantitative and qualitative experiments. Future work will study the effect of the sampling strategy on the quality of the model, explore the use of non-linear models, and also address stop-and-go scenarios by extending the original spatiotemporal state lattice formulation.

ACKNOWLEDGMENT

We thank Nicolas Vignard, Jean-Alix David and Jérôme Lussereau for their assistance with the experimental vehicle during the data collection.

REFERENCES

[1] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-dynamic programming*. Athena Scientific, 1996, vol. 3.

[2] R. S. Sutton and A. G. Barto, *Reinforcement learning: An Introduction*, ser. Adaptive computation and machine learning. MIT Press, 1998.

[3] J. Randløv and P. Alstrøm, "Learning to Drive a Bicycle Using Reinforcement Learning and Shaping," in *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998), Madison, Wisconsin, USA, July 24-27, 1998*, 1998, pp. 463–471.

[4] A. Y. Ng, D. Harada, and S. J. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999), Bled, Slovenia, June 27 - 30, 1999*, 1999, pp. 278–287.

[5] J. Kober, J. A. D. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," July 2013.

[6] B. Argall, S. Chernova, M. M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.

[7] D. A. Pomerleau, "Efficient Training of Artificial Neural Networks for Autonomous Navigation," *Neural Computation*, vol. 3, no. 1, pp. 88–97, March 1991.

[8] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*, 2004.

[9] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning," in *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*, 2000, pp. 663–670.

[10] S. J. Russell, "Learning agents for uncertain environments (extended abstract)," in *Proceedings of the Eleventh Annual Conference on Computational Learning Theory, COLT 1998, Madison, Wisconsin, USA, July 24-26, 1998.*, 1998, pp. 101–103.

[11] P. Abbeel, A. Coates, and A. Y. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *I. J. Robotics Res.*, vol. 29, no. 13, pp. 1608–1639, 2010.

[12] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert, "Activity forecasting," in *Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part IV*, 2012, pp. 201–214.

[13] P. Abbeel, D. Dolgov, A. Y. Ng, and S. Thrun, "Apprenticeship learning for motion planning with application to parking lot navigation," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, September 22-26, 2008, Acropolis Convention Center, Nice, France*, 2008, pp. 1083–1090.

[14] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, 2016, pp. 49–58.

[15] M. T. Wolf and J. W. Burdick, "Artificial potential functions for highway driving with collision avoidance," in *2008 IEEE International Conference on Robotics and Automation*, May 2008, pp. 3731–3736.

[16] M. Montemerlo, J. Becker, S. Bhat, *et al.*, "Junior: The stanford entry in the urban challenge," in *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic, George Air Force Base, Victorville, California, USA*, 2009, pp. 91–123.

[17] J. Wei and J. M. Dolan, "A robust autonomous freeway driving algorithm," in *2009 IEEE Intelligent Vehicles Symposium*, June 2009, pp. 1015–1020.

[18] D. Silver, J. A. Bagnell, and A. Stentz, "High performance outdoor navigation from overhead data using imitation learning," in *Robotics: Science and Systems IV, Eidgenössische Technische Hochschule Zürich, Zurich, Switzerland, June 25-28, 2008*, 2008.

[19] M. Shimosaka, T. Kaneko, and K. Nishi, "Modeling risk anticipation and defensive driving on residential roads with inverse reinforcement learning," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Oct 2014, pp. 1694–1700.

[20] D. Sierra González, V. Romero-Cano, J. Steeve Dibangoye, and C. Laugier, "Interaction-Aware Driver Maneuver Inference in Highways Using Realistic Driver Models," in *Proceedings of the 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC 2017)*, Yokohama, Japan, Oct. 2017.

[21] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverages effects on human actions," in *Proceedings of the Robotics: Science and Systems Conference (RSS)*, Jun 2016.

[22] S. Levine and V. Koltun, "Continuous Inverse Optimal Control with Locally Optimal Examples," in *ICML '12: Proceedings of the 29th International Conference on Machine Learning*, 2012.

[23] A. Byravan, M. Monfort, B. D. Ziebart, *et al.*, "Graph-based inverse optimal control for robot manipulation," in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, 2015, pp. 1874–1880.

[24] B. Okal and K. O. Arras, "Learning socially normative robot navigation behaviors with bayesian inverse reinforcement learning," in *2016 IEEE International Conference on Robotics and Automation, ICRA 2016, Stockholm, Sweden, May 16-21, 2016*, 2016, pp. 2889–2895.

[25] N. Ratliff, J. A. D. Bagnell, and M. Zinkevich, "Maximum margin planning," in *International Conference on Machine Learning*, Pittsburgh, PA, July 2006.

[26] M. Kuderer, S. Gulati, and W. Burgard, "Learning driving styles for autonomous vehicles from demonstration," in *IEEE International Conference on Robotics and Automation, ICRA 2015, Seattle, WA, USA, 26-30 May, 2015*, 2015, pp. 2641–2646.

[27] K. Shiarlis, J. Messias, and S. Whiteson, "Rapidly exploring learning trees," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 1541–1548.

[28] S. H. Lee and S. W. Seo, "A learning-based framework for handling dilemmas in urban automated driving," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 1436–1442.

[29] M. Shimosaka, J. Sato, K. Takenaka, and K. Hitomi, "Fast inverse reinforcement learning with interval consistent graph for driving behavior prediction," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, 2017, pp. 1532–1538.

[30] M. McNaughton, C. Urmson, J. M. Dolan, and J. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice," in *IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China, 9-13 May 2011*, 2011, pp. 4889–4895.

[31] T. M. Howard, C. J. Green, A. Kelly, and D. Ferguson, "State space sampling of feasible motions for high-performance mobile robot navigation in complex environments," *J. Field Robotics*, vol. 25, no. 6-7, pp. 325–345, 2008.

[32] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, 2008, pp. 1433–1438.

[33] M. Pivtoraiko and A. Kelly, "Efficient constrained path planning via search in state lattices," in *The 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, September 2005.

[34] A. Kelly and B. Nagy, "Reactive nonholonomic trajectory generation via parametric optimal control," *I. J. Robotics Res.*, vol. 22, no. 7-8, pp. 583–602, 2003.

[35] C. Guestrin and D. Ormoneit, "Robust combination of local controllers," in *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, University of Washington, Seattle, Washington, USA, August 2-5, 2001*, 2001, pp. 178–185.

[36] G. Neumann, M. Pfeiffer, and W. Maass, *Efficient Continuous-Time Reinforcement Learning with Adaptive State Graphs*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 250–261.

[37] L. Rummelhard, A. Nègre, and C. Laugier, "Conditional monte carlo dense occupancy tracker," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, Sept 2015, pp. 2485–2490.

[38] M. Wulfmeier, D. Z. Wang, and I. Posner, "Watch this: Scalable cost-function learning for path planning in urban environments," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 2089–2095.

[39] D. Vasquez, B. Okal, and K. O. Arras, "Inverse reinforcement learning algorithms and features for robot navigation in crowds: An experimental comparison," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2014, pp. 1341–1346.

[40] P. Henry, C. Vollmer, B. Ferris, and D. Fox, "Learning to navigate through crowded environments," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 981–986, 2010.