Learning Driving Styles for Autonomous Vehicles from Demonstration

Markus Kuderer Shilpa Gulati Wolfram Burgard

***Abstract*— It is expected that autonomous vehicles capable of driving without human supervision will be released to market within the next decade. For user acceptance, such vehicles should not only be safe and reliable, they should also provide a comfortable user experience. However, individual perception of comfort may vary considerably among users. Whereas some users might prefer sporty driving with high accelerations, others might prefer a more relaxed style. Typically, a large number of parameters such as acceleration profiles, distances to other cars, speed during lane changes, etc., characterize a human driver's style. Manual tuning of these parameters may be a tedious and error-prone task. Therefore, we propose a learning from demonstration approach that allows the user to simply demonstrate the desired style by driving the car manually. We model the individual style in terms of a cost function and use feature-based inverse reinforcement learning to find the model parameters that fit the observed style best. Once the model has been learned, it can be used to efficiently compute trajectories for the vehicle in autonomous mode. We show that our approach is capable of learning cost functions and reproducing different driving styles using data from real drivers.**

I. Introduction

Recent studies indicate that the pace of innovation and investment in "self-driving" vehicles is accelerating and that consumers are open to the idea of such vehicles [8, 9]. Some of the key factors in user acceptance of autonomous vehicles are safety, reliability, and comfort. Comfort is subjective and can be influenced by various factors including driving style, which is the way in which a driver habitually drives [5] and is a trade-off among features such as speed, acceleration, jerk, distance to other vehicles, etc. Studies suggest that driving styles vary across users [19]. To be comfortable for different users, an autonomous vehicle should adapt its driving style according to user preferences in addition to maintaining safety.

Different driving styles for an automated vehicle can be achieved by varying the model parameters of its motion planning algorithm. However, manual tuning of these parameters is in general difficult to perform because of the high number of parameters that may have antagonistic effects. If all the variability in user preferences falls into a small set of categories, it may be possible to manually tune the parameters once and choose the parameters for a user based on their preference category. If not, manual tuning, if at all possible, is likely to be a tedious and time-consuming process.

In this paper, we propose a learning from demonstration approach to learn the model parameters for each user from their observed driving style. We assume that the desired driving style maximizes some notion of a reward, i.e., the driver's style can be explained by a cost function. The challenge is to find the cost function that best explains the observed style and that also generalizes to different situations. We propose a feature-based inverse reinforcement learning (IRL) method to learn driving styles from demonstrations. Features are mappings from trajectories to real values which capture important properties of the driving style that we want to reproduce. Our model uses a cost function that is a linear combination of these features. The goal of the learning method is to find the feature weights that fit the observed style best.

Markus Kuderer and Wolfram Burgard are with the Department of Computer Science at University of Freiburg, Germany. At the time this work was conducted, Shilpa Gulati was with Robert Bosch LLC, Chassis Systems Control, Palo Alto, USA.

Fig. 1. A Bosch highly automated driving development vehicle.

Once the model has been learned, we use it to compute trajectories online during autonomous driving tasks. Especially for highway driving it is important to also capture higher-order properties such as velocity, acceleration and jerk. Studies show that acceleration and jerk have a strong influence on the comfort of the passengers [7]. To capture these properties, we propose to use a continuous representation of trajectories.

In the remainder of this paper we discuss related work and describe the state representation and an IRL approach. Furthermore, we introduce features that are relevant to the task of highway driving and present experimental results that suggest that our method is suitable for learning individual driving styles from demonstrated trajectories.

II. Related Work

In the past, machine learning techniques in various fashions have been used to improve the performance of autonomous vehicles. In 1991, Pomerleau [14] used neural networks to learn to steer a vehicle on a highway by observing a person drive. The network receives live input from a camera on the vehicle and learns a control function of the steering angle to keep the vehicle on track. Riedmiller et al. [16] use reinforcement learning to learn a steering controller from

scratch. Their approach learns a controller that is able to navigate the vehicle on the track within 25 min of driving a real car. In this work, our goal is to learn a more complex behavior that does not only allow the vehicle to stay in the current lane but to maintain a desired speed, keep safe distance to other vehicles, and also change lanes.

The method we propose in this paper builds on inverse reinforcement learning (IRL) which was introduced by Ng and Russell [13] as the problem of deriving a cost function from observed behavior. Abbeel and Ng [1] use a linear combination of features to represent the cost function. Ziebart et al. [21] and Choi and Kim [4] use variants of IRL and apply their methods to learn taxi drivers' preferences for road segments from GPS trace data.

Abbeel and Ng [1] motivate the use of IRL for learning driving styles [1]. They suggest that drivers typically trade off many different factors, which we would have to weigh when specifying a reward function. Even though most drivers know how to drive competently, it is difficult to state the exact reward function for the task of *driving well*. They apply their method to learn different behavior styles on a highway simulation with three lanes and five discrete actions.

Babes et al. [2] introduced maximum likelihood inverse reinforcement learning. Similar to Ziebart et al. [20] they maximize the likelihood of the data assuming an exponential family distribution. They furthermore introduce a method to automatically cluster observed behavior styles and learn individual feature weights for each cluster. Chen et al. [3] propose a method to automatically discover a set of relevant features in IRL. In these approaches, the respective IRL algorithms are used to learning behavior patterns on a discrete highway simulator with a small number of states and actions, similar to Abbeel and Ng [1]. Silver et al. [17] apply Maximum Margin Planning (MMP) to learn more complex behavior for autonomous vehicles and show experiments in which learning by demonstration significantly outperforms manual tuning of the parameters. MMP is an IRL variant that aims to minimize the margin between the observations and the optimal trajectory in an MDP by adjusting the cost of discrete state-action pairs.

In contrast to a discrete state and action space, we consider trajectories in continuous state spaces. This allows our method to learn higher order properties such as lateral jerk, which is important for the comfort of users. Levine and Koltun [12] also present a method for IRL in continuous state and action spaces. They assume locally optimal demonstrations and approximate the resulting distribution using Taylor expansion. In contrast to their method, we do not model the dynamics as an Markov decision process, but compute expected feature values using a time-continuous spline representation of the trajectories. These continuous trajectories are directly suitable for online control of an automated car.

The method we utilize in this paper builds on previous approaches for learning pedestrian navigation behavior [10, 11]. However, in the context of highway driving we encounter different preconditions and challenges. For example, in contrast to free movement in all directions, vehicles need to follow their respective lanes. Furthermore, continuous acceleration and bounded jerk are necessary for driving comfort especially when driving with high speed.

The approach presented in this paper relies on a finite-dimensional representation of trajectories. We choose to encode the 2D position along the trajectories using quintic splines, similar to Sprunk et al. [18]. They optimize trajectories with respect to a user defined cost function to navigate a holonomic mobile robot. Gulati [6] proposes an alternative trajectory representation. Instead of the positions along the trajectory, Gulati parametrizes a trajectory using orientation and speed. This allows boundary conditions with zero speed. However, this parametrization requires numerical integration to compute the position along the trajectory, in contrast to an efficient closed form representation. The learning approach proposed in this paper is independent of the representation of trajectories, as long as it allows computation of the features and their derivatives.

III. Learning Navigation Behavior from Demonstration

In this section we describe our feature-based approach for learning from demonstration in the context of highway driving.

A. Trajectory Representation

We represent the 2D driving style of vehicles using trajectories $\mathbf{r}$ that are mappings $\mathbf{r} : \mathbb{R} \to \mathbb{R}^2$ from time to a 2D position. In general, the space of such trajectories has infinitely many dimensions. In this work, we use splines as a finite-dimensional representation of trajectories to overcome this problem. More precisely, we utilize quintic splines in $\mathbb{R}^2$ to represent the x and y position of the vehicle over time. Each spline segment

$$s_j : [t_j, t_{j+1}] \to \mathbb{R}^2 \tag{1}$$

is a 2D quintic polynomial that defines the position of the vehicle in a time interval $[t_j, t_{j+1}]$, where $0 \leq j < S$ for a spline with S segments. Thus, the trajectory of the vehicle is given by

$$\mathbf{r}(t) = \mathbf{s}_j(t), \text{ for } t \in [t_j, t_{j+1}]. \tag{2}$$

We use the position $\mathbf{p}_j := \mathbf{r}(t_j)$, velocity $\mathbf{v}_j := \dot{\mathbf{r}}(t_j)$, and acceleration $\mathbf{a}_j := \ddot{\mathbf{r}}(t_j)$ for $j \in \{0, \ldots, S\}$ to parameterize the spline. The set of these control points at time t_j and t_{j+1} fully defines the spline segment s_j, which has six degrees of freedom. Since two adjacent spline segments share control points, we have continuous velocity $\mathbf{v}(t) = \dot{\mathbf{r}}(t)$ and acceleration $\mathbf{a}(t) = \ddot{\mathbf{r}}(t)$ along the trajectory. As a result, the curvature of the path and hence the lateral acceleration is continuous as long as speed is non-zero, which is necessary for achieving comfortable motion.

B. Maximum Entropy Inverse Reinforcement Learning

Given a set of N observed trajectories $\tilde{\mathbf{r}}_1, \ldots, \tilde{\mathbf{r}}_N$ our goal is to learn a model that explains the observations and that is capable of generating trajectories of similar characteristics in other situations. However, it is not obvious what *similar*

means in this context. For example, if we have training examples from a certain highway, an unbiased learning algorithm without any assumptions might learn that it is very important to drive on this exact highway. However, we rather want a system that learns acceleration profiles or local obstacle avoidance behavior and that is able to generate similar trajectories on any given highway.

To introduce such domain knowledge, we propose a feature-based learning algorithm. Each feature f_k is a function that maps a trajectory to a real value, capturing some relevant characteristic. The vector of all features $\mathbf{f}$ is thus a function that maps trajectories to a real vector

$$\mathbf{f} : \mathbf{r} \mapsto (f_1(\mathbf{r}), \ldots, f_n(\mathbf{r})) \in \mathbb{R}^n. \tag{3}$$

In this way, the empirical feature values of the observed trajectories $\tilde{\mathbf{f}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{f}(\tilde{\mathbf{r}}_i)$ encode relevant properties of the demonstrations, such as accelerations, velocities, or distances to other vehicles.

Our goal is to find a generative model that yields trajectories that are similar to the observations, where the features serve as a measure of similarity. More specifically, given a probabilistic model that yields a probability distribution over trajectories $p(\mathbf{r} \mid \boldsymbol{\theta})$, our goal is to find the parameters $\boldsymbol{\theta}$ such that the expected feature values match the observed empirical feature values, i.e.

$$\mathbb{E}_{p(\mathbf{r}|\boldsymbol{\theta})}[\mathbf{f}] = \tilde{\mathbf{f}}. \tag{4}$$

In general, there are many distributions with this property. Within the class of all distributions that match features, Ziebart et al. [20] propose to select the one that maximizes the entropy. Following the principle of maximum entropy, this is the distribution that describes the data best since it is the least biased distribution. The solution of the constraint optimization problem of optimizing the entropy given the constraints in Eq. (4) has the form

$$p(\mathbf{r} \mid \boldsymbol{\theta}) = \exp\Big(-\boldsymbol{\theta}^T \mathbf{f}(\mathbf{r})\Big). \tag{5}$$

One can interpret $\boldsymbol{\theta}^T\mathbf{f}(\mathbf{r})$ as a cost function, where agents are exponentially more likely to select trajectories with lower cost. Unfortunately, it is not possible to compute $\boldsymbol{\theta}$ analytically, but we can compute the gradient of the Lagrange function of the constraint optimization problem with respect to $\boldsymbol{\theta}$. It can be shown that this gradient is the difference between the expected and the empirical feature values

$$\mathbb{E}_{p(\mathbf{r}|\boldsymbol{\theta})}[\mathbf{f}] - \tilde{\mathbf{f}}. \tag{6}$$

There is an intuitive explanation for this gradient: when the expected value $\mathbb{E}_{p(\mathbf{r}|\boldsymbol{\theta})}[f_k]$ for a feature f_k is too high, we should increase the corresponding weight θ_k, which in turn assigns lower likelihood to any trajectories with high feature values $f_k(\mathbf{r})$. As a result, the expected feature value $\mathbb{E}_{p(\mathbf{r}|\boldsymbol{\theta})}[f_k]$ decreases. More details about the derivation of this learning algorithm can be found in Ziebart et al. [20] and Kretzschmar et al. [10].

C. Maximum Likelihood Approximation

The main challenge of the learning approach described above is to compute the expected feature values $\mathbb{E}_{p(\mathbf{r}|\boldsymbol{\theta})}[\mathbf{f}]$. For the high-dimensional distributions it is in general not possible to compute the integral

$$\mathbb{E}_{p(\mathbf{r}|\boldsymbol{\theta})}[\mathbf{f}] = \int p(\mathbf{r} \mid \boldsymbol{\theta})\mathbf{f}(\mathbf{r})d\mathbf{r} \tag{7}$$

analytically. Kretzschmar et al. [10] describe how to sample trajectories from the high dimensional space of trajectories using Hamiltonian Markov chain Monte Carlo methods. However, this sampling method is computationally very expensive. A possible approximation of the expected feature values is to compute the feature values of the most likely trajectory, instead of computing the expectations by sampling [11]:

$$\mathbb{E}_p[\mathbf{f}] \approx \mathbf{f}(\arg\max_{\mathbf{r}} p(\mathbf{r} \mid \boldsymbol{\theta})). \tag{8}$$

The resulting learning method is also known as *inverse optimal control*. With this approximation, we assume that the demonstrations are in fact generated by minimizing a cost function, in contrast to the assumption that demonstrations are samples from a probability distribution. Our experiments suggest that this approximation is suitable in the context of learning individual driving styles on highways.

D. Learning Highway Navigation Style

We apply the learning algorithm as described above to driving style learning on highways. In the following, we propose features that capture relevant properties of driving styles such as velocities, acceleration and distances to other agents, the distance to the desired lane, and the desired speed.

1) Acceleration: Integrating the squared acceleration over the trajectory yields the feature

$$f_{\mathrm{a}} = \int_t \|\ddot{\mathbf{r}}(t)\|^2 \; dt. \tag{9}$$

2) Normal Acceleration: In addition, we use a feature that represents the acceleration perpendicular to the direction of the lane.

$$f_{\mathrm{a}_n} = \int_t \left(d_x(t)\ddot{\mathbf{r}}_y(t) - d_y(t)\ddot{\mathbf{r}}_x(t)\right)^2 \; dt, \tag{10}$$

where $\mathbf{d}(t)$ is the current normalized direction vector of the lane. Experimental studies showed that passengers of ground vehicles react very sensitive to lateral accelerations.

3) Jerk: For the comfort of passengers the jerk along the trajectory is also an important quantity.

$$f_{\mathrm{j}} = \int_t \|\dddot{\mathbf{r}}(t)\|^2 \; dt. \tag{11}$$

4) Normal Jerk: Similar to acceleration, we also capture the lateral jerk perpendicular to the direction of the lane

$$f_{\mathrm{j}_n} = \int_t \left(d_x(t)\dddot{\mathbf{r}}_y(t) - d_y(t)\dddot{\mathbf{r}}_x(t)\right)^2 \; dt, \tag{12}$$

where $\mathbf{d}(t)$ is the current normalized direction vector of the lane.

5) Curvature: Since a real vehicle has typically a limited turning circle, we additionally introduce a feature that represents the squared curvature

$$f_\kappa = \int_t \|\kappa(t)\|^2 \, dt. \tag{13}$$

6) Desired Speed: To represent deviation from a desired speed, we use the feature

$$f_\mathrm{v} = \int_t \|\mathbf{v}_\mathrm{des} - \dot{\mathbf{r}}(t)\| \, dt, \tag{14}$$

where $\mathbf{v}_\mathrm{des}$ is the desired velocity vector in direction of the current lane. The desired speed could either be the speed limit on the given highway, or a lower individual speed that is comfortable for the user.

7) Lane: When driving on highways, the vehicle should drive close to the center of the current lane. We represent this property by the feature

$$f_\mathrm{l} = \int_t \|\mathbf{l}(t) - \mathbf{r}(t)\| \, dt. \tag{15}$$

where $\mathbf{l}(t)$ is the closest centerline point at time t.

8) Collision Avoidance: Obviously the distance to other vehicles is important to avoid crashes. Therefore, we introduce the feature

$$f_\mathrm{d} = \sum_c \int_t \frac{1}{\|\mathbf{r}(t) - \mathbf{o}_c(t)\|^2} \, dt, \tag{16}$$

where $\mathbf{o}_c(t)$ is the closest point to vehicle c at time t. This feature increases as the car gets closer to any obstacle.

9) Following distance: When following other vehicles, we want to keep a safety distance which is greater than the minimum allowed distance between two vehicles on neighboring lanes. To account for the distance to the following vehicle on the same lane, we propose the feature

$$f_\mathrm{fd} = \int_t \max\left(0, \hat{d} - d(t)\right) \, dt, \tag{17}$$

where $d(t)$ is the current distance parallel to the lane and $\hat{d}$ the desired distance.

E. Learning from Demonstration

In this section, we outline the learning process when approximating the expectations using maximum likelihood trajectories, as described in Sec. III-C. Given a set of demonstrated trajectories $\{\tilde{\mathbf{r}}_1, \ldots, \tilde{\mathbf{r}}_N\}$, the following steps lead to the desired policy. Here, we outline the process

1) Compute the empirical feature vector averaged over all demonstrations $\tilde{\mathbf{f}} = \frac{1}{N}\sum_{i=1}^N \mathbf{f}(\tilde{\mathbf{r}}_i)$ by evaluating the feature functions for all demonstrated trajectories.
2) Initialize the weight vector $\boldsymbol{\theta}$ with arbitrary values.
3) For each demonstrated trajectory, fix the environment state including position, velocity and acceleration at the start, lane information and the state of nearby vehicles. Then optimize the free parameters of these trajectories with respect to the cost function $\boldsymbol{\theta}^T\mathbf{f}(\mathbf{r})$. We denote the optimized trajectories by $\{\mathbf{r}_1^{\boldsymbol{\theta}}, \ldots, \mathbf{r}_N^{\boldsymbol{\theta}}\}$.
4) Compute the approximated expected feature values by evaluating the feature functions for all optimized trajectories $\mathbb{E}_{p(\mathbf{r}|\boldsymbol{\theta})}[\mathbf{f}] \approx \mathbf{f}_\mathrm{ML}^{\boldsymbol{\theta}} := \frac{1}{N}\sum_{i=1}^N \mathbf{f}(\mathbf{r}_i^{\boldsymbol{\theta}})$.
5) The gradient for the optimization is given by $\mathbf{f}_\mathrm{ML}^{\boldsymbol{\theta}} - \tilde{\mathbf{f}}$, as stated in Eq. (6). Use this gradient to update the current estimation of the weight vector $\boldsymbol{\theta}$.
6) Repeat from step 3 until convergence.

IV. Navigating an Autonomous Vehicle using the Learned Model

The learned model allows us to compute trajectories for navigating an autonomous vehicle. To this end, we continuously compute the most likely trajectory, which is the trajectory with lowest cost $\boldsymbol{\theta}^T\boldsymbol{f}$, using the optimization algorithm RPROP [15].

A. Efficient Computation of Feature Gradients

For online optimization during navigation, it is crucial to efficiently compute the gradients

$$\frac{\partial \boldsymbol{\theta}^T\mathbf{f}(\mathbf{r})}{\partial(\boldsymbol{p}_0, \boldsymbol{v}_0, \boldsymbol{a}_0, \ldots, \boldsymbol{p}_S, \boldsymbol{v}_S, \boldsymbol{a}_S)} \tag{18}$$

$$= \sum_k \theta_k \frac{\partial f_k(\mathbf{r})}{\partial(\boldsymbol{p}_0, \boldsymbol{v}_0, \boldsymbol{a}_0, \ldots, \boldsymbol{p}_S, \boldsymbol{v}_S, \boldsymbol{a}_S)} \tag{19}$$

$$= \sum_k \theta_k \nabla f_k(\mathbf{r}). \tag{20}$$

For the features accounting for acceleration and jerk, we can compute the derivatives in closed form, since these features are integrals over polynomials. For more complex features we cannot compute the gradients analytically but use a combination of numerical integration and analytical derivatives. Applying numerical integration

$$f(\mathbf{r}) = \int_t c(\mathbf{r}, t) dt \tag{21}$$

$$\approx \sum_{t \in \{0, \Delta t, \ldots\}} \frac{\Delta t}{2}\left(c(\mathbf{r}, t) + c(\mathbf{r}, t_n + \Delta t)\right) \tag{22}$$

allows us to compute the derivatives at each sampling point:

$$\nabla f(\mathbf{r}) = \sum_{t \in \{0, \Delta t, \ldots\}} \frac{\Delta t}{2}\left(\nabla c(\mathbf{r}, t) + \nabla c(\mathbf{r}, t + \Delta t)\right). \tag{23}$$

For the features listed in Sec. III-D, it is feasible to compute the derivatives of the inner function $c(\mathbf{r}, t)$ in closed form and therefore to efficiently compute the feature gradients using numerical integration.

B. Navigation

We integrated our method into a planning framework for autonomous vehicles. At each planning cycle, we update the state of other perceived vehicles on the highway given the current sensor readings. We predict the future behavior of these vehicles assuming constant speed along their respective lanes. Based on this, we use the method described above to optimize the trajectory of the autonomous vehicles based on the learned cost function.

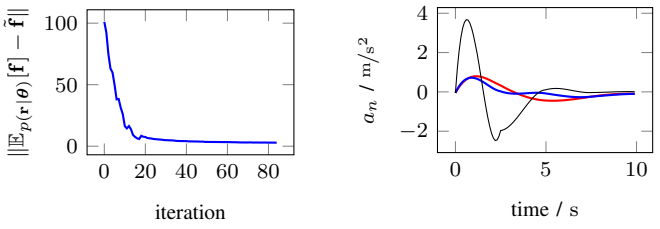

Fig. 2. Left: evolution of the norm of the difference between the empirical feature values and the expected feature values during learning on a dataset of 20 observed trajectories. Right: normal acceleration of a demonstrated trajectory (blue), the initial guess (black), and the optimized trajectory with the final learned policy (red).

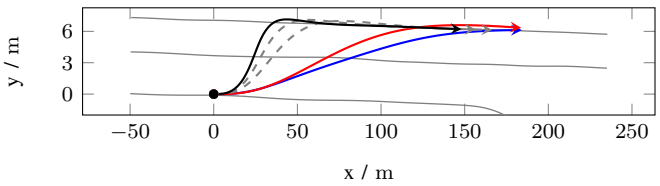

Fig. 3. Demonstrated trajectory (blue), the initial guess (black), and the optimized trajectory with the final learned policy (red). The trajectory shows a change of two lanes to the left. The dashed lines illustrate intermediate policies during the learning phase.

More precisely, we fix at each planning cycle the position $\mathbf{p}_0$, velocity $\mathbf{v}_0$, and acceleration $\mathbf{a}_0$ of the first node according to the trajectory that we sent to the controller in the previous planning cycle. The remaining spline control points $\mathbf{p}_j$, $\mathbf{v}_j$, and $\mathbf{a}_j$ for $j \in \{1, \ldots, S\}$ are variables for the optimization with respect to the learned cost function $\boldsymbol{\theta}^T \boldsymbol{f}(\mathbf{r})$. In this way, we can assure smooth transitions between the trajectories of subsequent planning cycles.

The cost function, which is a linear combination of the features described in Sec. III-D, yields a smooth, comfortable trajectory, keeping a safe distance to obstacles. The learned feature weighs encode the trade-off between antagonistic goals, such as the desired speed and limited accelerations. When a slower vehicle blocks the lane, our system predictively computes a trajectory that decelerates and keeps a safe distance until it is safe to drive at the desired speed again.

When we change the desired lane, the cost function assigns lower cost to trajectories that are near the new desired lane. As a consequence, the optimized trajectory immediately leads to a lane change. The features that capture the curvature, normal acceleration, and normal jerk guarantee a smooth lane change that shows characteristics as demonstrated before.

V. Experiments

In this section we present a set of experiments to evaluate the performance of our method to learn a policy for automated highway driving from demonstration.

A. Data Acquisition

To record training data, we used a car that is equipped with a range of sensors, which allow us to localize in an existing map. Furthermore, the car observes and tracks other vehicles in its vicinity. For the experiments in this section, we recorded the driving style of different drivers on a US highway with normal traffic. Each driver was instructed to demonstrate acceleration maneuvers in the velocity range of $20\,\mathrm{m/s}$ to $30\,\mathrm{m/s}$, as well as lane change maneuvers. From these datasets of about $8\,\mathrm{min}$ each, we manually selected the contained acceleration and lane change maneuvers. These observations consist of the trajectory of the car, the trajectories of all other vehicles in the vicinity, and lane information. For each sample, we set the desired speed according to the last observed speed. Similarly, we set the desired lane to the closest lane at the end of the observed trajectory. To avoid this manual pre-selection step, we could also obtain samples in an interactive training session, where the car issues commands such as "perform a lane change to the left lane now".

B. Learning Individual Navigation Styles

The goal of the learning method is to find weight vectors $\boldsymbol{\theta}$ that yield policies which explain the observed trajectories. We applied our learning algorithm as described in Sec. III-E to the observed trajectories of different users. The initial guess for the feature weights $\boldsymbol{\theta}$ was an all-ones vector. Fig. 2 (left) shows the deviation of the empirical features and the approximated, expected feature values during the learning procedure of one of the datasets. After about 30 iterations, the feature weights converge.

Fig. 3 shows an observed trajectory of changing two lanes to the left. In addition to the observed trajectory, which was excluded from the training set, the figure shows the initial guess as well as the trajectory optimized with the learned policy. The learned policy yields a trajectory with similar characteristics compared to the observation, which is also apparent in Fig. 2 (right) that shows the normal acceleration during the lane change maneuver.

Fig. 5 shows the velocity and acceleration profiles for a different trajectory, where the car accelerates from an initial velocity of 23 m/s to a desired velocity of 29 m/s. The plots show the observation, the initial guess, and the final learning result. Similar to the lane change behavior, the learned method better replicates the behavior compared to the initial guess.

To show the ability of our method to learn distinct policies for different users, Fig. 4 contains the mean acceleration and jerk over trajectories demonstrated by two users, and the learned policies applied to the same start configurations and desired velocities as observed. The figure shows that our method is able to reproduce the characteristics when applied to the test set. Additionally, it also shows that the characteristics transfer to a distinct test set of trajectories, which suggests that the learned policies also generalize to different situations.

In our experiments, the resulting trajectories after optimization do not fit the observation perfectly. The reason for this is twofold. First, the learned policy is an average over the set of different sample trajectories. Second, the observations do not exactly meet the optimality criteria for any cost function that is a linear combination of the features we use. However, the experimental results suggest that our algorithm learns the

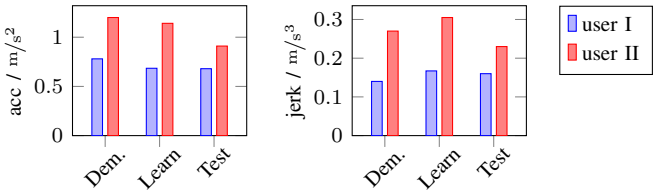

Fig. 4. The average acceleration (left) and jerk (right) over trajectories of datasets of two users, observed on US highways. The plots show the average accelerations and velocities over the demonstrations, the learned policy applied to the individual dataset used for learning, and a distinct test set. The results show that our algorithm captures distinct styles of two users.

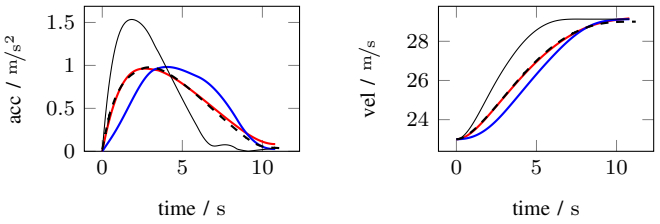

Fig. 5. Speed (left) and acceleration (right) of a demonstrated trajectory (blue), the initial guess (black), and the optimized trajectory with the final learned policy (red). The dashed line shows the trajectory of a a car in a realistic simulation that accelerates from 23 m/s to 29 m/s using the learned policy.

magnitude of the quantities that contribute to the comfort of the users. In practice, only the user itself is able to assess the comfort felt by driving the autonomous car. Planned future work in this field includes a user study in which users demonstrate behavior and subsequently evaluate the performance of the automated car that uses the learned policy.

C. Autonomous Driving

We applied the learned policy to autonomous navigation, as described in Sec. IV-B. Our system continuously computes the trajectory with lowest cost with 5 Hz and uses the trajectory to control a car in a realistic simulation environment. The dashed line in Fig. 5 shows the acceleration of the simulated car when changing speed from 23 m/s to 29 m/s. The acceleration profiles coincide with the trajectories optimized offline for the same acceleration task. This experiment suggests that the learned policy is suitable to autonomously control a car with similar characteristics as observed from real drivers.

VI. Conclusion

In this paper, we presented an inverse reinforcement learning method to learn individual driving styles for self-driving cars from demonstration. To capture the relevant properties of highway driving, we propose a set of features that capture distances to other vehicles, the distance to the desired lane as well as higher order properties such as velocities and accelerations. By matching observed empirical feature values with the expected feature values of the model, the proposed method allows us to learn a policy that captures an individual driving style. Experiments carried out with oberved trajectories from a real car suggest that our method is able to reliably learn policies from demonstration suitable for autonomous navigation.

References

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proc. of the 21st Int. Conf. on Machine Learning (ICML)*, 2004.

[2] M. Babes, V. Marivate, M. Littman, and K. Subramanian. Apprenticeship learning about multiple intentions. In *Proc. of the 28th Int. Conf. on Machine Learning (ICML)*, 2011.

[3] S. Chen, H. Qian, J. Fan, Z. Jin, and M. Zhu. Modified reward function on abstract features in inverse reinforcement learning. *Journal of Zhejiang University SCIENCE C*, 11(9), 2010.

[4] J. Choi and K.-E. Kim. Bayesian nonparametric feature construction for inverse reinforcement learning. In *Int. Joint Conf. on Artificial Intelligence (IJCAI)*. AAAI Press, 2013.

[5] J. Elander, R. West, and D. French. Behavioral correlates of individual differences in road-traffic crash risk: An examination of methods and findings. *Psychological bulletin*, 113(2), 1993.

[6] S. Gulati. *A framework for characterization and planning of safe, comfortable, and customizable motion of assistive mobile robots*. PhD thesis, The University of Texas at Austin, 2011.

[7] I. D. Jacobson, L. G. Richards, and A. R. Kuhlthau. Models of human comfort in vehicle environments. *Human factors in transport research*, 20, 1980.

[8] KPMG2012. Self-driving cars: the next revolution. http://www.kpmg.com/US/en/IssuesAndInsights/ArticlesPublications/Documents/self-driving-cars-next-revolution.pdf, 2012.

[9] KPMG2013. Self-driving cars: are we ready? http://www.kpmg.com/US/en/IssuesAndInsights/ArticlesPublications/Documents/self-driving-cars-are-we-ready.pdf, 2013.

[10] H. Kretzschmar, M. Kuderer, and W. Burgard. Learning to predict trajectories of cooperatively navigating agents. In *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, Hong Kong, China, 2014.

[11] M. Kuderer, H. Kretzschmar, C. Sprunk, and W. Burgard. Feature-based prediction of trajectories for socially compliant navigation. In *Robotics: Science and Systems (RSS)*, 2012.

[12] S. Levine and V. Koltun. Continuous inverse optimal control with locally optimal examples. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2012.

[13] A. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2000.

[14] D. A. Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1), 1991.

[15] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proc. of the IEEE Int. Conf. on Neural Networks (ICNN)*, 1993.

[16] M. Riedmiller, M. Montemerlo, and H. Dahlkamp. Learning to drive a real car in 20 minutes. In *Frontiers in the Convergence of Bioscience and Information Technologies*. IEEE, 2007.

[17] D. Silver, J. A. Bagnell, and A. Stentz. Learning autonomous driving styles and maneuvers from expert demonstration. In *Experimental Robotics*. Springer, 2013.

[18] C. Sprunk, B. Lau, P. Pfaff, and W. Burgard. Online generation of kinodynamic trajectories for non-circular omnidirectional robots. In *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011.

[19] O. Taubman-Ben-Ari, M. Mikulincer, and O. Gillath. The multidimensional driving style inventoryscale construct and validation. *Accident Analysis and Prevention*, 36(3), 2004.

[20] B. Ziebart, A. Maas, J. Bagnell, and A. Dey. Maximum entropy inverse reinforcement learning. In *AAAI Conf. on Artificial Intelligence (AAAI)*, 2008.

[21] B. D. Ziebart, A. L. Maas, A. K. Dey, and J. A. Bagnell. Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior. In *Proc. of the Int. Conf. on Ubiquitous Computing*. ACM, 2008.