2019 IEEE Intelligent Transportation Systems Conference (ITSC)
Auckland, NZ, October 27-30, 2019

Accelerated Inverse Reinforcement Learning with Randomly Pre-sampled Policies for Autonomous Driving Reward Design

Long Xin[1], Shengbo Eben Li*[1], Pin Wang[2], Wenhan Cao[1],
Bingbing Nie[1], Ching-Yao Chan[2], and Bo Cheng[1]

***Abstract*—To learn a reward function that a driver adheres to is of importance to the human-like design of autonomous driving systems. Inverse reinforcement learning (IRL) is one of the recent advances that can achieve this objective, but it often suffers from the low efficiency of generating optimal policy by reinforcement learning (RL) each time when updating reward weights. This paper presents an accelerated IRL method by approaching the optimal policy among randomly pre-sampled policies in designed sub-space instead of finding it through RL in the whole policy space. The corresponding trajectories are targeted via an optimal trajectory selector in the candidate trajectory library generated by pre-sampled policies. The weights then are updated by comparing the selected trajectories and the expert ones. The proposed method is very suitable for improving learning efficiency for low-dimensional problems like autonomous driving, whose expert policies are nearly tractable. Results with simulated driving data show that it only took 11 iterations to converge, while the average longitudinal RMS error of the recovered trajectories based on the learned reward function was only 2.14m.**

***Index Terms*—Inverse reinforcement learning, pre-sampled policy, reward design, maximum entropy, autonomous driving.**

I. Introduction

DECISON-MAKING is one of the key techniques in autonomous driving that significantly influences automation level. The decision making problem can be formulated in a Markov Decision Process (MDP) setting, where it usually assumes a reward function is given [1]. However, the reward function is frequently difficult to specify from scenarios to scenarios. For example, when driving on road, drivers typically trade off many different aspects, such as maintaining a safe following distance, maintaining a reasonable speed, driving along the center line, and so on. To specify a reward function for such a driving task, a set of weights have to be assigned to define exactly how drivers prefer to balance these driving goals [2]. Usually, it is very tricky and time-consuming to tweak these weights manually until the desired driving behavior is obtained.

One solution is to learn from experts, not to simply mimic the external actions that experts conduct under each state, but to deeply learn the internal reasoning mechanism. The problem of deriving a reward function from observed expert behaviors is referred as inverse reinforcement learning (IRL) [3]. The expert human driver is assumed to try to optimize an unknown reward function when he drives, and the goal of IRL is to find the specific reward function that yields to an corresponding optimal driving policy which performs as well as the expert's, where the performance is measured with respect to the difference of reward between the expert's trajectories and the learner's generated trajectories [4].

There are two popular class of methods for learning reward function weights. One is margin-based method, e.g., apprenticeship learning [4], max margin planning [5] and structured IRL [6], that finds the optimal weights by minimizing the feature expectations between the expert trajectories and the generated ones. The other one is entropy-based method, e.g., maximum entropy IRL [7] and relative entropy IRL [8], that finds the optimal weights by maximizing the entropy of the generated trajectories.

For margin-based IRL, given a feature mapping from states to reward and the experts feature expectations, the goal is to find a policy and its generated trajectories whose feature expectations are close to the expert's. Abbeel and Ng (2004) proved that feature matching was both necessary and sufficient to achieve the same performance as the expert if the reward function was a linear combination of different features [4]. To accomplish this, such a policy can be found by iteratively solving a max margin problem to achieve a relatively small difference of feature expectations. Unfortunately, recovering the expert's exact reward weights can be very difficult under such concept. The matching of feature expectations is ambiguous as many reward weights, including degeneracies (e.g, all zeroes), can satisfy optimality of expert trajectories, or, each policy can be optimal for many reward functions and many policies can lead to the same feature expectations.

This study is supported by International Sci&Tech Cooperation Program of China with 2016YFE0102200, NSF China with U1664263 and Beijing NSF with JQ18010. All correspondence should be sent to S. E. Li.

L. Xin, S. E. Li, W. Cao, B. Nie and B. Cheng are with State Key Laboratory of Automotive Safety and Energy and School of Vehicle and Mobility, Tsinghua University, Beijing, 100084 China (e-mail: {xin-l13, cwh19}@mails.tsinghua.edu.cn, {lishbo, nbb, chengbo}@tsinghua.edu.cn)

P. Wang and C-Y. Chan are with California PATH, University of California, Berkeley, Richmond, CA, 94804 USA (email: {pinwang, cychan}@berkeley.edu)

978-1-5386-7024-8/19/$31.00 ©2019 IEEE

Ziebart *et al.* (2008) employed the principle of maximum entropy to resolve ambiguities in choosing trajectory distributions [7]. This principle maximizes the uncertainty [9] and leads to the distribution over behaviors constrained to matching feature expectations, while being no more committed to any particular trajectory than this constraint requires. Maximizing the entropy of the distribution over trajectories subject to the feature constraints from expert's trajectories implies to maximize the likelihood under the maximum entropy (exponential family) distributions. The problem is convex for MDPs and the optima can be obtained using gradient-based optimization methods. The gradient is the difference between empirical feature expectations and the learners expected feature expectations.

With the development of the two aforementioned mainstreams of methods, many researchers made their own contributions to IRL. Babes *et al.* (2011) [10] applied apprenticeship learning to the problem of learning from unlabeled expert trajectories generated by varying intentions. They derived an expectation-maximum (EM) approach that clustered observed trajectories by inferring the objectives for each cluster, and then used the constructed clusters to quickly identify the intention of a trajectory. Kuderer *et al.* (2015) [11] represented the 2D driving behavior of vehicles using trajectories $\boldsymbol{r}$ that are mappings from time to a 2D position. The authors used quantic splines as a finite-dimensional representation of trajectories to overcome the problem that the space of such trajectories has infinitely many dimensions. Wulfmeier *et al.* (2015) [12] replaced the linear combination of features with a deep neural network, which had a better approximation of reward function based on the feature representation of MDP states.

IRL has made great progress in the past two decades, but still faces many challenges. One of the challenges is that the existing IRL algorithms need to find the optimal policy under the current reward function first, which is inseparable from the involvement of reinforcement learning (RL). However, it is very time-consuming to obtain the optimal policy through RL, especially for problems with continuous states, e.g., autonomous driving. Even worse, it is necessary to call RL repeatedly to solve the optimal strategy each time the weighs are updated, which makes the learning process extremely long. Moreover, different RL methods lead to different optimal policies and different generated trajectories, thus having influence on their feature expectations when updating weights [13]. Therefore, how to deal with the RL part brings new challenges to IRL.

The above-mentioned methods bring challenges of time-consuming learning procedure caused by dependency on RL's learning optimal policy, and randomness of results caused by the selection of specific RL methods. Therefore, this study proposes an inverse reinforcement learning method based on pre-sampled policies, which replaces calling RL to generate the optimal trajectory in each iteration with selecting the optimal trajectories from the candidate trajectory library generated by random sampled policies. The policy space constructed with human knowledge for driving creates a reasonably small neighborhood for the expert driver's policy, thus making it much easier to find the optimal policy (and its corresponding trajectories) and improving the efficiency of weights updating.

The rest of the paper is organized as follows: Section II formulates the problem mathematically. Section III introduces the proposed method. Section IV introduces the implementation of the proposed method for autonomous driving on highway. Section V introduces the simulation scenario and data simulated for test, while Section VI presents the results. Section VII summarizes the major contributions and concludes this paper.

II. Problem Formation

Given a set of N expert trajectories $\boldsymbol{D} = \{\boldsymbol{\xi}_1^{\text{Exp}}, \boldsymbol{\xi}_2^{\text{Exp}}, ..., \boldsymbol{\xi}_N^{\text{Exp}}\}$, each of them is a sequence of state-action pairs with length T_i:

$$\boldsymbol{\xi}_i^{\text{Exp}} = \{(\boldsymbol{s}_0^{(i)}, \boldsymbol{a}_0^{(i)}), (\boldsymbol{s}_1^{(i)}, \boldsymbol{a}_1^{(i)}), ..., (\boldsymbol{s}_{T_i}^{(i)}, \boldsymbol{a}_{T_i}^{(i)})\} \quad (1)$$

Define the feature vector $\boldsymbol{f} \in \mathcal{R}^H$ on trajectory, which is a mapping from a trajectory to feature values:

$$\boldsymbol{f} : \boldsymbol{\xi} \rightarrow [f_1(\boldsymbol{\xi}), ..., f_H(\boldsymbol{\xi})]^T \quad (2)$$

These features can be used to describe the characteristics of how a car is driven on road. Then, the expert driving characteristics are calculated by averaging all expert trajectories:

$$\boldsymbol{f}_{\text{Exp}} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{f}(\boldsymbol{\xi}_i^{\text{Exp}}) \quad (3)$$

where $\boldsymbol{f}_{\text{Exp}} \in \mathcal{R}^H$ is served as the expert feature expectation.

In this study, the reward function is assumed to be a linear combination of features weighted by $\boldsymbol{\omega} \in \mathcal{R}^H$:

$$R(\boldsymbol{\xi}) = \boldsymbol{\omega}^T \boldsymbol{f}(\boldsymbol{\xi}) \quad (4)$$

where $\boldsymbol{\omega}$ is the reward weight to be learned in IRL.

In general, there are many distributions with feature matching property. Within all distributions that match features, Ziebart *et al.* (2008) [7] proposed to select the one that maximized the entropy, because it was the best way to describe it since it was the least biased distribution. Therefore, given a probabilistic model that yields a distribution over trajectories $p(\boldsymbol{\xi}|\boldsymbol{\omega})$, the goal is to find the optimal weights $\boldsymbol{\omega}^*$ maximizing the entropy of these trajectories subject to matching feature expectations of expert trajectories:

$$\begin{aligned} \max_{\boldsymbol{\omega}} \quad & \int -p(\boldsymbol{\xi}|\boldsymbol{\omega}) \log p(\boldsymbol{\xi}|\boldsymbol{\omega}) \\ \text{s.t.} \quad & \boldsymbol{f}_{\text{Exp}} = \int p(\boldsymbol{\xi}|\boldsymbol{\omega}) \boldsymbol{f}(\boldsymbol{\xi}) \\ & \int p(\boldsymbol{\xi}|\boldsymbol{\omega}) = 1 \end{aligned} \quad (5)$$

The problem can be solved by Lagrangian multiplier method:

$$L(\boldsymbol{\omega}) = \int p \log p - \sum_{h=1}^{H} \lambda_h \left(\int p f^h - f^h_{\text{Exp}} \right) - \lambda_0 \left(\int p - 1 \right) \tag{6}$$

where $\lambda_h (h = 1, .., H)$ are auxiliary parameters. Let the partial differential for p be equal to 0, then the solution of the constrained optimization problem in (5) has the form:

$$p(\boldsymbol{\xi}|\boldsymbol{\omega}) = \frac{1}{Z} e^{-\sum_{h=1}^{H} \lambda_h f^h} \sim e^{-\boldsymbol{\omega}^T \boldsymbol{f}(\boldsymbol{\xi})} \tag{7}$$

where Z is the normalization factor and auxiliary parameters are associated with weights. One can interpret $\boldsymbol{\omega}^T \boldsymbol{f}(\boldsymbol{\xi})$ as a cost function, where drivers are exponentially more likely to select trajectories with lower cost.

The weights can be solved by maximum likelihood method. Usually, it is not possible to compute $\boldsymbol{\omega}$ analytically, but the gradient of the Lagrangian function of the constraint optimization problem with respect to $\boldsymbol{\omega}$ can be computed. It proves that this gradient is the difference of feature expectations between the selected trajectories and the expert ones:

$$\nabla L(\boldsymbol{\omega}) = \boldsymbol{f}_{\text{Exp}} - \int p(\boldsymbol{\xi}|\boldsymbol{\omega}) \boldsymbol{f}(\boldsymbol{\xi}) \tag{8}$$

Unfortunately, it is difficult to compute $E_{p(\boldsymbol{\xi}|\boldsymbol{\omega})}[\boldsymbol{f}] = \int p(\boldsymbol{\xi}|\boldsymbol{\omega}) \boldsymbol{f}(\boldsymbol{\xi})$ due to the curse of dimensionality of trajectory states in the integral. An alternative approach is to compute the feature values of the most likely trajectory as an approximation of the feature expectation [14]:

$$E_{p(\boldsymbol{\xi}|\boldsymbol{\omega})}[\boldsymbol{f}] \approx \boldsymbol{f}(\underset{\boldsymbol{\xi}}{\text{argmax}}\ p(\boldsymbol{\xi}|\boldsymbol{\omega})) \tag{9}$$

With this approximation, only the optimal trajectory associated to the optimal policy is needed, in contrast to regarding the generated trajectories as a probability distribution. The test results in Sec. V also corroborate this assumption in the reward design for autonomous driving.

III. IRL with Pre-sampled Policies

A. Proposed Method

The objective of IRL is to find the optimal weights $\boldsymbol{\omega}^*$ based on which reward function yields to an optimal policy with its optimal trajectories whose feature expectations are the same as the expert's. The optimal reward is usually obtained by gradient descend method, and the main issue of IRL is the low efficiency of finding such an optimal policy at each iteration, because it usually means involvement of reinforcement learning which is very time-consuming. Therefore, the goal of the proposed method is to avoid calling RL repeatedly each time when updating weights. An intuitive idea is to randomly sample a massive of policies in advance and then to pick one of them as the optimal policy instead of finding it via RL. Fig.1 shows the concept of the proposed pre-sampled IRL method. The method consists of two procedures:

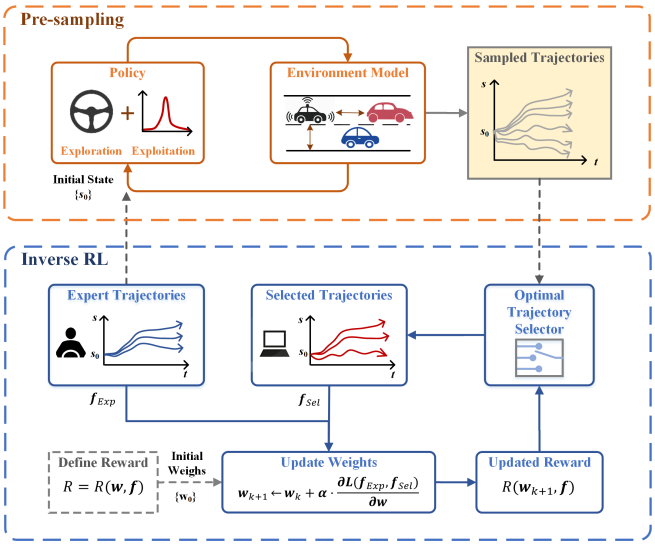


Fig. 1. Concept of the proposed IRL method

(1) To generate a large candidate trajectory library with random policy. The sample space is designed with human knowledge and becomes a relative small subspace of policy, thus making it faster to target the optimal policy. Then, with randomly sampled policies in the sub-space, candidate trajectories can be generated given the same initial states as the expert's and the environment model.

(2) To update reward weights with selected trajectories. The above-mentioned step is only executed once. When learning, an optimal trajectory selector is designed to locate the optimal policy and its corresponding trajectories from the candidate trajectory library at each iteration. Then, a gradient descent updater is used for weights updating based on the designed likelihood function with respect to the selected trajectories and the expert ones.

B. Trajectory Generator with Random Policy

A closed loop system with policy and environment model is constructed for trajectory generation. The policy here is not only limited to RL, but also open to control-based methods or search-based methods. Note that human drivers often selects action in a specific policy space, therefore, a subspace of policy $\Pi_{\text{Sub}} \subset \Pi$ is selected by using human knowledge of vehicle controllers. As shown in Fig.2, the selected subspace

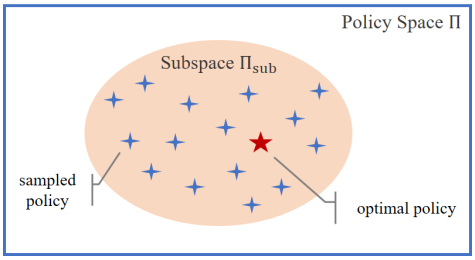


Fig. 2. Policy pre-sampling in designed subspace

shall be the neighbor of the optimal policy which contains the optimal policy within it. The smaller the Π_{Sub} is chosen, the faster speed IRL can achieve to target the optimal policy.

By randomly sampling in Π_{Sub} for M times (large enough), a set of candidate policies are obtained:

$$\{\pi_j^{\mathrm{Gen}} \in \Pi_{\mathrm{Sub}},\ j = 1, ..., M\} \tag{10}$$

where each π_j^{Gen} can generate N corresponding candidate trajectories:

$$\begin{aligned}
\boldsymbol{\xi}_{1,j}^{\mathrm{Gen}} &= \{(\boldsymbol{s}_0^{(1,j)}, \boldsymbol{a}_0^{(1,j)}), ..., (\boldsymbol{s}_{T_i}^{(1,j)}, \boldsymbol{a}_{T_i}^{(1,j)})\} \\
&\vdots \\
\boldsymbol{\xi}_{i,j}^{\mathrm{Gen}} &= \{(\boldsymbol{s}_0^{(i,j)}, \boldsymbol{a}_0^{(i,j)}), ..., (\boldsymbol{s}_{T_i}^{(i,j)}, \boldsymbol{a}_{T_i}^{(i,j)})\} \\
&\vdots \\
\boldsymbol{\xi}_{N,j}^{\mathrm{Gen}} &= \{(\boldsymbol{s}_0^{(N,j)}, \boldsymbol{a}_0^{(N,j)}), ..., (\boldsymbol{s}_{T_i}^{(N,j)}, \boldsymbol{a}_{T_i}^{(N,j)})\}
\end{aligned} \tag{11}$$

Note that each of them starts from the same initial state as the expert trajectory, i.e., $\boldsymbol{s}_0^{(i,j)} = \boldsymbol{s}_0^{(i)}$ and moves abide by sampled policy and environment model:

$$\boldsymbol{a}_t^{(i,j)} = \pi_j^{\mathrm{Gen}}(\boldsymbol{s}_t^{(i,j)}) \tag{12}$$

$$\boldsymbol{s}_{t+1}^{(i,j)} = F(\boldsymbol{s}_t^{(i,j)}, \boldsymbol{a}_t^{(i,j)}) \tag{13}$$

where $F(\cdot, \cdot)$ is a representation of environment dynamics. In IRL, it is often selected to be a mathematical model for fast computation in simulation. In fact, actual environment can be also used so long as sampled policy is feasible.

Note that in very few cases the optimal policy may locates at the edge of or even outside Π_{Sub}. Therefore, to cover all different kinds of policies, the sampling mechanism of driving policies can be designed to have the ability for policy exploration at certain probability as well.

C. Learner of Reward Weights

The learning process of reward weights includes two parts: (1) optimal trajectory selector, which is designed to select trajectories for comparison with the expert ones, and (2) gradient descent updater, which updates the reward weights by maximizing an likelihood function.

The goal of the optimal trajectory selector is to find the optimal trajectories $\{\boldsymbol{\xi}_i^{\mathrm{Sel}}, i = 1, ..., N\}$, given the initial state of each expert trajectory, which originally generated by the optimal policy solved iteratively through RL each time when updating the weights.

According to (7), the probability of trajectory under current weights is $p(\boldsymbol{\xi}|\boldsymbol{\omega}) = e^{-\boldsymbol{\omega}^T \boldsymbol{f}(\boldsymbol{\xi})}$. It is known that the optimal trajectory generated by the optimal policy must have the highest probability. Therefore, for a candidate trajectory in the library, the higher the probability of its occurrence, the closer it is to the optimal trajectory. Hence, the optimal trajectory selection mechanism is designed as follows:

$$\boldsymbol{\xi}_i^{\mathrm{Sel}} = \boldsymbol{\xi}_{i,j^*}^{\mathrm{Gen}} \sim \underset{j}{\operatorname{argmax}} \frac{1}{N} \sum_{i=1}^{N} p(\boldsymbol{\xi}_{i,j}^{\mathrm{Gen}}|\boldsymbol{\omega}) \tag{14}$$

where the trajectory with the highest probability is selected as the optimal trajectory given pre-sampled policies.

Then, the feature expectations of the selected trajectories can be approximated by calculating the average feature values for all the selected optimal trajectories:

$$\boldsymbol{f}_{\mathrm{Sel}} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{f}(\boldsymbol{\xi}_{i,j*}^{\mathrm{Gen}}) \tag{15}$$

and (8) is rewritten as:

$$\nabla L(\boldsymbol{\omega}) = \boldsymbol{f}_{\mathbf{Exp}} - \boldsymbol{f}_{\mathbf{Sel}} \tag{16}$$

with which weights are updated by gradient descent method:

$$\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} + \alpha \cdot \nabla L(\boldsymbol{\omega}) \tag{17}$$

where α is learning rate. Like many other methods, the optimal weights can be obtained by repeating the above-mentioned steps until convergence.

IV. Implementation for Autonomous Driving

The proposed pre-sampled IRL method is applied to the design of the reward function for autonomous driving on highway. This section introduces the details of the environment model and the random policy generator in the pre-sampling part, as well as the design of the reward function and the optimal trajectory selector in the learning part.

A. Environment Model

In this paper, surrounding vehicles are driven according to the track recorded in the data set instead of being predicted each time during learning [15]. For the autonomous vehicle, the kinematic bicycle model is used and the continuous non-linear equations that describe the kinematic bicycle model in an inertial frame are:

$$\dot{y} = v \cos(\beta + \psi) \tag{18}$$

$$\dot{x} = v \sin(\beta + \psi) \tag{19}$$

$$\dot{\psi} = \frac{\sin \beta}{l_r} v \tag{20}$$

$$v \cos \beta = v_f \cos \delta_f \tag{21}$$

where x and y are coordinates of vehicle centroid, ψ is heading, v is speed, l_r is distance from centroid to rear axles, β is angle of velocity with respect to longitudinal axis of the vehicle. It is assumed that vehicle centroid coincides with geometric center and initial front wheel angle of any trajectory is zero.

Note that the control inputs of the vehicle model are front wheel steering angle δ_f, and its acceleration $\dot{v_f} = a_f$. The vehicle model is implemented in the decretized form.

B. Random Policy Sampling

As introduced in Sec. III, it is unnecessary to use RL-based policies. In this study, linear controllers with uncertain gains are chosen as the policy subspace Π_{Sub} as they are widely used in vehicle control [16]. Since only longitudinal scenarios is evaluated, the longitudinal controller is designed as follows:

$$a_f = \boldsymbol{K}_{\mathrm{Lon}}^T \boldsymbol{s}_{\mathrm{Lon}} \tag{22}$$

$$= K_{\mathrm{Lon}}^1 \Delta d_{\mathrm{Lon}} + K_{\mathrm{Lon}}^2 \Delta v_{\mathrm{Lon}} + K_{\mathrm{Lon}}^3 \Delta a_{\mathrm{Lon}} \tag{23}$$

where $\boldsymbol{K}_{\mathrm{Lon}}$ is vector for controller' feedback gains; a_{Lon} is longitudinal acceleration of the autonomous vehicle, Δd_{Lon}, Δv_{Lon}, and Δa_{Lon} are relative longitudinal distance, speed, and acceleration to front vehicle.

There are several advantages in choosing vehicle controller with uncertain feedback gains as a stochastic policy space: (1) the vehicle controllers are designed based on driver models, which contains abundant human prior knowledge. The policy subspace with respect to feedback gains can be regarded as a neighbor of the optimal policy, which greatly reduces the policy search space and improves the efficiency when targeting the optimal policy; (2) The vehicle controllers naturally cooperate with the aforementioned vehicle model to connect states with actions, so that the change of vehicle states obeys the constraint of the controller inputs, thus significantly compressing the generated trajectory state space. (3) Compared with RL-based policy with thousands of neural network parameters, linear controllers have intuitive and explicit mathematical expression with only a few feedback gains, which can facilitates the generation of random policy by simply sampling in them.

Based on the designed controllers, driving policies can be generated by random sampling in the feedback gains space. Just in case the sampled policy set doesn't contain the optimal policy, exploration of policy is introduced as well for supplement. Regardless of current states, actions associated are selected randomly in the entire action space:

$$\boldsymbol{a}_t^{(i,j)} = random(\boldsymbol{a}) \tag{24}$$

which can be regarded as an unbiased Monte Carlo sampling of policy in the whole policy space. According to the law of large numbers, the sampled policy set, when sampling enough times, shall contain the optimal strategy.

C. Features for Reward Function

The pre-sampled IRL method is applied to highway scenarios. Features are proposed that capture relevant properties of driving behaviors related to aspects of safety, comfort, efficiency and targeted driving maneuvers. The specific features are defined as following:

1) Front Vehicle Time Headway: the relative spatial relation to the front vehicle considering the ego vehicle speed:

$$THW_{\mathrm{f}} = |\frac{y_{\mathrm{front}} - y}{\dot{y}} - THW_{\mathrm{f}}^{\mathrm{des}}| \tag{25}$$

Algorithm 1 Pre-sampled IRL for Autonomous Driving

Input: Expert trajectories $\{\boldsymbol{\xi}_1^{\mathrm{Exp}}, ..., \boldsymbol{\xi}_N^{\mathrm{Exp}}\}$
Output: Learned weights $\boldsymbol{\omega}^*$ for the reward function

Random Trajectory Generation
Construct the policy subspace Π_{sub} for autonomous driving with (22)
Randomly sampling in Π_{sub} with exploration to obtain $\pi_j^{\mathrm{Gen}}, j = 1, ..., M$
for each expert trajectory $bm\xi_i^{\mathrm{Exp}}$ **do**
    Extract initial states $\boldsymbol{s}_0^{(i)}$ for each candidate trajectory
    Generate candidate trajectory $\boldsymbol{\xi}_{i,j}^{\mathrm{Gen}}$ with sampled policy π_j^{Gen} and the environment model in (18)
Reward Weighs Learning
Compute the empirical feature vector averaged over all expert trajectories $\boldsymbol{f}_{\mathrm{Exp}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{f}(\boldsymbol{\xi}_i^{\mathrm{Exp}})$
Initialize the weight vector $\boldsymbol{\omega}_0$ with arbitrary values
while not convergent **do**
    Computer probability with respect to reward for all generated candidate trajectories
    Select optimal trajectories $\boldsymbol{\xi}_i^{\mathrm{Sel}}$ according to (14)
    Compute approximated feature expectation for selected trajectories $\boldsymbol{f}_{\mathrm{Sel}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{f}(\boldsymbol{\xi}_i^{\mathrm{Sel}})$
    Use the gradient in (16) to update weights
return

where y_{front} is longitudinal position of front vehicle, y and $\dot{y}$ are longitudinal position and speed of the ego vehicle, $THW_{\mathrm{f}}^{\mathrm{des}}$ is desired time headway to front vehicle.This feature is related to driving safety.

2) Longitudinal acceleration $|\ddot{y}|$, which is related to driving comfort and targeted driving maneuver.

3) Desired Speed: the ego vehicle is expected to drive at desired speed v_{des}, which is set as speed limit:

$$f_{\mathrm{des}} = |v - v_{\mathrm{des}}| \tag{26}$$

$\boldsymbol{Algorithm\ 1}$ presents the proposed pre-sampled IRL method for autonomous driving reward design.

V. Simulation Data

The car-following driver model (policy) has been fully studied while lane change model (policy) needs further research. Therefore, the typical car-following scenario with the ego vehicle and one front car is chosen as the simulation scenario. As illustrated in Fig.3, when a driver keeps driving in one lane, he or she shall consider the impact from its front vehicle.

The simulation data for expert trajectories used in this paper is simulated using modified linear car-following (MLCF) model, which has been widely used as car-following policy [17] recently. The MLCF model is as follows:

$$a_f = SVE \cdot k_V \cdot (v_f - v) + SDE \cdot k_D \cdot (y - y_{\mathrm{des}}) \tag{27}$$

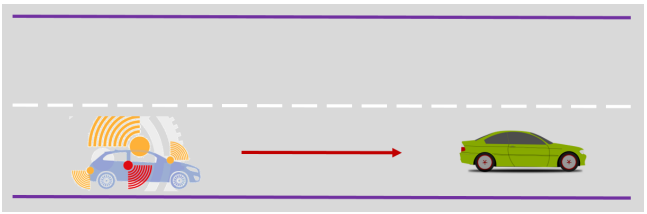


Fig. 3. Illustration of car-following scenario

where $k_V = 0.2583$ and $k_D = 0.0125$ are nominal control gains, v_f is front vehicle speed. y_{des}, SVE is driver sensitivity to velocity error, SDE is driver sensitivity to distance error, and they are described as:

$$y_{\text{des}} = r \cdot v^2 + T_h \cdot v + y_0 \tag{28}$$

$$SVE^{-1} = k_{SVE} v_f + d_{SVE} \tag{29}$$

$$SDE^{-1} = k_{SDE} v_f + d_{SDE} \tag{30}$$

where $y_0 = 5.6\text{m}$ is the stopping offset, $T_h = 2.4\text{s}$ is the equivalent to the time headway, $r = -0.006\text{m/s}^2$ is a quadratic coefficient, reflecting T_hs changes at high speed, and $k_{SVE} = -0.002$, $d_{SVE} = 1.025$, $k_{SDE} = 0.053$ and $d_{SDE} = 0.339$ are model coefficients.

The initial relative position between the ego vehicle and the front vehicle is randomly chosen from a normal distribution $\sigma(40\text{m}, 6\text{m})$ and the initial speed for both vehicles is randomly chosen from a normal distribution $\sigma(10\text{m/s}, 5\text{m/s})$. The front vehicle speed obeys peirodical fluctuation $10 + 5\sin(2\pi t/10)$ during simulation while the ego vehicle speed is controlled by (22). In total, the simulated data set contains 100 expert trajectories, each with a length of 60s and a sampling rate at 10Hz.

The subspace π_{sub} is constructed based on (23), where the controller gains are within $[0, 0.01]$, $[0, 0.5]$, and $[0, 0.5]$. The Monte Carlo sampling method is then applied to randomly generate 1,000,000 candidate policies. Note that, there is 10% probability that the randomly generated controller gains could go beyond the pre-set range and another 10% probability that the acceleration is randomly chosen from $[-2.5\text{m/s}^2, 2.5\text{m/s}^2]$ instead of being calculated from (23).

VI. Results

The proposed method was tested on the data set simulated above with the vehicle kinematic model and feature normalized to $[0, 1]$. The learning process was conducted on a laptop with a 2.5GHz CPU and 8Gb RAM. The learning rate is 1.0 initially and adapted consequently. The proposed method was evaluated in terms of learning efficiency, learned reward weights reasonableness and recovered trajectory effect.

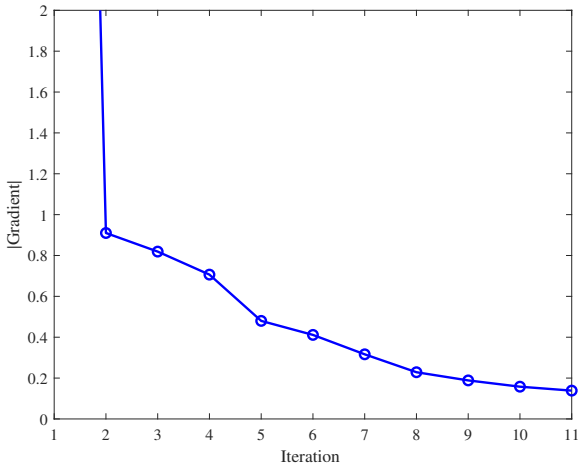


Fig. 4. Gradient convergence curve

A. Learning Efficiency

The convergence curve of gradient is shown in Fig.4. The $+\infty$-norm of gradient of the proposed method converged rapidly from around 100 to nearly zero (< 0.15) within 11 iterations. That shows the relationship $E_{p(\boldsymbol{\xi}_i^{\text{Sel}}|\boldsymbol{\omega}^*)} \approx \boldsymbol{f}_{\text{Exp}}$ is established with final optimal weights obtained, which means the distribution generated with them is similar to the distribution of expert's.

From another perspective, the time used during learning by the proposed method is significantly small. Table I shows the total number of iterations, total time and average iteration time consumed by the proposed method. It can be seen from the table that the proposed method only used 17.85s for online weights updating, which increased linearly with iteration. Even though the pre-sampling time consumed 239.11s, the proposed method still saves tremendous learning time and accelerating the learning process.

B. Learned Weights

The initial weights is $[0, 0, 0]$ as the first candidate policy is chosen as the optimal policy. The learned feature weighs are expected to encode the trade-off between antagonistic goals, such as the desired speed and limited accelerations. Table II shows the weights learned by the proposed pre-sampled IRL algorithm while that the corresponding gains value for the longitudinal controller K^1_{Lon}, K^2_{Lon}, and K^3_{Lon} are 0.0018, 0.280, and 0.230.

Among them, the time headway to the front vehicle is related to safety, which means that drivers put safety as the first priority during driving. Comfort comes as the second as

TABLE I
Time Spent for Learning

Method	Total Iteration	Total Time	Time/Iteration
Pre-sampled IRL	11	17.85	1.62s

TABLE II
Learned Weights

Feature	THW_{f}	$\lvert\ddot{y}\rvert$	$\lvert v - v_{\text{des}}\rvert$
Weight	0.4491	0.3784	0.2961

TABLE III
RMSE FOR LONGITUDINAL POSITION

Method	Max	Min	Mean
Pre-sampled IRL	5.23m	0.00m	2.14m

the longitudinal acceleration of vehicles affects the comfort of passengers. Compliance comes last as the weight of the deviation from the expected speed is the smallest. One possible reason is that the radical driving behavior like overtaking is restrained under car-following scenario, which leads to drivers' maintaining a relatively stable speed in the course of driving.

C. Recovered Trajectories

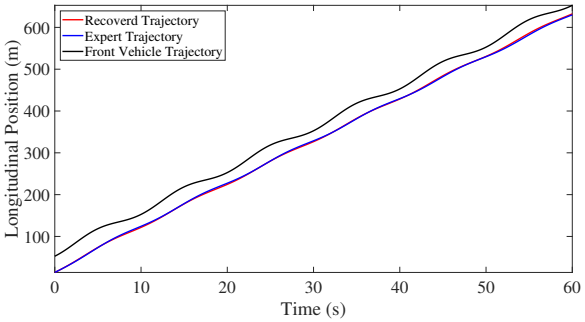
The learned weights are implemented with the corresponding driving policy for autonomous driving in the car-following scenario, which yields to smooth, comfortable trajectories, while keeping a safe distance to the front vehicle. Fig.5 shows the comparison of expert vehicle and learner vehicle, in terms of car-following trajectory, speed, acceleration and time headway (the latter three are related to features designed). When the ego vehicle drove in the lane, it tried to balance the different feature reward. As a consequence, the time headway feature kept a safe distance between the ego vehicle and the front vehicle during driving. Meanwhile, the features that capture the longitudinal acceleration and speed deviation guaranteed a comfort and reasonable trajectory that showed characteristics as the expert's.

Quantitatively speaking, the trajectories recovered using the proposed methods is compared with the expert trajectories in terms of root mean square error (RMSE). Shown in Table III, RMSE is calculated for all recovered trajectories with max, min and mean error being 5.23m, 0m and 2.14m. The means the average car-following error is less then one car length while the max error is no more than twice the car length when the average speed is 10m/s.

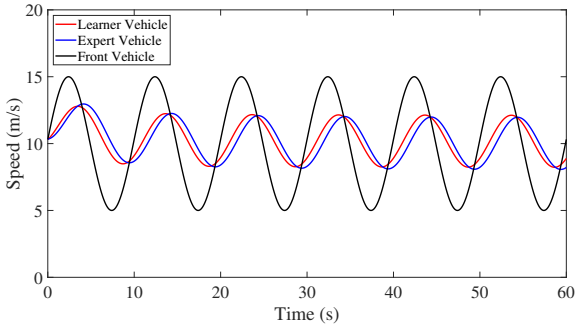
VII. CONCLUSION

A pre-sampled inverse reinforcement learning method for autonomous driving reward design is proposed in this paper. The candidate trajectory library is generated by pre-sampling in the designed driving policy subspace, while the corresponding optimal trajectories at each iteration for weights updating are chosen via the designed optimal trajectory selection mechanism. The traditional RL module for obtaining optimal policy is avoided for generating optimal trajectories for matching feature expectations of the expert's, thus accelerating the learning process. Results show that it only took 11 iterations to converge, while the average longitudinal RMS error of trajectories recovered by the learned reward is 2.14m compared to the expert trajectories.

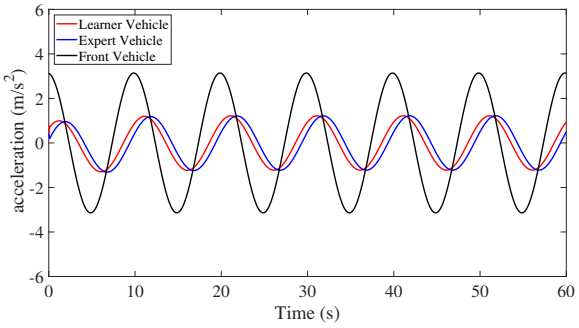
This study provides a promising approach to accelerating the IRL learning process for problems like autonomous driving with pre-designed policy subspace by human knowledge. The preliminary research opens various perspectives



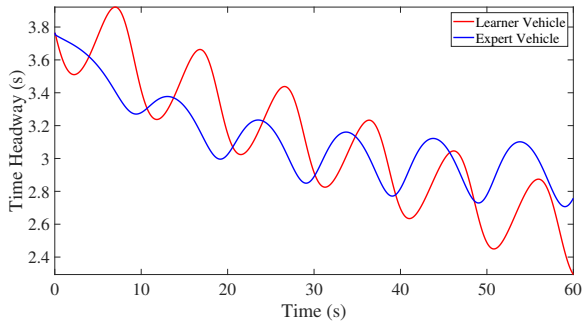
(a) Trajectory



(b) Speed



(c) Acceleration



(d) Time Headway

Fig. 5. Car-following scenario comparison of expert vehicle and learner vehicle

for future research: (1) to generalize the proposed method in different driving scenarios, e.g. intersections and unstructured roads; (2) to generalize the proposed method under a distributed learning framework as it does for RL [18]; (3) to address the learning problem as a stochastic one which

requires probability distributions of generated trajectories and changing environments.

REFERENCES

[1] P. Wang, C.-Y. Chan, and A. de La Fortelle, "A reinforcement learning based approach for automated lane change maneuvers," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1379–1384.

[2] Y. Guan, S. E. Li, J. Duan, W. Wang, and B. Cheng, "Markov probabilistic decision making of self-driving cars in highway with random traffic flow: a simulation study," *Journal of Intelligent and Connected Vehicles*, vol. 1, no. 2, pp. 77–84, 2018.

[3] A. Y. Ng, S. J. Russell *et al.*, "Algorithms for inverse reinforcement learning," in *ICML*, 2000, pp. 663–670.

[4] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the 21th International Conference on Machine learning*. ACM, 2004, p. 1.

[5] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "Maximum margin planning," in *Proceedings of the 23rd International Conference on Machine Learning*. ACM, 2006, pp. 729–736.

[6] E. Klein, M. Geist, B. Piot, and O. Pietquin, "Inverse reinforcement learning through structured classification," in *Advances in Neural Information Processing Systems*, 2012, pp. 1007–1015.

[7] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *AAAI*, 2008, pp. 1433–1438.

[8] A. Boularias, J. Kober, and J. Peters, "Relative entropy inverse reinforcement learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 182–189.

[9] E. T. Jaynes, "Information theory and statistical mechanics," *Physical Review*, vol. 106, no. 4, p. 620, 1957.

[10] M. Babes, V. Marivate, K. Subramanian, and M. L. Littman, "Apprenticeship learning about multiple intentions," in *Proceedings of the 28th International Conference on Machine Learning (ICML)*, 2011, pp. 897–904.

[11] M. Kuderer, S. Gulati, and W. Burgard, "Learning driving styles for autonomous vehicles from demonstration," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2641–2646.

[12] M. Wulfmeier, P. Ondruska, and I. Posner, "Maximum entropy deep inverse reinforcement learning," *arXiv preprint arXiv:1507.04888*, 2015.

[13] J. Duan, S. E. Li, B. Cheng, Y. Luo, and K. Li, "Hierarchical reinforcement learning for decision making of self-driving cars without reliance on labeled driving data," in *the 14th International Symposium on Advanced Vehicle Control (AVEC)*, Beijing, 2018.

[14] M. Kuderer, H. Kretzschmar, C. Sprunk, and W. Burgard, "Feature-based prediction of trajectories for socially compliant navigation." in *Robotics: science and systems*, 2012.

[15] L. Xin, P. Wang, C.-Y. Chan, J. Chen, S. E. Li, and B. Cheng, "Intention-aware long horizon trajectory prediction of surrounding vehicles using dual lstm networks," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 1441–1446.

[16] P. Wang, T. Shi, C. Zou, L. Xin, and C.-Y. Chan, "A data driven method of feedforward compensator optimization for autonomous vehicle control," *arXiv preprint arXiv:1906.02277*, 2019.

[17] S. Li, J. Wang, K. Li, X. Lian, H. Ukawa, and D. Bai, "Modeling and verification of heavy-duty truck drivers car-following characteristics," *International journal of automotive technology*, vol. 11, no. 1, pp. 81–87, 2010.

[18] S. E. Li, C. Liu, Y. Zheng, J. Duan, and L. Keqiang, "Distributed sensing, learning, and control for connected and automated vehicles," *Science (Special Supplement)*, pp. 42–44, 2018.