


Learning to Simulate on Sparse Trajectory Data

Hua Wei¹, Chacha Chen¹, Chang Liu², Guanjie Zheng¹, and Zhenhui Li¹

¹ Pennsylvania State University, University Park, PA 16802, USA
{hzw77, gjz5038, jessiel1}@ist.psu.edu, cjc6647@psu.edu

² Shanghai Jiao Tong University, Shanghai, China
only-changer@sjtu.edu.cn

Abstract. Simulation of the real-world traffic can be used to help validate the transportation policies. A good simulator means the simulated traffic is similar to real-world traffic, which often requires dense traffic trajectories (i.e., with high sampling rate) to cover dynamic situations in the real world. However, in most cases, the real-world trajectories are sparse, which makes simulation challenging. In this paper, we present a novel framework *ImIn-GAIL* to address the problem of learning to simulate the driving behavior from sparse real-world data. The proposed architecture incorporates data interpolation with the behavior learning process of imitation learning. To the best of our knowledge, we are the first to tackle the data sparsity issue for behavior learning problems. We investigate our framework on both synthetic and real-world trajectory datasets of driving vehicles, showing that our method outperforms various baselines and state-of-the-art methods.

Keywords: imitation learning · data sparsity · interpolation

1 Introduction

Simulation of the real world is one of the feasible ways to verify driving policies on autonomous vehicles and transportation policies like traffic signal control [22, 23, 25] or speed limit setting [27] since it is costly to validate them in the real world directly [24]. The driving behavior model, i.e., how the vehicle accelerates/decelerates, is the critical component that affects the similarity of the simulated traffic to the real-world traffic [7, 9, 14]. Traditional methods to learn the driving behavior model usually first assumes that the behavior of the vehicle is only influenced by a small number of factors with predefined rule-based relations, and then calibrates the model by finding the parameters that best fit the observed data [5, 16]. The problem with such methods is that their assumptions oversimplify the driving behavior, resulting in the simulated driving behavior far from the real world.

In contrast, imitation learning (IL) does not assume the underlying form of the driving behavior model and directly learns from the observed data (also called demonstrations from expert policy in IL literature). With IL, a more sophisticated driving behavior policy can be represented by a parameterized model like neural nets and provides a promising way to learn the models that behave



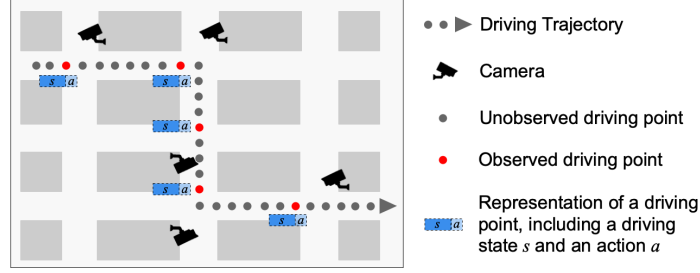


Fig. 1: Illustration of a driving trajectory. In the real-world scenario, only part of the driving points can be observed and form a sparse driving trajectory (in red dots). Each driving point includes a driving state and an action of the vehicle at the observed time step. Best viewed in color.

similarly to expert policy. Existing IL methods (e.g., behavior cloning [13, 21] and generative adversarial imitation learning [4, 3, 18, 30]) for learning driving behavior relies on a large amount of behavior trajectory data that consists of dense vehicle driving states, either from vehicles installed with sensors, or road-side cameras that capture the whole traffic situation (including every vehicle driving behavior at every moment) in the road network.

However, in most real-world cases, the available behavior trajectory data is sparse, i.e., the driving behavior of the vehicles at every moment is difficult to observe. It is infeasible to install sensors for every vehicle in the road network or to install cameras that cover every location in the road network to capture the whole traffic situation. Most real-world cases are that only a minimal number of cars on the road are accessible with dense trajectory, and the driving behavior of vehicles can only be captured when the vehicles drive near the locations where the cameras are installed. For example, in Figure 1, as the cameras are installed only around certain intersections, consecutive observed points of the same car may have a large time difference, resulting in a sparse driving trajectory. As data sparsity is considered as a critical issue for unsatisfactory accuracy in machine learning, directly using sparse trajectories to learn the driving behavior could make the model fail to learn the behavior policy at the unobserved states.

To deal with sparse trajectories, a typical approach is to interpolate the sparse trajectories first and then learn the model with the dense trajectories [10, 28, 31]. This two-step approach also has an obvious weakness, especially in the problem of learning behavior models. For example, linear interpolation is often used to interpolate the missing points between two observed trajectory points. But in real-world cases, considering the interactions between vehicles, the vehicle is unlikely to drive at a uniform speed during that unobserved time period, hence the interpolated trajectories may be different from the true trajectories. However, the true trajectories are also unknown and are exactly what we aim to imitate. A better approach is to integrate interpolation with imitation because they should inherently be the same model. To the best of our knowledge, none of the existing



literature has studied the real-world problem of learning driving policies from sparse trajectory data.

In this paper, we present *ImIn-GAIL*, an approach that can learn the driving behavior of vehicles from observed sparse trajectory data. *ImIn-GAIL* learns to mimic expert behavior under the framework of generative adversarial imitation learning (GAIL), which learns a policy that can perform expert-like behaviors through rewarding the policy for deceiving a discriminator trained to classify between policy-generated and expert trajectories. Specifically, for the data sparsity issue, we present an interpolator-discriminator network that can perform both the interpolation and discrimination tasks, and a downsampler that draws supervision on the interpolation task from the trajectories generated by the learned policy. We conduct experiments on both synthetic and real-world data, showing that our method can not only have excellent imitation performance on the sparse trajectories but also have better interpolation results compared with state-of-the-art baselines. The main contributions of this paper are summarized as follows:



- We propose a novel framework *ImIn-GAIL*, which can learn driving behaviors from the real-world sparse trajectory data.
- We naturally integrate the interpolation with imitation learning that can interpolate the sparse driving trajectory.
- We conduct experiments on both real and synthetic data, showing that our approach significantly outperforms existing methods. We also have interesting cases to illustrate the effectiveness on the imitation and interpolation of our methods.

2 Preliminaries

Definition 1 (Driving Point). A driving point $\tau^t = (s^t, a^t, t)$ describes the driving behavior of the vehicle at time t , which consists of a driving state s^t and an action a^t of the vehicle. Typically, the state s^t describes the surrounding traffic conditions of the vehicle (e.g., speed of the vehicle and distance to the preceding vehicle), and the action $a^t \sim \pi(a|s^t)$ the vehicle takes at time t is the magnitude of acceleration/deceleration following its driving policy $\pi(a|s^t)$.

Definition 2 (Driving Trajectory). A driving trajectory of a vehicle is a sequence of driving points generated by the vehicle in geographical spaces, usually represented by a series of chronologically ordered points, e.g. $\tau = (\tau^{t_0}, \dots, \tau^{t_N})$.

In trajectory data mining [11, 12, 32], a *dense trajectory* of a vehicle is the driving trajectory with high-sampling rate (e.g., one point per second on average), and a *sparse trajectory* of a vehicle is the driving trajectory with low-sampling rate (e.g., one point every 2 minutes on average). In this paper, the observed driving trajectory is a sequence of driving points with large and irregular intervals between their observation times.

Problem 3. In our problem, a vehicle observes state s from the environment, take action a following policy π^E at every time interval Δt , and generate a raw driving trajectory τ during certain time period. While the raw driving trajectory is dense (i.e., at a high-sampling rate), in our problem we can only observe a set of sparse trajectories \mathcal{T}_E generated by expert policy π^E as expert trajectory, where $\mathcal{T}_E = \{\tau_i | \tau_i = (\tau_i^{t_0}, \dots, \tau_i^{t_N})\}$, $t_{i+1} - t_i \gg \Delta t$ and $t_{i+1} - t_i$ may be different for different observation time i . Our goal is to learn a parameterized policy π_θ that imitates the expert policy π^E .

3 Method

In this section, we first introduce the basic imitation framework, upon which we propose our method (*ImIn-GAIL*) that integrates trajectory interpolation into the basic model.

3.1 Basic GAIL Framework

In this paper, we follow the framework similar to GAIL [4] due to its scalability to the multi-agent scenario and previous success in learning human driver models [8]. GAIL formulates imitation learning as the problem of learning policy to perform expert-like behavior by rewarding it for “deceiving” a classifier trained to discriminate between policy-generated and expert state-action pairs. For a neural network classifier \mathcal{D}_ψ parameterized by ψ , the GAIL objective is given by $\max_\psi \min_\theta \mathcal{L}(\psi, \theta)$ where $\mathcal{L}(\psi, \theta)$ is :

$$\mathcal{L}(\psi, \theta) = \mathbb{E}_{(s,a) \sim \tau \in \mathcal{T}_E} \log \mathcal{D}_\psi(s, a) + \mathbb{E}_{(s,a) \sim \tau \in \mathcal{T}_G} \log(1 - \mathcal{D}_\psi(s, a)) - \beta H(\pi_\theta) \quad (1)$$

where \mathcal{T}_E and \mathcal{T}_G are respectively the expert trajectories and the generated trajectories from the interactions of policy π_θ with the simulation environment, $H(\pi_\theta)$ is an entropy regularization term.

- *Learning ψ :* When training \mathcal{D}_ψ , Equation (1) can simply be set as a sigmoid cross entropy where positive samples are from \mathcal{T}_E and negative samples are from \mathcal{T}_G . Then optimizing ψ can be easily done with gradient ascent.

- *Learning θ :* The simulator is an integration of physical rules, control policies and randomness and thus its parameterization is assumed to be unknown. Therefore, given \mathcal{T}_G generated by π_θ in the simulator, Equation (1) is non-differentiable w.r.t θ . In order to learn π_θ , GAIL optimizes through reinforcement learning, with a surrogate reward function formulated from Equation (1) as:

$$\tilde{r}(s^t, a^t; \psi) = -\log(1 - \mathcal{D}_\psi(s^t, a^t)) \quad (2)$$

Here, $\tilde{r}(s^t, a^t; \psi)$ can be perceived to be useful in driving π_θ into regions of the state-action space at time t similar to those explored by π^E . Intuitively, when the observed trajectory is dense, the surrogate reward from the discriminator in Equation (2) is helpful to learn the state transitions about observed trajectories.

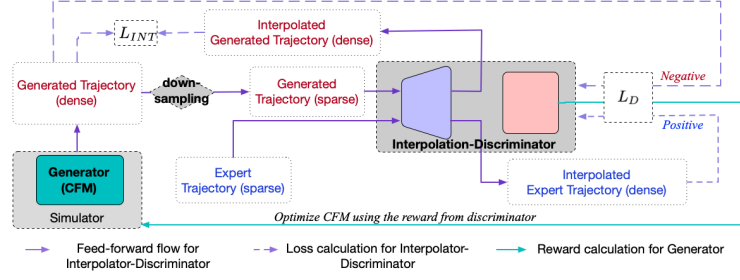


Fig. 2: Proposed *ImIn-GAIL* Approach. The overall framework of *ImIn-GAIL* includes three components: generator, downsampler, and interpolation-discriminator. Best viewed in color.

However, when the observed data is sparse, the reward from discriminator will only learn to correct the observed states and fail to model the behavior policy at the unobserved states. To relieve this problem, we propose to interpolate the sparse expert trajectory within the based imitation framework.

3.2 Imitation with Interpolation

An overview of our proposed Imitation-Interpolation framework (*ImIn-GAIL*) is shown in Figure 2, which consists of the following three key components.

Generator in the simulator Given an initialized driving policy π_θ , the dense trajectories \mathcal{T}_G^D of vehicles can be generated in the simulator. In this paper, the driving policy π_θ is parameterized by a neural network which will output an action a based on the state s it observes. The simulator can generate driving behavior trajectories by rolling out π_θ for all vehicles simultaneously in the simulator. The optimization of the driving policy is optimized via TRPO [17] as in vanilla GAIL [4].

Downsampling of generated trajectories The goal of the downsampler is to construct the training data for interpolation, i.e., learning the mapping from a sparse trajectory to a dense one. For two consecutive points (i.e., τ^{t_s} and τ^{t_e} in generated sparse trajectory \mathcal{T}_G), we can sample a point τ^{t_i} in \mathcal{T}_G^D where $t_s \leq t_i \leq t_e$ and construct training samples for the interpolator. The sampling strategies can be sampling at certain time intervals, sampling at specific locations or random sampling and we investigate the influence of different sampling rates in Section 4.5.

Interpolation-Discriminator The key difference between *ImIn-GAIL* and vanilla GAIL is in the discriminator. While learning to differentiate the expert trajectories from generated trajectories, the discriminator in *ImIn-GAIL* also

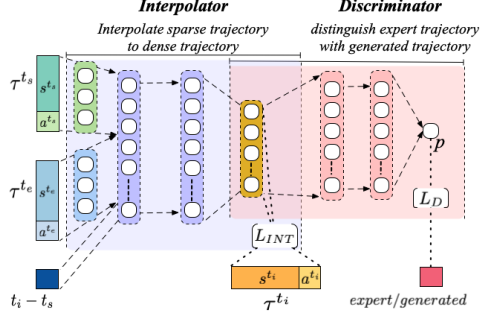


Fig. 3: Proposed interpolation-discriminator network.

learns to interpolate a sparse trajectory to a dense trajectory. Specifically, as is shown in Figure 3, the proposed interpolation-discriminator copes with two subtasks in an end-to-end way: *interpolation* on sparse data and *discrimination* on dense data.

Interpolator module The goal of the interpolator is to interpolate the sparse expert trajectories \mathcal{T}_E to the dense trajectories \mathcal{T}_E^D . We can use the generated dense trajectories \mathcal{T}_G^D and sparse trajectories \mathcal{T}_G from previous downsampling process as training data for the interpolator.

For each point τ^{t_i} to be interpolated, we first concatenate state and action and embed them into an m -dimensional latent space:

$$h_s = \sigma(\text{Concat}(s^{t_s}, a^{t_s})W_s + b_s), h_e = \sigma(\text{Concat}(s^{t_e}, a^{t_e})W_e + b_e) \quad (3)$$

where K is the feature dimension after the concatenation of s^{t_e} and a^{t_e} , $W_s \in \mathbb{R}^{K \times M}$, $W_e \in \mathbb{R}^{K \times M}$, $b_s \in \mathbb{R}^M$ and $b_e \in \mathbb{R}^M$ are weight matrix to learn, σ is ReLU function (same denotation for the following σ). Here, considering t_s and t_e may have different effects on interpolation, we use two different embedding weights for t_s and t_e .

After point embedding, we concatenate h_s and h_e with the time interval between t_s and t_i , and use a multi-layer perception (MLP) with L layers to learn the interpolation.

$$\begin{aligned} h_{in} &= \sigma(\text{Concat}(h_s, h_e, t_i - t_s)W_0 + b_0) \\ h_1 &= \sigma(h_{in}W_1 + b_1), h_2 = \sigma(h_1W_2 + b_2), \dots \\ h_L &= \tanh(h_{L-1}W_L + b_L) = \hat{\tau}^{t_i} \end{aligned} \quad (4)$$

where $W_0 \in \mathbb{R}^{(2M+1) \times N_0}$, $b_0 \in \mathbb{R}^{N_0}$ are the learnable weights; $W_j \in \mathbb{R}^{N_j \times N_{j+1}}$ and $b_j \in \mathbb{R}^{N_{j+1}}$ are the weight matrix for hidden layers ($1 \leq j \leq L-1$) of interpolator; $W_L \in \mathbb{R}^{N_L \times K}$ and $b_L \in \mathbb{R}^K$ are the weight matrix for the last layer of interpolator, which outputs an interpolated point $\hat{\tau}^{t_i}$. In the last layer of interpolator, we use \tanh as all the feature value of τ^{t_i} is normalized to $[-1, 1]$.

Discriminator module When sparse expert trajectories \mathcal{T}_E are interpolated into dense trajectories \mathcal{T}_E^D by the interpolator, the discriminator module learns to differentiate between expert dense trajectories \mathcal{T}_E^D and generated dense trajectories \mathcal{T}_G^D . Specifically, the discriminator learns to output a high score when encountering an interpolated point $\hat{\tau}^{t_i}$ originated from \mathcal{T}_E^D , and a low score when encountering a point from \mathcal{T}_G^D generated by π_θ . The output of the discriminator $\mathcal{D}_\psi(s, a)$ can then be used as a surrogate reward function whose value grows larger as actions sampled from π_θ look similar to those chosen by experts.

The discriminator module is an MLP with H hidden layers, takes h_L as input and outputs the probability of the point belongs to \mathcal{T}_E .

$$\begin{aligned} h_1^D &= \sigma(h_L W_1^D + b_1^D), h_2^D = \sigma(h_1^D W_2^D + b_2^D), \dots \\ p &= \text{Sigmoid}(h_{H-1}^D W_H^D + b_H^D) \end{aligned} \quad (5)$$

where $W_i^D \in \mathbb{R}^{N_{i-1}^D \times N_i^D}$, $b_i^D \in \mathbb{R}^{N_i^D}$ are learnable weights for i -th layer in discriminator module. For $i = 1$, we have $W_1^D \in \mathbb{R}^{K \times N_1^D}$, $b_1^D \in \mathbb{R}^{N_1^D}$, K is the concatenated dimension of state and action; for $i = H$, we have $W_H^D \in \mathbb{R}^{N_{H-1}^D \times 1}$, $b_H^D \in \mathbb{R}$.

Loss function of Interpolation-Discriminator The loss function of the Interpolation-Discriminator network is a combination of interpolation loss \mathcal{L}_{INT} and discrimination loss \mathcal{L}_D , which interpolates the unobserved points and predicts the probability of the point being generated by expert policy π^E simultaneously, :

$$\begin{aligned} \mathcal{L} &= \lambda \mathcal{L}_{INT} + (1 - \lambda) \mathcal{L}_D = \lambda \mathbb{E}_{\tau^t \sim \tau \in \mathcal{T}_G^D} (\hat{\tau}^t - \tau^t) + \\ & (1 - \lambda) [\mathbb{E}_{\tau^t \sim \tau \in \mathcal{T}_G} \log p(\tau^t) + \mathbb{E}_{\tau^t \sim \tau \in \mathcal{T}_E} \log(1 - p(\tau^t))] \end{aligned} \quad (6)$$

where λ is a hyper-parameter to balance the influence of interpolation and discrimination, $\hat{\tau}^t$ is the output of the interpolator module, and $p(\tau)$ is the output probability from the discriminator module.

3.3 Training and Implementation

Algorithm 1 describes the *ImIn-GAIL* approach. In this paper, the driving policy is parameterized with a two-layer fully connected network with 32 units for all the hidden layers. The policy network takes the driving state s as input and outputs the distribution parameters for a Beta distribution, and the action a will be sampled from this distribution. The optimization of the driving policy is optimized via TRPO [17]. Following [3, 8], we use the features in Table 1 to represent the driving state of a vehicle, and the driving policy takes the drivings state as input and outputs an action a (i.e., next step speed). For the interpolation-discriminator network, each driving point is embedded to a 10-dimensional latent space, the interpolator module uses a three-layer fully connected layer to interpolate the trajectory and the discriminator module contains a two-layer fully connected layer. Some of the important hyperparameters are listed in Table 2.

Algorithm 1: Training procedure of *ImIn-GAIL*

Input: Sparse expert trajectories \mathcal{T}_E , initial policy and interpolation-discriminator parameters θ_0, ψ_0
Output: Policy π_θ , interpolation-discriminator $InDNet_\psi$

```

1 for  $i \leftarrow 0, 1, \dots$  do
2   Rollout dense trajectories for all agents
    $\mathcal{T}_G^D = \{\tau | \tau = (\tau^{t_0}, \dots, \tau^{t_N}), \tau^{t_j} = (s^{t_j}, a^{t_j}) \sim \pi_{\theta_i}\};$ 
3   (Generator update step)
4   • Score  $\tau^{t_j}$  from  $\mathcal{T}_G^D$  with discriminator, generating reward using Eq. 2;
5   • Update  $\theta$  in generator given  $\mathcal{T}_G^D$  by optimizing Eq. 1;
6   (Interpolator-discriminator update step)
7   • Interpolate  $\mathcal{T}_E$  with the interpolation module in  $InDNet$ , generating
     dense expert trajectories  $\mathcal{T}_E^D$ ;
8   • Downsample generated dense trajectories  $\mathcal{T}_G^D$  to sparse trajectories  $\mathcal{T}_G$ ;
9   • Construct training samples for  $InDNet$ 
10  • Update  $InDNet$  parameters  $\psi$  by optimizing Eq. 6

```

Table 1: Features for a driving state

Feature Type		Detail Features
Road network		Lane ID, length of current lane, speed limit
Traffic signal		Current phase of traffic signal
Ego vehicle	Velocity, position in current lane, distance to the next traffic signal	
Leading vehicle	Relative distance, velocity and position in the current lane	
Indicators	Leading in current lane, exiting from intersection	

4 Experiment

4.1 Experimental Settings

We conduct experiments on CityFlow [29], an open-source traffic simulator that supports large-scale vehicle movements. In a traffic dataset, each vehicle is described as (o, t, d, r) , where o is the origin location, t is time, d is the destination location and r is its route from o to d . Locations o and d are both locations on the road network, and r is a sequence of road ID. After the traffic data is fed into the simulator, a vehicle moves towards its destination based on its route. The simulator provides the state to the vehicle control method and executes the vehicle acceleration/deceleration actions from the control method.

Dataset In experiment, we use both synthetic data and real-world data.

Synthetic Data In the experiment, we use two kinds of synthetic data, i.e., traffic movements under ring road network and intersection road network, as shown in Figure 4. Based on the traffic data, we use default simulation settings

Table 2: Hyper-parameter settings for *ImIn-GAIL*

Parameter	Value	Parameter	Value
Batch size for generator	64	Batch size for <i>InDNet</i>	32
Update epoches for generator	5	Update epoches for <i>InDNet</i>	10
Learning rate for generator	0.001	Learning rate for <i>InDNet</i>	0.0001
Number of layers in generator	4	Balancing factor λ	0.5

of the simulator to generate dense expert trajectories and sample sparse expert trajectories when vehicles pass through the red dots.

- *Ring*: The ring road network consists of a circular lane with a specified length, similar to [19, 26]. This is a very ideal and simplified scenario where the driving behavior can be measured.
- *Intersection*: A single intersection network with bi-directional traffic. The intersection has four directions (West→East, East→West, South→North, and North→South), and 3 lanes (300 meters in length and 3 meters in width) for each direction. Vehicles come uniformly with 300 vehicles/lane/hour in West↔East direction and 90 vehicles/lane/hour in South↔North direction.

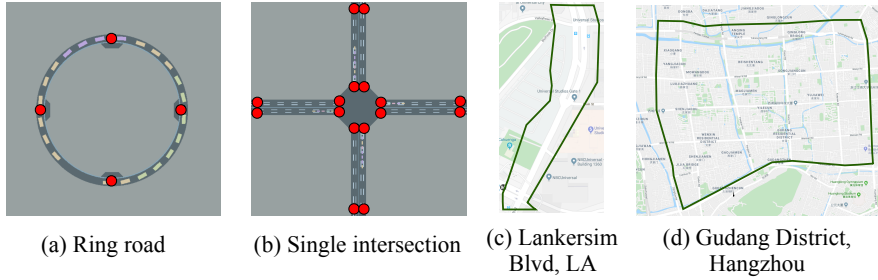


Fig. 4: Illustration of road networks. (a) and (b) are synthetic road networks, while (c) and (d) are real-world road networks.

Real-world Data We also use real-world traffic data from two cities: Hangzhou and Los Angeles. Their road networks are imported from OpenStreetMap³, as shown in Figure 4. The detailed descriptions of how we preprocess these datasets are as follows:

- $LA_{1 \times 4}$. This is a public traffic dataset collected from Lankershim Boulevard, Los Angeles on June 16, 2005. It covers an 1×4 arterial with four successive intersections. This dataset records the position and speed of every vehicle at every 0.1 second. We treat these records as dense expert trajectories and sample vehicles’ states and actions when they pass through intersections as sparse

³ <https://www.openstreetmap.org>

Table 3: Statistics of dense and sparse expert trajectory in different datasets

	<i>Ring Intersection LA_{1×4} HZ_{4×4}</i>			
Duration (seconds)	300	300	300	300
# of vehicles	22	109	321	425
# of points (dense)	1996	10960	23009	87739
# of points (sparse)	40	283	1014	1481

expert trajectories.

- *HZ_{4×4}*. This dataset covers a 4×4 network of Gudang area in Hangzhou, collected from surveillance cameras near intersections in 2016. This region has relatively dense surveillance cameras and we sampled the sparse expert trajectories in a similar way as in *LA_{1×4}*.

Data Preprocessing To mimic the real-world situation where the roadside surveillance cameras capture the driving behavior of vehicles at certain locations, the original dense expert trajectories are processed to sparse trajectories by sampling the driving points near several fixed locations unless specified. We use the sparse trajectories as expert demonstrations for training models. To test the imitation effectiveness, we use the same sampling method as the expert data and then compare the sparse generated data with sparse expert data. To test the interpolation effectiveness, we directly compare the dense generated data with dense expert data.

4.2 Compared Methods

We compare our model with the following two categories of methods: calibration-based methods and imitation learning-based methods.

Calibration-based methods For calibration-based methods, we use Krauss model [7], the default car-following model (CFM) of simulator SUMO [6] and CityFlow [29]. Krauss model has the following forms:

$$v_{safe}(t) = v_l(t) + \frac{g(t) - v_l(t)t_r}{\frac{v_l(t) + v_f(t)}{2b} + t_r} \quad (7)$$

$$v_{des}(t) = \min[v_{safe}(t), v(t) + a\Delta t, v_{max}] \quad (8)$$

where $v_{safe}(t)$ the safe speed at time t , $v_l(t)$ and $v_f(t)$ is the speed of the leading vehicle and following vehicle respectively at time t , $g(t)$ is the gap to the leading vehicle, b is the maximum deceleration of the vehicle and t_r is the driver's reaction time. $v_{des}(t)$ is the desired speed, which is given by the minimum of safe speed, maximum allowed speed, and the speed after accelerating at a for Δt . Here, a is the maximum acceleration and Δt is the simulation time step.

We calibrate three parameters in Krauss model, which are the maximum deceleration of the vehicle, the maximum acceleration of the vehicle, and the maximum allowed speed.

- **Random Search (CFM-RS)** [2]: The parameters are chosen when they generate the most similar trajectories to expert demonstrations after a finite number of trial of random selecting parameters for Krauss model.
- **Tabu Search (CFM-TS)** [16]: Tabu search chooses the neighbors of the current set of parameters for each trial. If the new CFM generates better trajectories, this set of parameters is kept in the Tabu list.

Imitation learning-based methods We also compare with several imitation learning-based methods, including both traditional and state-of-the-art methods.

- **Behavioral Cloning (BC)** [21] is a traditional imitation learning method. It directly learns the state-action mapping in a supervised manner.
- **Generative Adversarial Imitation Learning (GAIL)** is a GAN-like framework [4], with a generator controlling the policy of the agent, and a discriminator containing a classifier for the agent indicating how far the generated state sequences are from that of the demonstrations.

4.3 Evaluation Metrics

Following existing studies [8, 3, 30], to measure the error between learned policy against expert policy, we measure the position and the travel time of vehicles between generated dense trajectories and expert dense trajectories, which are defined as:

$$RMSE_{pos} = \frac{1}{T} \sum_{t=1}^T \sqrt{\frac{1}{M} \sum_{i=1}^m (l_i^t - \hat{l}_i^t)^2}, RMSE_{time} = \sqrt{\frac{1}{M} \sum_{i=1}^m (d_i - \hat{d}_i)^2} \quad (9)$$

where T is the total simulation time, M is the total number of vehicles, l_i^t and \hat{l}_i^t are the position of i -th vehicle at time t in the expert trajectories and in the generated trajectories relatively, d_i and \hat{d}_i are the travel time of vehicle i in expert trajectories and generated trajectories respectively.

4.4 Performance Comparison

In this section, we compare the dense trajectories generated by different methods with the expert dense trajectories, to see how similar they are to the expert policy. The closer the generated trajectories are to the expert trajectories, the more similar the learned policy is to the expert policy. From Table 4, we can see that *ImIn-GAIL* achieves consistently outperforms over all other baselines across synthetic and real-world data. *CFM-RS* and *CFM-TS* can hardly achieve satisfactory results because the model predefined by CFM could be different from the real world. Specifically, *ImIn-GAIL* outperforms vanilla *GAIL*, since *ImIn-GAIL* interpolates the sparse trajectories and thus has more expert trajectory data, which will help the discriminator make more precise estimations to correct the learning of policy.

Table 4: Performance w.r.t Relative Mean Squared Error (RMSE) of time (in seconds) and position (in kilometers). All the measurements are conducted on dense trajectories. Lower the better. Our proposed method *ImIn-GAIL* achieves the best performance.

	<i>Ring</i>		<i>Intersection</i>		<i>LA_{1×4}</i>		<i>HZ_{4×4}</i>	
	time (s)	pos (km)	time (s)	pos (km)	time (s)	pos (km)	time (s)	pos (km)
<i>CFM-RS</i>	343.506	0.028	39.750	0.144	34.617	0.593	27.142	0.318
<i>CFM-TS</i>	376.593	0.025	95.330	0.184	33.298	0.510	175.326	0.359
<i>BC</i>	201.273	0.020	58.580	0.342	55.251	0.698	148.629	0.297
<i>GAIL</i>	42.061	0.023	14.405	0.032	30.475	0.445	14.973	0.196
<i>ImIn-GAIL</i>	16.970	0.018	4.550	0.024	19.671	0.405	5.254	0.130

4.5 Study of *ImIn-GAIL*

Interpolation Study To better understand how interpolation helps in simulation, we compare two representative baselines with their two-step variants. Firstly, we use a pre-trained non-linear interpolation model to interpolate the sparse expert trajectories following the idea of [28, 20]. Then we train the baselines on the interpolated trajectories.

Table 5 shows the performance of baseline methods in *Ring* and *Intersection*. We find out that baseline methods in a two-step way show inferior performance. One possible reason is that the interpolated trajectories generated by the pre-trained model could be far from the real expert trajectories when interacting in the simulator. Consequently, the learned policy trained on such interpolated trajectories makes further errors.

In contrast, *ImIn-GAIL* learns to interpolate and imitate the sparse expert trajectories in one step, combining the interpolator loss and discriminator loss, which can propagate across the whole framework. If the trajectories generated by π_θ is far from expert observations in current iteration, both the discriminator and the interpolator will learn to correct themselves and provide more precise reward for learning π_θ in the next iteration. Similar results can also be found in *LA_{1×4}* and *HZ_{4×4}*, and we omit these results due to page limits.

Sparsity Study In this section, we investigate how different sampling strategies influence *ImIn-GAIL*. We sample randomly from the dense expert trajectories at different time intervals to get different sampling rates: 2%, 20%, 40%, 60%, 80%, and 100%. We set the sampled data as the expert trajectories and evaluate by measuring the performance of our model in imitating the expert policy. As is shown in Figure 5, with denser expert trajectory, the error of *ImIn-GAIL* decreases, indicating a better policy imitated by our method.

Table 5: RMSE on time and position of our proposed method *ImIn-GAIL* against baseline methods and their corresponding two-step variants. Baseline methods and *ImIn-GAIL* learn from sparse trajectories, while the two-step variants interpolate sparse trajectories first and trained on the interpolated data. *ImIn-GAIL* achieves the best performance in most cases.

	<i>Ring</i>		<i>Intersection</i>	
	time (s)	position (km)	time (s)	position (km)
<i>CFM-RS</i>	343.506	0.028	39.750	0.144
<i>CFM-RS</i> (two step)	343.523	0.074	73.791	0.223
<i>GAIL</i>	42.061	0.023	14.405	0.032
<i>GAIL</i> (two step)	98.184	0.025	173.538	0.499
<i>ImIn-GAIL</i>	16.970	0.018	4.550	0.024

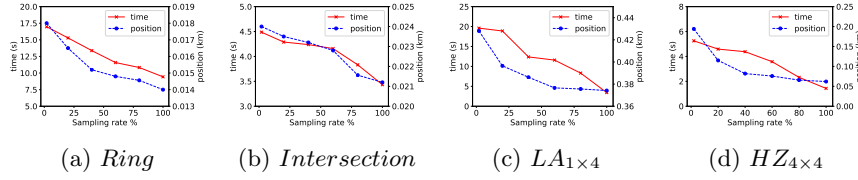


Fig. 5: RMSE on time and position of our proposed method *ImIn-GAIL* under different level of sparsity. As the expert trajectory become denser, a more similar policy to the expert policy is learned.

4.6 Case Study

To study the capability of our proposed method in recovering the dense trajectories of vehicles, we showcase the movement of a vehicle in *Ring* data learned by different methods.

We visualize the trajectories generated by the policies learned with different methods in Figure 6. We find that imitation learning methods (*BC*, *GAIL*, and *ImIn-GAIL*) perform better than calibration-based methods (*CFM-RS* and *CFM-TS*). This is because the calibration based methods pre-assumes an existing model, which could be far from the real behavior model. On the contrast, imitation learning methods directly learn the policy without making unrealistic formulations of the CFM model. Specifically, *ImIn-GAIL* can imitate the position of the expert trajectory more accurately than all other baseline methods. The reason behind the improvement of *ImIn-GAIL* against other methods is that in *ImIn-GAIL*, policy learning and interpolation can enhance each other and result in significantly better results.

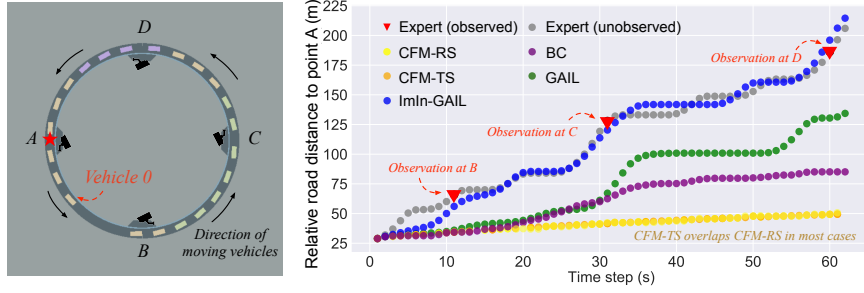


Fig. 6: The generated trajectory of a vehicle in the *Ring* scenario. Left: the initial position of the vehicles. Vehicles can only be observed when they pass four locations *A*, *B*, *C* and *D* where cameras are installed. Right: the visualization for the trajectory of *Vehicle 0*. The x-axis is the timesteps in seconds. The y-axis is the relative road distance in meters. Although vehicle 0 is only observed three times (red triangles), *ImIn-GAIL* (blue points) can imitate the position of the expert trajectory (grey points) more accurately than all other baselines. Better viewed in color.

5 Related Work

Parameter calibration In parameter calibration-based methods, the driving behavior model is a prerequisite, and parameters in the model are tuned to minimize a pre-defined cost function. Heuristic search algorithms such as random search, Tabu search[16], and genetic algorithm [5] can be used to search the parameters. These methods rely on the pre-defined models (mostly equations) and usually fail to match the dynamic vehicle driving pattern in the real-world.

Imitation learning Without assuming an underlying physical model, we can solve this problem via imitation learning. There are two main lines of work: (1) behavior cloning (BC) and Inverse reinforcement learning (IRL). BC learns the mapping from demonstrated observations to actions in a supervised learning way [13, 21], but suffers from the errors which are generated from unobserved states during the simulation. On the contrast, IRL not only imitates observed states but also learns the expert’s underlying reward function, which is more robust to the errors from unobserved states [1, 15, 33]. Recently, a more effective IRL approach, GAIL [4], incorporates generative adversarial networks with learning the reward function of the agent. However, all of the current work did not address the challenges of sparse trajectories, mainly because in their application contexts, e.g., game or robotic control, observations can be fully recorded every time step.

6 Conclusion

In this paper, we present a novel framework *ImIn-GAIL* to integrate interpolation with imitation learning and learn the driving behavior from sparse trajectory data. Specifically, different from existing literature which treats data interpolation as a separate and preprocessing step, our framework learns to interpolate and imitate expert policy in a fully end-to-end manner. Our experiment results show that our approach significantly outperforms state-of-the-art methods. The application of our proposed method can be used to build a more realistic traffic simulator using real-world data.

Acknowledgments

The work was supported in part by NSF awards #1652525 and #1618448. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

References

1. Abbeel, P., Ng, A.Y.: Apprenticeship learning via inverse reinforcement learning. In: ICML (2004)
2. Asamer, J., van Zuylen, H.J., Heilmann, B.: Calibrating car-following parameters for snowy road conditions in the microscopic traffic simulator vissim. IET Intelligent Transport Systems **7**(1) (2013)
3. Bhattacharyya, R.P., Phillips, D.J., Wulfe, B., Morton, J., Kuefler, A., Kochenderfer, M.J.: Multi-agent imitation learning for driving simulation. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE (2018)
4. Ho, J., Ermon, S.: Generative adversarial imitation learning. In: NeurIPS (2016)
5. Kesting, A., Treiber, M.: Calibrating car-following models by using trajectory data: Methodological study. Transportation Research Record **2088**(1), 148–156 (2008)
6. Krajzewicz, D., Erdmann, J., Behrisch, M., Bieker, L.: Recent development and applications of SUMO - Simulation of Urban MObility. International Journal On Advances in Systems and Measurements **5**(3&4), 128–138 (December 2012)
7. Krauss, S.: Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics. Ph.D. thesis (1998)
8. Kuefler, A., Morton, J., Wheeler, T., Kochenderfer, M.: Imitating driver behavior with generative adversarial networks. In: IEEE Intelligent Vehicles Symposium (IV). IEEE (2017)
9. Leutzbach, W., Wiedemann, R.: Development and applications of traffic simulation models at the karlsruhe institut fur verkehrwesen. Traffic engineering & control **27**(5) (1986)
10. Li, S.C.X., Marlin, B.M.: A scalable end-to-end gaussian process adapter for irregularly sampled time series classification. In: NeurIPS (2016)
11. Liu, Y., Zhao, K., Cong, G., Bao, Z.: Online anomalous trajectory detection with deep generative sequence modeling. In: ICDE (2020)
12. Lou, Y., Zhang, C., Zheng, Y., Xie, X., Wang, W., Huang, Y.: Map-matching for low-sampling-rate gps trajectories. In: SIGSPATIAL. ACM (2009)

13. Michie, D., Bain, M., Hayes-Miches, J.: Cognitive models from subcognitive skills. *IEEE Control Engineering Series* **44** (1990)
14. Nagel, K., Schreckenberg, M.: A cellular automaton model for freeway traffic. *Journal de physique I* **2**(12) (1992)
15. Ng, A.Y., Russell, S.J., et al.: Algorithms for inverse reinforcement learning. In: *ICML* (2000)
16. Osorio, C., Punzo, V.: Efficient calibration of microscopic car-following models for large-scale stochastic network simulators. *Transportation Research Part B: Methodological* **119**, 156–173 (2019)
17. Schulman, J., Levine, S., Moritz, P., Jordan, M.I., Abbeel, P.: Trust region policy optimization. In: *ICML* (2015)
18. Song, J., Ren, H., Sadigh, D., Ermon, S.: Multi-agent generative adversarial imitation learning. In: *NeurIPS* (2018)
19. Sugiyama, Y., Fukui, M., Kikuchi, M., Hasebe, K., Nakayama, A., Nishinari, K., Tadaki, S.i., Yukawa, S.: Traffic jams without bottlenecks—experimental evidence for the physical mechanism of the formation of a jam. *New journal of physics* **10**(3) (2008)
20. Tang, X., Gong, B., Yu, Y., Yao, H., Li, Y., Xie, H., Wang, X.: Joint modeling of dense and incomplete trajectories for citywide traffic volume inference. In: *The World Wide Web Conference*. ACM (2019)
21. Torabi, F., Warnell, G., Stone, P.: Behavioral cloning from observation. In: *IJCAI* (2018)
22. Wei, H., Chen, C., Zheng, G., Wu, K., Gayah, V., Xu, K., Li, Z.: Presslight: Learning max pressure control to coordinate traffic signals in arterial network. In: *KDD* (2019)
23. Wei, H., Xu, N., Zhang, H., Zheng, G., Zang, X., Chen, C., Zhang, W., Zhu, Y., Xu, K., Li, Z.: Colight: Learning network-level cooperation for traffic signal control. In: *CIKM* (2019)
24. Wei, H., Zheng, G., Gayah, V., Li, Z.: A survey on traffic signal control methods. *arXiv preprint arXiv:1904.08117* (2019)
25. Wei, H., Zheng, G., Yao, H., Li, Z.: Intellilight: A reinforcement learning approach for intelligent traffic light control. In: *KDD* (2018)
26. Wu, C., Kreidieh, A., Vinitsky, E., Bayen, A.M.: Emergent behaviors in mixed-autonomy traffic. In: *Conference on Robot Learning* (2017)
27. Wu, Y., Tan, H., Ran, B.: Differential variable speed limits control for freeway recurrent bottlenecks via deep reinforcement learning. *arXiv preprint arXiv:1810.10952* (2018)
28. Yi, X., Zheng, Y., Zhang, J., Li, T.: St-mvl: filling missing values in geo-sensory time series data. In: *IJCAI*. AAAI Press (2016)
29. Zhang, H., Feng, S., Liu, C., Ding, Y., Zhu, Y., Zhou, Z., Zhang, W., Yu, Y., Jin, H., Li, Z.: Cityflow: A multi-agent reinforcement learning environment for large scale city traffic scenario. In: *International World Wide Web Conference* (2019)
30. Zheng, G., Liu, H., Xu, K., Li, Z.: Learning to simulate vehicle trajectories from demonstrations. *ICDE* (2020)
31. Zheng, K., Zheng, Y., Xie, X., Zhou, X.: Reducing uncertainty of low-sampling-rate trajectories. In: *ICDE* (2012)
32. Zheng, Y.: Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)* **6**(3) (2015)
33. Ziebart, B.D., Maas, A.L., Bagnell, J.A., Dey, A.K.: Maximum entropy inverse reinforcement learning. In: *AAAI*. vol. 8 (2008)