2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

Imitation Learning of Hierarchical Driving Model: From Continuous Intention to Continuous Trajectory

Yunkai Wang[1], Dongkun Zhang[1], Jingke Wang[1], Zexi Chen[1], Yuehua Li[2], Yue Wang[1†], Rong Xiong[1]

***Abstract*— One of the challenges to reduce the gap between the machine and the human level driving is how to endow the system with the learning capacity to deal with the coupled complexity of environments, intentions, and dynamics. In this paper, we propose a hierarchical driving model with explicit models of continuous intention and continuous dynamics, which decouples the complexity in the observation-to-action reasoning in the human driving data. Specifically, the continuous intention module takes perception to generate a potential map encoded with obstacles and intentions. Then, the potential map is regarded as a condition, together with the current dynamics, to generate a continuous trajectory as output by a continuous function approximator network, whose derivatives can be used for supervision without additional parameters. Finally, our method is validated by both datasets and stimulation, demonstrating that our method has higher prediction accuracy of displacement and velocity and generates smoother trajectories. Our method is also deployed on the real vehicle with loop latency, validating its effectiveness. To the best of our knowledge, this is the first work to produce the driving trajectory using a continuous function approximator network. Our code is available at `https://github.com/ZJU-Robotics-Lab/CICT`.**

I. Introduction

Autonomous driving is an appealing research topic in recent years because of its ability to reduce labor costs and traffic accidents. In typical autonomous driving systems, vehicles need to perform observation-to-action reasoning to generate safe and efficient driving in various scenarios. One of the challenges to reduce the gap between the machine and the human level driving is how to endow the system with the learning capacity to deal with the coupled complexity of environments, intentions, and dynamics [1].

Toward this goal, there are several learning paradigms. The most common paradigm is to decompose the observation-to-action reasoning into separated modules (e.g., detection, tracking, and planning), which has good generalization given the carefully human-designed system structure, but it calls for intensive human supervision for each module [2]. To relieve the difficulty, another paradigm is to model the observation-to-action reasoning as a black box, thus massive labeled data can be automatically generated by simply recording the human driving process [3]. However, this paradigm has much lower generalization capability, partly due to the almost missing of the human system design, losing the human knowledge intrinsically reserved in the system structure [4].

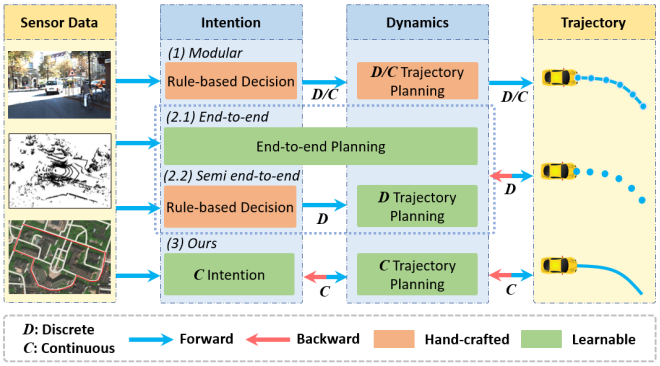

Fig. 1: Different trajectory planning approaches for autonomous vehicles. (1) Classical modular design; (2.1) End-to-end planning method; (2.2) Simi end-to-end planning method; (3) Our hierarchical model.

In this paper, we follow the second paradigm to only use the hand-free annotation data to learn a vehicle driving model. At the same time, motivated by the idea in the first paradigm, we also inject human knowledge via the system structure design, which is to explicitly model a hierarchical structure to decouple the intentions and dynamics from the human driving data, as shown in Fig. 1.

Specifically, to model the trajectory, we regard the network as a continuous function approximator, whose high-order derivatives can be obtained analytically without adding model parameters, and they can also be used for supervision to achieve better results. A trajectory can be supervised at any sampling interval without adding model parameters, making it sample efficient. In contrast, the previous works mainly use only the discrete positions as supervision, which loses the continuous motion characteristics of the demonstration data [5]. Some previous works also use additional outputs to capture the velocity, but they fail to guarantee the relation of derivatives between displacement and velocity [5] [6].

For the high-level intention, we follow our previous work [7] to model the driving intention generation process as a mapping from coarse route planning to a potential map without time parametrization. Therefore, the intention generation can be regarded as a continuous tactical decision maker learned from human data. Since many previous works model the intention as a discrete command [8] (e.g., turn left and go straight), their motion model has to learn a process actually with larger variance than ours.

As one can see, beyond imitating the human driving data, we imitate the design of the conventional hierarchical plan-

[1]Yunkai Wang, Dongkun Zhang, Jingke Wang, Zexi Chen, Yue Wang, Rong Xiong are with the State Key Laboratory of Industrial Control Technology and Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou, China.
[2]Yuehua Li is with the Intelligent Space Robotics Joint Research Laboratory, Zhejiang Lab, Hangzhou, China.
† Corresponding author, `wangyue@iipc.zju.edu.cn`

ning system structure that consists of a local path planner and a trajectory planner, using the driving intention as an explicit intermediate representation to decompose the intention and dynamics. From a network training perspective, the potential map is explicitly supervised as a side task [9], which acts as a semantic regularizer and brings better interpretability.

We validate our method on both datasets and simulator, demonstrating that our method has higher prediction accuracy and generates smoother trajectories. To summarize, the main contributions of this paper include the following:

- We propose a novel representation of trajectory, which brings better driving performance.
- We propose a hierarchical driving model for end-to-end learning from human driving data with explicit intermediate representation, which decomposes the intention and dynamics.
- Open-loop and closed-loop validation on both datasets and simulator, demonstrating that our method has higher prediction accuracy and generates smoother trajectories. Besides, practical implementation is proposed to tolerate the network latency, which enables generalization testing in real world.

II. Related Work

In 2016, benefiting from powerful GPUs and deep neural networks, Bojarski *et al.* [3] developed *DAVE-2* and tested their method on a real-world platform to prove that deep learning methods can achieve autonomous driving. Although this method can just achieve lane following in simple scenarios, it was a bold start.

Recently, a lot of works followed this approach to achieve more intelligent autonomous driving algorithms, including imitation learning methods [8] [10] and reinforcement learning methods [11] [12].

To make the driving models more generic and long-sighted, another paradigm called end-to-end planning is been widely studied. The work [13] generates paths to guide the vehicle driving, increasing understanding of the environment in driving tasks. The work [14] generates diverse paths by learning the multi-mode distribution of paths. These methods focus more on the perception of the environment, while the paths generated by these methods do not contain dynamic information.

The work [8] [15] [5] [16] generate trajectories with dynamic information, which can be further used for closed-loop tasks. However, most current trajectory generation methods have the following disadvantages: 1) When the controller is controlling, it cannot get the trajectory point at any moment, but only take the approximate point, which will reduce the control accuracy. 2) These methods output a fixed number of trajectory points with fixed time intervals. To output more points or other physical quantities, these methods require additional model parameters, which makes these methods data inefficient.

Our method corresponds with the end-to-end trajectory planning paradigm, but has some differences from the current methods: 1) Compared with the end-to-end learning methods [17] [18] using RGB images directly, our method using an explicit continuous intention as an intermediate representation [4] to decompose the intention (path) and dynamics (trajectory), enhancing model interpretability. 2) Compared with other trajectory generation methods [8] [15] [5] that generate discrete trajectories, we propose a new representation of trajectories, whose contributions are: 1) High-order derivatives can be obtained analytically without adding model parameters, and they can also be used for supervision to achieve better results. 2) A trajectory can be supervised at any sampling interval without adding model parameters, making it sample efficient.

III. Methodology

In our previous work [7], the generator network finally obtains the potential map, which is the driving intention that reflects the path information, but without any system dynamic information or temporal information. That means the vehicle only knows where to go, but does not know how to get there. Hence, that approach loses current dynamics information in the training data and does not allow end-to-end learning of human driving trajectories. In this paper, we propose the trajectory generation model, combining with the driving intention module proposed in our previous work [7], so that the whole hierarchical model can be trained end-to-end to learn from human driving trajectories.

The overall system architecture of our proposed method is shown in Fig. 2, which is divided into a *driving intention module* and a *trajectory generation module*. The driving intention module maps the image V and routing planning R into continuous intention I, modeling it as goal guided potential, and further injects it with obstacle potentials, generating a potential map C. Then the trajectory generation module maps a stack of past potential maps $C_{t_0-K:t_0}$, and the current velocity v_{t_0} into a continuous trajectory $\mathcal{T}$.

A. Driving Intention Module

The driving intention module is an extension of our previous work [7], which uses an undifferentiable motion planner to generate control, thus blocking the end-to-end learning. We first briefly introduce the generation of continuous intention and the potential map.

Intention Modeling: The module takes as input front-view RGB image V combined with routing planning R. The routing planning R is generated by usual navigation software, like GoogleMap, and represented as a local crop of the map according to the current position of the vehicle from GPS and IMU. Thus R is a robot-centric representation, which changes its coordinates when the robot moves. For more details please refer to [19]. We model the continuous driving intention I as a mask of V to indicate the local path to the goal in the current image coordinates, which is generated by a U-Net based conditional generator $\mathcal{G}$ with a concatenation of V and R being the input:

$$I = \mathcal{G}(V, R) \tag{1}$$

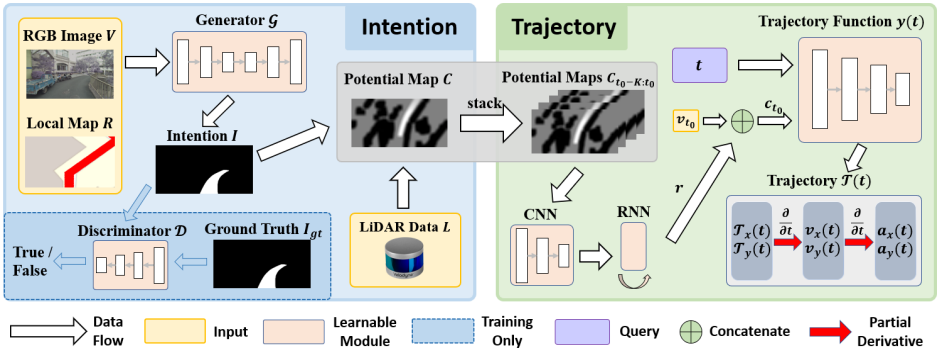

Fig. 2: Overview of our hierarchical driving model. The hierarchical driving model is divided into a *driving intention module* (upper part) and a *trajectory generation module* (lower part). The RGB image V is acquired from the vehicle, and local routing planning map R is cropped from an offline map according to the current low-cost GPS and IMU data. The intention I is a binary map generated by Generator $\mathcal{G}$ to indicate the local path (white area) to the goal in the current image coordinates, and it is supervised by its ground truth I_{gt} and Discriminator $\mathcal{D}$ which is trained to discriminate whether the input intention is generated by the Generator or is the ground truth. Then the generated intention I is then projected into the top view and fused with LiDAR data L to synthesize the top view grayscale potential map C, which is encoded with obstacles and intentions being expressed as a grid-based map. The trajectory generation module leverages multi-frame potential maps to extract spatial-temporal feature r, and concatenate it with current speed v_{t_0} at time t_0 to obtain a condition c_{t_0}. Then the trajectory function network $y(t)$ can leverage the condition to generate longitudinal displacement $\mathcal{T}_x$ and lateral displacement $\mathcal{T}_y$ according to the query time t. High-order quantities such as velocity and acceleration can be analytically obtained through partial derivative operation over time t, and they form the trajectory $\mathcal{T}$ together with the displacement.

To build the ground truth I_{gt} of I, we refer to the future path executed by the robot in the real world and transform it into the current image coordinates. Therefore, we have pixel-level supervision for I formulated as a loss term:

$$\mathcal{L}_{int}(\mathcal{G}) = \mathbb{E}_{V,R,I_{gt}}[||I_{gt} - \mathcal{G}(V,R)||_1] \tag{2}$$

Adversarial Training: As the robot only executes in one direction in an intersection in the real world, we only have one-way ground truth for the image taken at intersections. For other directions in the intersection, we cannot build (2). To improve the diversity, we use both paired routing planning R and unpaired routing planning R' which is randomly sampled with RGB image V, and utilize adversarial training for supervision. Specifically, we build a discriminator $\mathcal{D}$ to discriminate whether the driving intention map is generated from the distribution of ground truth data I_{gt}, so that exact one-to-one supervision is not required, leading to a conditional adversarial loss term:

$$\begin{aligned}\mathcal{L}_{intadv}(\mathcal{G},\mathcal{D}) =& \mathbb{E}_{V,R,I_{gt}}[\log \mathcal{D}(V,R,I_{gt})]+ \\ & \frac{1}{2}(\mathbb{E}_{R,V}[\log(1-\mathcal{D}(V,R,\mathcal{G}(V,R)))]+ \\ & \mathbb{E}_{R',V}[\log(1-\mathcal{D}(V,R',\mathcal{G}(V,R')))])\end{aligned} \tag{3}$$

Potential Map Building: Given the driving intention conditional by the global routing, we have goal guidance in robot-centric coordinate. As this mask must lie on the local ground plane, we can transform I into a bird-eye view representation. By smoothing the binary mask, we have a goal guided potential map C_{goal}. Furthermore, we represent the perception result as an obstacle potential map $C_{obstacle}$ in the same bird-eye view coordinates. In this paper, we use LiDAR for obstacle detection, but any other obstacle perception is usable. Combining the two potential maps, we have a local potential map C for motion planning, which can be regarded as a local artificial potential field. For more details about the potential map, please refer to [19].

B. Trajectory Generation Module

The trajectory generation module acts as a motion planner in a conventional pipeline, but it is differentiable, leveraging the end-to-end imitation learning from the human demonstration driving data. The main difficulty for this module is to keep the continuous and smooth nature of the trajectory. Thus the down-stream controller can track the trajectory in a high frequency asynchronously.

Continuous Function Modeling: For a network with linear decoder, we have a linear mapping from the last hidden layer $f(z)$ to the output y as

$$y = \sum_i w_i f(z_i) + b \tag{4}$$

where w_i and b are the network parameters, f is the nonlinear activation function. Furthermore, regarding the input to the network as condition c and a function variable x, we have an interpretation of (4) as

$$y(x) = \sum_i w_i f(z_{i,c}(x)) + b \tag{5}$$

In this form, the nonlinear activation functions of the last hidden layer form a set of basis functions f parameterized by $z_{i,c}$, deriving a linear combination of basis functions to represent $y(x)$. Compared with the conventional basis functions, the basis functions of (5) have adaptive forms

inferred by the input condition c and x. Consequently, this form provides a continuous function approximator, of which the high-order derivatives are naturally derived during the network back-propagation process:

$$\begin{aligned}\frac{\partial y}{\partial x} &= \sum_i w_i \frac{\partial f}{\partial z_{i,c}} \frac{\partial z_{i,c}}{\partial x} \\ \frac{\partial^2 y}{\partial x^2} &= \sum_i w_i \left(\frac{\partial f}{\partial z_{i,c}} \frac{\partial^2 z_{i,c}}{\partial x^2} + \frac{\partial^2 f}{\partial z_{i,c} \partial x} \frac{\partial z_{i,c}}{\partial x}^T \right)\end{aligned} \tag{6}$$

The 2^{nd} order derivative of $y(x)$ is the Hessian matrix. When the dimension of x is high, the computation is intractable. However, if the continuous function is parameterized by the low dimensional variable x, evaluating the high-order derivatives is feasible.

Trajectory Representation: At the perspective of continuous function approximator (5), we therefore use it for trajectory learning. Note that the variable of trajectory $\mathcal{T}$ is the time t, we have the high-order derivatives for trajectory, i.e., the velocity $v(t)$ and acceleration $a(t)$:

$$v(t) = \frac{\partial \mathcal{T}(t)}{\partial t}, a(t) = \frac{\partial^2 \mathcal{T}(t)}{\partial t^2} \tag{7}$$

By simply regarding x as t, we represent a continuous trajectory $y(t)$ as a neural network with high-order derivatives and is differentiable for learning. Thanks to the 1-dimensional variable t, the evaluation of (6) is very efficient. Moreover, inspired by the conventional Fourier analysis, which is very effective for modeling dynamic system input and output, we also use sinusoidal function [20] as the form of nonlinear activation function z, instead of the usual ReLU functions. Finally, we have the neural trajectory as

$$y(t) = \sum_i w_i \cos(z_{i,c}(t)) + b \tag{8}$$

This function can be supervised by any order of derivatives to learn the parameters [20]. In this paper, at time t_0, we build loss terms based on the observed position $\hat{\mathcal{T}}(k)$, velocity $\hat{v}(k)$ and the acceleration $\hat{a}(k)$ at several sampling time k:

$$\begin{aligned}\mathcal{L}_{\mathcal{T}}(y) =& \mathbb{E}_{t_0}[\sum_{k \in \{t_0, t_0+T\}} ||\hat{\mathcal{T}}(k) - y(k)||_2^2 \\ &+ \lambda_1 ||\hat{v}(k) - \frac{\partial y(t)}{\partial t}|_{t=k}||_2^2 \\ &+ \lambda_2 ||\hat{a}(k) - \frac{\partial^2 y(t)}{\partial t^2}|_{t=k}||_2^2]\end{aligned} \tag{9}$$

where T is the time horizon for the trajectory.

We briefly summarize the advantage of modeling continuous trajectory as the *smoothness* for execution, which is guaranteed by the infinitely order differentiable sinusoidal basis functions, and the *minimal parameterization* for the compatible displacement, velocity, and acceleration generation, which is guaranteed by the continuous function approximator when regarding the last layer of the network as a linear combination to the output.

Conditional Generation: To embed the proposed differentiable representation into the trajectory generation module, we need to specify the condition c. In driving scenarios, the factors we consider for trajectory planning include the ego-vehicle dynamics, the surrounding obstacles dynamics, as well as the stationery environmental obstacles. For the first factor, we include the current velocity v_{t_0}, as one of the condition. For the second and third factors, since the potential map C only reflects a cut of the surrounding obstacles, we encode a window of multiple previous potential maps $C_{t_0-K:t_0}$ via a CNN and an RNN, denoted as r. Resultantly, we have a condition at t_0 as

$$c_{t_0} = \left[\begin{array}{cc} v_{t_0} & r(C_{t_0-K:t_0}) \end{array} \right]^T \tag{10}$$

which becomes a mapping from t to the frequency and phase of the sinusoidal basis function. Then we can derive the trajectory from t_0 to t_0+T by sweeping t from t_0 to t_0+T.

C. Loss Function Design and Training

Based on the loss terms design introduced above, we can simply set the loss function for training the network as a sum of (2), (3) and (9):

$$\mathcal{L} = \mathcal{L}_{intadv}(\mathcal{G}, \mathcal{D}) + \mathcal{L}_{int}(\mathcal{G}) + \mathcal{L}_{\mathcal{T}}(y) \tag{11}$$

Data Relabeling: When predicting the future trajectory, there can be non-causal labeling for the current data. For example, a vehicle stops in a queue, waits for the traffic light to turn green. Assume there are $3s$ left for the light turning. Then the retrospective trajectory label with a time horizon of $5s$ for the current time is: stop for $0-3s$ and accelerate for $3s-5s$. Therefore, the supervision becomes a trajectory crossing the preceding vehicle. However, the driver at the current time actually cannot know that the light turns in $3s$, hence such supervision is reasonable unless the driver is informed with external aid. To relieve this labeling problem, we re-label the supervision for such crossing cases with causality by modifying the trajectory with deceleration at a constant acceleration before the collision happens. We consider it is reasonable behavior for human driving without referring to the future.

D. Closed-loop System Implementation

Different from the works using only a point position for tracking, which may not perform well due to the limited onboard computing frequency, we implement an asynchronous driving architecture by separating the long-horizon trajectory planning and trajectory tracking in two threads, achieving a receding horizon controller [21], to overcome the time-delay issue, which is shown in Fig. 3.

Controller design: Specifically, the planning thread generates continuous trajectories with time horizon $T = 3s$ in a fixed frequency or triggers and annotates the trajectory with the vehicle's pose and the starting time t_{start}. The control thread is running in another fixed higher frequency. It gets vehicle's current pose at time t_{now} and queries the trajectory with $t = t_{now} - t_{start}$ as the reference point. Then we use a feedback controller for throttle and brake control, and a nonlinear rear-wheel feedback controller [22] for steering control. For more details about the controller, please refer to [19].

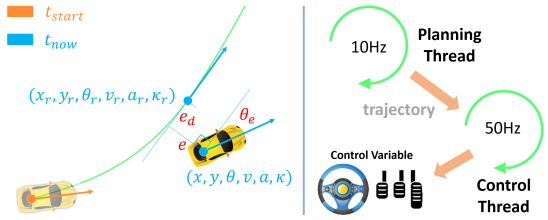

Fig. 3: Asynchronous planning and control. The planning thread generates a trajectory at t_{start}, and the control thread tracks the whole trajectory w.r.t. time. At time t_{now}, the lateral error e and heading error θ_e, together with curvature κ_r are fed into a steering controller. The throttle and brake commands are computed by a PID controller according to the difference between v_r and current v.

IV. Experiments

A. Open-loop Experiments on Datasets

We first validate the effectiveness of our proposed method on two publicly available and widely recognized datasets: KITTI Raw Data [23] and Oxford Radar RobotCar [24].

Datasets: *KITTI Raw Data (KITTI)* contains 6 hours of traffic scenarios at 10 Hz using a variety of sensor modalities including high-resolution stereo cameras, a Velodyne 3D laser scanner, and a high-precision GPS/IMU inertial navigation system (INS). *Oxford Radar RobotCar (RobotCar)* is a radar extension to The Oxford RobotCar Dataset, providing data from Dual Velodyne HDL-32E LIDARs and one Bumblebee XB3 trinocular stereo camera with optimized ground truth visual odometry for $280km$ of driving around Oxford University.

Implementation Details: Since there is no routing planning R, intention image I and potential map C in these datasets directly, we first build them up before training. For KITTI, we extract a time-independent $30m$ path from INS which is then projected into image coordinate by perspective mapping to generate intention map I. Potential map C is acquired by fusing point cloud and reprojected generated intention image I by driving intention module using inverse perspective mapping. Note that the point cloud data is first transformed into INS coordinate, ahead of fusion, with calibration information between Velodyne and INS. As for Oxford, the data preparation process is similar to KITTI, with the exception that we use visual odometry (VO) data as ground truth trajectory due to poor GPS signals provides. Additionally, we undistort image data collected by the stereo camera according to [24]. For both datasets, we split them as 70% are used for training with 10% for testing and 20% for evaluation.

Evaluation Metrics: We use six evaluation metrics to assess the effectiveness of various models, all of which are computed over the evaluation part of datasets.

- Average Displacement Error (ADE, $\mathcal{E}_{ad}$) [5]: Average L2 distance between the ground truth and generated trajectories over all positions. Note that the corresponding positions are found w.r.t time rather than the nearest neighbor.
- Final Displacement Error (FDE, $\mathcal{E}_{fd}$) [5]: L2 distance between the last ground truth and generated positions.

TABLE I: Ablation Study on KITTI Dataset

Algorithm	$\mathcal{E}_{ad}$ (m)	$\mathcal{E}_x$ (m)	$\mathcal{E}_y$ (m)	$\mathcal{E}_{fd}$ (m)	$\mathcal{E}_v$ (m/s)	$\mathcal{S}$ (m/s^3)
w/o intention	2.65	2.15	0.85	3.35	2.24	0.36
w/o v0	1.75	1.36	0.57	2.65	2.18	0.30
w/o cos	1.04	0.89	0.43	1.80	0.96	0.50
w/o HOS	2.02	1.62	0.79	2.80	1.78	0.64
w big HOS	1.88	1.47	0.56	2.25	0.89	**0.26**
Ours	**0.99**	**0.80**	**0.40**	**1.64**	**0.88**	0.28

- Average Longitudinal Error $\mathcal{E}_x$, Average Lateral Error $\mathcal{E}_y$, and Average Velocity Error $\mathcal{E}_v$ [5] represent mean longitudinal, lateral displacement and velocity scalar error between the ground truth and the corresponding quantities estimated from the generated trajectories respectively. Among them, $\mathcal{E}_v$ can measure the model's ability to model system dynamics.
- Smoothness of the trajectory, which is average jerk [25] of the trajectory $\mathcal{S}$.

Ablation Study: We exhibit the principle of utilizing our method by defining five ablated models from the original model as below:

- w/o intention: we skip the intention module and use raw RGB images as input instead of potential maps;
- w/o v0: we remove the initial velocity input to the model;
- w/o cos: we replace sinusoidal function to ReLU;
- w/o HOS: the model has no high-order supervision;
- w big HOS: big weights ($\lambda_1 = 1.0, \lambda_2 = 0.5$) of high-order supervision in the loss function;
- Ours: normal weights ($\lambda_1 = 0.2, \lambda_2 = 0.05$) of high-order supervision in the loss function.

We refer to Tab. I for checking the influence of different models on error metrics. When some stimuli are missing, performance drops significantly, which validates the importance of intention. The performance degenerates most in the model w/o v0, because the initial velocity v_0 reflects the current dynamics, without which the trajectory does not have boundary value condition. Results on w/o HOS and big HOS indicate that the trade-off between different order supervisions is indispensable. On the one hand, high-order supervision rectifies dynamic outputs of Neural Trajectory. w/o cos verifies a better approximation accuracy when the basis function is selected more appropriate for the dynamic system.

Comparative Study: For comparison, we investigate three recent baseline methods, and these models are introduced as follows:

- NI+DT and NI+DT+RNN: No intention (NI) input and discrete trajectory (DT) output. We follow the method in [15]. In NI+DT, the CNN module intends to extract features from sequential RGB image inputs, and the concatenated features are fed into fully connected layers to generate a discrete trajectory. In NI+DT+RNN, the network runs analogical to NI+DT except that the extracted features are fused by *Long Short-Term Memory (LSTM)* instead of simply concatenating them.

- DI+DT: Discrete intention (DI) input and discrete trajectory (DT) output. We follow the idea of [8] to give three discrete commands as discrete intentions to switch three different networks. Instead of generating control commands, we change it to output a series of discrete trajectory points and velocities.
- CI+DT and CI+DT+RNN: Continuous intention (CI) input and discrete trajectory (DT) output. The same structure to NI+DT and NI+DT+RNN, but we replace sequential RGB image inputs with sequential potential maps as continuous intention.
- VTGNet [5]: one of the state-of-the-art methods we compared, which uses discrete intention and generates discrete trajectories
- CI+DWA: The trajectory generation method used in our previous work [7]. We keep the pre-trained driving intention module and use the traditional dynamic window approach (DWA) to generate simple trajectories with fixed velocity and angular velocity to avoid obstacles and navigate to the destination.
- CI+Ploy: A learning-based parametric model that models trajectories as polynomials. For a fair comparison, we keep the driving intention module and only modify the trajectory generation module proposed in our method by removing time t from the input and changing the output to a K^{th}-order polynomial coefficients. We take $K = 5$ [25] for the experiment.
- CI+CT (Ours): Continuous intention (CI) input and continuous trajectory (CT) output method proposed in this paper.

From the results shown in Tab. II, we can clearly see that our network performs the best on all predictive metrics, which means the trajectories our method generates are more smooth and accurate in accordance with displacement-level metrics such as $\mathcal{E}_{ad}$ and high-order metrics like $\mathcal{E}_v$. Tab. II shows that DI+DT outperforms NI+DT and NI+DT+RNN, which indicates that the superiority of introducing intention. The intention helps to guide the direction of trajectory generation, resulting in the improvement of displacement-level metrics. Additionally, model performance is enhanced by introducing continuous intention rather than discrete commands. Most importantly, our model significantly decreases the high-order metrics on both KITTI and RobotCar. Thanks to the continuous representation, we manage to precisely model the smoothness and the intrinsic constraints between derivatives. For more details about the results, please refer to [19]. We also prove that our trajectory generation method has better prediction results than the polynomial method shown in Tab. II. We believe that the polynomial trajectory generation method introduces two types of errors: 1) Fitting error from approximating trajectories with polynomials. 2) Prediction error due to prediction polynomial coefficients. While our method only has prediction error, which can reduce the uncertainty of our model. This proves the superiority of our method over the parametric polynomial model for trajectory prediction. And it also proves the superiority of continuous trajectory generation methods, comparing with discrete trajectory generation methods.

TABLE II: Comparative Study Results on Two Benchmark Datasets

Dataset	Algorithm	$\mathcal{E}_{ad}$ (m)	$\mathcal{E}_x$ (m)	$\mathcal{E}_y$ (m)	$\mathcal{E}_{fd}$ (m)	$\mathcal{E}_v$ (m/s)	$\mathcal{S}$ (m/s^3)
KITTI	NI+DT [15]	4.91	4.53	1.11	8.54	3.88	0.96
	NI+DT+RNN [15]	4.85	4.46	0.90	8.44	3.79	0.87
	DI+DT [8]	4.43	3.76	1.30	6.72	3.21	0.70
	DI+CT	2.60	2.00	0.51	2.55	1.68	0.39
	CI+DT	3.96	3.76	0.56	7.27	3.96	0.87
	CI+DT+RNN	3.36	3.20	0.42	6.00	2.31	0.79
	VTGNet [5]	2.98	2.51	0.51	4.89	1.48	0.58
	CI+DWA [7]	2.27	1.55	1.29	3.80	2.17	5.13
	CI+Poly	1.22	1.03	0.44	2.46	1.27	0.33
	CI + CT(Ours)	**0.99**	**0.80**	**0.40**	**1.64**	**0.88**	**0.28**
RobotCar	NI+DT [15]	2.83	2.81	0.33	5.12	2.02	0.23
	NI+DT+RNN [15]	2.58	2.46	0.25	4.89	1.77	0.83
	DI+DT [8]	2.31	2.08	0.56	4.06	1.66	0.78
	DI+CT	1.51	1.00	0.46	1.58	0.84	0.23
	CI+DT	1.76	1.66	0.35	3.33	1.32	0.41
	CI+DT+RNN	1.75	1.56	0.37	3.15	1.41	0.60
	VTGNet [5]	0.88	0.78	0.28	1.77	0.56	0.79
	CI+DWA [7]	0.83	0.71	0.36	1.78	0.73	4.70
	CI+Poly	0.82	0.69	0.30	1.87	0.72	0.24
	CI + CT(Ours)	**0.76**	**0.55**	**0.26**	**1.43**	**0.42**	**0.16**

B. Closed-loop Experiments in Simulation

As a driving task, only open-loop validation is not sufficient, since the error is accumulated. To further validate the efficacy of our proposed method, we evaluate our method with closed-loop experiments both in the CARLA simulation and on a real-world vehicle platform.

Experiments Setup: Since we need to obtain highly interpretable and smooth trajectories as expert-provided data to imitate, we use human driving data as our training data rather than built-in AI driving data. We use Logitech G29 driving force-racing wheel as our data acquisition equipment, and collect about 2 hours of human driving data in CARLA simulation with a speed limit of 30 km/h. We collect data in Town01 with 4 different weather as *training condition* and test our method in Town02 with other 2 different weather as *testing condition*.

We compare the performance of our method with the performance of several previously proposed approaches [6], [8]–[11], [26]–[28] on the original CARLA benchmark [29] and the NoCrash benchmark [10]. The original CARLA benchmark allows us to compare algorithms on sets of strictly defined goal-directed navigation tasks. It provides 4 simple navigation tasks, the last of which has a few dynamic obstacles. The NoCrash benchmark is much more challenging. The vehicle is driven under the tasks of 3 different traffic conditions. Almost all current methods have a low success rate under the most difficult dense traffic and testing condition.

Comparative Study: The comparison results on the CARLA benchmark are shown in Tab. III. We achieve a 100% success rate on all tasks under training condition and 3 tasks under testing condition, outperforming all other methods. On the last task under testing conditions, our success rate is slightly lower than that of LaTeS. Overall, our method achieves competitive performance with the state-of-the-art methods.

We test our method and variants of our method 3 times on the NoCrash benchmark and the results are shown in Tab. IV.

TABLE III: CARLA Benchmark Evaluation Results on Success Rate (%)

Task	Condition	MT [9]	CIL [8]	CIRL [11]	CAL [26]	CILRS [10]	LSD [27]	LaTeS [6]	**Ours**†[1]
Straight	Training	96	98	98	**100**	96	-	-	**100**
One turn		87	89	97	97	92	-	-	**100**
Navigation		81	86	93	92	95	-	**100**	**100**
Nav. dynamic		81	83	82	83	92	-	**100**	**100**
Straight	New Town & New Weather	96	80	98	94	96	**100**	-	**100**
One turn		82	48	80	72	92	**100**	-	**100**
Navigation		78	44	68	68	92	98	98	**100**
Nav. dynamic		62	42	62	64	90	92	**98**	96

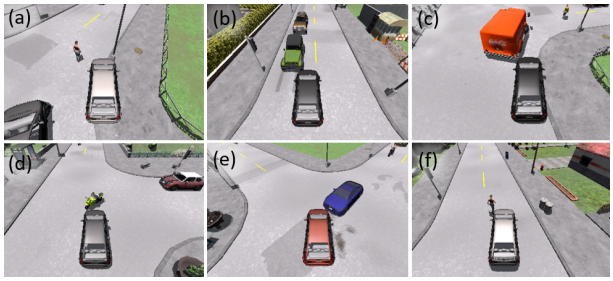

Fig. 4: Representative failure cases among all methods. (a) Hit a static obstacle; (b) Crash into the wrong side of the road; (c) Hit a car from the rear side; (d-f) Hit moving agents at the side. Most of our failure cases are (d-f).

TABLE IV: NoCrash Benchmark Evaluation Results on Success Rate (%)

Method	Training Condition			New Town & New Weather		
	Empty	Regular	Dense	Empty	Regular	Dense
CIL [8]	79 ± 1	60 ± 1	21 ± 2	24 ± 1	13 ± 2	2 ± 0
CAL [26]	81 ± 1	73 ± 2	42 ± 3	25 ± 3	14 ± 2	10 ± 0
MT [9]	84 ± 1	54 ± 2	13 ± 4	57 ± 0	32 ± 2	14 ± 2
CILRS [10]	97 ± 2	83 ± 0	42 ± 2	90 ± 2	56 ± 2	24 ± 8
LSD+ [27]	-	-	-	95 ± 1	65 ± 4	32 ± 3
LaTeS [6]	$\mathbf{100 \pm 0}$	$\mathbf{94 \pm 2}$	54 ± 3	83 ± 1	68 ± 7	29 ± 2
DA-RB [28]	-	-	66 ± 5	-	-	35 ± 2
DI+CT†[1]	97 ± 3	92 ± 2	73 ± 2	91 ± 2	70 ± 2	56 ± 3
CI+DT†	97 ± 2	83 ± 3	60 ± 3	93 ± 1	71 ± 2	50 ± 3
Ours†	$\mathbf{100 \pm 0}$	$\mathbf{94 \pm 2}$	$\mathbf{89 \pm 3}$	$\mathbf{100 \pm 0}$	$\mathbf{92 \pm 1}$	$\mathbf{67 \pm 2}$

In the empty task, our method achieves a 100% success rate both in training and testing conditions. In the regular traffic task and dense traffic task, the success rates of our method are also higher or equal to than other methods in both training and testing conditions. Several common and representative failure cases in all methods are shown in Fig. 4. In our test, there are very few cases of (a) and (b), which indicates the superiority of introducing continuous intention and long-term planning, and there are also few cases of (c), which indicates our method learns the system dynamics better. Almost all failure cases are collisions with side-on vehicles and pedestrians at intersections in our test, as shown in Fig. 4 (d-f). All methods encounter this problem. One reason is that agents in CARLA simulation are not strong enough to avoid obstacles at intersections. And we suspect that there is less data for this scenario in our training data and thus our model is less concerned with dynamic obstacles that are not in the driving intention area.

Generalization to other vehicle: Similar to [5], we also transfer our method from cars to motorcycles to validate the generalization ability for different vehicles of our method, because motorcycles have a smaller turning radius, greater acceleration, and its camera view tilts and shifts when turning and braking. We test in the regular task and training condition on the NoCrash benchmark, and we even do not change anything including controller parameters. The success rate is 91%, which is only 3% below the original 94%, validating the strong generalization ability of our method.

Robustness against computation latency: We also evaluate our method in different time delays to test the ability to eliminate the impact due to latency, comparing with synchronous driving architecture which is the built-in autonomous driving AI in CARLA simulation. To achieve this, we artificially add different time delays to the planning process in the regular task and training condition on the NoCrash benchmark. The result is shown in Fig. 5. Our method still has a comparable success rate when the latency is up to $500ms$, which validates the robustness against the computation latency of our method.

C. Closed-loop Experiments in Real World

We use a real-world vehicle with Ackermann steering for experiments in a campus environment, shown in Fig. 6. We build a route map of campus and plan a global route to guide the vehicle. We train our model and VTGNet [5] with real-world data and test the two models in the real-world closed-loop experiment, counting the number of

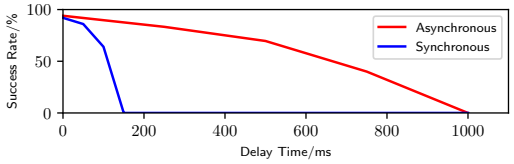

Fig. 5: Success rate under different time delays.

Fig. 6: Real-world vehicle platform and the actual travel route in the map.

[1]Adding † denotes the model is trained and evaluated on CARLA 0.9.9.4, while not adding it denotes that the model is trained and evaluated on CARLA $\leq$ 0.9.5.

human interruptions. The result is that VTGNet has 8 human interruptions that occur totally, while our method has 5 human interruptions that occur totally, which proves the superiority of our method.

We also use another dataset YQ21[1] for testing generalization ability of our method and VTGNet [5]. We directly transfer the models trained on the KITTI dataset to the YQ21 dataset for testing without re-training. The experimental result also shows our method has a stronger generalization ability. For more details, please refer to [19].

V. Conclusion

In this paper, we propose a hierarchical driving model with explicit intermediate representation to decompose the intention (path) and dynamics (trajectory), bringing better interpretability and without losing the advantages of end-to-end training. In addition, we propose a new representation of trajectory which is continuous and differentiable. However, our method has still some limitations: 1) Since our method is still imitation learning or called behavior cloning, it is difficult for our method to learn good causality from training data. 2)Similar to many current trajectory generation methods, our approach does not consider lanes, traffic lights, and traffic rules. We hope to address these issues in our future work.

Acknowledgement

This work was supported in part by the National Nature Science Foundation of China (61903332) and in part by the stable support project of State Administration of Science, Technology and Industry for National Defence Grant, PRC under grant No.HTKJ2019KL502005.

References

[1] S. Kuutti, R. Bowden, Y. Jin, P. Barber, and S. Fallah, "A survey of deep learning applications to autonomous vehicle control," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[2] W. Ding, L. Zhang, J. Chen, and S. Shen, "Safe trajectory generation for complex urban environments using spatio-temporal semantic corridor," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2997–3004, 2019.

[3] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[4] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2722–2730, 2015.

[5] P. Cai, Y. Sun, H. Wang, and M. Liu, "Vtgnet: A vision-based trajectory generation network for autonomous vehicles in urban environments," *arXiv preprint arXiv:2004.12591*, 2020.

[6] A. Zhao, T. He, Y. Liang, H. Huang, G. V. d. Broeck, and S. Soatto, "Lates: Latent space distillation for teacher-student driving policy learning," *arXiv preprint arXiv:1912.02973*, 2019.

[7] H. Ma, Y. Wang, R. Xiong, S. Kodagoda, and L. Tang, "Deepgoal: Learning to drive with driving intention from human control demonstration," *Robotics and Autonomous Systems*, p. 103477, 2020.

[8] F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–9, IEEE, 2018.

[9] Z. Li, T. Motoyoshi, K. Sasaki, T. Ogata, and S. Sugano, "Rethinking self-driving: Multi-task knowledge for better generalization and accident explanation ability," *arXiv preprint arXiv:1809.11100*, 2018.

[10] F. Codevilla, E. Santana, A. M. López, and A. Gaidon, "Exploring the limitations of behavior cloning for autonomous driving," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9329–9338, 2019.

[11] X. Liang, T. Wang, L. Yang, and E. Xing, "Cirl: Controllable imitative reinforcement learning for vision-based self-driving," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 584–599, 2018.

[12] A. Amini, I. Gilitschenski, J. Phillips, J. Moseyko, R. Banerjee, S. Karaman, and D. Rus, "Learning robust control policies for end-to-end autonomous driving from data-driven simulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1143–1150, 2020.

[13] D. Barnes, W. Maddern, and I. Posner, "Find your own way: Weakly-supervised segmentation of path proposals for urban autonomy," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 203–210, IEEE, 2017.

[14] N. Rhinehart, K. M. Kitani, and P. Vernaza, "R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 772–788, 2018.

[15] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2174–2182, 2017.

[16] Y. Sun, W. Zuo, and M. Liu, "See the future: A semantic segmentation network predicting ego-vehicle trajectory with a single monocular camera," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3066–3073, 2020.

[17] Z. Yang, Y. Zhang, J. Yu, J. Cai, and J. Luo, "End-to-end multi-modal multi-task vehicle control for self-driving cars with visual perceptions," in *2018 24th International Conference on Pattern Recognition (ICPR)*, pp. 2289–2294, IEEE, 2018.

[18] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating," in *Conference on Robot Learning*, pp. 66–75, PMLR, 2020.

[19] Y. Wang, D. Zhang, J. Wang, Z. Chen, Y. Wang, and R. Xiong, "Imitation learning of hierarchical driving model: from continuous intention to continuous trajectory," *arXiv preprint arXiv:2010.10393*, 2020.

[20] V. Sitzmann, J. N. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," *arXiv preprint arXiv:2006.09661*, 2020.

[21] M. Watterson and V. Kumar, "Safe receding horizon control for aggressive mav flight with limited range sensing," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3235–3240, IEEE, 2015.

[22] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on intelligent vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[23] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[24] D. Barnes, M. Gadd, P. Murcutt, P. Newman, and I. Posner, "The oxford radar robotcar dataset: A radar extension to the oxford robotcar dataset," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6433–6438, IEEE, 2020.

[25] H. Fan, F. Zhu, C. Liu, L. Zhang, L. Zhuang, D. Li, W. Zhu, J. Hu, H. Li, and Q. Kong, "Baidu apollo em motion planner," *arXiv preprint arXiv:1807.08048*, 2018.

[26] A. Sauer, N. Savinov, and A. Geiger, "Conditional affordance learning for driving in urban environments," *arXiv preprint arXiv:1806.06498*, 2018.

[27] E. Ohn-Bar, A. Prakash, A. Behl, K. Chitta, and A. Geiger, "Learning situational driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11296–11305, 2020.

[28] A. Prakash, A. Behl, E. Ohn-Bar, K. Chitta, and A. Geiger, "Exploring data aggregation in policy learning for vision-based urban autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11763–11773, 2020.

[29] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," *arXiv preprint arXiv:1711.03938*, 2017.

[1] https://tangli.site/projects/academic/yq21/