End-to-End Interactive Prediction and Planning with Optical Flow Distillation for Autonomous Driving

Hengli Wang¹, Peide Cai¹, Rui Fan², Yuxiang Sun³, and Ming Liu¹
¹ The Hong Kong University of Science and Technology
² University of California San Diego
³ The Hong Kong Polytechnic University

Abstract

With the recent advancement of deep learning technology, data-driven approaches for autonomous car prediction and planning have achieved extraordinary performance. Nevertheless, most of these approaches follow a non-interactive prediction and planning paradigm, hypothesizing that a vehicle's behaviors do not affect others. The approaches based on such a non-interactive philosophy typically perform acceptably in sparse traffic scenarios but can easily fail in dense traffic scenarios. Therefore, we propose an end-to-end interactive neural motion planner (INMP) for autonomous driving in this paper. Given a set of past surrounding-view images and a high definition map, our INMP first generates a feature map in bird’s-eye-view space, which is then processed to detect other agents and perform interactive prediction and planning jointly. Also, we adopt an optical flow distillation paradigm, which can effectively improve the network performance while still maintaining its real-time inference speed. Extensive experiments on the nuScenes dataset and in the closed-loop Carla simulation environment demonstrate the effectiveness and efficiency of our INMP for the detection, prediction, and planning tasks. Our project page is at sites.google.com/view/inmp-ofd.

1. Introduction

Autonomous driving aims at safely and efficiently maneuvering self-driving vehicles (SDVs) from a starting point to a target point with the input of sensor data and a pre-built map [15, 7, 4, 26]. Most existing approaches are designed to follow a “perception-prediction-planning” paradigm, as shown in Fig. 1 (a), where the perception module detects other agents from the sensor data, the prediction module estimates possible future trajectories of the detected agents, and the planning module generates a safe trajectory to drive the SDV towards the given target location based on the output from the perception and prediction modules [13]. The approaches designed under this paradigm are non-interactive, as the planning module is assumed to have no effects on the results of the prediction module. This implies that each SDV is a passive agent, and its behavior does not affect the other agents. Based on this philosophy, these non-interactive approaches typically perform acceptably in sparse traffic scenarios but can easily fail in dense traffic scenarios [25]. For instance, Fig. 1 (b) illustrates a dense traffic scenario where the non-interactive SDV can struggle merging into the left lane, while our interactive SDV can perform a satisfactory lane merge by reasoning about how other agents will react to its behaviors.
In this paper, we propose an end-to-end Interactive Neural Motion Planner (INMP), for autonomous driving. Our INMP, illustrated in Fig. 2, takes a set of past surrounding-view images and a high definition (HD) map as input to jointly 1) detect other agents and 2) perform interactive prediction and planning for the SDV. Specifically, we first lift these images into three dimensions (3-D), and then combine them with the HD map to generate a feature map in bird’s-eye-view (BEV) space. This BEV feature map is then processed to 1) detect other agents via a single-shot detection header and 2) jointly estimate the future trajectories of the detected agents and produce safe motion plans for the SDV via an interactive prediction and planning model with a joint probability distribution and a set of learnable costs. This paradigm enables the SDV to reason about how other agents will react to its behaviors, and thus can effectively improve the driving performance. Please note that the whole pipeline is differentiable, enabling end-to-end learning from raw sensor data to the outputs. In addition, we follow [40] and adopt a similar optical flow distillation paradigm to further improve the performance. Specifically, we refer to the above network as the student network (INMP-S) and additionally develop a teacher network (INMP-T), which adopts a similar architecture to the student network but takes optical flow as an additional type of input. We then distill the knowledge from the teacher network to the student network, which can effectively improve the student network performance while still maintaining its real-time inference speed.
illustrates the overview of the proposed approach. Our INMP can achieve competitive performance on the detection, prediction, and planning tasks. Moreover, the adopted optical flow distillation paradigm enables our student network to achieve a much faster inference speed than the teacher network with similar driving performance.

2. Related Work

2.1. Trajectory Prediction

Trajectory prediction aims to estimate the future trajectories of the agents based on their past states. The major challenges of this task are modeling the interactions between different agents and generating accurate multi-modal trajectory predictions. Traditional approaches generally achieve it based on manually designed models, e.g., the Kalman filter [22]. With the advancement of deep learning techniques, many data-driven approaches have achieved impressive performance in this field. These approaches typically use the past states to learn a latent representation for each agent, and model the interactions between different agents based on their latent representations [1, 21, 16]. Recently, some researchers have developed a new paradigm that takes the raw sensor data as input to jointly perform object detection and trajectory prediction [46, 47, 24]. These approaches usually use LiDARs, since trajectory prediction is often performed in BEV space and the point clouds provided by LiDARs meet this requirement inherently. Considering that images can provide more semantic information than point clouds and cameras are much cheaper than LiDARs, we take images as input in our approach. Extensive experiments have demonstrated that the proposed vision-based approach can achieve competitive performance compared with previous LiDAR-based approaches, as presented in Section 4.

2.2. Motion Planning

The goal of motion planning is to generate a trajectory to drive the SDV towards its given destination safely and efficiently. Traditional approaches generally sample a large set of candidate trajectories based on the input perception and prediction results [29, 11, 12, 42, 43], and then use a cost function to select the executed trajectory, which has the minimal cost [34]. Recently, many end-to-end approaches that directly map the raw sensor data to the planned trajectories or control commands have been proposed [15, 7, 4]. These approaches are optimized jointly from data, and thus can compensate the adverse effects caused by the accumulated errors in traditional approaches [23]. However, the end-to-end approaches are often criticized for their lack of interpretability, which makes these approaches hard to explain the generated behaviors and further leads to their limited applications in practice. To address it, some approaches have adopted the multi-task learning paradigm, jointly conducting detection, prediction, and planning tasks [46, 47, 33]. The generated intermediate results, i.e., detection and planning results, can effectively help people understand why the model can produce specific motion planning results. However, these approaches typically follow the non-interactive prediction and planning paradigm, and can easily fail in dense traffic scenarios, as mentioned above.

Recently, some researches have proposed the end-to-end interactive paradigm [31, 37, 25]. These approaches typically take the point clouds provided by LiDARs as input, and utilize joint probability distribution models to perform interactive prediction and planning. In this paper, we follow this paradigm and explore its feasibility and effectiveness when images are given as input.

2.3. Knowledge Distillation

Knowledge distillation aims at leveraging the dark knowledge of a teacher network to improve the performance of a student network with fewer parameters. This paradigm was first proposed in [18] for image classification. After that, researchers have presented more effective and efficient knowledge distillation techniques [32, 45]. Specifically, [32] proposed hint training (HT), which aims at training the intermediate representation of the student network such that it can mimic the latent representation of the teacher network. [45] defined attention maps for two networks and then forced the student network to mimic the attention maps of the teacher network. Knowledge distillation has been adopted in many other applications, e.g., object detection [6] and semantic segmentation [17], to improve their performance. In this paper, we follow [40] and adopt a similar optical flow distillation paradigm for autonomous driving. Different from [40], we utilize this technique for the detection, prediction, and planning tasks jointly. In addition, [40] adopts a non-interactive paradigm, while our INMP can perform interactive prediction and planning.

3. Methodology

Fig. 2 illustrates the overview of the proposed approach. Our INMP first generates a BEV feature map \( \mathcal{F}_b \), as introduced in Section 3.1. \( \mathcal{F}_b \) is then processed to 1) detect other agents and 2) perform interactive prediction and planning, as presented in Section 3.2 and Section 3.3, respectively. After that, Section 3.4 elaborates the proposed optical flow distillation paradigm. Finally, we introduce the training phase in Section 3.5.

3.1. BEV Feature Map Generation

Let \( I^k \in \mathbb{R}^{H \times W \times 3} \) denote the input RGB image, where \( t = t_0 - 4, \ldots, t_0 \) denotes the timestamp of the past five frames; and \( k = 1, \ldots, 6 \) denotes the six cameras used in our experiments. The six cameras with known extrinsic and intrinsic parameters roughly point in the forward,
forward-left, forward-right, backward, backward-left, and backward-right directions respectively. We also take the HD map $M_b$ that contains the road, lane and intersection information as input, since it can provide a strong prior about the driving scenario. Then, given all images in the past five frames $\{I^k_i\}_{i=6}^{1}$ and the current HD map $M_b$, we aim to generate a BEV feature map $F_b$, as presented in Fig. 3. $F_b$ plays an important role in the following object detection and interactive prediction and planning.

Considering that images are located in perspective-view space, we first conduct monocular depth estimation for each $I^k_i$, which builds a bridge between perspective-view space and BEV space. To achieve it, we follow [30] and generate contextual features at all possible depths for each pixel. Specifically, we associate each pixel with a set of $|D|\Delta d$ discrete depths, where $D = \{d_0 + \Delta d, \ldots, d_0 + |D|\Delta d\}$. Then, we use the known intrinsic parameters to produce a point cloud $P^k_i$ that contains $H \cdot W \cdot |D|$ 3-D points for each $I^k_i$. To obtain the contextual feature for each point in $P^k_i$, we first use an image backbone to generate a contextual feature $f\in\mathbb{R}^C$ and a distribution $\pi$ over the discrete depth set $D$ for each pixel $p$. Afterwards, the contextual feature $f_d\in\mathbb{R}^C$ for point $p_d$ is computed as a combination of the feature for the corresponding pixel and the discrete depth inference:

$$f_d = \pi_d \cdot f,$$

where $d \in D$ refers to any discrete depth in $D$.

For the teacher network, we incorporate optical flow information into $P^k_i$ to enhance the network’s capability to model dynamic relationships for performance improvement. Specifically, we use an existing optical flow estimation network [38] to compute the backward optical flow $O^k_i \in \mathbb{R}^{H \times W \times 2}$:

$$I^k_i(u,v) = I^k_{i-1}(u + O^k_i(u,v,1), v + O^k_i(u,v,2)).$$

$O^k_i$ can be regarded as containing the explicit past motion information from $I^k_{i-1}$ to $I^k_i$. Then, we use a flow backbone to produce a contextual feature $f' \in \mathbb{R}^C$ for each pixel $p$. After that, we concatenate $f'$ with $f$ and produce a new feature. Please note that the new feature is still denoted as $f$ for notational simplicity. However, $f$ in the teacher network contains the explicit motion information provided by the optical flow while $f$ in the student network does not. We then use (1) to compute a contextual feature $f_d \in \mathbb{R}^C$ for every point $p_d$ in the teacher network.

Then, we can use the known extrinsic parameters to aggregate $\{P^k_i\}_{i=1}^{6}$ into a large point cloud $P_i$ for each timestamp $i$. After that, we follow [20] to convert $P_i$ into “pillars”, which refer to voxels with infinite height. To be specific, we assign each point to its nearest pillar and use pooling operation to construct a feature map $F_i$, which contains the information in BEV space and can be processed by convolutional layers. We also use a map backbone to produce a feature map $F_m$, which is then concatenated with the features of all five past frames $\{F_i\}_{i=0}^{5}$ and processed by convolutional layers to generate the BEV feature map $F_b$, as shown in Fig. 3.

### 3.2. Object Detection

Given the BEV feature map $F_b$, we first detect other agents via a single-shot detection header. Specifically, following [27], we apply two convolutional layers on $F_b$ separately, one for classifying the class of a location, the other one for regressing the position offset, size, heading angle, and velocity of each agent. After that, we use a non-maximum suppression (NMS) operation [28] to obtain the bounding boxes and velocities of all other agents, which are then utilized to perform interactive prediction and planning.

The training loss for object detection $L_O$ is defined as a summation of a classification loss $L_{OC}$ and a regression loss $L_{OR}$, i.e., $L_O = L_{OC} + L_{OR}$. To be specific, in $L_{OC}$, we use a cross entropy classification loss and assign the label of each anchor based on its intersection over union (IoU) with any agent as follows:

$$L_{OC}(\hat{C}, C) = H(\hat{C}, C),$$

where $H(\cdot, \cdot)$ denotes the cross entropy; and $\hat{C}$ and $C$ denote the ground-truth and the predicted classification distribution, respectively. In our experiments, we detect two kinds of agents, i.e., vehicles and pedestrians. In addition, $L_{OR}$ is defined as a smooth $L_1$ loss between the regression ground truth $S$ and regression predictions $\hat{S}$ as follows:

$$L_{OR}(\hat{S}, S) = \sum_k SL_1(\hat{S}_k, S_k),$$

where $H(\cdot, \cdot)$ denotes the cross entropy; and $\hat{C}$ and $C$ denote the ground-truth and the predicted classification distribution, respectively. In our experiments, we detect two kinds of agents, i.e., vehicles and pedestrians. In addition, $L_{OR}$ is defined as a smooth $L_1$ loss between the regression ground truth $S$ and regression predictions $\hat{S}$ as follows:

$$L_{OR}(\hat{S}, S) = \sum_k SL_1(\hat{S}_k, S_k),$$

where $H(\cdot, \cdot)$ denotes the cross entropy; and $\hat{C}$ and $C$ denote the ground-truth and the predicted classification distribution, respectively. In our experiments, we detect two kinds of agents, i.e., vehicles and pedestrians. In addition, $L_{OR}$ is defined as a smooth $L_1$ loss between the regression ground truth $S$ and regression predictions $\hat{S}$ as follows:
where $SL_1(\cdot, \cdot)$ denotes the smooth $L_1$ loss; and the regression state set contains the position offsets in two dimensions, the width and height of the bounding box, the sine and cosine value of the orientation angle, and the velocities in two dimensions.

### 3.3. Interactive Prediction and Planning

Given the object detection results, we then focus on generating $T = \{\tau_0, \tau_1, \ldots, \tau_N\}$, which contains the planned trajectory of the SDV $\tau_0$ and the trajectory predictions of $N$ detected agents $T_r = \{\tau_1, \ldots, \tau_N\}$. Considering that performing prediction and planning in continuous space can consume much computational resources, we follow [46, 47, 25] and sample trajectories in a discrete space, which contains $K$ possible candidate trajectories for each trajectory $\tau \in T$. The adopted trajectory sampler takes the past trajectories of each agent as input, and generates a set of straight lines, circular curves and euler spirals as candidate trajectories. Now, the generation of each trajectory $\tau \in T$ is transformed to a classification problem.

To achieve it, we first define a joint probability distribution over the prediction and planning results $T$ conditioned on the environmental context $X$ as follows:

$$p(T|X; w) = \frac{1}{N} \exp (-E(T, X; w)), \quad (5)$$

where $N$ is a normalizer; $X$ includes the BEV feature map $T_b$ and the past trajectories of each agent; $E(T, X; w)$ denotes the defined joint energy of the prediction and planning results $T$; and $w$ denotes the parameters of the model.

To be specific, $E(T, X; w)$ is defined as a summation of an agent-specific term $E_a$, a safety term $E_s$ and a goal-directed term $E_g$ as follows:

$$E(T, X; w) = \sum_{i=0}^{N} E_a(\tau_i; X; w) + \sum_{i,j} E_s(\tau_i, \tau_j) + E_g(\tau_0), \quad (6)$$

where $E_a$ is used to evaluate all $K$ candidate trajectories for each agent; $E_s$ is designed to penalize the occurrence of dangerous cases such as collision; and $E_g$ is utilized to encourage the SDV to follow the input high-level route. Specifically, the BEV feature map $T_b$ is combined with the candidate trajectories and then processed via a multi-layer perceptron (MLP) to produce a $K \times (N + 1)$ matrix of evaluation scores for $E_a$. Moreover, we follow [34] and define $E_s$ as a summation of a collision term and a safety distance violation term. The former will present 1 if the collision between a pair of future trajectories happens and 0 if not; while the latter is defined as a squared penalty within the safety distance of each agent's bounding box, scaled by the velocity of the SDV. Additionally, $E_g$ is defined as the average projected distance between the planned trajectory of the SDV $\tau_0$ and the input high-level route.

Then, we can determine the planned trajectory of the SDV $\tau_0$ by selecting the candidate trajectory with the minimal cost of a defined cost function $f_1$ as follows:

$$\tau_0^* = \arg \min_{\tau_0} f_1(T_a; X; w). \quad (7)$$

Compare with the non-interactive approaches, our INMP enables the SDV to reason about how other agents will react to its behaviors, i.e., considering the trajectory predictions of the other agents $T_r$ conditioned on $\tau_0$ in the planning objective. Based on this philosophy, the cost function $f_1$ is defined as an expectation of the joint energy over the trajectory prediction distribution of the other agents conditioned on the planned trajectory of the SDV, as follows:

$$f_1(T_a, X; w) = \mathbb{E}_{T_r \sim p(T_r|\tau_0, X; w)}[E(T_a, X; w)]. \quad (8)$$

By substituting (6) into (8), we can have:

$$f_1 = E_a(\tau_0) + E_g(\tau_0) + \mathbb{E}_{T_r \sim p(T_r|\tau_0, X; w)} \sum_{i=1}^{N} E_a(\tau_i) + \sum_{i=1}^{N} \sum_{j=1}^{N} E_s(\tau_i, \tau_j), \quad (9)$$

where $f_1$ is short for $f_1(T_a, X; w)$; and $E_a(\tau_i)$ is short for $E_a(\tau_i; X; w)$. Since [25] have shown that excluding the interaction term between the other agents, i.e., $\sum_{i=1, j=1}^{N} E_s(\tau_i, \tau_j)$, does not lead to significant performance degradation, we follow it and also exclude this term in (9) for computational efficiency. Then, for any given $\tau_0$, we can compute the expectation directly by simplifying the terms into the marginal probabilities as follows:

$$f_1 = E_a(\tau_0) + E_g(\tau_0) + \sum_{i, \tau_i} \mathbb{E}_{\tau_i|\tau_0, X; w} p(\tau_i | \tau_0, X; w) E_a(\tau_i)$$

$$+ \sum_{i, \tau_i} \mathbb{E}_{\tau_i|\tau_0, X; w} p(\tau_i | \tau_0, X; w) E_s(\tau_0, \tau_i). \quad (10)$$

Now, we can successfully perform the proposed interactive prediction and planning, i.e., estimating the future trajectories of the detected agents and producing safe motion plans for the SDV jointly, by computing these marginal probabilities via loopy belief propagation (LBP) [43] efficiently.

Considering that the generation of each trajectory $\tau \in T$ is transformed to a classification problem, we define the training loss for interactive prediction and planning $L_P$ as follows:

$$L_P = \sum_{i} H(p(\tilde{\tau}_i), p(\tau_i)) + \sum_{i,j} H(p(\tilde{\tau}_i, \tilde{\tau}_j), p(\tau_i, \tau_j)), \quad (11)$$

where $\tilde{T} = \{\tilde{\tau}_0, \tilde{\tau}_1, \ldots, \tilde{\tau}_N\}$ denotes the prediction and planning ground truth; and $p(\cdot)$ and $p(\cdot, \cdot)$ denote the marginal
probabilities. Please note that we follow [25] and define $U(\tau_i)$ as a set of the trajectories close to $\tau_i$. In (11), we only compute the loss for $\tau_i \notin U(\tau_i)$, since we do not want to penalize the trajectory close to the ground truth.

### 3.4. Optical Flow Distillation Paradigm

As mentioned previously, optical flow can provide explicit motion information, leading to significant performance improvement for the teacher network. However, the computation of the optical flow seriously hinders the whole pipeline to achieve real-time performance. We then follow [40] and distill the knowledge from the teacher network to the student network, which can effectively enhance the student network while still maintaining its real-time performance. The distillation loss $L_D$ is defined as follows:

$$L_D = \lambda_{DO} L_{DO} + \lambda_{DP} L_{DP} + \lambda_{DF} L_{DF}. \quad (12)$$

where $L_{DO}$, $L_{DP}$ and $L_{DF}$ denote the distillation loss for object detection, interactive prediction and planning, and the BEV feature map $\mathcal{F}_b$, respectively; and $\lambda_{DO}$, $\lambda_{DP}$ and $\lambda_{DF}$ are the hyperparameters that scale the three loss terms.

Similar to $L_O$, $L_{DO}$ is defined as a summation of a classification distillation loss $L_{DOC}$ and a regression loss $L_{DOR}$, i.e., $L_{DO} = L_{DOC} + L_{DOR}$. We follow [18] and define $L_{DOC}$ as follows:

$$L_{DOC} = L_{OC}(C^T, C^S) = H(C^T, C^S), \quad (13)$$

where $C^T$ and $C^S$ denote the predicted classification distributions of the teacher and student networks, respectively. Different from $\hat{C}$ in (3) that can only provide hard information, $C^T$ can provide useful soft information to effectively improve the student network. In addition, inspired by [6], we design $L_{DOR}$ as follows:

$$L_{DOR} = \sum_k \left\{ \begin{array}{ll} SL_1 \left( S^T_k, S^S_k \right), & \text{if} \| S^T_k - S^S_k \|_1 > \| S^T_k - S^T_i \|_1, \\ 0, & \text{otherwise.} \end{array} \right. \quad (14)$$

where $\| \cdot \|_1$ denotes the $L_1$ norm; and $S^T$ and $S^S$ denote the regression predictions of the teacher and student networks, respectively. $L_{DOR}$ encourages the student network to be close or better than the teacher network, but does not push the student once it reaches the teacher’s performance.

Moreover, we define $L_{DP}$ as follows:

$$L_{DP} = \left\{ \begin{array}{ll} L_P(T^T, T^S), & \text{if} \sum_i D(\tilde{T}_i, \tau_i^S) > \sum_i D(\tilde{T}_i, \tau_i^T), \\ 0, & \text{otherwise.} \end{array} \right. \quad (15)$$

where $T^T$ and $T^S$ denote the prediction and planning results of the teacher and student networks, respectively; $\tilde{T}_i$ denotes the trajectory ground truth; $\tau_i^T$ and $\tau_i^S$ denote the trajectories of the teacher and student networks with the minimal cost of $f_i$, respectively; and $D(\cdot, \cdot)$ measures the average projected distance between two trajectories. Similar to $L_{DOR}$, $L_{DP}$ also encourages the student network to perform better than the teacher network but does not push the student too much.

Considering that $\mathcal{F}_b$ of the teacher network incorporates the explicit motion information provided by the optical flow while $\mathcal{F}_b$ of the student network does not, we follow HT [32] and further design $L_{DF}$ as:

$$L_{DF} = \| \mathcal{F}_b^T - \mathcal{F}_b^S \|_1, \quad (16)$$

where $\mathcal{F}_b^T$ and $\mathcal{F}_b^S$ denote the BEV feature maps $\mathcal{F}_b$ of the teacher and student networks, respectively. $L_{DF}$ encourages the student network to mimic the BEV feature map $\mathcal{F}_b$ of the teacher network.

### 3.5. Training Phase

In the training phase, we first use the following teacher training loss $L^T$ to train the teacher network:

$$L^T = \lambda_O L^T_O + \lambda_P L^T_P, \quad (17)$$

where $L^T_O = L_{OC}(\hat{C}, C^T) + L_{OR}(\hat{S}, S^T)$; $L^T_P = L_P(\hat{T}, T^T)$; and $\lambda_O$ and $\lambda_P$ are the hyperparameters that scale the two loss terms.

After that, we utilize the following student training loss $L^S$ to train the student network based on the trained teacher network:

$$L^S = \lambda_O L^S_O + \lambda_P L^S_P + \lambda_D L_D, \quad (18)$$

where $L^S_O = L_{OC}(\hat{C}, C^S) + L_{OR}(\hat{S}, S^S)$; $L^S_P = L_P(\hat{T}, T^S)$; and $\lambda_O$, $\lambda_P$ and $\lambda_D$ are the hyperparameters that scale the three loss terms.

### 4. Experimental Results and Discussions

#### 4.1. Datasets and Implementation Details

In our experiments, we first evaluate the performance of our approach for object detection and trajectory prediction on the nuScenes dataset [2], which contains around 1000 human driving sequences. The dataset is split into a training, a validation and a test set that consists of 18072, 8019 and 8033 samples, respectively. The best-performing networks are selected on the validation set and evaluated on the test set. We also conduct closed-loop evaluation in the Carla simulation environment [9]. Specifically, we first collect a large-scale driving dataset on different maps with different weather and illumination conditions, e.g., clear, rainy, daytime and sunset. We also set random roaming pedestrians and vehicles, which are controlled by the Carla simulator [9]. The dataset is split into a training set with 200K samples and a validation set with 50K samples. Finally,
each network is evaluated thoroughly with 1800 episodes (around 1000 km) in a closed-loop manner.

For the implementation details, we adopt EfficientNet-B0 [39] as the map, flow and image backbones. In addition, our INMP takes the information of past 2 s as input and performs interactive prediction and planning for the future 4 s. We adopt the Adam optimizer [19] with an initial learning rate of $10^{-3}$ to train our INMP-T and INMP-S on two NVIDIA GeForce RTX 2080 Ti GPUs. Moreover, we train the student network without the proposed optical flow distillation paradigm, referred to as INMP-S-ND, for performance comparison.

### 4.2. Object Detection Results

We follow the nuScenes benchmark [2] and adopt the average precision (AP) at the 1 m distance threshold as our evaluation metric. We compute the AP for vehicles and pedestrians respectively, and also compute its mean value (mAP) across the two classes. The evaluation results are shown in Table 1. It is evident that the three variants of our INMP all achieve competitive performance compared to the existing vision-based approaches, and our INMP-T achieves the best performance. In addition, our INMP-S presents a better performance than INMP-S-ND and a similar performance to INMP-T thanks to the adopted optical flow distillation paradigm. The qualitative results in Fig. 4 also confirm the above conclusions. Moreover, we adopt INMP-S in practice due to its real-time inference speed, as analyzed in Section 4.4.

### 4.3. Trajectory Prediction Results

We use the $L_2$ distance at $t = 4s$ [47] and the minMSD (5 agents and $K = 12$) [31] for performance comparison between trajectory prediction approaches. These two metrics both measure the distance between the trajectory prediction and the ground truth for the correctly detected agents, and the evaluation results are presented in Table 2. We can see that the conclusions in Section 4.2 also hold for the trajectory prediction task. Excitingly, our INMP-S and INMP-T can even achieve competitive performance compared to existing LiDAR-based approaches, which strongly demonstrates that our energy-based model can effectively generate accurate trajectory predictions.

### 4.4. Closed-loop Evaluation Results

We adopt the success rate (SR) and right lane rate (RL) as our evaluation metrics [5]. SR is defined as the proportion of the successfully finished episodes to the total testing episodes, while RL is defined as the proportion of the period when the SDV drives in the input high-level route to the total driving time. To verify the effectiveness of our INMP, we further develop a non-interactive neural motion planner (NINMP) by using the following cost function: $f_{N\text{INMP}}(T, X; w) = \mathbb{E}_{T_{\text{n}} \sim P_{\text{T}_{\text{n}}}} [C(T, X; w)]$. Different from (8), $f_{N\text{INMP}}$ considers the trajectory predictions of the other agents $T_{\text{n}}$ unconditioned on the planned trajectory of the SDV $T_0$. Moreover, we record the inference time of each approach on the NVIDIA GeForce RTX 2080 Ti GPU.

Table 3 presents the evaluation results, where it is evident that the conclusions in Section 4.2 also hold for the closed-loop...
Table 3. Closed-loop evaluation results in the Carla simulator [9].
Best results are shown in the bold type.

<table>
<thead>
<tr>
<th>Approach</th>
<th>SR (%)</th>
<th>RL (%)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIL [7]</td>
<td>60.72</td>
<td>82.97</td>
<td>0.07</td>
</tr>
<tr>
<td>VTP [3]</td>
<td>76.11</td>
<td>80.89</td>
<td>0.08</td>
</tr>
<tr>
<td>NINMP-S</td>
<td>80.94</td>
<td>82.31</td>
<td><strong>0.05</strong></td>
</tr>
<tr>
<td>NINMP-S-ND</td>
<td>81.39</td>
<td>85.63</td>
<td>0.06</td>
</tr>
<tr>
<td>INMP-S (Used)</td>
<td>91.28</td>
<td>93.96</td>
<td>0.06</td>
</tr>
<tr>
<td>INMP-T</td>
<td><strong>92.33</strong></td>
<td><strong>95.20</strong></td>
<td>0.21</td>
</tr>
</tbody>
</table>

Table 4. Closed-loop evaluation results of our INMP-S with some of the loss terms disabled in the Carla simulator [9]. Best results are shown in the bold type.

<table>
<thead>
<tr>
<th>Variant</th>
<th>$\mathcal{L}_{DO}$</th>
<th>$\mathcal{L}_{DP}$</th>
<th>$\mathcal{L}_{DF}$</th>
<th>SR (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) INMP-S</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>81.39</td>
</tr>
<tr>
<td>(b) INMP-S</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>89.61</td>
</tr>
<tr>
<td>(c) INMP-S</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>83.39</td>
</tr>
<tr>
<td>(d) INMP-S</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
<td>87.83</td>
</tr>
<tr>
<td>(e) INMP-S (Used)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td><strong>91.28</strong></td>
</tr>
</tbody>
</table>

Figure 5. An example scenario in the Carla simulator [9], where the SDV is trying to merge into the left lane. The non-interactive SDV (NINMP-S) drifts slowly to the left lane instead of completing a lane merge, while our interactive SDV (INMP-S) can complete a satisfactory lane merge efficiently without bringing safety risk and inconvenience to other agents. The SDV is marked with an orange dashed box in the BEV perspective.

5. Conclusions

In this paper, we proposed INMP, an end-to-end interactive neural motion planner for autonomous driving. Given a set of past surrounding-view images and a high definition map, our INMP first generated a feature map in the SDV’s planning objective, our interactive prediction and planning model can effectively improve the driving performance for the SDV. Fig. 5 presents an example scenario, where the SDV is trying to merge into the left lane. It is evident that the non-interactive SDV (NINMP-S) drifts slowly to the left lane instead of completing a lane merge, while our interactive SDV (INMP-S) can complete a satisfactory lane merge efficiently without bringing safety risk and inconvenience to other agents. All the analysis have demonstrated the effectiveness and efficiency of our proposed approach.

4.5. Ablation Study

We conduct ablation studies to demonstrate the effectiveness of our selection in the loss functions. Specifically, we take INMP-S as the baseline, and test its closed-loop performance with different combinations of loss terms in the Carla simulator [9]. The evaluation results are presented in Table 4, where it can be observed that our paradigm that employs the three loss terms together achieves the best performance for our INMP-S.

Acknowledgements

This work was supported in part by the Collaborative Research Fund by Research Grants Council Hong Kong under Project C4063-18G, in part by the Department of Science and Technology of Guangdong Province Fund under Project GDST20EG54, and in part by the Zhongshan Municipal Science and Technology Bureau Fund under project ZSST21EG06, awarded to Prof. Ming Liu.
References


[26] Tianyu Liu, Qinghai Liao, Lu Gan, Fulong Ma, Jie Cheng, Xupeng Xie, Zhe Wang, Yingbing Chen, Yilong Zhu, Shuyang Zhang, Zhengyong Chen, Yang Liu, Meng Xie, Yang Yu, Zitong Guo, Guan Li, Peidong Yuan, Dong Han, Yuying Chen, Haoyang Ye, Jianhao Jiao, Peng Yun, Zhenhua Xu, Hengli Wang, Huaiyang Huang, Sukai Wang, Peide Cai,


