ViP3D: End-to-end Visual Trajectory Prediction via 3D Agent Queries

Junru Gu[1*] Chenxu Hu[1*] Tianyuan Zhang[2,3] Xuanyao Chen[2,4]
Yilun Wang[5] Yue Wang[6] Hang Zhao[1,2†]

[1]IIIS, Tsinghua University [2]Shanghai Qi Zhi Institute
[3]CMU [4]Fudan University [5]Li Auto [6]MIT

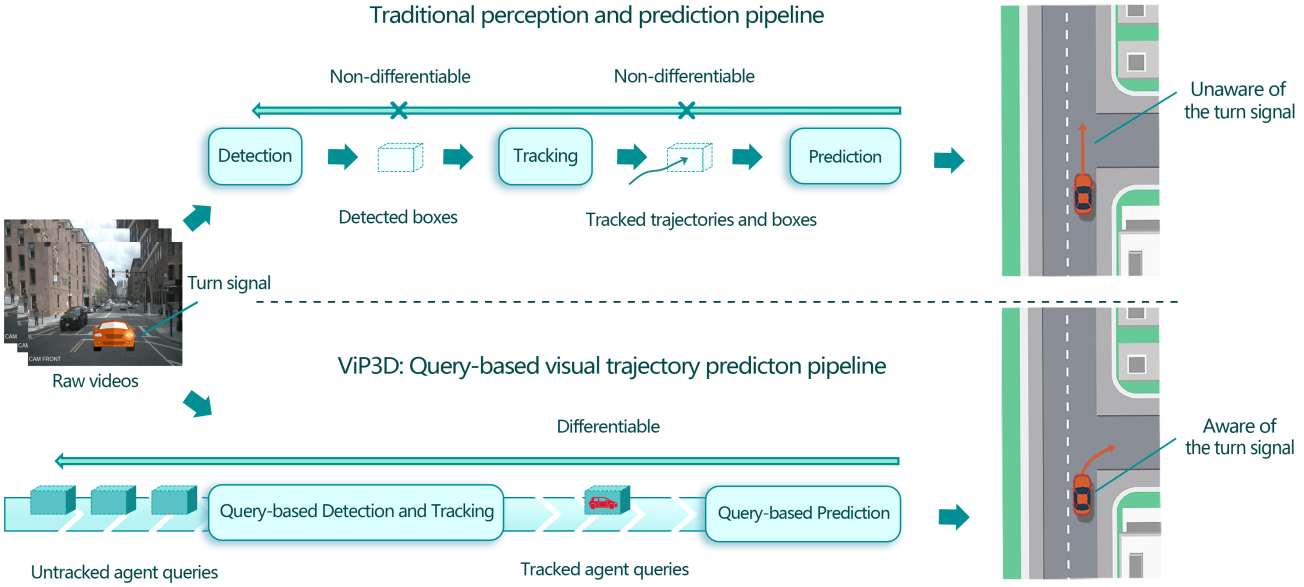

Figure 1. Comparison of a traditional multi-stage perception-prediction pipeline in autonomous driving and our proposed ViP3D. The traditional pipeline involves multiple non-differentiable modules, *i.e.*, detection, tracking, and prediction. ViP3D uses 3D agent queries as the main thread of the pipeline, enabling end-to-end future trajectory prediction from raw video frame inputs. The novel design improves trajectory prediction performance by effectively leveraging fine-grained visual information such as the turning signals of vehicles.

Abstract

Perception and prediction are two separate modules in the existing autonomous driving systems. They interact with each other via hand-picked features such as agent bounding boxes and trajectories. Due to this separation, prediction, as a downstream module, only receives limited information from the perception module. To make matters worse, errors from the perception modules can propagate and accumulate, adversely affecting the prediction results. In this work, we propose ViP3D, a query-based visual trajectory prediction pipeline that exploits rich information from raw videos to directly predict future trajectories of agents in a scene. ViP3D employs sparse agent queries to detect, track, and predict throughout the pipeline, making it the first fully differentiable vision-based trajectory prediction approach. Instead of using historical feature maps and trajectories, useful information from previous timestamps is encoded in agent queries, which makes ViP3D a concise streaming prediction method. Furthermore, extensive experimental results on the nuScenes dataset show the strong vision-based prediction performance of ViP3D over traditional pipelines and previous end-to-end models.[1]

1. Introduction

An autonomous driving system should be able to perceive agents in the current environment and predict their future behaviors so that the vehicle can navigate the world

*Equal contribution.
†Corresponding to: hangzhao@mail.tsinghua.edu.cn

[1]Code and demos are available on the project page: https://tsinghua-mars-lab.github.io/ViP3D

safely. Perception and prediction are two separate modules in the existing autonomous driving software pipeline, where the interface between them is often defined as hand-picked geometric and semantic features, such as historical agent trajectories, agent types, agent sizes, *etc*. Such an interface leads to the loss of useful perceptual information that can be used in trajectory prediction. For example, tail lights and brake lights indicate a vehicle's intention, and pedestrians' head pose and body pose tell about their attention. This information, if not explicitly modeled, is ignored in the existing pipelines. In addition, with the separation of perception and prediction, errors are accumulated and cannot be mitigated in later stages. Specifically, historical trajectories used by trajectory predictors come from an upstream perception module, which inevitably contains errors, leading to a drop in the prediction performance. Designing a trajectory predictor that is robust to upstream output errors is a non-trivial task [60].

Recent works such as IntentNet [3], FaF [34], PnPNet [30] propose end-to-end models for LiDAR-based trajectory prediction. They suffer from a couple of limitations: (1) They are not able to leverage the abundant fine-grained visual information from cameras; (2) these models use convolutional feature maps as their intermediate representations within and across frames, thus suffering from non-differentiable operations such as non-maximum suppression in object decoding and object association in multi-object tracking.

To address all these challenges, we propose a novel pipeline that leverages a query-centric model design to predict future trajectories, dubbed **ViP3D** (**Vi**sual trajectory **P**rediction via **3D** agent queries). ViP3D consumes multi-view videos from surrounding cameras and high-definition maps, and makes agent-level future trajectory prediction in an end-to-end and concise streaming manner, as shown in Figure 1. Specifically, ViP3D leverages 3D agent queries as the interface throughout the pipeline, where each query can map to (at most) an agent in the environment. At each time step, the queries aggregate visual features from multi-view images, learn agent temporal dynamics, model the relationship between agents, and finally produce possible future trajectories for each agent. Across time, the 3D agent queries are maintained in a memory bank, which can be initialized, updated and discarded to track agents in the environment. Additionally, unlike previous prediction methods that utilize historical agent trajectories and feature maps from multiple historical frames, ViP3D only uses 3D agent queries from one previous timestamp and sensor features from the current timestamp, making it a concise streaming approach.

In summary, the contribution of this paper is three-fold:

1. ViP3D is the first **fully differentiable vision-based** approach to predict future trajectories of agents for autonomous driving. Instead of using hand-picked features like historical trajectories and agent sizes, ViP3D leverages the rich and fine-grained visual features from raw images which are useful for the trajectory prediction task.
2. With **3D agent queries as interface**, ViP3D explicitly models agent-level detection, tracking and prediction, making it interpretable and debuggable.
3. ViP3D is a concise model with **high performance**. It outperforms a wide variety of baselines and recent end-to-end methods on the visual trajectory prediction task.

2. Related Work

3D Detection. There are a great number of works on 3D object detection and tracking from point clouds [26, 42, 64]. In this paper, we focus on 3D detection and tracking from cameras. Monodis [46] and FCOS3D [52] learn a single-stage object detector with instance depth and 3D pose predictions on monocular images. Pseudo-LiDAR [53] first predicts depth for each image pixel, then lifts them into the 3D space, and finally employs a point cloud based pipeline to perform 3D detection. DETR3D [54] designs a sparse 3D query-based detection model that maps queries onto 2D multi-view images to extract features. BEVFormer [28] and PolarFormer [24] further propose a dense query-based detection model. Lift-Splat-Shoot [40] projects image features into BEV space by predicting depth distribution over pixels, BEVDet [22] performs 3D object detection on top of it. Furthermore, PETR [32] develops an implicit approach to transform 2D image features into BEV space for 3D detection.

3D Tracking. The majority of 3D tracking approaches follow the tracking-by-detection pipeline [38, 55]. These methods first detect 3D objects, then associate existing tracklets with the new detections. CenterTrack [57, 63] uses two consecutive frames to predict the speed of each detection box, then performs association using only ℓ_2 distances of the boxes. Samuel *et al*. [45] uses PMBM filter to estimate states of tracklets and match them with new observations. DEFT [4] uses a learned appearance matching network for association, together with an LSTM estimated motion to eliminate implausible trajectories. QD3DT [21] uses cues from depth-ordering and learns better appearance features via contrastive learning. MUTR3D [61] introduces track queries to model objects that appear in multiple cameras across multiple frames.

Trajectory Prediction. Several seminal trajectory prediction works have studied historical trajectory and map geometry encoding using graph neural networks [12, 29] and Transformers [36, 37, 51]. To make multiple plausible future predictions [5, 8, 10, 11, 39, 39], variety loss is a regression-

based method that only optimizes the closest predicted trajectory during training. A Divide-And-Conquer [35] approach is also a good initialization technique to produce diverse outputs. Modeling uncertainty using latent variables [2, 7, 19, 27, 43, 48, 49, 56, 58] is another popular approach, which predicts different future trajectories by randomly sampling from the latent variables. Goal-based methods recently achieve outstanding performance by first predicting the intentions of agents, such as the endpoint of trajectories [14–16, 50, 62], lanes to follow [25, 29, 47], and then predicting trajectories conditioning on these goals.

End-to-End Perception and Prediction. In the last couple of years, there has been growing interest in jointly optimizing detection, tracking, and prediction. FaF [34] employs a single convolutional neural network to detect objects from LiDAR point clouds, and forecast their corresponding future trajectories. IntentNet [3] adds high-level intention output to this framework. More recently, Phillips *et al*. [41] further learns localization together with perception and prediction. FIERY [20] predicts future BEV occupancy heatmaps from visual data directly. Mostly related to our work is PnPNet [30], which explicitly models tracking in the loop. Our method is related to these methods in the sense that we also perform end-to-end prediction based on sensor inputs. However, they all rely on BEV feature maps or heatmaps as their intermediate representation, which leads to unavoidable non-differentiable operation while going from dense feature maps to instance-level features, such as non-maximum suppression (NMS) in detection, and association in tracking. Our method, on the other hand, employs sparse agent queries as representation throughout the model, greatly improving the differentiability and interpretability.

3. Method

Overall, ViP3D leverages a query-centric model design to address the trajectory prediction problem from raw videos in an end-to-end manner. As shown in Figure 2, 3D agent queries serve as the main thread across time. At each time step, a query-based detection and tracking module extracts multi-view image features from surrounding cameras to update agent queries, forming a set of tracked agent queries. The tracked agent queries potentially contain much useful visual information, including the motion dynamics and visual characteristics of the agents. After that, a query-based prediction module takes the tracked agent queries as input and associates them with HD map features, and finally outputs agent-wise future trajectories. Over time, analogous to traditional trackers, the 3D agent queries are initialized, updated and discarded within a query memory bank, making ViP3D work in a concise streaming fashion. The design details of each module are explained in the following subsections.

3.1. Query-based Detection and Tracking

For each input frame, a query-based detection and tracking first extracts visual features from surrounding cameras, as shown in the upper part of Figure 2. Specifically, we follow DETR3D [54] to extract 2D features from multi-view images and use cross attention to update agent queries. For temporal feature aggregation, inspired by MOTR [59], we design a query-based tracking scheme with two key steps: query feature update and query supervision. Agent queries are updated across time to model the motion dynamics of agents.

3.1.1 Query Feature Update

Each agent query corresponds to at most one agent that appeared in the scene. We use $\mathbf{Q}$ to denote a set of agent queries, which are initialized as learnable embeddings with 3D reference points [54]. At each time step, we first extract 2D image features of surrounding cameras via ResNet50 [18] and FPN [31]. Then we project the 3D reference points of agent queries onto the 2D coordinates of multi-view images using camera intrinsic and extrinsic transformation matrices. Finally, we extract the corresponding image features $\mathbf{L}$ to update the agent queries via cross attention. Let $\mathbf{Q}'_t = \mathbf{Q}_t\mathbf{W}^Q, \mathbf{K} = \mathbf{L}\mathbf{W}^K, \mathbf{V} = \mathbf{L}\mathbf{W}^V$ be query / key / value vectors, respectively, where $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V \in \mathbb{R}^{d_h \times d_k}$ are the matrices for linear projection, $t \in \{1, \dots, T\}$ is the current time step, d_k is the dimension of query / key / value vectors. Then the cross attention is: $\tilde{\mathbf{Q}}_t = \mathrm{softmax}\left(\frac{\mathbf{Q}'_t\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}$. Finally, we update the agent queries: $\mathbf{Q}'_t = \mathrm{FFN}\left(\mathbf{Q}_t + \tilde{\mathbf{Q}}_t\right)$, where FFN is a two-layer MLP with layer normalization.

3.1.2 Query Supervision

Since each agent query corresponds to at most one certain agent, supervision is required at each time step to make sure each query extracts features of the same agent across different historical frames. There are two types of queries. One is the matched queries that have been associated with ground truth agents before this time step. The other is the empty queries that have not been associated with any ground truth agent. Suppose we have done association at time step $t-1$, and now we perform association at time step t. For the matched queries, we assign the same ground truth agents to them as before: $\mathbf{Q}_{\text{matched}} \cong \mathcal{A}_{t-1}$, where $\mathcal{A}_{t-1}$ denotes the ground truth agents at time step $t-1$. If an agent disappears at time step t, we assign an empty label to supervise the corresponding agent query and reinitialize it as an empty unmatched query for later use. For the unmatched queries,

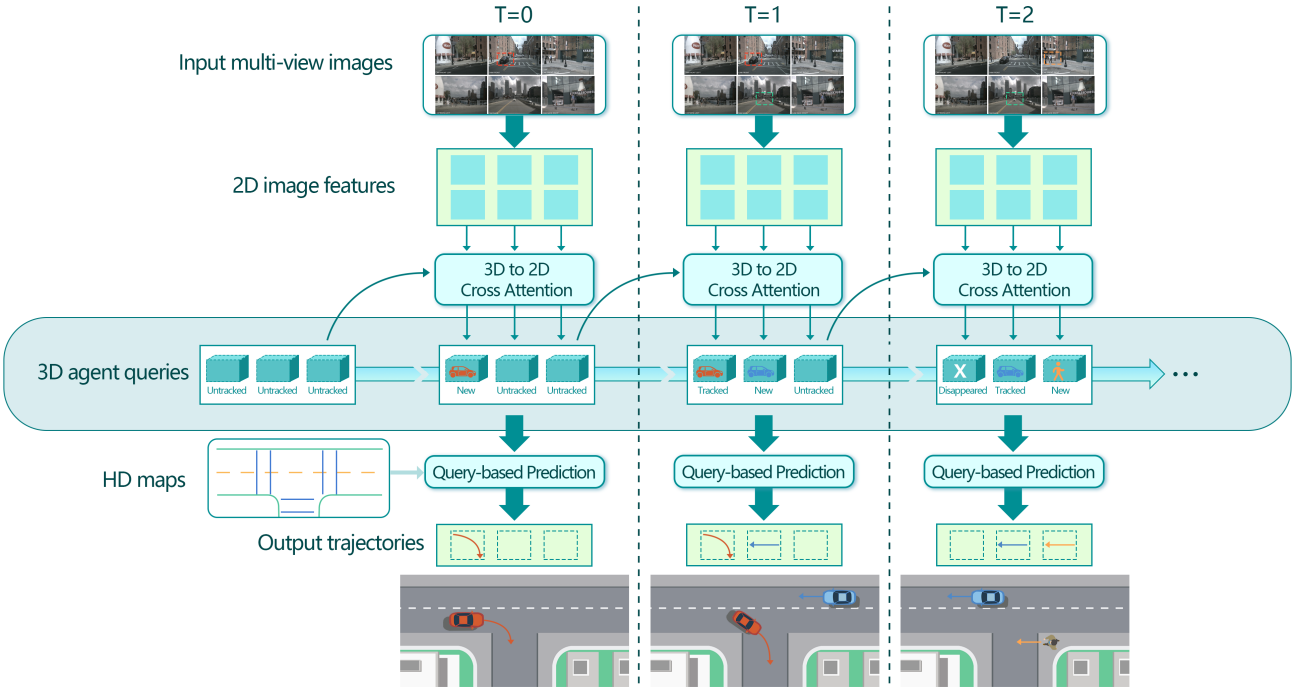

Figure 2. ViP3D model pipeline. 3D agent queries serve as the main thread and intermediate representations over time. At each time step, the agent queries aggregate visual features from multi-view images to obtain tracked agent queries. The tracked queries further interact with HD maps and are decoded into predicted trajectories. The agent queries are managed in a dynamic memory bank, and the model works in a concise streaming manner.

we perform a bipartite matching between the unmatched queries and the new appeared agents $\mathcal{A}_{t,\text{new}}$ at time step t: $\mathbf{Q}_{\text{empty}} \cong \mathcal{A}_{t,\text{new}}$.

To perform the bipartite matching, we utilize a query decoder that outputs the center coordinates of each query at time step t. The pair-wise matching cost [1] between ground truth y_i and a prediction $\hat{y}_{\sigma(i)}$ for the bipartite matching is: $\mathcal{L}_{\text{match}}\left(y_i, \hat{y}_{\sigma(i)}\right) = -\mathbb{1}_{\{c_i \neq \varnothing\}} \hat{p}_{\sigma(i)}\left(c_i\right) + \mathbb{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{\text{box}}\left(b_i, \hat{b}_{\sigma(i)}\right)$, where c_i is the target class label, $\mathcal{L}_{\text{box}}$ is the ℓ_1 loss for bounding box parameters, b_i is the target box, $\hat{b}_{\sigma(i)}$ and $\hat{p}_{\sigma(i)}\left(c_i\right)$ are the predicted box and predicted probability of class c_i, respectively.

After the bipartite matching, we get the optimal assignment $\hat{\sigma}$. We compute the query classification loss $\mathcal{L}_{\text{cls}}$ and query coordinate regression loss $\mathcal{L}_{\text{coord}}$ as follows:

$$\mathcal{L}_{\text{cls}} = \sum_{i=1}^{N} -\log \hat{p}_{\hat{\sigma}(i)}\left(c_i\right), \tag{1}$$

$$\mathcal{L}_{\text{coord}} = \sum_{i=1}^{N} \mathbb{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{\text{box}}\left(b_i, \hat{b}_{\hat{\sigma}}(i)\right), \tag{2}$$

where $\mathcal{L}_{\text{box}}$ is the ℓ_1 loss for bounding box parameters.

3.1.3 Query Memory Bank

To model long-term relationships for agent queries of different time steps, we maintain historical states for each agent query in a query memory bank. Following MOTR [59], the memory query bank is a first-in-first-out queue with a fixed size S_{bank}. After each time step, the attention mechanism is only applied between each query and its historical states in the memory bank for efficiency. For the i^{th} agent query q_t^i at the time step t, the corresponding historical states in the memory bank are denoted as $\mathbf{Q}_{\text{bank}}^i = \{q_{t-S_{\text{bank}}}^i, \ldots, q_{t-2}^i, q_{t-1}^i\}$. Then the temporal cross attention is $\tilde{q_t^i} = \text{softmax}\left(\frac{q_{t,\text{query}}^i \mathbf{Q}_{\text{bank,key}}^{i\top}}{\sqrt{d}}\right) \mathbf{Q}_{\text{bank,value}}^i$, where $q_{t,\text{query}}^i$, $\mathbf{Q}_{\text{bank,key}}^i$, $\mathbf{Q}_{\text{bank,value}}^i$ are query / key / value vectors after linear projection, respectively, and d is the dimension of the agent queries. The i^{th} agent query is updated by: $q_t^{i\prime} = \text{FFN}\left(q_t^i + \tilde{q_t^i}\right)$, where FFN is a two-layer MLP with layer normalization. Finally, the historical states of the i^{th} agent query in the memory bank become: $\mathbf{Q}_{\text{bank}}^{i\prime} = \{q_{t-S_{\text{bank}}+1}^i, \ldots, q_{t-1}^i, q_t^{i\prime}\}$.

3.2. Query-based Prediction

Typical trajectory prediction models can be divided into three components: an agent encoder that extracts agent trajectory features, a map encoder that extracts map features, and a trajectory decoder that outputs predicted trajectories. In our pipeline, the query-based detection and tracking gives tracked agent queries, which is equivalent to the output of the agent encoder. Therefore, by taking agent queries as input, the query-based prediction module is composed of only a map encoder and a trajectory decoder.

3.2.1 Map Encoding

HD semantic maps are crucial for trajectory prediction since they include detailed road information, such as lane types, road boundaries, and traffic signs. HD maps are typically represented by vectorized spatial coordinates of map elements and the topological relations between them. To encode this information, we adopt a popular vectorized encoding method VectorNet [12]. The map encoder produces a set of map features $\mathbf{M}$, which further interacts with agent queries via cross attention: $\mathbf{Q}' = \mathrm{Attention}(\mathbf{Q}, \mathbf{M})$.

3.2.2 Trajectory Decoding

The trajectory decoding takes the agent queries as input and outputs K possible future trajectories for each agent. ViP3D is compatible with a variety of trajectory decoding methods, such as regression-based methods [9, 17, 29, 44], goal-based methods [62] and heatmap-based methods [13, 14, 16]. We introduce the key ideas of these methods here and leave the details in the Appendix. (1) The regression-based method, namely variety loss (or min-of-K), predicts future trajectories based on regression. During inference, this decoder directly outputs a set of predicted trajectories. During training, we first calculate the distance between each predicted trajectory and the ground truth trajectory. Then we select a predicted trajectory with the closest distance and only calculate regression loss between it and the ground truth trajectory. (2) The goal-based method first defines sparse goal anchors heuristically and then classifies these anchors to estimate and select the goals. Finally, a trajectory is completed for each selected goal. (3) The heatmap-based method first generates a heatmap indicating the probability distribution of the goal. Then a greedy algorithm or a neural network is used to select goals from the heatmap. Finally, same as the goal-based method, the trajectories are completed. We use $\mathcal{L}_{\text{traj}}$ to denote the loss of trajectory decoding and leave the detailed definition in the Appendix.

3.3. Loss

ViP3D is trained end-to-end with query classification loss and query coordinate regression loss of the query-based detection and tracking, and trajectory decoding loss of the query-based prediction: $\mathcal{L} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{coord}} + \mathcal{L}_{\text{traj}}$.

4. Experiments

4.1. End-to-end Prediction Accuracy

To evaluate the performance of multi-future trajectory prediction, we adopt the common metrics including minimum average displacement error (minADE), minimum final displacement error (minFDE), and miss rate (MR). However, the inputs of end-to-end prediction are raw pixels, models may detect more false positive agents which should not exist (an example shown in Figure 3). In these metrics, we find the closest predicted trajectory for each ground truth trajectory to calculate displacement error, which does not account for false positives. Therefore, we propose a more comprehensive evaluation metric for end-to-end visual trajectory prediction, named End-to-end Prediction Accuracy (EPA).

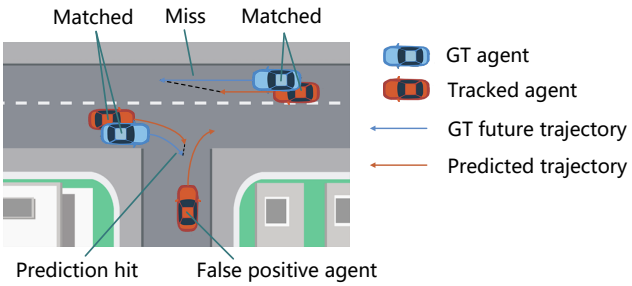

Figure 3. An example of End-to-end Prediction Accuracy (EPA) calculation. Blue and red agents are ground truth and detected agents, respectively. After matching the ground truth and the detection results, the red agent in the lower part is considered a false positive agent. A predicted trajectory is considered a hit when its final displacement error is below a certain threshold.

Let us denote predicted and ground truth agents as unordered sets $\hat{\mathcal{S}}$ and $\mathcal{S}$, respectively, where each agent is represented by K future trajectories of different modalities. First, for each agent type c, we calculate the prediction precision between $\hat{\mathcal{S}}_c$ and $\mathcal{S}_c$, where the subscript c indicates the agents of type c. We define the cost between a predicted agent $\hat{\mathbf{s}}$ and a ground truth agent $\mathbf{s}$ as:

$$C_{\text{EPA}}(\mathbf{s}, \hat{\mathbf{s}}) = \begin{cases} ||s_0 - \hat{s}_0||, & \text{if } ||s_0 - \hat{s}_0|| \leqslant \tau_{\text{EPA}} \\ \infty, & \text{if } ||s_0 - \hat{s}_0|| > \tau_{\text{EPA}} \end{cases}, \quad (3)$$

where $\hat{s}_0$ and s_0 indicate the coordinates of the ground truth agent and the predicted agent at the current time step, and we set the threshold of successful matching to $\tau_{\text{EPA}} = 2.0\text{m}$. We utilize bipartite matching according to C_{EPA} to find the correspondence between predicted agents and ground truth agents. Then the number of false-positive predicted agents is $N_{\text{FP}} = |\hat{\mathcal{S}}| - |\hat{\mathcal{S}}_{\text{match}}|$, where $\hat{\mathcal{S}}_{\text{match}} \subset \hat{\mathcal{S}}$ is the set of predicted agents which have been matched with ground truth agents. For each matched agent, we calculate minFDE (minimum final displacement error) between its predicted multiple future trajectories and the ground truth trajectory $\text{minFDE}(\hat{\mathbf{s}}, \mathbf{s}) = \min_{k \in 1 \ldots K} ||\hat{s}^{(k)}_{T_{\text{future}}} - s_{T_{\text{future}}}||$, where $\hat{\mathbf{s}}^{(k)}$ is the k^{th} trajectory of the matched agent $\hat{\mathbf{s}}$, and T_{future} is the final time step of the future trajectory. Now the set of agents which have matched and hit a ground truth agent is $\hat{\mathcal{S}}_{\text{match,hit}} = \{\hat{\mathbf{s}} : \hat{\mathbf{s}} \in \hat{\mathcal{S}}_{\text{match}}, \text{minFDE}(\hat{\mathbf{s}}, \mathbf{s}) \leqslant$

$\tau_{\text{EPA}}\}$. The EPA between $\hat{\mathcal{S}}_c$ and $\mathcal{S}_c$ is defined as:

$$\text{EPA}(\hat{\mathcal{S}}_c, \mathcal{S}_c) = \frac{|\hat{\mathcal{S}}_{\text{match,hit}}| - \alpha N_{\text{FP}}}{N_{\text{GT}}}, \tag{4}$$

where N_{GT} is the number of ground truth agents, and we set the penalty coefficient $\alpha = 0.5$ for all experiments. For different scenes, each number in the equation is defined as the sum over all scenes. Finally, the EPA between $\hat{\mathcal{S}}$ and $\mathcal{S}$ is averaged over all agent types.

4.2. Experimental Settings

Dataset. We train and evaluate ViP3D on the nuScenes dataset, a large-scale driving dataset including the urban scenarios in Boston and Singapore. It contains 1000 scenes, and each scene has a duration of around 20 seconds. The full dataset has more than one million images from 6 cameras and 1.4M bounding boxes for different types of objects. Bounding boxes of objects are annotated at 2Hz over the entire dataset.

Trajectory Prediction Settings. Popular trajectory prediction benchmarks, such as Argoverse Motion Prediction Benchmark [6], require the prediction of one target agent in each scene. In our visual trajectory prediction task, we simultaneously predict all agents in each scene, which is the same as real-time usage. A commonly used trick is to predict trajectories in allocentric view, *i.e.*, taking the last position of the target agent as the origin and its direction as y-axis. It makes prediction models focus on future modality prediction instead of coordinate transformation, thereby improving the prediction performance. In our experiments, we use this trick for all baselines and our ViP3D. Metrics averaged over vehicles and pedestrians are used to compare their performance on visual trajectory prediction task.

4.3. Baseline Settings

Traditional Perception and Prediction Pipeline. The traditional pipeline is composed of a vision-based detector, a tracker, and a predictor. For a fair comparison, the vision-based detector is the same as ViP3D. For the tracker, we test the performance of the classical IoU association with Kalman Filter, and an advanced tracking method named CenterPoint [57]. Compared with ViP3D, the outputs of the tracker are agent trajectories and agent attributes instead of agent queries. These agent attributes are manually-defined in common tracking tasks, and we use as many attributes as possible, including agent types, agent sizes, agent velocities, *etc*.

PnPNet-vision. PnPNet [30] only takes LiDAR data as input, and it cannot be directly used for our visual trajectory prediction task. Following the original PnPNet, we propose *PnPNet-vision* by replacing the LiDAR encoder of the original PnPNet with DETR3D, which is the same as the detector of ViP3D. Instead of using the query-based tracker and predictor, PnPNet associates boxes across frames according to affinity matrix and uses Kalman Filter as the motion model, which is a non-differentiable operation. For prediction, PnPNet crops features from the BEV feature map according to tracked trajectories, and takes the cropped features as the inputs of the prediction. We use Lift-Splat-Shot to obtain the BEV feature map for PnPNet-vision.

4.4. Evaluation and Analysis

4.4.1 Main Results

We compare our ViP3D with traditional perception and prediction pipeline and PnPNet-vision on the nuScenes dataset, as shown in Table 1. The traditional perception and prediction pipeline uses historical trajectories as the interface between tracking and prediction, so it cannot utilize visual information for prediction. Our proposed PnPNet-vision follows the key idea of the original PnPNet to obtain agent features by cropping from BEV feature maps, and takes the cropped features as the inputs of the predictor. More implementation details are described in Section 4.3. All baselines and our ViP3D use DETR3D as the detector and regression-based trajectory decoding method as the predictor for a fair comparison. We can see that ViP3D outperforms these baselines on all the metrics, indicating the effectiveness and superiority of directly learning from visual information with a fully differentiable approach.

4.4.2 Ablation Study

Trajectory Prediction Inputs. To better understand the necessity of visual features and end-to-end training, we compare ViP3D with different baselines. These baselines have the same architecture as ViP3D except for the prediction inputs. We use the default regression-based method for trajectory decoding. Results are shown in Table 2. It can be seen that *Agent trajectories* + *Agent queries* outperforms *Agent trajectories*, demonstrating that the agent queries provide more fine-grained and detailed visual information to improve prediction performance. ViP3D surpasses *Agent trajectories* and *Agent trajectories* + *Agent queries*, demonstrating that fully differentiable end-to-end learning is helpful in avoiding the error accumulation problem in the multi-stage pipeline.

Trajectory Decoding Methods. We compare our ViP3D with traditional perception and prediction pipeline under other trajectory decoding methods, goal-based TNT [62] and heatmap-based HOME [14], which recently achieve state-of-the-art performance. As shown in Table 3, ViP3D surpasses the traditional perception and prediction pipeline

		Traditional		PnPNet-vision [30]		ViP3D (Ours)
Architechture	detector	DETR3D		DETR3D		DETR3D
	detector-tracker interface	boxes		boxes		queries
	tracker	Kalman Filter	CenterPoint	Kalman Filter	CenterPoint	query-based
	tracker-predictor interface	trajectories		cropped features		queries
	predictor	regression-based		regression-based		regression-based
Metrics	minADE↓	2.07	2.06	2.04	2.04	**2.03**
	minFDE↓	3.10	3.02	3.08	3.03	**2.90**
	MR↓	0.289	0.277	0.277	0.271	**0.239**
	EPA↑	0.191	0.209	0.198	0.213	**0.236**

Table 1. Comparing ViP3D with traditional multi-stage pipeline. Classical metrics include minADE, minFDE and Miss Rate (MR), and End-to-end Prediction Accuracy (EPA) which is our proposed metric for the end-to-end setting. For each agent, 6 future trajectories with a time horizon of 6 seconds are evaluated.

	Prediction inputs	Differentiable	minADE ↓	minFDE ↓	MR ↓	EPA↑
	Agent trajectories	✗	2.30	3.33	0.282	0.186
	Agent trajectories + Agent queries	✗	2.20	3.19	0.274	0.211
ViP3D	Agent queries	✓	**2.03**	**2.90**	**0.239**	**0.236**

Table 2. Ablation study on the inputs of the trajectory prediction module of ViP3D. Trajectory decoding defaults to a regression-based method.

on these metrics under the two trajectory decoding methods, demonstrating that ViP3D is compatible with various state-of-the-art trajectory decoders and achieves superior performance.

Decoder	Pipeline	mADE	mFDE	MR	EPA
Goal [62]	Traditional	2.50	3.93	0.266	0.195
	ViP3D	**2.24**	**3.33**	**0.238**	**0.219**
Heatmap [14]	Traditional	2.53	3.81	0.264	0.197
	ViP3D	**2.33**	**3.42**	**0.218**	**0.214**

Table 3. Comparing trajectory prediction performance on the nuScenes validation set with another two trajectory decoding methods: goal-based and heatmap-based. mADE and mFDE denote minADE and minFDE, respectively.

View of Trajectory Prediction. We test the performance of the pipelines in two different prediction coordinates. One is in the egocentric view, and the other is in the allocentric view [23]. The egocentric view indicates predicting trajectories in the coordinate system of the ego vehicle, while the allocentric view indicates predicting trajectories in the coordinate system of the predicted agent itself. Predicting trajectories in the allocentric view is a commonly used normalization trick, and it has a better performance compared with the egocentric view. As shown in Table 4, the same results are obtained in our experiments. So experiments of baselines and ViP3D in other sections are performed in the allocentric view by default.

View	Pipeline	minADE	minFDE	MR	EPA
Egocentric	Traditional	2.51	3.57	0.353	0.132
	ViP3D	2.10	3.01	0.261	0.199
Allocentric	Traditional	2.06	3.02	0.277	0.209
	ViP3D	**2.03**	**2.90**	**0.239**	**0.236**

Table 4. The comparison between different types of view of trajectory prediction.

Analysis of Different Detectors We also conduct experiments on other vision-based detectors, such as PETRv2 [33], which leverages the temporal information of previous frames to assist 3D object detection. When using PETRv2 as the detection backbone, ViP3D achieves a better performance in short-term inference (< 3s) but fails in long-term inference (> 10s). It indicates that the performance of long-term inference is sensitive to the detection backbone, and more efforts are needed to adapt ViP3D to different detectors. A possible solution is to run ViP3D on longer scene segments (currently 3 frames) during training if the GPU memory is large enough. We regard it as a limitation of ViP3D.

4.4.3 Qualitative Results

We provide examples of the predicted results by ViP3D and traditional pipeline in Figure 4. In the upper example, we can see that the left turn signal of the vehicle in the blue box is flashing, indicating that the vehicle is about to turn left. ViP3D can use this visual information to predict the correct trajectory. In contrast, the traditional pipeline can only use

historical trajectory information to predict that the vehicle is about to go straight incorrectly. In the lower example, we can see that the pedestrian is facing the coming vehicle, indicating that he has probably noticed the approaching vehicle and will stop and wait for the vehicle to go first. ViP3D makes use of the pedestrian's head pose to correctly predict that the pedestrian will stop, while the traditional pipeline incorrectly predicts that pedestrians will cross the road. These two examples show that ViP3D improves trajectory prediction performance due to utilizing visual information.

5. Conclusion

We present ViP3D, a fully differentiable approach to predict future trajectories of agents from multi-view videos. It exploits the rich visual information from the raw sensory input and avoids the error accumulation problem in the traditional pipeline. Moreover, by leveraging 3D agent queries, ViP3D models agent instances explicitly, making the pipeline interpretable and debuggable.

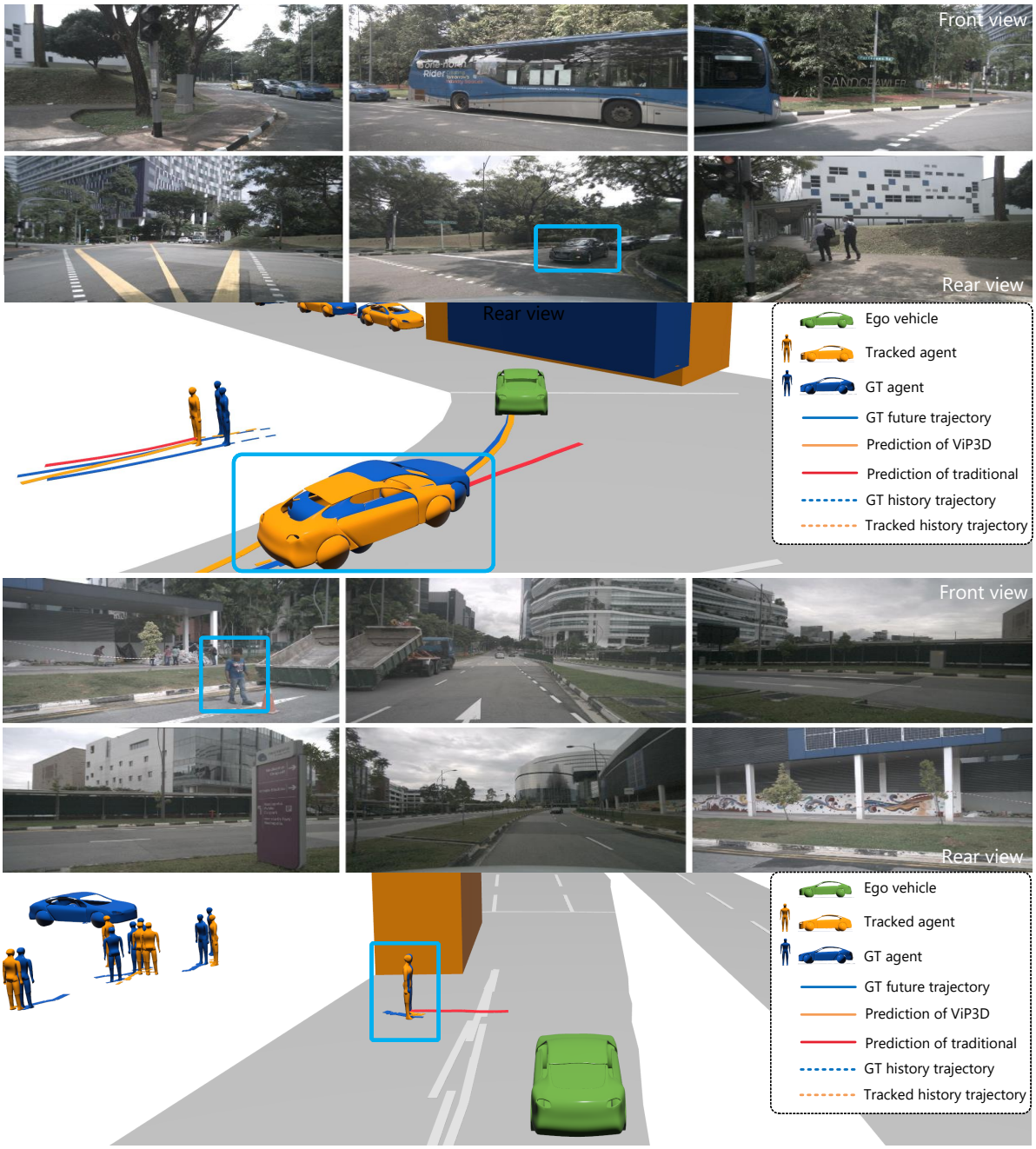

Figure 4. Qualitative results. Input camera images are shown on the top. The green vehicle is the ego agent. The blue and orange agents indicate ground-truth and tracked agents, respectively. The blue, orange and red curves indicate ground-truth trajectories, prediction of ViP3D and prediction of the traditional pipeline, respectively. For each agent, only the predicted trajectory with the highest probability is drawn.

References

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 4

[2] S. Casas, Cole Gulino, Simon Suo, Katie Luo, Renjie Liao, and R. Urtasun. Implicit latent variable model for scene-consistent motion forecasting. In *ECCV*, 2020. 3

[3] Sergio Casas, Wenjie Luo, and Raquel Urtasun. Intentnet: Learning to predict intention from raw sensor data. In *Conference on Robot Learning*, pages 947–956. PMLR, 2018. 2, 3

[4] Mohamed Chaabane, Peter Zhang, J Ross Beveridge, and Stephen O'Hara. Deft: Detection embeddings for tracking. *arXiv preprint arXiv:2102.02267*, 2021. 2

[5] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. *arXiv preprint arXiv:1910.05449*, 2019. 2

[6] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8748–8757, 2019. 6

[7] Dooseop Choi and KyoungWook Min. Hierarchical latent structure for multi-modal vehicle trajectory forecasting. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXII*, pages 129–145. Springer, 2022. 3

[8] Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2090–2096. IEEE, 2019. 2

[9] Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2090–2096. IEEE, 2019. 5

[10] Nachiket Deo and Mohan M Trivedi. Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1179–1184. IEEE, 2018. 2

[11] Liangji Fang, Qinhong Jiang, Jianping Shi, and Bolei Zhou. Tpnet: Trajectory proposal network for motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6797–6806, 2020. 2

[12] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11525–11533, 2020. 2, 5

[13] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. Gohome: Graph-oriented heatmap output for future motion estimation. *arXiv preprint arXiv:2109.01827*, 2021. 5

[14] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. Home: Heatmap output for future motion estimation. *arXiv preprint arXiv:2105.10968*, 2021. 3, 5, 6, 7

[15] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. Thomas: Trajectory heatmap output with learned multi-agent sampling. In *International Conference on Learning Representations*, 2021. 3

[16] Junru Gu, Chen Sun, and Hang Zhao. Densetnt: End-to-end trajectory prediction from dense goal sets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15303–15312, 2021. 3, 5

[17] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2255–2264, 2018. 5

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, pages 770–778, 2016. 3

[19] Joey Hong, Benjamin Sapp, and James Philbin. Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8454–8462, 2019. 3

[20] Anthony Hu, Zak Murez, Nikhil Mohan, Sofía Dudas, Jeffrey Hawke, Vijay Badrinarayanan, Roberto Cipolla, and Alex Kendall. Fiery: Future instance prediction in bird's-eye view from surround monocular cameras. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15273–15282, 2021. 3

[21] Hou-Ning Hu, Yung-Hsu Yang, Tobias Fischer, Trevor Darrell, Fisher Yu, and Min Sun. Monocular quasi-dense 3d object tracking. *arXiv preprint arXiv:2103.07351*, 2021. 2

[22] Junjie Huang, Guan Huang, Zheng Zhu, and Dalong Du. Bevdet: High-performance multi-camera 3d object detection in bird-eye-view. *arXiv preprint arXiv:2112.11790*, 2021. 2

[23] Xiaosong Jia, Liting Sun, Hang Zhao, Masayoshi Tomizuka, and Wei Zhan. Multi-agent trajectory prediction by combining egocentric and allocentric views. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 1434–1443. PMLR, 08–11 Nov 2022. 7

[24] Yanqin Jiang, Li Zhang, Zhenwei Miao, Xiatian Zhu, Jin Gao, Weiming Hu, and Yu-Gang Jiang. Polarformer: Multi-camera 3d object detection with polar transformers. *arXiv preprint arXiv:2206.15398*, 2022. 2

[25] ByeoungDo Kim, Seong Hyeon Park, Seokhwan Lee, Elbek Khoshimjonov, Dongsuk Kum, Junsoo Kim, Jeong Soo Kim, and Jun Won Choi. LaPred: Lane-aware prediction of multi-modal future trajectories of dynamic agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14636–14645, 2021. 3

[26] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. PointPillars: Fast Encoders

for Object Detection from Point Clouds. In *CVPR*, pages 12697–12705, 2019. 2

[27] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 336–345, 2017. 3

[28] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Qiao Yu, and Jifeng Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. *arXiv preprint arXiv:2203.17270*, 2022. 2

[29] Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning lane graph representations for motion forecasting. In *European Conference on Computer Vision*, pages 541–556. Springer, 2020. 2, 3, 5

[30] Ming Liang, Bin Yang, Wenyuan Zeng, Yun Chen, Rui Hu, Sergio Casas, and Raquel Urtasun. Pnpnet: End-to-end perception and prediction with tracking in the loop. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11553–11562, 2020. 2, 3, 6, 7

[31] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection. In *CVPR*, pages 2117–2125, 2017. 3

[32] Yingfei Liu, Tiancai Wang, Xiangyu Zhang, and Jian Sun. Petr: Position embedding transformation for multi-view 3d object detection. *arXiv preprint arXiv:2203.05625*, 2022. 2

[33] Yingfei Liu, Junjie Yan, Fan Jia, Shuailin Li, Qi Gao, Tiancai Wang, Xiangyu Zhang, and Jian Sun. Petrv2: A unified framework for 3d perception from multi-camera images. *arXiv preprint arXiv:2206.01256*, 2022. 7

[34] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 3569–3577, 2018. 2, 3

[35] Sriram Narayanan, Ramin Moslemi, Francesco Pittaluga, Buyu Liu, and Manmohan Chandraker. Divide-and-conquer for lane-aware diverse trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15799–15808, 2021. 3

[36] Nigamaa Nayakanti, Rami Al-Rfou, Aurick Zhou, Kratarth Goel, Khaled S Refaat, and Benjamin Sapp. Wayformer: Motion forecasting via simple & efficient attention networks. *arXiv preprint arXiv:2207.05844*, 2022. 2

[37] Jiquan Ngiam, Benjamin Caine, Vijay Vasudevan, Zhengdong Zhang, Hao-Tien Lewis Chiang, Jeffrey Ling, Rebecca Roelofs, Alex Bewley, Chenxi Liu, Ashish Venugopal, et al. Scene transformer: A unified multi-task model for behavior prediction and planning. *arXiv preprint arXiv:2106.08417*, 2021. 2

[38] Ziqi Pang, Zhichao Li, and Naiyan Wang. Simpletrack: Understanding and rethinking 3d multi-object tracking. *arXiv preprint arXiv:2111.09621*, 2021. 2

[39] Tung Phan-Minh, Elena Corina Grigore, Freddy A Boulton, Oscar Beijbom, and Eric M Wolff. Covernet: Multimodal behavior prediction using trajectory sets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14074–14083, 2020. 2

[40] Jonah Philion and Sanja Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *European Conference on Computer Vision*, pages 194–210. Springer, 2020. 2

[41] John Phillips, Julieta Martinez, Ioan Andrei Barsan, Sergio Casas, Abbas Sadat, and Raquel Urtasun. Deep multi-task learning for joint localization, perception, and prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4679–4689, June 2021. 3

[42] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *CVPR*, pages 652–660, 2017. 2

[43] Nicholas Rhinehart, Kris M Kitani, and Paul Vernaza. R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 772–788, 2018. 3

[44] Christian Rupprecht, Iro Laina, Robert DiPietro, Maximilian Baust, Federico Tombari, Nassir Navab, and Gregory D Hager. Learning in an uncertain world: Representing ambiguity through multiple hypotheses. In *Proceedings of the IEEE international conference on computer vision*, pages 3591–3600, 2017. 5

[45] Samuel Scheidegger, Joachim Benjaminsson, Emil Rosenberg, Amrit Krishnan, and Karl Granström. Mono-camera 3d multi-object tracking using deep learning detections and pmbm filtering. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 433–440. IEEE, 2018. 2

[46] Andrea Simonelli, Samuel Rota Bulo, Lorenzo Porzi, Manuel López-Antequera, and Peter Kontschieder. Disentangling monocular 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1991–1999, 2019. 2

[47] Haoran Song, Di Luan, Wenchao Ding, Michael Yu Wang, and Qifeng Chen. Learning to predict vehicle trajectories with model-based planning. *arXiv preprint arXiv:2103.04027*, 2021. 3

[48] Chen Sun, Per Karlsson, Jiajun Wu, Joshua B Tenenbaum, and Kevin Murphy. Stochastic prediction of multi-agent interactions from partial observations. *arXiv preprint arXiv:1902.09641*, 2019. 3

[49] Yichuan Charlie Tang and Ruslan Salakhutdinov. Multiple futures prediction. *arXiv preprint arXiv:1911.00997*, 2019. 3

[50] Hung Tran, Vuong Le, and Truyen Tran. Goal-driven long-term trajectory prediction. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 796–805, 2021. 3

[51] Balakrishnan Varadarajan, Ahmed Hefny, Avikalp Srivastava, Khaled S Refaat, Nigamaa Nayakanti, Andre Cornman, Kan Chen, Bertrand Douillard, Chi Pang Lam, Dragomir Anguelov, et al. Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction. In

2022 International Conference on Robotics and Automation (ICRA), pages 7814–7821. IEEE, 2022. 2

[52] Tai Wang, Xinge Zhu, Jiangmiao Pang, and Dahua Lin. Fcos3d: Fully convolutional one-stage monocular 3d object detection. *arXiv preprint arXiv:2104.10956*, 2021. 2

[53] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-LiDAR from Visual Depth Estimation: Bridging the Gap in 3D Object Detection for Autonomous Driving. In *CVPR*, pages 8445–8453, 2019. 2

[54] Yue Wang, Vitor Campagnolo Guizilini, Tianyuan Zhang, Yilun Wang, Hang Zhao, and Justin Solomon. Detr3d: 3d object detection from multi-view images via 3d-to-2d queries. In *5th Annual Conference on Robot Learning*, 2021. 2, 3

[55] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. 3D Multi-Object Tracking: A Baseline and New Evaluation Metrics. *IROS*, 2020. 2

[56] Raymond A Yeh, Alexander G Schwing, Jonathan Huang, and Kevin Murphy. Diverse generation for multi-agent sports games. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4610–4619, 2019. 3

[57] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3D Object Detection and Tracking. *arXiv preprint arXiv:2006.11275*, 2020. 2, 6

[58] Ye Yuan and Kris M Kitani. Diverse trajectory forecasting with determinantal point processes. In *International Conference on Learning Representations*, 2019. 3

[59] Fangao Zeng, Bin Dong, Tiancai Wang, Xiangyu Zhang, and Yichen Wei. Motr: End-to-end multiple-object tracking with transformer. *arXiv preprint arXiv:2105.03247*, 2021. 3, 4

[60] Pu Zhang, Lei Bai, Jianru Xue, Jianwu Fang, Nanning Zheng, and Wanli Ouyang. Trajectory forecasting from detection with uncertainty-aware motion encoding. *arXiv preprint arXiv:2202.01478*, 2022. 2

[61] Tianyuan Zhang, Xuanyao Chen, Yue Wang, Yilun Wang, and Hang Zhao. Mutr3d: A multi-camera tracking framework via 3d-to-2d queries. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4537–4546, 2022. 2

[62] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Benjamin Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai, Cordelia Schmid, et al. Tnt: Target-driven trajectory prediction. *arXiv preprint arXiv:2008.08294*, 2020. 3, 5, 6, 7

[63] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. In *European Conference on Computer Vision*, pages 474–490. Springer, 2020. 2

[64] Yin Zhou and Oncel Tuzel. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In *CVPR*, pages 4490–4499, 2018. 2