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

Supplementary Material

A. Implementation Details

Training and Inference Details. In our experiments, all models are trained on the nuScenes training set with a batch size of 8 for 24 epochs. The ADAM optimizer [2] is adopted to train the whole pipeline. The learning rate has an initial value of $2e^{-4}$ and decays to 10% at the 20th and the 23rd epochs. The hidden size of the query-based detection and tracking module is set to 256, and that of the trajectory predictor is set to 128. A pretrained detection backbone is used for model initialization. We evaluate all models on the nuScenes validation set. All models are tested online by feeding raw multi-view images of each time step to the model in chronological order. The metric computation is performed at every step except for steps that do not have enough future frames. Different from popular trajectory prediction benchmarks that only require predictions of selected agents, we simultaneously predict all agents at each step.

Query-based Detection and Tracking. The query-based detection and tracking takes ResNet50 [8] as the image backbone and DETR3D [11] as the detection head. The detection head consists of 6 layers, and each layer contains a feature refinement layer and a multi-head attention layer with layer normalization. The hidden size for the detection head is set to 256. Finally, one branch predicts center coordinates and size of agents, and the other branch predicts agent type. Each branch consists of two fully connected layers, where the hidden size is also 256.

Map Encoding. Same as typical trajectory prediction models, ViP3D also encodes HD maps to facilitate trajectory prediction. VectorNet [3] is the first trajectory prediction method to encode vectorized HD maps using a hierarchical graph neural network, and we follow it to convert each lane into a sequence of vectors. Each vector represents a segment of the lane, including the endpoints of the segment, the attributes of the lane, and the numerical order of the segment in the lane.

B. Trajectory Decoding

ViP3D can leverage a variety of trajectory decoding methods, such as regression-based methods [1, 7, 9, 10], goal-based methods [12] and heatmap-based methods [4–6]. We conduct experiments on these three trajectory decoding methods. In this section, we introduce the implementation details of these methods.

Regression-based. The regression-based trajectory decoder is a 2-layer MLP that takes the agent queries as input and directly outputs multiple future trajectories. During inference, the regression-based trajectory decoder directly outputs a set of predicted trajectories. During training, we first calculate the distance between each predicted trajectory $\hat{\mathbf{s}}$ and ground truth trajectory $\mathbf{s}$: $d(\mathbf{s}, \hat{\mathbf{s}}) = \sum_{t=1}^{T_{\text{future}}} ||s_t - \hat{s}_t||$, where $||\cdot||$ is the ℓ_2 distance between two points. Then, we select the predicted trajectory with the closest distance: $\hat{k} = \text{argmin}_{k \in 1 \dots K}\, d(\mathbf{s}, \mathbf{s}^{(k)})$, where $\mathbf{s}^{(k)}$ is the k^{th} predicted trajectory. Finally, we calculate regression loss between the closest predicted trajectory $\mathbf{s}^{(\hat{k})}$ and the ground truth trajectory $\mathbf{s}$ as

$$\mathcal{L}_{\text{trajectory}} = \sum_{t=1}^{T_{\text{future}}} \mathcal{L}_{\text{reg}}(s_t, s_t^{(\hat{k})}), \tag{1}$$

where $\mathcal{L}_{\text{reg}}$ is the smooth ℓ_1 loss between two points.

Goal-based. The goal-based trajectory decoder consists of a goal encoder, a probability decoder, an offset decoder, and a trajectory completion module. These modules are implemented using MLP. For each agent, we first randomly generate a set of candidate goals. The goal encoder is used to obtain the features of candidate goals by taking their coordinates as input. After that, a concatenation of the agent query and the features of goal coordinates is fed into the probability decoder and offset decoder. The probability decoder and the offset decoder output predicted goal probabilities and goal offsets, respectively. Let $\mathcal{L}_{\text{cls}}$ be the binary cross-entropy loss for the probability decoder, and let

$\mathcal{L}_{\text{reg}}$ be the smooth ℓ_1 loss for the offset decoder. To obtain K trajectories, Non-maximum supervision (NMS) is employed to select K goals (after adding the goal offsets), and the trajectory completion module takes the K selected goals and outputs K trajectories. Let $\mathcal{L}_{\text{completion}}$ be the smooth ℓ_1 loss for the trajectory completion module. Then the overall loss is

$$\mathcal{L}_{\text{trajectory}} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{reg}} + \mathcal{L}_{\text{completion}}. \tag{2}$$

Heatmap-based. The heatmap-based trajectory decoder only consists of a goal encoder, a probability decoder, and a trajectory completion module. These modules are implemented using MLP. For each agent, to obtain a heatmap indicating the probability distribution of the final positions of the trajectories, we first densely sample goals with a sampling density of 1m. The goal encoder is used to obtain the features of the goals by taking their coordinates as input. After that, a concatenation of the agent query and the features of goal coordinates is fed into the probability decoder. The probability decoder outputs predicted goal probabilities, and we obtain the heatmap. Let $\mathcal{L}_{\text{cls}}$ be the binary cross-entropy loss for the probability decoder. To obtain K trajectories, we also use NMS to select K goals for simplification, instead of using greedy algorithms as in origin heatmap-based methods [5]. The trajectory completion module takes the K selected goals and outputs K trajectories. Let $\mathcal{L}_{\text{completion}}$ be the smooth ℓ_1 loss for the trajectory completion module. Then the overall loss is

$$\mathcal{L}_{\text{trajectory}} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{completion}}. \tag{3}$$

C. Qualitative Results

The visualizations of predicted results of both ViP3D and the traditional pipeline are included in the paper. In this section, we provide more visualizations for ViP3D, including some failure cases. As the cases shown in Figure 1, ViP3D can predict accurate future trajectories. As the failure cases shown in Figure 2, because ViP3D is a vision-based pipeline, it is difficult for ViP3D to detect agents far away from the ego vehicle or agents partially obscured. In the upper part of Figure 2, a vehicle (surrounded by a red box) that is far away from the ego vehicle and is partially obscured by other vehicles, so it is difficult to be detected. In the lower part of Figure 2, a pedestrian (surrounded by a red box) is mostly obscured by a billboard, so ViP3D can not detect this pedestrian.

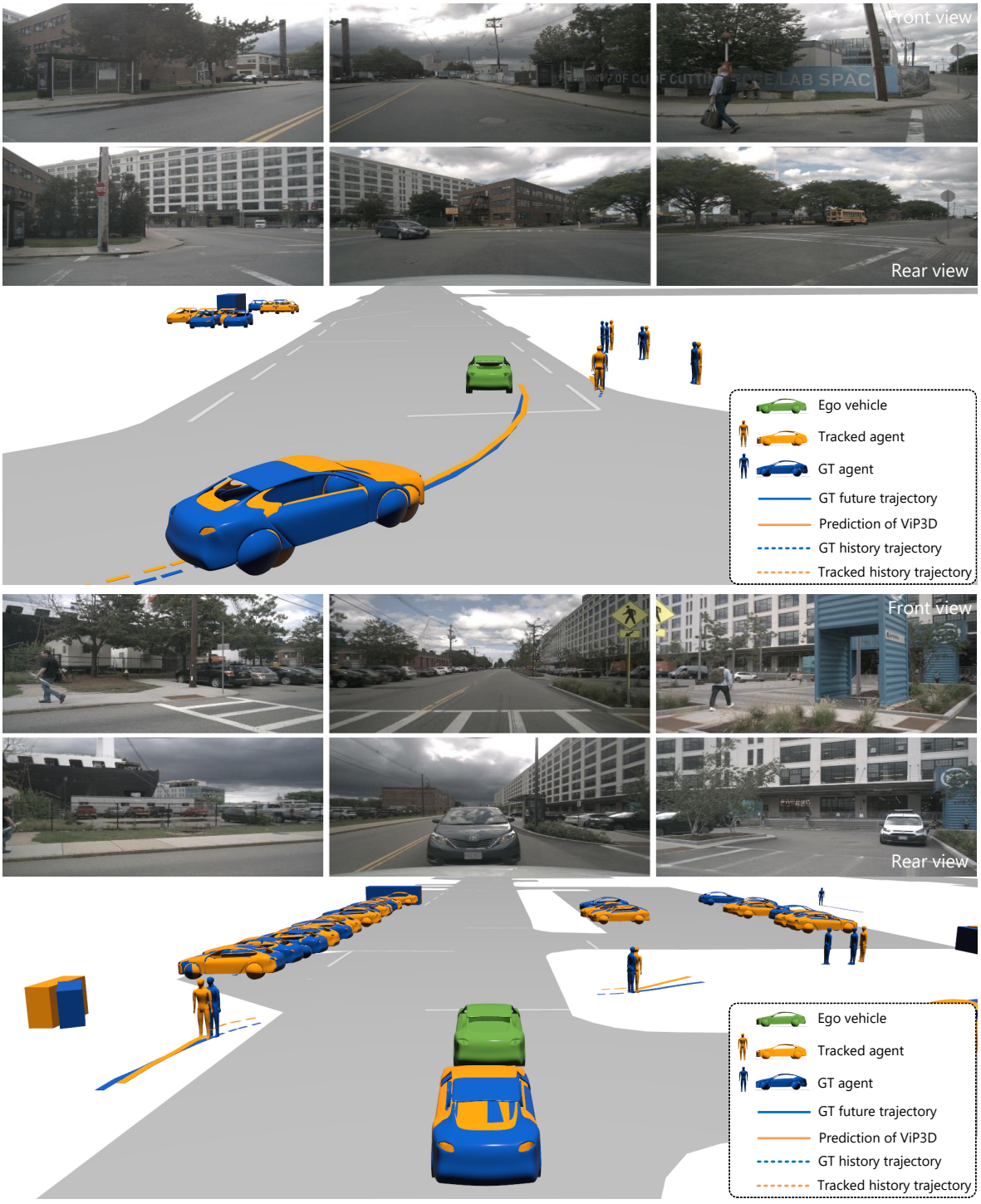

Figure 1. Qualitative results of ViP3D on the nuScenes validation set. Input camera images are shown on the top. The green vehicle is the ego agent. The blue and orange agents indicate ground-truth and tracked agents, respectively. The blue and orange curves indicate ground-truth trajectories and predicted trajectories of ViP3D, respectively. For each agent, only the predicted trajectory with the highest probability is drawn.

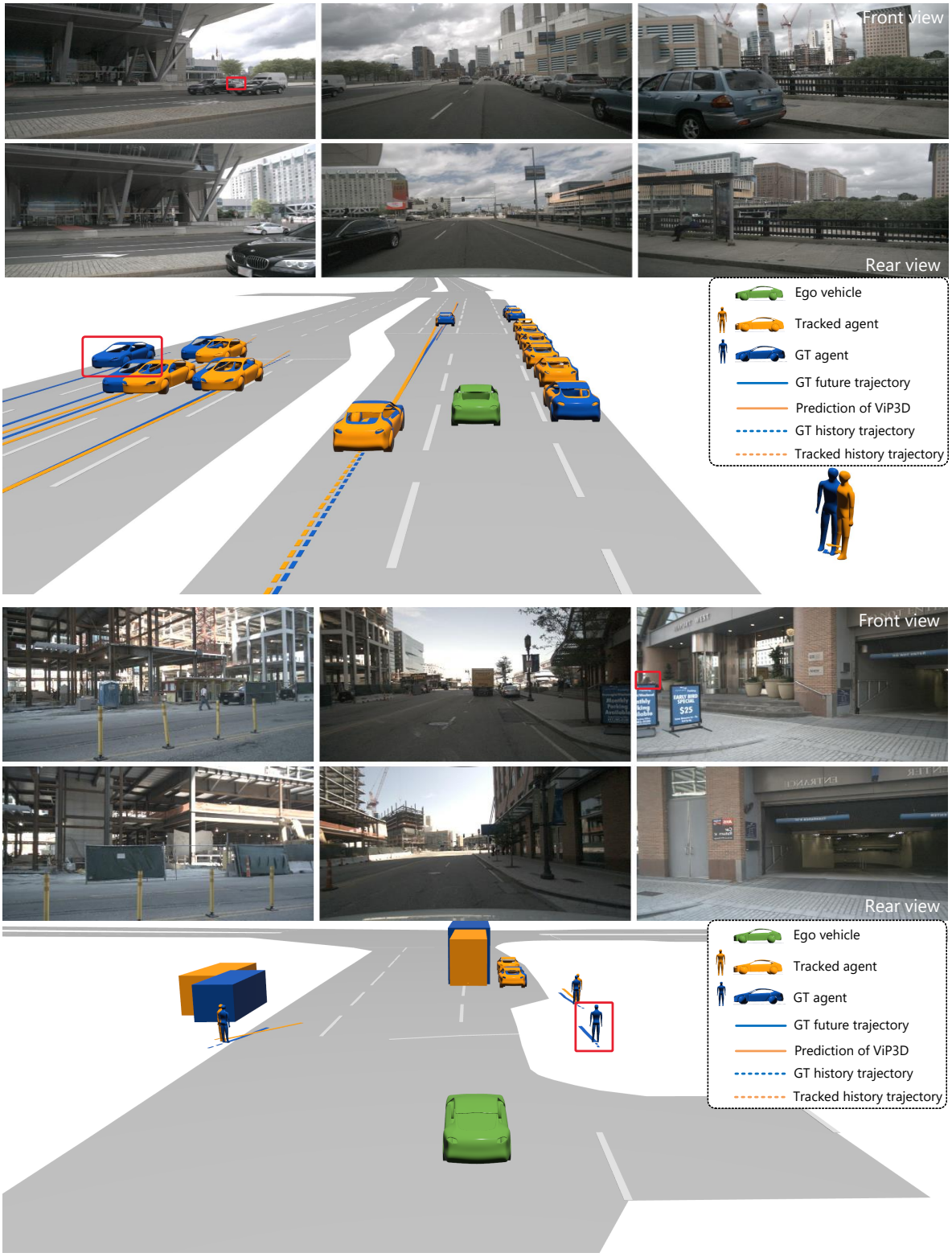

Figure 2. Failure cases of ViP3D on the nuScenes validation set. Input camera images are shown on the top. The green vehicle is the ego agent. The blue and orange agents indicate ground-truth and tracked agents, respectively. The blue and orange curves indicate ground-truth trajectories and predicted trajectories of ViP3D, respectively. For each agent, only the predicted trajectory with the highest probability is drawn. The agent surrounded by a red box indicates that it is not detected by ViP3D.

References

[1] 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. 1

[2] Kingma Da. A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1

[3] 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. 1

[4] 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. 1

[5] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. Home: Heatmap output for future motion estimation. *arXiv preprint arXiv:2105.10968*, 2021. 1, 2

[6] 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. 1

[7] 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. 1

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, pages 770–778, 2016. 1

[9] 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. 1

[10] 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. 1

[11] 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. 1

[12] 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. 1