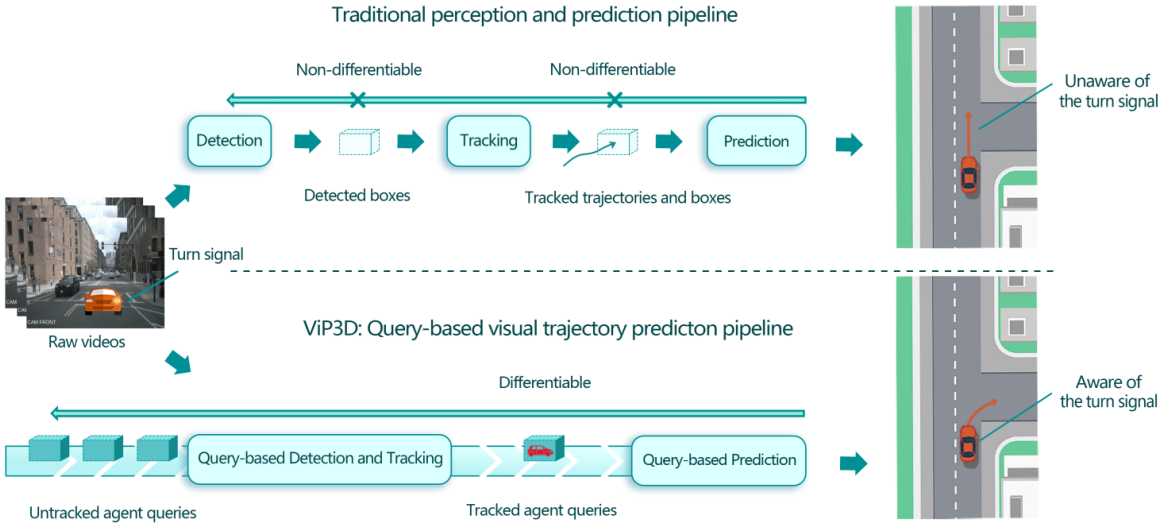
ViP3D: End-to-end Visual Trajectory Prediction via 3D Agent Queries

动机

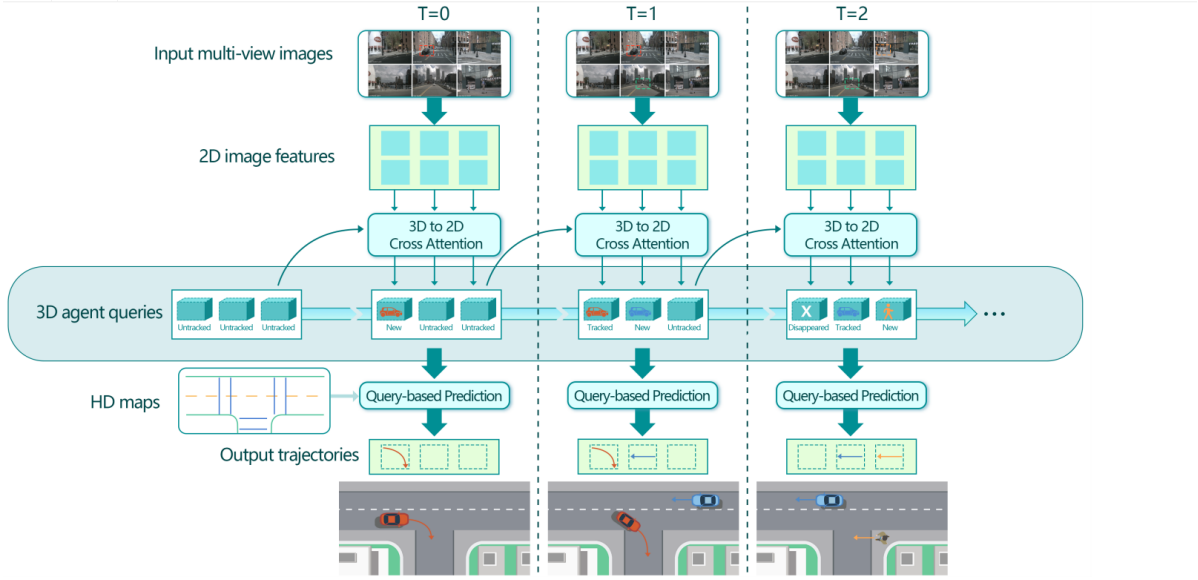
1. 传统的perception与prediction是分开建模的，导致：
    1. 两个模块只能通过接口通信，通信的信息十分有限，一般是geometric和semantic特征，比如historical agent trajectories, agent types and agent sizes. 因此许多明显的、预示着主题接下来的行动的信息没有被显式地建模，比如刹车灯亮、车头的偏移等。
    2. prediction模块作为perception模块的downstream，要承担perception模块产生的错误带来的影响，并且这种错误难以消解、会逐渐积累。
2. LiDAR-based trajectory prediction存在两大问题：
    1. 无法完全利用camera提供的细粒度信息
    2. 模型使用feature map作为帧的中间表示，因此会在non-differentiable operations上受到阻碍，例如object decoding中的non-maximum suppression.



如图所示，传统的方法直接忽略了转向灯的影响。而ViP3D模型的prediction模块可以通过query捕捉到转向灯的改变传递的信息。

ViP3D模型

整体架构



对于每个个时间点，流程如下：

1. 获取输入，输入为多张不同角度的图片。
2. 通过DETR3D模型来从multi-view images 获得2D image features.
3. 通过Temporal Feature Aggregation对agent queries进行管理
4. 通过Query-based Prediction输出预测结果

Temporal Feature Aggregation

其实就是如何在时间线上管理queries

对于每个query，它要么就和一个agent相关联，要么就为空。

Query Feature Update

可以利用cross-attention来进行更新：

Q是原来的queries集合，K_L和V_L分别是features序列的key和value.

那么更新后的$Q_{new} = FFN(softmax(\frac{QK_L^T}{\sqrt{d_k}})V_L + Q)$

Query Feature Supervision

描述了每个agent query在时间线上的变动，即如何利用t-1时刻的query来更新t时刻的query.

就如同我们之前所说，query有两类：

1. 已经跟一个agent相关联。
2. 空。

那么更新的策略就有两种：

1. 若当前的query在t-1时刻已经matched：
    1. 若agent还在，那么$q_t = q_{t-1}$
    2. 否则$q_t = EMPTY$
2. 若当前query在t-1时刻没有matched：

    直接将所有新出现的agent和该query进行匹配。

Query-based Prediction

Outline：

1. 输入为agent queries.
2. 包含Map encoder来抓取map features
3. 包含trajectory decoder来输出预测的轨迹

Map Encoding

利用VectorNet进行encode，得到结果Map features，记为M

则之后agent query和map进行交互时则依赖于$Attention(Q, M)$

Trajectory Decoding

Outline:

1. 输入为agent queries.
2. 输出为对于每个agent的K条可能的轨迹。

该模型兼容了多种trajectory decoding方法，如regression-based method, goal-based method, heatmap-based method.

本文并没有详述具体的方法。

Loss

loss为前面agent query supervision的loss和轨迹预测的loss之和

$L = L_{cls} + L_{coord} + L_{traj}$

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

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

对于L_{traj}，作者说在Appendix里，但我没找到附录。

Conclusion

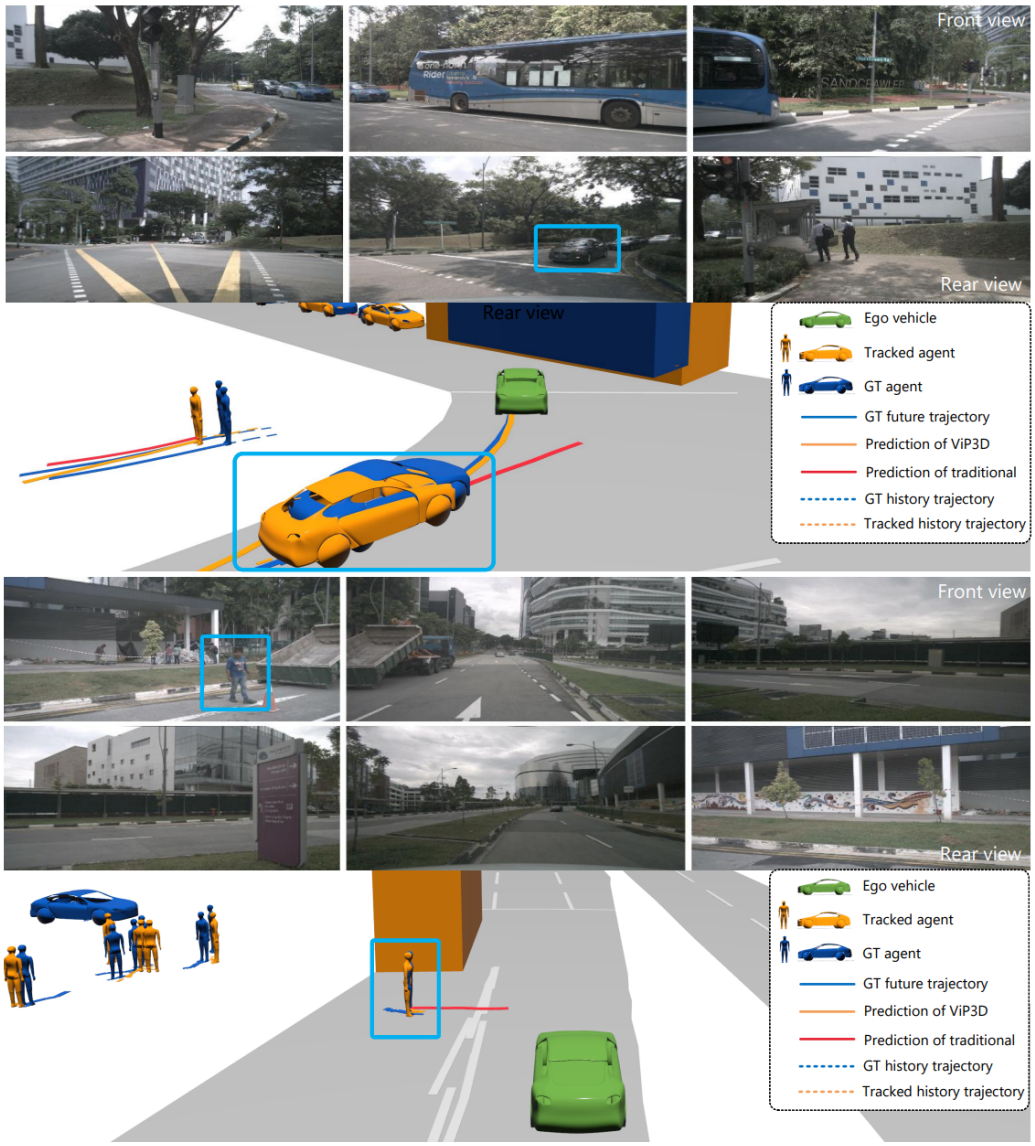
		Traditional		PnPNet-vision [30]		ViP3D (Ours)
Architechture	detector	DETR3D		DETR3D		DETR3D
	detector-tracker interface	boxes		boxes		queries
	tracker	Kalman Filter	CenterPoint	Kalman Filter	CenterPoint	query-based
	tracker-predictor interface	trajectories		cropped features		queries
	predictor	regression-based		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**

	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**

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**

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**

Effect


Front view
Rear view
Rear view
Ego vehicle
Tracked agent
GT agent
GT future trajectory
Prediction of ViP3D
Prediction of traditional
GT history trajectory
Tracked history trajectory
Front view
Rear view
Ego vehicle
Tracked agent
GT agent
GT future trajectory
Prediction of ViP3D
Prediction of traditional
GT history trajectory
Tracked history trajectory