Supplementary Materials for SIMPL: A Simple and Efficient Multi-agent Motion Prediction Baseline for Autonomous Driving

Lu Zhang[1], Peiliang Li[2], Sikang Liu[2], and Shaojie Shen[1]

APPENDIX

A. Discussion on Efficiency

SIMPL directly builds the global all-to-all relationship and uses the standard multi-head attention for feature fusion. We point out that the temporal information of each instance, such as historical motion, is encoded during the feature embedding (tokenization), therefore, the time complexity of the fusion process is only related to the number of total instances in the scene (please see Fig. 1).

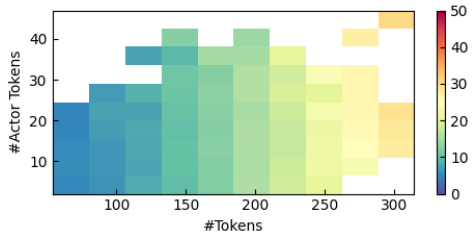


Fig. 1: Inference latency distribution with respect to the number of total instance tokens and agent tokens. We can find that the runtime is highly related to the number of total instances. Meanwhile, the runtime does not increase substantially as the number of agent tokens grows when the number of all instance tokens remains constant.

Previous methods leverage hierarchical structure [1, 2] and factorized attention [3, 4] to reduce the time complexity. For example, HiVT [2] first crops a local region and normalizes the local context centered on each target agent. Then, the local encoder is introduced to generate a single feature vector for each local region, which incorporates information related to the centered target agents. The local feature extraction contains three steps: 1) model agent-agent interaction feature per time step, 2) perform temporal fusion using a BERT-like network, and 3) fuse map features to local region features. After that, the global interaction module fuses and updates the local features in an all-to-all manner to aggregate the global information using Transformer.

We list the time complexity for each stage in Tab. I for a clear comparison. Benefiting from the hierarchical and factorized design, HiVT greatly reduces the theoretical time complexity. However, as a baseline method, SIMPL aims to minimize the involvement of inductive bias in the model design to demonstrate its effectiveness. Although real-time inference is already achieved by SIMPL in most real-world cases, it is noteworthy that benefiting from the use of standard Multi-Head Attention, SIMPL can be further accelerated by incorporating recent advancements in efficient Transformers, such as FlashAttention [5, 6] and token merge techniques [7, 8]. We leave them as interesting and important future works.

B. Discussion on Trajectory Parameterization

As stated in the paper, leveraging Bézier curves as the output trajectory form brings advantages when compared with predicting raw coordinates. One may ask what if we use the post-hoc curve fitting after generating the raw coordinate trajectories? To verify its performance, we add another curve fitting module after the trajectory decoder, and the detailed implementation is described as follows.

We denote the initial trajectory output generated by the decoder as $\bar{\mathbf{Y}} \in \mathbb{R}^{T\times 2}$, where T is the prediction length. We use an n-th order Bézier curve to fit the initial output, which can be denoted as

$$\begin{aligned}\mathbf{Y} &= \mathbf{B}\times\mathbf{P}\\ &= \begin{bmatrix} b_n^0(t_1) & b_n^1(t_1) & \dots & b_n^n(t_1)\\ \vdots & \vdots & & \vdots\\ b_n^0(t_T) & b_n^1(t_T) & \dots & b_n^n(t_T)\end{bmatrix}\begin{bmatrix} p_0^x & p_0^y\\ \vdots & \vdots\\ p_n^x & p_n^y\end{bmatrix},\end{aligned}$$

where $\mathbf{B}\in\mathbb{R}^{T\times(n+1)}$ is a constant basis matrix by definition, $\mathbf{P}\in\mathbb{R}^{(n+1)\times 2}$ is the control points. We use ordinary least squares (OLS) estimation to calculate the optimal control points of the fitted Bézier curve,

$$\hat{\mathbf{P}} = (\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T\bar{\mathbf{Y}}.$$

Therefore, we can further generate the fitted trajectory $\hat{\mathbf{Y}} = \mathbf{B}\hat{\mathbf{P}} = \mathbf{B}(\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T\bar{\mathbf{Y}}$. We add this procedure behind SIMPL with raw coordinates output to get the fitted trajectory and evaluate the results. As shown in Tab. II, cubic, quintic, and septic order Bézier curves are applied in the post-fitting module, and all of them can not achieve the same level of effectiveness as directly using the Bézier curve parameterization. Interestingly, we also find that higher degree Bézier curves bring better performance. We believe this is due to the higher representational capability of higher-order curves.

Actually, if we go a little deeper, it is easy to see using networks to predict the coefficients is basically identical to the aforementioned predict-and-fit procedure in the inference stage. Let $\mathbf{X}\in\mathbb{R}^{F\times 2}$ be the output feature of the final nonlinear layer, where F is the feature dimension. For control point prediction, we employ a linear layer to map the feature to the control points: $\mathbf{P} = \mathbf{W_1}\mathbf{X}$, where $\mathbf{W_1}\in$

TABLE I: Time complexity of each module. We denote the number of target agents as N, the number of total map features as M_{global}, and the number of historical frames as T. For HiVT, we use N_{local} and M_{local} to represent the maximum number of local neighbor agents and local map features.

Method	Embedding Stage	Fusion Stage
HiVT	$TN \times (N_{local} + M_{local})$	$\underbrace{TNN_{local} + NT^2 + NM_{local}}_{\text{local encoder}} + \underbrace{N^2}_{\text{global interaction}}$
SIMPL	$N + M_{global}$	$(N + M_{global})^2$

TABLE II: Results of the trajectory fitting methods on the Argoverse 2 validation set.

Parameterization	minADE_6	minFDE_6	MR_6	b-minFDE_6
Bézier curve (n=7)	**0.780**	1.457	**0.198**	**2.085**
Raw coords	**0.780**	**1.452**	0.203	2.088
Raw coords + fitting (n=3)	0.816	1.560	0.218	2.190
Raw coords + fitting (n=5)	0.803	1.482	0.205	2.105
Raw coords + fitting (n=7)	0.791	1.471	0.205	2.097

$\mathbb{R}^{(n+1)\times F}$ is the learnable weights. Then, the predicted trajectory can be obtained by multiplying the basis matrix $\mathbf{Y} = \mathbf{BP} = \mathbf{BW_1X}$ as stated in the paper. For the predict-and-fit method, we map the feature $\mathbf{X}$ to the raw coordinates using a linear layer $\bar{\mathbf{Y}} = \mathbf{W_2X}$, where $\mathbf{W_2} \in \mathbb{R}^{T\times F}$ is also the learnable weights. As stated above, adding the fitting procedure after the initial trajectory prediction can be represented as $\hat{\mathbf{Y}} = \mathbf{B}\hat{\mathbf{P}} = \mathbf{B}(\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T\mathbf{W_2X}$. Essentially, both of these two methods apply a linear mapping to the latent feature $\mathbf{X}$, and the only difference is the composition of the linear mapping. If curve fitting is introduced during the training phase, these two methods can be considered essentially equivalent. However, if the fitting is not performed during training and is only applied during inference, the two methods will differ in the gradient backpropagation process.

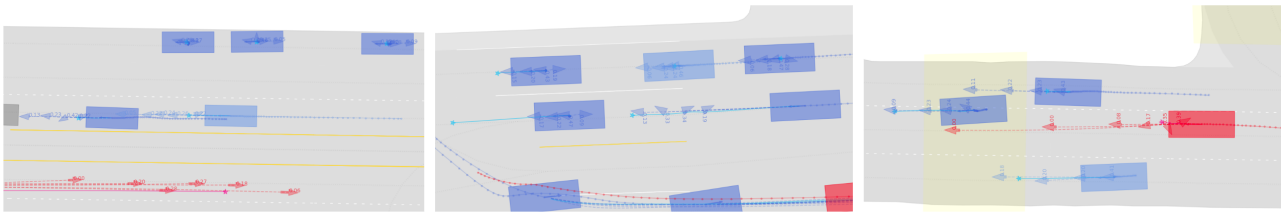
Intuitively, direct learning of the parameters of Bézier curves introduces more inductive biases, such as continuity and smoothness, making the task simpler, which may explain the experimental results shown in Tab. II. Additionally, the process of directly predicting Bézier coefficients is simpler than the predict-and-fit method, and it also involves fewer learnable parameters ($T \gg n$). Considering its marginally superior performance, we believe that the Bézier curve-based trajectory parameterization is a better approach.

C. Discussion on `YawLoss` for Low-speed Vehicles

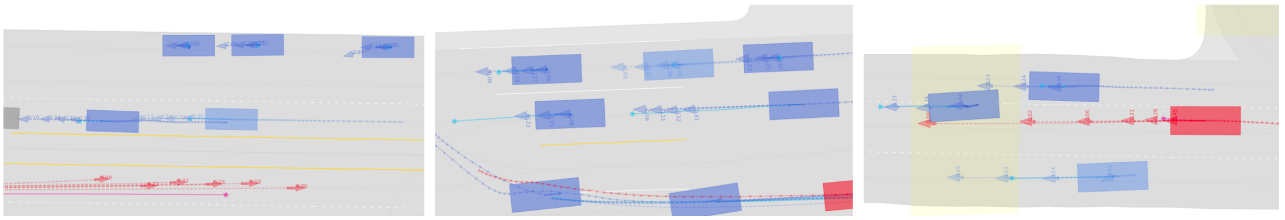
In this paper, the trajectory regression loss contains two parts, namely, the positional coordinate regression loss and the yaw regression loss. The positional coordinate regression is commonly used in previous literature, which only focuses on the positional displacement error and enforces no constraint between consecutive states in the trajectory. As we use the Bernstein basis polynomials as the parameterization, calculating yaw angles involves computing the first-order derivatives of the predicted trajectory. Therefore, incorporating the yaw angle loss implicitly strengthens the consistency between consecutive states, making the predicted trajectories with higher smoothness and kinematic feasibility. In practice, the aforementioned issue has little impact on medium to high-speed moving objects, but it has a significant effect on low-speed moving objects. We believe the primary reason is that at low speeds, the state positions along the trajectory are closely spaced, and even small amounts of noise or prediction errors can have a significant impact on their relative positional relationships, while this effect is not as pronounced for high-speed objects. We provide several examples in Fig. 2 to demonstrate the effectiveness of yaw angle loss. If we only use the positional displacement loss, the predicted trajectories contain more noise and distortion for low-speed and static target objects. Even worse, when we calculate the yaw angle by the positional displacement, the results will be more unreasonable (see Fig. 2a). In contrast, predicted trajectories are much smoother after applying the yaw angle loss, especially for low-speed objects.

D. Discussion on Robustness

Following previous methods based on symmetric scene modeling [2, 9]–[11], the term "robust" is specifically used to describe the characteristic of viewpoint invariance in our predictive model. This viewpoint invariance ensures that the model's predictions are stable across different rotations or translations, highlighting its robustness in a variety of scenarios. As stated in our paper, all methods based on symmetric scene modeling inherently possess this characteristic of viewpoint invariance. Specifically, SIMPL employs instance-centric scene representation and relative positional encoding (RPE) to represent the relative spatial relationships between different instances. This approach naturally decouples the network's output from the absolute positional coordinates of the inputs. As a result, the output remains consistent irrespective of changes in the input's spatial positioning, further reinforcing the robustness of our model in handling various scenarios. To verify this property, here we provide additional experimental results on the robustness against viewpoint shifting. We rotate the scenes at various angles and then evaluate the output of networks. As illustrated in Fig. 3, both SIMPL and HiVT achieve constant performance with the change of rotation, while LaneGCN is sensitive to the rotation angle of the scene. Similar results can also be found in HiVT [2] and GoRela [9].



(a) Without yaw loss



(b) With yaw loss

Fig. 2: Comparative results on the effectiveness of yaw angle loss for low-speed moving objects. The ground truth endpoints are denoted as stars and the predicted trajectories are depicted as dashed curves with the final poses marked as arrows. We can see that the introduction of yaw loss smoothens the predicted trajectories significantly, and the final heading angle is much more reasonable.

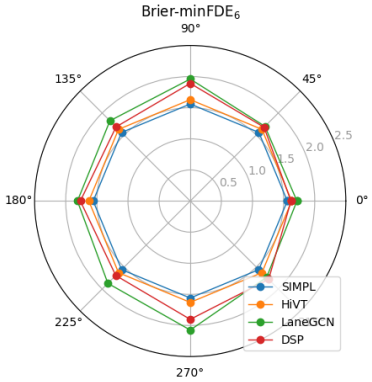


Fig. 3: Illustration of the robustness against viewpoint shifting. We use brier-minFDE$_6$ on the Argoverse 1 dataset validation split as the evaluation metric. We can see that methods based on symmetric scene modeling (SIMPL and HiVT) perform stably while others are sensitive to the rotation of scenes.

E. Discussion on Statistical Results

Given the inherent stochasticity in the training process, statistical analysis is important in evaluating the model's performance. Here we provide statistical results from multiple trainings of SIMPL. We trained 8 model instances with different random seeds and the results are shown in Tab. III. It is evident that the training of SIMPL is quite stable, as indicated by the relatively small standard deviations of its metrics.

It's noteworthy that in the motion prediction domain, most studies typically report only the best results, and detailed statistical metrics from these studies are often unavailable, restricting us from performing comparative statistical analyses, such as the Student's t-test. Nevertheless, we simply compare SIMPL's statistical results against the best results of other baseline methods, which are demonstrated in Tab. IV. From the data presented, we can find that SIMPL constantly outperforms other baselines w.r.t. the official ranking metric (brier-minFDE$_6$), and achieves competitive results in other metrics even using the statistical results.

TABLE III: Results of different SIMPL instances on the test split of Argoverse 1 motion forecasting dataset.

Models	minFDE$_6$	MR$_6$	b-minFDE$_6$
$\mathcal{M}1$	1.181	12.1	1.822
$\mathcal{M}2$	1.177	12.1	1.814
$\mathcal{M}3$	1.186	12.4	1.821
$\mathcal{M}4$	1.179	12.3	1.809
$\mathcal{M}5$	1.183	12.4	1.821
$\mathcal{M}6$	1.187	12.2	1.822
$\mathcal{M}7$	1.186	12.5	1.822
$\mathcal{M}8$	1.184	12.4	1.817
mean±std	1.183±0.00331	12.4±0.13	1.819±0.00470

TABLE IV: Comparative results on the test split of Argoverse 1 motion forecasting dataset, where b-minFDE$_6$ is the official ranking metric.

Method	minFDE$_6$	MR$_6$	b-minFDE$_6$
LaneGCN	1.362	16.2	2.053
mmTrans	1.338	15.4	2.033
SceneTrans	1.232	12.6	1.887
HiVT	1.169	12.7	1.842
MacFormer	1.216	12.1	1.827
SIMPL (best)	1.179	12.3	1.809
SIMPL (statistical)	1.183±0.00331	12.4±0.13	1.819±0.00470

REFERENCES

[1] Y. Liu, *et al.*, "Multimodal motion prediction with stacked transformers," in *CVPR*, 2021, pp. 7577–7586.

[2] Z. Zhou, *et al.*, "HiVT: Hierarchical vector transformer for multi-agent motion prediction," in *CVPR*, 2022, pp. 8823–8833.

[3] J. Ngiam, *et al.*, "Scene Transformer: A unified architecture for predicting multiple agent trajectories," *arXiv preprint arXiv:2106.08417*, 2021.

[4] N. Nayakanti, *et al.*, "Wayformer: Motion forecasting via simple & efficient attention networks," in *ICRA*. IEEE, 2023, pp. 2980–2987.

[5] T. Dao, *et al.*, "Flashattention: Fast and memory-efficient exact attention with io-awareness," *NeurIPS*, vol. 35, pp. 16 344–16 359, 2022.
[6] T. Dao, "Flashattention-2: Faster attention with better parallelism and work partitioning," *arXiv preprint arXiv:2307.08691*, 2023.
[7] D. Bolya, *et al.*, "Token merging: Your vit but faster," *arXiv preprint arXiv:2210.09461*, 2022.
[8] C. Renggli, *et al.*, "Learning to merge tokens in vision transformers," *arXiv preprint arXiv:2202.12015*, 2022.
[9] A. Cui, *et al.*, "GoRela: Go relative for viewpoint-invariant motion forecasting," in *ICRA*. IEEE, 2023, pp. 7801–7807.
[10] X. Jia, *et al.*, "HDGT: Heterogeneous driving graph transformer for multi-agent trajectory prediction via scene encoding," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2023.
[11] Z. Zhou, *et al.*, "Query-centric trajectory prediction," in *CVPR*, 2023, pp. 17 863–17 873.