Higher-order Relational Reasoning for Pedestrian Trajectory Prediction

Supplementary Material

This supplementary material provides additional explanations and experimental results of our paper 'Higher-order Relational Reasoning for Pedestrian Trajectory Prediction'. First, we formulate our proposed collision-aware kernel function in detail. Next, we elaborate on how our HighGraph is implemented into each baseline. Furthermore, we present more qualitative examples that demonstrate the advantages of HighGraph. Finally we conclude with discussions of limitations and direction of our future work.

1. Collision-aware Kernel Function

As mentioned in our main paper, the proposed collision-aware kernel function quantifies the pair-wise interactions based on the observed movement from the previous timestamp. First, we construct a graph $G^{(t)}$ for each timestamp $t \in \{1, 2, ..., T\}$, where T is the total number of observed timestamps. We consider agents in the scene as nodes and their connections as edges. The connections are encoded in the adjacency matrix $\mathbf{A} \in \mathbb{R}^{T \times N \times N}$, where N is the number of agents in the scene. Then for each $\mathbf{A}^{(t)} \in \mathbf{A}$, its elements $\{a_{ij}^{(t)} | \forall i, j \in \{1, 2, ..., N\}\}$ are initialized as follows:

$$a_{ij}^{(t)} = \begin{cases} 0, & \text{if } i = j \\ 1, & \text{otherwise.} \end{cases} \tag{1}$$

Next, our collision-aware kernel function takes these adjacency matrices as input, and updates their elements. As mentioned in the main paper, our collision-aware kernel function is designed based on the prior that the pedestrians in a crowd walk in a way that avoids collisions. Therefore, we explicitly compute the potential collision points for all agent pairs in each timestamp.

The potential collision point for each pair is computed by the intersection of two half-lines, each of which starts from the agent's previous location P and passes the current location Q. For example, the potential collision point for an agent pair $\{i, j\}$ at time t can be formulated as:

$$C_{ij}^{(t)} = (x_{ij}^{(t)}, y_{ij}^{(t)}) = \overrightarrow{P_i Q_i} \cap \overrightarrow{P_j Q_j}. \tag{2}$$

Then, if the intersection exists (i.e. $\exists C_{ij} \in \mathbb{R}^2$), the function assigns weights reciprocally to the distance from the agents to the potential collision point. In other words, the function weighs a higher value for the pairs with imminent potential collision and a lower value for the pairs with remote potential collision. What also matters is the distance between each agent. It can distinguish the pairs with the same distance to the potential collision point, but with different distances between the agents. Therefore, the inverse of the multiplication of the two distance values is assigned as an updated weight for each pair. The overall formulation of the collision-aware kernel function is summarized in Algorithm 1. Additionally, in Figure 1, we provide a visualization of how our collision-aware kernel function finds the potential collision point for each consecutive timestamp.

Algorithm 1 Collision-aware Kernel Function

Input: Number of agents N, Number of timestamps T

Initialize adjacency matrices $\mathbf{A} \in \mathbb{R}^{T \times N \times N}$ (Eq.1)
for each $t \in \{1, 2, ..., T\}$ **do**
  if $t == 1$ **then**
    continue
  else
    for each $i \in \{1, 2, ..., N\}$ **do**
      for each $j \in \{i+1, i+2, ..., N\}$ **do**
        $P_i, Q_i \leftarrow (x_i^{(t-1)}, y_i^{(t-1)}), (x_i^{(t)}, y_i^{(t)})$
        $P_j, Q_j \leftarrow (x_j^{(t-1)}, y_j^{(t-1)}), (x_j^{(t)}, y_j^{(t)})$
        $C_{ij}^{(t)} = (x_{ij}^{(t)}, y_{ij}^{(t)}) \leftarrow \overrightarrow{P_i Q_i} \cap \overrightarrow{P_j Q_j}$ (Eq.2)
        if $\exists C_{ij} \in \mathbb{R}^2$ **then**
          $d_{ij}^{(t)} \leftarrow \|Q_i - Q_j\|_2$
          $a_{ij}^{(t)} \leftarrow 1 / \left(d_{ij}^{(t)} \sum_{e \in \{i,j\}} \|C_{ij}^{(t)} - Q_e\|_2 \right)$
          $a_{ji}^{(t)} \leftarrow 1 / \left(d_{ij}^{(t)} \sum_{e \in \{i,j\}} \|C_{ij}^{(t)} - Q_e\|_2 \right)$
        else
          $a_{ij}^{(t)} \leftarrow 0$
          $a_{ji}^{(t)} \leftarrow 0$
      if $t == 2$ **then**
        $a_{ij}^{(t-1)} \leftarrow a_{ij}^{(t)} \; \forall i, j \in \{1, 2, ..., N\}$

Output: Updated adjacency matrices

2. Implementation Detail

Our HighGraph module is designed to be plug-and-play, which is put together with existing trajectory predictors to provide fruitful higher-order relational feature embeddings. In this section, we describe how HighGraph is implemented in our selected baselines: Social-GAN [3], SoPhie [6], Social-STGCNN [5], BiTraP [9], SocialVAE [8] and EigenTrajectory [1]. For all baselines, the rest of the modules and settings that are not mentioned in this section are left untouched. The details of the hyperparameters are reported in Table 1.

2.1. Social-GAN + HighGraph

Social-GAN proposes a pooling module to encode social interaction among agents. We place our HighGraph before the

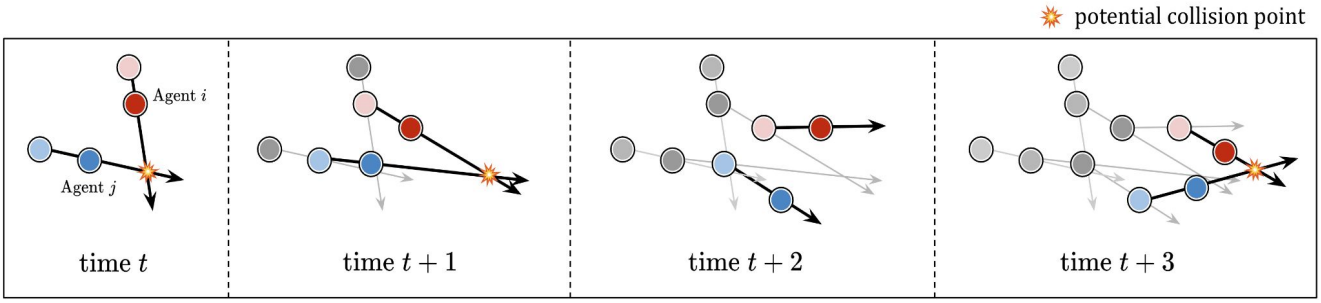

Figure 1. An illustration of how our collision-aware kernel function finds the potential collision points in four timestamps.

pooling module to first provide the model with more comprehensive social features. In detail, HighGraph takes the features generated from their existing LSTM [4] encoder as input, and the learned features of HighGraph are then delivered to the pooling module.

2.2. SoPhie + HighGraph

SoPhie models social and physical interactions jointly using an attention mechanism. However, due to the absence of the image features in their repository, we implement HighGraph on SoPhie-$\mathbf{T_A}$, which obtains social features only with the social attention module. In our experiment, we locate HighGraph between the feature extraction module and the attention module of SoPhie.

2.3. Social-STGCNN + HighGraph

Social-STGCNN is a representative graph-based trajectory prediction method. First, we enhance their existing interaction modeling with our collision-aware kernel function. Additionally, since our higher-order graph convolution of HighGraph includes the existing pair-wise graph convolution, we replace it with our module.

2.4. BiTraP + HighGraph

We implement HighGraph on BiTraP-NP, a non-parametric model of BiTraP, and plug our HighGraph parallel to the original encoder of BiTraP. Since BiTraP uses RNN to condense the temporal movements of the agents as one hidden feature, we also place one gated recurrent unit (GRU) [2] layer after HighGraph operation to match the feature dimensions. Then, the features are column-concatenated and mapped into the CVAE [7] latent space.

2.5. SocialVAE + HighGraph

SocialVAE is a recent CVAE-based trajectory prediction method that models time-wise CVAE latents. Similar to the implementation of BiTraP, our HighGraph is placed parallel to its existing observational encoder, and the features are column-concatenated. However, unlike BiTraP, the concatenation is done for the features in each timestamp to be aligned with their main contribution.

2.6. EigenTrajectory + HighGraph

EigenTrajectory is a recently introduced plug-and-play method for pedestrian trajectory prediction that transforms the raw data into the proposed EigenTrajectory space. In our experiment, we attach HighGraph at the end of the Euclidean space decoder module. The baseline module is set as EigenTrajectory+SGCN.

3. Qualitative Analysis

In this section, we provide more examples from qualitative experiments that demonstrate the benefits of HighGraph. First, we visualize the single-agent predictions with the SDD dataset in Figure 2. Then, in Figure 3, we present more predictions of multi-agent higher-order scenarios.

3.1. Single-agent General Prediction.

Similar to the reported analysis in the main paper, HighGraph noticeably improves the baseline performance in a single-agent prediction. Especially, HighGraph significantly reduces the number of socially-unacceptable trajectories. Also, we observe that by incorporating HighGraph, the baselines become better at capturing the nonlinear movements of the agents.

3.2. Multi-agent Higher-order Prediction.

Continuing from section 5.6.2 of the main paper, we present more results in higher-order social scenarios with multiple agents in the scene.

Scenario 5. In this scenario, a stationary agent (yellow) influences the green agent to walk to its right (first-order). It also influences the red agent to walk to its left (first-order). Then, the red agent encounters the green agent and slows down (second-order). This whole interaction between the three agents affects the purple agent to walk to its left (third-order), and the blue agent moves accordingly to the purple agent (fourth-order).

Method	Hyperparameter	ETH	HOTEL	UNIV	ZARA1	ZARA2	SDD
Social-GAN [3]	Epoch	200	200	200	200	200	200
+ HighGraph	Learning Rate	5e-4	5e-4	5e-4	5e-4	5e-4	5e-4
	Batch Size	64	64	64	64	64	64
	Graph Layers	1	1	1	1	1	1
	Higher-order Obs.	3	3	3	3	3	3
	Hidden Dim	64	64	64	64	64	64
SoPhie [6]	Epoch	250	250	250	250	250	250
+ HighGraph	Learning Rate	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3
	Batch Size	1	1	1	1	1	1
	Graph Layers	2	2	2	2	2	2
	Higher-order Obs.	3	3	3	3	3	2
	Hidden Dim	64	64	64	64	64	64
Social-STGCNN [5]	Epoch	250	250	250	250	250	250
+ HighGraph	Learning Rate	1e-2	1e-2	1e-2	1e-2	1e-2	1e-2
	Batch Size	128	128	128	128	128	128
	Graph Layers	2	2	2	2	2	2
	Higher-order Obs.	2	2	2	2	2	2
	Hidden Dim	5	5	5	5	5	5
BiTraP [9]	Epoch	50	50	50	50	50	50
+ HighGraph	Learning Rate	1e-3	5e-3	5e-3	5e-3	5e-3	5e-3
	Batch Size	128	64	64	64	32	32
	Graph Layers	3	3	1	2	1	3
	Higher-order Obs.	3	3	2	5	3	3
	Hidden Dim	32	32	32	32	32	32
SocialVAE [8]	Epoch	1000	800	200	600	600	600
+ HighGraph	Learning Rate	4e-4	8e-4	1e-3	2e-4	4e-4	3e-4
	Batch Size	32	32	128	128	128	512
	Graph Layers	1	1	1	1	1	5
	Higher-order Obs.	3	3	3	3	3	2
	Hidden Dim	64	64	64	64	64	64
EigenTrajectory [1]	Epoch	256	256	256	256	256	200
+ HighGraph	Learning Rate	1e-3	1e-3	1e-3	1e-3	1e-3	5e-4
	Batch Size	128	128	128	128	128	128
	Graph Layers	2	2	2	2	2	3
	Higher-order Obs.	3	2	3	3	3	2
	Hidden Dim	128	128	128	128	128	128

Table 1. This table details the best-performing hyperparameters for each baseline. We mainly follow the default configurations provided by each repository, but some were inevitably manipulated due to the reproduction issue. After reproducing the metrics reported from each paper, we fix the default hyperparmeters, and conduct extensive experiments on the HighGraph hyperparameters which are the number of graph layers, the higher-order observation degree, and the graph hidden dimension.

Scenario 6. The two parallel walking agents (first-order) affect the yellow agent to stir to its left (second-order). Then, the yellow agent runs into the red and green agents from its left and turns right (third-order). The red and green agents accordingly slow down due to the movement of the yellow agent (fourth and fifth-order).

Scenario 7. The blue and gray agents are stationary in this scenario (first-order). Their position affects the trajectories of the yellow agent to slightly move to its left (second-order). Then the yellow agent runs into the red agent and both walk to their right to avoid collision with each other (third-order).

Scenario 8. In this scenario, the red and the purple agent first bend the trajectory of the green agent (first-order). This further influences the blue agent to turn right (second-order). Then, the gray agent encounters the blue agent and turns right (third-order).

Through the results of these examples, we observe that

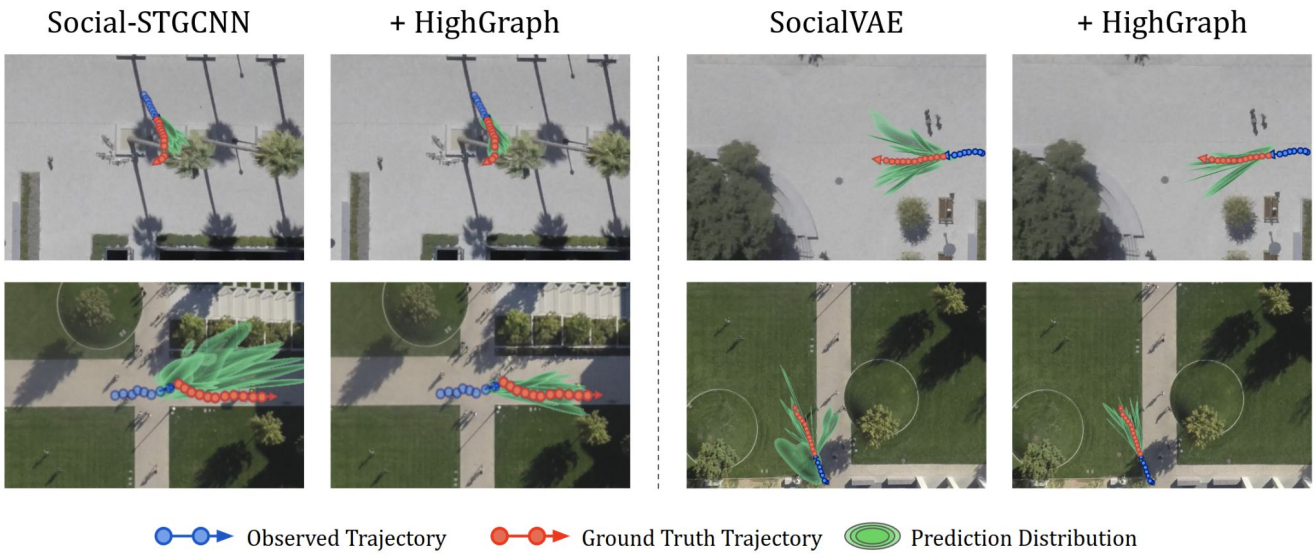

Figure 2. Visualization of the qualitative influence of HighGraph. We illustrate the result of SDD dataset, comparing the original model and the HighGraph-plugged model. Best viewed in color.

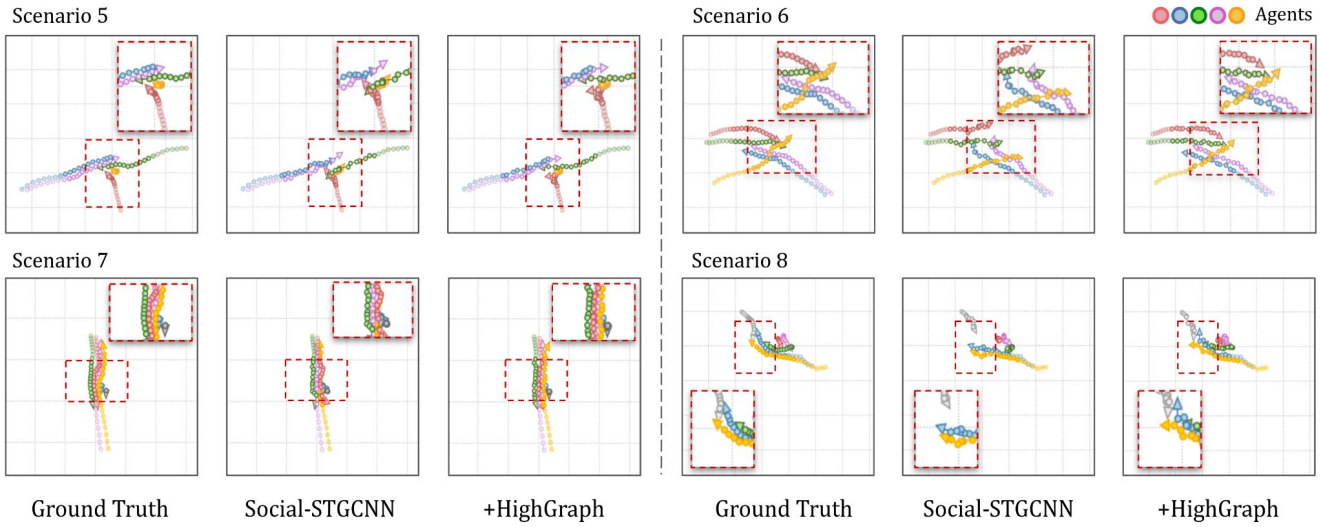

Figure 3. Visualization of how HighGraph can improve the predictions in higher-order social scenarios. Best viewed in color.

HighGraph conspicuously enhances the performance in higher-order scenarios. This is due to the explicit modeling of the collision with our collision-aware kernel function and the higher-order graph convolutions that encodes social influences from multiple distances.

4. Limitation and Future Work

HighGraph models the higher-order social dynamics only with the positional data. Therefore, as mentioned in Section 5.5 of our main paper, the benefits are minimal in cases where only a small number of pedestrians are present. To overcome this limitation, we plan to extend our model to incorporate heterogeneous data (e.g. images, texts) and discover more semantics that can complement the lack of data.

References

[1] Inhwan Bae, Jean Oh, and Hae-Gon Jeon. Eigentrajectory: Low-rank descriptors for multi-modal trajectory forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10017–10029, 2023. 1, 3

[2] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent

neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. 2

[3] 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, 3

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2

[5] Abduallah Mohamed, Kun Qian, Mohamed Elhoseiny, and Christian Claudel. Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14424–14432, 2020. 1, 3

[6] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, Hamid Rezatofighi, and Silvio Savarese. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1349–1358, 2019. 1, 3

[7] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015. 2

[8] Pei Xu, Jean-Bernard Hayet, and Ioannis Karamouzas. Socialvae: Human trajectory prediction using timewise latents. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part IV*, pages 511–528. Springer, 2022. 1, 3

[9] Yu Yao, Ella Atkins, Matthew Johnson-Roberson, Ram Vasudevan, and Xiaoxiao Du. Bitrap: Bi-directional pedestrian trajectory prediction with multi-modal goal estimation. *IEEE Robotics and Automation Letters*, 6(2):1463–1470, 2021. 1, 3