

Supplementary Materials: Neural Interaction Energy for Multi-Agent Trajectory Prediction

A DETAILS OF MODEL ARCHITECTURE

A.1 MATE-Encoder

We use a multi-edge graph neural network to get the interaction representation. Given observation trajectories of agents X_1, X_2, \dots, X_N , we omit the superscripts for convenience and compute the trajectories in following operations:

$$\begin{aligned} g_i^{(1)} &= f_{\text{emb}}(X_i), \\ v \rightarrow e : z_{ij}^{(1)} &= f_z^{(1)}([g_i^{(1)}, g_j^{(1)}]), \\ e \rightarrow v : g_i^{(2)} &= f_v^{(1)}(\sum_{i \neq j} z_{(i,j)}^{(1)}), \\ v \rightarrow e : z_{ij}^{(2)} &= f_z^{(2)}([g_i^{(2)}, g_j^{(2)}]), \\ &\dots \end{aligned} \quad (\text{Sp1})$$

where f_* are MLPs, $g_i^{(m)}$ and $g_j^{(m)}$ denote the embedding of agent i and agent j after m -th round of message passing, $z_{ij}^{(m)}$ denotes the embedding of edge (i, j) after m -th round of message passing. Note that we denote z_{ij}^k to describe the k -th dimension of the edge latent z_{ij} in the Methodology Section and denote $z_{ij}^{(m)}$ to describe the m -th round of message passing. $z_{ij}^{(1)}$ represent the interaction between agent i and agent j , and $z_{ij}^{(2)}$ involve whole information from the graph. We normalize the same dimension of edges between agent i and its neighbors j :

$$z_{ij}^k = \frac{\exp(z_{ij}^k)}{\sum_{j \in N_i} \exp(z_{ij}^k)}, \quad \forall k \in K. \quad (\text{Sp2})$$

N_i denotes the neighbor set of agent i . By doing this, we use the normalized latent code set $z = \{z_{ij} \mid 1 \leq i, j \leq N\}$ as the output of the Encoder.

A.2 MATE-Decoder

We use a Neural Interaction Energy Module (NIEM) unit to predict trajectories step by step. The basic structure of NIEM is as follows:

$$\begin{aligned} I_i^t &= [f_{\text{embx}}(x_i^t), f_{\text{embv}}(v_i^t)], \\ \text{MSG}_i^t &= \sum_{j \in N_i} \sum_{k=1}^{K} z_{ij}^k f_e^k([h_i^t, h_j^t]), \\ h_i^{t+1} &= \text{GRU}(\text{MSG}_i^t, I_i^t), \\ v_i^{t+1} &= f_{\text{Energy}}(h_i^{t+1}), \\ x_i^{t+1} &= x_i^t + \Delta t \cdot v_i^{t+1}, \\ \mathcal{L}_E, \mathcal{L}_D &= f_{\text{CM}}(E, x_i^{t+1}, x_i^t, v_i^t), \end{aligned} \quad (\text{Sp3})$$

where f_{embx} and f_{embv} are two MLP layers for embedding position x and velocity v of agent i into features, I_i^t denotes the concatenated feature of agent i at step t , h_i^t denotes the hidden state of agent i at

step t , x_i^{t+1} and v_i^{t+1} denotes the position and velocity of agent i at step $t + 1$, N_i denotes the neighbor set of agent i .

f_{CM} denotes the Constraint Module, which is the calculation process of the inter-agent interaction constraint and the intra-agent motion constraint we introduced in the Methodology Section.

f_{Energy} denotes the Energy Module, containing several MLPs to get the neural interaction energy features of agents and subsequently predict the kinematic states of agents. Note that the incorporation of inter-agent interaction constraint is related to the design of the Energy Module. We propose two implementation structures for the Energy Module in the following paragraphs.

Structure (1): using the inter-agent interaction constraint directly as part of the loss function. We design the Energy Module as two MLP layers and generate the neural interaction energy features for each agent, then we perform the calculation process discussed in the Methodology Section and add the inter-agent interaction constraint as part of the loss function for model optimization.

Structure (2): integrating the inter-agent interaction constraint into the Energy Module. One of our core ideas lies in preserving system-level temporal stability that the sum of changes in neural interaction energy of agents tends to be zero over a short period. To simplify the illustration, we will use two sets of agents, namely A and B as an example. In the ideal situation, the sum of the neural interaction energy changes of A is the negative sum of B . Considering this, we normalize neural interaction energy features of A and concatenate these features. Then we take the concatenated feature as the input of B , that is

$$\begin{aligned} N_s &= \frac{\partial E_s}{\partial h_s}, \quad \forall s \in A, \\ H_B &= [N_1, N_2, \dots, N_s], \end{aligned} \quad (\text{Sp4})$$

where N_s denotes the normalized neural interaction energy features of agent s . We predict the velocity of agents in set A and set B separately.

We show the architecture of two proposed structures in Fig. 1. For each dataset, the superior result from two structures was selected as the final outcome.

B MATHEMATICAL DERIVATION OF INTRA-AGENT MOTION CONSTRAINT

Here we give a brief derivation process of Eq.(6) in the Methodology Section. Taking the inspiration from dynamic equations, we have

$$F^t = ma^t, \quad x^t \approx x^{t-1} + \Delta t \cdot v^{t-1} + \Delta t^2 \cdot F^t, \quad (\text{Sp5})$$

where F^t is the force of agents and Δt is timestep. x^t and v^t denote the position and velocity of the current step, respectively. x^{t-1} and v^{t-1} denote that of the previous step. We consider mass m as a constant for simplicity. We leverage partial differential equations (PDEs) to represent the relation between predicted states and input states. Given the partial equation concerning x , F and E , $F^t = \frac{\partial E^t}{\partial x^t}$,

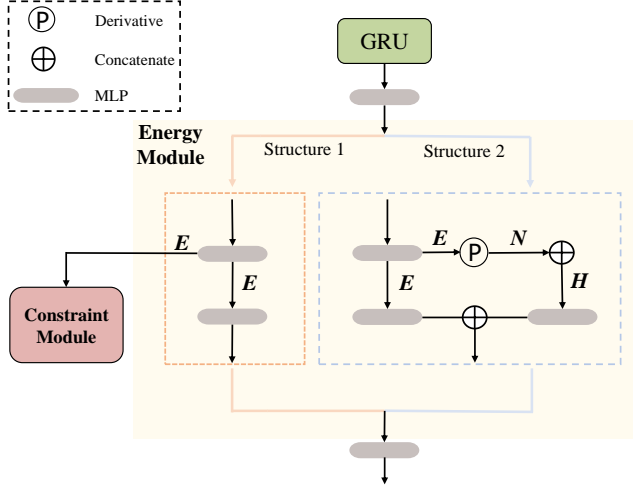


Figure 1: Two alternative structures of Energy Module.

Dataset	PHASE	Socialnav	Charged	NBA
α	0.5	0.5	0.4	0.8
β	-1	-1	-1	-1
γ	0.0625	0.0625	0.04	0.16

Table 1: Hyperparameters for intra-agent motion constraint.

we have,

$$\frac{\partial \mathbf{x}^t}{\partial \mathbf{v}^{t-1}} \approx \frac{\partial \mathbf{x}^{t-1}}{\partial \mathbf{v}^{t-1}} + \Delta t \cdot \frac{\partial \mathbf{v}^{t-1}}{\partial \mathbf{v}^{t-1}} + \Delta t^2 \cdot \frac{\partial \mathbf{F}^t}{\partial \mathbf{v}^{t-1}}, \quad (\text{Sp6})$$

$$\begin{aligned}
u^t &\approx \frac{\partial \mathbf{x}^{t-1}}{\partial \mathbf{v}^{t-1}} + \Delta t \cdot \frac{\partial \mathbf{v}^{t-1}}{\partial \mathbf{v}^{t-1}} + \Delta t^2 \cdot \frac{\partial \mathbf{E}^t}{\partial \mathbf{v}^{t-1} \partial \mathbf{x}^t} - \frac{\partial \mathbf{x}^t}{\partial \mathbf{v}^{t-1}}, \\
&\approx \Delta t + \Delta t + \Delta t^2 \cdot \frac{\partial \nabla_{\mathbf{x}^t} E_i(\mathbf{x}^t)}{\partial \mathbf{v}^{t-1}} - \frac{\partial \mathbf{x}^t}{\partial \mathbf{v}^{t-1}}, \quad (\text{Sp7}) \\
&= \gamma \cdot \frac{\partial \nabla_{\mathbf{x}} E^t(\mathbf{x})}{\partial \mathbf{v}^{t-1}} + \beta \cdot \frac{\partial \mathbf{x}^t}{\partial \mathbf{v}^{t-1}} + \alpha.
\end{aligned}$$

We present the hyperparameter determination in Table 1. Specifically, we set β to -1 and determine the value of α and γ according to the timestep Δt .

As presented in Eq. (Sp7), u^t is a 1×2 matrix and it measures the gap between the predicted movement ($\frac{\partial \mathbf{x}^t}{\partial \mathbf{v}^{t-1}}$) and the approximate movement ($\frac{\partial \mathbf{x}^{t-1}}{\partial \mathbf{v}^{t-1}} + \Delta t \frac{\partial \mathbf{v}^{t-1}}{\partial \mathbf{v}^{t-1}} + \Delta t^2 \frac{\partial \mathbf{F}^t}{\partial \mathbf{v}^{t-1}}$). By regularizing the norm of u , the predicted trajectory is refined to better align with the ground-truth motion, yielding coherent and physically plausible agent motions. Consequently, we can leverage this intra-agent motion constraint to preserve agent-level temporal stability.

C BROADER IMPACTS

This work has the potential for wide-ranging applications in autonomous driving, mobile robot navigation, molecular simulation, and team sports. Such systems have the potential for negative societal impact, and our algorithm has limitations and uncertainties. We must critically evaluate such applications and promote beneficial

ones. We also should ensure that the application scenarios of the algorithm are scientific.

D COMPLEXITY AND SCALABILITY

we supplement the discussion about the computational cost of our approach. Compared to previous works, we additionally employ two constraints to preserve temporal stability by computing the partial derivatives of the neural interaction energy E concerning the agent's position \mathbf{x} and velocity \mathbf{v} in Eq. (4) and Eq. (6). Our additional complexity brought by the partial derivatives is approximately $O(b * T * d)$, where b and T denote batch size and time period, d is the feature dimension of E . We take experiments on the Socialnav dataset (containing 100K trajectory samples) for example to analyze computational costs. We take 4 hours to train our model. The model has 0.63M parameters. During the inference stage, we spent 0.04ms to predict a sample's trajectory. It means that our model is efficient and applicable in real-time processing. Thus, our model can be used in real-world scenarios.

E REPRODUCIBILITY

We submit the code of our model as part of the supplementary to show the reproducibility of our method. The code contains an MATE-Encoder and an MATE-decoder. We use the multi-edge graph network to get the interaction latent for agents. We use a sequential model to predict trajectories step by step.

F ADDITIONAL QUALITATIVE ANALYSIS

We show the visualizations of our model and other models on three datasets in Fig. 2, Fig. 3, and Fig. 4. In each dataset, our model performs the best prediction accuracy.

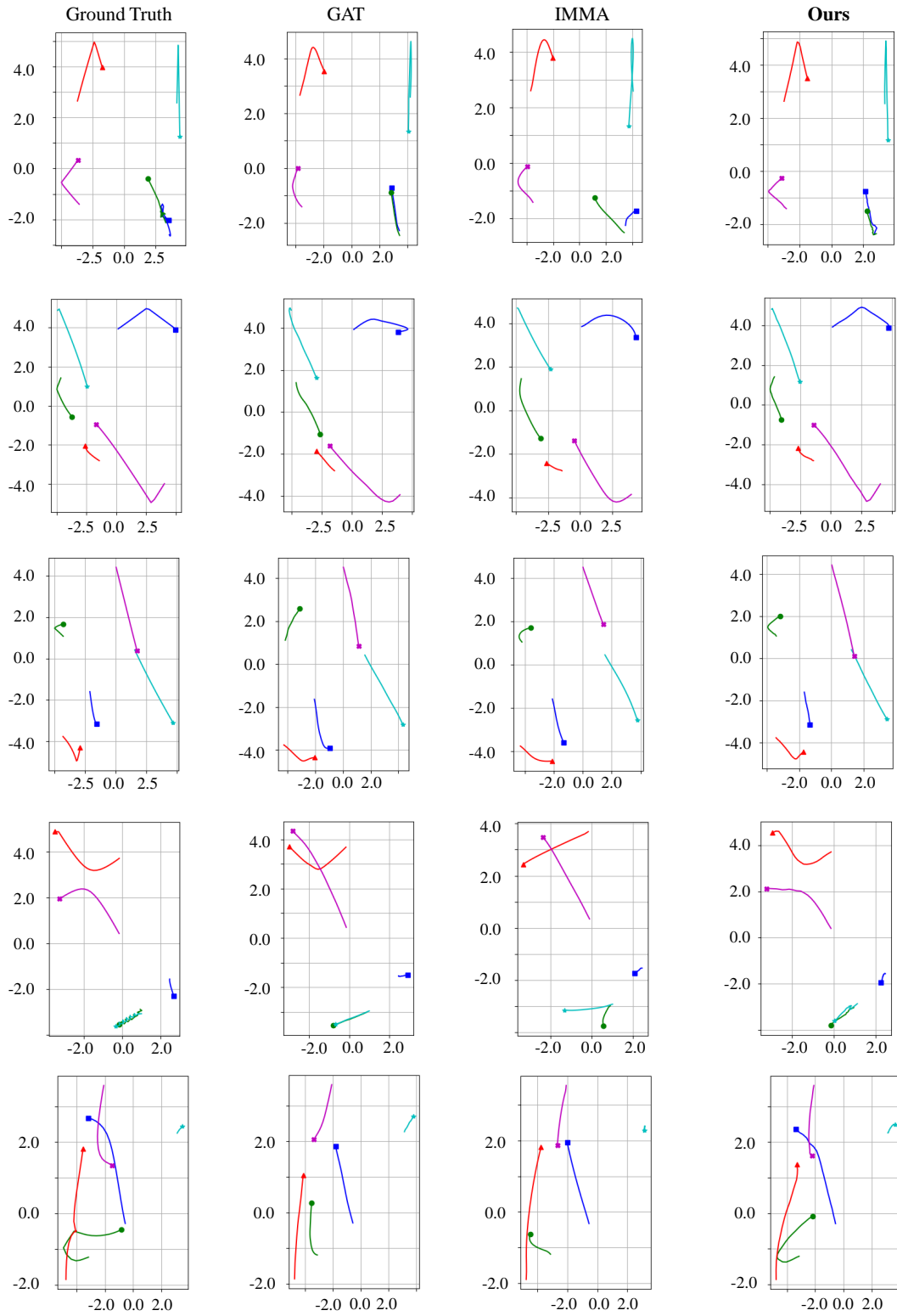


Figure 2: Visualized prediction results of our model and other models on Charged dataset. Each row is a test sample and different colored lines denote trajectories of different charges.

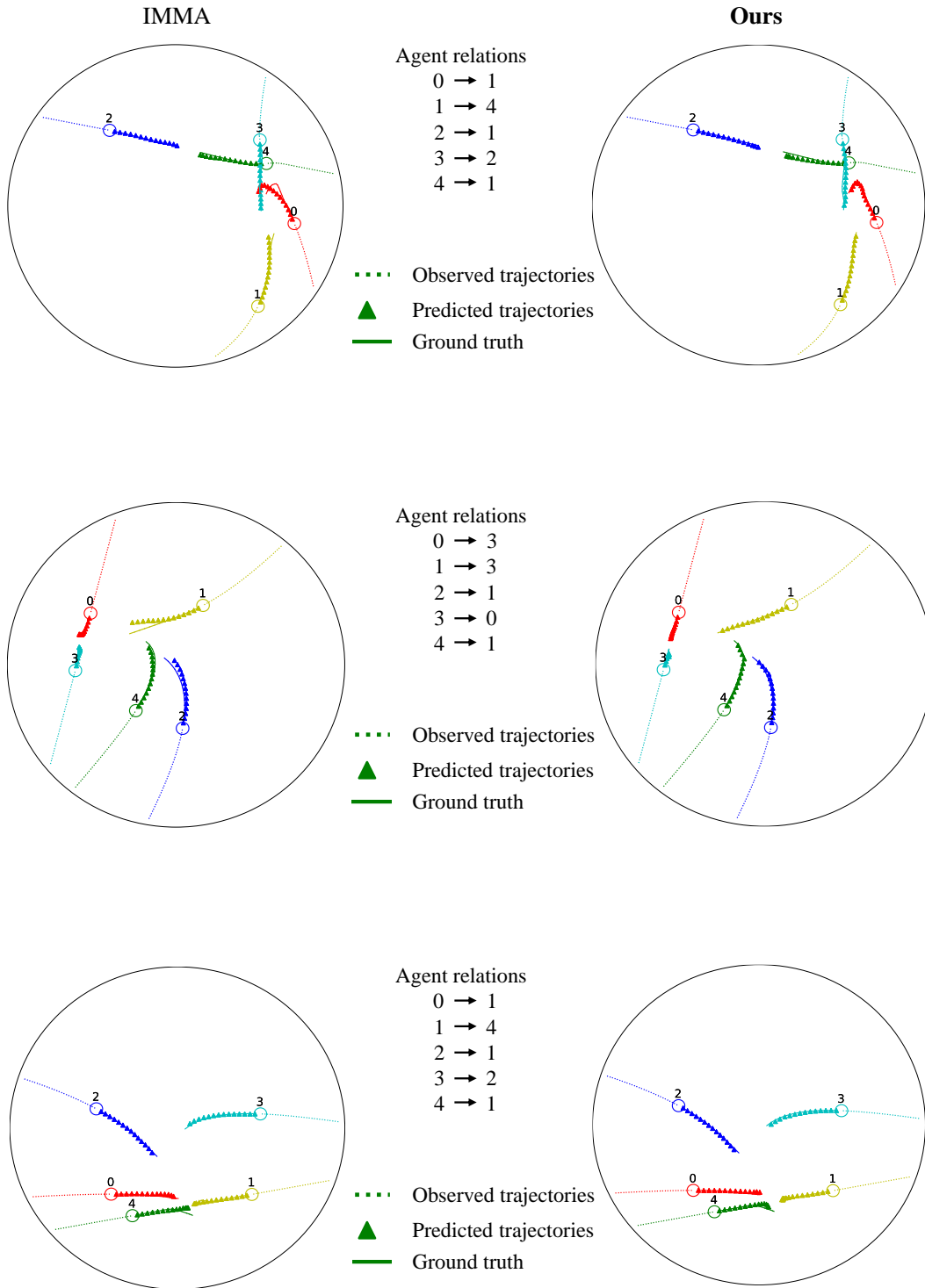


Figure 3: Visualized prediction results of our model and IMMA model on Socialnav dataset. Each row is a test sample and different colored lines denote trajectories of different agents. Dotted lines: past trajectories. Solid lines: ground truth future trajectories. Triangle: predicted future trajectories. Agents relations means that the agent to the left of the arrow moves towards the agent to the right of the arrow.

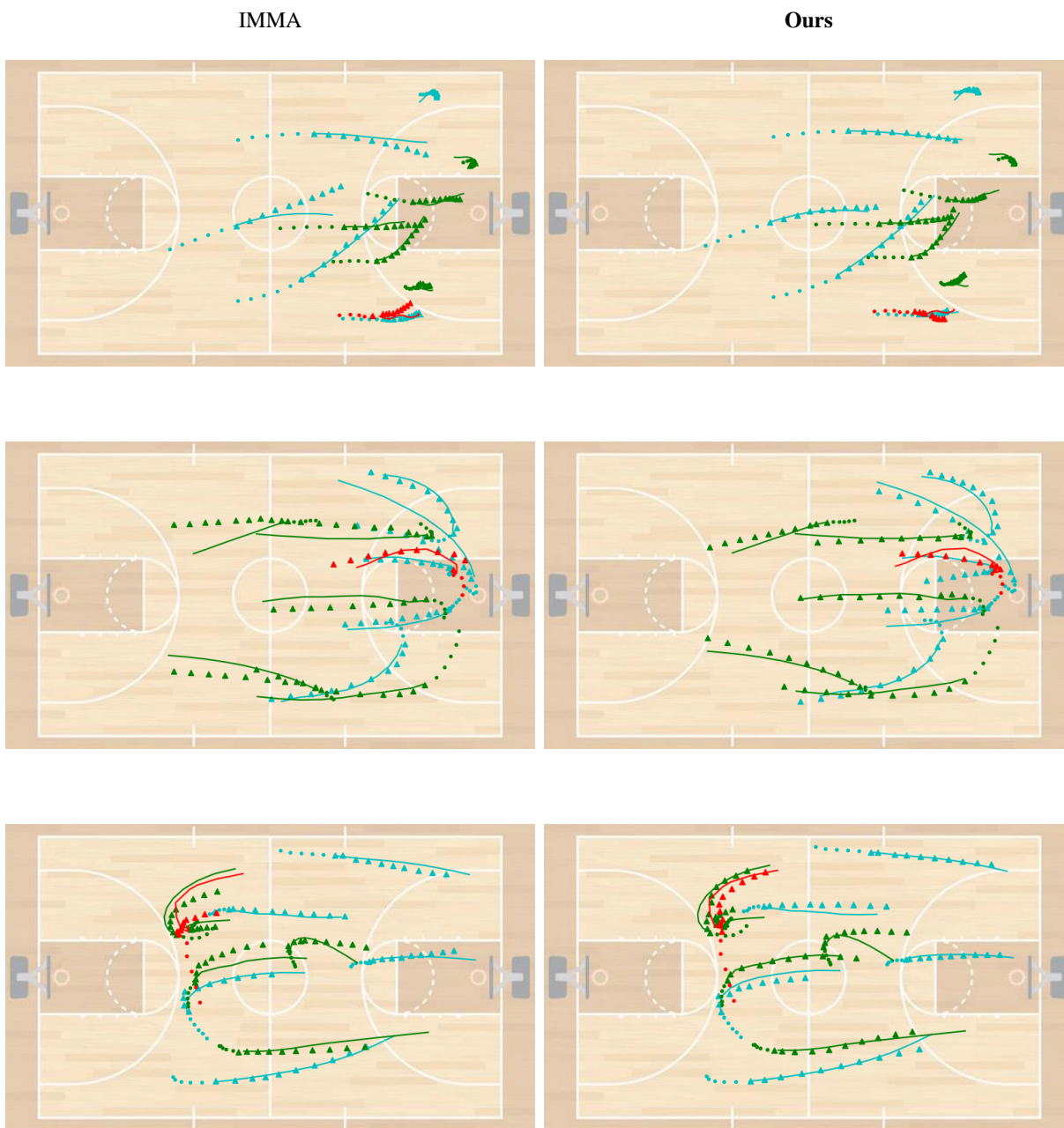


Figure 4: Visualized prediction results of our model and IMMA on NBA dataset. Each row is a test sample and different colored lines denote the trajectories of different players and the ball. Dotted lines: past trajectories. Solid lines: ground truth future trajectories. Triangle: predicted future trajectories.