Supplementary Material of Transformer with Implicit Edges for Particle-based Physics Simulation

Yidi Shao[1], Chen Change Loy[1], and Bo Dai[2]

[1] S-Lab for Advanced Intelligence, Nanyang Technological University
yidi001@e.ntu.edu.sg, ccloy@ntu.edu.sg
[2] Shanghai AI Laboratory
daibo@pjlab.org.cn

A Model Details

A.1 Decomposing GNN

In the following, we show the detailed deduction of implicitly modeling edges in TIE. When omitting the normalization, bias, and activation, we can update the edge propagation function implemented by MLPs in GNN by

$$\boldsymbol{e}_{ij}^{(l+1)} = W^{(l)} \left[\boldsymbol{v}_i^{(l)}; \boldsymbol{v}_j^{(l)}; \boldsymbol{e}_{ij}^{(l)} \right], \tag{1}$$

where $W^{(l)} \in \mathbb{R}^{d \times 3d}$ is the parameter for MLPs, and $[\cdot;\cdot]$ denotes the concatenation. By splitting $W^{(l)}$ into 3 different square blocks $W^{(l)} = [W_r^{(l)}, W_s^{(l)}, W_m^{(l)}]$ and expanding the edge embeddings, we have

$$\begin{aligned}
\boldsymbol{e}_{ij}^{(l+1)} &= W_r^{(l)} \boldsymbol{v}_i^{(l)} + W_s^{(l)} \boldsymbol{v}_j^{(l)} + W_m^{(l)} \boldsymbol{e}_{ij}^{(l)} && (2)\\
&= \left(W_r^{(l)} \boldsymbol{v}_i^{(l)} + W_m^{(l)} W_r^{(l-1)} \boldsymbol{v}_i^{(l-1)} \right) && (3)\\
&\quad + \left(W_s^{(l)} \boldsymbol{v}_j^{(l)} + W_m^{(l)} W_s^{(l-1)} \boldsymbol{v}_j^{(l-1)} \right) && (4)\\
&\quad + W_m^{(l)} W_m^{(l-1)} \boldsymbol{e}_{ij}^{(l-1)} && (5)\\
&= \left(W_r^{(l)} \boldsymbol{v}_i^{(l)} + \sum_{u=1}^{l} \left(\prod_{k=0}^{u-1} W_m^{(l-k)} \right) W_r^{(l-u)} \boldsymbol{v}_i^{(l-u)} \right) && (6)\\
&\quad + \left(W_s^{(l)} \boldsymbol{v}_j^{(l)} + \sum_{u=1}^{l} \left(\prod_{k=0}^{u-1} W_m^{(l-k)} \right) W_s^{(l-u)} \boldsymbol{v}_j^{(l-u)} \right), && (7)
\end{aligned}$$

where we assume $\boldsymbol{v}_i^{(0)} = \boldsymbol{x}_i$, and $[W_r^{(0)}, W_s^{(0)}] = W^{(0)}$ are the parameters for the edge initialization function $f_E^{\text{enc}}(\cdot)$. Assuming $\boldsymbol{r}_i^{(l)} = W_r^{(l)} \boldsymbol{v}_i^{(l)} + W_m^{(l)} \boldsymbol{r}_i^{(l-1)}$ and $\boldsymbol{s}_j^{(l)} = W_s^{(l)} \boldsymbol{v}_j^{(l)} + W_m^{(l)} \boldsymbol{s}_j^{(l-1)}$, we can further simplify Eq.7 by

$$\begin{aligned}
\boldsymbol{e}_{ij}^{(l+1)} &= \left(W_r^{(l)} \boldsymbol{v}_i^{(l)} + W_m^{(l)} \boldsymbol{r}_i^{(l-1)} \right) + \left(W_s^{(l)} \boldsymbol{v}_j^{(l)} + W_m^{(l)} \boldsymbol{s}_j^{(l-1)} \right) && (8)\\
&= \boldsymbol{r}_i^{(l)} + \boldsymbol{s}_j^{(l)}, && (9)
\end{aligned}$$

where we assume $\boldsymbol{r}_i^{(0)} = W_r^{(0)}\boldsymbol{v}_i^{(0)}$ and $\boldsymbol{s}_j^{(0)} = W_s^{(0)}\boldsymbol{v}_j^{(0)}$, $l \in \{1, 2, \cdots, L\}$ is the index of the block.

By combining Equation 7 into the self-attention formula in Transformer [3], we have:

$$\omega'_{ij} = (W_Q^{(l)}\boldsymbol{v}_i^{(l)})^\top(\boldsymbol{r}_i^{(l)} + \boldsymbol{s}_j^{(l)}) \tag{10}$$

$$= (W_Q^{(l)}\boldsymbol{v}_i^{(l)})^\top\boldsymbol{r}_i^{(l)} + (W_Q^{(l)}\boldsymbol{v}_i^{(l)})^\top\boldsymbol{s}_j^{(l)}, \tag{11}$$

$$\hat{\omega}'_{ij} = \text{softmax}\left(\frac{\omega'_{ij}}{\sqrt{d}}\right), \tag{12}$$

$$\boldsymbol{v}_i^{(l+1)} = \sum_j \hat{\omega}'_{ij} \cdot (\boldsymbol{r}_i^{(l)} + \boldsymbol{s}_j^{(l)}) \tag{13}$$

$$= \sum_j \hat{\omega}'_{ij} \cdot \boldsymbol{r}_i^{(l)} + \sum_j \hat{\omega}'_{ij} \cdot \boldsymbol{s}_j^{(l)} \tag{14}$$

$$= \boldsymbol{r}_i^{(l)} + \sum_j \hat{\omega}'_{ij} \cdot \boldsymbol{s}_j^{(l)}. \tag{15}$$

A.2 Normalization Effects

Given Equation 7, we further modify the self-attention in Equation 11 and Equation 15 to include the effects of normalization in GNN for edges. Since GNN-based methods usually incorporate LayerNorm [1] in their network architectures that computes the mean and std of edge features to improve their performance and training speed, we propose to apply the effects of normalization for each edge in GNN to our model. For the mean $\mu_{ij}^{(l)}$ and std $\sigma_{ij}^{(l)}$ of each interaction between particle i and j, we can compute them from receiver token $\boldsymbol{r}_i^{(l)}$ and sender token $\boldsymbol{s}_j^{(l)}$ by

$$\mu_{ij}^{(l)} = \frac{1}{d}\sum_k \left(r_{ik}^{(l)} + s_{jk}^{(l)}\right) \tag{16}$$

$$= \frac{1}{d}\left(\sum_k r_{ik}^{(l)} + \sum_k s_{jk}^{(l)}\right) \tag{17}$$

$$= \mu_{r_i}^{(l)} + \mu_{s_j}^{(l)}, \tag{18}$$

$$\left(\sigma_{ij}^{(l)}\right)^2 = \frac{1}{d}\sum_k \left(r_{ik}^{(l)} + s_{jk}^{(l)} - \mu_{ij}^{(l)}\right)^2 \tag{19}$$

$$= \frac{1}{d}\sum_k \left((r_{ik}^{(l)})^2 + (s_{jk}^{(l)})^2 + (\mu_{ij}^{(l)})^2 + 2r_{ik}^{(l)}s_{jk}^{(l)} - 2\mu_{ij}^{(l)}(r_{ik}^{(l)} + s_{jk}^{(l)})\right) \tag{20}$$

$$= \frac{1}{d}\left((\boldsymbol{r}_i^{(l)})^\top \boldsymbol{r}_i^{(l)} + (\boldsymbol{s}_j^{(l)})^\top \boldsymbol{s}_j^{(l)}\right) + (\mu_{ij}^{(l)})^2 + \frac{2}{d}(\boldsymbol{r}_i^{(l)})^\top \boldsymbol{s}_j^{(l)} - 2(\mu_{ij}^{(l)})^2 \tag{21}$$

$$= \frac{1}{d}(\boldsymbol{r}_i^{(l)})^\top \boldsymbol{r}_i^{(l)} + \frac{1}{d}(\boldsymbol{s}_j^{(l)})^\top \boldsymbol{s}_j^{(l)} + \frac{2}{d}(\boldsymbol{r}_i^{(l)})^\top \boldsymbol{s}_j^{(l)} - (\mu_{ij}^{(l)})^2 \tag{22}$$

$$= \frac{1}{d}(\boldsymbol{r}_i^{(l)})^\top \boldsymbol{r}_i^{(l)} + \frac{1}{d}(\boldsymbol{s}_j^{(l)})^\top \boldsymbol{s}_j^{(l)} + \frac{2}{d}(\boldsymbol{r}_i^{(l)})^\top \boldsymbol{s}_j^{(l)} - (\mu_{r_i}^{(l)} + \mu_{s_j}^{(l)})^2 \tag{23}$$

where $r_{ik}^{(l)}$ and $s_{jk}^{(l)}$ are respectively the k-th element in $\boldsymbol{r}_i^{(l)}$ and $\boldsymbol{s}_j^{(l)}$, $\mu_{r_i}^{(l)}$ and $\mu_{s_j}^{(l)}$ are respectively the mean of receiver token $\boldsymbol{r}_i^{(l)}$ and sender token $\boldsymbol{s}_j^{(l)}$ after l-th block. Hence, by replacing $\boldsymbol{r}_i^{(l)} + \boldsymbol{s}_j^{(l)}$ in Equation 10 and Equation 13 by $\frac{\boldsymbol{r}_i^{(l)} + \boldsymbol{s}_j^{(l)} - \mu_{ij}^{(l)}}{\sigma_{ij}^{(l)}}$, we have

$$\omega_{ij}'' = \frac{(W_Q^{(l)}\boldsymbol{v}_i^{(l)})^\top(\boldsymbol{r}_i^{(l)} - \mu_{r_i}^{(l)}) + (W_Q^{(l)}\boldsymbol{v}_i^{(l)})^\top(\boldsymbol{s}_j^{(l)} - \mu_{s_j}^{(l)})}{\sigma_{ij}^{(l)}}, \tag{24}$$

$$\hat{\omega}_{ij}'' = \mathrm{softmax}\left(\frac{\omega_{ij}''}{\sqrt{d}}\right), \tag{25}$$

$$\boldsymbol{v}_i^{(l+1)} = \sum_j \hat{\omega}_{ij}'' \cdot \frac{\boldsymbol{r}_i^{(l)} - \mu_{r_i}^{(l)}}{\sigma_{ij}^{(l)}} + \sum_j \hat{\omega}_{ij}'' \cdot \frac{\boldsymbol{s}_j^{(l)} - \mu_{s_j}^{(l)}}{\sigma_{ij}^{(l)}}, \tag{26}$$

When it comes to the scaling and shifting parameters in LayerNorm [1], we add them into Equation 26 to better resemble the normalization effects.

B Experiment Details

B.1 Implementation Details

Inputs and outputs details. For *FluidFall*, *FluidShake*, and *BoxBath*, we only use particles' states at time t as inputs and output the velocities at time $t+1$. For *RiceGrip*, we concatenate particles states from $t-2$ to t as inputs and output 6-dim vector for the velocity of the current observed position and the resting position. For *BoxBath*, we output 7-dim vectors, where 3 dimensions for the predicted velocities, and 4 dimensions for rotation constrains. The rotation constraints, which predict the rotation velocities, are applied only on rigid particles, which is the same as mentioned in DPI-Net [2]. All states of particles, such as the positions and velocities, are first normalized by mean and standard deviations calculated on corresponding training set before they are fed into the models.

Training. We train four models independently on four domains, with 5 epochs on *FluidShake* and *BoxBath*, 13 epochs on *FluidFall*, and 20 epochs on *RiceGrip*. For common settings, we adopt Adam optimizer with an initial learning rate of 0.0008, which has a decreasing factor of 0.8 when the validation loss stops to decrease after 3 epochs. The batch size is set to 16 on all domains. All models are trained and tested on V100 for all experiments, with no augmentation involved.

Baseline details. For fair comparison, the following settings are the same with TIE: inputs for models, number of training epochs on different domains, learning rate schedules, and training loss on velocities. Hyper-parameters for baselines are first chosen the same as their original papers, and then fine-tuned within a small range of changes. For example, in terms of the batch size, 16 works better for DPI-Net than the original settings.

B.2 Data Generation

Basic Domains. We use the same setting for our datasets as mentioned in previous work [2]. *FluidFall* contains two fluid droplets with different sizes. The sizes for droplets are randomly generated with one droplet larger than the other. Positions and viscosity for droplets are randomly initialized. This domain contains 189 particles with 121 frames for each rollout. There are 2700 rollouts in training set and 300 rollouts in validation set. *FluidShake* simulates the water in a moving box. The speed of the box is randomly generated at each timestamp. In addition, the size of the box and the number of particles are various for different rollouts. In basic training and validation sets, the number of particles varies from 450 to 627. This domain has 301 frames for each rollout. There are 1800 rollouts in training set and 200 rollouts in validation set. *RiceGrip* contains two grippers and a sticky rice. The grippers' positions and orientations are randomly initialized. The number of particles for rice varies from 570 to 980 with 41 frames for each rollout in training and validation sets. There are 4500 rollouts in training set and 500 rollouts in validation set. *BoxBath* simulates a rigid cube washed by water in a fixed container. The initial positions of fluid block and rigid cube are randomly initialized. This domain contains 960 fluid particles and 64 rigid particles with 151 frames for each rollout. There are 2700 rollouts in training set and 300 rollouts in validation set.

Generalization Domains. We release the details of generalization settings in Table 1. We add more particles for *FluidShake* and *RiceGrip*, which we refer to as *L-FluidShake* and *L-RiceGrip* respectively. The *L-FluidShake* includes 720 to 1368 particles, while *L-RiceGrip* contains 1062 to 1642 particles. On *BoxBath*, we enlarge the fluid block and change the size and shape of rigid object. Specifically, we add more fluid particles in *Lfluid-BoxBath* to 1280 fluid particles, while we enlarge the rigid cube in *L-BoxBath* to 125 particles. We also change the shape of the rigid object into ball and bunny, which we refer to *BallBox* and *BunnyBath* respectively. The number of test rollouts and the number of frames for each rollout are the same as the corresponding basic domains.

Table 1. Details of generalization settings. We list the number of particles in both training domains and generalization domains. The lists of numbers in *L-FluidShake* and *L-RiceGrip* are the range of particles, while the number of rigid and fluid particles in generalized *BoxBath* are listed separately.

Domains	Training Settings	Generalization Settings
L-FluidShake	[450, 627]	[720, 1368]
L-RiceGrip	[570, 980]	[1062, 1642]
Lfluid-BoxBath	Fluid: 960. Rigid: 64	Fluid: 1280. Rigid 64
L-BoxBath	Fluid: 960. Rigid: 64	Fluid: 960. Rigid: 125
BunnyBath	Fluid: 960. Rigid: 64	Fluid: 960. Rigid: 41
BallBath	Fluid: 960. Rigid: 64	Fluid: 960. Rigid: 136

B.3 Rendered Rollouts

We visualize some rollouts on *BoxBath* and its generalized domains, which are complex domains with multi-materials interactions. We simplify *BoxBath* into 5 key steps: 1. the rigid object flooded by fluid hits the wall; 2. the rigid object is thrown into the air; 3. the rigid object falls into the fluid; 4. the fluid, after hitting the wall, pushes the rigid object; 5. the rigid object slows down and stops moving. The visual results and analysis are shown in the following. Our TIE+ achieves more faithful rollouts on all the domains, suggesting the effectiveness of our implicitly modeled edges and the abstract particles.

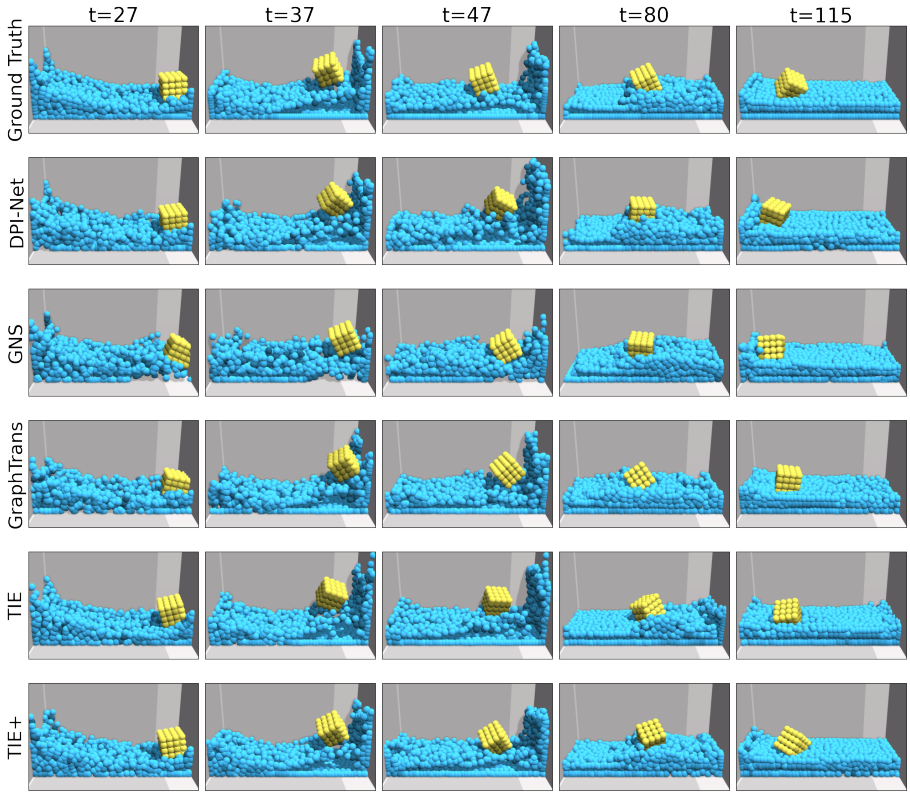

Fig. 1. Qualitative results on *BoxBath*. For step 2 and 3, the rigid cube is flooded into the air and pushed away from the right wall. Our TIE+ is able to achieve more faithful rollouts even with more accurate rotation angles, and the cube is pushed far away enough from the right wall. For step 5, when the cube slows down and finally stops, the position of the rigid cube predicted by our TIE+ is closer to the ground truth. For the simulations of the fluid particles, our model achieves more vivid results compared with other models. For example, the surface of water is smooth and is closer to the ground truth.

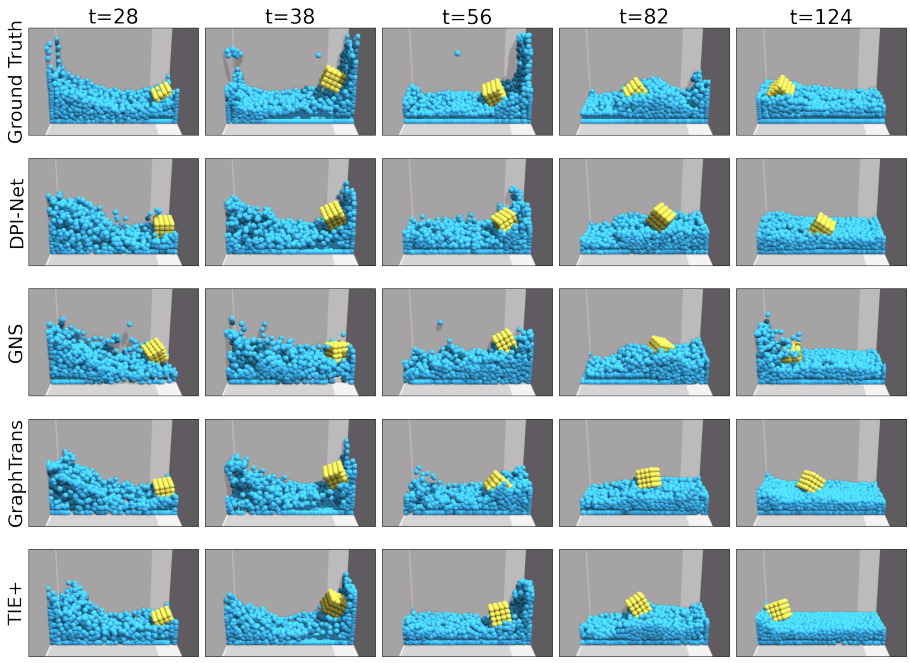

Fig. 2. Qualitative results on *Lfluid-BoxBath*, where we add more fluid particles. When focusing on the rigid cube, the positions and rotation angles achieved by our TIE+ are much closer to the ground truth. When focusing on the fluid particles, our TIE+ predicts more vivid wave, which floods the box and pushes it towards left in a more faithful manner.

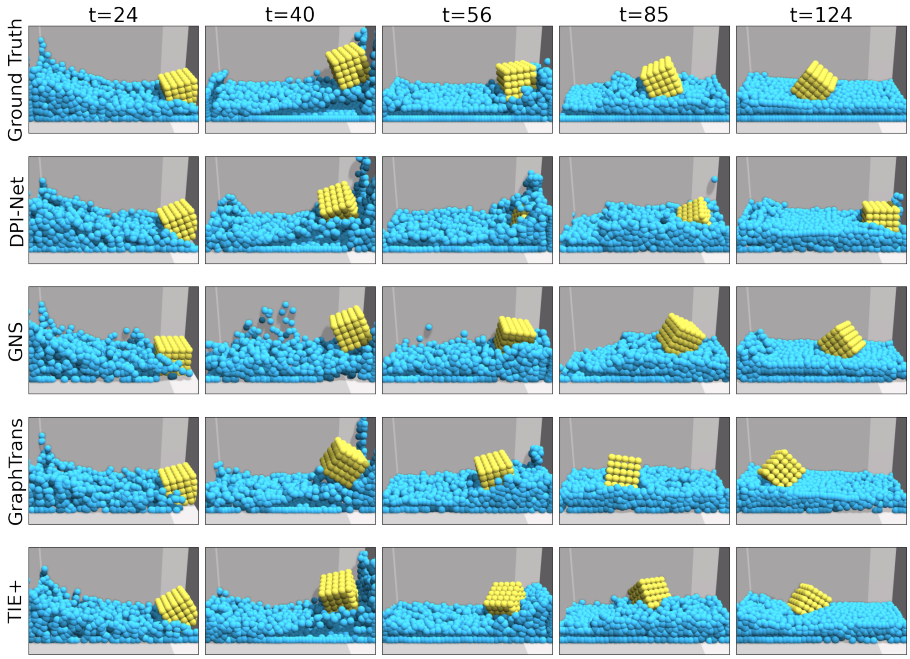

Fig. 3. Qualitative results on *L-BoxBath*, where we enlarge the rigid cube. Notice that the rigid cubes predicted by DPI-Net and GNS tends to rotate and move in the same place, DPI-Net and GNS have more difficulties in being generalized to this domain. The rigid cube predicted by GraphTrans is overly pushed by the wave at $t = 56$, while does not fully interact with the left wall to bounce back at $t = 124$. For both the fluid part and the rigid part, TIE+ still predicts more faithful results.

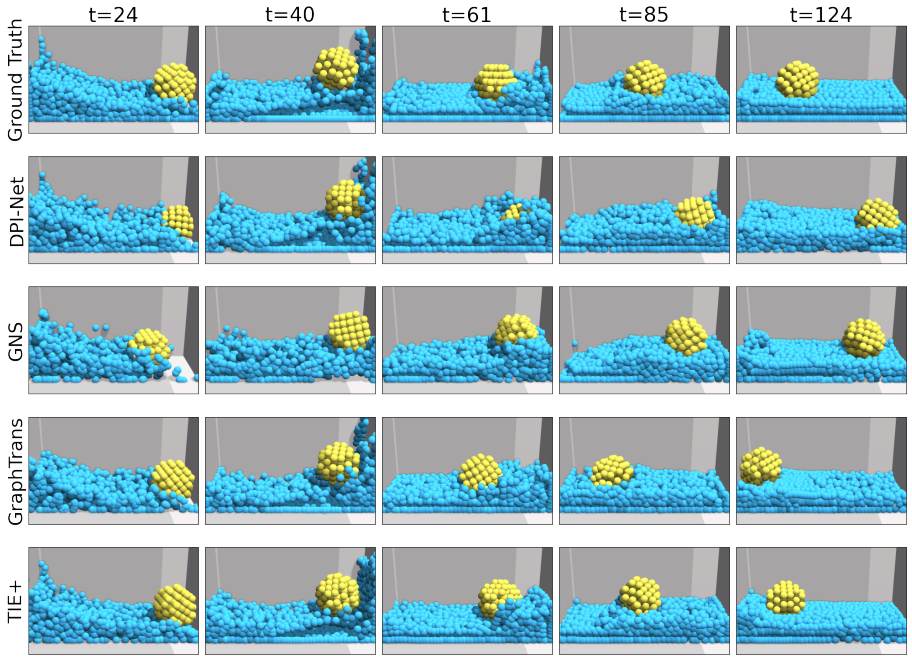

Fig. 4. Qualitative results on *BallBath*, where we change the cube into ball. The balls predicted by both DPI-Net and GNS rotate and move in the same place till the end, while GraphTrans has difficulties predicting the positions of the ball. TIE+ still achieves faithful rollout.

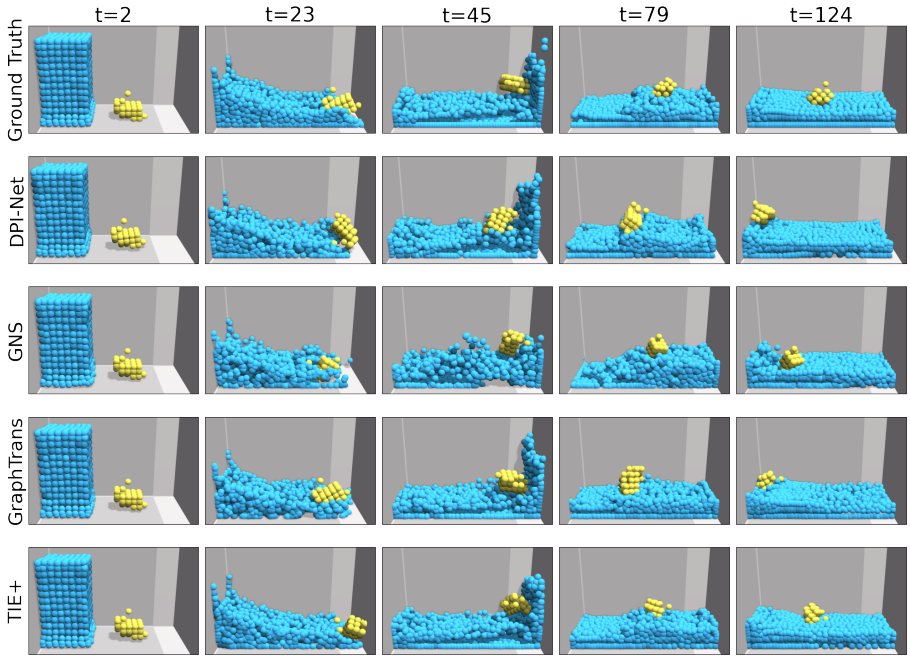

Fig. 5. Qualitative results on *BunnyBath*, where we change the cube into bunny with more complex surfaces. TIE+ achieves more faithful rollout. At time $t = 2$, we show the shape of the bunny, which has complex surfaces. At time $t = 45$, while TIE+ is able to rollout vivd dynamics of fluid particles, TIE+ can predict closer positions of the bunny, which is rotating and flying in the air.

References

1. Ba, L.J., Kiros, J.R., Hinton, G.E.: Layer normalization. CoRR (2016)
2. Li, Y., Wu, J., Tedrake, R., Tenenbaum, J.B., Torralba, A.: Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019 (2019)
3. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA (2017)