(Appendix) MeshGraphNetRP: Improving Generalization of GNN-based Cloth Simulation

A DATASET CONSTRUCTION

Figure 1 shows the segments of the translation path for our training dataset. For the rotation trajectory, we rotated the mesh along the

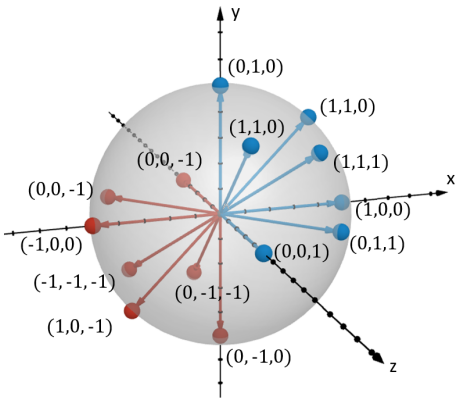

Figure 1: The different trajectory segments on 3D space we sampled to train our model. Training on these segments are enough for the model to infer the cloth dynamics on all directions in 3D space.

axis of the midpoint of its handles. For each trajectory, we moved the handle by 1 unit length for 1 second and move it back and forth along the segment path. Figure 2 shows the position and velocity of the handles as seen on its line of trajectory. By using only

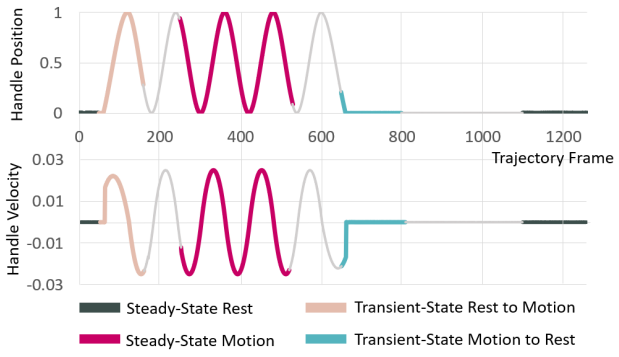

Figure 2: The trajectory of cloth handles as it moves to the different training trajectory segments. It shows the motion sections we used to optimize the training dataset size.

these training data, the model is able to learn the cloth dynamics for unseen directions for translation and different curvatures for rotation.

Preprocessing. From the trajectories we obtained, we further split it into sections to reduce the training dataset by avoiding redundancy in the exhibited patterns of cloth dynamics as much as possible. We obtained four section from the trajectories: (1) steady-state rest, (2) steady-state motion, (3) transient-state rest to motion and (4) transient-state motion to rest. Figure 2 shows the sections for a trajectory.

For steady-state rest, since all 16 trajectories exhibit the same pattern such that all start with the cloth in rest state and all ends with the cloth in rest state, we only obtained the steady-state rest section for any of the 16 trajectories. We get these steady-state rest section with 50 frames at the start and 100 frames at the end of any trajectory for a total of 150 frames. For the stead-state motion, a cloth moving in the segment of (1,0,0) and a cloth moving in the segment (-1,0,0) have the same movement pattern. The same goes with other segments and its opposite segment counterpart. Thus, we only get the steady-state motion section of 280 frames from any of the two.

Finally, both transient-state rest to motion and transient-state motion to rest show unique patterns for each trajectory. For example, even (1,0,0) and (-1,0,0) segments have different patterns of transient-state. We found that if we train the model with the transient-states on only one segment without using the opposite counterpart, the model fails to infer the times when the cloth is in the opposite counterpart of the segment and starts to move from rest to motion or when it goes from motion to rest. Thus, we get the transient-states of 110 frames for rest to motion and 150 frames for motion to rest for each of the 16 trajectories.

B MODEL DETAILS

B.0.1 Scheduled Sampling. We applied scheduled sampling in our training scheme by modifying the probability p that determines if the output of the model would be used as the input for the next time step. We trained our model such that $p = 1$ (supervised training) from the starting epochs 1 to 75. Starting from epoch 76, we linearly decay p to zero until epoch 300. And after epoch 300, we set $p = 0$ (auto-regressive training) until the last epoch, epoch 800. Take note that we only allow a maximum of 30 frames of rollout in the training to ensure stability.

B.0.2 EPD Architecture Configuration. For our RNN encoder, we use GRU with 2 hidden layers followed by a fully connected output layer whose output is normalized by LayerNorm. This is the case for both the node model and edge model of the encoder. The output dimension is set to 128. With the input dimensions set to the corresponding size of the node and edge features.

For the processor, we use MLP with 2 hidden layers followed by a fully connected output layer whose output is normalized by LayerNorm. This is also the case for both the node model and edge model. The input as well as the output dimension of these MLPs are 128. We set the number of message passing to be 15 which means that we have a total of 15 processor blocks in the processor. For

casting the associated edge features on a given node, we use sum operation as the message passing aggregator.

Lastly, for the decoder we only have one node model made up of an MLP with 2 hidden layers and a fully connected output layer with an input dimension of 128 and output dimension of 3.