MeshGraphNetRP: Improving Generalization of GNN-based Cloth Simulation

Emmanuel Ian Libao
KAIST
Daejeon, South Korea
iandlibao@kaist.ac.kr

Myeongjin Lee
KAIST
Daejeon, South Korea
myeongjin.lee@kaist.ac.kr

Sumin Kim
KAIST
Daejeon, South Korea
gattonero@kaist.ac.kr

Sung-Hee Lee
KAIST
Daejeon, South Korea
sunghee.lee@kaist.ac.kr

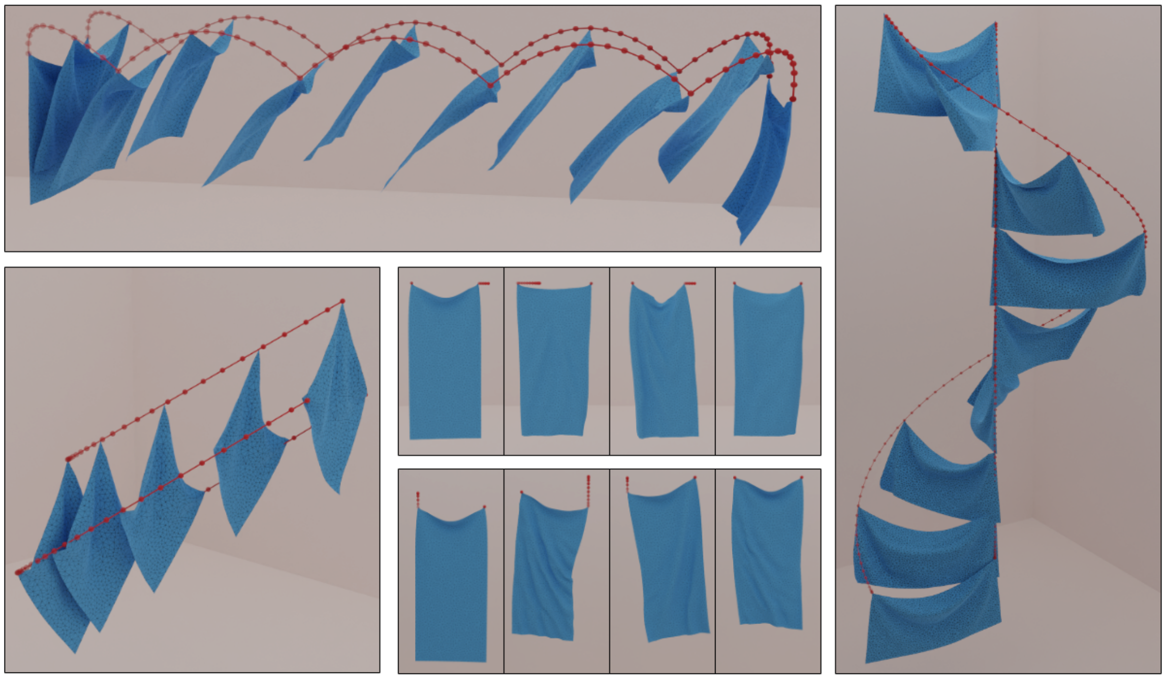

Figure 1: Results of our model running on different trajectories and topologies.

ABSTRACT

Deep learning-based cloth simulation approaches have potential in achieving real-time simulation of complex cloth by directly learning a mapping from control input to resulting cloth movement, bypassing the need for time-consuming dynamic solving and collision processing. Recent advancements have demonstrated the effectiveness of Graph Neural Networks (GNN) in learning cloth dynamics. However, existing GNN-based models have limitations in predicting scenarios involving complex cloth movement. To overcome this limitation, we propose a novel GNN-based model that incorporates several components, including RNN-based state encoding and physics-informed features. Our model significantly improves the accuracy of cloth dynamics prediction in various scenarios, including those with complex cloth movement driven by control handles. Furthermore, our model demonstrates generalization capabilities for cloth mesh topology and control handle configurations. We validate the effectiveness of our approach through ablation studies and comparisons with a baseline model.

CCS CONCEPTS

• **Computing methodologies → Physical simulation; Neural networks**.

KEYWORDS

cloth simulation, neural networks, data-driven simulation

ACM Reference Format:
Emmanuel Ian Libao, Myeongjin Lee, Sumin Kim, and Sung-Hee Lee. 2023. MeshGraphNetRP: Improving Generalization of GNN-based Cloth Simulation. In *ACM SIGGRAPH Conference on Motion, Interaction and Games (MIG '23), November 15–17, 2023, Rennes, France*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3623264.3624441

1 INTRODUCTION

Cloth simulation has been widely used in various industries, such as video games, VR, animation, and fashion. As such, cloth simulation has been an important issue in computer graphics research for decades, but achieving realistic simulation of complex cloth in real-time still remains challenging.

The most common method to generate realistic cloth deformation is physics-based simulation. While physics-based methods can yield stable and high-quality cloth, they require substantial computational resources, making it challenging to interactively use them for complex scenes. To address this issue, researchers have developed data-driven methods, which reproduce results efficiently for the cloth learned during the training phase [3, 5, 9, 12, 15, 18, 25]. Many of these methods deal with garments, rather than cloth, primarily focusing on the deformation caused by the body movement and not considering various dynamics that occur in cloth beyond the deformation caused by the body which makes the method limited to tight-fitting garments.

Recently, Graph Neural Networks (GNN) have been adopted to learn dynamics of physical systems without being constrained by the discretization topology of materials. MeshGraphNets [19] have shown the capacity of GNN to predict dynamics in a broad range of physical systems, including cloth, structural mechanics, and aerodynamics. However, since they targeted on broadly-defined dynamics, when the cloth is in control with a little more complexity, such as when the cloth moves and rotates instead of being fixed, it does not predict the dynamics well.

To address this, we introduce a novel GNN-based model that has achieved improved learning of cloth dynamics. Starting from MeshGraphNets as our baseline, we propose a number of components, including RNN-based state encoding as well as physics-based features and loss terms, to tailor the model more specifically to cloth and extend the generalization capability of cloth movement. Hence we dub our framework *MeshGraphNetRP* (i.e., with Recurrent encoding and Physical features).

The contributions of this paper can be summarized as follows:

- The introduction of physics-inspired features significantly improves the accuracy and robustness of cloth simulation for unseen challenging movements including rotation and translation at different directions and speeds.
- We show that our RNN-based state encoding is critical for reproducing the characteristic oscillatory behavior of cloth before reaching the equilibrium.
- Our model can be generalized for arbitrary cloth topology that has not been seen in training.

In addition, the paper introduces other components that contributed to enhancing the accuracy and stability of cloth simulation such as scheduled sampling and rotation-invariance with local coordinates. Shown in Fig. 1 are the results of our model running on different topologies such as square, wide, long and diamond-shaped cloths moving on different trajectories like bouncing trajectory, rotating upwards, and cloth handles moving separately from each other.

2 RELATED WORK

Physics-based simulation is the most common way to create realistic cloth in computer graphics. Since the seminal work of [23], various physics-based approaches have been introduced that employ physical laws to generate accurate cloth deformation[7, 16, 26, 27, 29]. Simulating cloth involves complex computations for vertex interactions that takes huge computational time.

To address this challenge, numerous learning-based approaches [3, 5, 14, 18] have been developed. Their main emphasis lies on capturing the effects of body movement on the garments rather than the dynamics occurring beyond body-induced deformations. Some approaches represent garments using parametric body models that use skinning weights on skeletons [14] or displacements from the body [3, 18]. However, with their body-centric representations, these methods are inherently limited to handling tight-fitting garments such as T-shirts and pants.

Other techniques have also been proposed to address loose-fitting garments with complex deformations that do not closely follow the body [10, 17, 28, 30, 31]. However, these methods may not be robust when the garment undergoes highly dynamic scenarios and may fail to produce stable deformations with high details.

Another avenue of research has aimed to predict physically plausible and realistic cloth dynamics by incorporating physics-inspired loss functions[4, 6, 9, 20]. The loss functions encompass stretching, bending, gravity, and body-cloth collision potentials to minimize. However, these methods are not fully exploiting the dynamics that arise from the physical interaction between the cloth vertices in the deformation procedure. Taking inspiration from these prior approaches, we include physical notions in our framework to maintain the quality and stability of cloth deformation.

Meanwhile, MeshGraphNets [22] employed Graph Neural Networks (GNN) [21] and obtained plausible dynamic simulation results in a broad range of physical systems including cloth, through message passing between vertices. MeshGraphNets demonstrated its capability to accurately predict the overall dynamics caused by various factors like wind effects or simple collisions on a stationary cloth. However, when the cloth undergoes movements and rotation instead of being stationary, the model was unable to output a stable result. Since the advent of MeshGraphNets, there has been a noticeable rise in the adoption of Graph Neural Networks (GNN) for cloth simulations [4, 8, 13, 24, 30].

We build our model based on prior research demonstrating the efficacy of graph neural networks (GNNs) and physics-informed features for dynamic cloth simulation. Our model features a GNN framework that enhances the generalizability to a range of mesh topologies and motion. By incorporating physics-informed features and leveraging the relational nature that occurs at a vertex level in a cloth enhanced the model its ability to infer the true dynamics of the cloth. Our approach has achieved unprecedented generalization on dynamic cloth motions driven by constraint handles.

3 OVERVIEW

We develop a GNN-based framework for cloth simulation of planar topologies that is stable for translation and rotation motions. An overview of our framework is shown in Fig. 2. We first collect a dataset of the trajectories of a square cloth by using a physical cloth

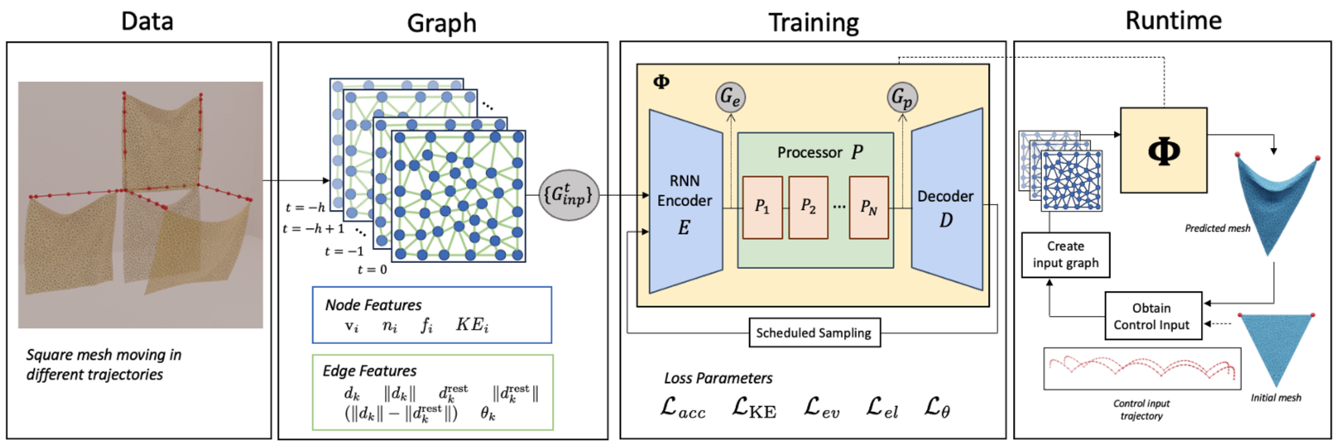

Figure 2: The overall framework of our model. We gather data using a physical cloth simulator and use these to make a graph with rich physical features for our GNN-based model. We used scheduled sampling and physics-based loss. At runtime, we would only need the trajectory of the control handles that could go anywhere in 3D space and the initial cloth mesh

simulator. From the dataset we train the model by constructing a graph that includes relevant physical information of the current mesh as node features and edge features. We use the Encode-Process-Decode of MeshGraphNets as our main architecture. We employ scheduled sampling in our training scheme to make the rollout results stable. After training, we can run the model by inputting the initial state of the mesh and the trajectory of the handles (i.e., constraints on cloth nodes). The trained model can be generalized for motions in unseen directions, speed and topology.

Dataset Preparation. We use Arcsim [16] as the cloth simulator for obtaining our dataset. We simulated an irregular square mesh with 1024 vertices driven by two handles on the upper left and upper right corners of the cloth. Our training dataset includes 16 trajectories containing 14 trajectories for translation and 2 trajectories for rotation run at 60 fps. These trajectories are split into 42 sections with a total of 6550 frames. Aside from the garment position, we also save the mass of each cloth vertex and area of each face. We also save the default rest state of the cloth. These data will be used for computing the features in graph representation of the input which will be detailed in Sec. 5. Appendix **??** details how we obtained our training dataset.

4 ENCODE-PROCESS-DECODE ARCHITECTURE

We follow the GNN-based Encode-Process-Decode architecture (Fig. 2) introduced in [1]. Please refer to the paper for more details and variations of this architecture. The first part of the network is an encoder E that encodes the input graph G_{inp} or in our case a sequence of input graph $\{G^t_{inp}\}$ into a latent representation G_e. Message passing is applied in the processor P that contains N sequential P_j $(j = \cdots N)$ blocks to get G_p. Finally the decoder D extracts the desired dynamic information from G_p. Like MeshGraphNets, the final output of the graph is the acceleration of each node, which is numerically integrated to get the position for next time step.

Input Graph. We use a cloth M containing v vertices and e edges. At time t we represent the cloth as M^t with positions $\{p^t_i\}_{i=1:v}$. From the cloth states M^t and M^{t-1} at current and previous time steps, we extract relevant node features and edge features to make a graph $G_{inp} = (V, E)$ with V containing the node features $\{v_i\}_{i=1:v}$ and E containing the edge set $\{(e_k, r_k, s_k)\}_{k=1:e}$ where e_k contains the edge features, and r_k and s_k are the indices of receiver and sender nodes representing the connectivity of the edges.

Encoder. We modified the MLP-based encoder of the original MeshGraphNets to RNN architecture with GRU to consider the historical movement of the cloth. The RNN encoder takes in a sequence of input graph $\{G^t_{inp}\}(t = -h, -h+1, \cdots, -1, 0)$ with $t = 0$ representing the current time step. For our final model we use $h = 9$ previous time steps for a total of 10 input sequence.

Processor. The processor has $P_j(j = 1, \cdots, N)$ blocks for nodes and edges to pass their information to the local neighborhoods within N steps. This message-passing among nodes and edges on graphs is an important part for learning the dynamics. For our model we used $N = 15$ as our message-passing steps.

Decoder. The decoder takes in the final graph output of the processor, G_p, and decodes it to be the predicted acceleration for each node a_i. It contains an MLP node model ϵ^V and gets the node features from G_p. We integrate a_i twice to get the next time step position $p^{t+1}_i = a_i + 2p^t_i - p^{t-1}_i$.

5 MODEL FRAMEWORK

As discussed in previous section, we represent cloth as a graph containing node features and edge features to describe its current state. We introduce several physics-based features and losses in our model to enable a rich understanding on the current physical state of the cloth for the model to learn cloth dynamics properly. We also employed scheduled sampling and local transformations in our model training to make the results stable throughout the whole rollout.

5.1 Node Features

We will first discuss the node features of the baseline MeshGraphNets, velocity $\mathbf{v}_i$ and node type n_i, and then discuss new node features we introduce in the model: external force f_i and kinetic energy KE_i.

Velocity. The velocity at time t is obtained for each node by subtracting the positions of current mesh M^t and the previous mesh M^{t-1}.

$$\mathbf{v}_i^t = p_i^t - p_i^{t-1}$$

Node Type. The nodes belonging to the handles serve as the control input that drives the remaining nodes to motion. As such we differentiate the handle nodes from the other nodes with the node type n_i represented with one-hot encoding.

External Force. The external force f_i on each node is a 3D vector that is computed at each time step. In this work, f_i accounts for the gravity and the wind drag. We used the saved values of the mass of the vertices and area of the faces to compute the external force.

Kinetic Energy. The kinetic energy at time t is obtained as $KE_i^t = \frac{1}{2}m_i||\mathbf{v}_i^t||^2$, where m_i is the node's mass.

5.2 Edge Features

We will first discuss the existing edge features from MeshGraphNets, edge 3D vector and length, and then discuss the edge features we introduce in our model: rest state edge 3D vector and length, stretch and bending.

Edge 3D vector and length. We obtain the 3D vector of an edge k by subtracting the current positions of the sender s_k and receiver r_k nodes of the edge, $d_k = p_{s_k}^t - p_{r_k}^t$, and its L^2 norm as the edge length, $||d_k||$.

Rest state edge 3D vector and length. To represent the rest state of the cloth, we include the edge 3D vector, d_k^{rest}, and its length, $||d_k^{\text{rest}}||$, at rest state as an edge feature. The rest state of cloth is obtained from physics simulation when the cloth is sagged with static handle positions. The difference of this feature from MeshGraphNets is that we use the 3D rest state of the cloth while the baseline method used 2D UV-coordinates for this feature.

Stretch. Stretching force is a major factor for cloth dynamics that acts to preserve the original length of the cloth. We simply model the stretch feature of an edge as the difference of the current edge length from its rest length, $||d_k|| - ||d_k^{\text{rest}}||$.

Bending. To help the model learn the cloth dynamics that generates bending force to restore the original angle between adjacent faces, we add bending as an edge feature. The bending angle θ_k of an edge k is 0 when two adjacent faces are flat, and π when two faces collapse on each other. Bending is set to zero when an edge is connected to only one face.

5.3 Training Objectives

Our model is trained by minimizing the following loss functions that include physics-based loss terms. We will first discuss the loss of the baseline MeshGraphNets, acceleration loss, and then discuss the new loss terms we introduce in the model: kinetic energy, edge vector, edge length, and bending loss.

Acceleration Loss. The first term is the acceleration loss that measures the difference between the target acceleration a_i^t and the predicted acceleration $\overline{a}_i$ of each node.

$$\mathcal{L}_{acc} = \frac{1}{v}\sum_{i=1}^{v}||a_i - \overline{a_i}||^2, \tag{1}$$

Kinetic Energy Loss. We compute the target kinetic energy KE using the target position and ground truth current position to get the velocity. The predicted kinetic energy $\overline{KE}$ is obtained from the predicted and current positions.

$$\mathcal{L}_{KE} = \frac{1}{v}\sum_{i=1}^{v}(KE_i - \overline{KE_i})^2 \tag{2}$$

Edge Vector Loss. Using the target position and predicted position we compute the target and predicted edge 3D vectors to compute the edge vector loss as:

$$\mathcal{L}_{ev} = \frac{1}{e}\sum_{k=1}^{e}||d_k^{t+1} - \overline{d_k}^{t+1}||^2 \tag{3}$$

Edge Length Loss. We compute the norm of the target and predicted relative cloth position to get the length of an edge. We use these edge length values to compute the loss as:

$$\mathcal{L}_{el} = \frac{1}{e}\sum_{k=1}^{e}(||d_k^{t+1}|| - ||\overline{d_k}^{t+1}||)^2 \tag{4}$$

Bending Loss. We compute the target and predicted bending using the target and predicted positions. We compute the loss as:

$$\mathcal{L}_{\theta} = \frac{1}{e}\sum_{k=1}^{e}(\theta_k - \overline{\theta_k})^2 \tag{5}$$

The total loss function is the weighted sum of the above terms:

$$\mathcal{L}_{total} = \mathcal{L}_{acc} + \lambda_{KE}\mathcal{L}_{KE} + \lambda_{ev}\mathcal{L}_{ev} + \lambda_{el}\mathcal{L}_{el} + \lambda_{\theta}\mathcal{L}_{\theta} \tag{6}$$

We empirically set the loss weights λ_{KE}, λ_{ev}, λ_{el}, λ_{θ} to 1, 30, 30 and 0.5 to get the optimal learned model.

5.4 Model Training

At each timestep t in the training data trajectory we get source triplet positions $X_t = (x_{-1}, x_0, x_1)$ where x_{-1} corresponds to previous position, x_0 to the current position, and x_1 to the target position. To make our framework rotation-invariant, we transform the positions X_t at time t to the local coordinate frame of x_0. The local coordinate frame is determined from the positions of the handles. As such, a planar cloth with its normal facing in the x-axis direction would be seen the same way by the model even if this cloth is rotated along the vertical axis. Adding noise in the training data is important in reducing rollout errors, which was done by adding Gaussian noise with zero mean and fixed variance to both x_1 and x_0. We use the same training noise strategy as in GNS [11] and set the noise magnitude hyperparameter to 0.3, which worked well for our model.

To gently introduce rollout during training, we adapted scheduled sampling scheme [2], which was critical for successful training of our model. With a probability **p** we performed rollout for the next frame of data, i.e., the predicted position was used for the next time step. If **p** = 1 for an epoch, the training is performed without any rollout, and if **p** = 0, the training is conducted in a purely auto-regressive manner where the predicted output will be the input for the next time step prediction. In our training, we set **p** = 1 for the first set of epochs and linearly decreased **p** until **p** = 0. In addition, we set the maximum number of rollout steps to 30, after which the ground truth values for the previous and current time steps replaced the predicted values to prevent excessive deviation of the rollout simulation from the training data. Refer to Appendix B in our supplementary file for details on the scheduled sampling and the model configuration.

6 RESULTS AND EVALUATION

We evaluate the effectiveness of our model through a comparison with MeshGraphNets and an ablation study. The results show that our model produces stable results and can move in any direction in 3D space even though the model is trained with only a sample of directions. In contrast, MeshGraphNets fails to generate stable rollouts. Furthermore, our model exhibits the ability to generalize to new motion and mesh topologies (Fig. 1). Please refer to the supplementary video for viewing our results.s

6.1 Comparison with MeshGraphNets

We compared our method with MeshGraphNets on our test dataset containing trajectories of a square cloth moving in directions not seen on training (Fig. 3).

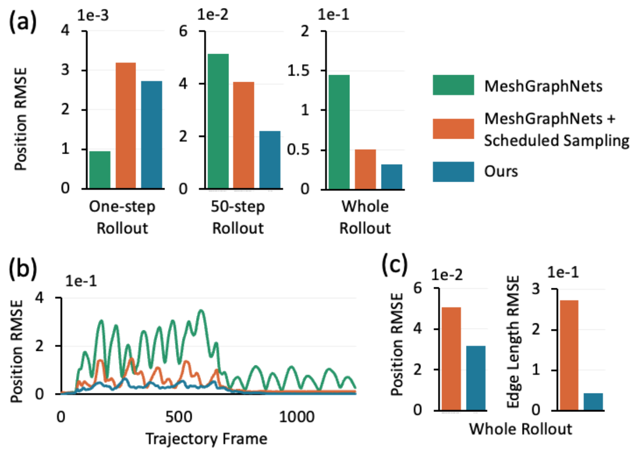

Figure 3: Evaluation of our model compared with MeshGraphNets. (a) Position RMSE at different rollout steps (b) Position RMSE at each frame on a test trajectory (c) A closer comparison between our model and MeshGraphNets with scheduled sampling

For one-step rollouts, baseline MeshGraphNets works better than our model but MeshGraphNets fails to maintain stable results with a rapid increase in RMSE as the rollout steps increase. At whole rollouts, the RMSE of baseline MeshGraphNets is significantly higher than our model. It could also be seen in Fig. 3b how the results of baseline MeshGraphNets destabilizes in the middle of the rollout where the cloth is in motion.

We applied scheduled sampling on the baseline MeshGraphNets as one of our comparisons in the evaluation. With scheduled sampling the results of the model become more stable. We see at Fig. 3c a closer comparison between this model and our model. Position RMSE is lower in our model and we could also see a clearer difference at the edge length RMSE which we observed is a sensitive metric for measuring stability.

We observe that instabilities commonly arise from the areas of the cloth near the handle wherein the neighboring nodes of the handle cannot keep up with the cloth in motion. Figure 4 shows the result of the models on a test dataset at the same frame.

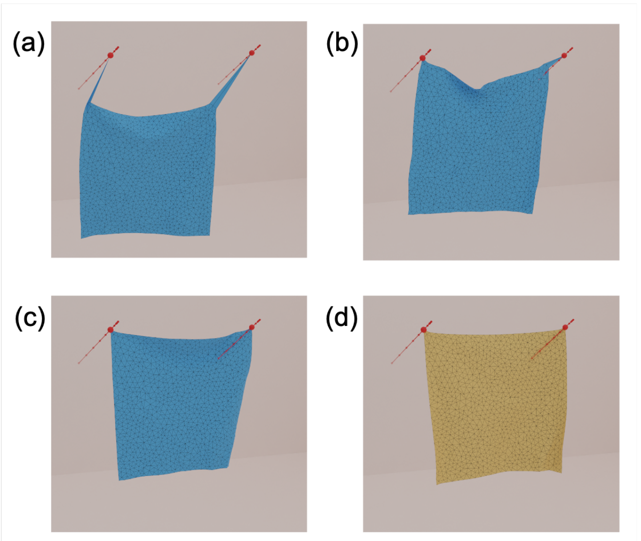

Figure 4: Frame result for a test trajectory moving in (0,1,-1) segment. (a) Baseline MeshGraphNets (b) MeshGraphNets + scheduled sampling (c) Our model (d) Ground truth.

6.2 Ablation Study

We first perform ablations studies by looking at the effect of the RNN encoder on the model. Our final model uses an RNN encoder that takes in an input sequence of length 10. We compare it with a model that uses a simple MLP encoder and a model that uses an RNN encoder of length 5. Shown in Fig. 5 are the results. Two different epochs were compared to also show the effect of scheduled sampling with the encoders. At 300th epoch, the model starts to operate in full auto-regressive manner.

The model that uses MLP encoder actually begin to worsen its performance starting from the 300th epoch. It fails to learn the important relationship of the previous inputs to the output acceleration of the cloth.

Increasing the input sequence length to the RNN improved the overall performance of the model. From the figure, we see that at 300th epoch the performance of the RNN encoders are similar with the MLP encoder. But with the next epochs of auto-regression

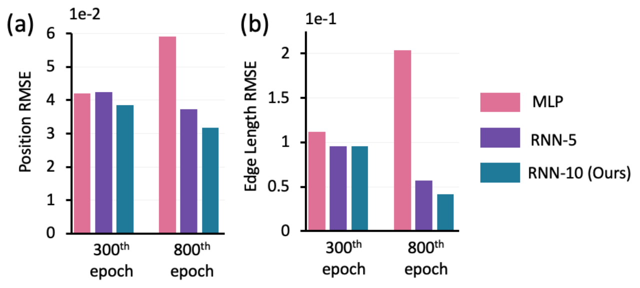

Figure 5: Comparison of the MLP and RNN encoders. (a) Position RMSE (b) Edge Length RMSE.

Table 1: Features and Loss Ablations.

Model	Position RMSE $\times 10^{-2}$	Edge Length RMSE $\times 10^{-2}$
w/o external force feature	3.1	4.463
w/o 3d rest state feature (use 2d rest state)	3.321	4.361
w/o stretch feature	3.373	4.685
w/o edge vector loss	3.289	5.02
w/o edge length loss	3.329	6.508
w/o KE feature & loss	3.22	4.169
w/o bending feature & loss	3.255	4.51
w/o all added features and loss (external force, 3d rest state, stretch, KE and bending features, and edge vector, edge length, KE and bending losses)	3.365	10.5
Ours	3.167	4.197

training, the RNN encoders continued to improve with the RNN-10 having more improvement than RNN-5.

Next are the ablation studies done by removing the physical features and loss we introduced in our model one at a time and evaluating both position RMSE and edge length RMSE. Shown in Table 1 are the results wherein our model performs better than the models without the physical features and loss. Especially we can see from the table the huge gap between the edge length RMSE of the model without the new features and loss we introduced and our model. Looking at the rollout results of this model without the new features and loss, the elongation near the handles is apparent as well as the shaking when the cloth is supposedly in rest state.

6.3 Runtime Performance

Our model with an RNN sequence length of 10 demonstrates efficient performance achieving processing time of 37 ms per frame on an RTX 3080 for 1024-vertices mesh. Reducing the RNN sequence length to 5 results in a slightly better performance of 26 ms per frame. The original MeshGraphNets configuration which showed unstable rollouts for most motions runs at 12 ms per frame. In contrast, Arcsim, which exclusively operates on CPU and was evaluated on Ryzen 5 5600x, exhibits a signifiantly slower performance of 404 ms per frame.

6.4 Generalization and Limitations

Our method generalizes well beyond the motion trajectories seen by training in regards of direction of translation, curvature of rotation, and motion speed producing stable results throughout the rollout. We could also run motions where the cloth translates and rotates at the same time. We believe this is achieved because of the physics-based features we introduce in our model.

In addition, our method could simulate cloth with other mesh topology with different mesh shape and size or different number of handles. This generalizability is a strength of using GNN as our main architecture.

Our model can generalize to motion speeds less than that of the speed of training trajectories however our model fails on trajectories with higher speeds. The rollout becomes unstable as the cloth cannot catch up with the movement of the handles although the cloth could recover to its stable result after the handles stop.

Our model can also generalize with handles moving separately from each other as long as the distance between the handles is kept at a certain range. Otherwise, when the handles are too far from each other, it will fail to show the stretched rest state. In addition, if the handles are too close, self-collision between the cloth vertices may occur.

One limitation in our work is the assumption of no collisions with external objects. We leave future work for representing collision for the model. One possible way is through the solution of MeshGraphNets with the representation of the external object as another graph and making connecting edges between the cloth graph when the external object is close to the cloth which could be computationally inefficient. Another limitation of our work is the assumption that the rest state of a given cloth is the same. This is not the case when we have the four corners of a square cloth as control handle and manipulate the bottom corners to bring the cloth upwards. We believe that a proper representation of the rest state of the cloth is essential in the model and future work is left for better representation of this rest state.

7 CONCLUSION

We propose a novel GNN-based cloth simulation framework that leverages physics-informed features in the networks. Our framework accurately predicts cloth dynamics in highly generalized motions that are not seen during training, while also generalizing for arbitrary cloth mesh topology and control handle.

Our results demonstrate the effectiveness of deep learning approaches in reproducing cloth motions for a wide range of scenarios. However, we acknowledge that more complex cloth motions, such as tearing or colliding with complex objects, may present a challenge for deep learning-based methods. Exploring these challenging scenarios would contribute to advancing both deep learning architectures and cloth simulation studies, ultimately leading to more robust and versatile cloth simulation frameworks.

ACKNOWLEDGMENTS

This work was supported by NCSoft and National Research Council of Science & Technology, Korea [CRC 21011].

REFERENCES

[1] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria

Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. 2018. Relational inductive biases, deep learning, and graph networks. arXiv:1806.01261 [cs.LG]

[2] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks *(NIPS'15)*. MIT Press, Cambridge, MA, USA, 1171–1179.

[3] Hugo Bertiche, Meysam Madadi, and Sergio Escalera. 2020. CLOTH3D: Clothed 3D Humans. arXiv:1912.02792 [cs.CV]

[4] Hugo Bertiche, Meysam Madadi, and Sergio Escalera. 2020. PBNS: physically based neural simulator for unsupervised garment pose space deformation. *arXiv preprint arXiv:2012.11310* (2020).

[5] Hugo Bertiche, Meysam Madadi, and Sergio Escalera. 2021. PBNS: Physically Based Neural Simulator for Unsupervised Garment Pose Space Deformation. arXiv:2012.11310 [cs.CV]

[6] Hugo Bertiche, Meysam Madadi, and Sergio Escalera. 2022. Neural Cloth Simulation. *ACM Transactions on Graphics* 41, 6 (nov 2022), 1–14. https://doi.org/10.1145/3550454.3555491

[7] Robert Bridson, Sebastian Marino, and Ronald Fedkiw. 2005. Simulation of clothing with folds and wrinkles. In *ACM SIGGRAPH 2005 Courses*. 3–es.

[8] Artur Grigorev, Bernhard Thomaszewski, Michael J Black, and Otmar Hilliges. 2022. HOOD: Hierarchical Graphs for Generalized Modelling of Clothing Dynamics. *arXiv preprint arXiv:2212.07242* (2022).

[9] Erhan Gundogdu, Victor Constantin, Amrollah Seifoddini, Minh Dang, Mathieu Salzmann, and Pascal Fua. 2019. Garnet: A two-stream network for fast and accurate 3d cloth draping. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 8739–8748.

[10] Doyub Kim, Woojong Koh, Rahul Narain, Kayvon Fatahalian, Adrien Treuille, and James F. O'Brien. 2013. Near-Exhaustive Precomputation of Secondary Cloth Effects. *ACM Trans. Graph.* 32, 4, Article 87 (jul 2013), 8 pages. https://doi.org/10.1145/2461912.2462020

[11] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. arXiv:1609.02907 [cs.LG]

[12] Zorah Lahner, Daniel Cremers, and Tony Tung. 2018. Deepwrinkles: Accurate and realistic clothing modeling. In *Proceedings of the European conference on computer vision (ECCV)*. 667–684.

[13] Alberta Longhini, Marco Moletta, Alfredo Reichlin, Michael C Welle, David Held, Zackory Erickson, and Danica Kragic. 2022. EDO-Net: Learning Elastic Properties of Deformable Objects from Graph Dynamics. *arXiv preprint arXiv:2209.08996* (2022).

[14] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. 2015. SMPL: A Skinned Multi-Person Linear Model. *ACM Trans. Graph.* 34, 6, Article 248 (nov 2015), 16 pages. https://doi.org/10.1145/2816795.2818013

[15] E. Miguel, D. Bradley, B. Thomaszewski, B. Bickel, W. Matusik, M. A. Otaduy, and S. Marschner. 2012. Data-Driven Estimation of Cloth Simulation Models. *Comput. Graph. Forum* 31, 2pt2 (may 2012), 519–528.

[16] Rahul Narain, Armin Samii, and James F O'brien. 2012. Adaptive anisotropic remeshing for cloth simulation. *ACM transactions on graphics (TOG)* 31, 6 (2012), 1–10.

[17] Xiaoyu Pan, Jiaming Mai, Xinwei Jiang, Dongxue Tang, Jingxiang Li, Tianjia Shao, Kun Zhou, Xiaogang Jin, and Dinesh Manocha. 2022. Predicting Loose-Fitting Garment Deformations Using Bone-Driven Motion Networks. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Proceedings*. ACM. https://doi.org/10.1145/3528233.3530709

[18] Chaitanya Patel, Zhouyingcheng Liao, and Gerard Pons-Moll. 2020. TailorNet: Predicting Clothing in 3D as a Function of Human Pose, Shape and Garment Style. arXiv:2003.04583 [cs.CV]

[19] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. 2021. Learning Mesh-Based Simulation with Graph Networks. arXiv:2010.03409 [cs.LG]

[20] Igor Santesteban, Miguel A. Otaduy, and Dan Casas. 2022. SNUG: Self-Supervised Neural Dynamic Garments. arXiv:2204.02219 [cs.CV]

[21] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE transactions on neural networks* 20, 1 (2008), 61–80.

[22] An Ping Song, Xin Yi Di, Xiao Kang Xu, and Zi Heng Song. 2020. MeshGraphNet: An effective 3D polygon mesh recognition With topology reconstruction. *IEEE Access* 8 (2020), 205181–205189.

[23] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. 1987. Elastically Deformable Models. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '87)*. Association for Computing Machinery, New York, NY, USA, 205–214. https://doi.org/10.1145/37401.37427

[24] Lokender Tiwari and Brojeshwar Bhowmick. 2023. GarSim: Particle Based Neural Garment Simulator. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 4472–4481.

[25] Yunjin Tong, Shiying Xiong, Xingzhe He, Guanghan Pan, and Bo Zhu. 2021. Symplectic neural networks in Taylor series form for Hamiltonian systems. *J. Comput. Phys.* 437 (2021), 110325.

[26] Pascal Volino, Nadia Magnenat-Thalmann, and Francois Faure. 2009. A simple approach to nonlinear tensile stiffness for accurate cloth simulation. *ACM Transactions on Graphics* 28, 4 (2009), Article–No.

[27] Huamin Wang, James F O'Brien, and Ravi Ramamoorthi. 2011. Data-driven elastic models for cloth: modeling and measurement. *ACM transactions on graphics (TOG)* 30, 4 (2011), 1–12.

[28] Tuanfeng Y. Wang, Tianjia Shao, Kai Fu, and Niloy J. Mitra. 2019. Learning an Intrinsic Garment Space for Interactive Authoring of Garment Animation. *ACM Trans. Graph.* 38, 6, Article 220 (nov 2019), 12 pages. https://doi.org/10.1145/3355089.3356512

[29] Martin Wicke, Daniel Ritchie, Bryan M Klingner, Sebastian Burke, Jonathan R Shewchuk, and James F O'Brien. 2010. Dynamic local remeshing for elastoplastic simulation. *ACM Transactions on graphics (TOG)* 29, 4 (2010), 1–11.

[30] Meng Zhang, Duygu Ceylan, and Niloy J Mitra. 2022. Motion guided deep dynamic 3d garments. *ACM Transactions on Graphics (TOG)* 41, 6 (2022), 1–12.

[31] Meng Zhang, Tuanfeng Wang, Duygu Ceylan, and Niloy J. Mitra. 2021. Deep Detail Enhancement for Any Garment. *Computer Graphics Forum* 40, 2 (2021), 399–411. https://doi.org/10.1111/cgf.142642 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.142642