ClothTransformer: Unified Latent-Space Transformers for Scalable Cloth Simulation

Yu Zhang[1] **Yidi Shao**[2] **Wenqi Ouyang**[1] **Yushi Lan**[3] **Zhexin Liang**[1]
Chengrui Wu[4] **Xudong Xu**[5] **Xingang Pan**[1]
[1]S-Lab, Nanyang Technological University [2]Feeling AI
[3]University of Oxford [4]Nanyang Technological University [5]Shanghai AI Laboratory

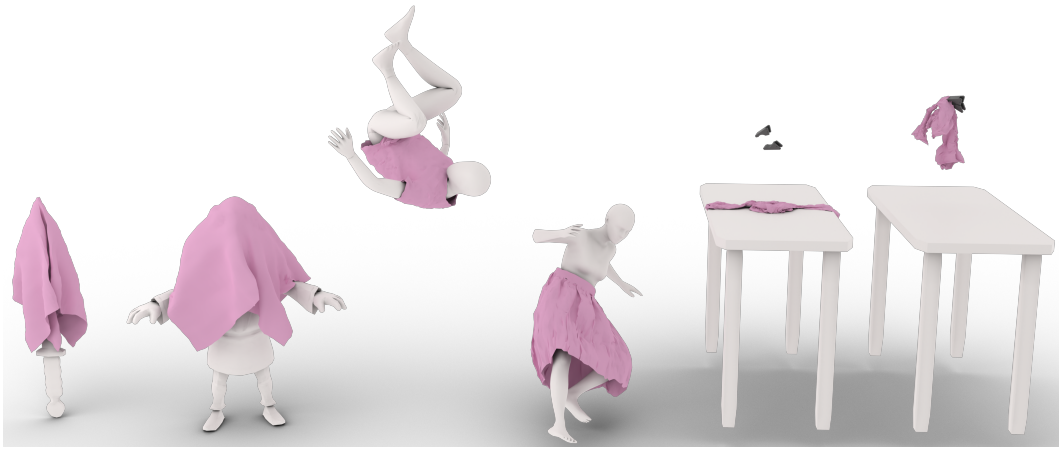

Figure 1: **ClothTransformer** generalizes to unseen test cases across three diverse scenarios. *Left two:* Diverse Object Collision—cloth falling onto unseen rigid objects (sword, character). *Middle two:* Human Garment—unseen body, garment, and animation combinations (front-flip, dancing). *Right two:* Robotic Manipulation—unseen cloth meshes grasped and lifted by a robotic gripper.

Abstract

Unified and scalable Transformers have recently achieved remarkable success in modeling diverse phenomena traditionally associated with computer graphics, such as 3D visual effects, rendering processes, and motion in videos. In this work, we take a step further by investigating whether modern Transformer techniques can tackle the challenging task of cloth simulation. To this end, we present ClothTransformer, a framework that reformulates cloth simulation as autoregressive sequence modeling in a learned latent space. Existing neural cloth simulators are largely specialized to single scenarios, intrinsically coupled to the mesh discretization, and lack robust collision handling. Our approach addresses these limitations through three contributions: (1) a unified Transformer architecture that handles diverse scenarios—body-driven garments, robotic manipulation, and free-fall collisions—under a single model and achieves approximately 4–9$\times$ lower error than prior state-of-the-art across all scenarios; (2) a scalable latent-space formulation that compresses arbitrary-resolution meshes into a fixed-size set of latent tokens, making temporal dynamics computation independent of mesh resolution; and (3) a diverse-scenario high-fidelity penetration-free dataset of $\sim$493.4k frames spanning all three settings, which enables a differentiable Continuous Collision Detection (CCD) module to suppress penetration artifacts. Project Page: https://yucrazing.github.io/clothtransformer/

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

Realistic cloth simulation is essential for a wide range of applications. In film and visual effects, convincing fabric motion brings digital characters to life; in gaming and virtual reality, interactive garments are key to immersion; and in embodied AI, the recent rapid development further intensifies the demand for efficient and physically plausible simulation. Despite decades of progress, however, simultaneously achieving high fidelity and real-time performance remains challenging. Physically Based Simulation (PBS) methods [1], including advanced variational contact solvers such as IPC [19], can produce highly accurate results; yet even with modern GPU acceleration [16], high-resolution cloth can still take tens of seconds per frame—far beyond real-time budgets.

Learning-based neural simulators offer a promising alternative. Most recent progress is driven by Graph Neural Networks (GNNs) [32, 10], which predict vertex dynamics via message passing on mesh edges. Still, existing approaches face three fundamental limitations:

Lack of Generalization. Existing learning-based cloth simulators are largely specialized to a single setting—typically human-garment dressing on an animated body—or require training a separate model for each scenario. In both cases, they lack a single unified architecture and model capable of handling diverse scenarios such as robotic manipulation or free-fall collisions, which hinders their applicability to broader simulation tasks.

The Resolution Bottleneck. GNN simulators are tightly coupled to mesh discretization: inference cost grows with vertex/edge count. This creates a direct conflict between visual fidelity (dense meshes) and efficiency (fast, memory-light inference), undermining the core motivation of neural simulation.

The Penetration Problem. Almost all existing learning-based methods rely on Discrete Collision Detection (DCD) for collision handling during training, which only checks for intersections at discrete time steps and leads to the tunneling problem under fast motions (see Figure 5 in Appendix B for an illustration). Continuous Collision Detection (CCD) resolves this by sweeping the inter-frame trajectory, but demands high-quality penetration-free supervision that public datasets do not provide.

Motivated by the recent success of Transformers [41] with minimal inductive bias in vision and graphics tasks, we present **ClothTransformer**, a unified Transformer-based framework that reformulates cloth simulation as autoregressive sequence modeling in a learned latent space. Transformers have proven effective at capturing complex physical rules in computer graphics, such as 3D-to-2D projection [26] and rendering [44]; here, we study their potential in the more challenging cloth simulation task and address the aforementioned limitations with several key designs.

Our framework jointly addresses all three aforementioned limitations with several key designs. The Transformer's minimal inductive bias enables a single unified architecture that handles diverse scenarios—body-driven garments, robotic manipulation, and free-fall collisions—without per-scenario tuning (Figure 1). To overcome the resolution bottleneck, we compress the cloth state into a compact, fixed-size set of latent vectors via cross-attention and evolve dynamics entirely in latent space, making temporal computation effectively independent of mesh resolution. To suppress penetration artifacts, we construct a diverse-scenario penetration-free dataset spanning all three settings, which enables a differentiable CCD loss during training and CCD post-processing at inference. Our results highlight the strong potential of Transformer-based autoregressive models for learning-based physical simulation.

In summary, our contributions are:

- We propose **ClothTransformer**, a unified Transformer architecture that handles diverse cloth simulation scenarios—body-driven garments, robotic manipulation, and free-fall collisions—under a single model, achieving approximately 4–$9\times$ lower error than prior state-of-the-art methods across all scenarios.
- Our latent-space formulation compresses arbitrary-resolution meshes into a fixed-size set of latent tokens, making temporal dynamics computation efficient and independent of mesh resolution.
- We construct a diverse-scenario high-quality penetration-free dataset of $\sim$493.4k frames spanning body-driven garments, robotic manipulation, and free-fall collisions, which enables

a differentiable Continuous Collision Detection (CCD) loss with CCD post-processing to suppress penetration artifacts.

2 Related Work

2.1 Physics-Based Cloth Simulation

Cloth simulation has a long history in computer graphics. Early mass-spring models were intuitive but numerically stiff, requiring very small time steps. Baraff and Witkin [1] addressed this with implicit integration, enabling large stable steps. Finite Element Methods (FEM) further improved accuracy by treating cloth as a continuum with well-defined constitutive models for stretching, bending, and shearing.

A central challenge in traditional cloth simulation is collision handling. Bridson *et al*. [4] combined geometric intersection tests with repulsion forces, while IPC [19] unified contact into a variational framework guaranteeing intersection-free results. The current state-of-the-art solver GIPC [16] further accelerated IPC with GPU-based Gauss-Newton optimization, yet a single high-resolution frame can still take tens of seconds.

Real-time simulators [30, 29, 7] instead favor speed, operating under a much smaller per-frame time budget with simplified contact models at the cost of simulation quality. Moreover, these real-time simulators aim only for visually plausible results, which can differ substantially from those of high-fidelity simulators; hence, they cannot reproduce the behavior of high-fidelity simulators.

2.2 Learning-Based Cloth Simulation

Learning-based approaches have become increasingly popular for cloth simulation [31, 2, 37, 14, 28, 36, 3]. Their key advantage is that they can learn the behavior of high-fidelity simulators and serve as their fast surrogates. They are therefore orthogonal to traditional real-time simulators, whose fidelity is bounded by their own numerical approximations. Among them, GNN-based and Transformer-based methods are the two dominant paradigms.

Graph Neural Networks. Graph Neural Network (GNN)–based methods [32, 10, 11, 35, 25, 42, 5] are widely adopted for learning-based cloth simulation, representing vertices as graph nodes and propagating information along mesh edges. The current state-of-the-art GNN backbone for cloth simulation, adopted by both HOOD [10] and ContourCraft [11], introduces hierarchical message passing on a multi-resolution mesh graph, enabling faster long-range information flow across the garment.

Transformers. Transformer-based models have also been applied to physics simulation [38, 39, 20, 15, 21–23, 13, 8, 6]. LayersNet [39] groups mesh vertices into UV-derived patch tokens to reduce token count, but the fixed UV boundaries can cause spatial discontinuities and mesh collapse. Manifold-aware Transformer [20] tokenizes the garment at the mesh-face level and modulates self-attention with local mesh connectivity to predict per-frame deformation gradients, yet its per-face tokenization keeps the attention cost coupled to the mesh resolution.

Limitations of Existing Paradigms. Existing GNN- and Transformer-based methods share two fundamental limitations: (i) they are developed and evaluated solely within the human-garment dressing setting, with no demonstrated capability on broader scenarios such as robotic manipulation and free-fall collisions; and (ii) their inference cost remains coupled to the input mesh resolution, whether through edge-message-passing in GNNs or UV-patch / per-face tokenization in Transformers. Although data-driven compression has been explored in other domains [45, 17], no such approach has yet been developed for cloth simulation. In contrast, our method learns a *data-driven* compression through cross-attention with learnable queries, reducing the dynamics complexity to $O(N_{\text{latents}}^2)$ independent of the input mesh size and avoiding scenario-specific structural priors, thereby supporting diverse cloth simulation scenarios under a single unified architecture.

Collision Handling in Neural Cloth Simulation. Most neural cloth simulators rely on DCD [32, 10, 39, 33, 34] or post-hoc penalty forces, both prone to tunneling under fast motions. Alternative strategies include repulsion units [40] and auxiliary self-collision graphs [24]. ContourCraft [11] uses DCD to detect intersecting triangle pairs, groups them into intersection contours, and learns to resolve multi-garment interpenetrations as a post-processing step. However, none integrates CCD

into the training loop, leaving them vulnerable to tunneling. Moreover, CCD-based training requires high-quality penetration-free ground truth; if the training data itself contains residual intersections, CCD gradients become contradictory, yet most existing datasets are generated by solvers that tolerate such artifacts. In contrast, we construct a high-fidelity penetration-free dataset with strict intersection-free guarantees, enabling a differentiable CCD loss that penalizes trajectory-level intersections during training.

3 Methodology

To handle the high dimensionality and varying topology of cloth meshes, we propose a **Cloth-Transformer** architecture. This framework compresses the geometric and dynamic state into a compact latent representation, modeling the temporal dynamics in this latent space, and subsequently reconstructing the mesh.

3.1 Problem Formulation

We formulate cloth simulation as an autoregressive sequence modeling task. Let a cloth mesh be represented as $\mathcal{M} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the set of N_v vertices and $\mathcal{E}$ the set of edges. To fully capture the physical state of the system, we define the state at time step t using both vertex positions $\mathbf{X}_t \in \mathbb{R}^{N_v \times 3}$ and their instantaneous velocities $\mathbf{V}_t \in \mathbb{R}^{N_v \times 3}$. Formally, the input is the current cloth state $\mathcal{S}_t = \{\mathbf{X}_t, \mathbf{V}_t\}$. The system is conditioned on the collision environment, represented by the collision object mesh $\mathbf{C}_{t+1}$ at the target frame. Our goal is to learn a mapping function F_θ parameterized by a neural network that predicts the future position state:

$$\hat{\mathbf{X}}_{t+1} = F_\theta(\mathbf{X}_t, \mathbf{V}_t, \mathbf{C}_{t+1} \mid \mathbf{X}_{rest}) \tag{1}$$

where $\mathbf{X}_{rest}$ represents the rest shape of the cloth.

3.2 Architecture Overview

As illustrated in Figure 2, the framework consists of three primary components: (1) a Spatial Encoder that compresses the geometry and dynamics of the cloth and collision objects into latent tokens; (2) a Temporal Transformer that propagates dynamics in the latent space; and (3) a Spatial Decoder that reconstructs the vertex positions from the predicted latents. More details of our architecture can be found in Appendix A.

Spatial Encoder. To efficiently process high-resolution meshes, we encode the physical state at frame T into a fixed-size set of latent vectors $\mathbf{Z}_T \in \mathbb{R}^{K \times D}$. The encoder processes two distinct inputs: 1) The cloth mesh at frame T. Each input cloth vertex is processed to obtain both a position embedding and a velocity embedding. Then, these embeddings are fused and processed by a 2-layer GNN, yielding a set of topology-aware cloth vertex tokens. 2) The collision object is represented as a set of triangles from the lookahead frame $T + 1$. Similar to the cloth mesh, we encode these using a combination of vertex embeddings (position and velocity) and explicit geometric descriptors, including the surface normal and triangle area. This produces collision triangle tokens.

To decouple the latent representation from the mesh resolution, we employ a cross-attention mechanism. We initialize a set of K learnable query tokens $\mathbf{Q}_{learn}$. These queries attend to the concatenated sequence of cloth vertex tokens and collision triangle tokens (acting as keys $\mathbf{K}$ and values $\mathbf{V}$). This operation compresses the variable-sized geometric and dynamic input into a fixed set of latent tokens $\mathbf{Z}_T$. The number of latent tokens K is a hyperparameter that controls the trade-off between compression and accuracy.

Temporal Transformer. The core dynamics are modeled by a Transformer operating on the latent tokens. The input to the transformer is the latent state $\mathbf{Z}_T$. The Transformer processes the latent representation to evolve the state forward in time. During training, we stack multi-frame past latents and use a block-causal masking (inter-frame) and self-attention layers (intra-frame) to model the complex dependencies between latent vectors. The output is the predicted latent state for the next frame $\mathbf{Z}_{T+1}$.

Spatial Decoder. The decoder reconstructs the predicted cloth mesh $\hat{\mathbf{X}}_{next}$ from the evolved latent tokens output by the Temporal Transformer. The decoding process is conditioned on the cloth's

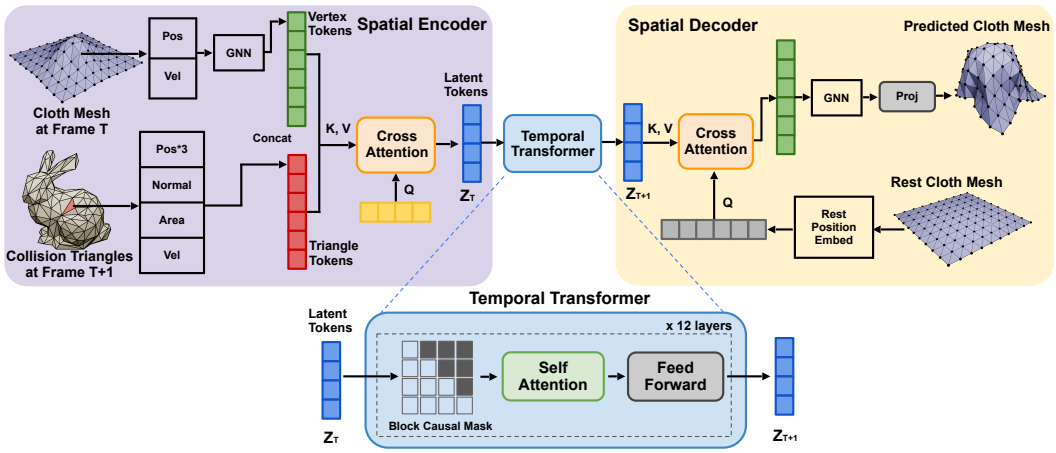

Figure 2: **Overview of the proposed auto-regressive cloth simulation architecture.** The framework consists of three main components: (1) A **Spatial Encoder** (left) that compresses the physical state of the history cloth mesh at frame T and the lookahead collision geometry at frame $T+1$ into a compact set of latent tokens. (2) A **Temporal Transformer** (middle) that models the dynamics in the latent space, predicting the future latent state Z_{T+1} (representing the next logical step in the sequence) from the current state Z_T. (3) A **Spatial Decoder** (right) that reconstructs the predicted cloth mesh from the latent representation. It queries the latent tokens using position-embedded vertices from the rest-shape mesh, followed by GNN refinement to ensure topological consistency in the final output.

rest shape to ensure the output maintains the material's intrinsic structure. We generate rest vertex tokens by applying the sinusoidal position embedding to the rest shape vertices. These tokens serve as queries $\mathbf{Q}$ in a cross-attention layer, where the keys $\mathbf{K}$ and values $\mathbf{V}$ are the predicted latent tokens. This mechanism allows the model to retrieve the dynamic state corresponding to each specific vertex based on its canonical position. The output of the cross-attention layer represents the coarse predicted state per vertex. To ensure local surface smoothness and resolve high-frequency noise, these features are passed through a final GNN block. A projection layer then maps the refined features to 3D coordinates, yielding the final predicted cloth mesh.

Unified Design for Diverse Scenarios. Notably, our architecture is *scenario-agnostic*: cross-attention compression handles arbitrary numbers of cloth vertices and collision triangles, the Transformer imposes no scenario-specific priors (e.g., humanoid context or fixed UV layouts), and collision objects are encoded as generic triangle tokens that generalize across articulated bodies, robotic grippers, and rigid objects. A single instance of our model is therefore trained jointly across all three scenarios without per-scenario adaptation.

3.3 Continuous Collision Detection Module

A learned simulator supervised only at discrete frames can produce inter-frame "tunneling": a vertex lies on the correct side of a collider at both frame T and frame $T+1$, yet passes through it in between. Discrete Collision Detection (DCD) inspects only the sampled frame states and is blind to such events, whereas Continuous Collision Detection (CCD) sweeps the linear trajectory between consecutive frames, locates the exact collision time, and corrects the vertex to a safe pre-collision position (Figure 5 in Appendix B). Our CCD module operates in two stages: a *differentiable CCD loss* $\mathcal{L}_{\text{CCD}}$ that shapes the trajectories during training, and a *non-differentiable CCD post-processor* that removes residual penetrations at inference. Both are enabled by our penetration-free training data (Sec. 4), without which a strict collision objective would penalize the ground truth itself.

Collision types. The module handles all five primitive-level contact types: Vertex-Face, Edge-Edge, Face-Vertex, and the two self-collision types Self-VF and Self-EE. The CCD loss focuses on the self-collision types, since cloth–object collisions are already supervised by the contact loss (Sec. 3.4); the post-processor covers all five types.

Differentiable CCD loss. As intersection tests are non-differentiable, we adopt a "detect-then-regress" strategy. A CCD pass first identifies the colliding primitive pairs $\mathcal{P}_{col}$, each with a collision time

Table 1: Mesh complexity statistics of our penetration-free dataset.

Subset	Cloth Mesh			Collision Mesh		Scale	
	#Vertices	#Faces	#Meshes	#Faces	#Meshes	Sequences	Frames
Human Garment	1k–3.6k	2k–7.1k	14	1k–5.1k	5	56	13.4k
Robotic Manip.	1k–4k	1.9k–7.9k	1000	0.6k	1	1000	240k
Diverse Object Collision	3.6k	7k	1	1k–4k	1000	1000	240k
Total	1k–4k	1.9k–7.9k	1015	0.6k–5.1k	1006	2056	493.4k

$t_c \in [0, 1]$ along the inter-frame trajectory. Each involved vertex is then regressed toward its safe position $\mathbf{x}_{safe} = \mathbf{x}_{start} + t_{safe}\,(\mathbf{x}_{end} - \mathbf{x}_{start})$, where $t_{safe} = \max(0,\, t_c - \epsilon)$:

$$\mathcal{L}_{\text{CCD}} = \frac{1}{|\mathcal{P}_{col}|} \sum_{i \in \mathcal{P}_{col}} \frac{1}{4} \sum_{j=1}^{4} \left\| \mathbf{x}_{end}^{(i,j)} - \mathbf{x}_{safe}^{(i,j)} \right\|^2, \tag{2}$$

where j runs over the four vertices of each colliding pair. Since the residual scales with $(1 - t_{safe})$, early collisions receive stronger gradients, so the most severe penetrations are prioritized. Collision times are obtained by cubic root finding [43]; full implementation details are provided in Appendix B.

3.4 Training Objective

We supervise the model with a combination of a reconstruction term, a contact term, and the differentiable CCD term introduced above, and train in two stages.

Reconstruction loss. The primary supervision is the mean squared positional error between the predicted vertices $\hat{\mathbf{x}}_i$ and the ground truth $\mathbf{x}_i$ over all N_v vertices:

$$\mathcal{L}_{mse} = \frac{1}{N_v} \sum_{i=1}^{N_v} \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|_2^2. \tag{3}$$

Contact loss. To instill basic cloth–object collision awareness already during pretraining, we add a contact term. For each cloth vertex we locate its nearest collision triangle via kNN, compute the signed distance along the face normal, and apply a cubic penalty on the penetration depth $d_i = \min(0,\, s_i)$, where s_i is the signed distance (negative inside the collider):

$$\mathcal{L}_{contact} = \frac{1}{N_v} \sum_{i=1}^{N_v} |d_i|^3. \tag{4}$$

The cubic form leaves shallow contacts nearly unpenalized while sharply punishing deep penetrations.

Two-stage training. We optimize the network in two stages. During *pretraining*, we use

$$\mathcal{L}_{pretrain} = \lambda_{mse}\, \mathcal{L}_{mse} + \lambda_{contact}\, \mathcal{L}_{contact}, \tag{5}$$

which yields physically plausible trajectories with only mild residual penetrations. We then *finetune* with the differentiable CCD loss added,

$$\mathcal{L}_{finetune} = \mathcal{L}_{pretrain} + \lambda_{ccd}\, \mathcal{L}_{\text{CCD}}. \tag{6}$$

Introducing $\mathcal{L}_{\text{CCD}}$ only at the finetuning stage is deliberate: the "detect-then-regress" gradients then act on already-accurate predictions rather than on the noisy outputs of an untrained model, where the estimated collision times would be unreliable and the gradients dominated by noise. The detailed training schedule is reported in Sec. 5.1.

4 Penetration-Free Dataset

We present a high-fidelity cloth dataset free of interpenetrations. Existing datasets often rely on approximate solvers that allow minor penetrations, which makes the training of a strict CCD loss impossible, as the ground truth itself would be penalized.

Simulation Settings. We generate ground-truth data using GIPC [16], a state-of-the-art penetration-free GPU cloth solver based on incremental potential contact methods. Each sequence spans 240 frames (4 s) at $\Delta t = 1/60$ s. Detailed physical parameters are provided in Appendix C.

Simulation Scenarios. To ensure the model generalizes across diverse topologies and interaction types, we construct three distinct subsets. 1) **Human Garment**, consists of T-shirts and skirts dressed on animated SMPL [27] avatars. The avatars perform a variety of complex motions [12]—including walking, running, dancing, and jumping—that induce rich cloth dynamics such as large-amplitude swinging, body-cloth contact, and self-folding in regions like the armpits and waist. 2) **Robotic Manipulation**, features over 1000 diverse cloth meshes sourced from [18], each grasped and lifted by a robotic gripper. This scenario introduces localized external forces and asymmetric deformation patterns that differ fundamentally from body-driven motion, testing the model's ability to handle point-contact manipulation and gravitational draping simultaneously. 3) **Diverse Object Collision**, simulates cloths falling freely onto rigid objects. We randomly sample over 1000 collision meshes from the Objaverse [9] dataset, covering a wide range of geometric features including sharp edges, concavities, and thin structures. This subset stresses the model's capacity to generalize across highly varied collision geometries unseen during training.

Dataset Statistics. Table 1 summarizes the dataset. Note that the ranges reflect the use of multiple mesh templates: in Human Garment, most cloth meshes have $\sim$3.6k vertices ($\sim$7k faces) with collision bodies at $\sim$5k faces, while the lower end corresponds to a few simpler garments; in Robotic Manipulation, most cloths have $\sim$4k vertices ($\sim$7.9k faces).

5 Experiments

We evaluate ClothTransformer with a *single unified model* trained jointly on all three scenarios (no per-scenario fine-tuning), demonstrating that our architecture learns shared latent dynamics across different interaction modes. We present quantitative and qualitative comparisons against state-of-the-art baselines, ablations of each design choice, and a scalability analysis of our latent-space formulation. Additional architecture, training, and evaluation details are in Appendices A, F, and G.

5.1 Implementation Details

Network Architecture. Our Spatial Encoder utilizes a 2-layer GNN to extract local features, followed by a cross-attention layer compressing the mesh into $N_{latents} = 1024$ latent tokens. The Temporal Transformer consists of 12 layers with 12 attention heads, an embedding dimension of 768, and a feed-forward dimension of 3072 with SwiGLU activation. The Spatial Decoder mirrors the encoder with a cross-attention layer followed by a 2-layer GNN for topological refinement.

Training Settings. We train our model end-to-end using the AdamW optimizer with a learning rate of 1×10^{-4} and a cosine annealing schedule decaying to 1×10^{-7}. The batch size is 32. Following the two-stage objective defined in Sec. 3.4, we first pretrain with $\mathcal{L}_{pretrain}$ (Eq. 5) for 160k steps, then finetune with $\mathcal{L}_{finetune}$ (Eq. 6) for 40k steps, taking approximately 300 NVIDIA H200 GPU hours in total; Gradient norms are clipped to 1.0. We employ a rollout curriculum strategy, starting with single-step predictions and linearly increasing the horizon to 5 steps over the first 180,000 training steps to mitigate error accumulation. The dataset is randomly split into training, validation, and test sets in an 8:1:1 ratio (per subset).

5.2 Comparative Results

We compare against three SOTA learning-based baselines: **SOTA GNN** [10, 11] (the hierarchical GNN backbone of HOOD/ContourCraft), **MAT** [20] (mesh-face tokenization with manifold-aware attention), and **LayersNet** [39] (UV-patch tokenization). All methods are trained on the same training split and evaluated on unseen test sequences. We report three metrics: **MVE** (mean vertex error, cm), **Collision Rate** (%, cloth–object penetration), and **Self-Collision Rate** (%, self-intersections detected via CCD); formal definitions are in Appendix G. We adopt complementary CCD-post-processing settings for the two views: Table 2 reports raw predictions (no post-processing) to isolate each model's underlying capability, whereas Figure 3 applies 10 CCD post-processing iterations uniformly to all methods so the visual comparison focuses on shape fidelity rather than residual penetration artifacts.

Quantitative Comparison. Our method achieves the best MVE across all three scenarios, with approximately 4–9$\times$ lower error than the strongest learning-based baseline on each scenario (and up to $\sim$16$\times$ lower than the SOTA GNN on Diverse Object Collision). We caution that low collision/self-

Table 2: Quantitative comparison on unseen test sequences. MVE: Mean Vertex Error (cm). Coll.: collision rate with collision objects (%). Self-C.: self-collision vertex rate (%). *Ours* denotes our model trained with the pretraining loss only; *Ours (CCD Loss)* additionally finetunes with the differentiable CCD loss.

Method	Human Garment			Robotic Manip.			Diverse Object Collision		
	MVE ↓	Coll. ↓	Self-C. ↓	MVE ↓	Coll. ↓	Self-C. ↓	MVE ↓	Coll. ↓	Self-C. ↓
SOTA GNN	59.13	27.14	57.43	59.42	17.42	50.02	142.67	16.92	9.67
MAT	31.13	22.0	1.0	66.63	16.63	1.4	77.15	10.15	2.18
LayersNet	149.09	13.07	79.12	69.54	13.01	77.91	154.02	18.87	71.31
Ours (CCD Loss)	**6.53**	**16.32**	**9.12**	**15.03**	**17.32**	**10.65**	**8.91**	**17.44**	**2.01**
Ours	**6.92**	**14.12**	**9.79**	**14.91**	**16.93**	**11.43**	**9.03**	**16.74**	**2.96**

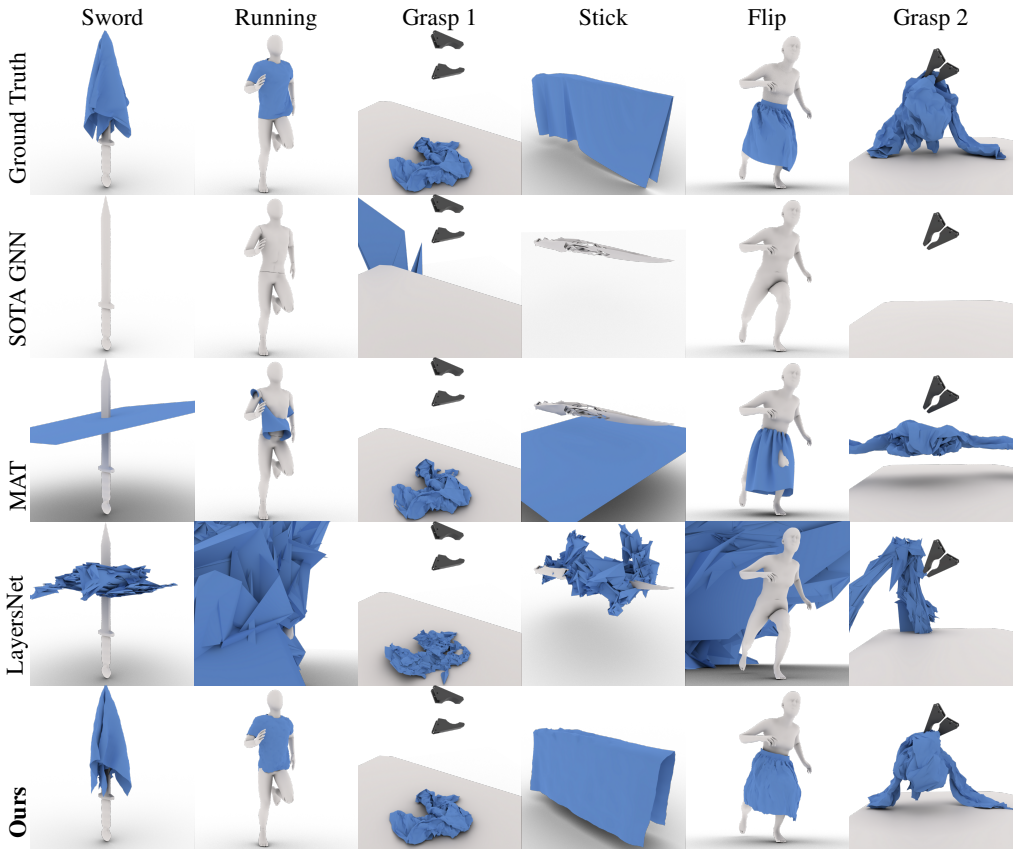

Figure 3: **Qualitative comparison on unseen test sequences.** Columns 1, 4: Diverse Object Collision (sword, stick). Columns 2, 5: Human Garment (running, front-flip). Columns 3, 6: Robotic Manipulation (static resting, grasping).

collision rates do not always indicate high quality: SOTA GNN's and LayersNet's clothes drift entirely away from the collision object (Figure 3), and MAT degenerates into a near-rigid body with minimal local deformation—both trivially achieve low contact rates but yield poor MVE. Our method instead delivers both accurate vertex predictions *and* the lowest Self-Collision Rate among non-degenerate methods.

Qualitative Comparison. Figure 3 compares visual results across the three scenarios on unseen test cases. The visual failure modes mirror the quantitative caveats above: SOTA GNN and LayersNet both diverge in long-horizon rollouts, while MAT yields near-rigid garments; our method remains visually plausible and less penetrated across all scenarios. An analysis of extreme cases is provided in Appendix J.

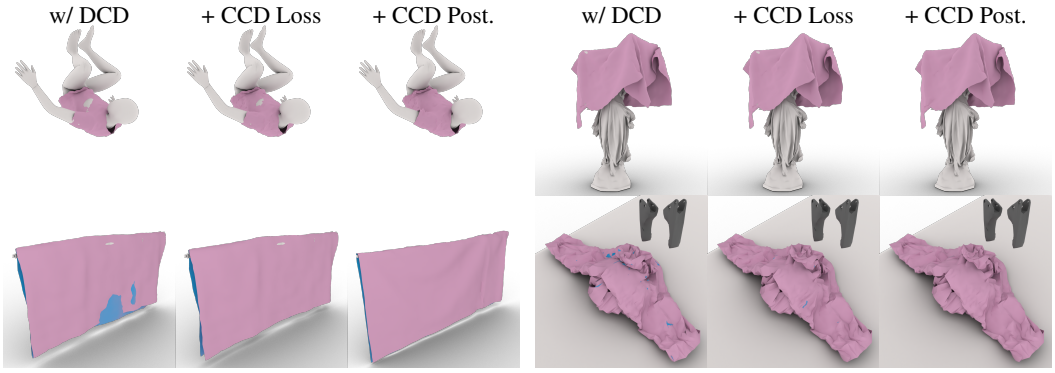

Figure 4: **Effect of differentiable CCD.** Close-up views of collision-prone regions. *Top row* (front flip, cloth on angel): cloth–collision penetrations that DCD loss cannot fully resolve; our CCD loss targets self-collision and does not improve these cases, but CCD post-processing eliminates them. *Bottom row* (cloth on stick, robotic grasping): cloth self-collision artifacts, rendered with different colors for front (pink) and back (blue) faces for clarity; the CCD loss significantly reduces self-penetrations, and CCD post-processing resolves the remainder.

Generalization Analysis. The performance gap reflects a fundamental architectural distinction: our latent-space formulation uniformly encodes cloth vertices and collision triangles into a fixed-size token set, while the SOTA GNN's mesh-edge message passing, MAT's manifold-constrained attention, and LayersNet's UV-based parameterization each impose scenario-specific structural priors that limit generalization.

Isolating the architectural factor. Since the SOTA GNN backbone (HOOD, ContourCraft) is originally trained with a self-supervised physics-based loss, one might worry that retraining it under our supervised setup (Table 2) understates its true capability. To rule this out, we re-evaluate it under its *native* self-supervised paradigm in two configurations: (i) trained on *only* Human Garment, matching its original deployment, and (ii) trained jointly on all three scenarios, matching our unified setting; both are compared against (iii) our unified model. Configuration (i) produces reasonable visual quality on Human Garment but still deviates noticeably from the ground truth, while (ii) degrades sharply across all scenarios, indicating that the SOTA GNN backbone struggles to absorb the increased data diversity. Our model outperforms (ii) everywhere and even surpasses the scenario-specialized (i) on its own distribution, confirming that the gap in Table 2 stems from the SOTA GNN architecture itself, not the supervised setup. Full numbers and qualitative results are in Appendix I (Table 6, Figure 8).

5.3 Ablation Studies

Impact of the CCD Module. Our high-fidelity penetration-free dataset additionally enables CCD-based operations, including the differentiable CCD loss during training and CCD post-processing at inference. We ablate the CCD module under three progressive settings: (1) *w/ DCD Loss* only, (2) + *CCD Loss* during training, and (3) + *CCD Post.* at inference. As shown in Figure 4, DCD loss alone leaves both cloth–collision and self-collision artifacts. Adding the CCD loss significantly reduces self-penetrations, while CCD post-processing eliminates remaining artifacts of both types. Furthermore, a direct comparison with ContourCraft [11], a state-of-the-art DCD-based approach is reported in Appendix H.

Latent Compression Rate. We vary the number of latent tokens $N_{latents}$ among 512, 1024, 2048, and no compression. As shown in Table 3, three key findings emerge: (1) MVE improves steadily from N=512 to N=2048, yet the uncompressed variant performs *worse* due to insufficient training convergence, highlighting the importance of latent compression; (2) N=1024 infers at ∼4.9 ms/frame, well beyond real-time speed and ∼18× faster than the uncompressed variant; (3) N=1024 offers the best accuracy–efficiency trade-off. A visual comparison and detailed analysis are provided in Appendix D.

Spatial GNN. We replace the GNN encoder/decoder with simple MLPs (*w/o Local GNN*), keeping the latent dimension fixed at $N_{latents}$=2048. As shown in Table 4, removing the GNN increases MVE from 6.01 cm to 7.18 cm (+19%) and raises the collision rate from 15.19% to 22.98%. Without the GNN's explicit message passing along mesh edges, the MLP-only encoder must infer local

Table 3: Ablation on latent compression rate (Human Garment, 50k steps). "No Comp." operates directly on all mesh vertices.

$N_{latents}$	MVE ↓	Coll. ↓	Infer. (ms) ↓	Train (h) ↓
512	9.79	22.55	4.88	9.78
1024 (default)	7.14	19.98	4.90	10.87
2048	6.01	15.19	10.75	15.68
No Comp.	10.01	18.56	90.07	18.01

Table 4: Ablation on spatial GNN (Human Garment, $N_{latents}$=2048, 50k steps).

Variant	MVE ↓	Coll. ↓
w/o GNN	7.18	22.98
Ours	**6.01**	**15.19**

Table 5: Scalability analysis across mesh resolutions on a single NVIDIA RTX 4090 (24 GB). MVE (cm) ↓ / inference time (ms/frame) ↓. LayersNet runs out of memory at 40k vertices.

	5k vertices		**10k vertices**		**20k vertices**		**40k vertices**	
Method	MVE ↓	Time ↓	MVE ↓	Time ↓	MVE ↓	Time ↓	MVE ↓	Time ↓
SOTA GNN	55.34	130.01	74.24	147.12	73.98	231.57	201.46	471.78
MAT	32.68	66.1	54.12	294	67.1	634.32	72.36	1449.27
LayersNet	149.27	59.67	79.15	90.64	69.24	138.08	OOM	
Ours	**7.21**	**22.24**	**5.81**	**43.03**	**34.91**	**90.87**	**57.32**	**275.27**

surface geometry purely from per-vertex features, losing the topological connectivity that is critical for accurate spatial encoding and decoding.

5.4 Scalability Analysis

A key advantage of the latent-space formulation is scalable dynamics. We validate this by training all methods on the Human Garment subset (∼3.6k vertices) and evaluating both accuracy (MVE) and inference speed (ms/frame) at four test-time mesh resolutions: 5k, 10k, 20k, and 40k vertices. All timings are measured on a single NVIDIA RTX 4090 (24 GB). Results are shown in Table 5.

Cross-Resolution Accuracy. Our method achieves the best MVE across all four resolutions, demonstrating strong cross-resolution generalization. Even at 40k vertices—roughly $11\times$ the training resolution—our method maintains reasonable predictions and outperforms all baselines.

Inference Speed. Our method is consistently the fastest across all resolutions, since the core Temporal Transformer cost stays fixed at $O(N_{latents}^2)$ regardless of mesh size. Notably, at 40k vertices our method is still $\sim 1.7\times$ faster than the second-fastest one (SOTA GNN). Per-method timings and end-to-end pipeline timing details are provided in Appendix E.

6 Conclusion

We presented **ClothTransformer**, a unified Transformer framework that reformulates cloth simulation as autoregressive sequence modeling in a learned latent space. A single model handles diverse scenarios—body-driven garments, robotic manipulation, and free-fall collisions—and achieves approximately 4–$9\times$ lower error than prior state-of-the-art methods across all scenarios. The latent-space formulation compresses arbitrary-resolution meshes into a fixed-size set of tokens, making temporal dynamics computation independent of mesh resolution. To support physically plausible training, we construct a diverse-scenario penetration-free dataset spanning all three settings, which enables our differentiable CCD module to suppress penetration artifacts.

Limitations and Future Work. Currently, our model infers material properties implicitly; future iterations could incorporate explicit physical parameters (*e.g.*, stiffness) for greater artistic control. Furthermore, we observe that in extremely challenging scenarios, single-step autoregressive (AR) predictions of highly nonlinear physical dynamics can still lead to considerable error accumulation (see Appendix J). This makes one-shot (or chunked, multi-step) prediction paradigms a potential alternative for future investigation. Additionally, extending the framework to handle topological changes (*e.g.*, tearing) and integrating the framework with multimodal foundation models for text-guided physics generation are promising directions for future research.

Acknowledgments and Disclosure of Funding

This research is supported by the National Research Foundation, Singapore, under its NRF Fellowship Award NRF-NRFF16-2024-0003 and NTU SUG-NAP. This research is also supported by cash and in-kind funding from NTU S-Lab and industry partner(s). The authors declare no competing interests.

References

[1] David Baraff and Andrew P. Witkin. Large steps in cloth simulation. In Steve Cunningham, Walt Bransford, and Michael F. Cohen, editors, *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1998, Orlando, FL, USA, July 19-24, 1998*, pages 43–54. ACM, 1998. doi: 10.1145/280814.280821. URL https://doi.org/10.1145/280814.280821.

[2] Hugo Bertiche, Meysam Madadi, Emilio Tylson, and Sergio Escalera. DeePSD: Automatic deep skinning and pose space deformation for 3D garment animation. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 5451–5460. IEEE, 2021. doi: 10.1109/ICCV48922.2021.00542. URL https://doi.org/10.1109/ICCV48922.2021.00542.

[3] Hugo Bertiche, Meysam Madadi, and Sergio Escalera. Neural cloth simulation. *ACM Transactions on Graphics (TOG)*, 41(6):1–14, 2022.

[4] Robert Bridson, Ronald Fedkiw, and John Anderson. Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph.*, 21(3):594–603, 2002. doi: 10.1145/566654.566623. URL https://doi.org/10.1145/566654.566623.

[5] Yadi Cao, Menglei Chai, Minchen Li, and Chenfanfu Jiang. Efficient learning of mesh-based physical simulation with bi-stride multi-scale graph neural network. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 3541–3558. PMLR, 2023. URL https://proceedings.mlr.press/v202/cao23a.html.

[6] Yue Chang, Peter Yichen Chen, Zhecheng Wang, Maurizio M. Chiaramonte, Kevin Carlberg, and Eitan Grinspun. LiCROM: Linear-subspace continuous reduced order modeling with neural fields. In June Kim, Ming C. Lin, and Bernd Bickel, editors, *SIGGRAPH Asia 2023 Conference Papers, SA 2023, Sydney, NSW, Australia, December 12-15, 2023*, pages 111:1–111:12. ACM, 2023. doi: 10.1145/3610548.3618158. URL https://doi.org/10.1145/3610548.3618158.

[7] Anka He Chen, Ziheng Liu, Yin Yang, and Cem Yuksel. Vertex block descent. *ACM Trans. Graph.*, 43(4):116:1–116:16, 2024. doi: 10.1145/3658179. URL https://doi.org/10.1145/3658179.

[8] Peter Yichen Chen, Jinxu Xiang, Dong Heon Cho, Yue Chang, G. A. Pershing, Henrique Teles Maia, Maurizio M. Chiaramonte, Kevin T. Carlberg, and Eitan Grinspun. CROM: continuous reduced-order modeling of PDEs using implicit neural representations. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=FUORz1tG8Og.

[9] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3D objects. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13142–13153, 2023.

[10] Artur Grigorev, Michael J. Black, and Otmar Hilliges. HOOD: hierarchical graphs for generalized modelling of clothing dynamics. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 16965–16974. IEEE, 2023. doi: 10.1109/CVPR52729.2023.01627. URL https://doi.org/10.1109/CVPR52729.2023.01627.

[11] Artur Grigorev, Giorgio Becherini, Michael J. Black, Otmar Hilliges, and Bernhard Thomaszewski. ContourCraft: Learning to resolve intersections in neural multi-garment simulations. In Andres Burbano, Denis Zorin, and Wojciech Jarosz, editors, *ACM SIGGRAPH 2024 Conference Papers, SIGGRAPH 2024, Denver, CO, USA, 27 July 2024- 1 August 2024*, page 81. ACM, 2024. doi: 10.1145/3641519.3657408. URL https://doi.org/10.1145/3641519.3657408.

[12] Zhiyang Guo, Jinxu Xiang, Kai Ma, Wengang Zhou, Houqiang Li, and Ran Zhang. Make-It-Animatable: An efficient framework for authoring animation-ready 3D characters. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 10783–10792, 2025.

[13] Xu Han, Han Gao, Tobias Pfaff, Jian-Xun Wang, and Liping Liu. Predicting physics in mesh-reduced space with temporal attention. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=XctLdNfCmP.

[14] Jing He, Yuanjie Cao, TangSheng Guo, Wei Liang, Jin Huang, Qian Liu, Huaiyuan Yang, Sen Liu, and Ruhan He. From physically-based to learning-based in cloth simulation: evolution and future - a scoping review. *Vis. Comput.*, 41(15):12711–12742, 2025. doi: 10.1007/S00371-025-04182-3. URL https://doi.org/10.1007/s00371-025-04182-3.

[15] Benjamin J. Holzschuh, Qiang Liu, Georg Kohl, and Nils Thuerey. PDE-Transformer: Efficient and versatile transformers for physics simulations. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=3BaJMRaPSx.

[16] Kemeng Huang, Floyd M. Chitalu, Huancheng Lin, and Taku Komura. GIPC: fast and stable gauss-newton optimization of IPC barrier energy. *ACM Trans. Graph.*, 43(2):23:1–23:18, 2024. doi: 10.1145/3643028. URL https://doi.org/10.1145/3643028.

[17] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and João Carreira. Perceiver: General perception with iterative attention. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 4651–4664. PMLR, 2021. URL http://proceedings.mlr.press/v139/jaegle21a.html.

[18] Maria Korosteleva and Sung-Hee Lee. Generating datasets of 3D garments with sewing patterns. In Joaquin Vanschoren and Sai-Kit Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/013d407166ec4fa56eb1e1f8cbe183b9-Abstract-round1.html.

[19] Minchen Li, Zachary Ferguson, Teseo Schneider, Timothy R. Langlois, Denis Zorin, Daniele Panozzo, Chenfanfu Jiang, and Danny M. Kaufman. Incremental potential contact: intersection- and inversion-free, large-deformation dynamics. *ACM Trans. Graph.*, 39(4):49, 2020. doi: 10.1145/3386569.3392425. URL https://doi.org/10.1145/3386569.3392425.

[20] Peizhuo Li, Tuanfeng Y. Wang, Timur Levent Kesdogan, Duygu Ceylan, and Olga Sorkine-Hornung. Neural garment dynamics via manifold-aware transformers. *Comput. Graph. Forum*, 43(2):e15028, 2024. doi: 10.1111/CGF.15028. URL https://doi.org/10.1111/cgf.15028.

[21] Tianxing Li, Rui Shi, Qing Zhu, and Takashi Kanai. SwinGar: Spectrum-inspired neural dynamic deformation for free-swinging garments. *IEEE Trans. Vis. Comput. Graph.*, 30(10):6913–6927, 2024. doi: 10.1109/TVCG.2023.3346055. URL https://doi.org/10.1109/TVCG.2023.3346055.

[22] Tianxing Li, Zhi Qiao, Zihui Li, Rui Shi, and Qing Zhu. GarTrans: Transformer-based architecture for dynamic and detailed garment deformation. *Comput. Vis. Media*, 11(6):1209–1226, 2025. doi: 10.26599/CVM.2025.9450448. URL https://doi.org/10.26599/cvm.2025.9450448.

[23] Tianxing Li, Rui Shi, Qing Zhu, Liguo Zhang, and Takashi Kanai. Spectrum-enhanced graph attention network for garment mesh deformation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 47(8):7153–7170, 2025. doi: 10.1109/TPAMI.2025.3570523. URL https://doi.org/10.1109/TPAMI.2025.3570523.

[24] Zhouyingcheng Liao, Sinan Wang, and Taku Komura. SENC: handling self-collision in neural cloth simulation. In Ales Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol, editors, *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part IX*, volume 15067 of *Lecture Notes in Computer Science*, pages 385–402. Springer, 2024. doi: 10.1007/978-3-031-72673-6_21. URL https://doi.org/10.1007/978-3-031-72673-6_21.

[25] Emmanuel Ian Libao, Myeongjin Lee, Sumin Kim, and Sung-Hee Lee. MeshGraphNetRP: Improving generalization of GNN-based cloth simulation. In Julien Pettré, Barbara Solenthaler, Rachel McDonnell, and Christopher Peters, editors, *Proceedings of the 16th ACM SIGGRAPH Conference on Motion, Interaction and Games, MIG 2023, Rennes, France, November 15-17, 2023*, pages 5:1–5:7. ACM, 2023. doi: 10.1145/3623264.3624441. URL https://doi.org/10.1145/3623264.3624441.

[26] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3D object. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 9264–9275. IEEE, 2023. doi: 10.1109/ICCV51070.2023.00853. URL https://doi.org/10.1109/ICCV51070.2023.00853.

[27] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: a skinned multi-person linear model. *ACM Trans. Graph.*, 34(6):248:1–248:16, 2015. doi: 10.1145/2816795.2818013. URL https://doi.org/10.1145/2816795.2818013.

[28] Qianli Ma, Jinlong Yang, Anurag Ranjan, Sergi Pujades, Gerard Pons-Moll, Siyu Tang, and Michael J. Black. Learning to dress 3D people in generative clothing. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 6468–6477. Computer Vision Foundation / IEEE, 2020. doi: 10.1109/CVPR42600.2020.00650. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Ma_Learning_to_Dress_3D_People_in_Generative_Clothing_CVPR_2020_paper.html.

[29] Miles Macklin, Matthias Müller, and Nuttapong Chentanez. XPBD: position-based simulation of compliant constrained dynamics. In Michael Neff, Roland Geraerts, and Hubert P. H. Shum, editors, *Proceedings of the 9th International Conference on Motion in Games, MIG 2016, Burlingame, California, USA, October 10-12, 2016*, pages 49–54. ACM, 2016. doi: 10.1145/2994258.2994272. URL https://doi.org/10.1145/2994258.2994272.

[30] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118, 2007.

[31] Chaitanya Patel, Zhouyingcheng Liao, and Gerard Pons-Moll. TailorNet: Predicting clothing in 3D as a function of human pose, shape and garment style. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 7363–7373. Computer Vision Foundation / IEEE, 2020. doi: 10.1109/CVPR42600.2020.00739. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Patel_TailorNet_Predicting_Clothing_in_3D_as_a_Function_of_Human_CVPR_2020_paper.html.

[32] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=roNqYL0_XP.

[33] Cristian Romero, Dan Casas, Jesús Pérez, and Miguel A. Otaduy. Learning contact corrections for handle-based subspace dynamics. *ACM Trans. Graph.*, 40(4):131:1–131:12, 2021. doi: 10.1145/3450626.3459875. URL https://doi.org/10.1145/3450626.3459875.

[34] Cristian Romero, Dan Casas, Maurizio M. Chiaramonte, and Miguel A. Otaduy. Contact-centric deformation learning. *ACM Trans. Graph.*, 41(4):70:1–70:11, 2022. doi: 10.1145/3528223.3530182. URL https://doi.org/10.1145/3528223.3530182.

[35] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to simulate complex physics with graph networks. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 8459–8468. PMLR, 2020. URL http://proceedings.mlr.press/v119/sanchez-gonzalez20a.html.

[36] Igor Santesteban, Nils Thuerey, Miguel A. Otaduy, and Dan Casas. Self-supervised collision handling via generative 3D garment models for virtual try-on. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 11763–11773. Computer Vision Foundation / IEEE, 2021. doi: 10.1109/CVPR46437.2021.01159. URL https://openaccess.thecvf.com/content/CVPR2021/html/Santesteban_Self-Supervised_Collision_Handling_via_Generative_3D_Garment_Models_for_Virtual_CVPR_2021_paper.html.

[37] Igor Santesteban, Miguel A. Otaduy, and Dan Casas. SNUG: self-supervised neural dynamic garments. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 8130–8140. IEEE, 2022. doi: 10.1109/CVPR52688.2022.00797. URL https://doi.org/10.1109/CVPR52688.2022.00797.

[38] Yidi Shao, Chen Change Loy, and Bo Dai. Transformer with implicit edges for particle-based physics simulation. In Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XIX*, volume 13679 of *Lecture Notes in Computer Science*, pages 549–564. Springer, 2022. doi: 10.1007/978-3-031-19800-7_32. URL https://doi.org/10.1007/978-3-031-19800-7_32.

[39] Yidi Shao, Chen Change Loy, and Bo Dai. Towards multi-layered 3D garments animation. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 14315–14324. IEEE, 2023. doi: 10.1109/ICCV51070.2023.01321. URL https://doi.org/10.1109/ICCV51070.2023.01321.

[40] Qingyang Tan, Yi Zhou, Tuanfeng Y. Wang, Duygu Ceylan, Xin Sun, and Dinesh Manocha. A repulsive force unit for garment collision handling in neural networks. In Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part III*, volume 13663 of *Lecture Notes in Computer Science*, pages 451–467. Springer, 2022. doi: 10.1007/978-3-031-20062-5_26. URL https://doi.org/10.1007/978-3-031-20062-5_26.

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[42] Raquel Vidaurre, Igor Santesteban, Elena Garces, and Dan Casas. Fully convolutional graph neural networks for parametric virtual try-on. *Comput. Graph. Forum*, 39(8):145–156, 2020. doi: 10.1111/CGF.14109. URL https://doi.org/10.1111/cgf.14109.

[43] Cem Yuksel. A fast & robust solution for cubic & higher-order polynomials. In *ACM SIGGRAPH 2022 Talks*, pages 1–2. ACM, 2022.

[44] Chong Zeng, Yue Dong, Pieter Peers, Hongzhi Wu, and Xin Tong. RenderFormer: Transformer-based neural rendering of triangle meshes with global illumination. In Ginger Alford, Hao (Richard) Zhang, and Adriana Schulz, editors, *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference, SIGGRAPH Conference Papers 2025, Vancouver, BC, Canada, August 10-14, 2025*, pages 48:1–48:11. ACM, 2025. doi: 10.1145/3721238.3730595. URL https://doi.org/10.1145/3721238.3730595.

[45] Biao Zhang, Jiapeng Tang, Matthias Nießner, and Peter Wonka. 3DShape2VecSet: A 3D shape representation for neural fields and generative diffusion models. *ACM Trans. Graph.*, 42(4):92:1–92:16, 2023. doi: 10.1145/3592442. URL https://doi.org/10.1145/3592442.

Appendix: Supplementary Material

A Architecture Details

Here we provide additional details of the three components introduced in Sec. 3 of the main paper.

Spatial Encoder. The encoder maps the physical state at frame T into a fixed-size set of latent vectors $\mathbf{Z}_T \in \mathbb{R}^{K \times D}$, processing two inputs: the cloth mesh at frame T and the collision geometry at frame $T{+}1$.

Cloth Feature Extraction. Each cloth vertex is encoded with two embeddings: (1) a **Position Embedding** that applies sinusoidal positional encoding to the 3D coordinates $\mathbf{X}_T$, mapping them into a high-dimensional feature space; (2) a **Velocity Embedding** that projects the vertex velocities $\mathbf{V}_T$ into the same feature dimension to capture instantaneous motion. In practice, directly using velocity as input leads to error accumulation during autoregressive rollout, even with data normalization. We therefore compute the velocity as the position difference between two consecutive frames, $\mathbf{V}_T = \mathbf{X}_T - \mathbf{X}_{T-1}$, and multiply it by a scaling coefficient to keep its magnitude comparable to the position embedding, which stabilizes long-horizon inference. These embeddings are fused and processed by a 2-layer GNN that aggregates features along mesh edges $\mathcal{E}$, yielding topology-aware *Cloth Vertex Tokens*.

Collision Triangle Embedding. Collision objects are represented as triangles from the lookahead frame $T{+}1$. Each triangle is encoded using: (1) **Vertex and Velocity Embedding** applied to its three vertices, ensuring the model is aware of the object's motion trajectory; (2) **Geometric Features** including the surface normal $\mathbf{n}$ and triangle area A. The geometric and dynamic features are concatenated to produce *Collision Triangle Tokens*.

Latent Compression. A set of K learnable query tokens $\mathbf{Q}_{\text{learn}}$ attends to the concatenated Cloth Vertex Tokens and Collision Triangle Tokens via cross-attention, compressing the variable-sized input into a fixed set of K latent tokens $\mathbf{Z}_T$. We set $K{=}1024$ by default.

Temporal Transformer. The Transformer takes the latent state $\mathbf{Z}_T$ as input and evolves it forward in time. It uses block-causal masking across frames and self-attention within each frame to model inter-token dependencies. The architecture consists of 12 layers with 12 attention heads, an embedding dimension of 768, and a feed-forward dimension of 3072 with SwiGLU activation. The output is the predicted next-frame latent state $\mathbf{Z}_{T+1}$.

Spatial Decoder. The decoder reconstructs vertex positions from the predicted latent tokens. Rest-shape vertices are encoded via sinusoidal Position Embedding into *Rest Vertex Tokens*, which serve as queries in a cross-attention layer against the predicted latent tokens. This retrieves the dynamic state for each vertex based on its canonical position. A final 2-layer GNN refines the output to ensure local surface smoothness, followed by a projection layer mapping features to 3D coordinates.

B CCD Implementation Details

Figure 5 illustrates the conceptual difference between DCD and CCD: DCD only inspects discrete frame states and can miss mid-step penetrations, whereas CCD sweeps the entire inter-frame trajectory to locate the exact collision time and correct the vertex to a safe pre-collision position. Below we describe how this principle is realized in our framework.

Collision Types. Our CCD module handles five primitive-level collision types:

1. **Vertex-Face (VF):** A cloth vertex colliding with a collision object triangle.
2. **Edge-Edge (EE):** A cloth edge colliding with a collision object edge.
3. **Face-Vertex (FV):** A collision object vertex colliding with a cloth triangle.
4. **Self-VF:** A cloth vertex colliding with another triangle on the same cloth mesh.
5. **Self-EE:** A cloth edge colliding with another edge on the same cloth mesh.

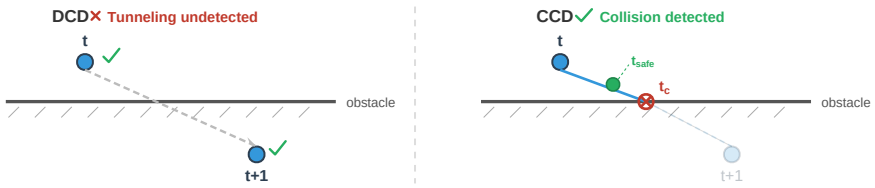

Figure 5: **DCD vs. CCD.** DCD checks only at discrete time steps and can miss mid-step penetrations (left). CCD sweeps the entire trajectory between two consecutive frames to locate the exact collision time t_c, then corrects the vertex to a safe position t_{safe} before the collision occurs, eliminating tunneling by construction (right).

During training, the CCD loss focuses on self-collision types (Self-VF and Self-EE), as cloth–object collisions are supervised by $\mathcal{L}_{\text{contact}}$. During inference, the CCD post-processing covers all five types.

Differentiable CCD Loss. Since the boolean result of a geometric intersection test is non-differentiable, we employ a "detect-then-regress" strategy. First, a non-differentiable CCD pass identifies the set of colliding primitive pairs $\mathcal{P}_{col}$ and the corresponding collision time $t_c \in [0, 1]$ for each pair. We then compute a safe time $t_{safe} = \max(0,\, t_c - \epsilon)$, just before the collision moment, and obtain the collision-free position for each involved vertex via linear interpolation: $\mathbf{x}_{safe} = \mathbf{x}_{start} + t_{safe} \cdot (\mathbf{x}_{end} - \mathbf{x}_{start})$. The loss regresses the predicted end position toward this safe position:

$$\mathcal{L}_{\text{CCD}} = \frac{1}{|\mathcal{P}_{col}|} \sum_{i \in \mathcal{P}_{col}} \frac{1}{4} \sum_{j=1}^{4} \|\mathbf{x}_{end}^{(i,j)} - \mathbf{x}_{safe}^{(i,j)}\|^2, \tag{7}$$

where the inner sum runs over the four vertices involved in each colliding pair (one free vertex and three triangle vertices for Self-VF, or two endpoints per edge for Self-EE). Since $\|\mathbf{x}_{end} - \mathbf{x}_{safe}\|^2 = (1 - t_{safe})^2 \|\mathbf{x}_{end} - \mathbf{x}_{start}\|^2$, early collisions ($t_c \approx 0$) receive stronger gradients than late ones ($t_c \approx 1$), naturally prioritizing severe penetrations.

Cubic Root Finding. Both Point-Triangle (VF, FV, Self-VF) and Edge-Edge (EE, Self-EE) CCD tests reduce to finding the roots of a cubic polynomial

$$P(t) = at^3 + bt^2 + ct + d = 0, \quad t \in [0, 1], \tag{8}$$

where t parameterizes the linear trajectory between consecutive frames. Standard iterative root solvers are computationally expensive and numerically unstable when applied to thousands of primitive pairs. We adopt the method of Yuksel [43], which analytically computes the critical points of $P(t)$ (roots of $P'(t)$) to decompose $[0, 1]$ into monotonic intervals, then checks for sign changes at interval boundaries and applies Newton-Raphson iteration only within intervals where a root is guaranteed. This approach enables efficient and robust CCD on large meshes.

Iterative Post-Processing. At inference, the CCD post-processing resolves collisions iteratively. For each detected collision at time t_c, we compute $t_{\text{safe}} = \max(0,\, t_c - \epsilon)$ and reset the penetrating vertices to their positions at t_{safe} via linear interpolation along the motion vector. Since resolving one collision may introduce secondary collisions, this process is repeated until convergence (i.e., no new collisions are detected) or a maximum iteration count is reached.

C Dataset Simulation Details

We simulate the Baraff and Witkin [1] cloth model using the GIPC solver [16] with the following material parameters: stretching Young's modulus $E_s = 10^6$ Pa, bending Young's modulus $E_b = 10^5$ Pa, Poisson's ratio $\nu = 0.49$, shear stiffness $G = 5 \times 10^6$ Pa, and cloth density $\rho = 200\,\text{g/m}^2$. The simulation uses a fixed time step of $\Delta t = 1/60\,\text{s}$, a friction coefficient of $\mu = 0.4$, and each sequence spans 240 frames (4 seconds).

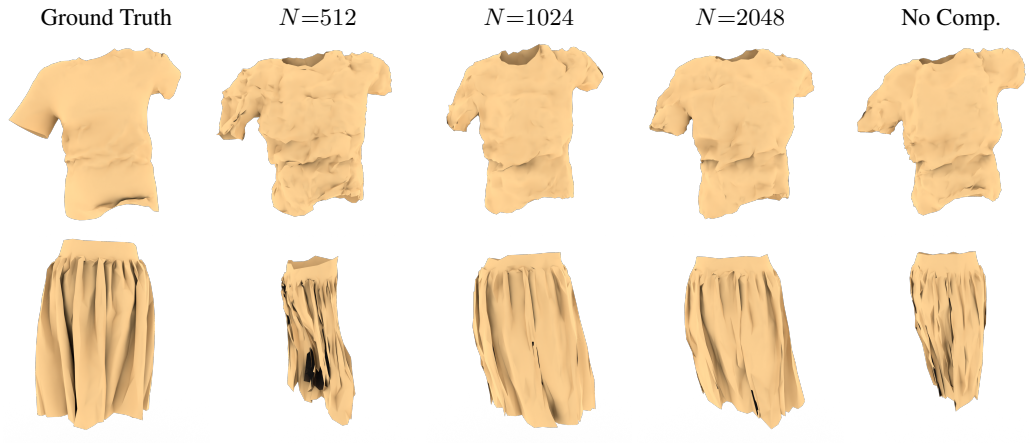

Figure 6: **Visual comparison of different latent compression rates.** N=512 produces abnormal deformation artifacts due to excessive compression. The uncompressed variant (No Comp.) exhibits similar artifacts due to insufficient training convergence. Our default N=1024 achieves a good balance between quality and efficiency.

D Latent Compression Analysis

We provide a detailed analysis of the latent compression ablation (Table 3 in the main paper). Figure 6 qualitatively illustrates the visual artifacts at different compression rates.

Accuracy. MVE improves steadily as $N_{latents}$ increases from 512 to 2048, with our default N=1024 reducing MVE by 27% relative to N=512 and N=2048 achieving a further 16% reduction. Notably, the uncompressed variant ("No Comp.") performs *worse* than both N=1024 and N=2048: because the full-resolution Transformer has substantially more parameters, it requires far more training iterations to converge, despite already consuming 18.01 GPU hours—nearly 1.7× the training cost of our default N=1024 (10.87 h). This emphasizes the importance of latent compression.

Efficiency. The latent bottleneck provides dramatic gains in both inference speed and training cost. Both N=512 and N=1024 run at ∼4.9 ms per frame—well within real-time budgets—while N=2048 is ∼2.2× slower at 10.75 ms due to the quadratic attention cost $O(N_{latents}^2)$. The uncompressed variant is ∼18× slower than our default at 90.07 ms. Training time follows a similar trend: N=512 and N=1024 require only 9.78 h and 10.87 h respectively, while N=2048 takes 15.68 h and the uncompressed variant 18.01 h.

E Scalability Analysis Details

Per-Method Inference Speed. We provide a per-method analysis of the scalability results in Table 5 of the main paper. Our method scales from 22.24 ms at 5k vertices to 275.27 ms at 40k vertices. This growth is approximately linear and stems solely from the lightweight spatial encoder/decoder; the core Temporal Transformer cost remains fixed at $O(N_{latents}^2)$ regardless of mesh size. LayersNet is the second fastest at lower resolutions (59.67 ms at 5k, 90.64 ms at 10k) but exhausts the 24 GB GPU memory at 40k vertices, as its UV-patch tokenization materializes full-resolution feature maps that grow quadratically with mesh size. The SOTA GNN scales moderately, from 130.01 ms at 5k to 471.78 ms at 40k, due to its multi-level message-passing graph. MAT scales the worst with mesh resolution, jumping from 66.1 ms at 5k to 1449.27 ms at 40k, as its per-face attention cost grows steeply with mesh size. At 5k vertices, our method is ∼2.7× faster than LayersNet and ∼5.8× faster than the SOTA GNN.

End-to-End Pipeline Timing. The per-frame timings in Table 5 of the main paper measure the neural network forward pass only. With CCD post-processing enabled (∼10 iterations on average), the full pipeline adds approximately 30 ms on a single RTX 4090, yielding a total of ∼52 ms/frame at 5k vertices and ∼305 ms/frame at 40k vertices. For reference, the GIPC solver [16] used to generate our ground truth requires approximately 10 s per frame on this scenario. Our full pipeline thus achieves

roughly a $200\times$ speedup over GIPC while maintaining competitive accuracy, making it practical for interactive applications.

F Ablation Study Settings

The ablation studies reported in the main paper use different configurations depending on the experiment:

Impact of the CCD Module (Figure 4 in the main paper and Figure 7 in Appendix H). This ablation was conducted on a smaller subset of the full training data to reduce computational cost.

Latent Compression Rate and Spatial GNN (Section 5.3 in the main paper). These two ablations were conducted using a smaller network configuration (hidden dimension D=256, 6 layers, 8 attention heads) on the Human Garment scenario, trained for 50k steps. This lighter setup allows efficient exploration of the design space; the trends observed are consistent with the full-scale model.

G Evaluation Details

Baseline Adaptation. The SOTA GNN backbone [10, 11] was originally designed for unsupervised training with physics-based losses. For a fair comparison, we adapt it to a supervised setting by replacing its unsupervised objectives with the same loss used in our pretraining stage, and train it on our penetration-free dataset.

Metric Definitions.

- **MVE (cm):** Mean Vertex Error, the average Euclidean distance between predicted and ground-truth vertex positions over all vertices and frames:

$$\text{MVE} = \frac{1}{T \cdot N_v} \sum_{t=1}^{T} \sum_{i=1}^{N_v} \|\hat{\mathbf{x}}_i^t - \mathbf{x}_i^t\|_2. \tag{9}$$

- **Collision Rate (%):** The percentage of cloth vertices that penetrate the collision object, averaged over all frames.
- **Self-Collision Rate (%):** The percentage of cloth vertices involved in self-intersections, detected via CCD between consecutive frames. Unlike discrete checks, this metric captures tunneling events where a vertex passes through and returns within a single time step. The involved vertices include the colliding vertex and the three triangle vertices (for Self-VF) or the four edge endpoints (for Self-EE).

H CCD vs. DCD-Based Self-Collision Handling

To further validate the advantage of CCD over DCD-based approaches, we compare against ContourCraft [11], a representative DCD method that detects self-intersecting contours and learns to resolve them. As shown in Figure 7, on a challenging robotic grasping scenario where the cloth is put on a table (producing dense self-collisions), both ContourCraft and our CCD loss reduce self-collisions to some extent, but neither fully eliminates them. Our CCD post-processing, which iteratively resolves trajectory-level intersections, achieves clean, intersection-free results.

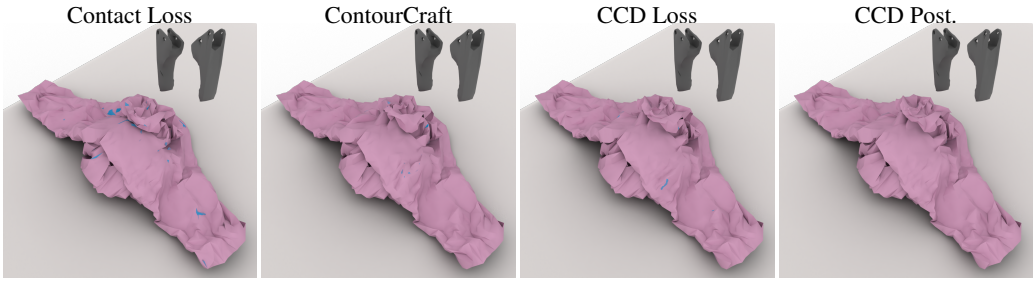

Figure 7: **CCD vs. DCD-based self-collision handling** on a challenging folded-cloth grasping scenario. ContourCraft [11] and our CCD loss both reduce self-collisions but leave residual artifacts. Our CCD post-processing fully resolves remaining intersections.

I Unified vs. Specialized Training

We provide quantitative and qualitative results for the comparison discussed in Section 5.2 of the main paper. SOTA GNN methods (e.g., HOOD [10], ContourCraft [11]) are typically trained with unsupervised physics-based losses. Table 6 reports MVE on the Human Garment scenario for the three settings, and Figure 8 shows the corresponding visual results. Training the SOTA GNN unsupervised on a single scenario (Human Garment) yields reasonable visual quality on that scenario but still deviates from the ground truth (16.95 cm MVE). Training the same SOTA GNN unsupervised on the unified dataset (all three scenarios) degrades sharply, with severe stretching artifacts on Human Garment frames (76.92 cm). Our unified model achieves the lowest MVE (6.92 cm) and produces visually plausible results that match or exceed the single-scenario specialized SOTA GNN on its own training distribution.

Table 6: Quantitative comparison of unified vs. specialized training on the Human Garment scenario. MVE (cm) ↓.

Method	MVE ↓
SOTA GNN (single, unsupervised)	16.95
SOTA GNN (unified, unsupervised)	76.92
Ours (unified)	**6.92**

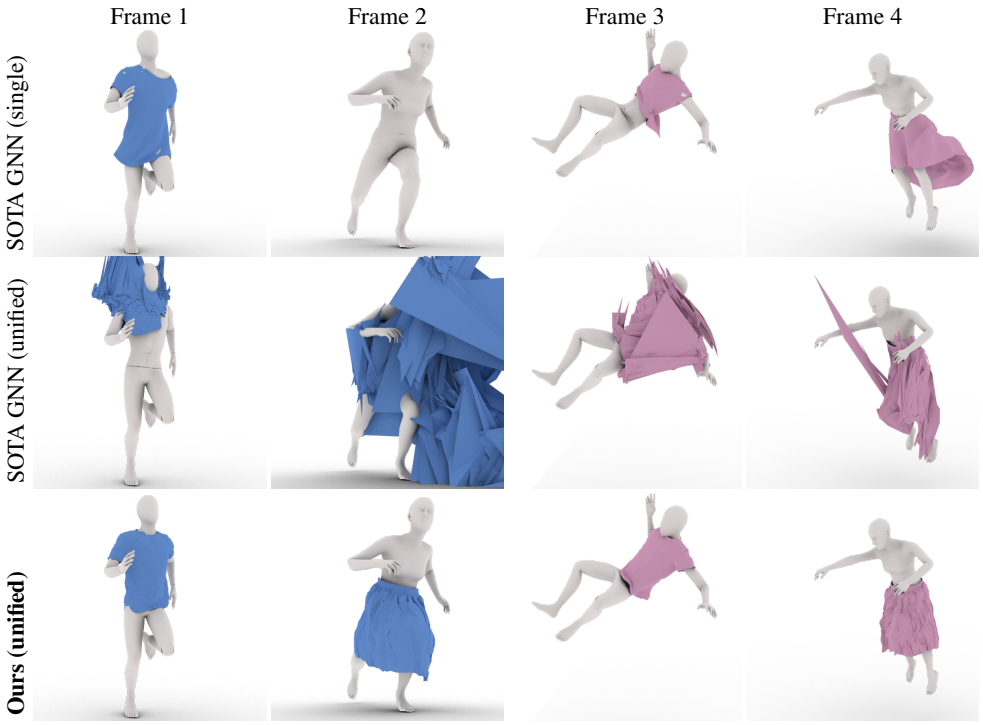

Figure 8: **Unified vs. specialized training on the Human Garment scenario.** Each column shows a different frame. *Top*: SOTA GNN trained unsupervised on Human Garment only. *Middle*: SOTA GNN trained unsupervised on the unified dataset (all three scenarios). *Bottom*: our unified model. The single-scenario SOTA GNN looks reasonable but still deviates from ground truth; the unified-training SOTA GNN degrades sharply; our unified model produces visually plausible results that match or exceed the specialized SOTA GNN.

J Limitation of AR Rollout in Extreme Cases

We observe that our model performs well in the vast majority of cases; however, on extreme cases the autoregressive (AR) rollout can still lead to considerable error accumulation. On an extreme Diverse Object Collision test case, the vertex error rapidly accumulates to $\sim$37 cm within the first 60 frames and never recovers, despite velocity-based inputs and the rollout curriculum. This suggests that for highly nonlinear, contact-rich dynamics, the error accumulation of single-step AR prediction is difficult to eliminate through training strategies alone, motivating one-shot or chunked multi-step prediction paradigms.

To select this case, we rank 100 Diverse Object Collision test sequences by AR rollout error (raw predictions, no CCD post-processing, same protocol as Table 2; split-mean MVE: 9.03 cm) and select the 95th-percentile sequence.

Table 7 reports the MVE of the AR rollout at increasing horizons, and Figure 9 plots the per-frame MVE over the full rollout. The error accumulates rapidly in the early rollout—2.9 cm at frame 1, 5.2 cm at frame 10, and 27.1 cm at frame 20—and then plateaus at $\sim$37 cm, roughly $4\times$ the split-mean level: early deviations feed back into subsequent inputs and drive the cloth toward an incorrect yet self-consistent resting state, in which later predictions are locally stable but globally wrong, and from which the rollout does not recover. This drift persists despite velocity-based inputs and the 5-step rollout curriculum (Sec. 5.1).

Table 7: AR rollout error (MVE, cm) at rollout horizon N on the 95th-percentile Diverse Object Collision test sequence, compared with the average over 15 randomly sampled typical test sequences. The values in parentheses denote the cumulative mean MVE over the first N frames.

	N=60	N=120	Full (240)
Extreme case	36.6 (26.3)	36.5 (31.4)	36.7 (34.0)
Typical (15 samples)	10.9 (8.5)	11.0 (9.7)	10.9 (10.3)

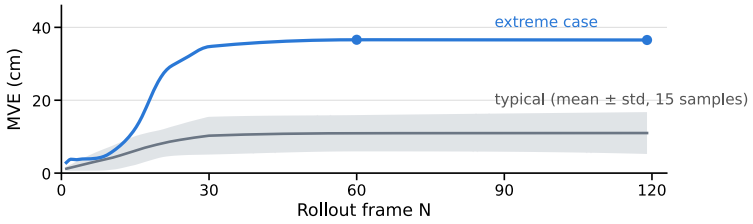

Figure 9: AR rollout error on the 95th-percentile Diverse Object Collision test sequence (blue), compared against 15 randomly sampled typical test sequences (gray: mean $\pm$ one standard deviation). Curves show the per-frame MVE over the first 30 frames and at horizons N=60/120 (markers), smoothly interpolated in between. Typical sequences quickly settle at a bounded error consistent with Table 2, whereas the extreme case accumulates roughly $4\times$ that level.

K Physical Plausibility Metrics

MVE measures average positional accuracy but is insensitive to local, high-frequency behavior. We therefore additionally evaluate four physical quantities on the full 100-sequence test split, using the GIPC ground truth as reference: *strain*, the mean per-edge stretch magnitude relative to the rest length (*net strain* averages the signed values, so a value near zero means local stretch and compression cancel without overall stretching); *wrinkles*, the surface-detail ratio relative to the ground truth; *temporal jitter*, the frame-to-frame variation ratio relative to the ground truth; and *drape/settling*, the cloth speed at rest (cm/s). For the two ratios, 1.00 matches the ground truth.

Table 8: Physical plausibility metrics on the 100-sequence test split. For wrinkles and jitter, 1.00 matches the ground truth; for settling, lower is better. Best result among the learned methods in each row is in bold.

Metric	GT	Ours	SOTA GNN	LayersNet
Strain (net)	0.36% (−0.3%)	**25.95%** (−0.6%)	27.78% (+15.5%)	373.58% (+356.4%)
Wrinkles (ratio)	1.00	1.65	**1.51**	35.25
Temporal jitter (ratio)	1.00	**3.02**	10.04	4.92
Drape/settling (cm/s)	1.60	**4.15**	186.33	623.54

As shown in Table 8, our model jitters least and settles closest to the ground truth among the learned methods, and its net strain (−0.6%) is on the same scale as the reference, so the cloth as a whole keeps its specified size. Its absolute strain, however, remains high: since training is supervised by vertex positions, reconstruction errors appear as zero-mean local stretch and compression and can produce occasional ill-shaped triangles. This error does not accumulate over the rollout and can be reduced with an explicit stretch penalty. Fine-tuning our model with an additional edge-stretch loss for 50k steps (Table 9) lowers the strain by more than 60% and further improves jitter and settling, at the cost of a slight increase in the wrinkle ratio.

Table 9: Effect of fine-tuning with an explicit stretch penalty (50k steps) on the 100-sequence test split. Both columns are measured with the same evaluation pipeline, in which settling is averaged over the final 1 s of each sequence.

Metric	Ours	Ours + stretch fine-tune
Strain (net)	25.95% (−0.6%)	**9.71%** (−1.5%)
Wrinkles (ratio)	**1.65**	1.87
Temporal jitter (ratio)	3.02	**2.59**
Drape/settling (cm/s)	3.83	**3.52**

L More Results

We present additional qualitative comparisons between our method and three baselines (the SOTA GNN [10, 11], MAT [20], and LayersNet [39]) across diverse scenarios. As shown in Fig. 10, our method consistently produces more accurate cloth geometry with fewer artifacts.

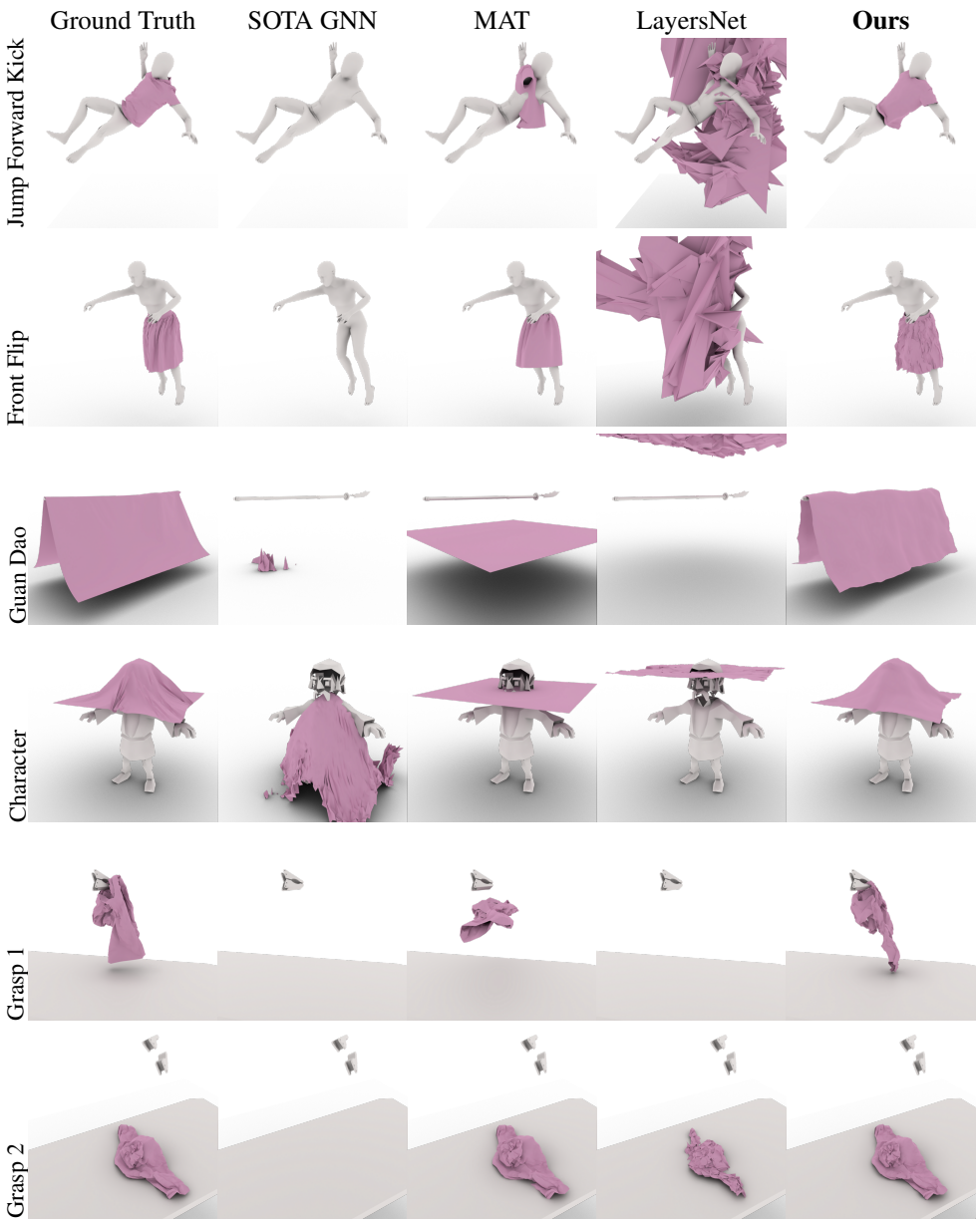

Figure 10: Additional qualitative comparisons. Each row shows a different scenario. From left to right: Ground Truth, the SOTA GNN [10, 11], MAT [20], LayersNet [39], and our method.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 explicitly state the three contributions (a unified Transformer architecture that handles diverse scenarios under a single model with approximately 4–9$\times$ lower error than prior SOTA; a scalable latent-space formulation whose temporal dynamics cost is independent of mesh resolution; and a diverse-scenario penetration-free dataset spanning all three settings that enables a differentiable CCD loss and post-processing), all of which are validated by experimental results in Section 5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: A dedicated "Limitations and Future Work" paragraph is included in Section 6, discussing the lack of explicit material parameter control, error accumulation of single-step autoregressive rollouts in highly nonlinear scenarios (analyzed in Appendix J), and the inability to handle topological changes such as tearing.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper is empirical in nature and does not present formal theorems or proofs.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 5.1 specifies the network architecture, optimizer, learning-rate schedule, batch size, training horizon, and curriculum strategy. Appendix A provides additional architectural details, Appendix B describes the CCD implementation, and Appendix C lists all simulation parameters used to generate the dataset.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The training code and pre-trained checkpoints are publicly released at https://github.com/YuCrazing/ClothTransformer, and the penetration-free dataset is publicly released on Hugging Face (https://huggingface.co/datasets/YuCrazing/ClothTransformer-dataset) under the CC BY 4.0 license, together with instructions for loading the data and reproducing the main results. See also the project page https://yucrazing.github.io/clothtransformer/.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 5.1 reports the optimizer (AdamW), learning-rate schedule, batch size, total training steps, gradient clipping, rollout curriculum, and the 8:1:1 train/val/test split per subset. Ablation-specific configurations are described in Appendix F.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Each model configuration is trained once due to the high computational cost ($\sim$300 NVIDIA H200 GPU hours per full run), so error bars from multiple seeds are not provided. Reported metrics are averaged over thousands of frames in the held-out test split per scenario, providing a tight estimate despite the single training run.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 5.1 reports the total training cost ($\sim$300 NVIDIA H200 GPU hours), and Section 5.4 reports inference timings on a single NVIDIA RTX 4090 (24 GB) for all baselines and our method. Ablation-specific GPU hours are reported in Table 3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The work uses only synthetic cloth simulation data and publicly available 3D asset datasets; it does not involve human subjects, sensitive personal data, or otherwise raise ethical concerns under the NeurIPS Code of Ethics.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: The work targets foundational research on neural cloth simulation for graphics, animation, and robotics applications. We are not aware of direct paths to negative societal impact beyond those generally associated with computer-graphics technology.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The released artifacts (a cloth simulation dataset and a physics-prediction model) do not pose foreseeable misuse risks, so no specific safeguards are required.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All external assets are cited in the paper and used within the terms specified by their original release. Solver and data assets: GIPC [16] (research-use license), SMPL [27] (non-commercial research license), the motion data of [12], the cloth meshes of [18] (MIT), and the 3D collision objects from Objaverse [9] (ODC-By 1.0). Baseline implementations: HOOD [10], ContourCraft [11], MAT [20], and LayersNet [39] are credited per their respective code releases.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Our newly constructed penetration-free cloth dataset is documented in Section 4, with mesh-complexity statistics in Table 1 and full simulation parameters in Appendix C. The dataset is publicly released on Hugging Face (`https://huggingface.co/datasets/YuCrazing/ClothTransformer-dataset`) under the CC BY 4.0 license, with accompanying documentation of the data format and an example loading script.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve research with human subjects, so IRB approval is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs are not used as a component of the core methodology of this research; any use of LLMs has been limited to incidental writing or editing assistance.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.