GarSim: Particle Based Neural Garment Simulator

Lokender Tiwari Brojeshwar Bhowmick
TCS Research, India
lokender.tiwari@tcs.com, b.bhowmick@tcs.com

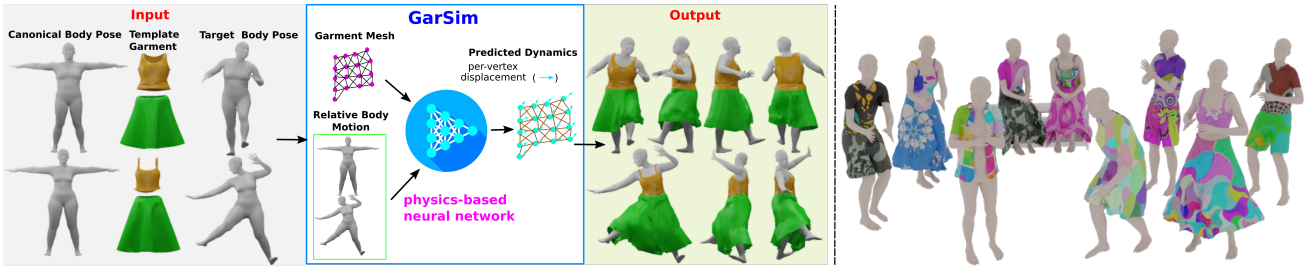


Figure 1: **GarSim** is a learned particle-based neural garment simulator, that can simulate garments of arbitrary types, varying fabric, and topologies on arbitrary body shapes and poses. (**left**) *GarSim* takes a canonical body mesh, a template garment mesh, fabric property, and the target body mesh as input and produces the displacements for the garment vertices to get the final simulated garment. (**right**) Sample textured result of *GarSim* on unseen arbitrary body shapes and poses estimated from a YouTube video. Note: *GarSim* trained only on tops and skirts generalizes well on unseen garment types (trousers, t-shirts, and dresses). Refer to the supplementary material for the video results.

Abstract

We present a particle-based neural garment simulator (dubbed as GarSim) that can simulate template garments on the target arbitrary body poses. Existing learning-based methods majorly work for specific garment type (e.g. top, skirt, etc) or garment topology, and needs retraining for a new type of garment. Similarly, some methods focus on a particular fabric, body shape, and pose. To circumvent these limitations, our method fundamentally learns the physical dynamics of the garment vertices conditioned on underlying body shape, motion, and fabric properties to generalize across garment types, topology, and fabric along with different body shape and pose. In particular, we represent the garment as a graph, where the nodes represent the physical state of the garment vertices, and the edges represent the relation between the two nodes. The nodes and edges of the garment graph encode various properties of garments and the human body to compute the dynamics of the vertices through a learned message-passing. Learning of such dynamics of the garment vertices conditioned on underlying body motion and fabric properties enables our method to be trained simultaneously for multiple types of garments (e.g., tops, skirts, etc) with arbitrary mesh resolutions, varying topologies, and fabric properties. Our experimental results show that GarSim with less amount of training data not only outperforms the SOTA methods on challenging CLOTH3D dataset both qualitatively and quantitatively, but also works reliably well on the unseen poses obtained from YouTube videos, and give satisfactory results on unseen cloth types which were not present during the training.

1. Introduction

Simulating garments on arbitrary body poses is crucial for many applications related to 3D content creation, virtual try-on, etc. Physics-Based Simulation (PBS) has always been a go-to option to accurately and realistically simulate garments on a target body pose. However, PBS methods are computationally expensive and require experts with good domain knowledge to govern the simulation quality.

To reduce the manual intervention and increase the speed, several attempts[21, 28, 7, 6, 29, 9, 12], have been made to learn garment deformations using the ground truth PBS data. Despite the advantages, learning-based methods have several limitations such as: *fixed garment topologies:* TailorNet[21] and DeepDraper[28] represent garment of different sizes (e.g., t-shirt) with the same number of vertices which limits their applicability for the loose garments

Table 1: Comparison of the characteristics of our method with the prior methods. In *GarSim* we consider the relative motion of the target body w.r.t the body in the canonical pose. *Though DeePSD has shown results on skirts in their original paper, they have also mentioned in their limitation section, that since they do not consider motion dynamics in their formulation they fail to predict correct deformations for the loose garments.

	Body Factors			Garment Factors				
	varying pose	varying shape	Relative body motion	multiple garments /outfits	varying topo-logies	fabric aware	varying garment mesh reso-lutions	loose garment (e.g., long skirts)
TailorNet[21]	✓	✓						
DeepDraper[28]	✓	✓						
GarNet[10]	✓	✓						
LVTON[25]	✓	✓						
PBNS[6]	✓					✓		
SNUG[26]	✓	✓				✓		
VirtualBones[20]	✓		✓			✓		✓
DeePSD[7]	✓	✓		✓	✓	✓		✓*
GarSim (ours)	✓	✓	✓	✓	✓	✓	✓	✓

as they require more vertices to represent, *garment representation:* methods like [21, 28] represent garments in the PCA space are difficult to generalize to other complex garments during test time. *fixed cloth type:* several methods like [21, 9, 28, 6, 25, 26, 20], are trained only for a single garment type, *fixed body shape and/or pose:* training for a single body shape or pose for garment simulation limits the applicability of the methods like [12, 29, 6, 20]. Moreover, majority of the existing methods [21, 7, 28] have shown their limitations for loose garments e.g., long skirts and are specific to a fabric. A comparison of the characteristics of the prior methods with ours is shown in Tab. 1.

Recent works like DeePSD [7] and Garnet++ [9] that do not consider body dynamics and are static in nature, have shown limitation in their paper about handling loose garments like long skirts like the one shown in Fig.1. Moreover, Garnet++ [9] is an outfit-specific method and does not generalize to the varying topology of the garment.

In this paper, we consider garment deformation as a physical phenomenon conditioned on the fabric property, body shape, pose, and relative motion of the target body w.r.t the body in the canonical pose. Since the relative body motion is also considered for predicting the vertex displacement, our method can handle the simulation of the loose garments effectively. Also, a key property of garment mesh deformation is that the vertices do not move (deform) in isolation, rather the movement is highly influenced by the force or motion of its neighboring vertices connected by the mesh edges and the distance with respect to the body. This relational inductive bias [4] property has been successfully used in modeling complex physical phenomena such as fluid dynamics, deformable materials, etc. Therefore, we represent the state of the garment as particles (corresponds to the garment vertices) and embed it into a latent garment graph. We encode various factors affecting the garment deformation on the target body pose into the graph nodes and edges. We exploit the relational inductive bias property using a learned message-passing module to update the node and edge features of the garment graph in a multi-step process. This allows propagating the relation representation or passing the forces to the neighboring nodes. Finally, the updated latent garment graph nodes features are decoded into per-vertex dynamics i.e., displacement.

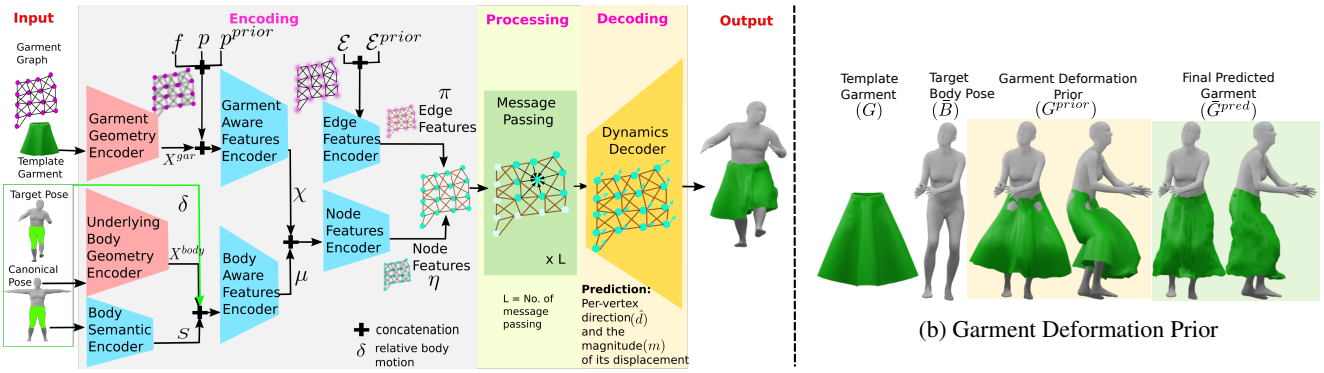
Another issue faced by the earlier methods is learning on high-resolution garment meshes, which increases the overall training time. Reducing the resolution degrades the quality of their results. Since, our method *GarSim* learns vertex level dynamics, it's performance doesn't deteriorate by training on low resolution garment meshes and outperforms the state-of-the-art methods such as DeePSD[7], PBNS[6], TailorNet[21], DeepDraper[28].

Therefore, our main contributions in this paper are:

- A *unified garment simulator* that can simultaneously learns deformation of multiple type of garments (e.g., tops, skirts etc.) of varying topologies and fabrics conditioned on the underlying body shapes, pose and motion w.r.t the canonical pose.
- A particle-based approach that exploits the relational inductive bias[4] in the garment data and helps generalize to unseen body shapes, poses, and garment types.
- *GarSim* can be trained and tested on the arbitrary resolution garment meshes, hence, it can be adaptive to the limitation of the underlying computation hardware without any significant change in the accuracy.

2. Related Work

Traditional physically-based simulator [14, 18, 3, 30] have been successfully used to get a realistic cloth deformations. While these simulators output high-quality cloth simulations, they are computationally very expensive and require expert intervention. Recent learning based simulators [15, 28, 7, 6, 10, 5, 21, 26, 20, 22, 24] aim to reduce the computational time and the expert intervention both. To this end, Lahner et al. [12] propose a learned Pose Space Deformation[13] for garments conditioned on temporal features; LVTON [25] learned *per-garment* non-linear mapping for Pose Space Deformation. However, this methods require retraining for new garments, and doesn't consider fabric property and relative body motion into account. TailorNet [21] and DeepDraper[28] use a parametric representation of the garment and hierarchically learn low-frequency and garment-specific high-frequency displacement due to the body pose, shape, and garment style. The representation of garments in these methods restricts them to be topology



(a) *GarSim* Overview

(b) Garment Deformation Prior

Figure 2: (a) *GarSim* takes a template garment draped on the canonical body pose, a target body pose as an input, and outputs the simulated garment on the target body pose. It operates in a three-step framework, (1). encoding step Sec. 3.2, (2). processing step Sec. 3.3, and (3). decoding step Sec. 3.4. The green mask on the body corresponds to the garment-aware body semantic vector q that consists of vertices (subset) of the body that will affect the deformation of the respective garment. (b). Garment deformation prior estimated using the Linear Blend Skinning (LBS) [17] method as explained in Sec. 3.1.

invariant. Furthermore, both [21] and [28] are specific to a particular garment type, and hence require retraining for every new garment type. CLOTH3D[5], DeePSD[7] require huge volume of data to train their models. Moreover, these methods do not work for loose garments (e.g long skirts) and the same is outlined by the authors as their limitation. PBNS[6] alleviates the need for huge data in training time but their model is outfit and body specific. SNUG[26] have shown improvement over PBNS by making it invariant to the human body. Both SNUG[26] and PBNS[6] still require training a new model for every new garment and have shown their limitation for loose garments e.g., long skirts. The methods [21, 10], where the primary training loss is the $L2$ loss with the GT data are generally biased to produce smooth results. Methods such as [7, 6] also apply a few physics-inspired losses with a belief that the underlying network (primarily the MLP's) will be able to learn the physics of garment deformation. Such methods ignore to leverage the fact that during deformation garment vertices physically interact with each other and induce a local bias in the deformation. This relational nature of garment data is largely ignored by these methods.

A recent method [20], uses the concept of virtual bones to animate loose garments based on the underlying body sequence. However, this also has limitations in handling a single garment at a time i.e., a trained model is the garment and the human body. It cannot handle variable topologies and resolutions of garment meshes. Hence, for every such variation, it requires retraining a new model.

3. GarSim Overview

Our method *GarSim* learns to simulate garments on arbitrary body poses using the concepts from particle-based modeling (Sec. 3.1). We represent the state of the garment as particles (corresponds to the garment vertices) which encodes various garment and body-specific factors that affect its dynamics. In our case, the dynamic is the vertices displacements (direction and the magnitude), which is be computed based on the particles' interaction within their local neighborhoods. *GarSim* takes a template garment draped on the canonical body pose, and the target body pose as inputs and produces the garment draped on the target body pose. It is a single-step simulator that takes into account the relative motion of the target body pose with respect to the canonical body pose (T-pose) as shown in Fig. 1. Fig. 2 shows that our *GarSim* operates in a three-step process. During the first (*encoding*) step (Sec. 3.2), it embeds the particle-based state representation of the garment into the latent *garment graph*. It encodes the various garment and the body-specific factors (Sec. 3.2) into the nodes and edges of the garment graph. The encoded garment graph is then passed to the second step called *processing* step (Sec. 3.3), where a learned message-passing algorithm is used to update the state (node features), and their relationships (edge features) (Algorithm 1), and output the final updated latent garment graph. In other words, each message passing step helps in passing the forces to the neighboring particles. We use multi-step message passing for long-range effect propagation [15, 16]. Finally, the updated latent garment graph is passed to the third step called *decoding* step (Sec. 3.4) that predicts the dynamics i.e., the direction and the magnitude of the displacement for each garment vertex from the respective nodes of the final updated latent garment graph.

In what follows, we describe the particle-based modeling followed by the encoding, processing, and decoding steps in detail.

3.1. Particle-based Garment Dynamics Modeling

Let G be the set vertices of a template garment mesh draped over a 3D canonical human body mesh B. After time t the body has moved and the new body mesh is denoted by $\bar{B}$. The human body motion will induce a secondary motion on garment G which results in a new deformed garment mesh $\bar{G}$. The vertices of the garment mesh after time t can be modeled using the following relation $\bar{G}_i = G_i + d_i$, where d_i is the displacement of the vertex i after time t. Majority of the recent works [21, 28, 25, 27] do garment skinning as a final or post-processing step. To do this, they assume that the garments closely follow the underlying body motion and borrow blend shape weights from the Linear Blend Skinning method [17] for each vertex of the template garment from the closest body vertex in the canonical pose. While this assumption simplifies the problem and works well for tight body-hugging garments, it drastically fails in the case of loose garments such as long skirts and dresses (See. prior mesh in Fig. 2(b)). Additionally, doing skinning at the end often leaves the artifacts in the final deformed garments. In our method, we use this strategy *only* to estimate a *garment deformation prior* G^{prior} as it helps the network to learn corrections on top of it. We henceforth find the displacement with respect to the G_i^{prior} instead of the template garment G (Eq. 1).

$$\bar{G}_i = G_i^{prior} + d_i = G_i^{prior} + \hat{d}_i m_i \tag{1}$$

Here, displacement d_i is divided into the direction of displacement $\hat{d}_i$, and its magnitude m_i i.e., $d_i = \hat{d}_i m_i$. An ablation study on the choice of splitting d_i into $\hat{d}_i$ and m_i pair is presented in Sec.4.2. In Fig. 2(b) we can observe strong artifacts in (G_i^{prior}) such as a big bulge in the front of the skirt with unrealistic spread out along with collisions, and its correction by our method. Both the $\hat{d}_i$ and the m_i are influenced by several garment and body specific factors, that we define in the next section.

3.2. Encoding Step

Garment Specific Factors: Garment Geometry: We map the template garment vertices to a high-dimensional per-vertex geometric features using a geometry encoder. Garment Fitting: Methods like [7, 5] use a 2 dimensional vector to denote the tightness of a garment. But it is difficult to obtain such a vector while testing on new unseen garments. To alleviate this issue, we capture tightness at the vertex level by measuring the garment vertex's (G_i) distance from the closest body (B) vertex which can be computed for any garment during the test time. The relative position of a garment vertex G_i with respect to the closest body vertex B_{q_i} can be computed as $p_i = G_i - B_{q_i}$. Similarly we can compute $p_i^{prior} = G_i^{prior} - \bar{B}_{q_i}$. Here q_i is the index of the body vertex closest to the garment vertex G_i. Throughout this paper, we consider *only* the subset of the human body that will affect the deformation of the garment, we denote this by a set of vertices $q = \{q_1, q_2, ..., q_i, ..q_{N^n}\}$ (see. Fig. 2). Fabric Type: The garment fabric is encoded as a one-hot vector denoted by f. We consider four fabrics: *leather*, *denim*, *silk* and *cotton*.

Body Specific Factors: Body Geometry: Similar to the garment geometry, we encode the geometry of the partial body represented by a subset of vertices in q using the same geometry encoder. Relative Body Motion: We compute the relative body motion between *canonical* (B) and the *target body*($\bar{B}$) pose as a relative motion vectors of the underlying body vertices as $\delta_i = \bar{B}_{q_i} - B_{q_i}$. Garment Aware Body Semantic: Garment at the different parts of the body deforms differently. For instance, garments like a tube top and skirt having similar geometries deform differently as there are less articulated body parts under the tube top compared to the skirt. Hence, we condition the garment deformation on the body semantics. We obtain the latent body semantic vector S for each garment type, by passing a binary vector (1 indicates that it's index is the member of the set q) of dimension equal to 6890 (the number of full-body vertices in SMPL[17] body model) to a MLP and get the encoded representation S.

3.2.1 Garment Graph

We create a garment graph and encode the above factors as the node and the edge features.

Node Features: We obtain the garment graph node features η by fusing the garment aware latent features χ with the body aware latent features μ using a node feature encoder.

Garment Aware Features (χ): We embed all the garment-specific information into the latent feature space (χ) by passing the concatenated per-vertex geometric features (X^{gar}), the fabric (f), the garment fitting (p and p^{prior}) to the garment aware features encoder. (see Fig. 2).

Body Aware Features (μ): Similar to above, we embed all the body-specific information into the latent feature space (μ) by passing the concatenated per-vertex geometric features (X^{body}), the relative motion (δ), the latent body semantic vector (S) to the body aware features encoder.

Edge Features: Let G_j and G_i be the end vertices of an edge of the garment graph, we compute the relative position as $\mathcal{E}_{ij} = G_j - G_i$. Similarly, we can also compute $\mathcal{E}_{ij}^{prior} = G_j^{prior} - G_i^{prior}$. We pass the concatenation of $\mathcal{E}_{ij}$ and $\mathcal{E}_{ij}^{prior}$ to a encoder to get the edge features π.

The garment graph with the learned node features (η) and edge features (π) is then passed to the processing step. With both the $\mathcal{E}_{ij}$ and $\mathcal{E}_{ij}^{prior}$, the network will learn to correct the significant artifacts in the deformation prior due to the elongation of the edges etc. present in the $\mathcal{E}_{ij}^{prior}$.

Our method learns to predict the direction and the mag-

Algorithm 1: Message Passing

1 **for** $l \in \{1...L\}$ **do**
2 **for** $k \in \{1...N^e\}$ **do**
3 $\pi'_k \leftarrow \Psi_{edge}(\pi_k, \eta_{r_k}, \eta_{s_k})$
4 **end**
5 **for** $i \in \{1...N^n\}$ **do**
6 $\lambda'_i = \{(\pi'_k, r_k, s_k)\}_{r_k=i, k=1:N^e}$
7 $\bar{\pi}'_i = \Omega^{edge \rightarrow node}(\lambda'_i)$
8 $\eta'_i = \Psi_{node}(\bar{\pi}'_i, \eta_i)$
9 **end**
10 $\pi_k \leftarrow \pi'_k \quad \forall k \in \{1...N^e\}$
11 $\eta_i \leftarrow \eta'_i \quad \forall i \in \{1...N^n\}$
12 **end**

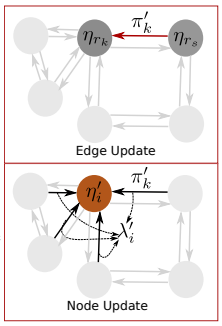


Figure 3: Node and Edge updates.

nitude of the displacement ($\hat{d}_i$ and m_i) for each garment vertex conditioned on the above factors using Eq. 2.

$$\hat{d}_i, m_i \leftarrow \textit{GarSim}(B, \bar{B}, q, \delta, G, \mathcal{E}, \mathcal{E}^{prior}, f, p, p^{prior}) \tag{2}$$

3.3. Processing Step

We process the encoded garment graph through a message-passing network to flow the relational entities responsible for the garment deformation from one node to another through the edges connecting to them. Our message-passing network uses the abstract framework of Graph Network block [4], with two update and one aggregation functions. Such message passing is shown to be effective in [22, 24] for modeling physical systems. Our complete message-passing algorithm is shown in Alg. 1. The update function Ψ_{edge} is applied for each edge (Alg. 1, line 3). It takes the edge feature π_k of the k^{th} edge, and the node features of its connecting nodes (η_{r_k} and η_{s_k}) as an input and output the updated edge feature π'_k (visually shown in Fig. 3). We accumulate the updated edge features along with the indices of the edge end nodes in $\lambda'_i = \{(\pi'_k, r_k, s_k)\}_{r_k=i, k=1:N^e}$ (Alg. 1, line 6). The aggregation function $\Omega^{edge \rightarrow node}$ is applied to λ'_i to aggregate the updated edge features for the edges connected to the i^{th} node into the $\bar{\pi}'_i$ (Alg. 1, line 7). The update function Ψ_{node} is applied per-node (Alg. 1, line 8) where it takes the aggregated edge features $\bar{\pi}'_i$, it's own node features and produce the updated node feature η'_k (visually shown in Fig. 3). The process repeats L ($L = 5$ in our experiments) times and then the next message passing takes the updated node and edge features (Alg. 1, line 10-11). The update functions Ψ_{edge} and Ψ_{node} are implemented using MLP's. The aggregation function $\Omega^{edge \rightarrow node}$ is an average function.

3.4. Decoding Step

The decoding step take the processed latent garment graph node features and for each vertex it predicts its dynamics i.e., the direction $\hat{d}_i^{pred}$ and the magnitude m_i^{pred} of its displacement. The position of the deformed i^{th} garment vertex is then computed as $\bar{G}_i^{pred} = G_i^{prior} + \hat{d}_i^{pred} m_i^{pred}$.

3.5. Training Losses

We train the *GarSim* in an end-to-end fashion using a combination of supervised and unsupervised losses.

Supervised Losses: We compute the $L2$ loss between the predicted ($\bar{G}_i^{pred}$) and the ground-truth garment vertices ($\bar{G}_i$) in the target pose. The total $L2$ loss is the weighted sum of the $L2$ loss of *non-pinned* and *pinned* vertices as:

$$\begin{aligned}\mathcal{L}_{L2} = &\frac{(1-\beta)}{N - |\mathcal{I}|} \sum_{i=1}^{N} (1 - \mathcal{I}_i) ||\bar{G}_i - \bar{G}_i^{pred}||_2 \\ &+ \frac{\beta}{|\mathcal{I}|} \sum_{i=1}^{N} \mathcal{I}_i ||\bar{G}_i - \bar{G}_i^{pred}||_2\end{aligned} \tag{3}$$

Here, $\mathcal{I}$ is a binary index vector of length equals to the total number of vertices/nodes N. The $\mathcal{I}_i = 1$ indicates the i^{th} template garment vertex needs to be pinned and should not move too far from the body. $|\mathcal{I}|$ is the total number of pinned vertices, and β balances the weight between pinned and non-pinned losses. We empirically fixed $\beta = 0.4$.

Unsupervised Losses: *Mesh Smoothing Loss:* We use uniform Laplacian smoothing loss [19] to enforce smoothness on the surface of the predicted garment mesh $\bar{G}^{pred}$. Here, Δ denote the Laplacian smoothing function

$$\mathcal{L}_{sm} = \Delta(\bar{G}^{pred}) \tag{4}$$

Mesh Normal Consistency Loss: We enforce the consistency between the neighboring faces of the predicted garment mesh to prevent extreme deformations and improve the quality of the garment mesh surface. Given F_1 and F_2 are the two adjacent faces of the garment mesh and $\hat{N}_1$ and $\hat{N}_2$ are their respective face normals. The normal consistency loss between these two faces is given by $\mathcal{NC}(F_1, F_2) = 1 - \frac{\hat{N}_1 \hat{N}_2}{||\hat{N}_1|| ||\hat{N}_2||}$. The total consistency loss can be computed as in Eq. 5, where, $\mathcal{F}$ is a set containing all neighboring face pairs and M is the number of such pairs.

$$\mathcal{L}_{NC} = \frac{1}{M} \sum_{\forall (F_i, F_j) \in \mathcal{F}} \mathcal{NC}(F_i, F_j) \tag{5}$$

Body Garment Collision Penalty Loss: To ensure predicted garment mesh is free from body garment collision we apply this penalty loss as shown in Eq. 6. Similar penalty loss has been found effective in [28, 7, 29, 10]. In our case this reduce total collisions from $\sim 11\%$ to $\sim 0\%$ during training. Here, $\bar{B}_i$ is the body vertex closest to the predicted garment vertex $\bar{G}_j^{pred}$, and $\mathcal{N}_i$ is its normal.

$$\mathcal{L}_{col} = \frac{1}{N} \sum_{i=1}^{N} \max(-\mathcal{N}_i(\bar{B}_i - \bar{G}_j^{pred}), \epsilon) \tag{6}$$

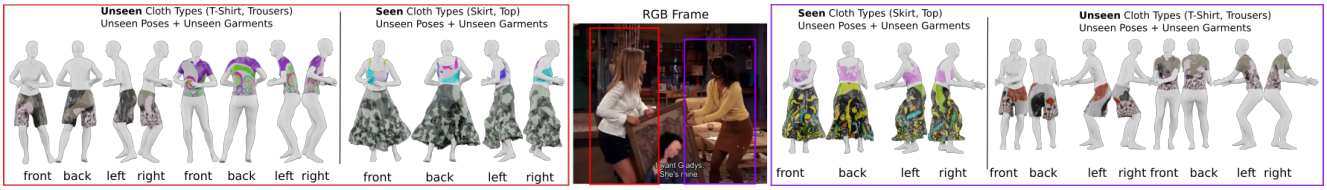


Figure 4: Textured qualitative results of seen (tops and skirts) and unseen (t-shirt and trousers) garment types on unseen body shapes and poses from a YouTube video. By seen we mean the type of garments available in the training set. Refer Sec. 4.1

The total training loss is the weighted sum of the supervised and unsupervised losses. Here, α's denote the weights of the respective losses.

$$\mathcal{L}_{tot} = \alpha_{L2}\,\mathcal{L}_{L2} + \alpha_{sm}\,\mathcal{L}_{sm} + \alpha_{NC}\,\mathcal{L}_{NC} + \alpha_{col}\,\mathcal{L}_{col} \quad (7)$$

4. Experiments

In this section, we show the quantitative and qualitative results of *GarSim* and compare with the SOTA methods.

Dataset: We evaluate *GarSim* on a publicly available CLOTH3D[5] dataset. It has high variability in the garment styles, topologies, body shapes, poses, and the fabric properties. Specifically, we use a *subset* of the official training and test set of the CLOTH3D[5] dataset. We select the sequences with at-least 50 frames where a female is wearing a *skirt* and a *top* of varying fabrics (*leather*, *denim*, *cotton* or *silk*). Our training and test subset consists of a total $38k$ and $10k$ frames respectively.

4.1. Generalization Capability

We demonstrate the generalization capability for *GarSim* to (a). unseen body shapes, poses, and garment types, (b) prediction at arbitrary garment mesh resolution.

Unseen Body Shapes, Poses and Garment Types: To demonstrate the generalization on unseen body shapes, poses, and garment types, we select a few frames from a random YouTube video and estimate the body shape and pose using PARE[11]. We show the results of *GarSim* for the loose skirt and top on the unseen body shapes and poses in Fig. 4 and 5, and textured result on seen (top and skirt) and unseen (t-shirt and trousers) garment types in Fig. 4. The physically plausible satisfactory predictions of both seen and unseen garment type shows *GarSim* is accurately able to learn the vertex level dynamics.

Prediction at arbitrary resolution: In *GarSim* we follow particle-based modeling of a garment. It allows flexibility to test garments at arbitrary resolutions. It is because of the learned-message passing, that the garment vertices as nodes in the latent garment graph are able to transfer forces or propagate long-range effects to neighborhood nodes connected through mesh edges at arbitrary resolutions. We have reported qualitative evaluation on arbitrary resolutions in Table. 3 and a sample qualitative results in Fig. 6.

Figure 5: Qualitative results of loose skirts on *unseen body shapes and poses* from a YouTube video.

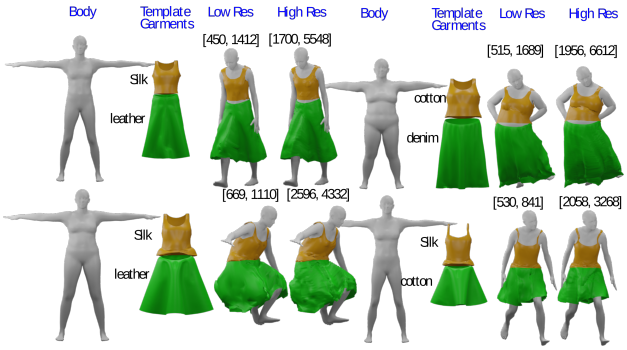


Figure 6: Sample qualitative results of *GarSim* . Notice the alignment of the garments on the body due to motion, and the wrinkles and folds in the skirts around the legs. The X and Y in [X,Y] denote the number of vertices of top and the skirt respectively.

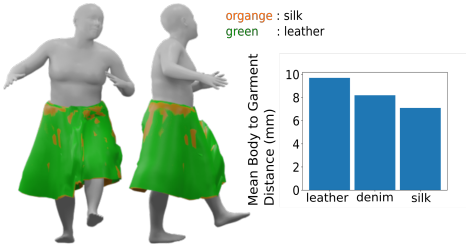


Figure 7: Fabric aware predictions (refer Sec. 4.2).

4.2. Analysis and Ablation Study

We analyze *GarSim* from two perspectives (1) fabric aware predictions, (2) corrections over deformation prior. We also show an ablation on alternate modeling choices. **Fabric Aware Predictions:** To demonstrate that *GarSim* learns fabric-aware deformations, we select a subset of 3000 frames of skirts from the test set and simulate them in three significantly different fabrics types *leather*,

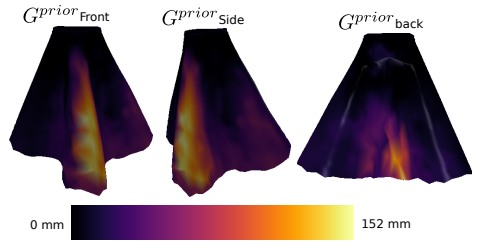


Figure 8: Corrections plot over the garment deformation prior: Note the corrections made by *GarSim* around the big bulge. The correction values range from 0 to 152 mm.

denim and *silk*. To analyze the influence of the different fabric types on garment deformation quantitatively, we compute the mean distance of all the garment vertices to their respective closest body vertices. Fabrics based on their stiffness can be ordered as leather $>$ denim $>$ silk, and intuitively it is expected that the mean garment to body distance should also follow this order. The chart in Fig. 7 empirically verifies this order. The prediction of the leather skirt (green) on top of the silk skirt (orange), visually verifies that *GarSim* produces the fabric aware deformations.

Correction Over Garment Deformation Prior: Garment deformation prior G^{prior} generally contains multiple artifacts, such as the big bulge in the front of the G^{prior} as shown in Fig.2(b). We show the magnitude of per-vertex corrections made by *GarSim* over the deformation prior in Fig.8. The Euclidean distance between the corresponding vertices of G^{prior} and the $\bar{G}^{pred}$ are plotted on top of the G^{prior} mesh. Notice the high magnitude of the corrections made by *GarSim* around the big bulge artifact. The final predicted garment $\bar{G}^{pred}$ is also shown in Fig.2(b).

Ablation on Alternate Modeling Choices: While we predict the magnitude and direction of the displacement for each vertex separately, there exist two alternate choices. The first is a direct prediction of the displacements from the decoder, and the second is the direct prediction of the vertex positions by applying $L2$ loss with the ground-truth data. Table 2 shows the evaluations of these two alternate choices. We find that predicting magnitude and direction separately gives better results than the other choices. This is due to the fact that vertices of two garments with similar geometry under the same body semantic may have similar direction of displacements, but their magnitudes may depend on the fabric property. For example, as we can see in Fig.7 the direction of displacement of vertices of the same skirt under two different fabric types is similar but the magnitude of displacement of vertices under silk fabric is on average less than the leather fabric.

4.3. Comparison with DeePSD[7]

The closest SOTA method to ours is DeePSD[7] (Refer table 1). Hence, we show a detailed comparison with this method. Its code[1] is publicly released without the pre-trained models. So for a fair comparison, we train it using the same subset of training data as *GarSim* . **Note:** in their original DeePSD is trained on $\sim$ **7500** simulated sequences with *7 different garment types*. Here, our training subset consists of **only 198** simulated sequences with *only two garment types*, top and skirt. Both *GarSim* and DeePSD are trained jointly for tops and skirts.

	Direction+ Magnitude	**Direct Displacement**	**Direct Vertex Positions**
Euclidean Error (mm)	40.01	45.695	64.695
Normal Consistency	0.08	0.09565	0.10165
Smoothness	0.01125	0.01225	0.03225
Collision (%)	2.35	4.9	10.9

Table 2: Ablation study on the alternate choices. Values are average of both tops and skirts for the low resolution.

Training on Low-Resolution Garment Meshes: As opposed to methods like DeePSD, PBNS, etc, we train *GarSim* on low-resolution garment meshes obtained by simplifying[8] the original garment meshes by a factor of ~ 4 retaining the overall garment geometry.

We show the evaluation of *GarSim* on both the low and high-resolution garment meshes. DeePSD is evaluated only on the high-resolution meshes. We have reported the evaluation of tops and skirts separately in Table. 3 to show the improvements in both the body-hugging garments (tops) and loose garments (skirts).

Evaluation Metrics: We use four quantitative evaluation metrics. *1. Euclidean error against the ground-truth. 2. Surface smoothness 3. Normal consistency* and *4. Collision percentage*. For smoothness and normal consistency score we use the Eq. 4 and Eq. 5 respectively.

	Low Res (**GarSim**)		High Res (x4) Tops		High Res (x4) Skirts	
	Tops	Skirts	**GarSim**	**DeePSD**	**GarSim**	**DeePSD**
Euclidean Error	21.75	58.27	**18.29**	39.63	**56.47**	77.41
Normal Consistency	0.073	0.087	**0.055**	0.1378	**0.074**	0.1399
Smoothness	0.0112	0.0113	**0.0026**	0.0032	**0.0039**	0.0042
Collision (%)	2.1	2.6	**1.7**	19.77	**1.2**	10.15

Table 3: Comparison with DeePSD on the test set.

The results in the Table. 3 shows *GarSim* despite being *trained on low resolutions* meshes outperforms DeePSD (*trained on high resolutions*) in all metrics. It shows that *GarSim* trained with a *small amount of data* accurately learns the physics of garment deformation. This becomes possible mainly due to the conditioning of the garment deformation on the relative body motion, and learned message-passing to propagate long-range effect across multiple neighboring vertices. Such, constraints are missing in the DeePSD formulation, hence leading to the poor simulation results for both loose and body-hugging garments. Additionally, our particle-based vertex level modeling of gar-

[1] https://github.com/hbertiche/DeePSD

ments deformation makes our method topology and mesh resolution invariant. This allows *GarSim* to be trained and tested on arbitrary garment mesh resolutions and generalize on unseen garment types like t-shirts, dresses and trousers as shown in Fig. 1, and Fig. 4). A qualitative comparison in Fig. 6 further validates that DeePSD is data hungry and if trained with a small amount of data it leads to unrealistic garment deformations with collision to the underlying human body.

4.4. Comparisons With the Other Related Works

In this section, we compare *GarSim* qualitatively with PBNS[6], TailorNet[21] and DeepDraper[28].

With PBNS [6]: PBNS trains a single model for a specific human subject and garment pair, while *GarSim* train a single model jointly for multiple subjects and multiple garment types. Due to high variability in CLOTH3D, training PBNS for every possible subject and garment pair or different topology garments is infeasible. Hence, a fair comparison would not be possible. The pre-trained models of PBNS are also not available. However, for a reference qualitative comparison, we pick one subject and a skirt (*loose* garment) from the Cloth3D test set with a random set of 1000 significantly different poses and train PBNS using their official code [2]. We show the result of two arbitrary selected poses in Fig.9, where we can observe the cloth quality predicted by PBNS is not good (irregular mesh surface); the straps in PBNS are floating in the air, and garments are colliding with the body, while *GarSim* produces significantly better pose aware deformations and straps are resting on the body. The difference in the result of PBNS reported here and in their original paper suggests that a significantly large pose set is required to get better quality results. However, there is no fixed rule to get a correct set of poses for a given garment and body pair to train the PBNS.

With TailorNet[21] and DeepDraper[28]: A fair comparison with both TailorNet and DeepDraper is infeasible due to the following reasons. 1) Both are *single garment type methods*, while *GarSim* learns for *multiple garment types at a time*. 2) They assume f*ixed vertex order and topology*, while we assume a*rbitrary vertex order and topologies*. Also, adapting their methods for the CLOTH3D dataset is infeasible due to their design choices (e.g., fixed 20 shape-style aware predictors in TailorNet). And it is also not possible to train *GarSim* on their TailorNet dataset, as crucial information such as fabric property, and ground-truth canonical pose (T-pose) simulations are not available. We still show a reference qualitative comparison with TailorNet and the DeepDraper as follows: We train the TailorNet[3] and DeepDraper using the training split of the TailorNet Skirt dataset[2]. While training DeepDraper we follow the skirt modeling strategy as suggested by the TailorNet. At test time, we find Skirts from the TailorNet test set which are similar to the Skirts used for the *GarSim* , and evaluate on the same using the test poses. For a fair comparison, we show the results of both the TailorNet and the DeepDraper before post-processing in Fig.9.

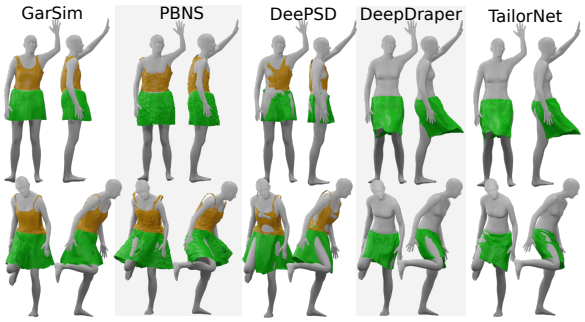


Figure 9: Qualitative Comparison with PBNS[6], DeePSD[7], DeepDraper[28] and TailorNet[21]. Notice the non-smooth surface, floating straps and collisions in the PBNS and DeePSD. Note: for a fair comparison the results of the TailorNet are shown before post-processing, hence it differs from their original paper. See Sec. 4.4.

5. Implementation Details

We implement *GarSim* using PyTorch[1]. We empirically fix α_{L2} and α_{col} equals to 1, α_{sm}=0.01 and α_{NC}=0.0001, and apply $\mathcal{L}_{col}$ only after other losses converge. Geometry encoders are inspired by PointNet++[23] and other encoders are implemented using MLP's. Please refer to the supplementary material for a detailed *GarSim* architecture. The inference time of *GarSim* for the garments with total vertices $\sim$ 600 and $\sim$2400 is 0.1 and 1.0 seconds respectively, which can further be reduced by optimizing the code and paralleling the encoding process.

6. Conclusion

We have presented a novel particle-based garment simulator that can simultaneously learn deformations of multiple garment types conditioned on underlying body shape, pose, motion w.r.t the canonical pose, and fabric properties. We overcome several limitations of the SOTA methods such as joint training for multiple garments of varying topologies, varying fabrics, loose garments, and arbitrary resolution training and testing. We show the benefit of leveraging the relational nature of garment data that significantly improved results compared to the other SOTA methods. We also show *GarSim* learns the fabric and pose-aware deformation of the garments. While we are able to largely handle the body-garment collisions, we observe a few cases of cloth-cloth collisions in the extra long full-length Skirts. This is a limitation that we share along with all SOTA methods.

[2] https://github.com/hbertiche/PBNS
[3] https://github.com/chaitanya100100/TailorNet

References

[1] Pytorch. https://pytorch.org/.

[2] Tailornet dataset. https://github.com/zycliao/TailorNet_dataset.

[3] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 43–54, 1998.

[4] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[5] Hugo Bertiche, Meysam Madadi, and Sergio Escalera. Cloth3d: clothed 3d humans. In *European Conference on Computer Vision*, pages 344–359. Springer, 2020.

[6] Hugo Bertiche, Meysam Madadi, and Sergio Escalera. Pbns: physically based neural simulation for unsupervised garment pose space deformation. *ACM Transactions on Graphics (TOG)*, 40(6):1–14, 2021.

[7] Hugo Bertiche, Meysam Madadi, Emilio Tylson, and Sergio Escalera. Deepsd: Automatic deep skinning and pose space deformation for 3d garment animation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5471–5480, 2021.

[8] Paolo Cignoni, Claudio Montani, and Roberto Scopigno. A comparison of mesh simplification algorithms. *Computers & Graphics*, 22(1):37–54, 1998.

[9] Erhan Gundogdu, Victor Constantin, Shaifali Parashar, Amrollah Seifoddini, Minh Dang, Mathieu Salzmann, and Pascal Fua. Garnet++: Improving fast and accurate static 3d cloth draping by curvature loss. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(1):181–195, 2020.

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

[11] Muhammed Kocabas, Chun-Hao P Huang, Otmar Hilliges, and Michael J Black. Pare: Part attention regressor for 3d human body estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11127–11137, 2021.

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

[13] John P Lewis, Matt Cordner, and Nickson Fong. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 165–172, 2000.

[14] Jie Li, Gilles Daviet, Rahul Narain, Florence Bertails-Descoubes, Matthew Overby, George E Brown, and Laurence Boissieux. An implicit frictional contact solver for adaptive cloth simulation. *ACM Transactions on Graphics (TOG)*, 37(4):1–15, 2018.

[15] Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *International Conference on Learning Representations*, 2018.

[16] Yunzhu Li, Jiajun Wu, Jun-Yan Zhu, Joshua B Tenenbaum, Antonio Torralba, and Russ Tedrake. Propagation networks for model-based control under partial observation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1205–1211. IEEE, 2019.

[17] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):1–16, 2015.

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

[19] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. Laplacian mesh optimization. In *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, pages 381–389, 2006.

[20] Xiaoyu Pan, Jiaming Mai, Xinwei Jiang, Dongxue Tang, Jingxiang Li, Tianjia Shao, Kun Zhou, Xiaogang Jin, and Dinesh Manocha. Predicting loose-fitting garment deformations using bone-driven motion networks. *arXiv preprint arXiv:2205.01355*, 2022.

[21] Chaitanya Patel, Zhouyingcheng Liao, and Gerard Pons-Moll. Tailornet: Predicting clothing in 3d as a function of human pose, shape and garment style. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7365–7375, 2020.

[22] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2020.

[23] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[24] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, pages 8459–8468. PMLR, 2020.

[25] Igor Santesteban, Miguel A Otaduy, and Dan Casas. Learning-based animation of clothing for virtual try-on. In *Computer Graphics Forum*, volume 38, pages 355–366. Wiley Online Library, 2019.

[26] Igor Santesteban, Miguel A Otaduy, and Dan Casas. Snug: Self-supervised neural dynamic garments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8140–8150, 2022.

[27] Igor Santesteban, Nils Thuerey, Miguel A Otaduy, and Dan Casas. Self-supervised collision handling via generative 3d garment models for virtual try-on. ieee. In *CVF Conference*

on Computer Vision and Pattern Recognition (CVPR), volume 2, page 3, 2021.

[28] Lokender Tiwari and Brojeshwar Bhowmick. Deepdraper: Fast and accurate 3d garment draping over a 3d human body. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1416–1426, 2021.

[29] Raquel Vidaurre, Igor Santesteban, Elena Garces, and Dan Casas. Fully convolutional graph neural networks for parametric virtual try-on. In *Computer Graphics Forum*, volume 39, pages 145–156. Wiley Online Library, 2020.

[30] Cyril Zeller. Cloth simulation on the gpu. In *ACM SIGGRAPH 2005 Sketches*, pages 39–es. 2005.