Real-Time Neural Cloth Deformation using a Compact Latent Space and a Latent Vector Predictor

Chanhaeng Lee[1,3], Maksym Perepichka[1,3], Saeed Ghorbani[3], Sudhir Mudur[1], Eric Paquette[2], and Tiberiu Popa[1]

[1] Concordia University, Canada
{l_chanha,m_perepi}@live.concordia.ca
{sudhir.mudur,tiberiu.popa}@concordia.ca
[2] Ecole de Technologie Supérieure, Canada
eric.paquette@etsmtl.ca
[3] Ubisoft La Forge, Canada
saeed.ghorbani@ubisoft.com

Abstract. We propose a method for real-time cloth deformation using neural networks. The computational overhead of most of the existing learning methods for cloth simulation often limits their use in interactive applications. Employing a two-stage training process, our method predicts garment deformations in real-time. In the first stage, a graph neural network extracts cloth vertex features which are compressed into a latent vector with a mesh convolution network. We then decode the latent vector to blend shape weights, which are fed to a trainable blend shape module. In the second stage, we freeze the latent extraction and train a temporal latent predictor network. The temporal latent predictor uses a subset of the inputs from the first stage, ensuring that inputs are restricted to those which are readily available in a typical game engine. Then, during inference, the latent predictor predicts the compacted latent which is processed by the decoder and blend shape networks from the first stage. The latent predictor is the crucial component to speed up our inference time by replacing the resource-intensive graph neural network from the first stage. Our experiments demonstrate that our method effectively balances computational efficiency and realistic cloth deformation, making it suitable for real-time use in applications such as games.

Keywords: Cloth synthesis · Deep neural network · Real time

1 Introduction

Modeling, synthesizing and rendering clothing on virtual characters is a critical task in many applications such as games, special effects, telepresence, and VR environments. Physical simulation has been typically used with excellent results [1, 18], but with significant limitations related especially to performance, stability, and controllability [18].

Modern video games contain many complex components related to graphics and animation working together with a limited computational budget and with a high number of assets (i.e. characters, garment types, accessories, motion types). Therefore, garment synthesis methods aimed at games require, in addition to the high quality of the results, high performance as well as a high degree of generalization to adapt to the large number of assets that typically appear in a game.

Most used methods in games predict displacement on a template garment in the canonical pose and subsequently drive cloth deformations using a combination of blend shapes and Linear Blend Skinning (LBS) [13, 16]. However, as noted by Grigorev et al. [8], this pose-driven deformation has difficulties both in correctly representing loose garments and with the dynamic behaviour of clothing. This is because pose-driven methods are not supervised by any real physics simulation. The HOOD method [8] tries to address these issues by learning complex cloth deformations using direct physics supervision at a vertex level and, as such, is agnostic to the underlying body. One major concern with HOOD though is that it is far too slow to be used in video games.

In this work, we propose a garment synthesis method specifically designed with game requirements in mind. Our method uses the blend shapes and LBS methods while producing good realistic cloth deformation in real time. Our method uses two key ideas. First, we distil the per-vertex feature vector obtained from a powerful but slower method such as HOOD [8] that encodes the complex information about the garment deformation to a per-garment compact latent space that can be efficiently decoded. Then we compute the posed garment from the compact latent space. This addresses only half of the problem: even if the decoding is efficient, the encoding based on the graph embedding and message passing in HOOD is quite slow. To address this issue, our second key idea is to create a fast latent predictor that computes the latent code of the next frame of the deformation. We demonstrate that this approach produces high-quality results at very fast speeds. Moreover, it generalizes over different body shapes thus allowing only one latent space for a variety of character shapes and sizes. We further demonstrate its suitability for game-like applications by showing a real-time demo where the body poses are generated ad-hoc and in real-time using motion matching [6, 9], a common method used in games for character pose synthesis. Our main contributions can be summarized as: (i) a network architecture for effective distillation of per-vertex features to a compact latent space, (ii) an efficient latent predictor for real-time purposes relying on a compact set of inputs readily available in game engines, and (iii) a two-stage training strategy to achieve quality and speed in computing cloth dynamics.

The rest of the paper is organized as follows: section 2 presents in more detail the related work, section 3 presents in detail our method, section 4 presents our results, comparisons and analysis and, finally, section 5 presents our conclusions, limitations and future work.

2 Related Work

In the last decade, an enormous amount of work has been levied toward applying deep learning-based methods to various academic fields including computer graphics and computer animation. Character garment animation has recently received increasing attention for said research. Garment synthesis using neural networks can be largely classified into pose-driven methods where the prediction is conditioned on the body poses and the outputs of the neural networks are displacement for a template garment [5, 19, 21, 23] and/or deformations encoded as blend shapes [2, 12, 14]. These approaches are appealing as the computation of linear blend skinning and blend shapes can be done extremely efficiently on GPU.

An early example of displacement based methods is TailorNet [19], which is a supervised method for predicting character clothing using pose, body shape, and garment style as input. The key to the methodology involves explicitly separating low-frequency and high-frequency garment deformations in-order to avoid the common problem of overly smooth output that neural networks suffer from. Low and high-frequency displacement are generated as functions of body shape, pose, and style using a Multi-Layered Perceptron (MLP). The high-frequency displacement is further refined using the mixture weights predicted by the garment style and the body shape. Self-Supervised Neural Dynamic Garments (SNUG) [21] formulates garment physical-based constraints as loss terms which are minimized during training, allowing for the model to learn displacement for dynamic garment deformations in a self-supervised manner removing the need of simulated data to train. Swish [12] is a quasi-static garment deformer, taking as input the character pose and outputting PCA weights which are used to reconstruct displacement with PCA vectors, similar to blend shapes. This is one of the only two methods that are suited for deployment in games and in fact was used in Electronic Arts Madden NFL 21 to deform football player jerseys. However, this method is limited to tight garments and it does not take into account the dynamics of the cloth. The largest limitation of the LBS-based methods is their failure on loose garments. Zhang et al. [23] address this by learning a generative space of plausible garment geometries. Then, their method learns a mapping to this space to capture the motion-dependent dynamic deformations, conditioned on the previous state of the garment as well as its relative position with respect to the underlying body. SMPLicit [7] is another generative model capable of representing body pose, shape, and clothing geometry. It can represent multiple garment topologies with the same model. This is achieved via a learned implicit function.

Another way to address the failure cases for loose garments is to model the deformed garment using blend shapes [2, 3, 11, 14]. In the training stage, a set of blend shapes is created to span the deformation space of the garment thus improving over the LBS-only methods. Physically-Based Neural Solver (PBNS) [2] uses a self-supervised learning approach for learning garment deformations similar to SNUG [21]. Neural cloth simulation (NCS) [3] utilizes an encoder-decoder

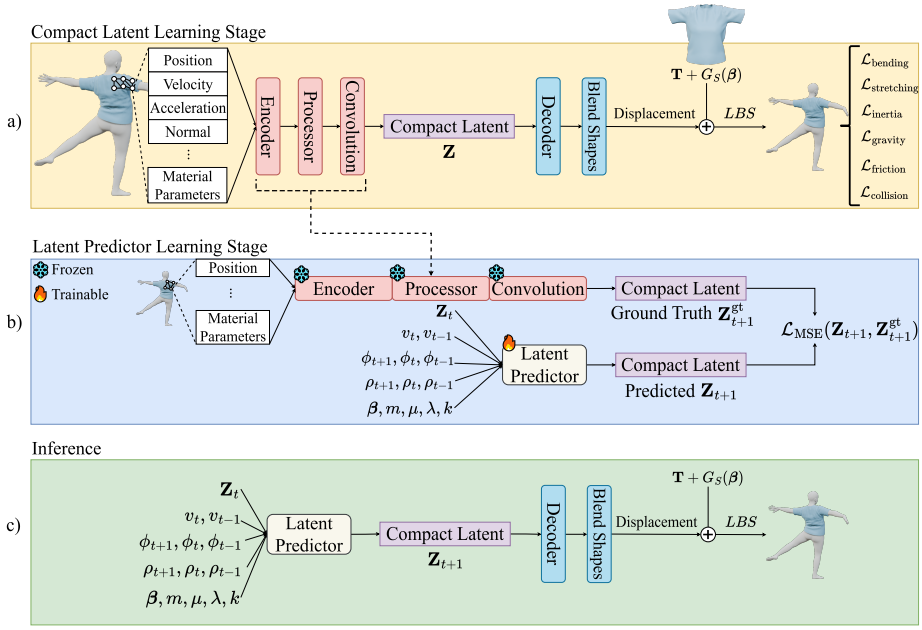


Fig. 1: Overview of our method. We develop a compact latent space for cloth deformation in the first stage (a), and then in the second stage (b) we train a latent predictor to efficiently predict a vector in the compact latent space. During inference (c), we utilize lightweight MLPs (the latent predictor and the decoder) and blend shapes for fast inference.

neural network architecture which explicitly disentangles static and dynamic cloth deformations.

HOOD [8] departs from the pose-based generation framework to get the initially posed garment and uses physics supervision at the vertex level to drive garment deformation. While the results are impressive, due to its usage of Graph Neural Networks coupled with the necessity of computing per-vertex features at every time step, this approach is ill suited to real-time in-game cloth deformation.

3 Method

Our method, depicted in Figure 1, predicts displacement on a template garment in the canonical space and uses LBS to perform garment synthesis for computational efficiency and ease of application. Designed for real-time applications, such as games, our method utilizes a two-stage training process to model dynamic cloth behavior. Our key idea is to obtain a compact latent representation for cloth deformation that can be predicted given sparse inputs and converted into displacement on a template garment.

While a method like HOOD [8] can simulate high-quality garment deformation with dynamics using time integration by predicting acceleration, it results in a high-dimensional latent representation and demands significant computational resources at inference time, making it unsuitable for game engines and real-time applications. Specifically, its encoder generates high-dimensional latent features, its processor updates these features with graph neural networks and message passing, and then its decoder predicts acceleration with the processed features. Utilizing the architecture of the encoder and the processor, which are effective at capturing garment deformation and dynamics, our compact latent learning stage aims to train a network that predicts displacement by compressing the high-dimensional latent features from the encoder and the processor into a compact latent vector using Mesh Convolution [25]. Leveraging the compact latent vector, we predict the displacement with our decoder and learnable blend shapes. The decoder predicts blend shape weights from the compact latent vector, and then the displacement is computed as a weighted sum of the learnable blend shapes with the blend shape weights. Finally, we perform LBS on the deformed garment with the displacement. Training these networks in an end-to-end manner, as illustrated in Fig. 1 (a), enables us to learn a compact latent space that effectively represents garment deformation and dynamics.

Our latent predictor learning stage trains a lightweight MLP to get rid of the encoder, the processor, and the convolutional neural network of the compact latent learning stage, ensuring fast inference, as shown in Fig. 1 (b). This block is trained to predict the latent vector at the next time step $\mathbf{Z}_{t+1}$ using the current latent vector $\mathbf{Z}_t$, root joint velocities of two frames $\{v_t, v_{t-1}\}$, joint rotations of three frames $\{\phi_{t+1}, \phi_t, \phi_{t-1}\}$, joint positions of three frames $\{\rho_{t+1}, \rho_t, \rho_{t-1}\}$, the body shape parameter $\boldsymbol{\beta}$, and material parameters $\{m, \mu, \lambda, k\}$, which are detailed in Sec. 3.3. These inputs are necessary and sufficient to predict the next step latent vector within interactive frame rates. During inference time, using this simple MLP with the decoder and the blend shapes enables our method to be applied in real-time applications, as presented in Fig. 1 (c).

3.1 Garment Model

Our garment model $G(\boldsymbol{\beta}, \boldsymbol{\theta}, \mathbf{X})$ is defined as:

$$G(\boldsymbol{\beta}, \boldsymbol{\theta}, \mathbf{X}) = LBS(T(\boldsymbol{\beta}, \boldsymbol{\theta}, \mathbf{X}), J(\boldsymbol{\beta}), \boldsymbol{\theta}, \tilde{\mathcal{W}}) \tag{1}$$

where $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ are shape and pose parameters used by the SMPL body model [15], $\mathbf{X}$ is a feature vector defined in sections 3.2 and 3.3, and LBS is the linear blend skinning function. This function transforms the deformed garment $T(\boldsymbol{\beta}, \boldsymbol{\theta}, \mathbf{X})$ from the canonical space to the posed space with joint locations $J(\boldsymbol{\beta})$ of the body in shape $\boldsymbol{\beta}$, pose $\boldsymbol{\theta}$, and the diffused skinning weights $\tilde{\mathcal{W}}$.

The core of our garment model is the deformed garment $T(\boldsymbol{\beta}, \boldsymbol{\theta}, \mathbf{X})$. This involves deforming the template garment $\mathbf{T}$ with diffused shape blend-shapes $G_S(\boldsymbol{\beta})$ based on the body shape $\boldsymbol{\beta}$ and further deforming the shaped garment

$(\mathbf{T} + G_S(\boldsymbol{\beta}))$ using displacements predicted by network inputs $\mathbf{X}$. The diffusion method for skinning weights $\tilde{\mathcal{W}}$ and shape blend shapes $G_S(\boldsymbol{\beta})$, introduced by Grigorev et al. [8], is necessary due to the challenge posed by the diverse SMPL body shapes. This method allows garments, which are initially designed to fit the template SMPL body, to be aligned with different shapes of the template body, ensuring appropriate deformation.

3.2 Compact Latent Learning Stage

The goal of this learning stage is to train our network Φ and obtain a low-dimensional latent representation for garment deformation and dynamics. Our network Φ follows an Encode-Process-Convolve-Decode architecture, which integrates the Encode-Process architecture [20] of HOOD with a mesh convolution network and our decoder. The network Φ is trained in an unsupervised manner to predict displacement on the shaped garment for the next time step. This is different from HOOD which predicts acceleration for the next time step. Refer to supplementary material for implementation details. With no ground-truth simulation data requirement, we optimize physics-based losses on the final garment vertices $G(\boldsymbol{\beta}, \boldsymbol{\theta}, \mathbf{X}_{H+})$ using the deformed garment $T(\boldsymbol{\beta}, \boldsymbol{\theta}, \mathbf{X}_{H+})$. The deformed garment in this stage is defined as:

$$T(\boldsymbol{\beta}, \boldsymbol{\theta}, \mathbf{X}_{\mathrm{H}+}) = \mathbf{T} + G_S(\boldsymbol{\beta}) + \Phi(\mathbf{X}_{\mathrm{H}+}), \tag{2}$$

where $\Phi(\mathbf{X}_{\mathrm{H}+})$ is the displacement predicted by our network for the given inputs $\mathbf{X}_{\mathrm{H}+}$.

Inputs $\mathbf{X}_{\mathrm{H}+}$: The network Φ takes inputs $\mathbf{X}_{\mathrm{H}+}$, similar to those in Grigorev et al. [8]. The inputs $\mathbf{X}_{\mathrm{H}+}$ consist of per-vertex and per-edge feature vectors from the garment mesh and the body mesh at time t. The feature vectors for each vertex include velocity, normals, material parameters, vertex type, and vertex level. Material parameters for each vertex consist of mass m, Lamé parameters μ and λ, and the bending coefficient k. The vertex type indicates whether a vertex is pinned to prevent a garment from falling down from a body, and the vertex level represents the coarse level of a vertex (HOOD uses coarsened graphs for its multi-level message passing). Additionally, we incorporate positions and accelerations for garment and body vertices to enhance garment behavior modelling. The feature vectors for each edge of the garment include the relative position of connected vertices at the current time t and in the rest pose, with norms of these relative positions, along with material parameters and delta time. The same per-edge feature vectors are included from coarsened graphs of the garment. For edges connecting body and garment vertices by their proximity, the edge feature vectors include relative positions at the current time t and the next time $t + 1$, with norms and delta time.

Encode and **Process**: The encoder comprises MLPs in the same way as HOOD. It transforms the input feature vectors $\mathbf{X}_{\mathrm{H}+}$ into latent vertex and edge features, which are then processed by a series of message passing blocks in the processor. This processor updates the features to capture garment deformation and

dynamics, following the same approach and inputs of latent vertex and edge features as HOOD. Since HOOD passes only the processed vertex features to its decoder after the processor, indicating that the vertex features contain the necessary information for garment deformation and dynamics, we also pass only the processed vertex features to the mesh convolution network without the edge features.

Convolve: We compress the processed vertex features into a compact latent vector $\mathbf{Z}$ with the mesh convolution network by Zhou et al. [25]. This network excels at constructing localized latent features, making it ideal for our compression task. We utilize its convolution and residual layers to reduce the dimensionality of the processed vertex features and flatten the compressed vertex features into a latent vector. Since the compressed latent vector $\mathbf{Z} \in \mathbb{R}^L$ with the dimension L is used at inference time, the dimension L of the latent vector has to be small enough to be evaluated in interactive frame rates, and large enough to encode garment deformation and dynamics. This latent vector is then passed to the decoder.

Decode: When predicting garment deformations, learnable blend shapes have proven effective in modeling non-linear garment behavior [2–4]. Following this approach, we build our decoder $f_{\text{dec}} : \mathbb{R}^L \to \mathbb{R}^D$ with an MLP, which takes the latent vector $\mathbf{Z} \in \mathbb{R}^L$ and predicts D blend shape weights for our learnable blend shapes. The learnable blend shapes consist of D blend shape matrices. Thus, the final displacement for each vertex is defined as:

$$\sum_{j}^{D} \mathbf{D}_{j,i} f_{\text{dec}}(\mathbf{Z})_j, \tag{3}$$

where $\mathbf{D}$ is the array of the learnable blend shape matrices and $\mathbf{D}_{j,i}$ indicates the blend shape basis for the ith garment vertex of the jth blend shape matrix in the array. A single blend shape matrix $\mathbf{D}_j \in \mathbb{R}^{N\times 3}$ consists of blend shape bases for N garment vertices. The combination of the MLP with the blend shapes is fast enough to enable real-time inference. Details about the implementation are elaborated in the supplementary material.

Loss Functions: We train our network with the same losses as described by Grigorev et al. [8]. We employ the bending loss $\mathcal{L}_{\text{bending}}$ that introduces smoothness by penalizing sharp bends, measured through the dihedral angles between adjacent triangles [1]. The stretching loss $\mathcal{L}_{\text{stretching}}$, based on the Saint Venant-Kirchhoff (StVK) model, enforces hyperelastic material behavior. The inertia loss $\mathcal{L}_{\text{inertia}}$ is used to generate realistic dynamic motion by resisting drastic changes in velocity. The gravity loss $\mathcal{L}_{\text{gravity}}$ applies a constant downward force on the vertices of the garment mesh, creating realistic drapes and falls. The friction loss $\mathcal{L}_{\text{friction}}$ prevents the sliding motion of the garment, enhancing its stability and realism. Since all the above losses can cause interpenetration between the garment and the body, the collision loss $\mathcal{L}_{\text{collision}}$ is used to move the garment vertices away from the body vertices. Therefore, the total loss $\mathcal{L}$ is defined as a

weighted sum of these individual losses:

$$\begin{aligned}\mathcal{L} =& w_b\mathcal{L}_{\text{bending}} + w_s\mathcal{L}_{\text{stretching}} + \\ & w_i\mathcal{L}_{\text{inertia}} + w_g\mathcal{L}_{\text{gravity}} + \\ & w_f\mathcal{L}_{\text{friction}} + w_c\mathcal{L}_{\text{collision}},\end{aligned} \tag{4}$$

where w_b, w_s, w_i, w_g, w_f, and w_c are scalar weights that control the contributions of each loss term. By adjusting these weights, we fine-tune the balance between different aspects of cloth behavior. Following unsupervised training methods with physics-based losses [2,3,8,21], optimizing our network using these losses allows for realistic cloth deformation, similar to solving equations of motion for cloth simulation through energy optimization.

3.3 Latent Predictor Learning Stage

The goal of this stage is to train a latent predictor f_{LP}, which will replace the computationally intensive Encode-Process-Convolve components of the network Φ in the compact latent learning stage.

To ensure that the latent predictor is lightweight and suitable for real-time use, it is constructed solely with an MLP. Refer to the supplementary material for implementation details. The latent predictor takes a set of simpler inputs $\mathbf{X}_c$ compared to the inputs $\mathbf{X}_{\text{H}+}$. The inputs $\mathbf{X}_c$ are optimized for instantaneous preparation in each frame while being robust enough to accurately predict the compact latent vector $\mathbf{Z}$. These inputs are defined as $\mathbf{X}_c = \{\mathbf{Z}_t, \boldsymbol{v}, \boldsymbol{\phi}, \boldsymbol{\rho}, \boldsymbol{\beta}, m, \mu, \lambda, k\}$. The latent vector at the current time step $\mathbf{Z}_t$ provides temporal context, aiding in the prediction of the next latent vector. This input latent vector is set to zero in the first frame. The inputs include joint velocities $\boldsymbol{v}$, local joint rotations $\boldsymbol{\phi}$, and global joint positions $\boldsymbol{\rho}$. Specifically, the joint velocities comprise root joint velocities over two frames $\{v_t, v_{t-1}\}$, and the pose consists of local joint rotations over three frames $\{\phi_{t+1}, \phi_t, \phi_{t-1}\}$ in 6D representation [24] and global joint positions over three frames $\{\rho_{t+1}, \rho_t, \rho_{t-1}\}$. We include the global joint positions which are evaluated by forward kinematics with animation sequences, in order to enable the latent predictor to find a better mapping function from the inputs to a latent vector. To prevent the latent predictor from overfitting to the global joint positions of animation sequences, we subtract global joint positions with the root joint translation and remove the root joint translation from the inputs. We also exclude joints such as wrists and hands from the inputs, due to their minimal impact on garment deformation. We additionally incorporate the body shape $\boldsymbol{\beta}$ and garment material parameters. The material parameters are mass m, Lamé parameters μ and λ, and the bending coefficient k, which are also used for the compact latent learning stage. Through these inputs, we can reduce the number of variables in $\mathbf{X}_{\text{H}+}$ from hundreds of thousands of parameters down to thousands in $\mathbf{X}_c$. For example, we reduce 886,165 floating-point variables in $\mathbf{X}_{\text{H}+}$ to 2,591 floating-point variables in $\mathbf{X}_c$ for a t-shirt garment in our training dataset.

Utilizing these inputs, the latent predictor predicts the next latent vector $\mathbf{Z}_{t+1} = f_{\text{LP}}(\mathbf{X}_c)$. To optimize the latent predictor, we compute the ground truth latent vector $\mathbf{Z}^{\text{gt}}_{t+1}$ using the Encode-Process-Convolve components with the inputs $\mathbf{X}_{\text{H}+}$, while keeping the weights of the components frozen. The optimization is performed by minimizing the mean squared error loss between the predicted latent vector and the ground truth:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{L}\sum_{i}^{L}\left(\mathbf{Z}^{\text{gt}}_{t+1,i} - \mathbf{Z}_{t+1,i}\right)^2 \tag{5}$$

During inference, our garment model $G(\boldsymbol{\beta}, \boldsymbol{\theta}, \mathbf{X}_c)$ gets the final transformed vertices using the deformed garment $T(\boldsymbol{\beta}, \boldsymbol{\theta}, \mathbf{X}_c)$. The deformed garment at inference time is defined as:

$$T(\boldsymbol{\beta}, \boldsymbol{\theta}, \mathbf{X}_c) = \mathbf{T} + G_S(\boldsymbol{\beta}) + \Psi(\mathbf{X}_c), \tag{6}$$

where the function Ψ predicts the displacement on the shaped garment using the latent predictor, the decoder, and the learnable blend shapes with the given inputs $\mathbf{X}_c$. The displacement for each vertex is computed as $\sum_j^D \mathbf{D}_{j,i} f_{\text{dec}}(f_{\text{LP}}(\mathbf{X}_c))_j$.

4 Results and Discussion

In this section, we evaluate the effectiveness and performance of our networks. We first detail our training setup. Next, we qualitatively compare our networks with a state-of-the-art method, NCS [3]. Additionally, we measure and compare the computational performance of our method against NCS. Furthermore, we validate the effectiveness of our two-stage training process by an ablation study, and we demonstrate the efficiency and applicability of our method by implementing it in a real-time application demo.

4.1 Training Setup

Training Dataset: For our experiments, we use pose sequences from the AMASS dataset [17]. We adopt the sequence list from the VTO dataset [22] for training purposes. The VTO dataset takes sequences from the AMASS dataset, but we replace some unavailable sequences of the list with others in similar pose categories of the AMASS dataset (The list of sequences is found in the supplementary material). This sequence list contains 56 sequences with a total of 19,145 frames, comprising 13 walking, 12 running, 10 jumping, 10 arm movements, 6 torso movements, 4 dancing, and 1 avoidance movement sequence. For testing, we use other sequences from the AMASS dataset, excluding those used for training. The garments for our experiments include a dress, a long-sleeve top, pants, a tank top, and a t-shirt, provided by Grigorev et al. [8] for training their HOOD network. These garments are aligned with the template female body of SMPL. Accordingly, we used the SMPL female body for our experiments.

Training Details: We trained five networks for compact latent learning and five latent predictors with the five garments in the training set. We implemented our method and trained the networks on a PC, equipped with an Intel Xeon W-2135 CPU, an NVIDIA RTX A4000 GPU, and 64GB of RAM. Training for the compact latent learning stage took approximately 40 hours per garment, with 120,000 iterations each. We used the Adam optimizer for training the networks in the compact latent learning stage, with a learning rate 5×10^{-5}, and applied gradient clipping with a max norm of 1.0. Since the training method from HOOD does not support batch training, we used a batch size of one. For each training pose, We randomly sampled the shape $\boldsymbol{\beta}$ from the uniform distribution $\mathcal{U}(-3, 3)$. We also randomly sampled material parameters for the inputs by sampling a value from the uniform distribution $\mathcal{U}(0, 1)$ and scaling it with its minimum and maximum values. The mass for each vertex of a garment m ranged from 4.34×10^{-2} to 7×10^{-1}, the range of Lamé's second parameter μ from 15909 to 63636, the range of Lamé's first parameter λ from 3535.41 to 93333.73, and the range of the bending coefficient k from 6.37×10^{-8} to 1.31×10^{-3}. To advance our garment deformation over time, we apply the autoregressive training from HOOD, predicting one next step at the beginning and increasing the number of prediction steps every 5000 iterations to 5. We further set the inertia loss weight w_i from 3.0 to 5.0 and the gravity loss weight w_g from 2.0 to 3.0 to make our garment deformation more dynamic, with the other weights w_b, w_s, w_f set to 1.0. The collision loss w_c starts at 5×10^3 at the beginning of training and increases linearly to 8×10^6 from 50,000 iterations to 100,000 iterations. Each latent predictor was trained for 5 million iterations, which took approximately 12 hours. We trained latent predictors for each garment using an Adam optimizer with a learning rate of 1×10^{-4} and a batch size of 512.

Normalization: When training a network in the compact latent learning stage, we adopted the inputs and outputs normalization from HOOD. Unlike the normalization method for the outputs from HOOD, which gathers statistics for acceleration from linearly-skinned garments, we collect statistics for displacement from our prediction and evaluate the final displacement by denormalizing the predicted outputs. When training a latent predictor, we also normalize the inputs $\mathbf{X}_c$ and the outputs $\mathbf{Z}$ by mean and standard deviations for each input and each output except for the material parameters. The material parameters in the inputs are normalized to fall between 0.0 and 1.0 based on their value ranges.

4.2 Comparison with State-of-the-art

For this comparison, we trained five NCS networks for each garment in the training set with the template SMPL female body, as NCS supports only a fixed-shaped body with a garment. We used the public code released by Bertiche et al. [3] and a batch size of 512. We chose the cloth model of Baraff et al. [1], since it is more optimal than the StVK model for convergence according to Bertiche et al. [3]. We further applied the heuristic method of Li et al. [14], which gradually

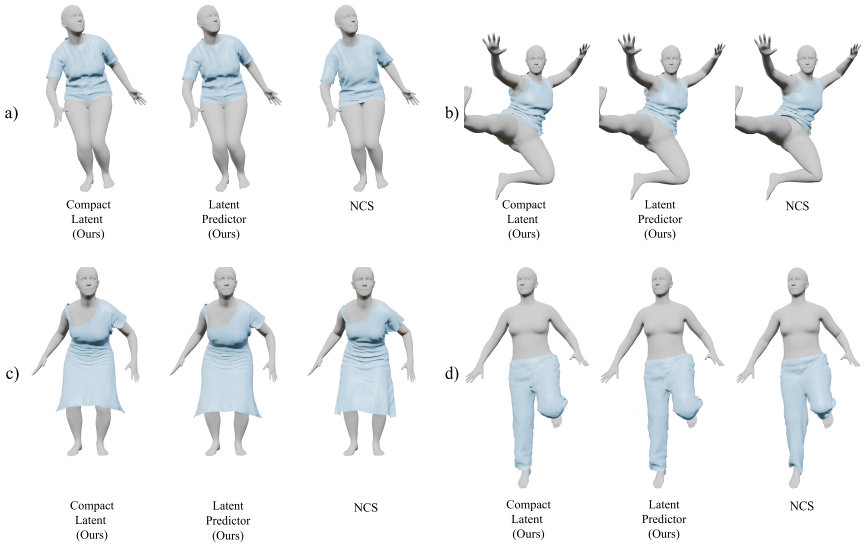


Fig. 2: Qualitative comparison with NCS [3]. For each garment, the first column shows the results from networks Φ in the compact latent learning stage, the second column shows results from networks Ψ using latent predictors f_{LP}, and the third column shows results from NCS networks.

increases the inertia loss weight from 0.1 to 1.0. This heuristic allows us to get the best cloth dynamics.

Unsupervised methods lack ground-truth data for quantitative analysis. Furthermore, the variety of network architectures, loss function implementations, and parameter ranges complicates a fair quantitative comparison. Moreover, the sole comparison of loss values does not work. For instance, a lower inertia loss value does not guarantee superior quality or more dynamic deformation, as demonstrated by Bertiche et al. [3]. Consequently, our comparison is solely qualitative.

We present our qualitative results with NCS in Fig. 2 and the supplementary video. The first column shows the inference from the compact latent learning stage network, the second column shows the inference using the latent predictor with the decoder and blend shapes, and the third column shows the inference from NCS. As seen in Fig. 2 and the supplementary video, our method produces wrinkles and dynamics of similar quality to NCS. Unlike NCS, our method supports generalization to different body shapes with the same trained network. This generalization is shown in Fig. 3 and the supplementary video, which includes a thin body, the template body, and a heavy body. Our latent predictor can handle different body shape parameters $\boldsymbol{\beta}$ and predicts a compact latent vector accordingly.

For computational performance comparisons, a wide range of neural models for cloth simulation exist. Graph neural network models such as HOOD [8] and

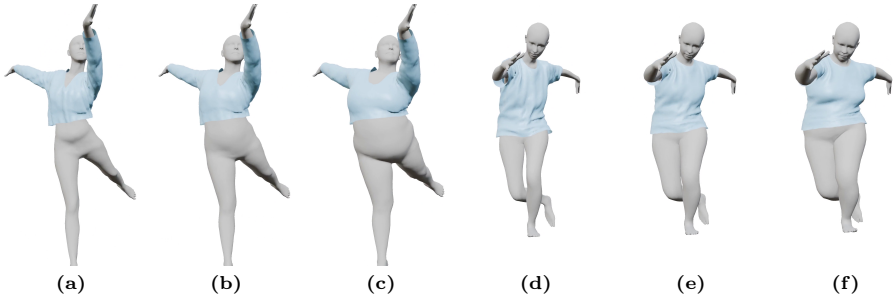


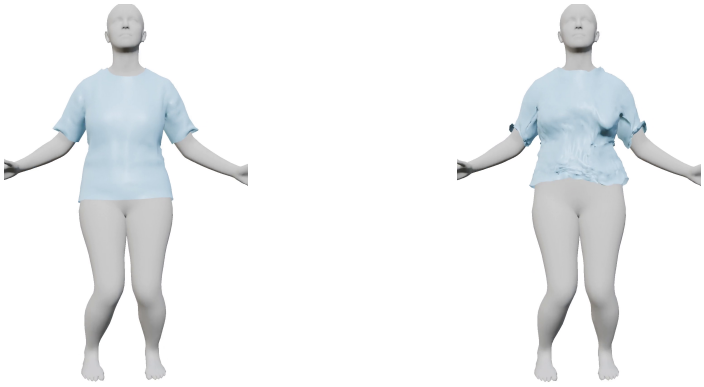
Fig. 3: The generalization capacity to different body shapes from networks Ψ using the latent predictor, the decoder, and the blend shapes. **(a)** and **(d)** are the thin bodies, **(b)** and **(e)** are the template bodies, and **(c)** and **(f)** are the heavy bodies.

SwinGar [14] do not claim to achieve engine-ready real-time performance, as their architecture limits their performance to orders of magnitude slower. Only pose-driven models such as PBNS [2], SNUG [21], and NCS [3] achieve the desired performance criteria. Out of these models, PBNS is purely a static model, not being capable of achieving garment dynamics, and SNUG has dynamics limited to three frames, as thoroughly discussed in Bertiche et al. [3]. As such, we limit our comparison to the closest competitor, NCS [3].

Performance Evaluation: We evaluated the per-frame inference times on GPU for the "full model" (including both model inference and blend shape resolution). We also separately evaluated the runtimes for solely the neural network inference without the blend shape resolution step. Our method takes 523 microseconds per frame for the full model and 238 microseconds for the neural network inference. In contrast, NCS takes 1266 microseconds per frame for the full model and 1030 microseconds for the neural network inference. This result shows that our method outperforms NCS on GPU in the computational performance. We believe this behavior is explained by architecture differences between our model and NCS: our model utilizes more neurons but a simpler structure than NCS, as we do not utilize a GRU layer. As such, when computing on GPU, our model benefits more from GPU parallelism than NCS, while when computing on CPU, the lack of parallelism hurts our performance. The evaluation was done using models exported to ONNX Runtime on a t-shirt garment with 4424 vertices. The hardware utilized was an Intel Xeon W-2255 CPU @ 3.7GHz CPU, and an NVIDIA RTX 2070 Super GPU.

4.3 Ablation Study

To validate the effectiveness of our two-stage training process in learning garment deformation and dynamics, we performed an ablation study. We directly trained only the components used during inference (Fig. 1 (c)), specifically the latent predictor, the decoder, and the learnable blend shapes. We used the same training setup with the t-shirt garment as in the compact latent learning stage,



(a) Full method **(b)** Training only the inference block (Fig. 1 (c))

Fig. 4: (a) Our method shows accurate garment deformation and realistic dynamics (results from network Ψ using the latent predictor, the decoder, and the blend shapes). **(b)** The ablation study, trained with only the inference block (Fig. 1 (c)), results in poor and invalid deformations, highlighting the necessity of the compact latent learning stage.

utilized the same losses (Eq. (4)) from the compact latent learning stage, and additionally supported batch training with a batch size of 32. The results, presented in Fig. 4 and the supplementary video, demonstrate that training with only the MLPs and blend shapes does not achieve proper garment deformation and dynamics. The results from the ablation study exhibit less realistic dynamics and invalid deformations in parts of the garment. This outcome validates that our compact latent learning stage is essential for effectively learning garment deformation and dynamics.

4.4 Application on Real-Time Demo

We implemented our inference model with the trained latent predictor, decoder, and blend shapes using C++ and OpenGL. In the supplementary video, we show a SMPL body controlled using motion matching [6, 9]. This shows the ease with which our method can be applied in real time, thanks to our inference model consisting of only two MLPs with blend shapes.

5 Conclusion, Limitations and Future Work

Realistic cloth deformation in game applications is complex due to varying character body shapes, poses, and movements, and many garments with different geometry, topology and materials. On top of these requirements, real-time cloth deformation is essential for an enjoyable game experience. Procedural physics-based techniques are being replaced by neural network approaches. The latter seems more robust and yields high-quality results. However, the representation

models required to capture realistic cloth deformation behavior are of very high dimension and correspondingly difficult to incorporate in game engines with limited resources and also far too slow for use in games. The two-stage training strategy presented in this work balances computational resource constraints and realistic cloth deformation effectively for application in games. The first stage learns a compact representation from the high-dimensional feature representation of a complex trained network, and the second stage learns a latent predictor with inputs in a format best suited for input to game engines. At inference time, the latent predictor predicts the compacted latent which is processed by the decoder and blend shape networks from the first stage, enabling framewise inference in real-time. This two-stage strategy is general and could be used in other neural network methods, wherein high-quality results mandate very high dimensional feature representations, and correspondingly high computational resources, limiting their application in real-time environments. Our experiments demonstrate the effectiveness and applicability of our method compared to NCS.

Our method has some limitations. Similar to other learning-based methods for garment synthesis, there are instances in test sequences where interpenetration between the garment and the body is not fully resolved. Since we utilize the previous compact latent in the latent predictor, these failures can lead to invalid garment deformation. This issue is especially problematic on loose garments. Future work could focus on enhancing collision handling by implementing an edge-based collision loss, which may provide more robust results compared to the vertex-based collision loss currently used [10]. The issue of drift is likewise a potential problem, notably for applications such as games where inference is run for long periods of time. In practice, our model proved to be robust to this: all of our experiments were exempt from drift. Our explanation for this robustness against drift is that our method first predicts learned latents before outputting absolute canonical space displacements. However, there are no explicit guarantees on the robustness of the model output with respect to drift. Another area of improvement is the generalization to garments with different topologies. The mesh convolution and the learnable blend shapes in our method are not agnostic to garment topology. In the future, we will explore other ways to compress the high-dimensional per-vertex features and compute the displacement from them, regardless of the garment topology.

Acknowledgements. We acknowledge the financial support of UBISOFT, Concordia University and the Natural Sciences and Engineering Research Council of Canada (NSERC) [funding reference number 456440596]. We would like to thank Marc-André Carbonneau, François Levesque, and Olivier Pomarez for their inputs and review of the work.

References

1. Baraff, D., Witkin, A.: Large steps in cloth simulation. In: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques. p. 43–54. SIGGRAPH '98, Association for Computing Machinery, New York, NY, USA (1998)

2. Bertiche, H., Madadi, M., Escalera, S.: Pbns: Physically based neural simulation for unsupervised garment pose space deformation. ACM Trans. Graph. **40**(6) (dec 2021)
3. Bertiche, H., Madadi, M., Escalera, S.: Neural cloth simulation. ACM Trans. Graph. **41**(6) (nov 2022)
4. Bertiche, H., Madadi, M., Tylson, E., Escalera, S.: Deepsd: Automatic deep skinning and pose space deformation for 3d garment animation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5471–5480 (2021)
5. Chen, H., Yao, Y., Zhang, J.: Neural-abc: Neural parametric models for articulated body with clothes. IEEE Transactions on Visualization and Computer Graphics (2024)
6. Clavet, S.: Motion matching and the road to next-gen animation. In: GDC 2016 (2016)
7. Corona, E., Pumarola, A., Alenya, G., Pons-Moll, G., Moreno-Noguer, F.: Smplicit: Topology-aware generative model for clothed people. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11875–11885 (2021)
8. Grigorev, A., Thomaszewski, B., Black, M.J., Hilliges, O.: HOOD: Hierarchical graphs for generalized modelling of clothing dynamics (2023)
9. Holden, D., Kanoun, O., Perepichka, M., Popa, T.: Learned motion matching. ACM Trans. Graph. **39**(4) (aug 2020)
10. Kim, T., Eberle, D.: Dynamic deformables: implementation and production practicalities (now with code!). In: ACM SIGGRAPH 2022 Courses. SIGGRAPH '22, Association for Computing Machinery, New York, NY, USA (2022)
11. Lahner, Z., Cremers, D., Tung, T.: Deepwrinkles: Accurate and realistic clothing modeling. In: Proceedings of the European conference on computer vision (ECCV). pp. 667–684 (2018)
12. Lewin, C.: Swish: Neural network cloth simulation on madden nfl 21. In: ACM SIGGRAPH 2021 Talks, pp. 1–2 (2021)
13. Lewis, J.P., Cordner, M., Fong, N.: 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. pp. 165–172 (2000)
14. Li, T., Shi, R., Zhu, Q., Kanai, T.: Swingar: Spectrum-inspired neural dynamic deformation for free-swinging garments. IEEE Transactions on Visualization and Computer Graphics pp. 1–16 (2023)
15. Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., Black, M.J.: SMPL: A skinned multi-person linear model. ACM Trans. Graphics (Proc. SIGGRAPH Asia) **34**(6), 248:1–248:16 (Oct 2015)
16. Magnenat-Thalmann, N., Laperrire, R., Thalmann, D.: Joint-dependent local deformations for hand animation and object grasping. In: In Proceedings on Graphics interface'88. Citeseer (1988)
17. Mahmood, N., Ghorbani, N., Troje, N.F., Pons-Moll, G., Black, M.J.: AMASS: Archive of motion capture as surface shapes. In: International Conference on Computer Vision. pp. 5442–5451 (Oct 2019)
18. Nealen, A., Müller, M., Keiser, R., Boxerman, E., Carlson, M.: Physically based deformable models in computer graphics. In: Computer graphics forum. vol. 25, pp. 809–836. Wiley Online Library (2006)
19. Patel, C., Liao, Z., Pons-Moll, G.: 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. pp. 7365–7375 (2020)

20. Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., Battaglia, P.W.: Learning mesh-based simulation with graph networks. In: International Conference on Learning Representations (2021)
21. Santesteban, I., Otaduy, M.A., Casas, D.: SNUG: Self-Supervised Neural Dynamic Garments. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2022)
22. Santesteban, I., Thuerey, N., Otaduy, M.A., Casas, D.: Self-Supervised Collision Handling via Generative 3D Garment Models for Virtual Try-On. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2021)
23. Zhang, M., Ceylan, D., Mitra, N.J.: Motion guided deep dynamic 3d garments. ACM Transactions on Graphics (TOG) **41**(6), 1–12 (2022)
24. Zhou, Y., Barnes, C., Jingwan, L., Jimei, Y., Hao, L.: On the continuity of rotation representations in neural networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
25. Zhou, Y., Wu, C., Li, Z., Cao, C., Ye, Y., Saragih, J., Li, H., Sheikh, Y.: Fully convolutional mesh autoencoder using efficient spatially varying kernels. Advances in neural information processing systems **33**, 9251–9262 (2020)