# Learning an Intrinsic Garment Space for Interactive Authoring of Garment Animation

TUANFENG Y. WANG, miHoYo Inc.
TIANJIA SHAO, University of Leeds
KAI FU, miHoYo Inc.
NILOY J. MITRA, University College London and Adobe Research

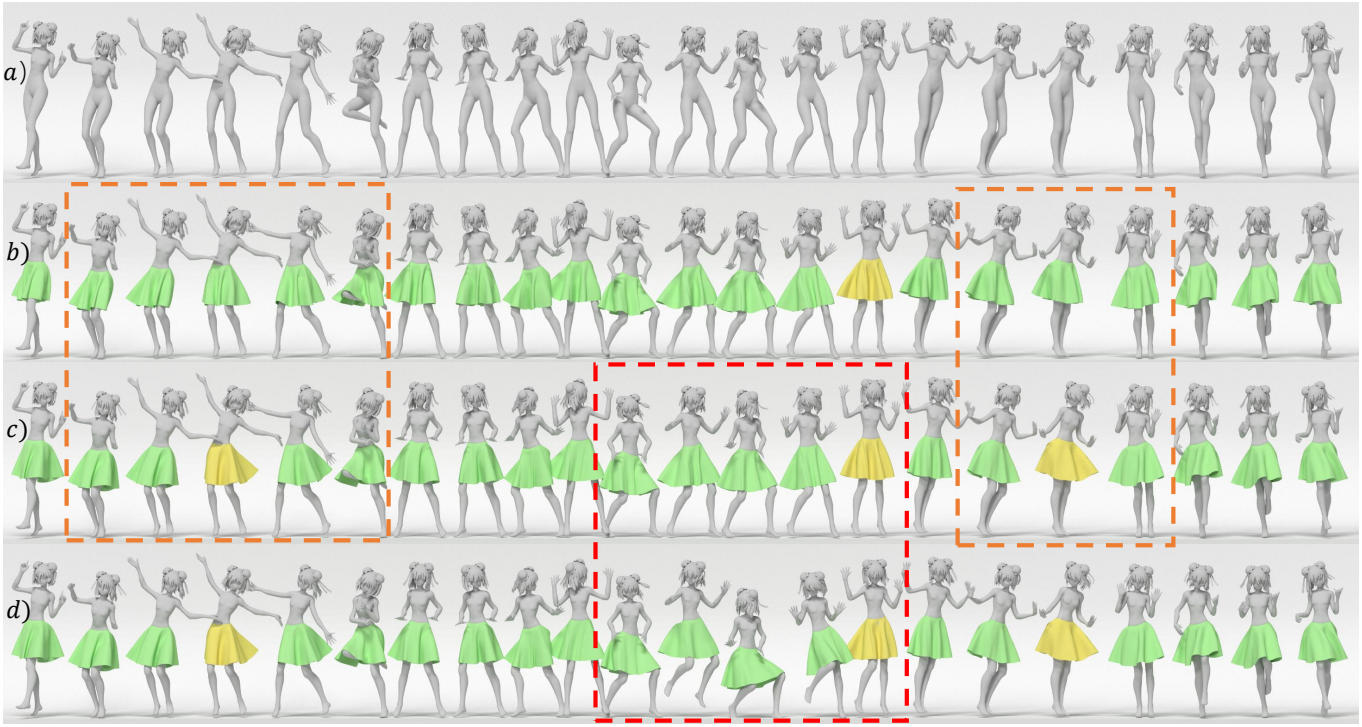



Fig. 1. **A semi-automatic garment animation authoring workflow.** Given a body animation sequence (a), the artist selects a keyframe (marked by yellow) and inputs the shape of the garment for that frame (b), and our system learns the intrinsic properties of this design and automatically propagates this design to other frames (marked by green). The artist may further adjust the garment animation (c) by inserting new keyframes (marked by yellow) and editing the garment shapes of the keyframes. The animation sequence is then automatically updated by linking the keyframes with plausible transitions (highlighted in orange boxes). Our workflow also allows the artist to modify the body animation (marked in red box) during the composition because our system infers the garment shape from the underlying body motion at an interactive rate (d).

Authoring dynamic garment shapes for character animation on body motion is one of the fundamental steps in the CG industry. Established workflows are either time and labor consuming (i.e., manual editing on dense frames with controllers), or lack keyframe-level control (i.e., physically-based simulation). Not surprisingly, garment authoring remains a bottleneck in many production pipelines. Instead, we present a deep-learning-based approach for semi-automatic authoring of garment animation, wherein the user provides the desired garment shape in a selection of keyframes, while our system infers a latent representation for its motion-independent *intrinsic parameters* (e.g., gravity, cloth materials, etc.). Given new character motions, the latent representation allows to automatically generate a plausible garment animation at interactive rates. Having factored out character motion, the learned intrinsic garment space enables smooth transition between keyframes on a new motion sequence. Technically, we learn an intrinsic garment space with an motion-driven autoencoder network, where the encoder maps the garment shapes to the intrinsic space under the condition of body motions, while the decoder acts as a *differentiable simulator* to generate garment

Authors' addresses: Tuanfeng Y. Wang, miHoYo Inc., yangtuanfeng.wang.14@ucl.ac.uk; Tianjia Shao, University of Leeds, tianjiashao@gmail.com; Kai Fu, miHoYo Inc., kai.fu@mihoyo.com; Niloy J. Mitra, University College London, n.mitra@cs.ucl.ac.uk.


0730-0301/2019/11-ART220 $15.00
https://doi.org/10.1145/3355089.3356512

shapes according to changes in character body motion and intrinsic parameters. We evaluate our approach qualitatively and quantitatively on common garment types. Experiments demonstrate our system can significantly improve current garment authoring workflows via an interactive user interface. Compared with the standard CG pipeline, our system significantly reduces the ratio of required keyframes from 20% to $1-2\%$.

CCS Concepts: • **Computing methodologies → Shape modeling**; *Neural networks*; *Animation*.

Additional Key Words and Phrases: intrinsic garment space, latent representation, interactive garment animation authoring

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

## 1 INTRODUCTION

With the rapid development of physical garment simulation techniques, it is now much simpler to create garment animations. Given a body shape and its motion sequence, the user sets up a simulator with simulation parameters (e.g., time scale, etc.), inputs the garment material properties (e.g., bend, stretch, etc.) and environment condition (e.g., gravity, air drag, etc.), and a simulation system can automatically produce a physically plausible garment animation. Understanding the effect of each simulation parameter, however, is difficult, especially for parameters, e.g., time step or max CG iteration, without any clear physical meaning. Understanding the effect of the combination of such parameters is probably beyond human capacity. This makes adjusting such parameters to achieve the desired simulation output remain very inefficient [Sigal et al. 2015]. Furthermore, due to the concern of aesthetics, a single valid set of the parameters may not exist. For example, one set of parameters can achieve the anticipated garment shapes for the first 10 frames while another set is preferred for the next 20 frames, etc. Therefore it still requires significant manual efforts of experienced artists to compose one desired garment animation.

A common workflow in the modern CG industry for garment animation composition/editing is the *keyframe approach*. For each keyframe, the artist adjusts the garment shapes commonly with skinning techniques such as Linear Blend Skinning (LBS) [Kavan and Žára 2005] and Dual Quaternion Skinning (DQS) [Kavan et al. 2007]. The input garment shapes in the keyframes are then propagated to other frames via interpolation. However, as the garment geometry is closely correlated to body motion, material properties, and environment, the garment shape space is exceedingly nonlinear and complex. In order to achieve physically plausible garment shapes consistent across motion, it requires very dense sample points for interpolation within such a space. Consequently, the keyframes must be densely distributed in the sequence (often as high as 20% of the frames), and hence it remains extremely labor-intensive.

In this paper, we seek a semi-automatic framework for garment animation composition. Such a framework requires addressing the following challenges. First, the generation of garment animation should be as automatic as physical simulation. Second, specified aesthetics preferences for selected keyframes should be respected and can be naturally interpolated across the keyframes without dense user input. Third, it should support nearly real-time interaction with the user to provide instant feedback.

To tackle these challenges, we introduce a deep-learning-based approach for semi-automatic authoring of garment animation. For a given character and garment template (e.g., skirt), we learn a model that can infer a latent representation of *intrinsic parameters* (i.e., simulator parameters, garment material parameters, and environment condition parameters) from an example keyframe. We call these parameters the *intrinsic parameters*, as they are independent of the character motions. To account for the automatic generation of garment animation, the learned model can further apply the learned latent representation to new motions to automatically produce physically plausible garment shapes consistent with the given motion, just as the physical simulator does (Fig 1(b)). Since the latent representation is independent of the body motion, the latent space is significantly simplified compared with the original $3D$ shape space. This allows simple linear interpolation from sparse sample points. Subsequently, as the artist edits more keyframes, to account for the natural interpolation across keyframes, the model first infers the latent representation for those sparse keyframes, and a linear interpolation is performed to compute the intermediate status. Then the intermediate garments can be recovered by applying those interpolated latent representations on intermediate motions (Fig 1(c)). Finally, as our approach is based on deep neural networks with a low computational cost at run-time, the artist can edit the animation at an interactive rate.

We propose a motion-invariant autoencoder neural network for our task. Given a keyframe, the encoder learns to map its garment shape descriptor into a latent space under the condition of corresponding body motion. The latent vector can be interpreted as a latent representation of the intrinsic parameters, i.e., all the garments generated with the same intrinsic parameters should be mapped to the same location in latent space by factoring out the body motion. The decoder learns to reconstruct the garment geometry from a latent vector also under the condition of a particular motion, i.e., it is a *differentiable simulator* for the automatic animation generation. Motion information is incorporated into the autoencoder via a motion descriptor learned from a motion encoder. Following the idea of Phase-Functioned Neural Network [Holden et al. 2017], the motion descriptor represents a set of coefficients which linearly blend the multiple sub-networks within an autoencoder layer. Thus, the network weights are updated dynamically according to the motion. The encoder, decoder, and the motion descriptor are jointly trained.

We qualitatively and quantitatively evaluated our method for different garment types (skirt, sailor collar, and long skirt) and character motion sequences. Experimental results show that our model can predict accurate latent representation from the keyframes and yield satisfactory garment shapes on other frames with different motions. The quality of generated garments is comparable with that from a physical-based simulation. Moreover, our method enables several important applications which might be challenging for physical simulation or keyframing, such as automatically adjusting garment animation under modifications of input body animation (Fig 1(d)) and simplifying creation of loop-animations. We have also integrated our model into the standard CG software (Maya) and

tested it in a garment production pipeline (Fig 14), significantly reducing the time and labor required.

In summary, our main contributions include:

• a novel semi-automatic pipeline for authoring garment animation;

• learning a motion-factorized latent space that encodes intrinsic information of the garment shape; and

• learning a *differentiable garment simulator* to automatically reconstruct garment shapes from an intrinsic garment representation and target body motion.

## 2 RELATED WORK

*Garment Simulation.* There has been a tremendous amount of literature on garment simulation. Many physics-based simulation methods focus on improving the simulation efficiency while retaining the accuracy, for example, with implicit Euler integration [Baraff and Witkin 1998], low-dimensional linear subspaces [Hahn et al. 2014], multigrid [Tamstorf et al. 2015], and iterative optimization [Liu et al. 2013; Wang and Yang 2016]. In order to capture dynamic details, adaptive remeshing techniques are applied during physical simulation [Narain et al. 2012; Weidner et al. 2018]. Various collision handling methods are also proposed to resolve collisions [Goldenthal et al. 2007; Tang et al. 2018]. While physics-based simulation is able to produce realistic garment animation, it is still very challenging to tune the parameters to achieve the desired visual effects for the artist. Our goal is to revise the garment animation to meet the visual requirements by only editing very sparse key frames, while still keeping the sequence physically plausible.

Data-driven methods are also commonly used in garment simulation. For example, much of recent work learns from offline simulations to achieve real time performance [de Aguiar et al. 2010; Kavan et al. 2011; Kim et al. 2013; Wang et al. 2010; Xu et al. 2014]. Other works focus on the transfer of simulated garments to different body shapes and poses. Guan et al. [2012] learn a garment model from simulated garments on many body shapes and poses, which can then be applied to new bodies and poses without simulation; Santesteban et al. [2019] learn a garment deformation model with deep neural networks that enables virtual try-on by different bodies and poses; while, Fulton et al. [2019] propose a reduced model simulation framework for deformable solid dynamics using autoencoder neural networks. Our aim is different. As in animation authoring, we not only care about the garment shape for each single frame but also the dynamic and consistency along a motion sequence.

*Garment Capture.* As an alternative to simulation, garment capture methods aim to faithfully reconstruct the garment animation from captured data. Early work utilizes color-coded patterns printed on garments to support video-based 3D capture [White et al. 2007]. Bradley et al. [2008] present a multi-view markerless motion capture system for garments. Popa et al. [2009] extend the multi-view system by adding folds into the captured garment model. The ClothCap [Pons-Moll et al. 2017] system estimates the garments and theirs motion from 4D scans assuming weak priors about where a specific type of garment is expected to be with respect to the body. Yang et al. [2018] further account for different cloth materials during reconstruction. A recent work [Lahner et al. 2018] also captures and reconstructs garments in motion from 4D scans with a conditional adversarial neural network to add wrinkles to low resolution normal maps. All these methods perform a faithful reconstruction of the garment animation with the folds and wrinkles. However, it is still challenging to edit such captured garment sequence to generate new animations due to the loss of garment properties and environment information. In contrast, our method infers the garment intrinsic parameters from the garment shape and body motion, which can subsequently be edited and re-purposed to new motions to synthesize novel garment sequences.

*Motion Control via Neural Networks.* Our network design is largely inspired by the work of motion control via neural networks. Holden et al. [2016] propose a CNN framework to map user instructions to the full body motion to synthesize and edit character motion, but it is an offline framework which is not suitable for our real-time requirements. Subsequently, Holden et al. [2017] introduce a real-time motion control framework with the Phase-Functioned Neural Network (PFNN), whose weights can be dynamically changed as a function of the phase. They demonstrate PFNN yields superior performance over the standard networks and Recurrent Neural Networks (RNN) [Fragkiadaki et al. 2015] which use the phase as an additional input. Recently, Zhang et al. [2018] extend this idea and propose mode-adaptive neural networks (MANN) for quadruped motion control and Lee et al. [2018] extend the RNN based approach to multi-objective character control. Inspired by PFNN, we aim to control the garment shapes with the body motion. Instead of using a phase function, we utilize a motion encoder to learn a motion descriptor from the body movement as the coefficients to linearly blend the sub-networks in each layer to update the network weights.

## 3 APPROACH

### 3.1 Overview

Our method takes as input a single keyframe with a garment shape and its corresponding character motion. Our goal is to infer the intrinsic properties that are independent of the character body motion. The learned intrinsic vector can be further edited (e.g., by linear interpolation, etc.) and applied to a new motion to synthesize a perceptually-consistent garment shape in a differentiable fashion.

We adopt a motion-invariant autoencoder neural network for this task (Section 3.4). The encoder takes as input the garment shape descriptor (Section 3.3) and character motion and maps them into a latent space representing the intrinsic parameters. We use synthetic data from a physically based simulator to train the autoencoder. We densely sample the parameters for the simulation. Thus the generated dataset may cover the range of all possible manual edits. In order to make sure the latent space is invariant to the character motion, as shown in Fig 2, all frames generated by the same intrinsic parameters (i.e., simulator parameters, environment parameters, and garment material parameters) should be mapped to a single point in the latent space during training. The decoder acts as a *differentiable simulator*, which reconstructs a garment shape from a latent vector and target motion. To efficiently incorporate the motion information into the network, inspired by the Phase-Functioned Neural Network [Holden et al. 2017], we use a learned motion

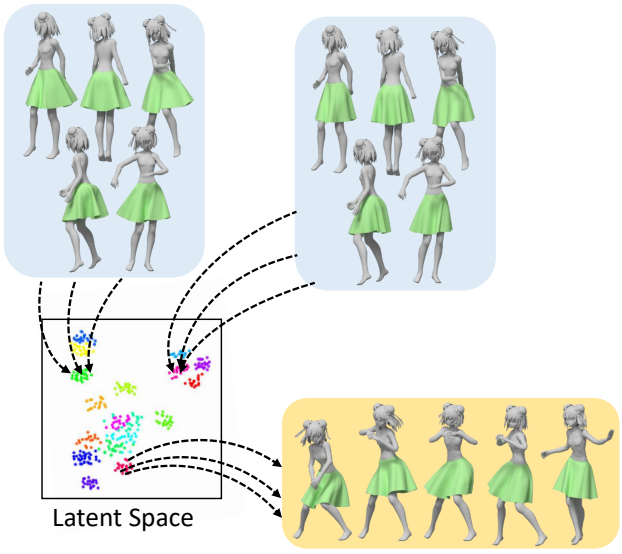


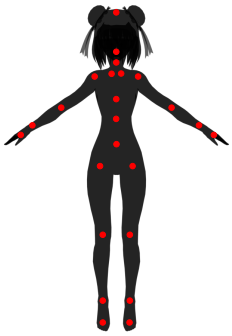

Fig. 2. We train a motion invariant encoding that maps the garment shapes with the same intrinsic parameters to the same location in the latent space, while factorizing the motion of the underlying character, and decodes a location in the latent space to various garment shapes using the provided character's motions. In the latent space, dots with the same color represent instances generated by the same intrinsic parameters.

descriptor as the coefficients to linearly blend the multiple sub-networks within each network layer to change the network behavior according to the motion. The encoder, decoder, and the motion descriptor are jointly trained to factor out the intrinsic property and motion-related information.

Our paradigm of motion-invariant autoencoder provides two advantages. First, after extracting the latent representation of intrinsic parameters from the raw frames, we obtain a much simpler space compared with the original 3D shape space which supports meaningful linear interpolation. Second, the motion-invariant latent representation can be coupled with a different motion to produce novel garment shape from the decoder. We consolidate our findings with thorough experiments (Section 4). Before describing the technical components in detail, we next introduce the specific representation choices used in this work.

## 3.2 Data Representation

Given a fixed character body shape, we assume the synthetic garment shape at a specific frame is determined by the body motion and intrinsic parameters, i.e., simulator parameters (e.g., time scale, max solving iterations, etc.), environment parameters (e.g., gravity, air drag, etc.), and garment material (e.g., bend, friction, etc.).



*Motion.* We describe the body motion as the pose aggregation of the current frame and past $W$ frames. The pose is represented by the 3D positions of body joints, so we have a $K \times 3$ pose matrix $P$ for a skeleton with $K$ joints. The motion signature is defined as the pose matrix for the current and past $W$ frames $M_{(W+1)\times K\times 3}$ to describe the status of a specific moment. All the poses are represented in the character space of the current frame. This definition implies that the dynamics of the garment are only related to the current and past $W$ frames.

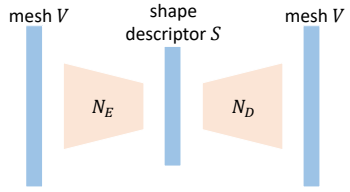
*Garment.* We assume the garment to be dressed on the character is deformed from a template mesh $(V_{tmp}, F_{tmp})$ where $V_{tmp}$ is a $N \times 3$ matrix that stores the 3D position of the vertices while $F_{tmp}$ stores the faces of the triangular mesh. At a frame with motion status $M$, the garment shape is represented by $V_M$, since the topology of the garment mesh remains unchanged from $F_{tmp}$. Garments are also represented in the character space of the current frame.

*Intrinsic parameters.* We treat everything independent of the body motion $M$ as intrinsic parameters. During data generation, explicit intrinsic parameters $\theta$ includes simulator parameters, environment parameters, and garment material parameters. Explicit intrinsic parameters sometimes are not continuous nor analytically differentiable (e.g., the number of solving iterations). Our goal is to learn a motion invariant representation as a latent vector $z$ in our motion invariant autoencoder.

*Dataset.* To train our network, we need a dataset that captures the dynamics of the garment under different parameters driven by different motions. With a given character and a given garment template, we run simulations with randomly sampled parameters $\theta$. We organize the dataset as a collection of $\{V, M, \theta\}$ including garment shape $V$, the motion signature $M$, and the simulation parameters $\theta$. The intrinsic vector $z$ can be interpreted as a latent representation of $\theta$ learned by our network.

## 3.3 Shape Feature Descriptor

Without loss of generality, our system takes 3D triangular mesh as the input representation of garment shape. Observing the space of valid 3D garment shape is a very small subset of the full vertices space $\mathbb{R}^{|V_{tmp}|}$, we train an autoencoder to extract low dimensional shape feature descriptors from the garment meshes with our synthetic data. It not only reduces the dimension of the input data, but more importantly ensures that our core network architecture remains unchanged across garments of different types or resolutions.




As shown in the inset, we use multilayer perceptrons (MLP) to encode $(N_E(\cdot))$ and decode $(N_D(\cdot))$ the input shape $V \in \mathbb{R}^{N\times 3}$ into/from a low dimensional shape descriptor $S \in \mathbb{R}^{N_S}$. We train this network via the combination of two types of loss, i.e., a $L_2$ loss between the 3D vertex positions of the input shape and the shape after encoding and decoding, and a $L_2$ loss between the mesh Laplacian [Taubin 1995] on the vertices of those two to preserve surface details. Thus, the combined loss function is defined as:

$$\|V - N_D(N_E(V))\|_2 + \omega \cdot \|\Delta(V) - \Delta(N_D(N_E(V)))\|_2,$$

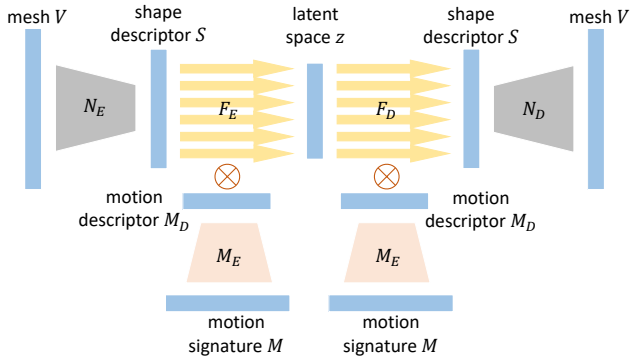



Fig. 3. A visual diagram of our motion invariant encoding network as discussed in Section 3.4. We use a learned motion descriptor as the coefficients to linearly blend the multiple sub-networks within each network layer to dynamically change the network behavior according to the motion.

where $\Delta(\cdot)$ is the Laplacian operator. The shape descriptor $S$ hence is defined as $S = N_E(V)$. We train our shape descriptor using only the garment shapes from our dataset. Once the training converges, we fix the weights of $N_E$ and $N_D$ for the rest of the pipeline.

### 3.4 Motion Invariant Encoding

Given the learned shape descriptor $S$, we now use the motion signature $M$ to drive a neural network to extract the motion-independent representation $z$. As shown in Fig. 3, our autoencoder architecture has an encoder $F_E(S \mid M) = z$ mapping the input shape descriptor into the latent space $z$, and a decoder $F_D(z \mid M) = S$ reconstructs the shape descriptor from $z$. The motion signature $M$ is provided as a condition for the mapping functions, as it provides the dynamic cues for the garment shape for a specific frame. Different intrinsic parameters with different motions may result in similar garment geometry (e.g., running slowly with high air-drag force may produce a similar garment shape as running fast with low air-drag force), so it is crucial to incorporate the motion into the network to resolve the ambiguity.

Inspired by the Phase-Functioned Neural Network [Holden et al. 2017] whose weights can be dynamically changed as a function of phase, we involve the motion into the network by dynamically updating the network weights as a function of motion signature. Intuitively, our network has multiple sub-networks within each layer, and each sub-network is associated with a blending coefficient. The final network weights are computed by linearly blending the sub-network weights. We employ the motion signature to control the coefficients so as to drive the network weights. Because the coefficients should be non-negative and normalized to 1 but the dimension of motion signature is very large (e.g. $\gg 10^3$ for $K = 24$ joints and $W = 100$ past frames), we introduce the motion descriptor $M_D \in \mathbb{R}^{N_M}$ as a normalized and compact latent representation for the motion signature $M \in \mathbb{R}^{(W+1)\times K\times 3}$, where $N_M$ is the number of sub-networks for each layer.

As shown in Fig. 3, a MLP motion encoder $M_E(\cdot)$ is applied over the motion signature $M$ to derive the motion descriptor $M_D$, i.e., $M_D = M_E(M)$. We adopt a Softplus activation [Dugas et al. 2001] after the last layer in $M_E$ to make it non-negative.

Now the building block of $F_E(\cdot)$ and $F_D(\cdot)$ can be annotated as

$$U(x \mid \gamma_1, \gamma_2, N_M, M_D, \Psi(\cdot)) = \Psi\left(\sum_{i=1}^{N_M} M_D^i \cdot \tilde{L}^i_{\gamma_1,\gamma_2}(x)\right),$$

where $\gamma_1/\gamma_2$ are the input/output dimension, $\tilde{L}_{\gamma_1,\gamma_2}(\cdot)$ is a standard linear transformation from $\mathbb{R}^{\gamma_1}$ to $\mathbb{R}^{\gamma_2}$ and $\Psi(\cdot)$ is an activation function. Our network consists of such a *U-block* sequence. The input/output dimension $\gamma_1/\gamma_2$ and the activation function $\Psi$ of each block may vary, but the size of motion descriptor $M_D$, $N_M$, remains the same for each *U-block*.

The motion-invariant autoencoder $F_{E/D}(\cdot)$ and the motion encoder $M_E(\cdot)$ are jointly trained. During the training, we pack a group of $\{V_i, M_i\}$ generated with the same $\theta$ in a single batch, i.e., the batch data correspond to the same intrinsic parameter. The training loss is defined as

$$E = \mathbf{Var}(z) + \lambda \cdot \|S_i - S_i^*\|,$$

where $S_i = N_E(V_i)$ is the input shape descriptor, $S_i^* = F_D(z_i \mid M_i)$ is the recovered shape descriptor, $z_i = F_E(S_i \mid M_i)$ is the latent vector, and **Var** is the variance of $z_i$ in the batch. The first term aims to minimize the variance in the latent space within the same batch, as the input $\{V_i, M_i\}$ generated with the same $\theta$ are supposed to be mapped to the same location in the latent space. The second term acts as a regularizer to penalize the difference between the input shape descriptor and the recovered one, so as to ensure the latent space will not degenerate to zero or an arbitrary constant. A weight coefficient $\lambda = 10^{-1}$ is applied to balance the scale of the two terms.

### 3.5 Refinement

Given a latent vector $z$ and motion descriptor $M$, our system predicts a plausible clothing mesh $V$ via $N_D(F_D(z \mid M))$ consistent with the motion. As our network is trained with synthetic data from a physically-based simulator, the simulation inevitably introduces 'noise' to the dataset. This is because, in the simulation, the momentum of some movements may last longer than $W$ frames and the simulator itself may suffer from numerical instability. This leads to the fact that $V$ may not be a perfect function of $M$ and $\theta$. Therefore, it is not guaranteed that the predicted garment shape is always collision-free (as shown in Fig. 4), especially when the garment is tight. Consequently, we apply an efficient refinement step to drag the garment outside the body while preserving its local shape feature. Specifically, given a body shape $B$ and the inferred garment mesh $V$, we detect all the garment vertices inside $B$ as $\tilde{V}$. For each

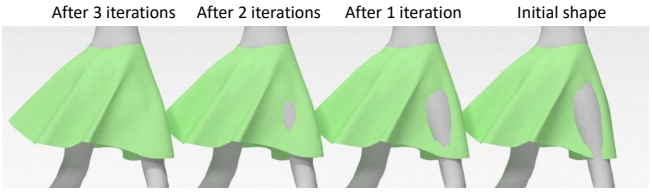



Fig. 4. From right to left, we progressively solve the interpenetration between the garment and the body.

vertex $\tilde{v}_i \in \tilde{V}$, we find its closest point over the body surface with position $v_i^B$ and normal $n_i^B$. Then we deform the garment mesh to update the garment vertices $V^*$ by minimizing the the following energy:

$$E_B = \|\Delta(V^*) - \Delta(V)\| + \sigma \sum_{i \in \tilde{V}} \|\tilde{v}_i^* - (v_i^B + \epsilon n_i^B)\|.$$

The first term penalizes the Laplacian difference between the deformed mesh and the inferred mesh, and the second term forces the garment vertices inside body to move outwards with $\epsilon$ being a small value ensuring the garment vertices lie sufficiently outside the body. In rare cases of heavy interpenetration, the closest point $v_i^B$ on body may be unwanted, i.e., on the opposite side of the body. To solve this issue, one can start from a vertex $v_i$ outside the body and check the interpenetration of the other vertices in an ascending order of geodesic distances to $v_i$. Then the optimization can be performed progressively for the newly found inside vertices. This usually requires less than 5 iterations to converge to a collision-free garment shape.

### 3.6 Implementation

We now provide details about the data generation process and the network architecture.

*Data Generation.* We generate a dataset supporting three different types of garments, i.e., skirt, sailor collar, and long skirt. Garments of the same type are deformed from the same template, which means the size is fixed for each type of garment among the dataset. The garment mesh has 5787 vertices for the skirt, 1638 vertices for the sailor collar, and 12540 vertices for the long skirt. We dress the character in a shrink-and-inflate fashion as [Wang et al. 2018]. We apply the MikuMikuDance motion [Wikipedia contributors 2019] to drive the animation of our character. We select 6 clips from MikuMikuDance which gives about $30K$ motion frames in total.

We use Maya Qualoth [Autodesk Inc. 2015] cloth simulation plug-in to generate the deformed garment shape. We sample three types of parameters for each round of simulation, i.e., simulator parameters, environment parameters, and garment material parameters. Simulator parameters include compression, proximity criterion, proximity force, sharp feature force, frame samples, time scale, max CG iteration and CG accuracy. Environment parameters include air drag and gravity. Garment material parameters include density, stretch, stretch damp, shear, bend, bend damp, bend yield and friction. Before generating the dataset, we find the proper range of each parameter by setting other parameters to a base value and varying the target parameter until the simulator fails to produce reasonable results or crashes. We randomly sample values within the calculated range for each parameter. We sampled 200 combinations of the parameters, and the parameters are fixed during each round of simulation. This provides us about $200 \times 30K = 6M$ pairs of $\{V, M, \theta\}$. It takes 24 hours for a CPU cluster with 6 Intel(R) Xeon(R) E5-2643 cores to generate this dataset[1].

[1]Code and sample data can be found at http://geometry.cs.ucl.ac.uk/projects/2019/garment_authoring/.

*Network Architecture.* The networks for learning the shape descriptor are composed of *linear blocks* which are linear layers followed by Parametric Rectifying Linear Unit (PRELU) activations [He et al. 2015] and batch normalization. Note that to enhance the compactness of descriptor, we adopt RELU−6 activation [Krizhevsky and Hinton 2010] after the last layer of the encoder network. Specifically, the encoder, $N_E$, takes as input a $N \times 3$-dimensional vector reshaped from the vertices of the garment shape (e.g., $N = 5787$ for skirt) and maps it to a $N_S = 50$ dimensional descriptor using 5 linear blocks (the output dimension size is 5000 in the first block and gradually decreased to 2000, 600, 200, and 50 in the remaining 4 blocks). The architecture of decoder, $N_D$, is symmetric with $N_E$.

The motion encoder network is also composed of the linear blocks. To ensure the motion descriptor to be non-negative, we adopt Softplus activation [Dugas et al. 2001] after the last layer of the motion encoder. After the Softplus activation, we linearly normalize the descriptor to unit length. Specifically, the motion encoder, $M_E$, takes as input a $(W+1) \times K \times 3$-dimensional vector reshaped from the motion matrix $M_{(W+1)\times K\times 3}$. As mentioned before, we have $W = 100$, $K = 24$ across our dataset. The motion encoder then maps it to the $N_M = 30$ dimensional motion descriptor space with 4 linear blocks (the output dimension size is 1200 in the first block and gradually decreased to 600, 120, and 30 in the remaining 3 blocks).

The motion invariant autoencoder consists of two networks, i.e., an encoder $F_E$ and a decoder $F_D$. Both $F_E$ and $F_D$ are composed of *U-blocks* as mentioned in section 3.4. Specifically, the encoder, $F_E$, takes as input a $N_S = 50$ dimensional shape descriptor vector and maps it to the $K_z = 100$ dimensional latent space with 5 *U-blocks* (the output dimension size is 100 for all the blocks). The network architecture of the decoder, $F_D$, is symmetric with $N_E$.

We first train the shape feature descriptor for 600 epochs with a learning rate of $10^{-3}$ and batch size of 64, and $N_E$ and $N_D$ are fixed for the rest of the training. After that, we train the motion encoder and the motion invariant encoding networks jointly for 600 epochs with the same learning rate and a batch size of 32. We use stochastic gradient descent for network back-propagation. We used PyTorch for implementation and the proposed network takes about 10 hours (for the shape descriptor) and 40 hours (for the rest) to train on each dataset with one Nvidia Titan X GPU. At runtime, the forward pass of the network takes, on an average, 24ms on the same GPU, and the refinement step takes, on an average, 40ms on a 48-thread Intel Xeon CPU, enabling an interactive authoring workflow for garment animation.

## 4 EVALUATION

We evaluate our method for three aspects: (i) the capacity of the shape feature descriptor; (ii) the capacity of the motion-invariant autoencoder; (iii) the applications in real garment animation authoring task. We evaluate our method qualitatively and quantitatively on different datasets. We split our synthetic dataset into training (95%) and testing sets (5%) so that no intrinsic parameters and body motions are shared across the splits.

*(i) Shape Feature Descriptor Evaluation.* We learn a low dimensional compact feature space to compress the input 3D shape. We utilize the combined loss with a Euclidean distance term and a

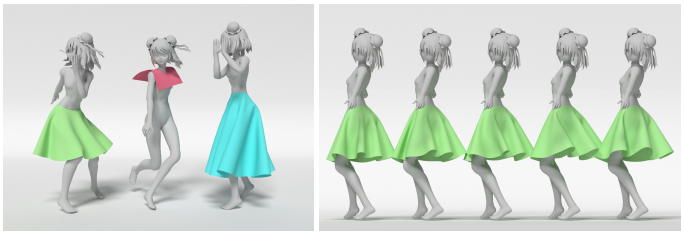


Fig. 5. (Left) We test our system on a dataset including three types of garments (skirt, sailor collar, and long skirt). The garment shapes in the dataset are generated by simulating with different simulator parameters, environment parameters, and material parameters. (Right) The variance in the intrinsic parameter space creates a variance in the garment shape space even for the same body motion sequence.

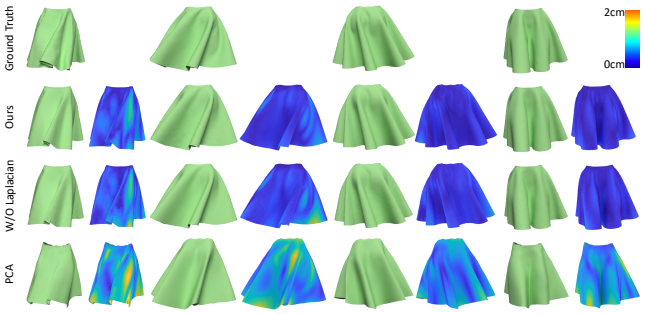



Fig. 6. For a given garment shape (top row), we show the reconstruction results using our learned shape feature descriptor with/without the Laplacian term (2nd and 3rd row) and using principal component analysis (4th row). A difference map is rendered side by side. The warmer color indicates a larger error. Our learned shape feature descriptor (2nd row) provides better visual quality for shape reconstruction.

Laplacian term to train our neural descriptor (Sec 3.3). As shown in the second row of Figure 6, our network successfully maps a 17361−dimensional 3D shape space to a 50−dimensional descriptor space and reconstructs the 3D shape close to the input from the low dimensional descriptor. The Laplacian term plays a vital role in the recovery of fine and sharp details (see the third row of Figure 6). As an alternative approach, the principal component analysis (PCA) performs linear compression by extracting bases and coefficients. However, we find that reconstruction with PCA bases and coefficients with the same dimension achieves lower reconstruction accuracy compared with our approach, as shown in the bottom row of Figure 6.

We also evaluate the effects of the dimension of shape descriptor by training the shape feature descriptor with different target dimensions (20/40/50/100/200). Quantitatively, as shown in Figure 7, reducing the dimension to 50 is a reasonable choice as further increasing the size has little improvement in the accuracy. We can also see that our method achieves much less error compared to the PCA reconstruction with the same dimension for coefficients.

Besides the reconstruction accuracy, we expect the learned descriptor space is 'convex' in $\mathbb{R}^{N_S}$, i.e., the decoder network should be able to produce smooth plausible transitions between two sampled garment shapes. In Figure 8, we show two interpolation examples. In each case, the two given shapes (leftmost and the rightmost one) are first passed through the encoder $N_E$ to calculate their descriptors. We then perform linear interpolation in the descriptor space and use the decoder, $N_D$, to map the interpolated descriptors back to the 3D shape space. We see that our decoder successfully produces smooth outputs in-between.

We also demonstrate the generalization ability of our shape descriptor network. As shown in Figure 9, our learned descriptor is robust to a 3D shape that is very different to our training data, or even not physically realistic. Our network successfully maps the *unseen* input 3D shape to descriptor space and reconstructs a similar but physically realistic garment.

*(ii) Motion-invariant Encoding Evaluation.* We train a motion-invariant autoencoder to factor out the intrinsic information from an input garment shape and its corresponding motion. Therefore we evaluate the network performance from two aspects: first, accuracy of factorization, i.e., whether the decoder can faithfully recover the input shape from the latent vector when given the corresponding motion data; and second, intrinsic-ness, i.e., whether the learned latent representation can be used to faithfully reconstruct the target shapes when given new motion data. As shown in Figure 10, for a given garment shape $V$ with a known motion $M$, our reconstruction $N_D(F_D(F_E(N_E(V) \mid M) \mid M))$ is able to produce a garment shape closely matching the input. When applied to new motion data, i.e., $N_D(F_D(F_E(N_E(V) \mid M) \mid M_*))$, the reconstructed garment shape is also almost the same as the groundtruth.

Next, we experiment with alternatives in the network architecture and training strategy. To demonstrate the efficacy of our motion-driven network design, we compare our result with the straightforward option, i.e., concatenating the motion signature with the shape descriptor as input and adopting a standard autoencoder [Umetani 2017] to learn the motion-invariant encoding. To evaluate our joint

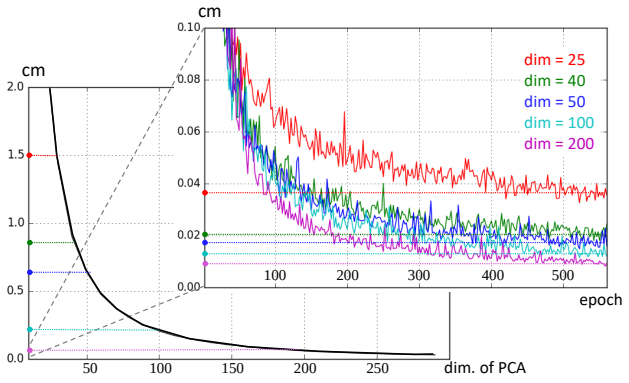



Fig. 7. Our learned shape feature descriptor quantitatively provides better accuracy compared with principal component analysis (PCA) using the same dimension for coefficients.

Fig. 8. We perform linear interpolation in the learned shape feature descriptor domain between two input garment shapes (leftmost and rightmost). Each row shows an example. The two input garment shapes are simulated with different parameters. The results indicate that our learned descriptor space is a convex space for reasonable garment shapes.

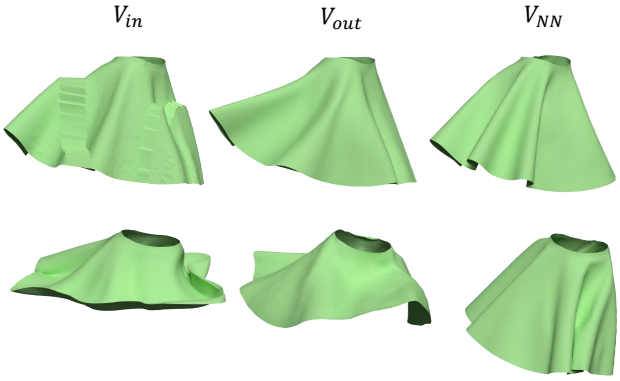



Fig. 9. If a garment shape $V_{in}$ is not covered by our dataset (i.e., a garment shape that is not physical realistic), our shape descriptor network maps it to the descriptor space and outputs a physically realistic $V_{out}$ to approximate the input shape. We also show the nearest shape $V_{NN}$ retrieved from the dataset.

training strategy, we test another standard strategy wherein the encoder and decoder are trained separately while using the distance in intrinsic parameter space to regularize the training. Specifically, since we have intrinsic parameters $\theta$ for each data entry, we can train the encoder with an embedding loss so that the latent vector $z$ preserves the Euclidean distance in the intrinsic parameter ($\theta$) space. Then we fix the encoder and separately train the decoder to reconstruct the shape descriptor from the latent vector $z$. As illustrated in Figure 11 (left), both of the alternatives converge to lower accuracy than our method.

An important parameter for our network is the number of frames $W$ used for motion signature. We evaluate the effects of $W$ by training with different $W$ settings (1/20/40/80/100). As shown Figure 11 (right), a larger $W$ tends to introduce higher accuracy. It is natural, as a larger $W$ indicates more identical motion signature for the current status. A small $W$ may introduce noise to the network, as the garment shape may vary a lot even if the recent body motion is similar. Such noise reduces the accuracy of the network training. We choose $W = 100$ for our network as a trade-off between the accuracy and computational efficiency.

Finally, the meshing quality of the reconstructed garment shape is another core concern when evaluating the performance of our system. In Figure 12, we qualitatively assess the meshing of the reconstructed garment shape by visualizing it with a colored dot texture. We apply UV parameterization on the template mesh, and

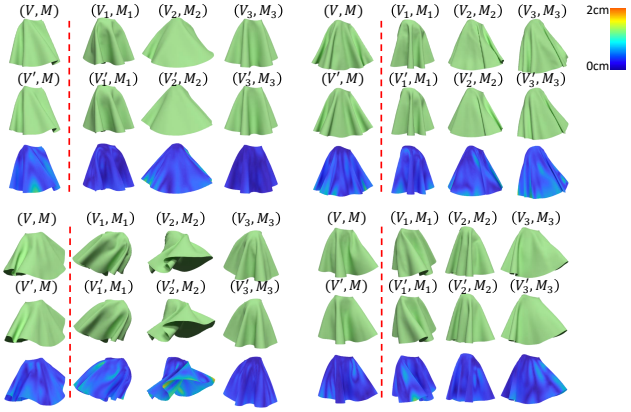



Fig. 10. Here we qualitatively show the reconstruction accuracy of our motion invariant encoding. For a given garment shape $V$ with known motion status $M$, we acquire its latent representation $z = F_E(N_E(V) \mid M)$. We present the reconstruction results $V' = N_D(F_E(z \mid M))$. We also decode $z$ with other motion statues $M_1/M_2/M_3$, so we have $V'_1/V'_2/V'_3$. Their groundturth reference from simulation is marked as $V_1/V_2/V_3$ in the figure. We show the difference maps in the 3$^{rd}$ row with the warmer color indicating a larger error. We show that our motion invariant encoding successfully factors out the intrinsic information from a given garment shape and its corresponding motion status. The learned intrinsic representation can then be used to recover the garment shape for any motion status.

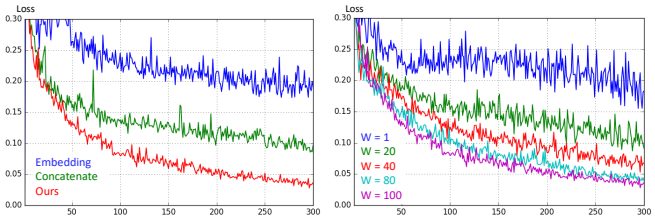



Fig. 11. (Left) comparisons with a different network architecture and training strategy. (Right) validation errors of using different frame numbers to encode the motion descriptor. Using a small frame number tends to reduce the accuracy in shape prediction.

the UV coordinates are copied to the reconstructed meshes since the topology of the new mesh remains unchanged. We can see that our system is able to produce consistent textures over different latent vectors and body motions.

*(iii) Application: Garment animation authoring.* With our neural network based motion-invariant encoding system, we enable three applications in the authoring of garment animation.

The *first application* is semi-automatic garment animation composition. The standard garment animation composition pipeline in modern CG industry requires a huge amount of user efforts. In the traditional pipeline, starting from the first frame, the user edits the template garment mesh to match the character pose. Then the user moves to the next keyframe where the body pose change exceeds a threshold value. After the garment on that keyframe is

Fig. 12. Our method produces consistent textures across different motion status. In this example, we parameterize the template mesh of the skirt (a truncated cone) and copy the UV coordinates to the vertices of the predicted shapes. We achieve a consistent texture for the composed garment animation.

modified, the garment shapes for intermediate frames are generated via linear interpolation in the garment shape space. The user then checks the intermediate frames to see if any adjustment is needed. If so, a new keyframe will be inserted between the two keyframes. Otherwise, the user may continue to the next keyframe. On average, one keyframe is essential for every five frames following such composition workflow, which indicates the *ratio of required keyframes* is 20%.

In contrast, our model enables a semi-automatic pipeline for the animation composition, largely reducing the user workload. To test our system in a real-world environment, we integrate our model into Autodesk Maya and build a user interface (UI) to allow users to test our system (as shown in Figure 14). Our UI allows the user to insert a keyframe, and modify its garment by selecting a desired design from the provided candidates (see the left window in Figure 14). The candidates are generated by random sampling from the latent space. Then our system automatically generates the garment animation for the whole sequence. The user can visually assess the animation result (see the right window in Figure 14). If not satisfied, the user is allowed to insert more keyframes, or delete frames to achieve his/her desired effects. Please refer to the supplementary video to see each of these editing options.

In Figure 13, we show that our method successfully performs physically plausible interpolation across 96 frames with only 2 keyframes, while linear interpolation in 3D space or shape descriptor space causes significant artifacts. Thanks to our motion-invariant encoding that learns a motion independent intrinsic representation from keyframes and deploys it to infer intermediate garment shapes, our system significantly reduces the *ratio of required keyframes* to $1-2\%$ in all of our testing cases. Figure 15 illustrates another composition example with sailor collar and long skirt.

The *second application* is simplifying creation of garment loop-animations. Since our system generates garment shapes only regarding to the intrinsic latent vector and the current and past $W = 100$ motions, it indicates if the latent vector remains unchanged, the same motion signature will always result in the same garment shape. This assumption does not hold for a physics simulator, for example, even the body movement in the first frame may still remain momentum on a very late status. This fact brings a tricky problem in garment loop animation generation. In the traditional workflow, the user loops the body animation several times and runs garment simulation on the looping body movements, hoping that after a certain number of iterations, the garment animation may loop itself as well. However, it is nearly impossible to obtain perfect results, because, for different iterations, the motion accumulated from the staring frame is different. The instability of numerical computation will further boost this difference, which leads the animation still diverging even after a huge number of iterations. The manual effort is inevitably needed to modify the garment animation to close the gap at last. In Figure 16, we show that our system naturally allows consistent loop animation composition once a looping body animation is given. Please refer to the supplementary video which shows the looping animation.

The *third application* is automatically adjusting garment animation under modifications of input body animation. The need for modifying body motion may happen during garment animation composition from time to time, which arises from the pursuing of character stylization or aesthetic effect. In a traditional workflow, once the body motion is modified, the garment shapes of all the frames related to the modification need to be adjusted manually. In contrast, in our system, since the garment shape is generated from motion independent intrinsic representation, and automatically driven by the body motion, there is no manual effort required to adjust the garment animation if the character body motion is changed. As shown in Figure 17, the user can try with different body motion designs without any extra efforts (see also supplementary video).

*User experience and feedback.* We have invited several CG designers from industry to test our system in different garment authoring tasks. We received mostly positive feedback as all the designers felt that our semi-automatic workflow dramatically improves the efficiency and allows them to really focus on the *design*, instead of imagining the 3D shapes from frame to frame or painfully experimenting with a large number of parameters in a black-box simulator. Here is the quote from a product manager:

> This system armed a single designer to beat a traditional project team. Achieving consistent quality for a big project, i.e., a CG movie or a video game with more than 1000 animation clips, is very challenging. It is even more difficult to control the quality when the team size is over 50. By using the new workflow, thanks to the high efficiency, we may have a much smaller team and much shorter production cycle to get the job done.

## 5 CONCLUSION AND DISCUSSION

We presented a data-driven learning framework for obtaining a motion independent latent space that factors out intrinsic design parameters and character body motion in the context of garment animation composition. The learned motion-invariant network enables a novel workflow for garment animation composition that

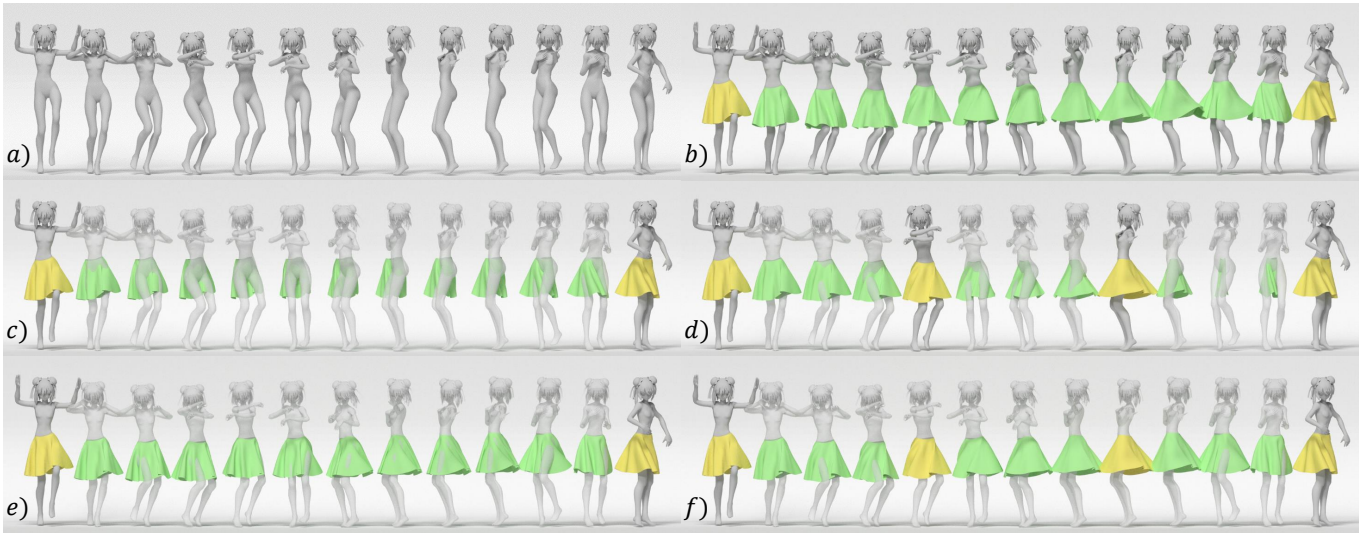



Fig. 13. We compare our approach with other baseline methods. For a given motion sequence (a), our system produces a garment animation sequence (b) from two keyframes (marked by yellow). Linear interpolation between the two keyframes in the 3D shape domain (c), or our learned shape feature descriptor domain (e) cannot produce reasonable results. Adding more keyframes may not lead the results (d) and (f) closer to the desired output (b). The sequence goes from right to left in this figure and we draw our results for every 8 frames.

allows users to improve the working efficiency by reducing the *keyframe ratio* from 20% to < 2%. The new system also seamlessly allows updating character body motion and creating loop animations. Finally, we evaluated our method, both quantitatively and qualitatively, in different usage scenarios and showed compelling results.

### 5.1 Limitation and Future Work

The main limitation of the proposed approach is that the learned network is character and garment template dependent. Although our system can work with a variety of body movements, for a different character with different body shape or wearing a different garment, we need to re-generate our dataset with the new assets and re-train our network. Luckily, in many CG industry pipelines, the character and its corresponding asset, i.e., garment, are usually fixed at the modeling stage and remains unchanged subsequently. However, it is still relevant to investigate the problem of generating different types of garment for different characters within one network. Advances in few-shot learning [Mason et al. 2018] may be a possible solution to effectively adapt a learned network to a new character and/or garment. Another limitation comes from the assumption that only the current frame and the past $W$ frames may have an impact on the garment dynamic. This makes the network inevitably lose some details and may result in interpenetration with the underlying body. An adversarial generative network may be helpful to solve this problem in the future. As our training data is generated using physically-based simulation, it avoids the heavy manual work for capturing real data, but still requires a long time to generate. Reducing the time for data generation is also an interesting direction.

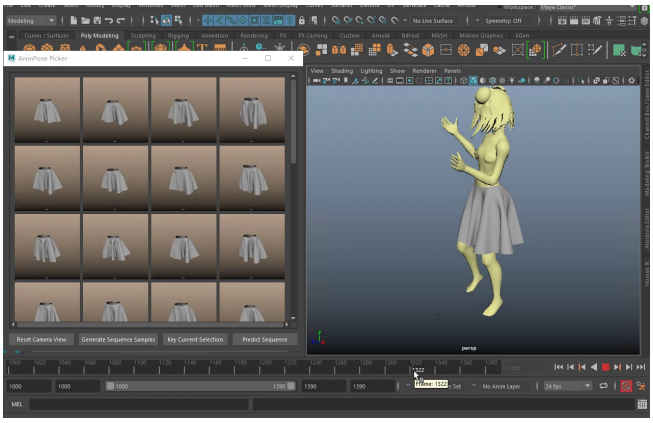


Fig. 14. Our user interface allows the user to edit the garment animation or the character motion and visualizes the updated garment animation at an interactive rate. Please refer to the supplementary video to see the interactive editing workflow.

## ACKNOWLEDGMENTS

We thank the reviewers for their comments and suggestions for improving the paper. The authors would also like to thank Zhiqing Zhong, Qi Tang, Jun Qian for making test cases for our system and Fox Dong for the video voice-over. This work is in part supported by miHoYo Inc., an ERC Starting Grant (SmartGeometry StG-2013-335373), ERC PoC Grant (SemanticCity), Google Faculty Award, Royal Society Advanced Newton Fellowship, gifts from Adobe, the New Faculty Start-Up Grant of University of Leeds, and NSF of China (No. 61772462, No. U1736217).

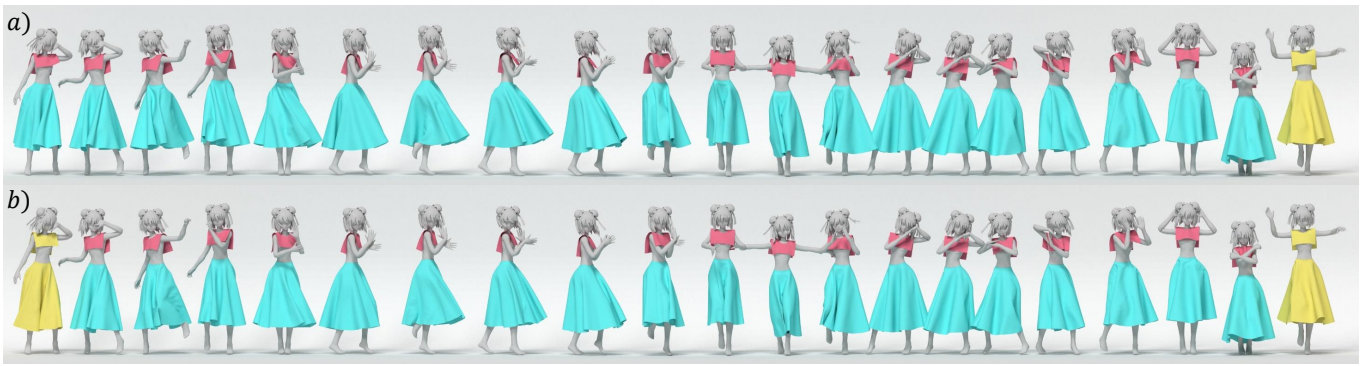



Fig. 15. Our result with different garment types. With a character wearing a sailor collar and a long skirt, a) the system generates a garment animation from one keyframe (marked in yellow); b) the user inserts another keyframe (with visually softer and heavier garment material, marked in yellow) and the system performs interpolation between the two keyframes. The animation sequence in this figure is illustrated from right to left.

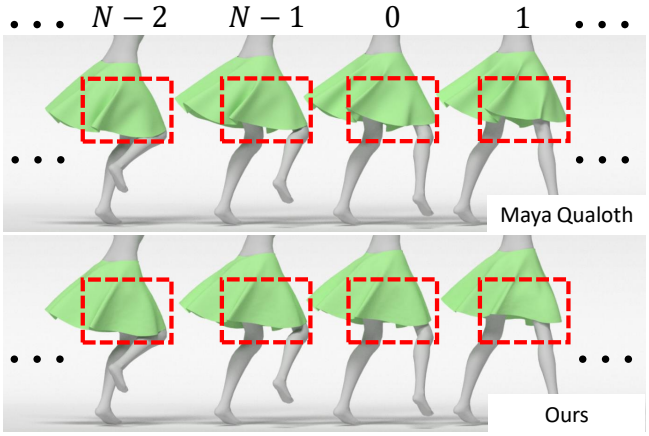



Fig. 16. Our approach naturally enables the composition of loop animation while simulation-based approach may not converge even after running for many iterations.

## REFERENCES


Autodesk Inc. 2015. Maya QUALOTH. http://www.qualoth.com/. (2015).

David Baraff and Andrew Witkin. 1998. Large Steps in Cloth Simulation. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98)*. 43–54.

Derek Bradley, Tiberiu Popa, Alla Sheffer, Wolfgang Heidrich, and Tamy Boubekeur. 2008. Markerless Garment Capture. *ACM Trans. Graph.* 27, 3 (Aug. 2008), 99:1–99:9.

Edilson de Aguiar, Leonid Sigal, Adrien Treuille, and Jessica K. Hodgins. 2010. Stable Spaces for Real-time Clothing. *ACM Trans. Graph.* 29, 4 (July 2010), 106:1–106:9.

Charles Dugas, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. 2001. Incorporating second-order functional knowledge for better option pricing. In *Advances in neural information processing systems*. 472–478.

Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. 2015. Recurrent Network Models for Human Dynamics. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV) (ICCV '15)*. 4346–4354.

Lawson Fulton, Vismay Modi, David Duvenaud, David I. W. Levin, and Alec Jacobson. 2019. Latent-space Dynamics for Reduced Deformable Simulation. *Computer Graphics Forum* (2019).

Rony Goldenthal, David Harmon, Raanan Fattal, Michel Bercovier, and Eitan Grinspun. 2007. Efficient Simulation of Inextensible Cloth. *ACM Trans. Graph.* 26, 3 (July 2007), 49:1–49:7.

Peng Guan, Loretta Reiss, David A. Hirshberg, Alexander Weiss, and Michael J. Black. 2012. DRAPE: DRessing Any PErson. *ACM Trans. Graph.* 31, 4 (July 2012), 35:1–35:10.

Fabian Hahn, Bernhard Thomaszewski, Stelian Coros, Robert W. Sumner, Forrester Cole, Mark Meyer, Tony DeRose, and Markus Gross. 2014. Subspace Clothing Simulation Using Adaptive Bases. *ACM Trans. Graph.* 33, 4 (July 2014), 105:1–105:9.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*. 1026–1034.

Daniel Holden, Taku Komura, and Jun Saito. 2017. Phase-functioned neural networks for character control. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 42.

Daniel Holden, Jun Saito, and Taku Komura. 2016. A Deep Learning Framework for Character Motion Synthesis and Editing. *ACM Trans. Graph.* 35, 4, Article 138 (July 2016), 138:1–138:11 pages.

Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. 2007. Skinning with Dual Quaternions. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games (I3D '07)*. 39–46.

Ladislav Kavan, Dan Gerszewski, Adam W. Bargteil, and Peter-Pike Sloan. 2011. Physics-inspired Upsampling for Cloth Simulation in Games. *ACM Trans. Graph.* 30, 4 (July 2011), 93:1–93:10.

Ladislav Kavan and Jiří Žára. 2005. Spherical Blend Skinning: A Real-time Deformation of Articulated Models. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games (I3D '05)*. 9–16.

Doyub Kim, Woojong Koh, Rahul Narain, Kayvon Fatahalian, Adrien Treuille, and James F. O'Brien. 2013. Near-exhaustive Precomputation of Secondary Cloth Effects. *ACM Trans. Graph.* 32, 4 (July 2013), 87:1–87:8.

Alex Krizhevsky and Geoff Hinton. 2010. Convolutional deep belief networks on cifar-10. *Unpublished manuscript* 40, 7 (2010).

Zorah Lahner, Daniel Cremers, and Tony Tung. 2018. DeepWrinkles: Accurate and Realistic Clothing Modeling. In *The European Conference on Computer Vision (ECCV)*.

Kyungho Lee, Seyoung Lee, and Jehee Lee. 2018. Interactive Character Animation by Learning Multi-objective Control. *ACM Trans. Graph.* 37, 6, Article 180 (Dec. 2018), 10 pages. https://doi.org/10.1145/3272127.3275071

Tiantian Liu, Adam W. Bargteil, James F. O'Brien, and Ladislav Kavan. 2013. Fast Simulation of Mass-spring Systems. *ACM Trans. Graph.* 32, 6 (Nov. 2013), 214:1–214:7.

Ian Mason, Sebastian Starke, He Zhang, Hakan Bilen, and Taku Komura. 2018. Few-shot Learning of Homogeneous Human Locomotion Styles. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 143–153.

Rahul Narain, Armin Samii, and James F. O'Brien. 2012. Adaptive Anisotropic Remeshing for Cloth Simulation. *ACM Trans. Graph.* 31, 6 (Nov. 2012), 152:1–152:10.

Gerard Pons-Moll, Sergi Pujades, Sonny Hu, and Michael J. Black. 2017. ClothCap: Seamless 4D Clothing Capture and Retargeting. *ACM Trans. Graph.* 36, 4 (July 2017), 73:1–73:15.

Tiberiu Popa, Q Zhou, Derek Bradley, V Kraevoy, Hongbo Fu, Alla Sheffer, and W Heidrich. 2009. Wrinkling Captured Garments Using Space-Time Data-Driven Deformation. *Computer Graphics Forum* 28 (03 2009), 427 – 435.

Igor Santesteban, Miguel A. Otaduy, and Dan Casas. 2019. Learning-Based Animation of Clothing for Virtual Try-On. *Computer Graphics Forum* 38, 2 (2019).

Leonid Sigal, Moshe Mahler, Spencer Diaz, Kyna McIntosh, Elizabeth Carter, Timothy Richards, and Jessica Hodgins. 2015. A perceptual control space for garment simulation. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 117.

Rasmus Tamstorf, Toby Jones, and Stephen F. McCormick. 2015. Smoothed Aggregation Multigrid for Cloth Simulation. *ACM Trans. Graph.* 34, 6 (Oct. 2015), 245:1–245:13.

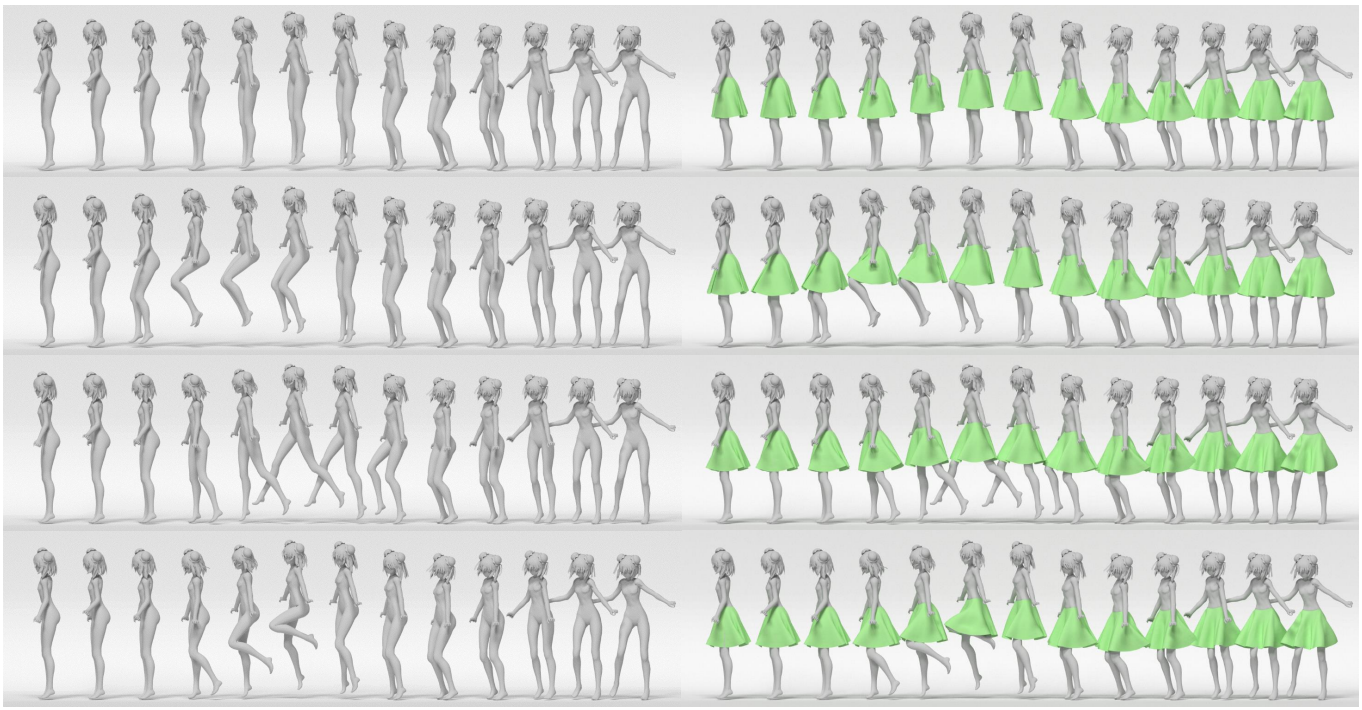


Fig. 17. Our workflow allows the artist to edit the body motion during the authoring of garment animation. Once the body motion is updated, the system automatically uses the modified motion signature to infer the garment shapes for the corresponding frames. The sequence goes from right to left in this figure.

Min Tang, Tongtong Wang, Zhongyuan Liu, Ruofeng Tong, and Dinesh Manocha. 2018. I-cloth: Incremental Collision Handling for GPU-based Interactive Cloth Simulation. *ACM Trans. Graph.* 37, 6 (Dec. 2018), 204:1–204:10.

Gabriel Taubin. 1995. A signal processing approach to fair surface design. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. ACM, 351–358.

Nobuyuki Umetani. 2017. Exploring generative 3D shapes using autoencoder networks. In *SIGGRAPH Asia 2017 Technical Briefs*. ACM, 24.

Huamin Wang, Florian Hecht, Ravi Ramamoorthi, and James F. O'Brien. 2010. Example-based Wrinkle Synthesis for Clothing Animation. In *ACM SIGGRAPH 2010 Papers (SIGGRAPH '10)*. 107:1–107:8.

Huamin Wang and Yin Yang. 2016. Descent Methods for Elastic Body Simulation on the GPU. *ACM Trans. Graph.* 35, 6 (Nov. 2016), 212:1–212:10.

Tuanfeng Y. Wang, Duygu Ceylan, Jovan Popović, and Niloy J. Mitra. 2018. Learning a Shared Shape Space for Multimodal Garment Design. *ACM Trans. Graph.* 37, 6, Article 203 (Dec. 2018), 13 pages. https://doi.org/10.1145/3272127.3275074

Nicholas J. Weidner, Kyle Piddington, David I. W. Levin, and Shinjiro Sueda. 2018. Eulerian-on-lagrangian Cloth Simulation. *ACM Trans. Graph.* 37, 4 (July 2018), 50:1–50:11.

Ryan White, Keenan Crane, and D. A. Forsyth. 2007. Capturing and Animating Occluded Cloth. *ACM Trans. Graph.* 26, 3, Article 34 (July 2007).

Wikipedia contributors. 2019. MikuMikuDance. https://en.wikipedia.org/wiki/MikuMikuDance. (2019). [Online; accessed 7-May-2019].

Weiwei Xu, Nobuyuki Umentani, Qianwen Chao, Jie Mao, Xiaogang Jin, and Xin Tong. 2014. Sensitivity-optimized Rigging for Example-based Real-time Clothing Synthesis. *ACM Trans. Graph.* 33, 4 (July 2014), 107:1–107:11.

Jinlong Yang, Jean-Sébastien Franco, Franck Hétroy-Wheeler, and Stefanie Wuhrer. 2018. Analyzing clothing layer deformation statistics of 3d human motions. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 237–253.

He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. 2018. Mode-adaptive Neural Networks for Quadruped Motion Control. *ACM Trans. Graph.* 37, 4, Article 145 (July 2018), 145:1–145:11 pages.