AgentDress: Realtime Clothing Synthesis for Virtual Agents using Plausible Deformations

Nannan Wu, Qianwen Chao, *Member, IEEE,* Yanzhen Chen, Weiwei Xu, *Member, IEEE,*
Chen Liu, Dinesh Manocha, *Fellow, IEEE,* Wenxin Sun, Yi Han, Xinran Yao, Xiaogang Jin*, *Member, IEEE*

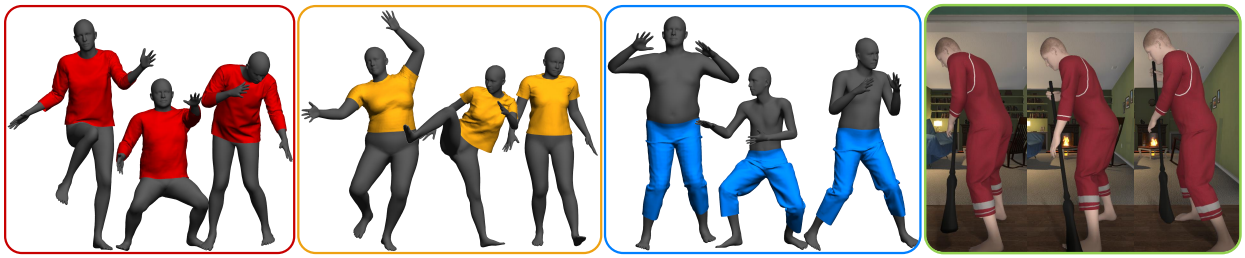


Fig. 1. Our method can generate real-time clothing animation results with detailed wrinkles for various body poses and shapes with different clothing types on a commodity CPU. Our method can also be applied in VR scenarios (right image).

Abstract— We present a CPU-based real-time cloth animation method for dressing virtual humans of various shapes and poses. Our approach formulates the clothing deformation as a high-dimensional function of body shape parameters and pose parameters. In order to accelerate the computation, our formulation factorizes the clothing deformation into two independent components: the deformation introduced by body pose variation (Clothing Pose Model) and the deformation from body shape variation (Clothing Shape Model). Furthermore, we sample and cluster the poses spanning the entire pose space and use those clusters to efficiently calculate the anchoring points. We also introduce a sensitivity-based distance measurement to both find nearby anchoring points and evaluate their contributions to the final animation. Given a query shape and pose of the virtual agent, we synthesize the resulting clothing deformation by blending the Taylor expansion results of nearby anchoring points. Compared to previous methods, our approach is general and able to add the shape dimension to any clothing pose model. Furthermore, we can animate clothing represented with tens of thousands of vertices at 50+ FPS on a CPU. We also conduct a user evaluation and show that our method can improve a user's perception of dressed virtual agents in an immersive virtual environment (IVE) compared to a realtime linear blend skinning method.

Index Terms—clothing animation, virtual agents, social VR, virtual try on clothing shape models.

1 Introduction

There is considerable interest in generating human-like virtual agents for AR and VR applications. These agents are used to generate immersive social experiences for games, training, entertainment, virtual space visitations, or social-phobia treatments. In order to maintain the sense of presence in virtual environments, it is essential that these virtual agents look realistic and interact in a plausible manner [2].

There has been a great deal of work on improving the realism of virtual humans in terms of rendering, body shapes, facial expressions, hair, locomotion, etc. A key issue is related to dressing three-dimensional virtual humans using garments and animating the cloth deformation corresponding to draping and wrinkles. This is crucial because up to 80% of a human body can be covered by clothing. As virtual agents move, bend, or interact with the environment, the clothing folds, wrinkles, and stretches to conform to the virtual agents' poses. This is also important in the context of efficient virtual try-on simulation. In these applications, the user can experience the fit of several garments on their 3D models or avatars. The goal is to experience real-life clothes on virtual models and evaluate the garment behavior when a user moves around or stands in different poses. Recently, many AR-based interfaces have been proposed for virtual try-on. All these applications must be able to simulate or animate the cloth and garments at interactive rates.

Physics-based simulation methods directly model the non-linear behavior of clothing and contacts to generate realistic cloth simulations [3, 13, 46]. However, high-resolution clothing meshes and computationally expensive nonlinear solvers are frequently used, making it difficult to simulate clothing at interactive rates (i.e. 30fps or more). Moreover, dressing bodies of different shapes requires a separate simulation or setup for each body shape. This makes it difficult to use physics-based methods for real-time VR or virtual try-on applications.

Many data-driven clothing animation methods synthesize the cloth deformation from pre-computed clothing deformation samples for different body poses [36, 75]. Peng et al. [27] considered the effect of body motion and shape on cloth deformation, and resized the garment to fit the body. However, their method cannot achieve real-time performance Recently, many learning-based or neural methods [38, 49, 54] have been proposed for realtime cloth synthesis. However, these methods need to use a large training database, which can be hard or time consuming to generate. Furthermore, most neural methods tend to use powerful desktop GPUs at runtime for real-time performance. These approaches may not be practical for VR or AR applications, which run on untethered all-in-one devices like Oculus Quest VR HMD or Hololens AR Headsets. As a result, we need fast solutions that have low computational runtime requirements on CPUs.

Main Results: In this paper, we present a novel CPU-based real-time algorithm for cloth synthesis for VR and virtual try-on applications.

- *N. Wu, Y. Chen, W. Xu, C. Liu, W. Sun, Y. Han, X. Yao and X. Jin are with State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310058, P. R. China. E-mail: wunannanzju@gmail.com, jin@cad.zju.edu.cn.*
- *Q. Chao is with the Department of Computer Science, Xidian University, Xi'an 710038, P. R. China.*
- *C. Liu is also with Zhejiang Linctex Digital Technology Co. Ltd., P. R. China.*
- *D. Manocha is with the Department of Computer Science, University of Maryland, College Park, MD 20742. E-mail: dm@cs.umd.edu.*
- *X. Jin is the corresponding author.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

We use a data-driven approach and present a new method to simulate plausible deformation effects at an interactive rate. We factorize the clothing deformation into two independent components: the pose-dependent deformation and the shape-dependent deformation. We refer to the pose-dependent deformation as the *clothing pose model*, which is used to predict clothing deformations under various poses; we refer to the shape-dependent deformation as the *clothing shape model*, which can predict clothing deformations under various shapes. We also present a clothing synthesis scheme that combines these two components through Taylor expansion. Our method adopts a pose- and shape-dependent skinning scheme for clothing synthesis to meet the needs of real-time virtual try-on on a CPU for synthetic bodies of various shapes and poses (as shown in Figure 1). The three novel components of our work are:

Clothing shape model: We present a novel pose-independent clothing shape model. Given a set of clothing instances simulated on bodies with a specific pose and various shapes, we first reduce the dimensionality with Principal Component Analysis (PCA). Next, we map the body shape parameters to the coefficients of the PCA basis of the clothing deformation space. Therefore, given a set of body shape parameters, we can predict the corresponding clothing deformation result.

Taylor expansion to approximate clothing deformations: We present a real-time clothing synthesis method by considering both the pose-dependent deformation and the shape-dependent deformation using Taylor expansions. We represent clothing deformation as a function of body shape parameters β and body pose parameters θ, $f(\beta,\theta)$. The clothing deformation in the neighborhood of the given anchoring point (β_0,θ_0) is approximated with a Taylor expansion. The partial derivatives of the clothing deformation are calculated numerically using the clothing pose model and the clothing shape model to predict $f(\beta_0,\theta_0+\Delta\theta)$ and $f(\beta_0+\Delta\beta,\theta_0)$, respectively. Given the new parameters $(\beta=\beta_0+\Delta\beta,\theta=\theta_0+\Delta\theta)$, we synthesize the clothing deformation by blending the Taylor expansion results from the nearby anchoring points. Accordingly, our approach can add the shape dimension to any clothing pose model. Moreover, we use a sensitivity-based measurement to compute the distance between the input and the anchoring points and to calculate the blending weights.

Sampling scheme: We present a pose space analysis method to generate a compact example database in parallel, which can significantly reduce the offline computational cost. In order to generate plausible results, we cluster the pose space into a small number of clusters. We use the cluster centers to calculate the anchoring points and thereby build a compact database. This sampling scheme can also be used to generate the database of the sensitivity-optimized rigging (SOR) method [75], and thereby significantly improving their results.

Our approach is general and provides a universal scheme to add the shape dimension to any clothing pose model with a compact database. We validate our method with SOR [75] and a sequence of simulated clothing instances. Our results show that, with a small number of anchor points (approximately 150), our method can generate realistic clothing deformations with an extra time consumption of 4ms per frame. On a commodity CPU, we obtain 56 FPS for a long-sleeved shirt with 12K vertices. We also perform a preliminary user study in an immersive VR setting and highlight the perceptual benefits of our approach compared to prior interactive methods based on linear blend skinning.

2 Related Work

Physics-based clothing simulation. In the past two decades, following the seminal work by Baraff et al. [3], physics-based clothing simulation has become a hot topic in the computer graphics community. The following works focus on integration methods [17, 32, 67], strain limiting [24, 42, 51, 63, 65, 71], and various clothing simulation models [8, 12, 13, 16, 26, 29, 31, 68, 79]. While these methods can produce highly realistic clothing deformations, they are typically quite time-consuming, especially with high-resolution cloth meshes. Many acceleration methods have been developed, including the projective dynamics method [7, 40], where integration is interpreted as an optimization problem; the Chebyshev semi-iterative approach, which accelerates the projective dynamics method [18, 69]; and position-based dynamics [45], where internal forces are replaced by position-based constraints to achieve both efficiency and stability. Recently, parallel GPU-based methods have been developed for implicit integration and contact handling [25, 39, 57, 59, 60], which perform implicit integration and accurate collision handling (including self-collisions). The underlying collision queries are performed using CCD tests that involve use of algebraic solvers and reliable computations [44, 58] for the elementary tests. The overall simulation algorithms exploit the parallelism on one or more GPUs and can accurately simulate at $2-10$ fps on high-end desktop GPUs. The actual running time can vary based on mesh resolution as well as the number of colliding configurations between triangles. Most VR applications require 30 fps (or higher performance) and current physics-based methods cannot offer such performance. Furthermore, in many applications, we need to perform cloth simulation on a mobile device (e.g., a smartphone) and we need methods that have a lower computational overhead. The time-consuming nature of collision processing has also prompted many researchers to focus on developing accelerated data structures such as bounding-volume hierarchies [37], distance fields [19], shape approximation with simple primitives [62, 74], and other spatial partitioning methods [61]. Combined with position-based dynamics [45], these methods can achieve real-time clothing animation with plausible dynamics for a medium resolution mesh. However, the resulting clothing animation lacks detailed wrinkles.

Data-driven clothing animation. Data-driven clothing animation techniques have received considerable attention in recent years for real-time applications that require high fidelity. De Aguitar et al. [15] reduced the dimension of cloth space and body space with PCA and then learned a conditional dynamical model of cloth in the low-dimensional linear cloth space. Their method is fast and stable but cannot generate clothing deformations with highly detailed wrinkles. Wang et al. [70] regarded wrinkles as a function of local joint angles and augmented coarse simulations with detailed wrinkles from a pre-computed wrinkle database. Kim et al. [36] exhaustively searched a motion graph to generate realistic secondary motion of clothing deformations. However, both the memory space and the computational resources for the database are prohibitively high. Hahn et al. [30] simulated clothes in low-dimensional linear subspaces learned from performing PCA on clothing instances in different poses. While they can reproduce detailed folding patterns with only a few bases, the method is still too costly to be used in real-time clothing animation. To balance speed and quality, Xu et al. [75] introduced real-time example-based clothing synthesis using sensitivity-optimized rigging to achieve physically plausible clothing deformation. Given a set of pre-computed example clothing deformations sampled at different poses, the method first rigs the example clothing shapes to their poses with the underlying body skeleton at each example pose in the offline stage. At runtime, the method synthesizes a clothing deformation of the input pose by blending skinned clothing deformations computed from nearby examples. Jin et al. [33] represented clothing shapes as offsets from the underlying body surface, and used convolutional neural networks to learn pose-dependent deformations in the image space. Gao et al. [20, 21] proposed a novel deformation representation along with a sparse blending method that can compactly models the target deformation with a small number of deformation modes. Wang et al. [73] learned a motion independent latent space that is used for interactive authoring of garment animation for an input body animation sequence. Other methods are based on neural networks designed for triangle meshes and used to predict high-resolution clothing deformations [9, 11].

In addition to the significant advances achieved in pose-dependent clothing animation, variability in the human body shape has also been considered in clothing animation for virtual try-on applications. Inspired by SCAPE [1], Peng et al. [27] introduced a model of clothing animation called DRAPE (DRessing Any PErson), which separated clothing deformations due to body shape from those due to pose variation. DRAPE can fit avatars of various poses and shapes with customized garments and change the clothing model according to the body shape. Recently, many techniques have been proposed for learning-based clothing simulation. Wang et al. [72] learned a shared shape

space in which users are able to indicate desired fold patterns simply by sketching, and the system generates corresponding draped garment and body shape parameters. However, their approach cannot generate clothing deformations corresponding to different poses. Inspired by SMPL [41], Santesteban et al. [54] introduced a learning model of cloth drape and wrinkles. The key innovation of their method is that they added corrective displacements caused by body shape and pose variations to the template cloth mesh and then deformed the template mesh using a skinning function. In many ways, our approach is complimentary to this method. A limitation of their method is that the training dataset is prohibitively large, which is similar to other learning-based or neural methods [28, 38, 49, 66]. Moreover, most of these neural methods need to use a desktop GPU for realtime runtime performance and may not be practical for many VR and AR applications. By treating clothing as an extra offset layer from the body, Ma et al. [43] trained a conditional Mesh-VAE-GAN to learn the clothing deformation from the SMPL body model. This work has the limitation in that the level of geometric details they can achieve is upper-bounded by the mesh resolution of SMPL. Yang et al. [76] modeled the clothing layer as an offset from the body and performed PCA to reduce the self-redundancies. In contrast, we perform PCA directly on the coordinates of clothing deformations to compute our clothing shape model. Many other works focused on cloth reconstruction from a single image or scan data [48, 50, 52], as well as cloth material recovery from video [77].

Sensitivity analysis. Sensitivity analysis was originally used to determine the impact of the input data on the output results in linear programming problems [53], and it has been widely used to solve optimization problems in the field of graphics, including shell design [35], composite silicone rubber design [78], and robotics design [23, 80]. Sensitivity analysis was first introduced to the clothing simulation community by Umetani et al. [64] to build up a mapping between variations of 2D patterns and 3D clothing deformations to achieve interactive clothing editing. After that, Xu et al. [75] proposed a technique called sensitivity optimized rigging (SOR) to perform real-time clothing animation. They use sensitivity to both rig the clothing instances in example poses and find the example poses nearest to the input pose. In this paper, we use sensitivity to find the anchoring points nearest to the input pose and shape.

Taylor expansion. A complex function can be approximated by its first order Taylor expansion in a neighborhood of the keypoint locations. Such a method has been applied in the field of simulation and animation to solve various approximation problems. To generate real-time facial animations, Barrielle et al. [5] applied first-order Taylor approximation to the computations of the Singular Value Decomposition, thereby significantly accelerating the simulation of volumetric forces. Shen et al. [55] adopted a finite Taylor series approximation of the potential energy to avoid numerical singularity and instability during the simulation of inextensible ribbon. In the cloth simulation community, Taylor expansion is generally used to make the first order approximation of the internal force. To solve the nonlinear equation involved in the implicit backward Euler method, Baraff et al. [3] applied a Taylor series expansion to the force acting on the cloth and made the first order approximation, which leads to a linear system. As a result, their cloth simulation system can handle large time steps in a stable manner. Chen et al. [10] proposed a fully geometric approach to simulate inextensible cloth that is subjected to a conservative force. They use Taylor expansion to linearize the constraint that preserves isometric deformations. In this paper, we use Taylor expansion to factor the clothing deformation (which is a complex high-dimensional nonlinear function without an analytical expression) into two parts, as introduced by body shape and pose variations.

3 METHOD

3.1 Taylor Expansion For Clothing Deformation

Algorithm 3.1 and Fig. 2 illustrates the pipeline of our approach. Our system consists of two stages: the offline training stage and the runtime clothing synthesis stage. To model the nonlinear deformation of clothing mesh under different poses and shapes, we precompute a pose training set of clothing meshes with a single template body in multiple poses and develop a shape training set fit to different body shapes in each pose. We use the pose training data to train a clothing pose model to predict the pose-dependent clothing deformation, given a template body (e.g. SOR); for each pose in the shape training data, we train a clothing shape model to predict the shape-dependent clothing deformation with a fixed pose. At run-time, the method separates deformations induced by pose and shape (i.e. a clothing pose model and a clothing shape model) through Taylor expansion. Given a query body pose and shape, we first find nearby clothing pose models and clothing shape models and then blend the Taylor expansion result of each nearby anchoring point to synthesize an intermediate mesh. We then resolve penetrations and add damping to get the final clothing mesh for the input body.

Algorithm 1 Pseudo-algorithm of our method

The offline stage
Step 1: Classifying the pose date-set into k clusters.
Step 2:
1: **for** each cluster center **do**
2: simulate clothing for different body shapes;
3: train a clothing shape model;
4: **end for**
Step 3: train an external clothing pose model.

The online stage
Input: body shape and pose parameters (β, θ).
Output: the clothing deformation for (β, θ).
Step 1: compute the clothing deformation for input pose θ for the clothing pose model.
Step 2: find several nearby anchoring points for the input (β, θ).
Step 3:
1: **for** each nearby anchoring point **do**
2: compute the clothing deformation for input shape β for the corresponding clothing shape model;
3: compute the Taylor expansion result of the anchoring point;
4: **end for**
Step 4: blend the Taylor expansion results for nearby anchoring points.

Specifically, we represent the input body with shape parameters β and pose parameters θ. The clothing deformation Y can be formulated as a high-dimensional function $Y = f(\beta, \theta)$ and approximated by the clothing on its nearby example body with shape parameters β_0 and pose parameters θ_0 (we refer to (β_0, θ_0) as the anchoring point) using first-order Taylor expansion:

$$f(\beta, \theta) = f(\beta_0, \theta_0) + \Delta\beta f_\beta(\beta_0, \theta_0) + \Delta\theta f_\theta(\beta_0, \theta_0), \tag{1}$$

where $\Delta\beta$ and $\Delta\theta$ are the shape and pose difference, respectively, between input body and its nearby example body.

We use forward difference to calculate the partial derivatives $f_\beta(\beta_0, \theta_0)$ and $f_\theta(\beta_0, \theta_0)$ in Eq. 1 as follows:

$$\begin{aligned} f(\beta, \theta) &= f(\beta_0, \theta_0) + \Delta\beta \cdot \frac{f(\beta, \theta_0) - f(\beta_0, \theta_0)}{\Delta\beta} \\ &\quad + \Delta\theta \cdot \frac{f(\beta_0, \theta) - f(\beta_0, \theta_0)}{\Delta\theta} \\ &= f(\beta_0, \theta) + (f(\beta, \theta_0) - f(\beta_0, \theta_0)), \end{aligned} \tag{2}$$

where $f(\beta_0, \theta)$ represents the clothing under a new pose with the body shape unchanged, which can be computed via a *clothing pose model*. $f(\beta, \theta_0)$ denotes the clothing for a new body shape with the pose fixed, which can be calculated through a *clothing shape model*. Correspondingly, $(f(\beta, \theta_0) - f(\beta_0, \theta_0))$ represents the shape-dependent cloth deformation, which is the cloth mesh deformation during the body shape change from β_0 to β under the anchoring pose θ_0.

In this way, the cloth deformations induced by body shape and pose can be separately computed and then combined. However, such an

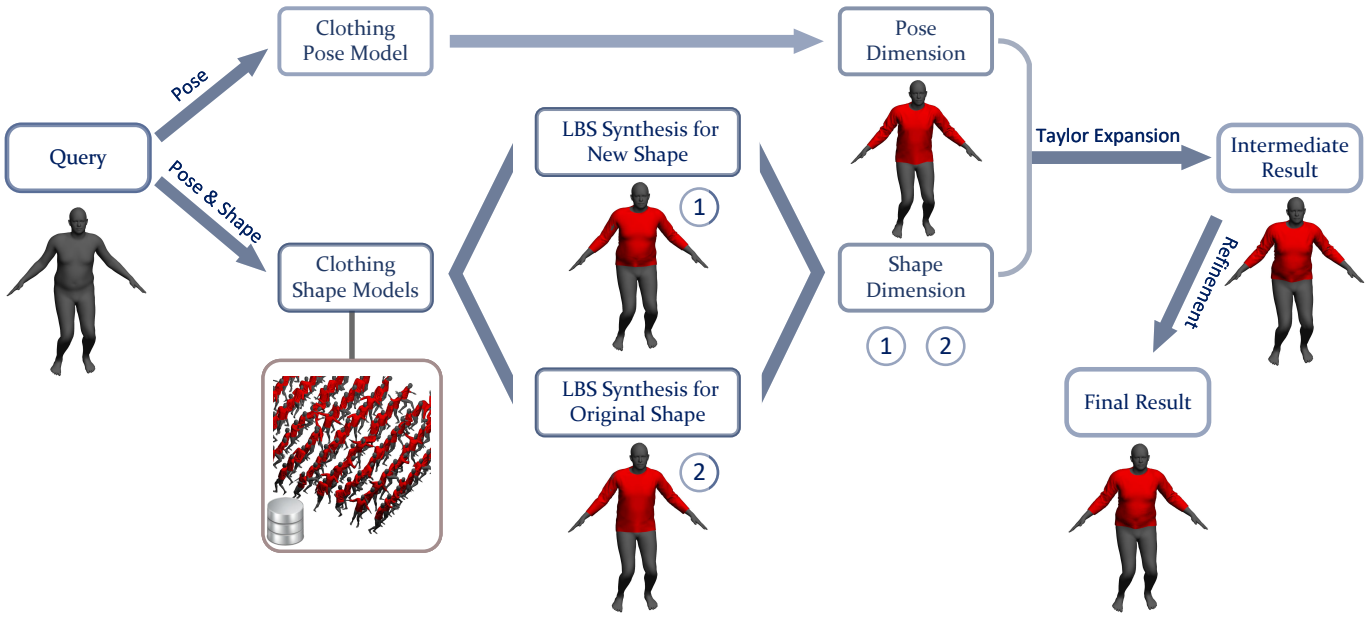


Fig. 2. Overview of our runtime clothing synthesis workflow. Given a query pose and shape, we first calculate the LBS synthesis results for the new shape and the original shape, and we refer to these as the shape dimension. The "pose dimension" is obtained from an external clothing pose model with the query pose and the original shape. Then we blend the "shape dimension" and the "pose dimension" using Taylor expansion to synthesize an intermediate result. Finally, we apply some refinement techniques such as penetration resolution to generate the final result. Sec.3.6 details the construction of clothing shape models.

approximation cannot accurately obtain the clothing deformation under (β,θ) since the term of shape-dependent cloth deformation in Eq. 2 should be measured under the new pose θ as $(f(\beta,\theta)-f(\beta_0,\theta))$, rather than under the anchoring pose θ_0. To improve the approximation accuracy, we apply the Linear Blend Skinning (LBS) method to predict the cloth deformation for the new pose θ from the cloth mesh under its nearby sample pose θ_0. Therefore, our method requires that the garment is not too loose on the avatar. The shape-and-pose dependent cloth mesh deformation $f(\beta,\theta)$ can be formulated as the Augmented Taylor Expansion:

$$f(\beta,\theta)=f(\beta_0,\theta)+\left(LBS^{\theta}_{\theta_0}(f(\beta,\theta_0))-LBS^{\theta}_{\theta_0}(f(\beta_0,\theta_0))\right), \quad (3)$$

where $LBS^{\theta}_{\theta_0}$ stands for the linear blend skinning from pose θ_0 to θ.

Since Taylor expansion can only predict the function value at points that are not far away from the expansion point, we apply Taylor expansion at multiple points, and then blend these approximation results.

Note that both training and running a clothing pose model are time-consuming in practice. Taking SOR as an example, it takes about 30 hours to construct the necessary database and more than 10ms to predict the clothing deformation under a new pose, which is too time-consuming in our scenario since we need to run the clothing pose model once at each expansion point. To address this problem, we generate expansion points that have the same β value, such as (β_0,θ_1), (β_0,θ_2), $(\beta_0,\theta_3)\cdots$, etc. In this way, all expansion points share one clothing pose model $f(\beta_0,\theta)$ trained at β_0, referred to as the original shape. In our implementation, the original shape is the medium stature body obtained by setting the shape parameters of the SMPL body model [41] to be all zeros.

At run-time, given a query pose and shape, our method first finds the nearby clothing shape models through a sensitivity-based distance measure. For each nearby anchoring point, we compute the approximation result for the query pose and shape according to Eq. 3. After that, we blend the approximation results with blending weights computed from the distance measure. In summary, our model predicts the clothing under any given body shape parameter β and pose parameter θ by

$$f(\beta,\theta)=f(\beta_0,\theta)+\sum_{s=1}^{N_s}w^s LBS^{\theta}_{\theta_s}(f(\beta,\theta_s))-\sum_{s=1}^{N_s}w^s LBS^{\theta}_{\theta_s}(f(\beta_0,\theta_s)), \quad (4)$$

where N_s is the number of clothing shape models and w^s is the blending weight of the s-th data point. Figure 2 illustrates the computation of this equation where the three items in the right-hand side of Eq. 4 are referred to as the *pose dimension*, the *LBS synthesis for new shape*, and the *LBS synthesis for original shape*. Details will be given in the following sections. Please see the Appendix for the validation of separability of pose-dependent deformation and shape-dependent deformation.

3.2 Clothing Shape Model

The clothing shape model captures clothing deformation induced by the body shape. The model is learned from the clothing shape examples that are simulated under various body shapes with a fixed pose. In our database construction stage, we use the following procedure for all anchoring poses. Each of them is used to generate a clothing shape model for a specific pose. We use a SMPL parametric human model to represent the variations in human body shape. For each of the first four principal components of the body shape parameters β in the SMPL model, we generate 4 body shapes ($\beta_k=-2,-1,1,2$) while keeping the remaining parameters in β as 0. Besides these 16 body shapes, we add the nominal shape with $\beta=0$. As a result, we generate 17 different body shapes.

For each generated body shape, we perform a simulation for one-second to drape the clothing on the avatar. All the simulations are generated using a clothing model that combines the StVK membrane model [68] and the isometric bending model [6] to simulate a garment. During these simulations, the connectivity and topology of the garment mesh do not change for the body shapes.

For the k-th generated clothing instances, we concatenate the clothing coordinates of all vertices into a single column vector $\vec{c^k}$. Next, we collect all 17 clothing samples into a matrix $S=[\vec{c^1},...,\vec{c^k},...,\vec{c^{17}}]$. We use Principal component analysis (PCA) to compute a low-dimensional subspace so that $\vec{c^k}$ can be approximated by the following equation:

$$\vec{c^k}=U\vec{\phi}^k+\vec{u}, \quad (5)$$

where $\vec{u}$ is the mean coordinates of clothing meshes and $\vec{\phi}^k$ is the clothing shape coefficients to represent the clothing shape $\vec{c^k}$. The

Fig. 3. Clothing shape model. Deviations from the mean shape: (a) the average coordinates of the training data, referred to as the mean shape; (b-d) mean shape deformed along the first three principal component directions ($\pm$ 3 standard deviations). Note that we apply PCA on the coordinates of cloth vertices, the global displacement in the vertical direction of (b) is introduced by the body height.

matrix U represents the first few principal components of the shape deformation space. As shown in Figure 6, the variance converges around 5 principal components, which corresponds 98.6% of the total variance. Moreover, we find that the remaining principal components have little effect on the final result. Therefore, we use the first 5 principal components in our clothing shape model to balance efficiency and accuracy. Figure 3 illustrates the mean and first three principal components for a men's long-sleeved shirt.

Given 17 body and clothing training pairs, we learn a linear mapping, W, between body shape parameters and clothing shape parameters using L2-regularized least squares with the weight of the regularized term being 0.2. For an input body shape β, the corresponding clothing shape coefficients $\vec{\phi}$ can be predicted as:

$$\vec{\phi} = W \cdot \left(\beta, \beta^2, 1\right)^T. \tag{6}$$

The clothing shape deformation under the body shape β can be predicted based on $\vec{\phi}$ shown in Eq. 5. In practice, we observe that there might be some small inter-penetrations between the predicted clothing mesh and the body mesh. We resolve this issue by pushing the implicated clothing vertex in the direction of the normal of its closest body vertex until there are no penetrations.

3.3 Clothing Pose Model

The clothing pose model captures clothing deformation introduced only by pose changes. Many excellent clothing pose models have been presented over the last decade [30, 36, 75]. Our animation scheme does not limit the type of clothing pose model; in extreme cases, we apply a simulated sequence as the clothing pose model (see Sec. 4.4). For real-time virtual try-on applications, we adopt the sensitivity-optimized rigging method proposed in [75] since it has a good balance of accuracy and speed.

3.4 Runtime Synthesis

Given an input body with shape parameters β and pose parameters θ, our method first finds nearby example poses through a sensitivity-based distance measure and then approximates the clothing deformation using Taylor expansion, as shown in Eq. 4. Specifically, the first term $f(\beta_0, \theta)$ computes the clothing under the input pose with the original body shape, which can be predicted through the clothing pose model described in Sec. 3.3.

The computation of the second term $\sum_{s=1}^{N_s} w^s LBS_{\theta_s}^{\theta}(f(\beta, \theta_s))$ consists of three steps: predicting new clothing instances $f(\beta, \theta_s)$ for each clothing shape model, applying LBS to $f(\beta, \theta_s)$, and blending the LBS results.

For each clothing shape model, we first calculate the clothing shape coefficients $\vec{\phi}$ through Eq. 6 using the input shape β. Then we compute the new clothing instances $f(\beta, \theta_s)$ by the right side of Eq. 5.

To apply LBS to new clothing instances $f(\beta, \theta_s)$, we first find the closest body vertex for each clothing vertex and set the bone weights, w_b^s, of a clothing vertex to that of its closest body vertex. We refer to this step as the *binding information updating step*. Then we deform each nearby clothing mesh $f(\beta, \theta_s)$ towards the query pose as $\bar{y}^s = \sum_{b=1}^{N_b} w_b^s \left(R_b^{\theta_s,\theta} y^s + T_b^{\theta_s,\theta}\right)$, where $R_b^{\theta_s,\theta}$ and $T_b^{\theta_s,\theta}$ are the relative rotation and translation of bone b from example pose θ_s to input pose θ, respectively, and w_b^s is the bone weight defined on the clothing vertex y^s of $f(\beta, \theta_s)$. We denote this equation as $LBS_{\theta_s}^{\theta}(f(\beta, \theta_s))$, as shown in Eq. 4.

We use a sensitivity-based distance measurement both to find nearby clothing shape models and to compute the blending weights w^s. To reduce the required clothing shape models in the database, we divide the clothing mesh into several regions, as in [75]. To this end, we manually partition the bones into $N_g = 7$ regions (shown on the top left of Figure 4). A region weight $w_{g,y}$ for a clothing vertex y is computed by summing the bone weights w_b^0 for the bones of the current region. w_b^0 is computed in the T-pose $s = 0$ for β through the *binding information updating step*. In this way, our method synthesizes the result for each region separately.

For each region g, we compute the sensitivity-based distance $D_g^s(\theta)$ between the input pose θ and the pose of the s-th data point as the weighted sum of differences of joint angles:

$$D_g^s(\theta) = \sum_{y \in Y} w_{g,y} \sum_{m=1}^{3N_L} ||\bar{\delta}_{y,m} \cdot \Theta_m(\theta, \theta_s)||^2 = \sum_{m=1}^{3N_L} Q_{g,m} ||\Theta_m(\theta, \theta_s)||^2, \tag{7}$$

where superscript and subscript s correspond to the values for the s-th clothing shape model, N_L is the number of joints, and $\Theta_m(\theta, \theta_s)$ calculates the m-th joint angle difference. $\bar{\delta}_{y,m} = \frac{1}{17} \sum_{k=1}^{17} ||\delta_{y,m}^k||$ is the average sensitivity of 17 training shapes, where $s_{y,m}^k$ indicates the three-dimensional coordinate differences of a clothing vertex y under a small joint rotation of m-th joint angle, calculated under T-pose and the k-th training shape. $Q_{g,m} = \sum_{y \in Y} w_{g,y} ||\bar{\delta}_{y,m}||^2$ reflects the influence of the m-th joint angle on region g. $\bar{\delta}_{y,m}$ indicates the influence of the m-th joint angle on clothing vertex y, which is computed in the database construction stage. Each time we change the body shape, we update $Q_{g,m}$ for each region once according to the new region weights of clothing vertices. At run-time, we can compute the sensitivity-based distance $D_g^s(\theta)$ efficiently since we only have to calculate the joint angle differences $\Theta_m(\theta, \theta_s)$.

Given the distance $D_g^s(\theta)$, the weight for the region is calculated as $W_g^s(\theta) = 1/(D_g^s(\theta) + \varepsilon)^k$, where ε is a small number in case of zero division and k regulates the influence of closer examples. A small k tends to smooth the animation and lose fine wrinkles, while a large k tends to preserve fine details but results in discontinuity. In our implementation, we set $k = 3$. In practice, the first five nearest clothing shape models are considered as nearby and used in synthesizing the final result, i.e. we set $W_g^s(\theta) = 0$ except for the top five largest ones.

Finally, the blending weight for each clothing vertex in example s is calculated as:

$$w^s = \sum_{g=1}^{N_G} \left(w_g W_g^s(\theta) / \sum_{s=1}^{N_s} W_g^s(\theta)\right). \tag{8}$$

The computation pipeline of the third term $\sum_{s=1}^{N_s} w^s LBS_{\theta_s}^{\theta}(f(\beta_0, \theta_s))$ is basically the same as the second term except for the body shape. In other words, we first compute clothing instances under the original shape β_0 for each clothing shape model, denoted as $f(\beta_0, \theta_s)$. Then we blend the LBS results of $f(\beta_0, \theta_s)$ using the same weights as in the second term. Note that $f(\beta_0, \theta_s)$ and their binding information are computed only once for an application.

3.5 Refinement

3.5.1 Decaying Effects

In practice, we find our clothing synthesis result may experience sudden changes for sudden input pose changes. Similar to [75], we prevent

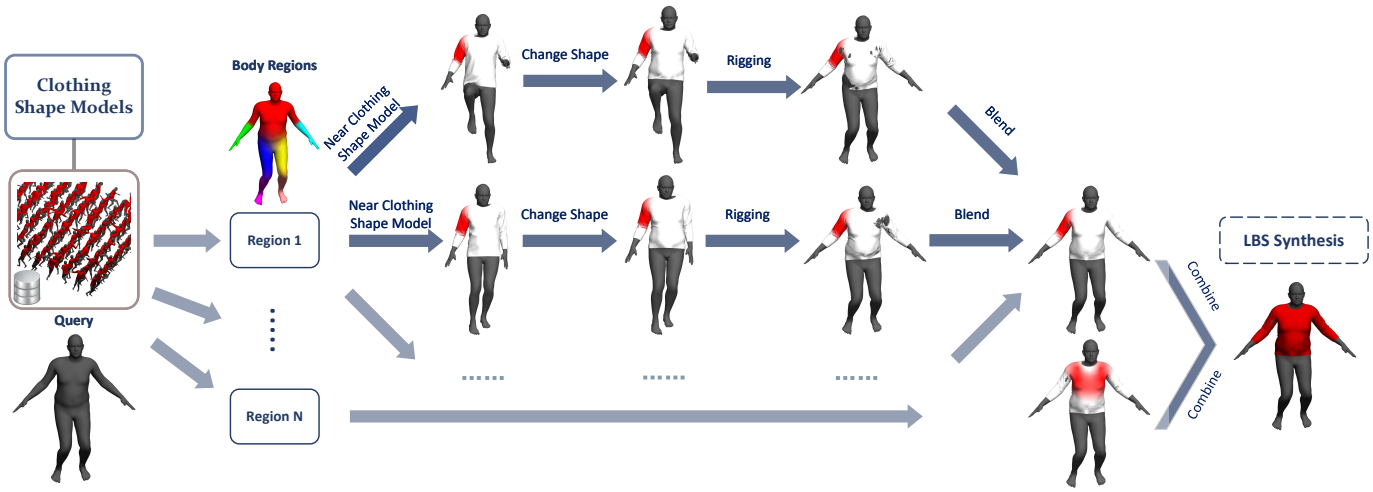


Fig. 4. The LBS synthesizing process. Given a query pose and shape, we first choose nearby clothing shape models. For each chosen clothing shape model, we change its shape to the query shape (the original shape), and deform the clothing mesh to the query pose using a linear blend skinning. Finally, we blend these deformations to get what we call the *LBS synthesis for the query shape (the original shape).*

Fig. 5. The result before (left) and after (right) our penetration handling process.

such a problem by blending the distance at the current time step D_g^s with that of the previous time step $D_g'^s$:

$$D_g^s = \eta D_g'^s + (1-\eta) D_g^s, \tag{9}$$

where η is the damping ratio, ranging from 0 to 1. To determine the damping ratio, we first calculate the relationship between the damping ratio and the norm of residual, computed under the original body shape. Then we choose the maximum value of the damping ratio while the norm of the residual is still low (please see the appendix for more details). The rationale is that the higher the value of the damping ratio, the stronger the decaying effect and the less flickering. We believe this simple technique will introduce some hysteresis while maintaining a natural result.

3.5.2 Penetration Handling

After we get the synthesized result through Eq. 4, there might be interpenetrations between the clothing mesh and the body mesh (as shown in Figure 5). Like SOR [75], we use a re-projection technique with three steps to resolve this problem.

First, every time we change the shape of the avatar, for each clothing shape model we re-calculate the initial distance of a clothing vertex from its closest body vertex. We refer to this as the initial clearance. Second, in the run-time stage, if the clearance between a clothing vertex and its closest body vertex is less than the initial clearance, for each nearby clothing shape model, we re-project the clothing vertex towards the direction of the normal of its closest body vertex as:

$$\begin{cases} \hat{y}^s = y + d \cdot \vec{n} \\ d^s = max(0, \quad d_0^s - h^s), \end{cases} \tag{10}$$

where y is a clothing vertex synthesized by Eq. 4, $\vec{n}$ is the normal of the closest body vertex of y, and h^s is current clearance. We set $d_0^s = min\{h_0^s, \varepsilon_p\}$, where h_0^s is the initial clearance, and ε_p is to mimic the penetration depth margin in cloth simulation (in our implementation, we empirically set $\varepsilon_p = 5mm$). Finally, we blend the re-projection results of all nearby examples as $\bar{y} = \sum_{s=1}^{N_s} w^s \hat{y}^s$.

3.6 Database Construction

Our database corresponds to clothing shape models corresponding to various poses. To generate our example database, we first select 32 motion sequences from the CMU motion capture library [14] and sample a pose for every four frames. In total, we obtain 17k different poses representing the whole pose space. Then we use weighted K-means to classify these poses into a certain number of clusters, which will be used to generate our example database of the clothing shape model. In our implementation, it takes about one hour to classify these poses into a typical value of 150 clusters.

The weight of a joint should reflect its importance in clothing animation. For instance, while the rotation of the knee joint has little, if any, influence on the animation result of a T-shirt, it plays a crucial role in the deformation of pants. To this end, we use the sum of the norm of the sensitivity of a joint, s^L, as its weight in the clustering process, calculated as

$$s^L = \sum_{y \in Y} \sum_{m=1}^{3} ||\bar{\delta}_{y,m}||, \tag{11}$$

where m represents the degree of freedom of joint L and $\bar{\delta}_{y,m}$ is the same as in Eq. 7 (i.e. s^L is computed over all the training shapes).

The cluster centers are essentially the anchoring points for Taylor expansion in Eq. 4. For each anchoring point, we generate a clothing shape model, which we elaborate on in Sec. 3.2.

4 Experiments

We implemented our approach in C++ and reported its performance on an off-the-shelf computer with an Intel Core i7-7700K CPU 4.20GHz and 16GB memory. The same material parameters are applied for all types of clothing: bending stiffness is $10^{-5}N/m$, stretching stiffness is $30N/m$, area density is $0.1kg/m^2$, and dynamics and static coefficient of friction is 0.3.

4.1 Database Construction and Run-time Performance

We create databases for three clothing models: a long-sleeved shirt and pants for a male body and a T-shirt for a female body (see Figure 1). For each clothing model, we first calculate its joint weights using Eq. 11; then we use weighted K-means to obtain anchoring points; finally, we generate a clothing shape model for each anchoring point. Based on the independence of the anchoring points, our data points can be generated

Clothing	Long-sleeved shirt	T-shirt	Pants
number of vertices	12.2k	12.0k	11.7k
number of triangles	23.4k	23.8k	22.4k
number of data points	150 / 150	150 / 150	150 / 170
database size (MB)	208.5 / 56.7	206.3 / 56.0	199.5 / 61.7
construction time (hrs)	15 / 7	10 / 6	14 / 8
runtime frame rate (FPS)	56	55	71

Table 1. Statistics for three different clothing databases. The number to the left of "/" is the data for our method while the number to the right of "/" indicates the data for the external clothing pose model we use, i.e. SOR. Note that we have discarded the translation item in SOR and replaced the sampling method with ours. These measures contribute to the high speed of the database generation process of SOR.

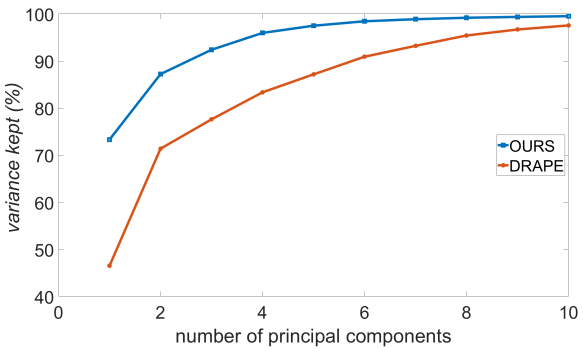


Fig. 6. Convergence of variance during PCA of the training data (left most column in Figure 2 in Appendix). Our method applies PCA to coordinates of clothing vertices while DRAPE applies PCA to deformation gradients of clothing triangles. We can see that our method converges faster and captures more variance with a given number of principal components.

in parallel. Take T-shirt as an example, we generate $150 \times 17 + 150$ (the data used for our method + the data for SOR) clothing instances to generate our database. On the other hand, [54] simulated $7,117 \times 17$ clothing instances. That is, the size of their database is more than 40 times larger than ours. We also compare the performance with TailorNet [49]. For a typical clothing deformation, TailorNet generates about 20 training style-shape pairs, each with $2,600$ training poses. As a result, TailorNet generate $20 * 2,600 = 52,000$ clothing instances, and the overall dataset size is about 19 times larger than ours. Moreover, it takes 20 hours to train their model on a single GeForce RTX 2080 GPU. On the contrary, our approach takes only 3 minutes on an Intel Core I7-7700K CPU 4.20GHZ to train our clothing shape models (i.e., about 400 times faster). At runtime, for a men's shirt with 7.9k vertices and 19k triangles, their takes about 80 ms per frame on the CPU, which is about 4 times slower than our method. Overall, our approach can provide realtime performance using a CPU for VR applications. Table 1 shows the details of databases constructed for these clothing models.

4.2 Improvement on SOR by applying our sampling method

Each step of Markov Chain Monte Carlo (MCMC) sampling in SOR aims to find the pose that maximizes the norm of the residuals of the synthesized clothing, i.e. they try to minimize the maximum error. However, a pose with the biggest error is sometime not a natural human pose. On the other hand, our sampling method can generate poses that are representative of real life. Therefore, we believe our sampling method is better than the one used in SOR. To demonstrate this, we replace the sampling scheme in SOR with our sampling method and regenerate the database for SOR. The weights of joints in the clustering process are computed with the sensitivity under the original body shape from SOR. As shown in Table 2, our method reduces the norm of the residual force by over 22%, as computed over the 32 motion sequences described in Sec. 4.1. Furthermore, the sampling points generated by our sampling method are independent of each other, which enables the database to be generated in parallel.

4.3 Clothing Shape Model

Many promising learning-based or neural methods can also offer realtime cloth synthesis performance and generate interesting effects [38, 49, 54]. However, they require a desktop GPU for runtime computations and the fidelity can vary based on human poses and movements. Therefore, we compare our clothing shape model to that of DRAPE [27], a state-of-the-art realtime CPU-based method that has low computational overhead. Instead of using the SCAPE body [1], we map the first four shape parameters of SMPL [41] to their clothing shape parameters and use our training data to train this mapping function. We take a men's long-sleeved shirt as the representative clothing type for quantitative experiments of clothing shape models. The results are similar for other clothing types. To better evaluate the performance of the clothing shape model, we use the average Euclidean vertex distance to measure the prediction error and compute the average prediction error for 100 randomly generated body shapes.

Figure 7 (left) illustrates the average errors for 20 different clothing shape models (20 different poses). Compared to DRAPE, our method reduces the error for each clothing shape model by 60%. This can also be seen in the prediction results of the second clothing shape model for a random body shape; please see the right part of Figure 7. As shown in Figure 6, our method captures more variance than DRAPE given the same number of principal components (98.6% vs. 87.2% when using the first 5 principal components), which contributes to the higher performance of our clothing shape model. It is also worth noting that the square term β^2 reduces the prediction error by 3%.

Furthermore, our mapping function from body shape to clothing deformation only involves multiplication and addition, and thus is more efficient to calculate than DRAPE, where an optimization problem needs to be solved. In practice, it takes 22ms for our clothing shape model to predict the new clothing information, compared to 300ms for DRAPE.

4.4 Animation Results

We use SMPL [41] to generate template body meshes under different shapes and apply dual quaternion blending [34] to animate the body mesh. The skinning weights are calculated using Pinnochio [4] and the motion sequences are from the CMU motion capture library [14].

At run-time, given a new input pose, we employ Eq. 4 to obtain a coarse result and then add the decaying effect and resolve interpenetrations to get the final result. Given a new input shape, we re-compute the new clothing instance $f(\beta, \theta_s)$ and update the binding information for each clothing shape model (See Sec. 3.4), which takes 0.022s each time. In our implementation, all clothing shape models can be handled in parallel, and it takes 0.7s to handle 150 data points for the men's long-sleeved shirt.

Theoretically, our algorithm can add the shape dimension to any clothing pose model. To demonstrate this, we run our method with both SOR and a sequence of simulated clothing instances. First, we use SOR as our clothing pose model. We turn off the penetration handling process in SOR and leave it for our method to address. We also discard the translation item (rightmost item in Eq. 1 of [75]) in SOR since we find it has little impact on the final result, especially when using our sampling method. These measures contribute to the high speed of our method. We use our sampling method to generate both the SOR database and our database. As shown in Figure 1, our method can predict clothing instances with fine wrinkles. Please refer to Table 1 for detailed statistics.

Second, we use simulated clothing instances as our clothing pose model. We simulate the garment mesh under a randomly chosen motion sequence while keeping the same body shape parameters. Then we use the resulting clothing instances as the clothing pose model in Eq. 4. Figure 8(b) shows that our method can recover realistic fold patterns. We can see that the quality of our result relies on the external clothing pose model, i.e. if the clothing pose model is more accurate, our result is closer to the ground truth (Figure 8(c)). This can also be seen in Figure 9. We will elaborate on this in the following section.

n_{dp}	1	10	20	30	40	50	60	70	80	90	100	110	120	130	140	150
SOR	45.6	28.0	25.7	28.1	29.0	29.8	28.9	26.0	25.8	25.2	23.7	23.3	23.1	22.6	22.3	22.2
Ours	27.6	20.5	20.0	18.4	16.5	18.7	18.2	17.3	17.4	18.9	17.6	17.2	16.4	16.9	17.4	17.2
Imp	39%	27%	22%	34%	43%	37%	37%	33%	33%	25%	26%	26%	29%	25%	22%	22%

Table 2. Quantitative comparison between the original SOR and our method. n_{dp} stands for the number of data points and "Imp" is an abbreviation of "improvement". We can see that our sample scheme can reduce the norm of the residual by more than 22%, meaning that the synthesized clothing deformations of our method are closer to the equilibrium states.

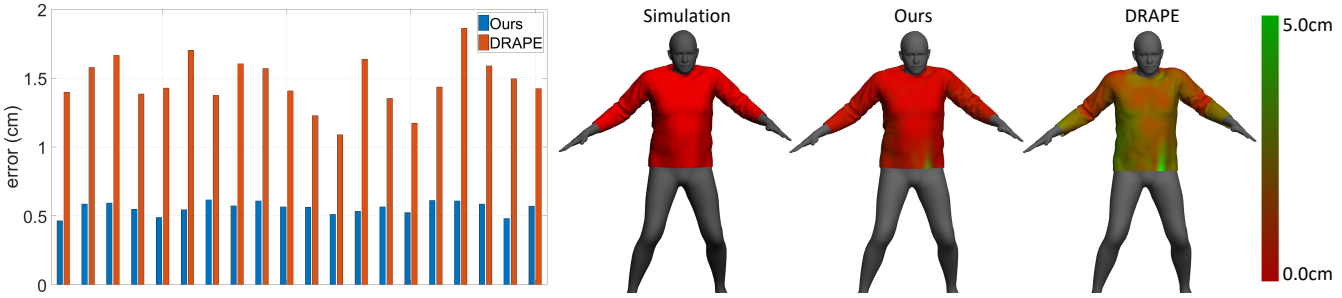


Fig. 7. Comparison of our clothing shape model and that of DRAPE. The bar graph on the left shows the errors for 20 clothing shape models. The prediction results for the second clothing shape model for a random body shape are shown on the right. The errors are marked in green.

4.5 Convergence

We use the average Euclidean vertex distance between the synthesized clothing and the simulation clothing as the error to measure the physical accuracy of our result. The error is calculated over 16×400 randomly chosen shape and pose pairs (16 shapes and 400 poses for each shape). As shown in Figure 9, the error of the *results with SOR* drops to 1.06 cm at 150 data points, which we believe is acceptable in most scenarios. Given the number of data points, the error of the *results with simulation* is much less than that of *results with SOR*, which demonstrates that our method can be improved if a more accurate clothing pose model is employed.

4.6 Ablation Study

The first-order module (see Eq. 2) plays a key role in predicting the clothing deformation for the input pose and shape based on the anchoring point. Without this module, we can only simulate a clothing deformation at each anchoring point without the prediction capability. Therefore, our approach can only synthesize clothing deformations at anchoring points. In this case, the synthesis equation reduces to $f(\beta,\theta) = \sum_{s=1}^{N_s} w^s f(\beta_0,\theta_s)$, where the symbols are the same as the ones in Eq. 4. According to Sec. 3.1, all the anchoring points would share the same β values. Given two anchoring points with different poses, a simple blending of the two clothing deformations could lead to an unrealistic deformation. Therefore, we apply LBS to $f(\beta_0,\theta_s)$ to improve the deformation prediction for the input pose. Our final equation for the ablation study corresponds to $f(\beta,\theta) = \sum_{s=1}^{N_s} w^s LBS_{\theta_s}^{\theta}(f(\beta_0,\theta_s))$. In practice, this formulation results in poor synthesis results (see Fig. 10).

4.7 VR Scenarios

Our method can be applied to VR scenarios. As shown in Figure 11, we provide the user with an immersive VR experience from a first-person perspective with an HTC Vive (left most). The clothing deformations for the agents are generated by our method.

4.8 User Evaluation

We conducted a user study to demonstrate the perceptual benefit of our method compared to the prior technique in generating clothing deformations for the agent in immersive settings.

Experiment Goals & Expectations: We hypothesize that the clothing deformations generated by our method will exhibit more detailed wrinkles and more dynamics compared to prior methods and that participants will strongly prefer our results to those of prior methods.

Experimental Design: The study was conducted based on a within-subjects, paired-comparison design. Participants explored two simulations with a fixed exposure time wearing a HTC Vive headset. The clothing deformations in the two simulations were generated using our method and the LBS method. The order of scenes was counterbalanced, as well as the order of the methods. After these two simulations, participants answered a set of questions, and our questionnaire design is inspired by prior methods [22, 47, 56].

Comparison Methods: Previous methods generally adopt skinning methods to deform the clothing mesh of an agent in virtual environments. Therefore, we evaluated our method against the Linear Blend Skinning (LBS) method.

Environments: We use three scenarios for the user study. The first scenario was comprised of a man sweeping in the living room, while wearing a long-sleeved shirt and pants. The second scenario consisted of a man that has a different shape and wiping the windows, while wearing the same clothes as the first scenario. The last scenario corresponded to a woman wandering in a living room, while wearing a t-shirt and shorts. Participants can walk around and observe the agent doing some tasks. Please refer to Figure 11 and the supplemental video for more details.

Metrics: Participants were asked to indicate their preference for a method using a 7-point Likert scale, with 1 indicating a strong preference for the method presented first, 7 indicating a strong preference for the method presented second, and 4 indicating no preference. In terms of reporting the results, we normalized the participant responses so that 1 indicates a strong preference for our method.

Results: Our study was taken by 18 participants, 9 male, with a mean age of 24.44 ± 1.95 years. The participant responses clearly demonstrate the benefits of our algorithm. For each question, we performed a one-sample t-test comparing the mean of the question with a hypothetical mean of 4 (no preference or no impact). The question "Which clothing animation looks more realistic?" was shown to be significant for each scenario: $t(17)$ = -6.336, $p < 0.001$; $t(17)$ = -8.166, $p < 0.001$; $t(17)$ = -7.114, $p < 0.001$. The question "Which wrinkles look more natural?" was also significant for each scenario: $t(17)$ = -6.761, $p < 0.001$; $t(17)$ = -7.432, $p < 0.001$; $t(17)$ = -6.101, $p < 0.001$, as was "Which agent looks more like a real person?": $t(17)$ = -4.579, $p < 0.001$; $t(17)$ = -6.216, $p < 0.001$; $t(17)$ = -5.532, $p < 0.001$;. Figure 12 and Table 3 provide further details on participant responses.

5 Discussion

The rationale of our sampling method is that Taylor expansion locally approximates surrounding values of an anchoring point while cluster

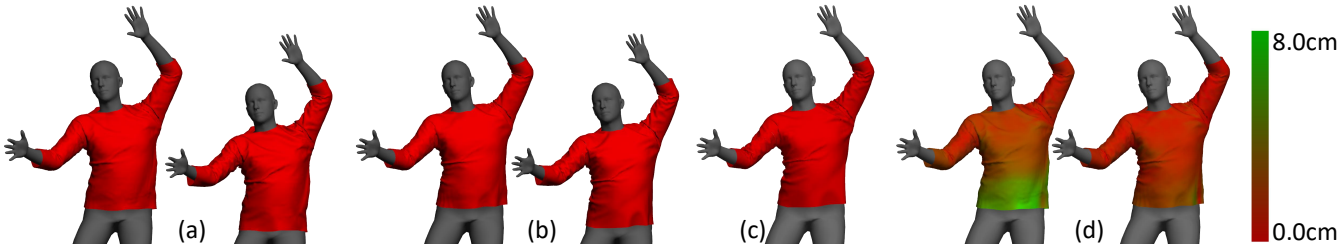


Fig. 8. Comparison between the synthesis results achieved using SOR as the clothing pose model and those achieved using simulation as the clothing pose model. (a) Synthesis result (left) using SOR (right); (b) synthesis result (left) using simulation (right); (c) simulation result; (d) Euclidean vertex distance between the synthesis results and the simulation clothing. The errors are marked in green. We can see that the synthesis result achieved using simulation (right) has fewer errors than the result achieved using SOR (left).

Question (Which...)	1	2	3	4	5	6	7	mean	SD
clothing animation looks more realistic?	4 / 4 / 4	5 / 6 / 9	6 / 7 / 3	3 / 1 / 1	0 / 0 / 1	0 / 0 / 0	0 / 0 / 0	2.44 / 2.28 / 2.22	±1.042 / ±0.895 / ±1.060
wrinkles look more natural?	6 / 4 / 1	7 / 12 / 11	1 / 1 / 4	4 / 0 / 0	0 / 0 / 2	0 / 1 /0	0 / 0 / 0	2.17 / 2.06 / 2.50	±1.150 / ±1.110 / ±1.043
agent looks more like a real person?	4 / 5 / 5	6 / 6 / 3	4 / 3 / 6	3 / 4 / 4	0 / 0 / 0	1 / 0 / 0	0 / 0 / 0	2.56 / 2.33 / 2.50	±1.338 / ±1.138 / ±1.150

Table 3. The response frequency of subjects in the study described in Sec. 4.8. Scores are normalized such that 1 indicates a strong preference for our method and 7 indicates a strong preference for the LBS method. For each scenario (separated by "/"), participants prefer our method over the prior method on several dimensions.

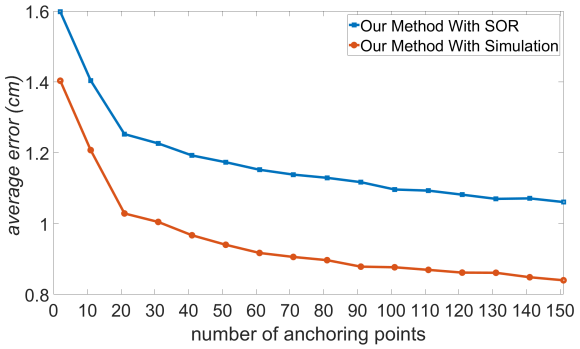
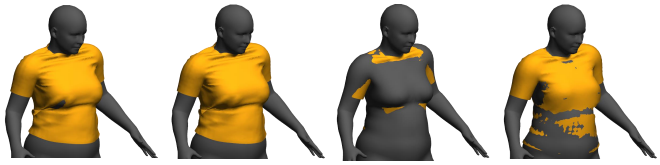


Fig. 9. Convergence of synthesis error while using an increasing number of data points.



(a) Our results with (right) and without (left) penetration handling.

(b) Ablation study results with (right) and without (left) penetration handling.

Fig. 10. Ablation study. Without the first-order module, the synthesis equation does not result in good deformation prediction results for new body shapes and could result in non-plausible deformations.

centers are the best set of points to approximate all the points, i.e. they are the best set of anchoring points. Compared to the MCMC sampling process used in SOR [75], our sampling process is able to sample more typical poses in real life, while their sampling step finds a pose that maximizes the error of the synthesized clothing, which might produce weird poses. Experimental results show that SOR results can be significantly improved when using our sampling method. Additionally, thanks to the mutual independence of our data points, our database can be constructed in parallel, while the MCMC process in SOR must run serially since their sampling process depends on the previously constructed data. Please refer to Sec. 4 for details.

Limitations: Our method assumes that the clothing coordinate differences introduced by body shape variation are linear with body pose, i.e. the clothing coordinate differences caused by body shape variation under the new pose can be predicted from the anchoring pose using

Fig. 11. The avatar in a VR scenario. We provide the user with an immersive VR experience from a first-person perspective with HTC Vive. The avatar looks around and observes an agent performing some tasks.

LBS. This hinders the application of our approach to garments that are loose on the avatar (See Fig. 13).

Furthermore, the final results of our method are highly dependent on the clothing pose model used. If the clothing pose model is not accurate, then our result is also not accurate. In addition, the efficiency of the clothing pose model creates a bottle-neck in our method. To overcome this difficulty, we plan to develop our own clothing pose model in the future. The wrinkles and folds generated in the animation suffer from some sudden changes when body poses change too quickly. The reason for this is that our scheme is trained on clothing instances of static equilibrium, and the estimations from different poses are inconsistent. Although we have employed a decaying factor to smooth the animation, it cannot solve this problem completely.

Finally, our method cannot guarantee that all the penetrations are resolved, especially when the clothing is too tight on the body. In this case, the LBS result for each anchoring pose may result in deep penetrations, and the blending result will make penetrations even worse, which is beyond the ability of our penetration resolving method to address. When this is the case, we believe that the clothing is not suitable for the body, suggesting that the clothing is a bad fit. In addition, when the clothing has too many folds, self-penetration might occur in our synthesis results.

Future work: We want to extend our method to other control parameters like clothing material and clothing patterns. For clothing material, for example, we first need to devise a clothing material model. Given a set of clothing material parameters, this model can predict the corresponding clothing deformation. Then, we sample some valuable poses as the Taylor expansion anchoring points. Finally, we blend local approximation results at run-time.

We currently use the average sensitivity of 17 training shapes to

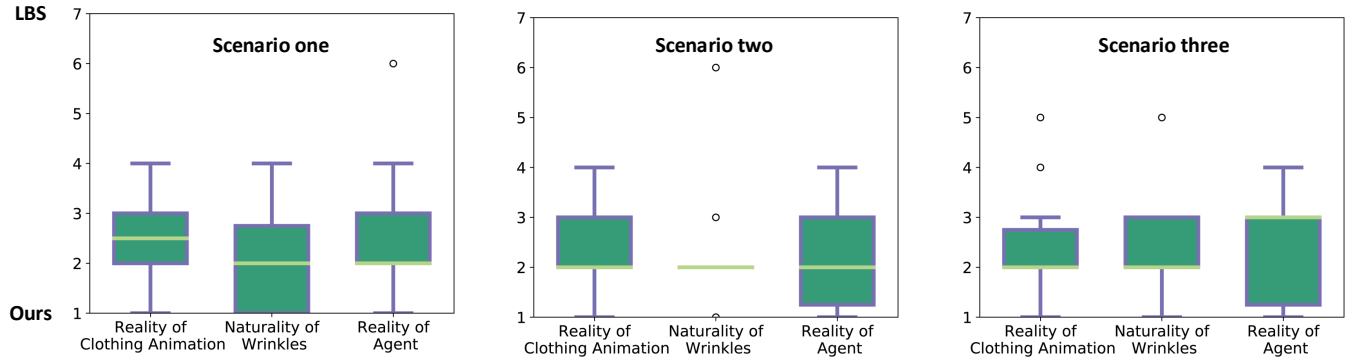


Fig. 12. Participant preferences in user evaluation. The questions were "Which clothing animation looks more realistic?", "Which wrinkles look more natural?", and "Which agent looks more like a real person?". Scores are normalized such that 1 indicates a strong preference for our method and 7 indicates a strong preference for LBS method. For each scenario, participants rated our method preferably in terms of generating more realistic or plausible clothing animation (2.44 ± 1.042, 2.28 ± 0.895, 2.22 ± 1.060), generating more natural-looking wrinkles (2.17 ± 1.150, 2.06 ± 1.110, 2.50 ± 1.043), and enhancing the presence of the agent (2.56 ± 1.338, 2.33 ± 1.138, 2.50 ± 1.150).

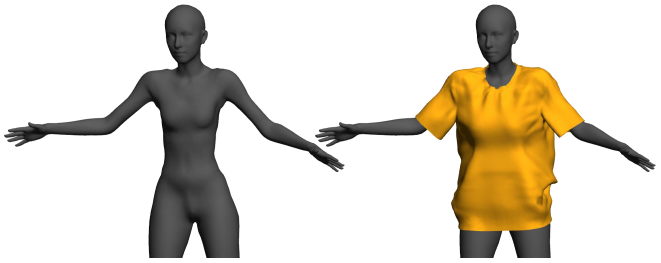


Fig. 13. Our method may not work well when the garment on the avatar is too loose. For a T-shirt model originally designed for the SMPL body with the first four parameters = [0,0,0,0], our method may produce poor results (right) for another avatar with parameters = [-2,2,0,0] (left).

calculate the distances of anchoring points. However, as the body shape changes, the sensitivity changes accordingly. In the future, we plan to train a sensitivity model, i.e. a model that can predict the corresponding sensitivity of the clothing given a new set of body shape parameters. We believe this will help us to find better anchoring points at run-time. It may also be useful to develop hybrid techniques that can combine our approach with physics-based simulation or neural methods to improve the fidelity or realism of cloth effects.

6 Conclusion

In this paper, we have presented a clothing synthesis scheme that uses Taylor expansion to combine two independent components: the pose-dependent deformation and the shape-dependent deformation. As a result, our method can add the shape dimension to various clothing pose models. Our method does not need to modify or retrain the clothing pose model. The core innovation here is that we regard clothing deformation as a function of body shape parameters and body pose parameters and use Taylor expansion to locally approximate clothing deformations around anchoring points. Due to the high computational cost of higher-order derivatives, we use linear Taylor expansion in both offline and online processes. Our clothing shape model can efficiently predict realistic clothing deformations under various body shapes and has the potential for use as a stand-alone application.

Using only a CPU, our method can generate realistic clothing deformations under various body poses and shapes in real time without resizing the cloth model. Furthermore, we believe that our method can be extended to add other parameters such as clothing material and clothing pattern to clothing pose models.

Acknowledgment

Xiaogang Jin was supported by the National Key R&D Program of China (Grant No. 2017YFB1002600), the Ningbo Major Special Projects of the "Science and Technology Innovation 2025" (Grant No. 2020Z007), the Key Research and Development Program of Zhejiang Province (Grant No. 2020C03096), and the National Natural Science Foundation of China (Grant No. 61732015)).

References

[1] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. Scape: shape completion and animation of people. In *ACM Trans. Graph.*, vol. 24, pp. 408–416. ACM, 2005.

[2] J. N. Bailenson, K. Swinth, C. Hoyt, S. Persky, A. Dimov, and J. Blascovich. The independent and interactive effects of embodied-agent appearance and behavior on self-report, cognitive, and behavioral markers of copresence in immersive virtual environments. *Presence: Teleoperators & Virtual Environments*, 14(4):379–393, 2005.

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

[4] I. Baran and J. Popović. Automatic rigging and animation of 3d characters. In *ACM Trans. Graph.*, vol. 26, p. 72. ACM, 2007.

[5] V. Barrielle and N. Stoiber. Realtime performance-driven physical simulation for facial animation. In *Computer Graphics Forum*, vol. 38, pp. 151–166. Wiley Online Library, 2019.

[6] M. Bergou, S. Mathur, M. Wardetzky, and E. Grinspun. Tracks: toward directable thin shells. *ACM Trans. Graph.*, 26(3):50–es, 2007.

[7] S. Bouaziz, S. Martin, T. Liu, L. Kavan, and M. Pauly. Projective dynamics: fusing constraint projections for fast simulation. *ACM Trans. Graph.*, 33(4):154, 2014.

[8] R. Bridson, R. Fedkiw, and J. Anderson. Robust treatment of collisions, contact and friction for cloth animation. In *ACM Trans. Graph.*, vol. 21, pp. 594–603. ACM, 2002.

[9] L. Chen, L. Gao, J. Yang, S. Xu, J. Ye, X. Zhang, and Y.-K. Lai. Deep deformation detail synthesis for thin shell models. *arXiv preprint arXiv:2102.11541*, 2021.

[10] M. Chen and K. Tang. A fully geometric approach for developable cloth deformation simulation. *The visual computer*, 26(6-8):853–863, 2010.

[11] N. Chentanez, M. Macklin, M. Müller, S. Jeschke, and T.-Y. Kim. Cloth and skin deformation with a triangle mesh based convolutional neural network. In *Computer Graphics Forum*, vol. 39, pp. 123–134. Wiley Online Library, 2020.

[12] K.-J. Choi and H.-S. Ko. Stable but responsive cloth. In *ACM SIGGRAPH 2005 Courses*, p. 1. ACM, 2005.

[13] G. Cirio, J. Lopez-Moreno, D. Miraut, and M. A. Otaduy. Yarn-level simulation of woven cloth. *ACM Trans. Graph.*, 33(6):207, 2014.

[14] CMU. CMU graphics lab motion capture database. *http://mocap.cs.cmu.edu.*, 2003.

[15] E. De Aguiar, L. Sigal, A. Treuille, and J. K. Hodgins. Stable spaces for real-time clothing. In *ACM Trans. Graph.*, vol. 29, p. 106. ACM, 2010.

[16] E. English and R. Bridson. Animating developable surfaces using nonconforming elements. In *ACM Trans. Graph.*, vol. 27, p. 66. ACM, 2008.

[17] B. Fierz, J. Spillmann, and M. Harders. Element-wise mixed implicit-explicit integration for stable dynamic simulation of deformable objects.

In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 257–266. ACM, 2011.

[18] M. Fratarcangeli, H. Wang, and Y. Yang. Parallel iterative solvers for real-time elastic deformations. In *SIGGRAPH Asia 2018 Courses*, pp. 1–45. 2018.

[19] A. Fuhrmann, G. Sobotka, and C. Groß. Distance fields for rapid collision detection in physically based modeling. In *Proceedings of GraphiCon 2003*, pp. 58–65. Citeseer, 2003.

[20] L. Gao, Y.-K. Lai, D. Liang, S.-Y. Chen, and S. Xia. Efficient and flexible deformation representation for data-driven surface modeling. *ACM Transactions on Graphics (TOG)*, 35(5):1–17, 2016.

[21] L. Gao, Y.-K. Lai, J. Yang, L.-X. Zhang, S. Xia, and L. Kobbelt. Sparse data driven mesh deformation. *IEEE Transactions on Visualization and Computer Graphics*, 27(3):2085–2100, 2021. doi: 10.1109/TVCG.2019.2941200

[22] M. Garau, M. Slater, D.-P. Pertaub, and S. Razzaque. The responses of people to virtual humans in an immersive virtual environment. *Presence: Teleoperators & Virtual Environments*, 14(1):104–116, 2005.

[23] M. Geilinger, R. Poranne, R. Desai, B. Thomaszewski, and S. Coros. Skaterbots: Optimization-based design and motion synthesis for robotic creatures with legs and wheels. *ACM Trans. Graph.*, 37(4):160, 2018.

[24] R. Goldenthal, D. Harmon, R. Fattal, M. Bercovier, and E. Grinspun. Efficient simulation of inextensible cloth. *ACM Trans. Graph.*, 26(3):49, 2007.

[25] N. K. Govindaraju, M. C. Lin, and D. Manocha. Quick-cullide: Fast inter-and intra-object collision culling using graphics hardware. In *IEEE Proceedings. VR 2005. Virtual Reality, 2005.*, pp. 59–66. IEEE, 2005.

[26] E. Grinspun, A. N. Hirani, M. Desbrun, and P. Schröder. Discrete shells. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 62–67. Eurographics Association, 2003.

[27] P. Guan, L. Reiss, D. A. Hirshberg, A. Weiss, and M. J. Black. Drape: Dressing any person. *ACM Trans. Graph.*, 31(4):35–1, 2012.

[28] E. Gundogdu, V. Constantin, A. Seifoddini, M. Dang, M. Salzmann, and P. Fua. Garnet: A two-stream network for fast and accurate 3d cloth draping. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8739–8748, 2019.

[29] Q. Guo, X. Han, C. Fu, T. Gast, R. Tamstorf, and J. Teran. A material point method for thin shells with frictional contact. *ACM Trans. Graph.*, 37(4):147, 2018.

[30] F. Hahn, B. Thomaszewski, S. Coros, R. W. Sumner, F. Cole, M. Meyer, T. DeRose, and M. Gross. Subspace clothing simulation using adaptive bases. *ACM Trans. Graph.*, 33(4):105, 2014.

[31] D. Harmon, E. Vouga, R. Tamstorf, and E. Grinspun. Robust treatment of simultaneous collisions. *ACM Trans. Graph.*, 27(3):23, 2008.

[32] M. Hauth, O. Etzmuß, and W. Straßer. Analysis of numerical methods for the simulation of deformable models. *The Visual Computer*, 19(7-8):581–600, 2003.

[33] N. Jin, Y. Zhu, Z. Geng, and R. Fedkiw. A pixel-based framework for data-driven clothing. In *Computer Graphics Forum*, vol. 39, pp. 135–144. Wiley Online Library, 2020.

[34] L. Kavan, S. Collins, J. Žára, and C. O'Sullivan. Geometric skinning with approximate dual quaternion blending. *ACM Trans. Graph.*, 27(4):105, 2008.

[35] J. Kiendl, R. Schmidt, R. Wüchner, and K.-U. Bletzinger. Isogeometric shape optimization of shells using semi-analytical sensitivity analysis and sensitivity weighting. *Computer Methods in Applied Mechanics and Engineering*, 274:148–167, 2014.

[36] D. Kim, W. Koh, R. Narain, K. Fatahalian, A. Treuille, and J. F. O'Brien. Near-exhaustive precomputation of secondary cloth effects. *ACM Trans. Graph.*, 32(4):87, 2013.

[37] J. T. Klosowski, M. Held, J. S. B. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):21–36, Jan. 1998.

[38] Z. Lahner, D. Cremers, and T. Tung. Deepwrinkles: Accurate and realistic clothing modeling. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 667–684, 2018.

[39] C. Li, M. Tang, R. Tong, M. Cai, J. Zhao, and D. Manocha. P-cloth: interactive complex cloth simulation on multi-gpu systems using dynamic matrix assembly and pipelined implicit integrators. *ACM Trans. Graph.*, 39(6):1–15, 2020.

[40] T. Liu, A. W. Bargteil, J. F. O'Brien, and L. Kavan. Fast simulation of mass-spring systems. *ACM Trans. Graph.*, 32(6):214, 2013.

[41] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. Smpl: A skinned multi-person linear model. *ACM Trans. Graph.)*, 34(6):248, 2015.

[42] G. Ma, J. Ye, J. Li, and X. Zhang. Anisotropic strain limiting for quadrilateral and triangular cloth meshes. In *Computer Graphics Forum*, vol. 35, pp. 89–99. Wiley Online Library, 2016.

[43] Q. Ma, J. Yang, A. Ranjan, S. Pujades, G. Pons-Moll, S. Tang, and M. J. Black. Learning to dress 3d people in generative clothing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6469–6478, 2020.

[44] D. Manocha. *Algebraic and numeric techniques in modeling and robotics*. University of California at Berkeley, 1992.

[45] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118, 2007.

[46] R. Narain, A. Samii, and J. F. O'Brien. Adaptive anisotropic remeshing for cloth simulation. *ACM Trans. Graph.*, 31(6):152, 2012.

[47] S. Narang, A. Best, T. Randhavane, A. Shapiro, and D. Manocha. Pedvr: Simulating gaze-based interactions between a real user and virtual crowds. In *Proceedings of the 22nd ACM conference on virtual reality software and technology*, pp. 91–100, 2016.

[48] R. Natsume, S. Saito, Z. Huang, W. Chen, C. Ma, H. Li, and S. Morishima. Siclope: Silhouette-based clothed people. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4480–4490, 2019.

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

[50] G. Pons-Moll, S. Pujades, S. Hu, and M. J. Black. Clothcap: Seamless 4d clothing capture and retargeting. *ACM Trans. Graph.*, 36(4):1–15, 2017.

[51] X. Provot et al. Deformation constraints in a mass-spring model to describe rigid cloth behaviour. In *Graphics interface*, pp. 147–147. Canadian Information Processing Society, 1995.

[52] S. Saito, Z. Huang, R. Natsume, S. Morishima, A. Kanazawa, and H. Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2304–2314, 2019.

[53] A. Saltelli, S. Tarantola, F. Campolongo, and M. Ratto. Sensitivity analysis in practice: a guide to assessing scientific models. *Chichester, England*, 2004.

[54] I. Santesteban, M. A. Otaduy, and D. Casas. Learning-based animation of clothing for virtual try-on. In *Computer Graphics Forum*, vol. 38, pp. 355–366. Wiley Online Library, 2019.

[55] Z. Shen, J. Huang, W. Chen, and H. Bao. Geometrically exact simulation of inextensible ribbon. In *Computer Graphics Forum*, vol. 34, pp. 145–154. Wiley Online Library, 2015.

[56] M. Slater, C. Guger, G. Edlinger, R. Leeb, G. Pfurtscheller, A. Antley, M. Garau, A. Brogni, and D. Friedman. Analysis of physiological responses to a social situation in an immersive virtual environment. *Presence: Teleoperators and Virtual Environments*, 15(5):553–569, 2006.

[57] M. Tang, R. Tong, R. Narain, C. Meng, and D. Manocha. A gpu-based streaming algorithm for high-resolution cloth simulation. In *Computer Graphics Forum*, vol. 32, pp. 21–30. Wiley Online Library, 2013.

[58] M. Tang, R. Tong, Z. Wang, and D. Manocha. Fast and exact continuous collision detection with bernstein sign classification. *ACM Trans. Graph.*, 33(6):1–8, 2014.

[59] M. Tang, H. Wang, L. Tang, R. Tong, and D. Manocha. Cama: Contact-aware matrix assembly with unified collision handling for gpu-based cloth simulation. In *Computer Graphics Forum*, vol. 35, pp. 511–521. Wiley Online Library, 2016.

[60] M. Tang, T. Wang, Z. Liu, R. Tong, and D. Manocha. I-cloth: incremental collision handling for gpu-based interactive cloth simulation. *ACM Trans. Graph.*, 37(6):1–10, 2018.

[61] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser, and P. Volino. Collision Detection for Deformable Objects. In *Eurographics 2004 - STARs*. Eurographics Association, 2004.

[62] J.-M. Thiery, É. Guy, and T. Boubekeur. Sphere-meshes: Shape approximation using spherical quadric error metrics. *ACM Trans. Graph.*, 32(6):178, 2013.

[63] B. Thomaszewski, S. Pabst, and W. Strasser. Continuum-based strain limiting. In *Computer Graphics Forum*, vol. 28, pp. 569–576. Wiley

Online Library, 2009.

[64] N. Umetani, D. M. Kaufman, T. Igarashi, and E. Grinspun. Sensitive couture for interactive garment modeling and editing. *ACM Trans. Graph.*, 30(4):90–1, 2011.

[65] T. Vassilev, B. Spanlang, and Y. Chrysanthou. Fast cloth animation on walking avatars. In *Computer Graphics Forum*, vol. 20, pp. 260–267. Wiley Online Library, 2001.

[66] R. Vidaurre, I. Santesteban, E. Garces, and D. Casas. Fully convolutional graph neural networks for parametric virtual try-on. In *Computer Graphics Forum*, vol. 39, pp. 145–156. Wiley Online Library, 2020.

[67] P. Volino and N. Magnenat-Thalmann. Implicit midpoint integration and adaptive damping for efficient cloth simulation. *Computer Animation and Virtual Worlds*, 16(3-4):163–175, 2005.

[68] P. Volino, N. Magnenat-Thalmann, and F. Faure. A simple approach to nonlinear tensile stiffness for accurate cloth simulation. *ACM Trans. Graph.*, 28(4):Article–No, 2009.

[69] H. Wang. A chebyshev semi-iterative approach for accelerating projective and position-based dynamics. *ACM Trans. Graph.*, 34(6):246, 2015.

[70] H. Wang, F. Hecht, R. Ramamoorthi, and J. F. O'Brien. Example-based wrinkle synthesis for clothing animation. In *ACM Trans. Graph.*, vol. 29, p. 107. ACM, 2010.

[71] H. Wang, J. O'Brien, and R. Ramamoorthi. Multi-resolution isotropic strain limiting. In *ACM Trans. Graph.*, vol. 29, p. 156. ACM, 2010.

[72] T. Y. Wang, D. Ceylan, J. Popović, and N. J. Mitra. Learning a shared shape space for multimodal garment design. *ACM Trans. Graph.*, 37(6), Dec. 2018.

[73] T. Y. Wang, T. Shao, K. Fu, and N. J. Mitra. Learning an intrinsic garment space for interactive authoring of garment animation. *ACM Transactions on Graphics (TOG)*, 38(6):1–12, 2019.

[74] N. Wu, D. Zhang, Z. Deng, and X. Jin. Variational mannequin approximation using spheres and capsules. *IEEE Access*, 6:25921–25929, 2018.

[75] W. Xu, N. Umentani, Q. Chao, J. Mao, X. Jin, and X. Tong. Sensitivity-optimized rigging for example-based real-time clothing synthesis. *ACM Trans. Graph.*, 33(4):107, 2014.

[76] J. Yang, J.-S. Franco, F. Hétroy-Wheeler, and S. Wuhrer. Analyzing clothing layer deformation statistics of 3d human motions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 237–253, 2018.

[77] S. Yang, J. Liang, and M. C. Lin. Learning-based cloth material recovery from video. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4383–4393, 2017.

[78] J. Zehnder, E. Knoop, M. Bächer, and B. Thomaszewski. Metasilicone: design and fabrication of composite silicone with desired mechanical properties. *ACM Trans. Graph.*, 36(6):240, 2017.

[79] C. Zheng and D. L. James. Energy-based self-collision culling for arbitrary mesh deformations. *ACM Trans. Graph.*, 31(4):98, 2012.

[80] S. Zimmermann, R. Poranne, J. M. Bern, and S. Coros. Puppetmaster: robotic animation of marionettes. *ACM Trans. Graph.*, 38(4):103, 2019.

Appendix: AgentDress: Realtime Clothing Synthesis for Virtual Agents using Plausible Deformations

Category: Research

Paper Type: Technology

1 Introduction

This document consists of additional information for the paper *AgentDress: Realtime Clothing Synthesis for Virtual Agents using Plausible Deformations*. Section 2 validates the local linear separability of pose-dependent deformation and shape-dependent deformation. Figure 2 presents qualitative comparison between the original SOR and our method (quantitative comparison is in Sec. 4.2 of the manuscript). Figure 3 illustrates how we choose the damping ratio described in Sec. 3.5.1 of the manuscript. Figure 4 presents more results of our method. Figure 5 illustrates part of our training data.

2 Separability

In this section, we validate the local linear separability of pose-dependent deformation and shape-dependent deformation. For an anchoring point, we uniformly sample m neighboring data points (we set $m = 100$ in our experiment), all of which are a distance r from the anchoring point. Specifically, we randomly generate $3N_L + 4$ (number of joint angles plus 4 shape parameters) numbers that are then concatenated into a vector $\vec{e}$. We normalize $\vec{e}$ in terms of 2-norm and then multiply it by r. The first $3N_L$ components of the final $\vec{e}$ are the displacement of joint angles from the anchoring point while the last 4 components are the displacement of the shape parameters. Then, for each neighboring point, we calculate the average Euclidean vertex distance between the simulation result and the synthesized result as the approximation error. The average approximation error of all neighboring points is regarded as the approximation error of the anchoring point at distance r.

We run this procedure both practically and theoretically. Practically, we apply Eq. 3 in the main document to compute the synthesized clothing; theoretically, we replace each term on the right-hand side of Eq. 3 (i.e. $f(\beta_0, \theta)$; $f(\beta, \theta_0)$; and $f(\beta_0, \theta_0)$) with the corresponding simulation result. Hence, in the practical experiment, the error comes from three sources: the clothing pose model, the clothing shape model, and the linear approximation. In the theoretical experiment, the error is caused only by the linear approximation. As shown in Figure 1, for a randomly chosen anchoring point, both results can recover the folds and wrinkles with good fidelity; the error of the theoretical experiment is smaller than that of the practical experiment given the same distance r. Meanwhile, when r is small, both approximation errors are relatively small. The errors converge to zero when r approaches zero. This demonstrates the approximate separability of pose-dependent deformation and shape-dependent deformation.

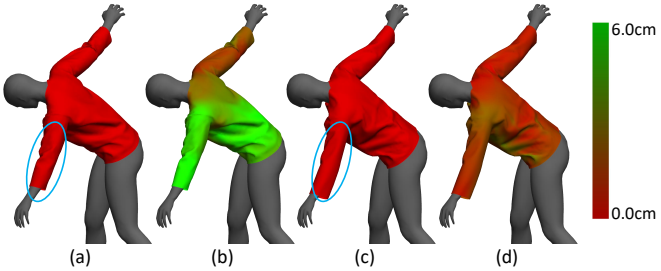


Fig. 2. Qualitative comparison between the original SOR and our method: (a) original SOR result; (b) error between original SOR result and the simulation result; (c) our result; (d) error between our result and the simulation result. The errors are marked in green. We can see that our result is closer to the ground truth and looks more natural, especially in the sleeve area.

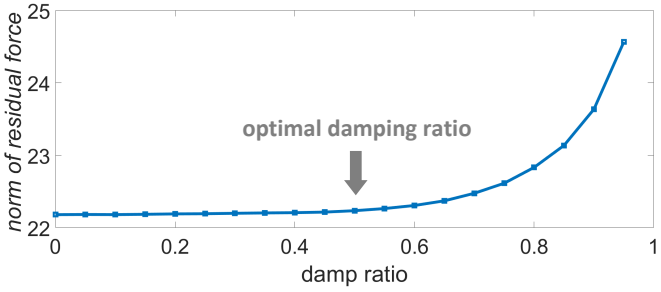


Fig. 3. Relationship between the damping ratio and the norm of the residual. We choose the maximum damping ratio that keeps the norm of the residual low.

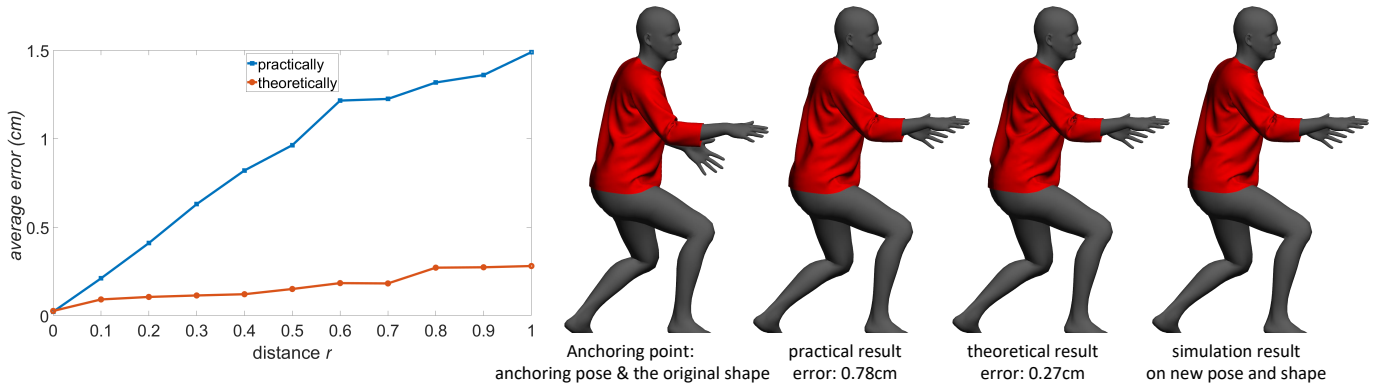


Fig. 1. Validation of the local linear separability of pose-dependent deformation and shape-dependent deformation. Practically, we use Eq. 3 in the manuscript to compute the synthesized clothing and, theoretically, we replace each term on the right-hand side of Eq. 3 with the corresponding simulation result. As shown in the graph, both approximation errors converge to zero when the distance to the anchoring point approaches zero. The images on the right are the results for a randomly generated pose and shape with $r = 0.5$. Both results (the middle two) can realistically recover the folds and wrinkles (the right-most figure is the ground truth) while the theoretical result has a smaller error.

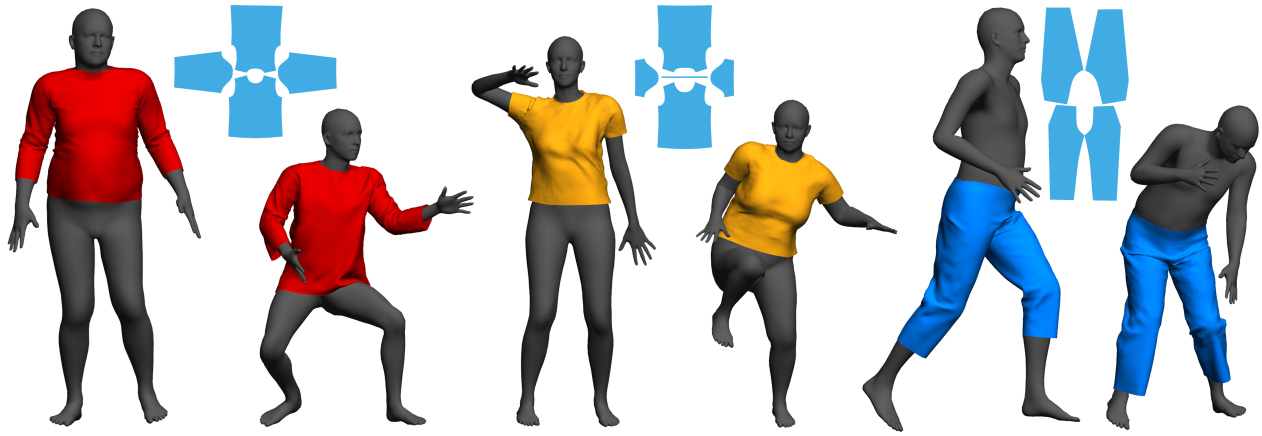


Fig. 4. Synthesis results. Clothing patterns are shown in blue.

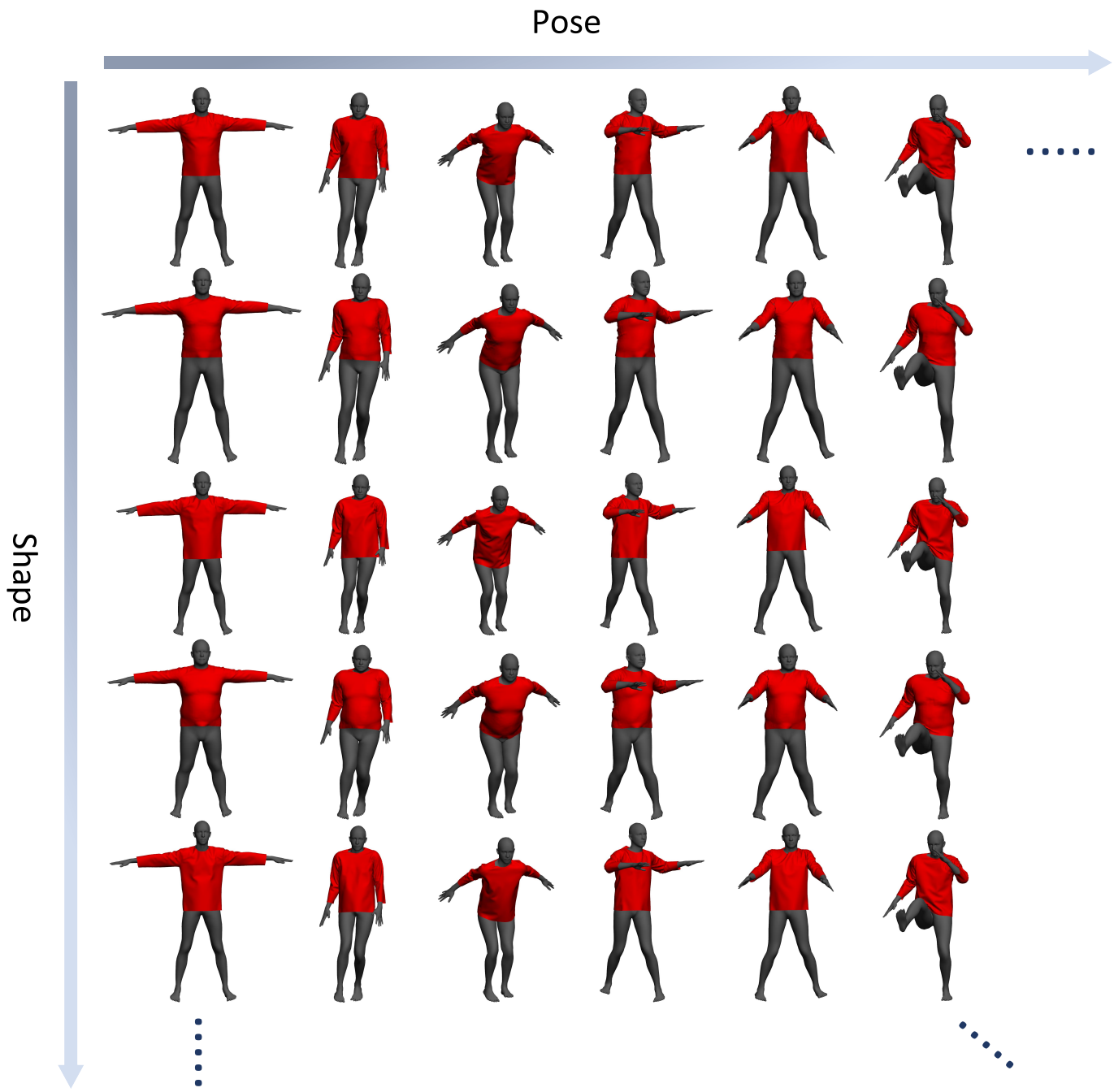


Fig. 5. Part of our training data. Each column is the training data for a clothing shape model. The poses in each row are the clustering result in the database construction stage.