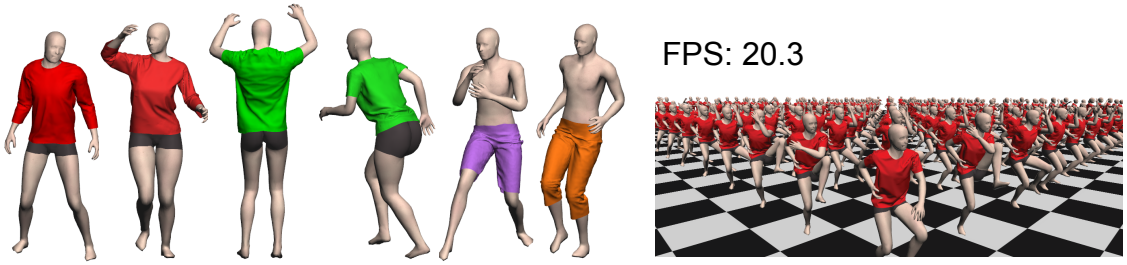
Sensitivity-optimized Rigging for Example-based Real-Time Clothing Synthesis

Weiwei Xu* [1] Nobuyuki Umentani* [2,3] Qianwen Chao[4] Jie Mao[5] Xiaogang Jin[4] Xin Tong[6]

[1]Hangzhou Normal University [2]Autodesk Research
[3]The University of Tokyo [4]Zhejiang University [5]Google Company [6]Microsoft Research



(a) (b)

Figure 1: *(a) Clothing deformations with detailed wrinkles for various clothing models and body poses synthesized using our method. (b) Our method generates hundreds of highly realistic clothing deformations for various poses in real time.*

Abstract

We present a real-time solution for generating detailed clothing deformations from pre-computed clothing shape examples. Given an input pose, it synthesizes a clothing deformation by blending skinned clothing deformations of nearby examples controlled by the body skeleton. Observing that cloth deformation can be well modeled with sensitivity analysis driven by the underlying skeleton, we introduce a sensitivity based method to construct a pose-dependent rigging solution from sparse examples. We also develop a sensitivity based blending scheme to find nearby examples for the input pose and evaluate their contributions to the result. Finally, we propose a stochastic optimization based greedy scheme for sampling the pose space and generating example clothing shapes. Our solution is fast, compact and can generate realistic clothing animation results for various kinds of clothes in real time.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation I.6.8 [Computer Graphics]: Simulation and Modeling—Animation;

Keywords: clothing animation, example-based animation, sensitivity analysis, Markov chain Monte Carlo

[1]Weiwei Xu and Nobuyuki Umentani are joint first authors.

[2]Part of this work was done when Nobuyuki Umentani and Jie Mao were interns in Microsoft Research.

1 Introduction

State-of-the-art clothing simulation techniques are capable of generating highly detailed clothing shapes and dynamics under various body poses and motions. However, producing high quality clothing deformation results in real time is still challenging. A subtle underlying body pose change can lead to rich wrinkles and complex deformation behaviors of clothing. Moreover, this non-linear influence of body to clothing is non-local. For example, clothing around the abdomen will be deformed when an arm is raised.

Physically based simulation directly models the non-linear behavior of clothing [Nealen et al. 2006; Choi and Ko 2005]. However, high-resolution clothing meshes and expensive nonlinear solvers are always required for generating realistic clothing deformation results, which makes them difficult to be used in real-time applications such as games and virtual try-ons. Data-driven clothing simulation approaches synthesize cloth deformations from pre-computed clothing deformation samples at different body poses. To achieve real-time performance, these methods compromise the result quality by simplifying the relationship between the deformed clothing and underlying body poses [de Aguiar et al. 2010; Wang et al. 2010; Guan et al. 2012] by assuming linearity or locality. A recent contribution by [Kim et al. 2013] exhaustively sampled clothing deformations with respect to a motion graph to synthesize realistic clothing deformations at run time. However, its exhaustive sampling procedure needs substantial computational resources.

In this paper, we present a pose-dependent skinning scheme to achieve real-time detailed clothing deformation. Our method consists of two stages: an offline rigging stage and a run-time synthesis stage. Given a set of pre-computed example clothing deformations sampled at different poses, we first rig the example clothing shapes to their poses with the underlying body skeleton bones at each example pose in the offline stage. To model the non-local influence of body movements on clothing deformations, our method associates each cloth vertex with both nearby bones and a remote bone. At run time, our method synthesizes a clothing deformation of the input pose by blending skinned clothing deformations computed from its nearby examples.

Constructing an optimal pose-dependent skinning scheme from sparse examples is a non-trivial task. A naive approach is to fit all parameters of the pose dependent skinning scheme with a large number of clothing shapes sampled in the pose space. However, the computational cost for both sampling and fitting are prohibitively high. We thus present a unified framework for constructing a pose-dependent rigging solution *optimally* based on the sensitivity analysis of static equilibrium of clothing deformation [Umetani et al. 2011]. Sensitivity analysis has been widely used to model the linear response of equilibrium simulation with respect to parameter perturbations based on the laws of physics. In our case, it is able to well predict the non-local influence of body pose change to the clothing deformation without expensive physical simulation.

The core of our method is an efficient sensitivity-optimized rigging algorithm for selecting the remote bones and trainings the skinning weights in the offline stage. To efficiently blend the predicted clothing deformations from our skinning scheme at run time, we also develop a sensitivity-based distance measure to find the nearby examples for an input pose and blend the skinning deformation results generated from these example poses for synthesizing the final result. Finally, we present a MCMC-based stochastic sampling process for selecting the sparse sample poses that are capable of covering a wide range of clothing deformation space, leading to a more compact and efficient example database.

Our pose-dependent skinning scheme provides a compact and efficient clothing animation solution for many real-time applications. Although our skinning scheme is trained from the static equilibrium of clothing deformation and can not capture salient inertia effects of clothing animation, it achieves a good balance between visual quality, performance and memory consumption. As illustrated in Figure 2, it well reproduces the non-local and nonlinear cloth deformation behaviors. Our method also inherits the compactness and high performance of the traditional skinning clothing deformation scheme. With a small number of examples (from 100 to 150), our method can generate detailed deformations of a clothing mesh with $12,000$ vertices over 60FPS on an ordinary consumer hardware. We validate our method with various kinds of garments, including T-shirts, long sleeves, long pants, and shorts. Figure 1 illustrates some results generated by our method.

In summary, the contributions of our method are:

- A pose-dependent skinning scheme that uses nearby bones and a remote bone for modeling the non-linear clothing deformation with moderate storage cost.
- A sensitivity-based rigging algorithm for constructing the skinning solution and efficiently blending the predictions generated from different examples.
- A stochastic optimization and incremental greedy technique for efficient example database construction.

2 Related Work

Physics-based clothing simulation. Physics-based clothing simulation has been a hot research topic in the field of computer graphics for almost two decades (please see [Nealen et al. 2006; Choi and Ko 2005] for a comprehensive review of this topic). Its central task is to construct physical models for different types of clothing deformation behaviors. These include the bending models of [Grinspun et al. 2003; English and Bridson 2008; Choi and Ko 2002] for buckling effects, the membrane model of [Volino et al. 2009] for nonlinear tensile stiffness, and the contact models of [Bridson et al. 2002; Harmon et al. 2008; Govindaraju et al. 2005; Zheng and James 2012] for self-collision avoidance. Aside from thin-shell models, yarn-based knitted cloth simulation was explored by [Kaldor et al. 2008]. Because high-resolution cloth mesh simulations impose a heavy computational load, researchers have introduced adaptive meshing to make the simulations more efficient [Narain et al. 2012]. Recent contributions start to simulate complex internal friction phenomena and contact friction for cloth to produce realistic wrinkling effects [Chen et al. 2013; Miguel et al. 2013].

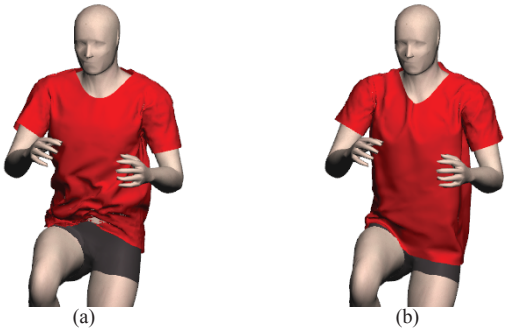


Figure 2: *Our pose-dependent skinning vs. traditional skinning for clothing deformation synthesis. (a) The result generated by traditional skinning. (b) The result generated by our skinning scheme. By blending deformation of different examples and using remote bones for modeling non-local clothing deformations, our method can synthesize realistic clothing deformation.*

Many commercial game engines, such as Nvidia PhysX™, Havok™and Bullet™support real-time clothing simulations using a simplified mass-spring system, and the GPUs are used to accelerate simulation [Müller et al. 2007]. However, most games still adopt skinning-like rigging techniques for clothing animation synthesis, as the costs of a matrix solver and collision handling raise the run-time cost, even at a low resolution. In this study, we improved upon current practices by synthesizing highly detailed clothing animations that do not require physical simulation.

Multi-resolution clothing simulation. One strategy for accelerating a high-fidelity clothing simulation is to separate it into coarse and fine meshes and compute their deformations separately: a fine mesh can be used to describe detailed wrinkles on top of a coarse mesh simulation. Wrinkle meshes [Müller and Chentanez 2010] are used to perform static high-resolution clothing simulations and can synthesize detailed wrinkles in a few iterations. The target application for this kind of simulation is computer games.

Another approach is to learn a map between coarse simulation results and the corresponding high-resolution clothing deformations, such as deformation transformers [Feng et al. 2010], implicit geometric deformer [Rohmer et al. 2010], physics-inspired upsampling [Kavan et al. 2011], or local displacement vectors [Zurdo et al. 2013]. Although these methods can generate detailed wrinkles, their generalization to new types of character motions is still limited by the training data. Wang et al. [2010] proposed modeling wrinkle deformations as a function of local joint angles reconstructed by blending pre-computed detailed clothing simulation data. Their local model works well for tight clothing, but it cannot generate medium-scale wrinkles. It may produce artifacts for loose clothing.

In contrast, our method directly trains prediction models on high-resolution meshes. Moreover, the trained model can be implemented simply as a skinning scheme at run time without coarse simulation.

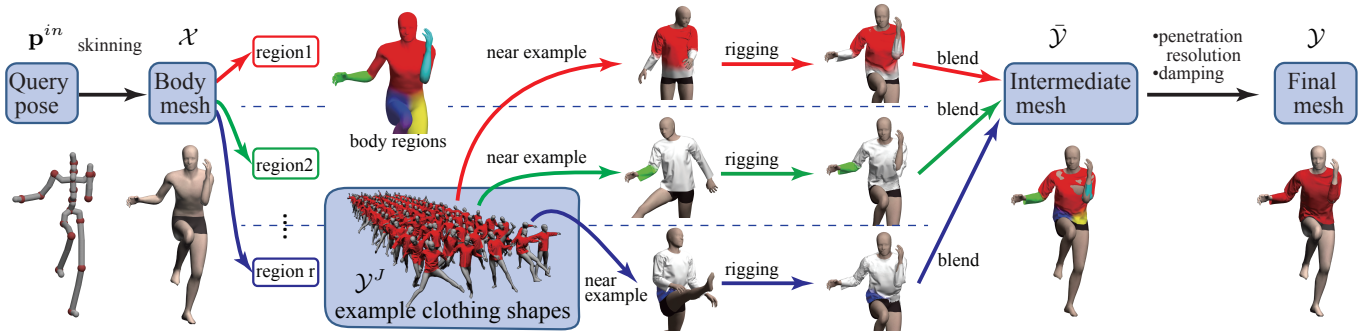


Figure 3: *Overview of our clothing synthesis workflow. Given a query pose and a skinned body mesh, we firstly choose nearby clothing examples for each region. For each example, we deform the clothing mesh with the examples' sensitivity-optimized rigging scheme (SOR). We then blend the deformations to synthesize an intermediate mesh. We then resolve penetrations and add dynamic effects and damping.*

Data-driven clothing deformation. Data-driven clothing deformation methods are designed to approximate the statics or dynamics of clothes from pre-computed clothing deformation data. Here, we focus solely on data-driven methods without coarse-level simulations. James et al. [2003] built an easy-to-control reduced state space model for clothing deformation, which enables real-time responses for simple user interactions, such as translation and rotation. Cordier et al. [2004] proposed a data-driven approach to figure out potential collision areas between clothing and the body. They also developed a geometric approach to deduce the detailed clothing deformation from real-time coarse simulation using offline clothing simulation data. The driven-shape technique [Kim and Vendrovsky 2008] computes the blending weights by matching body shape to blended clothing examples. Weber et al. [2007] also parameterized clothing deformation with the undrlying body pose. Different from these methods, we use optimized rigging and a fast weighting scheme via sensitivity analysis to synthesize realistic clothing draping effects. Recently, Kim et al. [2013] presented a technique to reproduce secondary motion of clothing deformation for a specific motion graph. However, this method needs substantial computational resource for pre-computation.

A compact representation of the relationship between the control parameters and the corresponding clothing deformation can be achieved by constructing a regression model. This has been widely used in character skinning to improve the quality of skeleton subspace deformation [Lewis et al. 2000; Kry et al. 2002; Wang et al. 2007]. For instance, Kry et al. [2002] proposed to precompute the PCA of the deformation influences of a single skeleton joint and interpolate the coefficients of the basis to reconstruct the deformation in real time. Wang et al. [2002] developed a multi-weight linear model to predict skin deformations with regard to pose parameters. For clothing deformation, De Aguiar et al. [2010] proposed a linear dynamic model from pre-computed clothing deformations on a specific virtual character. This method is stable, but not suitable for modeling highly detailed wrinkles. Recently, Peng et al. [2012] introduced the DRAPE model, which can automatically put clothing on characters with various body shapes and poses using a regression technique derived from SCAPE [Anguelov et al. 2005]. However, nonlinear clothing behavior arising from complicated interactions with the body surface is difficult to capture in a single regression model; it results in costly regularization and penetration-resolution computations at run time.

Sensitivity Analysis. Sensitivity analysis has been applied in the field of graphics to various optimization problems such as 2D rod design [Derouet-Jourdan et al. 2010] and 3D masonry design [Whiting et al. 2012]. Aside from optimization problems, Umetani et al. [2011] applied sensitivity analysis to the interactive design of clothing patterns and enabled interactive feedback from physical simulations during editing. They also applied this technique to suggestive design of structurally sound furniture [2012]. In this work, we apply sensitivity analysis for modeling nonlinear clothing deformation.

3 System Overview

System input. The input to our system is a body mesh $\mathcal{X}$ with $\mathbf{N}_{\mathcal{X}}$ vertices and a clothing mesh $\mathcal{Y}$ with $\mathbf{N}_{\mathcal{Y}}$ vertices on the body. We also define an articulated skeleton $\mathcal{B}$ with $\mathbf{N}_{\mathcal{B}}$ bones and $\mathbf{N}_{\mathcal{L}}$ joints inside the body mesh to drive the deformations of the body mesh and the clothing mesh. We represent a body pose by $\mathbf{p} = [\mathbf{R}_1, \ldots, \mathbf{R}_{\mathbf{N}_{\mathcal{B}}}]$, where $\mathbf{R}_b \in SO(3)$ is an absolute rotation of a bone b. The differences between two poses $\mathbf{p}^i$ and $\mathbf{p}^j$ can be encoded as a vector by concatenating all joint rotations' differences as $\mathbf{\Theta}(\mathbf{p}^i, \mathbf{p}^j) = [\boldsymbol{\theta}_1^{i,j}, \ldots, \boldsymbol{\theta}_{\mathbf{N}_{\mathcal{L}}}^{i,j}] \in \mathbb{R}^{3\mathbf{N}_{\mathcal{L}}}$, where $\boldsymbol{\theta}_l^{i,j} \in \mathbb{R}^3$ is the Euler angles of each joint l's rotation difference. Hereafter, we use b and l to index the function values defined for the bones and joints.

We assume the skinning scheme of the body mesh is known. In our implementation, we follow the dual-quaternion [Kavan et al. 2008] method for modeling the skinning deformation of body mesh and generate the skinning weights of body mesh vertices with Maya ™. We cancel out the translation freedoms of the body mesh by locating the skeleton's root bone at the origin of the world coordinate system. As a result, the body mesh $\mathcal{X}$ deformation can be computed as $\mathcal{X}(\mathbf{p})$. For the skinning weights $w_b \in \mathbb{R}$ defined at each vertex for each bone b, we have $\sum_{b=1}^{\mathbf{N}_{\mathcal{B}}} w_b = 1$. The top row of Figure 4 illustrates the skinning weights of the body mesh. We drop the vertex index for the skinning weights of the body mesh and clothing mesh to simplify the notation.

To model the non-linear deformation of clothing mesh under different poses, we also precomputes a set of examples of body meshes $\mathcal{X}^J$ $(J = 1, \ldots, \mathbf{N}_{\mathcal{J}})$ skinned under pose $\mathbf{p}^J$ and their corresponding clothing meshes $\mathcal{Y}^J$(Section 6). Hereafter, we denote values computed for an example pose $\mathbf{p}^J$ with a superscript J.

Our solution. Our system consists of two stages: the offline rigging stage and the run-time clothing synthesis stage. In the rigging stage, we construct a skinning scheme for each example pose. Specifically, to compute the position of a clothing vertex $\bar{\mathbf{y}}^J$ for an input pose $\mathbf{p}^{in}$, our skinning scheme transforms the position of the same clothing vertex $\mathbf{y}^J$ at its nearby example pose $\mathbf{p}^J$ by

$$\bar{\mathbf{y}}^J = \sum_{b=1}^{\mathbf{N}_{\mathcal{B}}} \left\{ w_b^J \left(\mathbf{R}_b^{J,in} \mathbf{y}^J + \mathbf{T}_b^{J,in} \right) + \mathbf{R}_b^{J,in} \tau_b^J \mathbf{T}_b^{J,in} \right\}, \quad (1)$$

where $\mathbf{R}_b^{J,in}$ and $\mathbf{T}_b^{J,in}$ represent the relative rotation and translation of a bone b from $\mathbf{p}^J$ to $\mathbf{p}^{in}$, and w_b^J and τ_b^J are bone rotation and translation weight defined on the clothing vertex respectively.

The bone rotation weight w_b^J is used to deform the example clothing deformations to an input pose according to bone rotations, which can be viewed as the standard weighting scheme in skeleton subspace deformation. The bone translation weight τ_b^J is a 3×3 matrix that transforms the bone center translation to its influence on the clothing deformation, which is similar to the multi-weighting scheme in [Wang and Phillips 2002]. Note that the translation weight defined for each clothing vertex actually plays a corrective linear model to improve the prediction from the rotation weight. We thus do not need partition of unity property of translational weighting scheme. The combination of rotation and translation weights is to approximate the non-local and nonlinear clothing deformation behaviors in prediction when example poses approach the input pose from different directions. For each example pose, our Sensitivity-Optimized Rigging scheme (SOR) selects the bones and computes their rotation and translation weights for each cloth vertex (Section 4).

In the runtime stage, our method computes the deformed clothing shape for each input pose $\mathbf{p}^{in}$ (Section 5). As shown in Figure 3, the system first searches the examples to find the nearby poses $\mathbf{p}^J$ and their clothing shapes $\mathcal{Y}^J$ using a sensitivity-based distance measure. For each nearby pose, its sensitivity-optimized skinning model is applied to predict the clothing deformation for the input pose $\mathbf{p}^{in}$ according to Equation 1. After that, we get the position of each clothing vertex $\mathbf{y}^{in}$ at the input pose by blending the clothing shapes deformed from all nearby example poses by

$$\mathbf{y}^{in} = \textstyle\sum_{J=1}^{\mathbf{N}_\mathcal{J}} w^J(\mathbf{p}^{in})\bar{\mathbf{y}}^J, \tag{2}$$

where $w^J(\mathbf{p}^{in})$ is the blending weight for the input pose defined at each vertex that is computed using our sensitivity-based distance measure technique. Finally, we refine the clothing shape by resolving penetrations between clothing mesh and the underlying body mesh.

Sensitivity analysis. Our method uses sensitivity analysis for constructing the skinning scheme of each example pose and measuring the distance of clothing shapes under different poses. Sensitivity analysis describes how the simulation results change with small changes in input parameters with first-order accuracy [Gallagher 1973]. In our case, we compute clothing deformations at a static equilibrium, where internal stress is in balance with external force. This can be described by an implicit function $\mathbf{F}(\mathcal{X}, \mathcal{Y}) = 0$, where $\mathbf{F} \in \mathbb{R}^{3\mathbf{N}_\mathcal{Y}}$ is the residual force of the system, i.e. the gradient of the total system energy [Umetani et al. 2011].

Let an example pose $\mathbf{p}^J$ be perturbed with a small joint rotation change $\Delta\Theta_m$, where $\Delta\Theta_m$ is the m-th parameter of the joint rotation angle vector $\mathbf{\Theta}(\mathbf{p}^J, \mathbf{p})$. The sensitivity analysis gives a sensitivity of $\Delta\mathcal{Y}_m$, a first-order approximation of the change in clothing deformation, which can be written into the following formula according to the implicit function theorem:

$$\Delta\mathcal{Y}_m \approx -\left(\partial\mathbf{F}/\partial\mathcal{Y}\right)^{-1}\left\{\mathbf{F}(\mathcal{X} + \Delta\mathcal{X}_m, \mathcal{Y}) - \mathbf{F}(\mathcal{X}, \mathcal{Y})\right\}, \tag{3}$$

where $\Delta\mathcal{X}_m$ is the change in body mesh with respect to a small variation of a joint $l = \lceil \frac{m}{3} \rceil$. Assuming that skinning is smooth with respect to joint rotation, we can simply compute the sensitivity using finite differencing as $\Delta\mathcal{X}_m = \mathcal{X}(\Delta\Theta_m) - \mathcal{X}^J$. This sensitivity information can be computed numerically: $\Delta\mathcal{Y}_m = \mathcal{Y}(\Delta\Theta_m) - \mathcal{Y}^J$. It is a reasonable choice when clothing is simulated by off-the-shelf commercial software packages that do not provide sensitivity information.

With Equation 3, we compute sensitivity for each clothing vertex–joint pair, namely $\mathcal{S}_m = \partial\mathcal{Y}/\partial\Theta_m \in \mathbb{R}^{3\mathbf{N}_\mathcal{Y}}$, at each example J. We can approximate the clothing sensitivity due to the change of joint angle at example J as

$$\mathcal{Y}(\mathbf{p}^{in}) - \mathcal{Y}^J \simeq \sum_{m=1}^{3\mathbf{N}_\mathcal{L}} \mathcal{S}_m^J \Delta\Theta_m, \tag{4}$$

where $\mathbf{p}^{in}$ is an arbitrary input for a parameterized pose, and $\Delta\Theta_m$ here is the m-th joint rotation angle of $\mathbf{\Theta}(\mathbf{p}^J, \mathbf{p}^{in})$ at that pose.

Although the linear prediction with sensitivity analysis in Equation 4 provides a good approximation of clothing deformation around an example pose, we can not directly apply it to clothing deformation. This is because the storage cost of the sensitivity matrix (in $3\mathbf{N}_\mathcal{Y} \times 3\mathbf{N}_\mathcal{L}$) is huge and the linear behavior of sensitivity matrix cannot well predict the non-linear behavior of cloth deformation caused by the rotation between the input pose and the example pose. In our solution, we model the cloth deformation with pose-dependent skinning scheme and apply sensitivity analysis for constructing our solution.

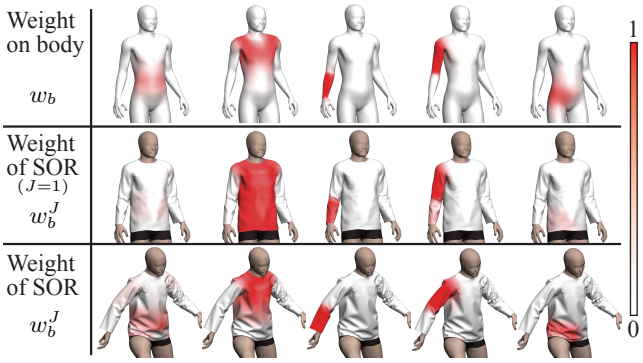


Figure 4: *Distributed weights for bones over the body surface (top) and their effects on clothing (bottom rows). Note that the SOR weights vary in different example poses (see the second / third row).*

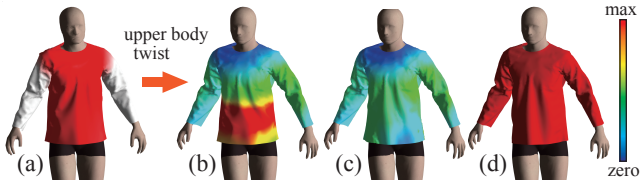


Figure 5: *Distribution of α for chest bone, showing global influence of this bone in the example pose (a). While there are unnatural wrinkles in the clothing deformation computed from local skinning weight only (b) with respect to upper body twist, our SOR method generates natural draping shape (c), which agree with ground truth equilibrium shape (d). The color bar here indicates the magnitude of ρ in Equation 7 that is difference from sensitivity analysis.*

4 Sensitivity-Optimized Rigging

Given a set of pre-computed examples of body meshes $\mathcal{X}^J$ skinned under pose $\mathbf{p}^J$ and their corresponding clothing meshes $\mathcal{Y}^J$, our sensitivity-optimized rigging scheme (SOR) determines the associated bones and their weights for each clothing mesh vertex at each example pose J in Equation 1. In particular, we first select the

nearby bones for each clothing vertex. We then determine the remote bone and compute the rotation weights of all bones w_b^J using the sensitivity analysis result at pose J. Finally, we compute the translation weights for all of the bones.

Nearby bone selection. To determine the nearby bones of a cloth vertex, we first find its closest vertex on the body mesh $\mathcal{Y}^J$ at the rest pose $J = 1$ (as shown in the middle row of Figure 4) and record it as the corresponding body mesh vertex of this cloth vertex. We then assign the associated bones of this corresponding body mesh vertex to the cloth vertex as its nearby bones. As a result, most cloth vertices are associated with only one nearby bone, while the cloth vertices that are close to skeleton joints are associated with two or more nearby bones.

Remote bone selection and rotation weights training. For each cloth vertex, we define the rotation weight of each bone as

$$w_b^J = (1 - \alpha^J)w_b + \alpha^J \delta(b, b^J), \tag{5}$$

where $w_b \in \mathbb{R}$ is the skinning weight of the corresponding body mesh vertex for the same bone b (as shown in Figure 4). The remote bone b^J is used to approximate the motion of cloth vertex affected by remote body surface motion. The Kronecker's delta $\delta(b, b^J)$ is 1 if b is equal to b^J; otherwise 0. Therefore, the first local influence term describes how the clothing vertex moves with nearby underlying body surface, while the second term defines how the clothing vertex moves with a remote bone b^J. The α^J is the blending ratio of two terms.

Given the rotation weight definition, we compute the blending ratio α^J and determine the remote bone b^J by minimizing the difference between the response of physical deformation $\Delta\mathbf{y}$ obtained from sensitivity analysis and $\Delta\bar{\mathbf{y}}^J$ obtained from the SOR scheme Equation 1 under small joint rotation $\Delta\Theta_m$. The change of each joint angle is set to be 0.5 degree in our system to form $\Delta\Theta_m$.

The clothing deformation difference can be converted to the difference in sensitivity, given the joint angle change $\Delta\Theta_m$. To this end, we first compute a sensitivity $\mathbf{s}_m^J$ that describes the change of clothing vertex $\mathbf{y}$ with respect to $\Delta\Theta_m$ (in Equation 3). Then, we compute the sensitivity $\Delta\bar{\mathbf{y}}^J$ of SOR using Equation 1:

$$\Delta\bar{\mathbf{y}}^J = \bar{\mathbf{y}}^J(\Delta\Theta_m) - \mathbf{y}^J = \bar{\mathbf{s}}_m^J(\alpha, b)\Delta\Theta_m, \tag{6}$$

where $\bar{\mathbf{s}}_m^J(\alpha, b)$ is the sensitivity matrix of SOR for a specific choice of bone b and ratio α. Similarly, $\bar{\mathbf{s}}_m^J(\alpha, b)$ can be easily computed using a numerical derivative.

The optimal bone b^J and ratio α^J are chosen to minimize the difference between $\mathbf{s}_m$ and $\bar{\mathbf{s}}_m^J(\alpha, b)$ in a least squares manner:

$$\rho(\alpha, b) = \sum_{m=1}^{3\mathbf{N}_\mathcal{C}} \left\| \mathbf{s}_m^J - \bar{\mathbf{s}}_m^J(\alpha, b) \right\|^2, \tag{7}$$

$$\left\{ b^J, \alpha^J \right\} = \underset{\alpha\in[0,1], 1\le b\le \mathbf{N}_\mathcal{B}}{\arg\min} \rho(\alpha, b), \tag{8}$$

where $\|\mathbf{s}\|$ is a L2 norm of the vector $\mathbf{s}$. The derivative of $\bar{\mathbf{s}}_m^J(\alpha, b)$ with respect to α is computed as $\partial\bar{\mathbf{s}}_m^J(\alpha, b)/\partial\alpha = \bar{\mathbf{s}}_m^J(1, b) - \bar{\mathbf{s}}_m^J(0, b)$. Equation 7 is optimized for each possible choice of b, and the optimal $\left\{ b^J, \alpha^J \right\}$ are chosen so that Equation 7 is minimized. Figure 5(b) and (c) show how SOR reduces the sensitivity approximation error compared to naïve rigging weights using the nearest body surface's weight w_b as the clothing rigging weight. Please notice the natural draping shape of the clothing around the abdomen produced by SOR.

Translation weights training. After the rotation weights and the remote bone are determined, we compute the translation weights for nearby bones only. The translation weights for the remote bone are set to zero.

According to Equation 1, the prediction errors of clothing deformations with bone rotation weights can be computed by $e_\mathbf{y} = \bar{\mathbf{y}}^J - \sum_{b=1}^{\mathbf{N}_\mathcal{B}} w_b^J \left(\mathbf{R}_b^{J,in}\mathbf{y}^J + \mathbf{T}_b^{J,in} \right)$. For translation weight computation of a clothing vertex, we first represent $e_\mathbf{y}$ in the local coordinate systems of bones, distribute $e_\mathbf{y}$ using the skinning weight of its closest body surface vertex, and then train the weights using regularized least-squares fitting. We modify the joint angles $\Delta\Theta_m$ to obtain enough samples for rotation weight training and translation weights training. With our two-step rigging scheme, our skinning scheme can efficiently reduce the prediction errors for new input poses.

5 Run-time Clothing Deformation

Given an input pose, our method first finds its nearby example poses through a sensitivity-based distance measure and then compute clothing deformation results with the constructed skinning schemes from each nearby example pose as shown in Equation. 1. After that, we blend the clothing deformation results with blending weights computed from the distance measure. To improve the result quality, we decay the blending weights change between frames and apply a simple scheme to resolve the penetrations between cloth and the body surface.

Sensitivity-based distance measure for blending. The sensitivity distance measure is a metric for searching the nearest neighbor examples and blending their skinned clothing deformations. While the distances using joint angles or body surfaces are natural candidates for the distance measure, they are only indirect measures of the differences in clothing deformations between examples and the input pose. Due to the curse of dimensionality, measuring distance based on differences in body surface is computationally expensive and may require a substantial amount of memory [de Aguiar et al. 2010].

Therefore, we resort to a distance measure that directly approximates differences between clothing meshes using sensitivity information. Specifically, the difference between two clothing meshes is: $||\mathcal{Y}(\mathbf{p}^{in}) - \bar{\mathcal{Y}}^J||^2$, where $\mathcal{Y}(\mathbf{p}^{in})$ are actual clothing meshes in the query pose and $\bar{\mathcal{Y}}^J$ is the prediction computed using SOR. But how can we get $\mathcal{Y}(\mathbf{p}^{in})$, which is not included in the query? The key idea again is sensitivity analysis to estimate $\mathcal{Y}(\mathbf{p}^{in})$ from examples clothing shapes. Thus, we call the distance measure the *sensitivity-based distance measure*. Moreover, the sensitivity analysis technique for clothing deformation used in Equation 4 can render the distance computation back to joint angles. It is computationally efficient since the number of joint angles is much less than clothing mesh vertices.

Following [Wang et al. 2010], we further loosely decompose the clothing mesh into several *regions* g so that the nearby examples can be determined for each region, since this can efficiently reduce the required examples in database. The regions are carefully determined to make the clothing deformation at each region relatively independent. As illustrated in the top-left of Figure 7, we manually separate the bones into $\mathbf{N}_G$ regions ($\mathbf{N}_G = 7$ in our examples) based on their influences. We then compute a region weight $\hat{w}_{g,y}$ for each clothing vertex y. Specifically, the region weight for each clothing vertex is computed as the sum of the skinning weights of its closest body mesh vertex at J = 1 for the bones that belong to the region. In this way, for a clothing vertex y whose associated

bones belong to the same region g, its weight $\hat{w}_{g,y}$ for the region will be 1; and 0 for the rest regions. For those clothing vertices associated to bones that are not in the same region, the sum of their region weights is guaranteed to be 1. Note that this region weight is smooth over the clothing mesh so that we can maintain smoothness around the joints.

For each region, we compute the distance $D_g^J(\mathbf{p}^{in})$ between the clothing shapes of an example and an input pose by approximating weighted sum of differences in clothing meshes as a weighted sum of the difference in joint angles:

$$\begin{aligned} D_g^J(\mathbf{p}^{in}) &\simeq \sum_{y\in\mathcal{Y}} \hat{w}_{g,y} \left\| \mathbf{y}^{in} - \bar{\mathbf{y}}^J \right\|^2 \\ &\simeq \sum_{y\in\mathcal{Y}} \hat{w}_{g,y} \sum_{m=1}^{3\mathbf{N}_\mathcal{L}} \left\| (\mathbf{s}_m^J - \bar{\mathbf{s}}_m^J)\Delta\Theta_m \right\|^2 \\ &\simeq \sum_{m=1}^{3\mathbf{N}_\mathcal{L}} U_{g,m}^J \left\| \Delta\Theta_m \right\|^2, \end{aligned} \tag{9}$$

where the weight $U_{g,m}^J = \sum_{y\in\mathcal{Y}} \hat{w}_{g,y} \left\| \mathbf{s}_m^J - \bar{\mathbf{s}}_m^J \right\|^2$ represents how much the joint angle differences result in clothing deformation differences, predicted by the sensitivity analysis using Equation 4. We compute the weight in the offline training stage at each data point, where $\mathbf{s}_m^J$ and $\bar{\mathbf{s}}_m^J$ are available.

Because this weighting scheme has a dimension of $\mathbf{N}_G \times (3\mathbf{N}_\mathcal{L})$ and is very compact, we can compute the distance very quickly with small storage cost. Our weighting scheme is pose-dependent. The distribution $\left\| \mathbf{s}_m^J - \bar{\mathbf{s}}_m^J \right\|^2$ over the clothing mesh varies from pose to pose as illustrated in Figure 6, where m is selected to be the right shoulder joint.

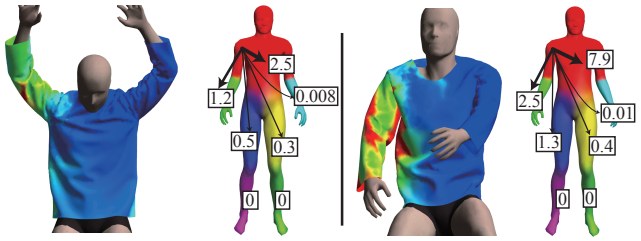


Figure 6: *Distribution of differences in the sensitivity matrix over a clothing mesh. We compute the sensitivity matrix with respect to the rotation of the right shoulder joint. The numbers on the regions represent values of $U_{g,m}^J$.*

Given the sensitivity based distance $D_g^J(\mathbf{p}^{in})$, we compute the distance weight for clothing shapes deformed from each example pose with the k-th power of the inverse distances $W_g^J(\mathbf{p}^{in}) = 1/(D_g^J(\mathbf{p}^{in})+\epsilon)^k$. Here ϵ is a small number (default of 10^{-15}) that prevents division by zero. We can control the influence of closer examples by increasing k, sacrificing the smoothness of animation. Our experiments revealed that blending a large number of example deformations with similar weights smoothed the fine details of the clothing. While using the result deformed from the nearest example pose results in discontinuity of the animation due to changes of a nearest example. In our current implementation, we set $k = 6$.

Finally, the blending weight for each clothing vertex $w^J(\mathbf{p}^{in})$ in Equation 2 is computed by its region weight and distance weight as

$$w^J(\mathbf{p}^{in}) = \sum_{r=1}^{\mathbf{N}_G} \left(\hat{w}_g W_g^J(\mathbf{p}^{in}) / \textstyle\sum_{J=1}^{\mathbf{N}_\mathcal{J}} W_g^J(\mathbf{p}^{in}) \right). \tag{10}$$

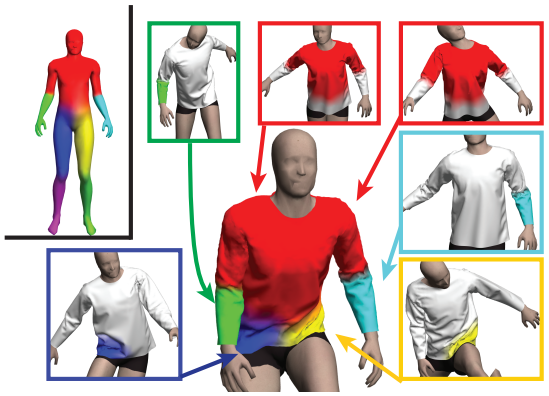


Figure 7: *Various deformation examples used to synthesize the deformation result for different parts of the human body.*

For simplicity, we also drop the vertex index for all variables related to clothing mesh vertices in this equation.

Decaying effects. Given an input pose, the corresponding clothing shape is generated by blending the example clothing shapes using SOR and sensitivity-based distance measure. In most cases, the sensitivity-based weighting scheme described above produces time-coherent output since the distance between input and examples changes smoothly over time. However, for sudden input pose changes, the weighting scheme may still experience sudden change and consequently produce sudden changes in the clothing synthesis result. We prevent such a problem by introducing *distance damping*, where damping is achieved by blending the distance at the current time step D_g^J with that of the previous time-step D'^J_g as $D_g^J := \xi D'^J_g + (1-\xi) D_g^J$. Here the ratio $\xi = \exp(-dt/t_{mix})$ is determined by step size dt and constant time t_{mix}. We chose t_{mix} as 0.05s. This means that the clothing will continuously change around 0.05s. This damping scheme is an ad-hoc method, which mimic the delayed deformation response of clothing with respect to rapid body motion.

Resolving penetrations. The final step in run-time clothing synthesis is to resolve penetrations, which is actually a re-projection scheme as illustrated in Figure 8. After obtaining the intermediate blending result for an input pose, we can return to each of its nearest deformation examples and resolve penetrations by projecting clothing vertices along the normal direction of their associated body vertices, which can be formulated into the following formulas:

$$\hat{\mathbf{y}}^J = \mathbf{y}^{in} + \mathbf{n}\bar{d}^J \tag{11}$$

$$\bar{d}^J = -\min\left\{ h^J - {d_0}^J, 0 \right\}, \tag{12}$$

where $\hat{\mathbf{y}}^J$ is the warped vertex position. The symbol $\bar{d}^J$ is the penetration depth in the current configuration, and ${d_0}^J$ is the minimum allowed penetration clearance chosen. We set ${d_0}^J = \min\{{h_0}^J, \epsilon_p\}$, where ${h_0}^J$ is the height of a clothing vertex over its associated vertex v^J in the example configuration. In all of our test, we set ϵ_p as $5mm$, the same margin as the penetration depth margin in the clothing physics simulation. This ensures that the depth of a clothing vertex point is not deeper than the initial depth ${h_0}^J$ or the clearance height ϵ_p. After we project the vertex using the nearest body vertex, we can blend the projected position again to update

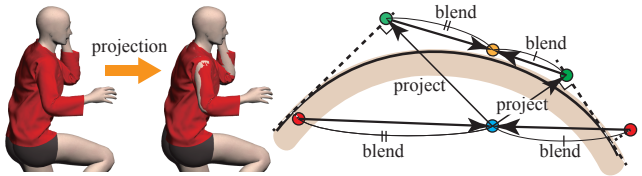


Figure 8: *Our overall penetration resolution scheme. First, a clothing vertex is predicted for the input pose from its nearby example deformations (red points), and then they are blended into an intermediate vertex position $\mathbf{y}^{in}$ (blue point). This is projected onto the body surface for each example $\hat{\mathbf{y}}^J$ (green points). Finally, it is blended again to produce the final output position $\bar{\mathbf{y}}^{in}$ (orange point).*

the final output vertex position using the weight in Equation 10 as:

$$\bar{\mathbf{y}}^{in} = \sum_{J=1}^{\mathbf{N}_\mathcal{J}} w^J(\mathbf{p}^{in})\hat{\mathbf{y}}^J. \tag{13}$$

Our penetration-resolving scheme works well for surfaces that can be locally approximated using spheres or cylinders, because in this case the projected vertex is guaranteed to be outside the body surface. Given the fact that skinned human body surfaces usually meet this requirement, especially in games, this scheme works well in most situations.

6 Example Database Construction

So far, we have described how to reconstruct deformed clothing shape for an input pose based on its nearby example poses. In this section, we will focus on how to sample example poses to build a compact database with high prediction capacity. Previous example-based clothing studies have used either empirically selected poses or sampled pose spaces with joint angles at certain intervals [Wang et al. 2010]. However, it is difficult to evaluate the performance of a database created by those methods: biased clothing deformations from the samples or wasted valuable storage space might occur when the sampled clothing deformation is not important to the final result.

Our sampling scheme is guided by a prediction error for example poses $\mathbf{p}^J \in \mathcal{P}$, where the error can be viewed as a measure of the physical plausibility of the predicted clothing deformation around such poses. The database is constructed by incrementally adding example poses over the pose space $\mathcal{P}$ to minimize the global prediction error by Monte Carlo Markov Chain (MCMC) sampling, as illustrated in Figure 9.

Prediction error. We use the norm of the residual force, $\|\mathbf{F}\|$, in a static simulation as the prediction error to measure the physical plausibility of the simulation. The residual force consists of the fabric's strain force and contact force for all clothing vertices. Once a clothing deformation deviates from the equilibrium status in a static simulation, the norm of the residual force describes the magnitude of the deviation. Therefore, this metric can capture both unnatural deformation and penetration. We compute $\|\mathbf{F}\|$ for the clothing deformation predicted by the SOR representation as a measure to locate the data points whose deviation from equilibrium status in the prediction is severe.

Sampling algorithm. Our database is parameterized by poses, i.e. the joint angles. The goal of sampling is to determine the locations of example poses and thereby minimize global error maxima over the entire pose space. Following a construction method based on sampling for nonlinear functions in [Bickel et al. 2010; Carr et al. 2001], we can incrementally construct a database using the *greedy method*. Specifically, once we sample a pose $\mathbf{p}$ uniformly from candidate motion sequences, the algorithm proceeds to find the sampling pose $\mathbf{p}^{max}$ around $\mathbf{p}$ that maximizes the error $\|\mathbf{F}(\mathbf{p}, \mathcal{X}, \mathcal{Y})\|$ of the synthesized deformation for $\mathbf{p}^{max}$ with the current database. A new data point is then created by storing its joint angles, SOR weights and $U^J_{g,m}$ weight for the sensitivity-based distance measure. The pseudocode of our sampling algorithm is shown in Algorithm 1.

Since the residual is a nonlinear function of joint angles with many local maxima and its derivative is difficult to compute, we use the Metropolis-Hastings algorithm [Press et al. 2007] to identify the next sample pose with the maximum error. Following previous research [Merrell et al. 2011], we define an objective function as

$$f(\mathbf{p}) = \exp\left\{\frac{\xi\,\|\mathbf{F}(\mathbf{p}, \mathcal{X}, \mathcal{Y})\|}{\|\mathbf{\Theta}(\mathbf{p}^1, \mathbf{p})\| + C}\right\}, \tag{14}$$

where ξ is a constant that determines to what degree the system accepts a candidate with a smaller objective function. The system accepts a new proposal $\mathbf{p}^*$ from a current parameter $\mathbf{p}$ at a probability of $\min\{1, f(\mathbf{p}^*)/f(\mathbf{p})\}$. Here, a smaller ξ leads to wider exploration and a larger ξ leads to local exploration. We gradually increase ξ over the sampling procedure.

The square norm $\left\|\mathbf{\Theta}(\mathbf{p}^1, \mathbf{p})\right\|$ is computed using joint rotation angles from the rest pose $\mathbf{p}^1$ to $\mathbf{p}$ to penalize extreme poses, and the constant C is chosen at about one tenth of the maximum value of $\left\|\mathbf{\Theta}(\mathbf{p}^1, \mathbf{p})\right\|$. This term is introduced according to the observation that unnatural extreme poses usually maximize the prediction error. However, sampling such poses is not efficient since they rarely appear in motion sequences. Hence, we prevent sampling such extreme poses by penalizing total joint rotation angles from a predefined rest pose $\mathbf{p}^1$, where the character is stand-still with arms down in our implementation.

We represent the pose space $\mathcal{P}$ so that each element of a parameter has a certain range $\Theta_m \in [\Theta_m^{min}, \Theta_m^{max}]$, where Θ_m is the m-th parameter of the joint rotation angle vector $\mathbf{\Theta}(\mathbf{p}^1, \mathbf{p})$. The system randomly samples poses over this pose space to find the global error maximum. One sampling pose $\mathbf{p}$ is changed to another one $\mathbf{p}^* \in \mathcal{P}$ with a proposal density function $\mathcal{Q}(\mathbf{p}|\mathbf{p}^*)$:

$$\mathcal{Q}(\mathbf{p}|\mathbf{p}^*) = \prod_{m=1}^{3\mathbf{N}_\mathcal{L}} \exp\left\{\frac{-\mu|\Theta_m - \Theta_m^*|^2}{|\Theta_m^{max} - \Theta_m^{min}|^2}\right\} \tag{15}$$

where μ is a parameter to control the distribution of system suggestions (we set $\mu = 0.1$). Note that this proposal density function is a symmetric scaled normal distribution.

During Metropolis-Hastings sampling, a fixed number of parameters $\#sample$ is sampled (we set $\#sample = 2000$) and then the

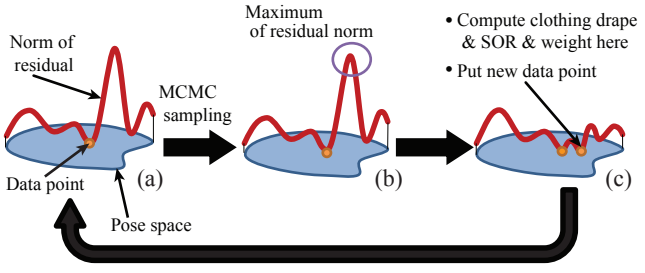


Figure 9: *Greedy procedure for constructing a database. (a) The database includes data points and parameters. (b) Sampling where the prediction error is maximized. (c) A new data point is created and the database is updated.*

Algorithm 1 Database Construction

for $J = 1$ to J_{max} **do**
  /*MH sampling*/
  for $ismpl = 1$ to $\#sample$ **do**
    Propose parameter $\mathbf{p}^*$ from $\mathbf{p}$ /* Equation 15 */
    Compute $\mathcal{X}^*(\mathbf{p}^*)$ using skinning
    Compute clothing deformation $\mathcal{Y}^*(\mathbf{p}^*)$ /* Equation 2 */
    Compute norm of residual $\|\mathbf{F}(\mathbf{p}^*, \mathcal{X}^*, \mathcal{Y}^*)\|$
    Accept $\mathbf{p} \leftarrow \mathbf{p}^*$ with probability $\min\{1, f(\mathbf{p}^*)/f(\mathbf{p})\}$
  end for
  $\mathbf{p}^{max} = \mathbf{p}$
  Find an example pose $\mathbf{p}^{near}$ closest to $\mathbf{p}^{max}$.
  /*Physics Simulation*/
  Initialize $\mathcal{Y}^J = \mathcal{Y}^{J_{near}}$
  for $t = 0$ to t_{conv} **do**
    $t \leftarrow t + dt$
    Interpolate body parameter $\mathbf{p}^t = (1-t)\mathbf{p}^{near} + t\mathbf{p}^{max}$
    Compute body surface mesh $\mathcal{X}_t(\mathbf{p}^t)$
    Step time simulation $\mathcal{Y}$ for body mesh $\mathcal{X}_t$
  end for
  /*Store data to database*/
  Compute sensitivity $\partial\mathcal{Y}/\partial\mathbf{\Theta}$ /* Section 3 */
  Compute b_i^J, α_i^J, $\bar{M}_i^J$ and weight $U_{g,m}^J$ /* Section 4 */
  Compute nearest body vertex v_i^J for each clothing vertex
  Add tuple $< \mathbf{p}^J, U_{g,m}^J, v^J, b^J, \alpha^J, \bar{M}^J, \mathcal{Y}^J >$ to the database
end for

pose with maximum error is kept as the next example pose. We also check for self-penetrations of the body mesh and prevent sampling poses with such penetrations. Once the example pose has been determined, we compute its clothing shape from an existing example shape considering its nearest pose $\mathbf{p}^{near}$. The distance measure in the example pose and current pose is determined by summing the distance measures in Equation 9 over all of the regions $D^J(\mathbf{p}^{in}) = \sum_{g=1}^{\mathbf{N}_G} D_g^J(\mathbf{p}^{in})$.

Clothing simulation. We run the physics-based clothing simulation and sensitivity analysis in the database-construction stage. Any clothing model can be used as long as it is based on the variational method; this is the only necessary requirement to perform a sensitivity analysis. We choose to use the same model that was used in previous research [Umetani et al. 2011], which combined the StVK membrane model [Volino et al. 2009] and the isometric bending model [Bergou et al. 2007]. Self-collision is handled using the technique described in [Harmon et al. 2008]. We can compute the static draping of clothing for a body shape with pose $\mathbf{p}^{max}$ using dynamic simulation based on the nearest body shape with pose $\mathbf{p}^{near}$. We gradually change the pose during the time period t_{conv} to obtain the convergent cloth, simulated using $t_{conv} = 1$s.

7 Experimental Results

Motion and body surface skinning. All motion sequences, except for the Kinect try-on demo, are obtained from the CMU Motion Capture library [CMU 2003]. We use a skeleton that has 22 bones (as shown in Figure 3) and each bone has a range of rotation angles that is already studied with the MoCap library. Our scheme doesn't have much restriction on the body surface parametrization algorithm as long as the body surface continuously changes with input parameter $\mathbf{p}$. We apply the dual quaternion blending [Kavan et al. 2008] skinning method for all the examples in this paper. Each body mesh consists of 12K triangles and 6K vertices. We used Pinnochio [Baran and Popović 2007] to define the skinning weight of a given character's mesh with respect to its skeleton.

Clothing	T	LS	Sh	LP
$\mathbf{N}_\mathcal{Y}$	11k	12k	10k	12k
number of triangles	22k	22k	19k	22k
runtime frame rate (FPS)	61	60	70	60
$\mathbf{N}_\mathcal{L}$	150	120	100	170
database size (MB)	52	44	32	45
construction time (hrs)	32	26	21	42
simulation 1 step (msec)	1140	1220	920	1280

Table 1: *Statistics for five different clothing databases. "T","LS","Sh", and "LP" stand for T-shirt, Long Sleeve shirt, Shorts and Long Pants respectively. This performance number was computed on an Intel I5 CPU. We used 32-bit integer value and floating point values in database size computation.*

Database. We create databases for four clothing models: T-shirt, long sleeve shirt, shorts and long pants for male and female body shapes (see Figure 1 and 10). Table 1 shows the details of databases constructed for the male body shape. As in [Kim et al. 2013], we adopt PCA to reduce the memory footprint. Moreover, to reduce reconstruction time, we first group the weights in the database into six-eight clusters and then perform local PCA for each cluster. This setting will reduce the number of bases required in the reconstruction (see Table 1 for the database sizes).

Each clothing in our paper is simulated with 2D cloth panels stitched together in the physics simulation. The panel patterns used are taken from a sewing textbook [Digest 2010]. The same material parameters are applied for all types of clothing: bending stiffness is 10^{-5} Nm, stretching stiffness is 30 N/m, area density is 0.1 kg/m^2, and dynamics and static coefficient of friction is 0.3. All the timings are obtained with a 64-bit desktop machine with a 3.1 GHz Intel Core™I5-750 processor, 8GB memory, and an Nvidia GeForce GTX™580 video card. Due to the simple formulation of our runtime clothing synthesis, we can easily implement our algorithm on the GPU. Using both the CPU and GPU, we can synthesize 200 characters' bodies at different poses and clothing on them with more than 20 frames per second (see accompaning video). All the codes are implemented with C++ except for GPU implementation with CUDA™.

Nonlinear deformation behavior. Our prediction and blending scheme can represent nonlinear and global relationships between body and clothing deformations. Figure 11 shows our clothing deformation results when a character rotates his shoulder and raises his left arm. One can see that the clothing around the abdomen is lifted up and forms new wrinkles, showing the global influence of the body geometry over the cloth both at coarse and fine levels. As the character raises the arm higher, new wrinkles forms due to nonlinear deformation behavior, which is well captured by our algorithm. Figure 1 and 10 illustrate the high-fidelity clothing animation result synthesized by our algorithm. Please also see the accompanying video for animation results.

Our method tries to seek a balance between physical plausibility and speed, and it achieves high quality detailed clothing deformations. However, artifacts in clothing animations, such as un-natural wrinkles and penetration artifacts, are still sometimes observable, for instance, the penetration artifacts between the T-shirt and the thigh of the virtual character around 45 seconds in the video.

Bone translation weight. The effect of bone translation weights is measured through the sum of accelerations at each frame. Figure 14 shows that bone translation weights can reduce about 15% of accelerations. It indicates that the velocity change of the ani-

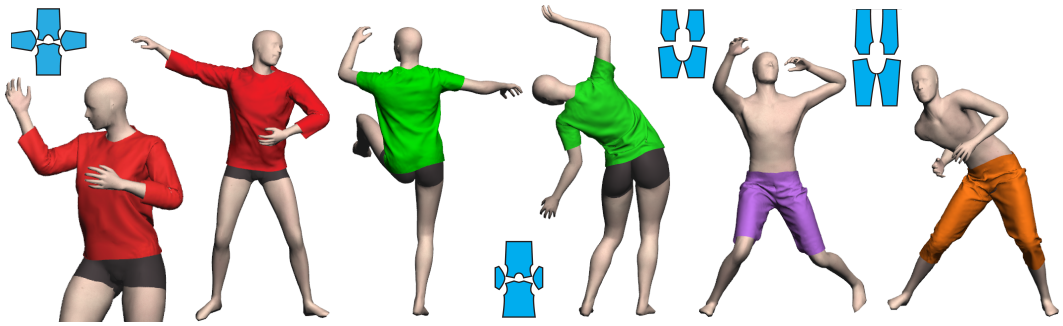


Figure 10: *Synthesis results (at least five examples are blended for the synthesis). Clothing patterns for these clothing are shown in blue.*

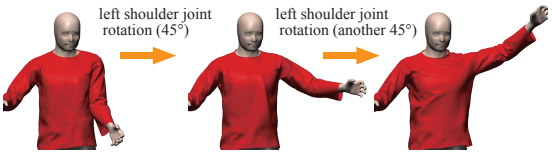


Figure 11: *Nonlinear and global deformations of clothing with respect to shoulder joint.*

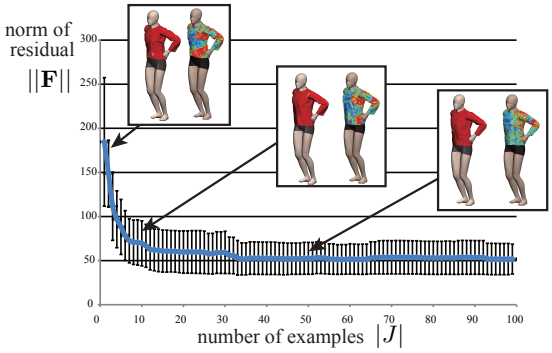


Figure 12: *Convergence of residual norm during database construction. The blue line shows average of residual sampled over 400 random poses and the error bars show their standard deviations. The synthesized clothing and their residual distributions at the same pose are illustrated from the database with examples* $|J| = 2, 10, 50$.

mation from bone translation weights is smaller, which will lead to smoother animations. Please also see the accompanying video for a comparison of the animation synthesized with and without bone translation weights.

Convergence. We examined the convergence of our database construction algorithm described in Section 6 through the norm of residual force in the constructed database with respect to increasing numbers of examples for the long sleeve shirt model (Figure 12). The residual force norm is computed over 400 randomly sampled poses from ten different motion capture sequences as the input for each database. First, the residual steadily decreases by sampling poses with a large residual. As illustrated in Figure 12, at around 40 examples, the residual norm converges and our algorithm produce only small residual for each input pose. We continue to add examples after convergence so that the clothing deformation provides various wrinkle patterns with respect to pose change. Figure 12 also shows that the distribution of the residual over the clothing become smaller and more distributed after adding examples.

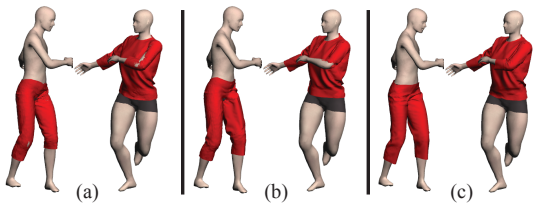


Figure 13: *Comparisons with other synthesis methods. The synthesis results from straightforward distance measures show unnatural clothing deformations: (a) using all joints' rotation difference with equal weights. (b) using region-based joint angle differences. (c) our distance measure. Please see the accompaning video for animation comparison.*

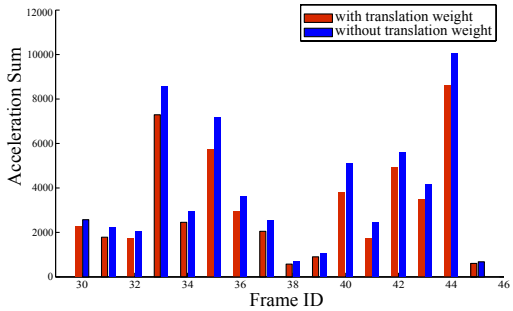


Figure 14: *The effect to reduce the velocity change using bone translation weights. It is measured on the arm rotating animation (please see video clips at 3:43).*

Distance measure. We investigated the performance of our sensitivity based distance measure by a comparison with two straightforward methods. One of them uses the sum of squared differences of all the joint angles as the distance. Precisely, the distance is computed by $||\mathbf{\Theta}(\mathbf{p}^a, \mathbf{p}^b)||$ for two poses $\mathbf{p}^a$ and $\mathbf{p}^b$. The other straightforward method is constructing the distance measure using the sum of joint angle differences at each region. We use the same set-up of regions as described in Section 5. Compared to the two straightforward methods, our distance measure properly considers the global influence of each joint and produces natural and smooth clothing animations. Figure 13-(a),-(b) and -(c) compare clothing deformations at the same pose synthesized by these two straightforward distance measures against our approach using the same database. For animation results, please see our accompanying video.

8 Discussion and Limitations

An important feature of our algorithm is the incorporation of sensitivity information to predict the clothing deformation near example poses for the subsequent blending. Since the prediction is derived from the static equilibrium equation, we can achieve a phys-

ically plausible result by blending without coarse-level simulation. The algorithm thus runs very fast on GPUs, supporting interactive highly-detailed clothing animations on hundreds of virtual characters simultaneously (please see the accompaning video). We believe the principle of predict-then-blend can be applied to improve the performance of data-driven methods in other areas of computer graphics.

Limitations. Although our algorithm does not require tight clothing, it still assumes that the clothing deformation can be parameterized by the body pose. It hinders the application of our algorithm to those clothing deformations with significant inertia effects, such as the deformation of a skirt worn on a dancer. As the clothing shape is affected globally and nonlinearly with respect to body geometry, the dynamic property of clothing should also be affected similarly. How to develop a compact model to predict the inertia effect of clothing deformation is an interesting research topic.

Our sensitivity-based rigging scheme is only a linear model to predict static clothing deformations locally. Therefore, the history-dependant nonlinear clothing deformation behaviors caused by frictions, such as cloth internal frictions and the frictions between clothing and body surface, can not be fully captured. In addition, since our scheme is trained based on static equilibrium, the wrinkles generated in the animation sometimes appear and disappear suddenly. Clothing hysteresis effect is thus not modeled well in our method. One possible solution is to extend our sensitivity-based prediction framework into locally second or high order models to capture such behaviors.

Our method cannot guarantee that all the penetrations are resolved, especially when the clothing slides on the body surface a lot. In such cases, the blending result may lead to deep penetrations that exceed the ability of our penetration resolving scheme. In our experiments, under most situations, the penetrations can be resolved well. We believe that only minor additions, such as the depth offset technique of [de Aguiar et al. 2010] and the optimization technique in [Guan et al. 2012], are enough to completely remove the visual artifacts for rare deep penetrations.

Finally, our model requires a decaying factor to produce smooth clothing deformation results under rapid pose changes. It might make the motion of clothing lag behind the body motion. This artifact is difficult to be completely eliminated through our blending method. We are considering to use our results as initial solutions of a fast simulation package to produce final deformation results to avoid the artifact.

9 Conclusion and Future Work

In this paper, we introduce real-time example based clothing synthesis using sensitivity-optimized rigging. At run time, our algorithm blends the predicted deformations to achieve physically-plausible clothing deformation. Coarse-level simulation is not needed in our case. Therefore, our run-time implementation is as simple as character skinning, which is ready to be integrated into games or interactive virtual reality applications.

In the future, we hope to integrate more control parameters, such as body physique, clothing pattern, and clothing material parameters, into our framework. Sensitivity analysis techniques should be applicable in this scenario to reduce the number of data points required in the database. We are also interested in extending our method to more general deformations, such as skin deformations considering underlying physical skeletal muscle movements. Employing our framework, we can compute complex deformations of skin quickly by optimizing the skin rigging weights and example deformations so that they agree with a detailed physics simulation of the skeletal system.

Acknowledgements

We would like to thank the anonymous reviewers for their constructive comments. The motion data used in our project is obtained from mocap.cs.cmu.edu. The database was created with funding from NSF EIA-0196217. Weiwei Xu is partially supported by NSFC (No.61272392, No.61322204) and the open project of the State Key Lab of CAD&CG(A1307). Xiaogang Jin was supported by NSFC (No.61272298, No.61328204) and National High-tech R&D Program (No. 2012AA011503) of China.

References

ANGUELOV, D., SRINIVASAN, P., KOLLER, D., THRUN, S., RODGERS, J., AND DAVIS, J. 2005. SCAPE: Shape completion and animation of people. *ACM Trans. Graph. 24*, 3, 408–416.

BARAN, I., AND POPOVIĆ, J. 2007. Automatic rigging and animation of 3d characters. *ACM Trans. Graph. 26*, 3.

BERGOU, M., MATHUR, S., WARDETZKY, M., AND GRINSPUN, E. 2007. TRACKS: toward directable thin shells. *ACM Trans. Graph. 26*, 3.

BICKEL, B., BÄCHER, M., OTADUY, M. A., LEE, H. R., PFISTER, H., GROSS, M., AND MATUSIK, W. 2010. Design and fabrication of materials with desired deformation behavior. *ACM Trans. Graph. 29*, 4, 63:1–63:10.

BRIDSON, R., FEDKIW, R., AND ANDERSON, J. 2002. Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph. 21*, 3, 594–603.

CARR, J. C., BEATSON, R. K., CHERRIE, J. B., MITCHELL, T. J., FRIGHT, W. R., MCCALLUM, B. C., AND EVANS, T. R. 2001. Reconstruction and representation of 3D objects with radial basis functions. In *Proc. of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, 67–76.

CHEN, Z., FENG, R., AND WANG, H. 2013. Modeling friction and air effects between cloth and deformable bodies. *ACM Trans. Graph. 32*, 4, 88:1–88:8.

CHOI, K.-J., AND KO, H.-S. 2002. Stable but responsive cloth. *ACM Trans. Graph. 21*, 3, 604–611.

CHOI, K.-J., AND KO, H.-S. 2005. Research problems in clothing simulation. *Computer-Aided Design 37*, 6, 585 – 592.

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

CORDIER, F., AND MAGNENAT-THALMANN, N. 2004. A data-driven approach for real-time clothes simulation. In *Computer Graphics and Applications, 2004. PG 2004. Proceedings. 12th Pacific Conference on*, 257–266.

DE AGUIAR, E., SIGAL, L., TREUILLE, A., AND HODGINS, J. K. 2010. Stable spaces for real-time clothing. *ACM Trans. Graph. 29*, 106:1–106:9.

DEROUET-JOURDAN, A., BERTAILS-DESCOUBES, F., AND THOLLOT, J. 2010. Stable inverse dynamic curves. *ACM Trans. Graph. 29*, 6, 137:1–137:10.

DIGEST, R. 2010. *The New Complete Guide to Sewing: Step-by-Step Techniques for Making Clothes and Home Accessories*

Updated Edition with All-New Projects and Simplicity Patterns (Reader's Digest). Readers Digest.

ENGLISH, E., AND BRIDSON, R. 2008. Animating developable surfaces using nonconforming elements. *ACM Trans. Graph. 27*, 3, 66:1–66:5.

FENG, W.-W., YU, Y., AND KIM, B.-U. 2010. A deformation transformer for real-time cloth animation. *ACM Trans. Graph. 29*, 4, 108:1–108:9.

GALLAGHER, R. H. 1973. *Optimum Structural Design: Theory and Applications*. John Wiley & Sons Inc.

GOVINDARAJU, N. K., KNOTT, D., JAIN, N., KABUL, I., TAMSTORF, R., GAYLE, R., LIN, M. C., AND MANOCHA, D. 2005. Interactive collision detection between deformable models using chromatic decomposition. *ACM Trans. Graph. 24*, 3, 991–999.

GRINSPUN, E., HIRANI, A. N., DESBRUN, M., AND SCHRÖDER, P. 2003. Discrete shells. In *Proc. of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '03, 62–67.

GUAN, P., REISS, L., HIRSHBERG, D. A., WEISS, A., AND BLACK, M. J. 2012. DRAPE: Dressing any person. *ACM Trans. Graph. 31*, 4, 35:1–35:10.

HARMON, D., VOUGA, E., TAMSTORF, R., AND GRINSPUN, E. 2008. Robust treatment of simultaneous collisions. *ACM Trans. Graph. 27*, 3, 23:1–23:4.

JAMES, D. L., AND FATAHALIAN, K. 2003. Precomputing interactive dynamic deformable scenes. *ACM Trans. Graph. 22*, 3, 879–887.

KALDOR, J. M., JAMES, D. L., AND MARSCHNER, S. 2008. Simulating knitted cloth at the yarn level. *ACM Trans. Graph. 27*, 3, 65:1–65:9.

KAVAN, L., COLLINS, S., ŽÁRA, J., AND O'SULLIVAN, C. 2008. Geometric skinning with approximate dual quaternion blending. *ACM Trans. Graph. 27*, 4, 105:1–105:23.

KAVAN, L., GERSZEWSKI, D., BARGTEIL, A. W., AND SLOAN, P.-P. 2011. Physics-inspired upsampling for cloth simulation in games. *ACM Trans. Graph. 30*, 4, 93:1–93:10.

KIM, T.-Y., AND VENDROVSKY, E. 2008. Drivenshape: a data-driven approach for shape deformation. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '08, 49–55.

KIM, D., KOH, W., NARAIN, R., FATAHALIAN, K., TREUILLE, A., AND O'BRIEN, J. F. 2013. Near-exhaustive precomputation of secondary cloth effects. *ACM Trans. Graph. 32*, 4, 87:1–87:8.

KRY, P. G., JAMES, D. L., AND PAI, D. K. 2002. Eigenskin: Real time large deformation character skinning in hardware. In *In ACM SIGGRAPH Symposium on Computer Animation*, ACM Press, 153–159.

LEWIS, J. P., CORDNER, M., AND FONG, N. 2000. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, SIGGRAPH '00, 165–172.

MERRELL, P., SCHKUFZA, E., LI, Z., AGRAWALA, M., AND KOLTUN, V. 2011. Interactive furniture layout using interior design guidelines. *ACM Trans. Graph. 30*, 4, 87:1–87:10.

MIGUEL, E., TAMSTORF, R., BRADLEY, D., SCHVARTZMAN, S. C., THOMASZEWSKI, B., BICKEL, B., MATUSIK, W., MARSCHNER, S., AND OTADUY, M. A. 2013. Modeling and estimation of internal friction in cloth. *ACM Trans. Graph. 32*, 6, 212:1–212:10.

MÜLLER, M., AND CHENTANEZ, N. 2010. Wrinkle meshes. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '10, 85–92.

MÜLLER, M., HEIDELBERGER, B., HENNIX, M., AND RATCLIFF, J. 2007. Position based dynamics. *Journal of Visual Communication and Image Representation 18*, 2, 109 – 118.

NARAIN, R., SAMII, A., AND O'BRIEN, J. F. 2012. Adaptive anisotropic remeshing for cloth simulation. *ACM Trans. Graph. 31*, 6, 152:1–152:10.

NEALEN, A., MÜLLER, M., KEISER, R., BOXERMAN, E., AND CARLSON, M. 2006. Physically based deformable models in computer graphics. *Computer Graphics Forum 25*, 4, 809–836.

PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. 2007. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, 3 ed. Cambridge University Press.

ROHMER, D., POPA, T., CANI, M.-P., HAHMANN, S., AND SHEFFER, A. 2010. Animation wrinkling: Augmenting coarse cloth simulations with realistic-looking wrinkles. *ACM Trans. Graph. 29*, 6, 157:1–157:8.

UMETANI, N., KAUFMAN, D. M., IGARASHI, T., AND GRINSPUN, E. 2011. Sensitive couture for interactive garment modeling and editing. *ACM Trans. Graph. 30*, 4, 90:1–90:12.

UMETANI, N., IGARASHI, T., AND MITRA, N. J. 2012. Guided exploration of physically valid shapes for furniture design. *ACM Trans. Graph. 31*, 4, 86:1–86:11.

VOLINO, P., MAGNENAT-THALMANN, N., AND FAURE, F. 2009. A simple approach to nonlinear tensile stiffness for accurate cloth simulation. *ACM Trans. Graph. 28*, 4, 105:1–105:16.

WANG, X. C., AND PHILLIPS, C. 2002. Multi-weight enveloping: Least-squares approximation techniques for skin animation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM, New York, NY, USA, SCA '02, 129–138.

WANG, R. Y., PULLI, K., AND POPOVIĆ, J. 2007. Real-time enveloping with rotational regression. *ACM Trans. Graph. 26*, 3.

WANG, H., HECHT, F., RAMAMOORTHI, R., AND O'BRIEN, J. 2010. Example-based wrinkle synthesis for clothing animation. *ACM Trans. Graph. 29*, 4, 107:1–107:8.

WEBER, O., SORKINE, O., LIPMAN, Y., AND GOTSMAN, C. 2007. Context-aware skeletal shape deformation. *Computer Graphics Forum (Proc. of EUROGRAPHICS) 26*, 3, 265–273.

WHITING, E., SHIN, H., WANG, R., OCHSENDORF, J., AND DURAND, F. 2012. Structural optimization of 3d masonry buildings. *ACM Trans. Graph. 31*, 6, 159:1–159:11.

ZHENG, C., AND JAMES, D. L. 2012. Energy-based self-collision culling for arbitrary mesh deformations. *ACM Trans. Graph. 31*, 4, 98:1–98:12.

ZURDO, J., BRITO, J., AND OTADUY, M. 2013. Animating wrinkles by example on non-skinned cloth. *Visualization and Computer Graphics, IEEE Transactions on 19*, 1, 149 –158.