Real-Time Human Pose and Shape Estimation for Virtual Try-On Using a Single Commodity Depth Camera

Mao Ye, *Student Member, IEEE*, Huamin Wang, *Member, IEEE*, Nianchen Deng,
Xubo Yang and Ruigang Yang, *Senior Member, IEEE*

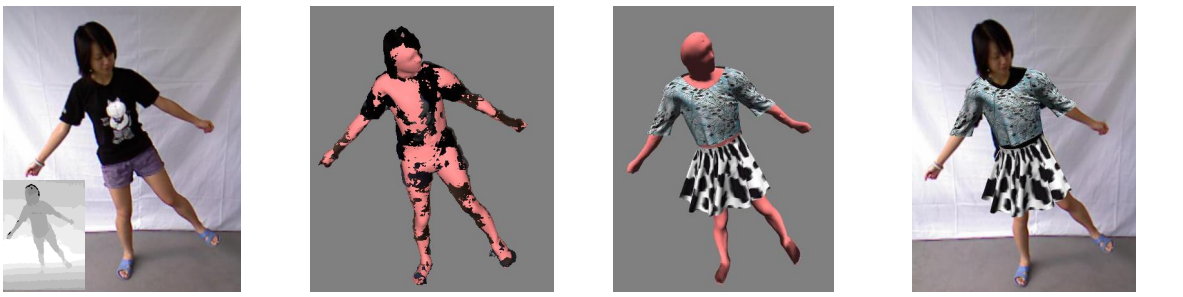


Fig. 1. Our virtual try-on system takes the RGBD data in (a) as input and captures the pose and body shape of the user accurately as shown in (b). It then realistically simulates virtual clothing on the user, so the user can examine the appearance of personalized virtual clothing, as illustrated in (c) and (d). Our system robustly handles a wide range of human motions and shape variations.

Abstract—We present a system that allows the user to virtually try on new clothes. It uses a single commodity depth camera to capture the user in 3D. Both the pose and the shape of the user are estimated with a novel real-time template-based approach that performs tracking and shape adaptation jointly. The result is then used to drive realistic cloth simulation, in which the synthesized clothes are overlayed on the input image. The main challenge is to handle missing data and pose ambiguities due to the monocular setup, which captures less than 50 percent of the full body. Our solution is to incorporate automatic shape adaptation and novel constraints in pose tracking. The effectiveness of our system is demonstrated with a number of examples.

Index Terms—Human pose estimation, human shape modeling, virtual try-on, depth sensor

1 Introduction

Commodity depth cameras, such as the Microsoft Kinect, have gained wide popularity in both academia and industry. With the capability to capture dynamic scenes in 3D, numerous researchers and developers are exploring new applications in many areas ranging from consumer electronics, entertainment, health care, to robotics. In this paper, we present how depth cameras can be used to improve the realism of virtual try-on systems, in which users can dress themselves up in different clothes in a virtual environment.

The concept of *virtual try-on*, due to its large commercial potential, has been explored before. The general idea is to track the user's motion, in either 2D or 3D [52, 43, 14], and synthesize clothes that can be overlayed on the user's image. Due to the complexity of human motion and the computational cost of cloth simulation, different systems have different trade-offs. Some treated virtual clothing as textures (e.g., [52]), over-simplifying the interactions between the user and the clothing; some required a pre-made avatar that is either quite crude or difficult to adapt to the user motion and shape (e.g., [14]).

We believe that an ideal virtual try-on system should realistically and efficiently simulate virtual clothing that reacts accurately to the user's body shape and motion. This aspect is especially crucial for users to correctly evaluate the appearance of different clothing on them, thus greatly improving their acceptance to such systems. Unfortunately, none of the existing systems satisfy all these constraints as far as we know. Using data acquired from commodity depth cameras in cloth simulation seems to be a natural solution to this problem. In fact, using depth maps to handle cloth-body collision is a well-studied GPU-based technique as shown in [23, 28, 20]. Unfortunately, the captured depth maps are often noisy and incomplete, with the frame rate limited by the hardware. One possible solution to this problem is to use multiple cameras, but camera synchronization is a challenging problem in the real world as no commodity depth camera currently supports inter-camera synchronization.

In this paper we present a virtual try-on system that requires only a single RGB+Depth stream for a realistic virtual try-on experience. The key to our system is a novel real-time tracking algorithm that jointly estimates a full-body shape and motion using a linear formulation. Requiring only a generic human body template, our formulation is able to handle significant occlusion in a single depth map ($\geq 50\%$ of missing data) as well as maintain temporal consistency from the noisy depth input generated by commodity depth cameras. The output of our tracking algorithm is a body mesh that accurately adapts to the user's motion and body shape. The mesh is complete and maintains the same topology over time, which makes it ideal for use in physically-based cloth simulation. As a result, our system is able to provide a more realistic virtual try-on experiences as the example in Fig. 1 shows.

The rest of this paper is organized as follows. In Sec. 2, we review

- *Mao Ye and Ruigang Yang are with Center for Visualization and Virtual Environments, University of Kentucky, Lexington, Kentucky, 40506. E-mail: mao.ye@uky.edu, ryang@cs.uky.edu.*
- *Huamin Wang is with Department of Computer Science and Engineering, The Ohio State University, Columbus, Ohio, 43210. E-mail: whmin@cse.ohio-state.edu.*
- *Nianchen Deng and Xubo Yang are with Digital Art Laboratory, School of Software, Shanghai Jiao Tong University, Shanghai, 200240, China. E-mail: dengnianchen@sjtu.edu.cn, yangxubo@sjtu.edu.cn.*

Manuscript received 12 September 2013; accepted 10 January 2014; posted online 29 March 2014; mailed on 1 May 2014.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

some of the state-of-the-art approaches that are related to our work. Sec. 3 describes the details of our methods for pose tracking and shape adaptation. The approaches used for our cloth simulation and final image composition are discussed in Sec. 4 and Sec. 5, respectively. Experiments are shown in Sec. 6, while conclusion and a discussion of future work are presented in Sec. 7.

2 Previous Work

Our virtual try-on system consists of two major components: pose tracking with shape adaptation and cloth simulation. In this section, we briefly review methods on both topics and recent research on virtual try-on applications.

2.1 Pose Estimation and Shape Adaptation

The problem of human pose estimation has been studied for decades [30, 34]. Most traditional approaches relied on multi-view setup, e.g. [3, 17]. More recently, there have been several studies using a single depth sensor to tackle this task [18, 36, 50, 2, 19, 48, 24]. The representative discriminative approach by Shotton et al. [36] trained a random forest classifier with a large collection of data and then used a clustering technique to estimate the joint positions. Despite of the fast running speed of this approach, the accuracy needs to be improved. More importantly, similar to many other discriminative approaches, it provided neither temporal coherence nor the subjects' body shape, which are important for cloth simulation in virtual try-on applications.

Generative approaches that inherently provide temporal correspondences are more suitable for our task. Ganapathi et al. [18] used Dynamic a Bayesian Network (DBN) to model the dynamics of motion states. However, their template size is fixed and cannot accommodate body size variations. Their more recent work [19] partially addressed this issue by allowing joint length to change within a small range. Furthermore, they utilized free space constraints to guide the tracker. Nonetheless, in order to achieve real-time performance, they used an over-simplified cylindrical mesh model which cannot realistically capture the observed surface geometry and could cause large visible artifacts for cloth simulation.

Some authors combined the complementary characteristics of discriminative and generative approaches. Ye et al. [50] and Baak et al. [2] both used a hybrid approach, yet they required a database at run time. Helten et al. [24] extended [2] with personalized shape estimation based on the SCAPE model [1]. Wei et al. [48] combined the DBN model in [18] and body detection in [36] for higher accuracy and robustness. While our tracking formulation is most similar to [24], we have enhanced the tracker with novel constraints that are capable of effectively and more accurately guiding the tracke. Consequently we do not require any discriminative detector compared to [24] and [48]. Besides, the discriminative detector in [36, 50, 2, 48, 24] tends to have trouble accommodating partial observation as part of the body moves out of view. By contrast, our tracker can effectively handle such situations with the the proposed constraints.

While discriminative approaches usually handle body size variation naturally, generative approaches generally need to explicitly consider the body size difference between the template model and the subject. Some methods assumed a good template that had a similar body size to the subject [18] or was scanned from the subject directly [17, 44, 11]; while some others performed fitting using parametric models [24]. The hybrid approaches usually relied on the discriminative component to address this issue [50, 48], which is not applicable as we do not use any detector. In our application, it is unnatural to assume prior knowledge of subject's body size. To this end, we propose a novel body size adaptation method based on the *differential bone coordinates* introduced by Straka et al. [38].

Besides the body size variations, the surface geometry of the template usually does not exactly match the subject's body shape. Therefore, surface estimation usually couples with or follows the pose estimation, in order to capture the surface geometry of the subject. One option is to perform a static scan of the subject's body shape with one [49] or multiple depth sensors [42]. For geometry estimation of dynamic scenes, silhouette has been widely used in both multi-view setups [17, 11, 44, 37, 38] and single-view setups [49]. Similar to [49, 16], we combine both depth and edge information to dynamically estimate the surface geometry. However, our method follows the linear formulation in [17] with novel constraints to deal with monocular data.

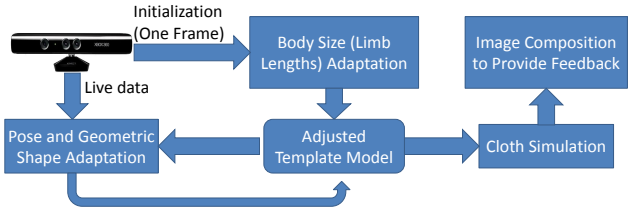


Fig. 2. The diagram of our virtual try-on system. During initialization, our system takes in one single frame of depth map and adjusts the limb lengths of the template. For each frame of the live data, we perform pose and shape adaptation to adjust the template that is then used by our cloth simulator to drive the virtual apparel. Finally a composite image of the virtual apparel and the input are delivered to the user to provide virtual try-on experience.

2.2 Cloth Simulation

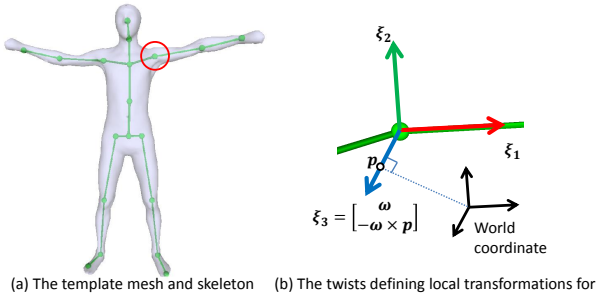
Recent research in physically based cloth simulation has resulted in a number of simulation approaches [4, 9, 6, 26] to improve the realism and efficiency of cloth animation. Survey articles [25, 8, 33] have summarized current state-of-the-arts. Among these methods, our work is particularly related to two types of novel simulation techniques.

Data-Driven Cloth. The first type contains data-driven techniques that combine synthetic or captured data with physically based simulation. Wang and his collaborators [46] developed a clothing wrinkle database and used human poses to guide fine wrinkle synthesis for clothing animation. De Aguiar and his colleagues [10] ignored physical models and proposed a purely data-driven model to efficiently generate wrinkle details instead. Feng and colleagues [13], as well as Kavan and collaborators [27], proposed data-driven models that can enrich coarse cloth simulation with fine details from data. While our data represents a human body rather than cloth, we think those techniques can further extend our work in the future, by combining data with synthetic or captured cloth data.

Cloth-Body Collision. The second type handles cloth-body collision by using synthetic depth maps. Due to their compatibility with the standard graphics pipeline, these techniques [23, 28, 20] can be accelerated by graphics hardware and are often used in GPU-based cloth simulation systems. For more accurate collision detection, 3D distance fields can also be efficiently constructed by GPU acceleration as Sud et al. [40] and Morvan et al. [31] demonstrated. Since these techniques considered depth maps (or distance fields) as intermediate representations from known body shapes, they could easily construct multiple depth maps from the input mesh model, and thus provide surface normal information to improve collision accuracy.

2.3 Virtual Try-On

Compared with the previous two problems, virtual try-on and personalized 3D garment design is a much less studied problem. Most existing systems treat virtual clothing as static texture patches and use image-based rendering techniques to virtually drive the cloth [21]. Many methods rely on a pre-captured database with subjects in a large variety of poses to find a best match and perform local refinement [12, 41, 52, 22]. While these methods, to a large extend, ignore



(a) The template mesh and skeleton (b) The twists defining local transformations for the joint circled in (a)

Fig. 3. Our skeletal template and its joint twists. The vector ω is the orientation of the rotation axis corresponding to $\boldsymbol{\xi}_i$.

the interaction between users and the clothes, some pioneered this area by combining real-world data with physically based cloth simulation. There are two main strategies to animate virtual clothing in a virtual try-on system. A straightforward and robust way is to create an avatar that has the same body shape as the user, and then simulate virtual clothing on it. The body size can either be specified by the user input [32, 15, 43], or using depth sensors [39]. While these techniques can accurately model virtual clothing on a static body shape, they cannot easily handle body motions. The triMirror system [43] simulated virtual clothing on a moving avatar, whose motion was controlled by the user's skeleton pose. However, as its result showed, their system seemed to use a pre-defined avatar which did not exactly match the user's body shape. Alternatively, it is preferable to obtain body shape from depth data. One such example is the Fitnect [14] system. While it successfully animated part of the clothing by body motion, the rest still needed to be static. In addition, it only treated clothes as a piece of cloth in front of the user, and it had difficulty in forcing the clothes to follow body motion exactly. Compared to the existing methods, our system can effectively capture the pose and shape of the user, and provide realistic cloth simulation.

3 Template-based Pose Tracking and Surface Fitting

For the virtual try-on application, it is important to accurately capture both the pose and the shape of the subject in order to realistically simulate the clothing. Our tracking method is most similar to [16] and [24], using a twist-based pose representation. However, we propose a set of constraints that are experimentally found to be effective for handling monocular data and therefore improve the robustness of the tracker. Moreover, our linearization of the exponential map better satisfies the small quantity requirement for first-order approximation compared to [16] and leads to linear optimization compared to the nonlinear one in [24]. There are three components in our tracking system: twist-based pose estimation, surface adaptation and body size estimation during initialization. In the following sections, we first describe our template and the twist-based deformation model, then our pose estimation method, and finally the shape and body size adaptation.

3.1 Twist-based Deformation Model

Our template model consists of four elements: surface vertices $\mathcal{V} = \{\mathbf{v}_i^0 | i = 1, \cdots, m\}$, surface connectivity $\mathcal{F}$, skinning weights $\mathcal{A}$ and an underlying skeleton $\mathcal{J}$ with 19 joints. The surface mesh and skeleton are shown in Fig. 3. The joints of the skeleton form a tree structure, which is called the kinematic tree. In general, a pose is defined as the 3D transformations of the joints. Similar to [5, 17], we represent a 3D transformation T via an exponential map:

$$T = e^{\hat{\boldsymbol{\xi}}\theta} = \sum_{k=0}^{\infty} \frac{\hat{\boldsymbol{\xi}}^k}{k!} \tag{1}$$

Here $\hat{\boldsymbol{\xi}}$ is the 4×4 matrix form of the twist $\boldsymbol{\xi}$ (a six dimensional vector representing the location and orientation of the rotation axis), while θ is the angle of rotation around $\boldsymbol{\xi}$. The advantage of this representation is its simplicity in linearizing the transformation with respect to the rotation angle θ, which leads to linear optimization for pose tracking.

Given the template model in a reference pose, the set of rotation axes are pre-defined, represented as the set of twists $\Xi = \{\boldsymbol{\xi}_k | k = 1, \cdots, n\}$. Therefore, a pose ($\Theta$) of the model is defined by the set of joint angles $\{\theta_k | k = 1, \cdots, n\}$ corresponding to each of the twists, as well as the global transformation $\boldsymbol{\xi}_g$. In practice, one joint might have more than one degree of freedoms, therefore is related to more than one joint twists and angles. For simplicity, we can treat each $< \boldsymbol{\xi}_i, \theta_i >$ as a joint. Without lost of generality, we assume the indexes of parent nodes in the kinematic tree are smaller than those of their children.

With an articulated motion model, the position of a vertex under pose Θ is obtained through the product of the exponential maps:

$$\mathbf{v}_i(\Theta) = e^{\hat{\boldsymbol{\xi}}_g} \prod_{k=1}^{n} e^{\delta_{k,b(i)} \hat{\boldsymbol{\xi}}_k \theta_k} \mathbf{v}_i^0 \tag{2}$$

where $b(i)$ denotes the index of the bone in the kinematic tree that vertex i belongs to. The indicator function $\delta_{p,q}$ equals to 1 if joint p is the same as q or is one of the ancestors of joint q in the kinematic tree. (Note that in this paper, for notation simplicity, the vector $\mathbf{v}$ could represent a homogeneous or an inhomogeneous coordinate, whichever is appropriate depending on the context.) For sequential motions, if the change in pose is relatively small, we can approximate the movement of a vertex with first order linearization as follows (see Appendix A):

$$\mathbf{v}_i(\Theta^{t+1}) \approx \mathbf{v}_i(\Theta^t) + I_i^t \Delta\boldsymbol{\xi}_g^t + \sum_{k=1}^{n} \delta_{k,b(i)} \hat{\boldsymbol{\xi}'}_k^t \mathbf{v}_i(\Theta^t) \Delta\theta_k^t \tag{3}$$

The definition of I_i^t can be found in Eq. 33. The vector $\Delta\boldsymbol{\xi}_g^t$ and the quantities $\{\Delta\theta_i^t\}$ are the changes of global transformation and local rotation angles, as defined in Eq. 25. The matrix $\hat{\boldsymbol{\xi}'}_k^t$ is the coordinate transformed twist (in matrix form) as defined in Eq. 31. Notice that this derivation is different from [16], as they directly linearize each exponential map around the rotation angle θ_k^t, which is not necessary small. However, Eq. 3 can be considered as the formulation in [5] extended to 3D.

As the deformation of human body is better modeled by Linear Blending Skinning (LBS) than articulated deformation, we further extend this formulation to skinning meshes. With LBS, the deformation of a vertex is represented as a linear combination of transformations from several controlling joints:

$$\mathbf{v}_i(\Theta) = \sum_{j=1}^{n} \alpha_{i,j} \Big(e^{\hat{\boldsymbol{\xi}}_g} \prod_{k=1}^{n} e^{\delta_{k,b(i)} \hat{\boldsymbol{\xi}}_k \theta_k} \Big) \mathbf{v}_i^0 \tag{4}$$

where $\{\alpha_{i,j}\}$ are the skinning weights in $\mathcal{A}$ which is one of the four elements of our template model. Eq. 3 can then be extended to this linear skinning model (see Appendix B):

$$\mathbf{v}_i(\Theta^{t+1}) \approx \mathbf{v}_i(\Theta^t) + I_i^t \Delta\boldsymbol{\xi}_g^t + \sum_{k=1}^{n} \beta_{i,k} \hat{\boldsymbol{\xi}'}_k^t \mathbf{v}_i(\Theta^t) \Delta\theta_k^t \tag{5}$$

$$\text{where} \quad \beta_{i,k} = \sum_{j=1}^{n} \delta_{k,j} \alpha_{i,j} \tag{6}$$

Here the weight $\beta_{i,k} \in [0, 1]$ reflects the influence of the joint k on the vertex i, by accumulating skinning weights from all the children of joint k in the kinematic tree.

3.2 Pose Tracking via Closest Points

The goal of pose tracking is to estimate the change in pose given the current configuration Θ^t and an observation of the surface vertices in

a new configuration Θ^{t+1}. Therefore, if the surface point correspondences between these two poses are given, we can use Eq. 5 and estimate the change of pose by minimizing the following energy function

$$E_c = \sum_{i=1}^{m} \left\| \omega_i^{t+1} \left(\mathbf{v}_i(\Theta^{t+1}) - \mathbf{c}_i^{t+1} \right) \right\|^2 \tag{7}$$

$$\approx \sum_{i=1}^{m} \left\| \omega_i^{t+1} \left(\mathbf{v}_i(\Theta^{t}) + I_i^t \Delta \boldsymbol{\xi}_g^t + \sum_{k=1}^{n} \beta_{i,k} \hat{\xi'}_k^t \mathbf{v}_i(\Theta^t) \Delta\theta_k^t - \mathbf{c}_i^{t+1} \right) \right\|^2 \tag{8}$$

where target surface point $\mathbf{v}_i(\Theta^{t+1})$ and its correspondence $\mathbf{c}_i^{t+1}$ are weighted by ω_i^{t+1}. Since $\mathbf{v}_i(\Theta^t)$ is known, the change of pose $\{\Delta\boldsymbol{\xi}_g^t, \Delta\theta_k^t\}$ can be obtained by solving this linear equation.

In reality, the true correspondences are not known a priori and therefore should be estimated. Similar to [16] and [24], we adopt the simple yet popular Iterative Closest Point (ICP) strategy. Consequently, Eq. 8 is solved iteratively with the point correspondences updated at each iteration. The strategies used to find correspondence are very important to the performance of the tracker. Rusinkiewicz et al. [35] conducted comparative studies of several alternatives for each component in the ICP framework for single rigid object registration. We also investigated these options and found that the following strategies worked well in our case:

1. 3D closest point for correspondence finding;
2. Distance and normal thresholding for correspondence pruning;
3. Rejection of correspondences containing edge points.

The projection based scheme used in [16] is found to be more sensitive to occlusions and therefore discarded. KD-tree is used for speeding up the closest point search.

In order to make our system more robust in accommodating monocular data, we additionally propose the following strategies:

1. Visibility constraint;
2. Relaxed bijective consistency constraint;
3. Edge-to-edge correspondences.

Notice that we first reject correspondences containing edge points following the suggestions in [35], and then explicitly construct edge-to-edge point correspondences. Such choices are designed to avoid correspondences between inner points and edge points that sometimes are problematic. A typical situation is when part of the body moves out of the view and false edges are created at the boundary. This constraint can avoid the matching of these points and thus help the tracker better deal with such partial observation. In the following, we will first discuss the first two constraints for pruning correspondences and then demonstrate how we use the edge-to-edge correspondences to better guide the tracker.

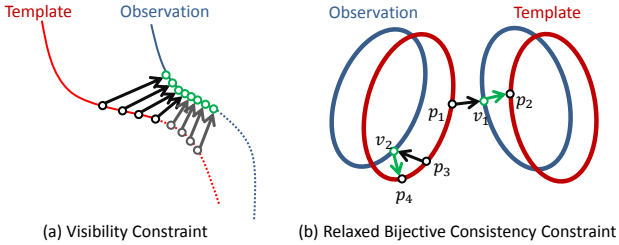
The visibility constraint means only visible points are used to construct point correspondences. On one hand, this strategy is important for dealing with monocular data by avoiding plenty of unnecessary computation and erroneous correspondences containing invisible points. Similar to the rejection of inner-to-edge correspondences mentioned above, the visibility constraint prevents the part of body that is out of camera view from causing errors. On the other hand, this constraint might reduce the power to handle large rotations, in which case the invisible points close to the visibility boundary are important for driving the surface towards its correct orientation. Therefore, we relax the constraint by including points close to the visibility boundary. More specifically, we render a depth map for the surface mesh and reject points whose projected depths are larger then the corresponding rendered depths by a certain threshold.

The relaxed bijective consistency constraint is designed to prevent the local closest point search from driving a surface vertex towards an observed point that is from another segment. Originally, the bijective consistency constraint requires a pair of points to be closest to each other among their own point sets to be considered as a correspondence [51]. However, this strictly bijective consistency constraint could be sensitive to noises and generally requires a relatively large number of iterations to converge. To overcome this limitation, we propose a *relaxed bijective consistency* that considers the consistency of joint belonging. Such constraint between a pair of points $< \mathbf{v}_i, \mathbf{c}_i >$ can be expressed as

$$b(i) \equiv b(f(\mathbf{c}_i)) \tag{9}$$

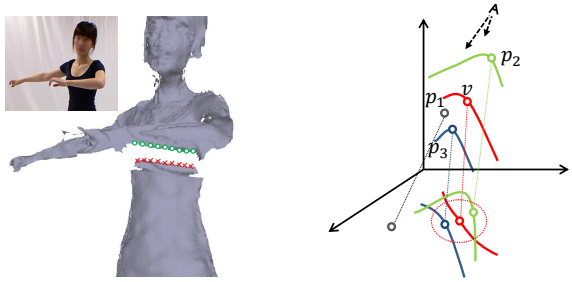
where $b(i)$ denotes the index of the joint that vertex i belongs to, and $f(\mathbf{c}_i)$ represents the index of the closest point of $\mathbf{c}_i$ on the surface mesh. This constraints requires that the closest point of the observation $\mathbf{c}_i$ on the surface mesh and the given surface vertex $\mathbf{v}_i$ belong to the same body segment. One could further set distance or normal thresholds between these two points; although we do not find this critical as evidenced by experimental observations. For our skinning mesh, we define $b(i)$ as the joint with maximum skinning weight for vertex i. Fig. 4 illustrates how the proposed constraint can prevent incorrect matching situations while still preserving effectiveness in identifying correct ones.

Besides the correspondences between surface points, we further enforce edge constraints to guide the tracker, similar to [17, 16]. First of



(a) Visibility Constraint (b) Relaxed Bijective Consistency Constraint

Fig. 4. 2D examples illustrating the effectiveness of two of our constraints. The arrows connect a point to its closest point on the other surface. In (a), the dash lines representing occluded regions. The visible points (black circles) attempt to move to the right as desired; while the invisible points (the gray circles) tend to move the surface upward and could lead to incorrect local optima if not excluded. In (b), since p_1 and p_2 belong to two different segments, our relaxed bijective consistency constraint will reject $< p_1, v_1 >$, so will the original version. However, the pair $< p_3, v_2 >$ will also be rejected by the original version. By contrast, our relaxed version will accept it, and therefore, could guide the tracker more effectively.



(a) True (green) and false (red) edge points (b) Advantage of our two stage edge correspondence search

Fig. 5. Edge point correspondences. (a) The false edge points due to self-occlusion are not used as they do not correspond to true shape boundary. (b) The dash lines represents projection, and the red dash circle indicates the range of candidate points to match v based on 2D measurement. With closest point in 3D, v is matched to a noise p_1. With closest point in 2D, it is matched to p_2. With our two stage approach, it can be mapped to the point p_3 on the correct target. However, it cannot handle the situation that projection of p_1 or p_2 lies in the red circle. In most cases, it is better than direct 3D or 2D closest point strategy.

all, edge points are extracted from the depth maps of the observed surface as well as our surface mesh. Edges that are not on the silhouette boundary are mainly due to self-occlusion. For these edge points, we only keep those points from the closer part, and ignore the occluded part as they do not correspond to real surface boundary, as shown in Fig. 5(a). For each of the template edge points, we find k_s nearest points in the 2D image plane, and keep the one closest in 3D space for reasons illustrated in Fig. 5(b). The distance and normal thresholding are again applied. However, in this case, the normals are 2D normals computed from the edge contour, instead of 3D vertex normals that are generally inaccurate for edge points of the observed surface.

With a set of edge point correspondences $\{< \mathbf{v}_i(\Theta^t), \mathbf{c}_i^{t+1} > | i \in \mathcal{E}^t\}$ extracted, the edge constraint requires the projections of the surface points to lie on the projections of their correspondences, denoted as $\{< u(\mathbf{c}_i^{t+1}), v(\mathbf{c}_i^{t+1}) > | i \in \mathcal{E}^t\}$. Given the projection matrix P, the edge constraint can be formulated as [17]

$$\begin{bmatrix} \mathbf{p}_r^1 - u(\mathbf{c}_i^{t+1}) \cdot \mathbf{p}_r^3 \\ \mathbf{p}_r^2 - v(\mathbf{c}_i^{t+1}) \cdot \mathbf{p}_r^3 \end{bmatrix} \cdot \mathbf{v}_i(\Theta^t) = \begin{bmatrix} u(\mathbf{c}_i^{t+1}) \cdot \mathbf{p}_t^3 - \mathbf{p}_t^1 \\ v(\mathbf{c}_i^{t+1}) \cdot \mathbf{p}_t^3 - \mathbf{p}_t^2 \end{bmatrix}, \quad i \in \mathcal{E}^t \tag{10}$$

where $\mathbf{p}_r^i$ is the i^{th} row of the first 3×3 sub-matrix of P, and $\mathbf{p}_t^i$ is the i^{th} element of the fourth column of P. For simplicity, we denote the 2×3 matrix on the left-hand side as H_i^{t+1} and the 2D vector on the right-hand side as $\mathbf{h}_i^{t+1}$. Since we want the vertex positions under a new pose Θ^{t+1} to satisfy this constraint, we can combine Eq. 5 and Eq. 10 to form the edge energy function:

$$E_e = \sum_{i \in \mathcal{E}^t} \left\| \tau_i^{t+1} \left(H_i^{t+1} (\mathbf{v}_i(\Theta^t) + I_i^t \cdot \Delta \boldsymbol{\xi}_g^t + \sum_{k=1}^{n} \beta_{i,k} \hat{\xi'}_k^t \mathbf{v}_i(\Theta^t) \Delta\theta_k^t) - \mathbf{h}_i^{t+1} \right) \right\|^2 \tag{11}$$

Our final tracking energy function consists of a weighted combination of Eq. 8 and Eq. 11, as well as a regularization term that penalizes large pose change which is required according to the first order linearization:

$$E = E_s + \lambda_e E_e + \lambda_r (\|\Delta \boldsymbol{\xi}_g^t\|^2 + \sum_{k=1}^{n} \|\Delta\theta_k^t\|^2) \tag{12}$$

The relative weight λ_r is set to 1, and λ_e is set as the inverse of the maximum of input depth map size in our experiments to avoid dependency on image size. For each new frame, this energy is iteratively optimized until the maximum movement of visible surface points is smaller than a given threshold, which is empirically set to 3mm in our experiments. Table. 1 summarizes the set of constraints and the parameters used for point matching in our tracker. The per-vertex weights $\{w_i\}$ and $\{\tau_i\}$ are currently set to 1 for points that satisfy these constraints and 0 otherwise.

	Distance	Normal	Ignore boundary	Visibility	Relaxed bijective
Surface	200mm	60°	Yes	Yes	Yes
Edge	200mm	90°	No	N/A	Yes

Table 1. The constraints and parameters used for point correspondence construction.

3.3 Surface Geometry Adaptation

After the pose has been estimated, the surface model should be fitted to the shape of the observation to ensure consistency between the final simulated clothes and the shape of the subject. We utilize both the captured surface geometry and edge information for this task. Note that the edge information from depth data is richer than silhouettes from color images. We rely on the same energy functions as in our pose tracker, namely Eq. 8 and Eq. 11. In general, the surface vertices are allowed to move in any direction. However, we constraint the movement of a vertex to be along its normal direction to partially overcome the ambiguities in monocular data. In fact, this constraint can partially prevent over-fitting the template to deformations of clothes. Therefore we only need to estimate the magnitudes of the movement, i.e. the displacements $\{d_i\}$. The energy function due to point correspondences, with the updated pose, then becomes

$$E_c^{\text{fit}} = \sum_{i=1}^{m} \left\| \omega_i^{t+1} \left(\mathbf{v}_i(\Theta^{t+1}) + \mathbf{n}_i^{t+1} \cdot d_i^{t+1} - \mathbf{c}_i^{t+1} \right) \right\|^2 + \lambda_e^{\text{fit}} \sum_{i \in \mathcal{E}} \left\| \tau_i^{t+1} \left(H_i^{t+1} (\mathbf{v}_i(\Theta^{t+1}) + \mathbf{n}^{t+1} \cdot d_i^{i+1} - \mathbf{h}_i^{t+1}) \right) \right\|^2 \tag{13}$$

with H_i^{t+1} and $\mathbf{h}_i^{t+1}$ defined in Eq. 10. The values of the per-vertex weights $\{\omega_i^{t+1}\}$ and $\{\tau_i^{t+1}\}$ are the same as in the pose tracker. Notice the vertex normals $\{\mathbf{n}_i^{t+1}\}$ are always calculated from the original surface mesh without displacement and then rotated based on the current pose. This strategy is designed to prevent surface distortion due to accumulated drifting in a long sequence of motions. Since before the fitting stage, the template mesh has been aligned to the observation with the estimated pose, we use a smaller distance threshold (50mm) for all correspondences, as well as a smaller angular threshold (60°) for edge correspondences. Similarly the global weight λ_e^{fit} is set to 2 to favor the edge correspondences.

Since this fitting process is performed for each frame independently, the fitted surface mesh might suffer from temporal jittering. This will eventually produce visual artifacts in cloth simulation. Therefore, we add another three energy functions to enforce temporal and spatial smoothness. The first term penalizes the distortion of the surface geometry via vertex Laplacian coordinates [17]; while the other two terms directly minimize large displacement changes temporally and spatially, respectively. The energy functions are defined as:

$$E_l^{\text{fit}} = \sum_{i=1}^{m} \left\| \sum_{<i,j> \in \mathcal{F}} (\mathbf{n}_j^{t+1} \cdot d_j^{t+1}) - \mathbf{n}_i^{t+1} \cdot d_i^{t+1} \right\|^2 \tag{14}$$

$$E_t^{\text{fit}} = \sum_{i=1}^{m} \frac{1}{\sigma_i^{2^t}} (d_i^{t+1} - \mu_i^t)^2 \tag{15}$$

$$E_s^{\text{fit}} = \sum_{<i,j> \in \mathcal{F}} (d_i^{t+1} - d_j^{t+1})^2 \tag{16}$$

Notice the first term is equivalent to minimizing the discrepancy of the Laplacian coordinates of the surface vertex with and without considering the displacement. μ_i^t and $\sigma_i^{2^t}$ are the mean and variance of the estimated displacements for vertex i up to frame t. One could alternatively measure the difference of displacement magnitude between consecutive frames. However, it does not prevent the surface from being overfitted in the presence of severe occlusions. Instead, our energy function forces the displacements to stabilize after sufficient observations are made, while still allowing flexibility for regions that are not rigid via the variance-based weighting. Both the means and variances are updated in an online fashion.

Our final fitting energy is then the weighted combination of all the terms defined above:

$$E^{\text{fit}} = E_c^{\text{fit}} + \lambda_l^{\text{fit}} E_l^{\text{fit}} + \lambda_t^{\text{fit}} E_t^{\text{fit}} + \lambda_s^{\text{fit}} E_s^{\text{fit}} \tag{17}$$

with empirical settings of $\lambda_l^{\text{fit}} = 4, \lambda_t^{\text{fit}} = 6$ and $\lambda_s^{\text{fit}} = 2$. The term E_t is set to take effect only after a certain number of frames (20 in our experiments) so that the means and variances are better estimated. For each frame, after the displacements are estimated, they are incorporated in the skinning model for pose estimation for the next frame. As the shape consistency increases via this fitting process, the accuracy of the pose estimation is also improved. Fig. 6 shows two examples of adapting the shape of our template model to the observed meshes.

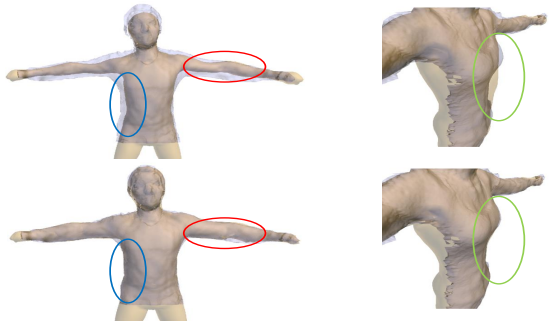


Fig. 6. A male (left) and a female (right) example showing the surface estimation results. The first row shows the template without displacements overlayed on the input, while the second row shows the one with displacements. The highlighted regions demonstrate its effectiveness.

3.4 Body Size Adaptation

Generative approaches generally need to address the issue of body sizes variations between the template and the subject. The effectiveness of a pose tracker usually relies heavily on such consistency. Our method iterates between updating the pose and adjusting the limb lengths, similar to [19]. We utilize the differential bone coordinates introduced by Straka et al. [38]. While their method results in non-linear optimization when incorporating constraints for limb lengths (e.g body symmetry or fixed lengths for some limbs), our method is always linear.

The differential body coordinates is defined by Straka et al. [38] in a way similar to the Laplacian coordinates:

$$\boldsymbol{\eta}_i = \sum_{k=1}^{n} \alpha_{i,k}\Big(\mathbf{v_i} - (\mathbf{g}_k - (1-\gamma_{i,k})\mathbf{d}_k)\Big) \tag{18}$$

where $\{\alpha_{i,k}\}$ are the skinning weights, $\mathbf{g}_k$ and $\mathbf{d}_k$ are the position of joint k and the vector from its parent joint to $\mathbf{g}_k$, respectively. The coefficient $\gamma_{i,k}$ is chosen such that the line between $\mathbf{v}_i$ and $\boldsymbol{\eta}_{i,k}$ is orthogonal to the bone vector $\mathbf{d}_k$. Each vector $\boldsymbol{\eta}_{i,k}$ encodes the relative position of the vertex $\mathbf{v}_i$ to bone k, while the differential bone coordinate $\boldsymbol{\eta}_i$ accumulates over all the controlling bones of this vertex.

To adapt the limb lengths of our template model to the observation in pose Θ^t, we estimate a scale for each limb so that the scaled surface mesh best matches the observation. With the set of scales $\mathcal{S} = \{s_k\}$, the bone vectors under the given pose then become $\{s_k\mathbf{d}_k^t\}$ and the joint positions can be computed as

$$\mathbf{g}_k(\Theta_t, \mathcal{S}) = \mathbf{g}_r(\Theta^t) + \sum_{j=1}^{n} \delta_{j,k} s_j \mathbf{d}_j^t \tag{19}$$

where $\mathbf{g}_r(\Theta^t)$ is the global position (position of the root) in pose Θ^t. According to Eq. 18, we can reconstruct the scaled vertex positions $\{\mathbf{v}_i(\Theta^t, \mathcal{S})\}$ as follows

$$\mathbf{v}_i(\Theta^t, \mathcal{S}) = \boldsymbol{\eta}_i + \sum_{k=1}^{n} \alpha_{i,k}\Big(\mathbf{g}_k(\Theta^t, \mathcal{S}) - (1-\gamma_{i,k}) s_k \mathbf{d}_k^t\Big) \tag{20}$$

The consistency of the scaled surface mesh and the observation is again measured via the distance between point correspondences. Specifically, we use the strategies described in Sec. 3.3 to construct point correspondences, and then minimize the energies in Eq. 8 and Eq. 11 parametrized on the scales:

$$E_c^s = \sum_{i=1}^{m} \Big\| \omega_i\Big(\mathbf{v}_i(\Theta^t, \mathcal{S}) - \mathbf{c}_i^t\Big)\Big\|^2 + \lambda_e^s \sum_{i\in\mathcal{E}} \Big\| \tau_i\Big(H_i^t \mathbf{v}_i(\Theta^t, \mathcal{S}) - \mathbf{h}_i^t\Big)\Big\|^2 \tag{21}$$

By substituting Eq. 20 into Eq. 21, we obtain the following energy function (see Appendix C):

$$E_c^s = \sum_{i=1}^{m} \Big\| \omega_i\Big(\sum_{k=1}^{n} \rho_{i,k} s_k \mathbf{b}_k^t + \boldsymbol{\eta}_i + \mathbf{g}_r(\Theta^t) - \mathbf{c}_i\Big)\Big\|^2 + \lambda_e^s \sum_{i\in\mathcal{E}} \Big\| \tau_i\Big(H_i(\sum_{k=1}^{n} \rho_{i,k} s_k \mathbf{b}_k^t + \boldsymbol{\eta}_i + \mathbf{g}_r(\Theta^t)) - \mathbf{h}_i\Big)\Big\|^2 \tag{22}$$

where the weights $\{\rho_{i,k}\}$ are defined in Eq. 36.

In addition, we add a regularization term that enforces scale consistency between a set of pre-defined bone pairs $\mathcal{B} = \{< j,k >\}$. The consistency enforces symmetry, e.g. left leg vs. right leg, as well as connected bone consistency, e.g. left upper leg vs. left lower leg. Therefore our final scale estimation energy function can be written as:

$$E^s = E_c^s + \lambda_r^s \sum_{<j,k>\in\mathcal{B}} \upsilon_{j,k}(s_j - s_k)^2 \tag{23}$$

In our experiments, we set the weights $\{\upsilon_{j,k}\}$ to 1 for body symmetry constraints and 0.5 for the connected bone consistency. To perform the body size adaptation, our method iterates between the pose estimation and the scales estimation. However, we require each step to converge before switching to another one (see Algorithm 1). This procedure is more stable and effective than switching at each iteration. The body size fitting is performed only during initialization for each subject to prevent over-fitting in the tracking procedure, for example due to partial observation. Fig. 7 shows an example of the template scaled to fit to the observation. The pose in this figure (the calibration pose) is preferable for the scale estimation as it implicitly encodes the position of the real joints and therefore helps to provide correct estimates of the scales.

3.5 The Tracking Pipeline

For a new subject, our system starts with the body size adaptation and then moves to the regular tracking-fitting procedure. In general, both a global transformation and a rough initial pose should be given due to the local nature of our method. The initial global transformation can be estimated via principal axes if the entire body of the subject is observable; via other discriminative approaches [50, 36]; or simply via human interaction. We do not intend to address this general problem but instead assume the rough initial alignment is given. As for the local body pose, we assume the that subject starts with a T-pose. However, as long as it is not dramatically different from the starting pose of our template, our tracker is able to drive the template to the initial subject pose. In practice, a frontal view pose is preferable due to its relatively small ambiguities. The entire pipeline of our tracking system is summarized in Algorithm 1.

4 Body-Guided Clothing Simulation

After we capture the human motion from a sequence, we use them to guide clothing animation in physically based cloth simulators. Since physically based cloth simulation is known for its large computational cost, we need to find a good balance between the simulation quality and the computational cost. Our idea is to incorporate two simulators into our system: a realtime cloth simulator for interactive preview purposes and an offline cloth simulator to generate high-quality clothing animation. At the beginning, the user chooses the realtime simulation mode to quickly examine how clothing behaves when it is draped on his/her body. After that, if the user would like to check more clothing details, he/she can switch to the offline simulation mode, under which the system generates either a high-quality clothing shape under a static pose in a few seconds, or the whole high-quality clothing animation in minutes or even hours.

Data: Captured surface sequence, template model, calibration pose Θ^c and initial tracking pose Θ^0.
Result: Scales $\mathcal{S}$, body poses $\{\Theta^t\}$ and displacements $\{d_i^t\}$.
begin Body size adaptation
  Initialize the template with pose Θ^c and Scales $\mathcal{S}' = \{1\}$
  while *Scales $\mathcal{S}$ not converged* **do**
    Update template with scales $\mathcal{S}$
    while *Pose not converged* **do**
      Construct correspondences
      Estimate pose via Eq. 12
    end
    while *Scales $\mathcal{S}'$ not converged* **do**
      Construct correspondences
      Estimate scales via Eq. 23
    end
    Set $\mathcal{S} = \mathcal{S}'$
  end
end
Initialize the template with pose Θ^0 and scales $\mathcal{S}$
for *Each frame t* **do**
  while *Pose Θ^t not converged* **do**
    Construct correspondences
    Estimate pose via Eq. 12
  end
  Construct correspondences
  Estimate displacements via Eq. 17
  Update template with the displacement;
end

Algorithm 1: Our pose tracking pipeline.

Near Realtime simulation. Our system needs a fast cloth simulator to simulate clothing motion and cloth-body interaction in near real time and preferably in real time. A simple yet effective strategy is to use a coarse mesh to represent each clothing piece. Such low-resolution cloth simulation cannot generate high-quality details such as wrinkles and folds, but its efficiency allows the user to quickly examine the clothing in different poses. In addition, we choose a simple mass spring model to handle both in-plane and bending deformation of cloth. We use Hooke's law to model the spring stiffness and solve the whole simulation using an implicit time integrator.

Compared with dynamic simulation, a more challenging problem is how to handle collisions and friction in real time. To achieve that, we create a set of virtual depth maps from virtual camera views around the body, similar to the other realtime collision techniques proposed in [23, 28, 20]. In particular, one of the virtual cameras should be collocated with our actual camera, and its depth map needs to be in high resolution to enable accurate cloth-body collisions in that view for the user to observe later. We note that this depth map should also be created virtually using the adapted template mesh, rather than using the raw depth map input, which may contain noises and errors. Once the algorithm detects a cloth vertex sufficiently close to the body, it applies a position-based constraint to move it away from the body. For simplicity, we ignore self intersections and consider cloth-body collisions only. In practice, we found that self collisions are rare under low-resolution in many simulation cases.

In the real world, the friction between the clothing and the human body can be highly complex, especially when the body performs dramatic motions, such as stretching or kicking. Our previous experience shows that the use of Coulomb's law and a small set of frictional parameters, as in many other simulators, is often insufficient to produce the desired clothing effects. Specifically, the clothing can either be too "rough", which suppresses its movement over the human body, or too "smooth", which causes dramatic clothing deformation. Our simple solution is to introduce a number of anchor points where the clothing piece is attached to the body. For shirts, the anchors are typically the shoulders; for pants and skirts, the anchors are typically the waist. We implement the anchoring points by setting them as position-based constraints on the clothing, so that the clothing can follow the body properly when the body moves over time.

Offline simulation. To produce highly detailed clothing animation in our offline simulation, we use the implicit Finite Element Method (FEM) to simulate in-plane cloth dynamics and the hinge edge bending model proposed by Bridson and colleagues [7] to animate bending deformation. Our implicit FEM solver is extended from the one developed by Volino and collaborators [45], which takes both nonlinear tensile deformation and nonlinear shearing deformation into consideration. As a result, we can incorporate the real-world material properties from the cloth elasticity database developed by Wang and collaborators [47] into the simulation to produce physically accurate clothing deformation behavior. Similar to the solver proposed by Volino and collaborators [45], our solver is not fully linearized and it is not unconditionally stable. However, compared with explicit solvers (that require the time steps to be approximately 10^{-6}s), it can robustly use orders-of-magnitude larger time steps even when handling highly stiff woven fabrics, which are commonly used to make everyday clothing.

Given a sequence of topologically consistent meshes, the handling of the interaction between the clothing and the human body is a relatively well defined problem. Here we use Continuous Collision Detection (CCD) [6] to detect and remove both cloth-body collisions and self collisions of cloth. We do not need to consider self collisions of the body, which are not supposed to affect the clothing animation result anyway. To speed up collision handling, we implement collision culling by using a regular grid data structure for spatial partitioning. Since the time step used by our simulator is small already, there is no need to use sub steps for collision handling. In practice, we run collision handling every three or four time steps, which is approximately $\frac{1}{900}$s. Similar to the realtime simulator, our offline simulator uses Coulomb's law to model cloth friction and sets a group of anchor points to prevent clothing from sliding a large amount.

5 Final Image Composition

The core of a virtual try-on system is the capability of providing visual feedbacks to the users. To achieve this, our system combines simulated cloth with the captured image data, and then displays them on a monitor. Correct visibility is the key to producing realistic try-on results. We found that this process can be easily done by using a two-pass rendering method under the basic OpenGL rendering pipeline. To begin with, we set the OpenGL projection matrix to be the same as the projection matrix of the depth sensor. During the first pass, we recover 3D locations of captured pixels according to their depth values, and we draw them as a combination of the human body and the background, using captured images as textures in OpenGL. In the second pass, we simply draw the simulated cloth, and OpenGL takes care of the visibility test using its depth buffer. This process is easy to implement and runs in real time (>100Hz).

6 Results and Discussions

We tested our system on an Intel Core i5-2500K 3.3GHz CPU with an NVIDIA Tesla C2075 GPGPU processor. Our input data are captured using a single Kinect camera with a resolution of 640×480 and sampling frequency of 30FPS . As our tracker does not require the entire body to be observed, except for the body size adaptation step, there is no specific restriction on the camera position and orientation. However, even though our tracking algorithm can deal with some noise, we do assume the input data are segmented and only the part of the surfaces belonging to the subject is provided to our tracker. Assuming the camera is fixed during data acquisition, we fit a plane to the ground and remove points that are close to the plane by some threshold (20mm for our test data). The background can be removed via depth thresholding in simple scenario. In our setup, we put a curtain on the background and also use plane fitting to remove the background points, yet with

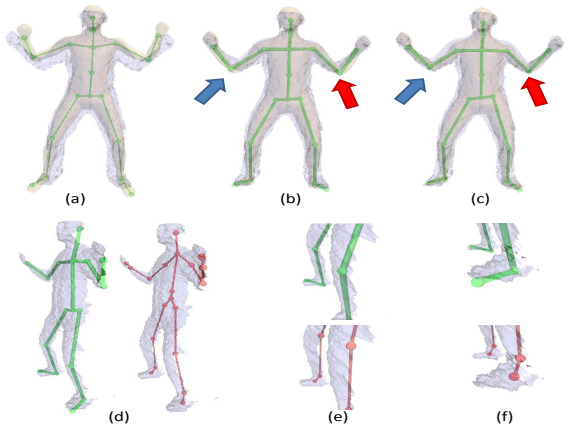


Fig. 7. An example showing results of our body size adaptation algorithm. In the first row, the input mesh, template mesh and template skeletons are shown together. (a) the initial step of the algorithm. (b) results with only pose estimation. (c) the adapted results. Notice the improvement for the joints indicated by the arrows. The second row compares our adapted skeleton, with the skeleton estimated by Kinect SDK [29]. (d) our skeleton (green) and the Kinect skeleton side by side. (e) and (f) comparisons of these two skeletons with close-up views for the legs and the feet, respectively. Discussions are provided in Sec. 6.1.

a larger distance threshold (200mm). In the rest of this section, we discuss the performance of both our pose tracker and cloth simulation components, and then show some final composite results.

6.1 Pose Tracker Performance

The template mesh model we use contains 12894 vertices. The number of vertices in the segmented input data is generally between 20K and 45K as the distance of the subject from the camera changes. Our tracking system is implemented on the GPU using CUDA and processes 40 to 60 frames per second, depending on the degree of change in pose, i.e. speed of motion. The body size adaptation usually takes less than 100ms to converge and is only performed during initialization. The overall tracking quality is shown in our supplemental video.

In Fig. 6, we show two representative examples of our surface adaptation results. Notice that when the subject is far away from the camera, and the captured geometry does not reflect the true body shape, our method will only attempt to fit to the observation and cannot overcome this limitation of the sensor. In our accompanied video, the consistency of the adapted template and the captured surface can be visually checked, which we believe is in general sufficient for our application.

Similarly, our body size adaptation algorithm can effectively capture the user's body size as shown in Fig. 7. Our pose tracker can correctly align the template and the observation in the calibration pose (from (a) to (b)), despite of the actual pose difference. Secondly, as highlighted with the arrows in Fig. 7(b) and (c), the refined arm joints are closer to their true positions, meaning the arms are properly scaled to fit to the user's body size. The second row compares our results with the skeletons estimated by the Kinect SDK [29]. As we can see from the close-up views in (e) and (f), our estimated joints more properly reflect the true joint locations, for example of the feet.

6.2 Cloth Simulation Performance

Offline Performance. The clothing meshes used in our offline cloth simulator contain 20K to 50K vertices. The dynamic simulation time step is typically 1/3600s and collisions are handled every five time steps. For most examples, each frame takes approximately one minute to simulate. We note that collision detection and handling is typically the bottleneck in our simulator and it uses at least 70 percent of the total computational cost.

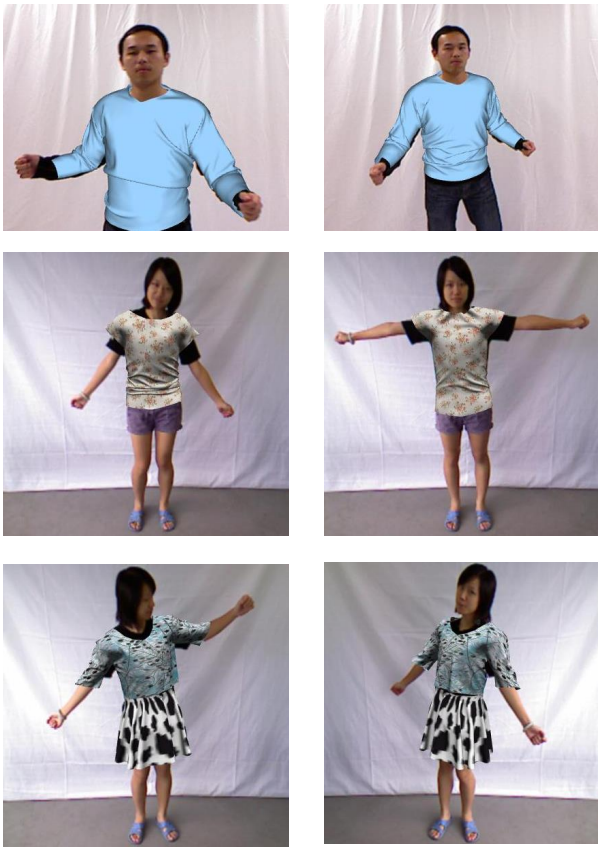


Fig. 8. Sample final results on both male and female subjects.

Near Real-time Performance. Our simulator is also able to run at an interactive rate of 8 to 12FPS, where the clothing meshes are down sampled to 1K to 2K vertices. Since our data capture system still captures image data at 30FPS, this means the human body cannot move arbitrarily fast. The computational cost of this simulator depends on the resolution of the clothing meshes and the resolution of virtual depth maps. We note that the result of the low-resolution simulation does not contain high-quality details as in the high-resolution simulation, even though it runs orders-of-magnitude faster.

6.3 Composite Results

Fig. 1 and Fig. 8 show some examples of our final results, that are provided to the user for a virtual try-on experience. As can be seen here, our system can effectively deal with both male and female subjects, partial and full body observations. Notice that we use the same template model for both cases, and rely on our shape adaptation method to captured the body shape of the user and eventually provide realistic results. The cloth simulation enables our system to deal with various type of clothes, for example the long sleeve, short sleeve and skirt (Fig. 8). Notice the realistic simulated wrinkles on the cloth. We believe such effects can provide a superior virtual try-on experience compared to many existing image-based methods. **More results are provided in our supplemental video**.

6.4 Limitations

Due to the inherent ambiguity for pose tracking using monocular data, as well as the local nature of our tracking method, our tracker could fail in some situations. One such case we observed is torso rotation while the body is only partially observed (Fig. 9). With such a severe occlusion, the point matching is ambiguous because the torso (the majority of the visible parts) is close to a cylindrical shape. To solve this issue, one option is to predict the pose with some motion model as in [17].

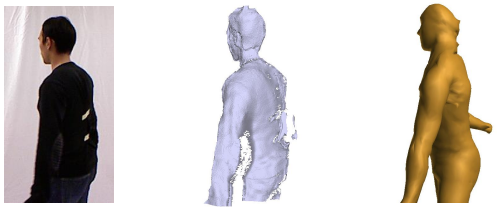


Fig. 9. A failure case for our pose tracking. The torso of the template (right) it not properly rotated to fit to the input surface (middle). In addition, the right arm is problematic as well due to lack of constraint.

Another limitation is the lack of constraints for invisible parts, as also can be seen from the right arm of the template in Fig. 9. Our visibility constraint can deal with it in most cases, however a more sophisticated approach might be to incorporate the free space constraint [19] into our framework.

7 Conclusion and Future Work

In this paper, we demonstrate the effectiveness of combining a real-time template-based joint pose and shape estimation, with physically-based cloth simulation in a virtual try-on system. With our proposed constraints for pose tracking as well as our novel method for shape adaptation, our system can effectively capture the motion and shape of the user, and then deliver realistic cloth simulation results to provide a virtual try-on experience.

The next step is to speed up the cloth simulation component in our system so that we can improve its performance and make it more practical for live demonstrations. We plan to further investigate into the limitations listed in Section 6.4, in order to better accommodate more challenging user poses and allow for a better user experience. Our long-term goal is to build a virtual try-on system, which is more efficient, accurate, robust, and convenient.

A Twist-Based Articulated Deformation Model

From Eq. 2, we have

$$\mathbf{v}_i(\Theta^{t+1}) = e^{\hat{\xi}_g^{t+1}} \prod_{k=1}^{n} e^{\delta_{k,b(i)}\hat{\xi}_k\theta_k^{t+1}} \mathbf{v}_i^0 \tag{24}$$

Let's denote the change in pose as:

$$e^{\hat{\xi}_g^{t+1}} = e^{\Delta\hat{\xi}_g^t} e^{\hat{\xi}_g^t}, \quad \theta^{t+1} = \theta^t + \Delta\theta^t. \tag{25}$$

With the assumption of small change of pose, we can use the following first-order approximation to linearize the exponential map:

$$e^{\Delta\hat{\xi}_g^t} \approx (I + \Delta\hat{\xi}_g^t), \quad e^{\delta_{k,b(i)}\hat{\xi}_k\theta_k^{t+1}} \approx e^{\delta_{k,b(i)}\hat{\xi}_k\theta_k^t}(I + \delta_{k,b(i)}\Delta\theta_k^t\hat{\xi}_k) \tag{26}$$

where I is the 4×4 identity matrix. Therefore, we can rewrite Eq.24 as:

$$\mathbf{v}_i(\Theta^{t+1}) \approx (I + \Delta\hat{\xi}_g^t)e^{\hat{\xi}_g^t} \prod_{k=1}^{n} e^{\delta_{k,b(i)}\hat{\xi}_k\theta_k^t}(I + \delta_{k,b(i)}\Delta\theta_k^t\hat{\xi}_k)\mathbf{v}_i^0 \tag{27}$$

Expand this product and ignore those with product of at least two terms of changes in angle, i.e. $\Delta\theta_j^t\Delta\theta_k^t$, we then get

$$\mathbf{v}_i(\Theta^{t+1}) \approx (I + \Delta\hat{\xi}_g^t)e^{\hat{\xi}_g^t}\Bigg[\prod_{k=1}^{n} e^{\delta_{k,b(i)}\hat{\xi}_k\theta_k^t} + \sum_{k=1}^{n}\delta_{k,b(i)}\Big(\prod_{j=1}^{k} e^{\delta_{j,k}\hat{\xi}_j\theta_j^t}\hat{\xi}_k\Delta\theta_k^t \prod_{j=k+1}^{n} e^{\delta_{j,b(i)}\hat{\xi}_j\theta_j^t}\Big)\Bigg]\mathbf{v}_i^0 \tag{28}$$

With our assumption that the index of a parent joint is smaller than those of its children, we can define two transformations M_b^t and T_b^t as

$$M_b^t = \prod_{k=1}^{n} e^{\delta_{k,b}\hat{\xi}_k\theta_k^t} \equiv \prod_{k=1}^{b} e^{\delta_{k,b}\hat{\xi}_k\theta_k^t}, \quad T_b^t = e^{\hat{\xi}_g^t}M_b^t \tag{29}$$

Therefore Eq. 28 becomes

$$\begin{aligned}
\mathbf{v}_i(\Theta^{t+1}) &\approx (I + \Delta\hat{\xi}_g^t)e^{\hat{\xi}_g^t}\Big[M_{b(i)}^t + \sum_{k=1}^{n}\delta_{k,b(i)}M_k^t\hat{\xi}_k(M_k^t)^{-1}M_{b(i)}^t\Delta\theta_k^t\Big]\mathbf{v}_i^0 \\
&= (I + \Delta\hat{\xi}_g^t)\Big[T_{b(i)}^t + \sum_{k=1}^{n}\Big(\delta_{k,b(i)}T_k^t\hat{\xi}_k(M_k^t)^{-1}(e^{\hat{\xi}_g^t})^{-1}e^{\hat{\xi}_g^t}M_{b(i)}^t\Delta\theta_k^t\Big)\Big]\mathbf{v}_i^0 \\
&= (I + \Delta\hat{\xi}_g^t)\Big[T_{b(i)}^t + \sum_{k=1}^{n}\Big(\delta_{k,b(i)}T_k^t\hat{\xi}_k(T_k^t)^{-1}T_{b(i)}^t\Delta\theta_k^t\Big)\Big]\mathbf{v}_i^0 \\
&= (I + \Delta\hat{\xi}_g^t)\Big[I + \sum_{k=1}^{n}\delta_{k,b(i)}\hat{\xi'}_k\Big]\mathbf{v}_i(\Theta^t)
\end{aligned} \tag{30}$$

The last equality is due to the fact that $\mathbf{v}_i(\Theta^t) = T_{b(i)}^t\mathbf{v}_i^0$ and the following definition of coordinate transformed twist:

$$\hat{\xi'}_k^t = T_k^t\hat{\xi}_k(T_k^t)^{-1} \tag{31}$$

Further expand Eq. 30 and ignore the higher order terms, we get

$$\begin{aligned}
\mathbf{v}_i(\Theta^{t+1}) &\approx \mathbf{v}_i(\Theta^t) + \Delta\hat{\xi}_g^t\mathbf{v}_i(\Theta^t) + \sum_{k=1}^{n}\delta_{k,b(i)}\hat{\xi'}_k^t\mathbf{v}_i(\Theta^t)\Delta\theta_k^t \\
&= \mathbf{v}_i(\Theta^t) + I_i^t\Delta\boldsymbol{\xi}_g^t + \sum_{k=1}^{n}\delta_{k,b(i)}\hat{\xi'}_k^t\mathbf{v}_i(\Theta^t)\Delta\theta_k^t
\end{aligned} \tag{32}$$

$$\text{where} \quad I_i^t = \begin{bmatrix} I & [\mathbf{v}]_\times \end{bmatrix} \tag{33}$$

and the operator $[]_\times$ converts a vector to a skew-symmetric matrix.

B Twist-Based Skinning Model

Substitute the result in Eq. 32 into Eq. 4, we get

$$\begin{aligned}
\mathbf{v}_i(\Theta^{t+1}) &\approx \sum_{j=1}^{n}\alpha_{i,j}\Big[\mathbf{v}_i(\Theta^t) + I_i^t\Delta\boldsymbol{\xi}_g^t + \sum_{k=1}^{n}\delta_{k,j}\hat{\xi'}_k^t\mathbf{v}_i(\Theta^t)\Delta\theta_k^t\Big] \\
&= \sum_{j=1}^{n}\alpha_{i,j}\mathbf{v}_i(\Theta^t) + \sum_{j=1}^{n}\alpha_{i,j}I_i^t\Delta\boldsymbol{\xi}_g^t + \sum_{j=1}^{n}\alpha_{i,j}\sum_{k=1}^{n}\delta_{k,j}\hat{\xi'}_k^t\mathbf{v}_i(\Theta^t)\Delta\theta_k^t \\
&= \mathbf{v}_i(\Theta^t) + I_i^t\Delta\boldsymbol{\xi}_g^t + \sum_{k=1}^{n}\Big(\sum_{j=1}^{n}\delta_{k,j}\alpha_{i,j}\Big)\hat{\xi'}_k^t\mathbf{v}_i(\Theta^t)\Delta\theta_k^t
\end{aligned} \tag{34}$$

Plug in the definition of the weights $\{\beta_{i,k}\}$ in Eq. 6, we get Eq. 5.

C Scales Optimization

Substitute Eq. 19 into Eq. 20, we have

$$\begin{aligned}
\mathbf{v}_i(\Theta^t, \mathcal{S}) &= \boldsymbol{\eta}_i + \sum_{k=1}^{n}\alpha_{i,k}\Big(\mathbf{g}_r(\Theta^t) + \sum_{j=1}^{n}\delta_{j,k}s_j\mathbf{d}_j^t - (1-\gamma_{i,k})s_k\mathbf{d}_k^t\Big) \\
&= \boldsymbol{\eta}_i + \mathbf{g}_r(\Theta^t) + \sum_{k=1}^{n}\alpha_{i,k}\sum_{j=1}^{n}\delta_{j,k}s_j\mathbf{d}_j^t - \sum_{k=1}^{n}\alpha_{i,k}(1-\gamma_{i,k})s_k\mathbf{d}_k^t\Big) \\
&= \boldsymbol{\eta}_i + \mathbf{g}_r(\Theta^t) + \sum_{j=1}^{n}\beta_{i,j}s_j\mathbf{d}_j^t - \sum_{k=1}^{n}\alpha_{i,k}(1-\gamma_{i,k})s_k\mathbf{d}_k^t
\end{aligned} \tag{35}$$

Define

$$\rho_{i,k} = \beta_{i,k} - \alpha_{i,k}(1-\gamma_{i,k}) \tag{36}$$

and plug it into Eq. 35 and then substitute the $\mathbf{v}_i(\Theta^t, \mathcal{S})$ in Eq. 21, we get Eq. 22.

Acknowledgments

We would like to thank Zhenyi He and Yang Cao for capturing data and creating video, and Anne Schmitz for proofreading the paper. This work is supported in part by US National Science Foundation grants IIS-1231545 and IIS-1208420 , and Natural Science Foundation of China (No.61173105,61373085, 61332017).

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., (Proc. of SIGGRAPH)*, pages 408–416. ACM, 2005.

[2] A. Baak, M. Müller, G. Bharaj, H.-P. Seidel, and C. Theobalt. A data-driven approach for real-time full body pose reconstruction from a depth camera. In *Int. Conf. on Comput. Vision*. IEEE, Nov. 2011.

[3] L. Ballan and G. M. Cortelazzo. Marker-less motion capture of skinned models in a four camera set-up using optical flow and silhouettes. In *3DPVT*, Atlanta, GA, USA, June 2008.

[4] D. Baraff and A. Witkin. Large steps in cloth simulation. In *ACM Trans. Graph., (Proc. of SIGGRAPH)*, pages 43–54. ACM, 1998.

[5] C. Bregler, J. Malik, and K. Pullen. Twist based acquisition and tracking of animal and human kinematics. *Int. J. Comput. Vision*, 56(3):179–194, Feb. 2004.

[6] R. Bridson, R. Fedkiw, and J. Anderson. Robust treatment of collisions, contact and friction for cloth animation. In E. Fiume, editor, *Proc. of ACM SIGGRAPH 2002*, volume 21, pages 594–603, 2002.

[7] R. Bridson, S. Marino, and R. Fedkiw. Simulation of clothing with folds and wrinkles. In *Proc. of SCA*, pages 28–36, 2003.

[8] K. Choi and H. Ko. Research problems in clothing simulation. *Computer Aided Design*, 37:585–592, 2005.

[9] K.-J. Choi and H.-S. Ko. Stable but responsive cloth. *ACM Transactions on Graphics (SIGGRAPH)*, 21(3):604–611, July 2002.

[10] E. de Aguiar, L. Sigal, A. Treuille, and J. K. Hodgins. Stable spaces for real-time clothing. *ACM Trans. Graph., (Proc. of SIGGRAPH)*, 29(4):106:1–106:9, July 2010.

[11] E. de Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H.-P. Seidel, and S. Thrun. Performance capture from sparse multi-view video. *ACM ToG*, 2008.

[12] J. Ehara and H. Saito. Texture overlay for virtual clothing based on pca of silhouettes. In *Mixed and Augmented Reality, 2006. ISMAR 2006. IEEE/ACM International Symposium on*, pages 139–142, 2006.

[13] W.-W. Feng, Y. Yu, and B.-U. Kim. A deformation transformer for real-time cloth animation. *ACM ToG (SIGGRAPH)*, 29(4), July 2010.

[14] Fitnect. Fitnect. http://www.fitnect.com, 2013.

[15] Fits.Me. Fits.Me. http://fits.me, 2012.

[16] J. Gall, A. Fossati, and L. Van Gool. Functional categorization of objects using real-time markerless motion capture. In *IEEE Conf. on Comput. Vision and Patt. Recog.*, 2011.

[17] J. Gall, C. Stoll, E. de Aguiar, C. Theobalt, B. Rosenhahn, and H.-P. Seidel. Motion capture using joint skeleton tracking and surface estimation. In *IEEE Conf. on Comput. Vision and Patt. Recog.*, pages 1746–1753. IEEE, 2009.

[18] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun. Real time motion capture using a single time-of-flight camera. In *IEEE Conf. on Comput. Vision and Patt. Recog.*, 2010.

[19] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun. Real-time human pose tracking from range data. In *European Conf. on Comput. Vision*, 2012.

[20] N. K. Govindaraju, S. Redon, M. C. Lin, and D. Manocha. Cullide: interactive collision detection between complex models in large environments using graphics hardware. In *Proc. of SGP*, pages 25–32, 2003.

[21] S. Hauswiesner, M. Straka, and G. Reitmayr. Temporal coherence in image-based visual hull rendering. *IEEE Transactions on Visualization and Computer Graphics*, 19(10):1758–1767, 2013.

[22] S. Hauswiesner, M. Straka, and G. Reitmayr. Virtual try-on through image-based rendering. *Visualization and Computer Graphics, IEEE Transactions on*, 19(9):1552–1565, 2013.

[23] B. Heidelberger, M. Teschner, and M. Gross. Realtime volumetric intersections of deforming objects. In *Proc. of Vision, Modeling and Visualization*, pages 461–468, 2003.

[24] T. Helten, A. Baak, G. Bharaj, M. Müller, H.-P. Seidel, and C. Theobalt. Personalization and evaluation of a real-time depth-based full body tracker. In *Proc. of the 3rd joint 3DIM/3DPVT Conference (3DV)*, 2013.

[25] D. House and D. Breen. *Cloth Modeling and Animation*. AK Peters, 2000.

[26] J. M. Kaldor, D. L. James, and S. Marschner. Simulating knitted cloth at the yarn level. *ACM ToG (SIGGRAPH)*, 27(3):65:1–65:9, August 2008.

[27] L. Kavan, D. Gerszewski, A. W. Bargteil, and P.-P. Sloan. Physics-inspired upsampling for cloth simulation in games. *ACM Transactions on Graphics (SIGGRAPH)*, 30(4):93:1–93:10, August 2011.

[28] D. Knott and D. K. Pai. Cinder: Collision and interference detection in real-time using graphics hardware. In *Proc. of Graphics Interface*, 2003.

[29] Microsoft. Kinect sdk. http://www.microsoft.com/en-us/kinectforwindows/, 2013.

[30] T. B. Moeslund, A. Hilton, and V. Krüger. A survey of advances in vision-based human motion capture and analysis. *CVIU*, 2006.

[31] T. Morvan, M. Reimers, and S. E. High performance GPU-based proximity queries using distance fields. In *Computer Graphics Forum*, volume 27, pages 2040–2052, december 2008.

[32] MVM. My Virtual Model. http://www.mvm.com/index.html, 2012.

[33] A. Nealen, M. Mueller, R. Keiser, E. Boxerman, and M. Carlson. Physically based deformable models in computer graphics. *Computer Graphics Forum*, 25(4):809–836, 2006.

[34] R. Poppe. A survey on vision-based human action recognition. *Image and Vision Computing*, 28(6):976 – 990, 2010.

[35] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Int. Conf. on 3D Digital Imaging and Modeling*, 2001.

[36] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *IEEE Conf. on Comput. Vision and Patt. Recog.*, Washington, DC, USA, 2011. IEEE.

[37] M. Straka, S. Hauswiesner, M. Ruther, and H. Bischof. Rapid skin: Estimating the 3d human pose and shape in real-time. In *3DIMPVT*, 2012.

[38] M. Straka, S. Hauswiesner, M. Rüther, and H. Bischof. Simultaneous shape and pose adaption of articulated models using linear optimization. In *European Conf. on Comput. Vision*, volume 7572, pages 724–737. 2012.

[39] Styku. Skytu. http://www.styku.com, 2012.

[40] A. Sud, N. Govindaraju, R. Gayle, and D. Manocha. Interactive 3D distance field computation using linear factorization. In *Proc. of I3D*, pages 117–124, 2006.

[41] H. Tanaka and H. Saito. Texture overlay onto flexible object with pca of silhouettes and k-means method for search into database. In *MVA*, 2009.

[42] J. Tong, J. Zhou, L. Liu, Z. Pan, and H. Yan. Scanning 3d full human bodies using kinects. *IEEE Transactions on Visualization and Computer Graphics*, 18:643–650, 2012.

[43] triMirror. triMirror. http://www.trimirror.com, 2013.

[44] D. Vlasic, I. Baran, W. Matusik, and J. Popović. Articulated mesh animation from multi-view silhouettes. In *ACM Trans. Graph., (Proc. of SIGGRAPH)*, pages 97:1–97:9, New York, NY, USA, 2008. ACM.

[45] P. Volino, N. Magnenat-Thalmann, and F. Faure. A simple approach to nonlinear tensile stiffness for accurate cloth simulation. *ACM Transactions on Graphics*, 28(4):105:1–105:16, September 2009.

[46] H. Wang, F. Hecht, R. Ramamoorthi, and J. O'Brien. Example-based wrinkle synthesis for clothing animation. *ACM Transactions on Graphics (SIGGRAPH)*, 29(4):107:1–, July 2010.

[47] H. Wang, J. O'Brien, and R. Ramamoorthi. Data-driven elastic models for cloth: Modeling and measurement. *ACM Transactions on Graphics (SIGGRAPH)*, 30(4):71:1–71:12, July 2011.

[48] X. Wei, P. Zhang, and J. Chai. Accurate realtime full-body motion capture using a single depth camera. *ACM Trans. Graph., (Proc. of SIGGRAPH Asia)*, 31(6):188:1–188:12, Nov. 2012.

[49] A. Weiss, D. Hirshberg, and M. J. Black. Home 3d body scans from noisy image and range data. In *Int. Conf. on Comput. Vision*, pages 1951–1958. IEEE, 2011.

[50] M. Ye, X. Wang, R. Yang, L. Ren, and M. Pollefeys. Accurate 3d pose estimation from a single depth image. In *Int. Conf. on Comput. Vision*, pages 731–738, 2011.

[51] M. Zeng, J. Zheng, X. Cheng, and X. Liu. Templateless quasi-rigid shape modeling with implicit loop-closure. In *IEEE Conf. on Comput. Vision and Patt. Recog.*, June 2013.

[52] Z. Zhou, B. Shu, S. Zhuo, X. Deng, P. Tan, and S. Lin. Image-based clothes animation for virtual fitting. In *SIGGRAPH Asia 2012 Technical Briefs*, SA '12, pages 33:1–33:4, New York, NY, USA, 2012. ACM.