SMPL: A Skinned Multi-Person Linear Model

Matthew Loper [*12] Naureen Mahmood[†1] Javier Romero[†1] Gerard Pons-Moll[†1] Michael J. Black[†1]

[1]Max Planck Institute for Intelligent Systems, Tübingen, Germany
[2]Industrial Light and Magic, San Francisco, CA

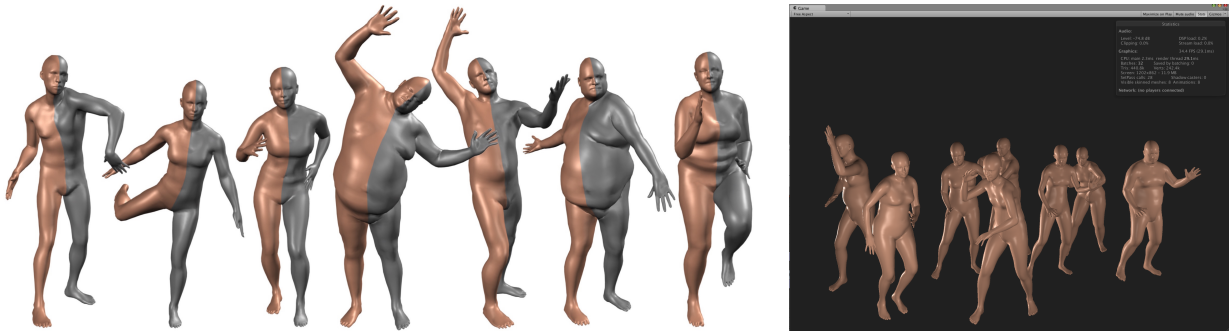


Figure 1: *SMPL is a realistic learned model of human body shape and pose that is compatible with existing rendering engines, allows animator control, and is available for research purposes. (left) SMPL model (orange) fit to ground truth 3D meshes (gray). (right) Unity 5.0 game engine screenshot showing bodies from the CAESAR dataset animated in real time.*

Abstract

We present a learned model of human body shape and pose-dependent shape variation that is more accurate than previous models and is compatible with existing graphics pipelines. Our Skinned Multi-Person Linear model (SMPL) is a skinned vertex-based model that accurately represents a wide variety of body shapes in natural human poses. The parameters of the model are learned from data including the rest pose template, blend weights, pose-dependent blend shapes, identity-dependent blend shapes, and a regressor from vertices to joint locations. Unlike previous models, the pose-dependent blend shapes are a linear function of the elements of the pose rotation matrices. This simple formulation enables training the entire model from a relatively large number of aligned 3D meshes of different people in different poses. We quantitatively evaluate variants of SMPL using linear or dual-quaternion blend skinning and show that both are more accurate than a BlendSCAPE model trained on the same data. We also extend SMPL to realistically model dynamic soft-tissue deformations. Because it is based on blend skinning, SMPL is compatible with existing rendering engines and we make it available for research purposes.

CR Categories: I.3.3 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

Keywords: Body shape, skinning, blendshapes, soft-tissue.

[*]e-mail:mloper@ilm.com
[†]e-mail:{nmahmood,jromero,gerard.pons.moll,black}@tue.mpg.de

1 Introduction

Our goal is to create realistic animated human bodies that can represent different body shapes, deform naturally with pose, and exhibit soft-tissue motions like those of real humans. We want such models to be fast to render, easy to deploy, and compatible with existing rendering engines. The commercial approach commonly involves hand rigging a mesh and manually sculpting blend shapes to correct problems with traditional skinning methods. Many blend shapes are typically needed and the manual effort required to build them is large. As an alternative, the research community has focused on learning statistical body models from example scans of different bodies in a varied set of poses. While promising, these approaches are not compatible with existing graphics software and rendering engines that use standard skinning methods.

Our goal is to automatically learn a model of the body that is both realistic and compatible with existing graphics software. To that end, we describe a "Skinned Multi-Person Linear" (SMPL) model of the human body that can realistically represent a wide range of human body shapes, can be posed with natural pose-dependent deformations, exhibits soft-tissue dynamics, is efficient to animate, and is compatible with existing rendering engines (Fig. 1).

Traditional methods model how vertices are related to an underlying skeleton structure. Basic linear blend skinning (LBS) models are the most widely used, are supported by all game engines, and are efficient to render. Unfortunately they produce unrealistic deformations at joints including the well-known "taffy" and "bowtie" effects (see Fig. 2). Tremendous work has gone into skinning methods that ameliorate these effects [Lewis et al. 2000; Wang and Phillips 2002; Kavan and Žára 2005; Merry et al. 2006; Kavan et al. 2008]. There has also been a lot of work on learning highly realistic body models from data [Allen et al. 2006; Anguelov et al. 2005; Freifeld and Black 2012; Hasler et al. 2010; Chang and Zwicker 2009; Chen et al. 2013]. These methods can capture the body shape of many people as well as non-rigid deformations due to pose. The most realistic approaches are arguably based on triangle deformations [Anguelov et al. 2005; Chen et al. 2013; Hasler et al. 2010; Pons-Moll et al. 2015]. Despite the above research, existing mod-

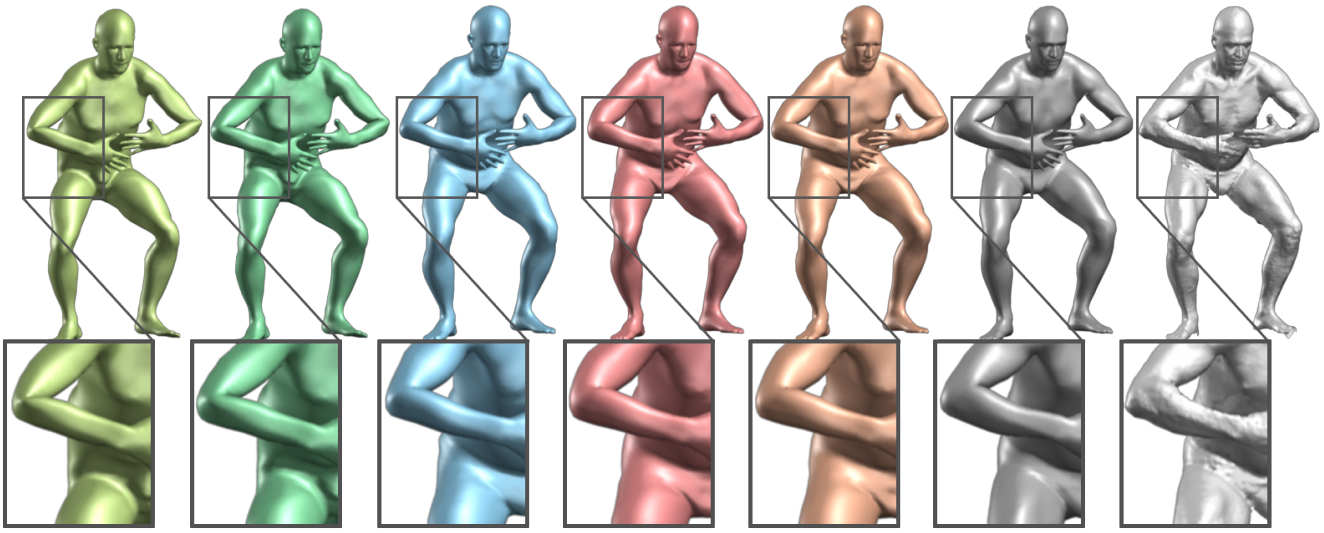


Figure 2: Models compared with ground truth. *This figure defines the color coding used throughout the paper and* **Supplemental Video**. *The far right (light gray) mesh is a 3D scan. Next to it (dark gray) is a registered mesh with the same topology as our model. We ask how well different models can approximate this registration. From left to right: (light green) Linear blend skinning (LBS), (dark green) Dual-quaternion blend skinning (DQBS), (blue) BlendSCAPE, (red) SMPL-LBS, (orange) SMPL-DQBS. The zoomed regions highlight differences between the models at the subject's right elbow and hip. LBS and DQBS produce serious artifacts at the knees, elbows, shoulders and hips. BlendSCAPE and both SMPL models do similarly well at fitting the data.*

els either lack realism, do not work with existing packages, do not represent a wide variety of body shapes, are not compatible with standard graphics pipelines, or require significant manual labor.

In contrast to the previous approaches, a key goal of our work is to make the body model as simple and standard as possible so that it can be widely used, while, at the same time, keeping the realism of deformation-based models learned from data. Specifically we learn blend shapes to correct for the limitations of standard skinning. Different blend shapes for identity, pose, and soft-tissue dynamics are additively combined with a rest template before being transformed by blend skinning. A key component of our approach is that we formulate the pose blend shapes as a linear function of the elements of the part rotation matrices. This formulation is different from previous methods [Allen et al. 2006; Merry et al. 2006; Wang and Phillips 2002] and makes training and animating with the blend shapes simple. Because the elements of rotation matrices are bounded, so are the resulting deformations, helping our model generalize better.

Our formulation admits an objective function that penalizes the per-vertex disparities between registered meshes and our model, enabling training from data. To learn how people deform with pose, we use 1786 high-resolution 3D scans of different subjects in a wide variety of poses. We align our template mesh to each scan to create a training set. We optimize the blend weights, pose-dependent blend shapes, the mean template shape (rest pose), and a regressor from shape to joint locations to minimize the vertex error of the model on the training set. This joint regressor predicts the location of the joints as a function of the body shape and is critical to animating realistic pose-dependent deformations for any body shape. All parameters are automatically estimated from the aligned scans.

We learn linear models of male and female body shape from the CAESAR dataset [Robinette et al. 2002] (approximately 2000 scans per gender) using principal component analysis (PCA). We first register a template mesh to each scan and pose normalize the data, which is critical when learning a vertex-based shape model. The resulting principal components become body shape blend shapes.

We train the SMPL model in various forms and compare it quantitatively to a BlendSCAPE model [Hirshberg et al. 2012] trained with exactly the same data. We evaluate the models both qualitatively with animations and quantitatively using meshes that were not used for training. We fit SMPL and BlendSCAPE to these meshes and then compare the vertex errors. Two main variants of SMPL are explored, one using linear blend skinning (LBS) and the other with Dual-Quaternion blend skinning (DQBS); see Fig. 2. The surprise is that a vertex-based, skinned, model such as SMPL is actually more accurate than a deformation-based model like BlendSCAPE trained on the same data. The test meshes are available for research purposes so others can quantitatively compare to SMPL.

We extend the SMPL model to capture soft-tissue dynamics by adapting the Dyna model [Pons-Moll et al. 2015]. The resulting Dynamic-SMPL, or DMPL model, is trained from the same dataset of 4D meshes as Dyna. DMPL, however, is based on vertices instead of triangle deformations. We compute vertex errors between SMPL and Dyna training meshes, transformed into the rest pose, and use PCA to reduce the dimensionality, producing dynamic blend shapes. We then train a soft-tissue model based on angular velocities and accelerations of the parts and the history of dynamic deformations as in [Pons-Moll et al. 2015]. Since soft-tissue dynamics strongly depend on body shape, we train DMPL using bodies of varying body mass index and learn a model of dynamic deformations that depends of body shape. Animating soft-tissue dynamics in a standard rendering engine simply requires computing the dynamic linear blend shape coefficients from the sequence of poses. Side-by-side animations of Dyna and DMPL reveal that DMPL is more realistic. This extension of SMPL illustrates the generality of our additive blend shape approach, shows how deformations can depend on body shape, and demonstrates how the approach provides a extensible foundation for modeling body shape.

SMPL models can be animated significantly faster than real time on a CPU using standard rendering engines. Consequently SMPL addresses an open problem in the field; it makes a realistic learned model accessible to animators. The SMPL base template is de-

signed with animation in mind; it has a low-polygon count, a simple vertex topology, clean quad structure, a standard rig, and reasonable face and hand detail (though we do not rig the hands or face here). SMPL can be represented as an Autodesk Filmbox (FBX) file that can be imported into animation systems. We make the SMPL model available for research purposes and provide scripts to drive our model in Maya, Blender, Unreal Engine and Unity.

2 Related Work

Linear blend skinning and blend shapes are widely used throughout the animation industry. While the research community has proposed many novel models of articulated body shape that produce high-quality results, they are not compatible with industry practice. Many authors have tried to bring these worlds together with varying degrees of success as we summarize below.

Blend Skinning. Skeleton subspace deformation methods, also known as blend skinning, attach the surface of a mesh to an underlying skeletal structure. Each vertex in the mesh surface is transformed using a weighted influence of its neighboring bones. This influence can be defined linearly as in Linear Blend Skinning (LBS). The problems of LBS have been widely published and the literature is dense with generic methods that attempt to fix these, such as quaternion or dual-quaternion skinning, spherical skinning, etc. (e.g. [Wang and Phillips 2002; Kavan and Žára 2005; Kavan et al. 2008; Le and Deng 2012; Wang et al. 2007]). Generic methods, however, often produce unnatural results and here we focus on learning to correct the limitations of blend skinning, regardless of the particular formulation.

Auto-rigging. There is a great deal of work on automatically rigging LBS models (e.g. [De Aguiar et al. 2008; Baran and Popović 2007; Corazza and Gambaretto 2014; Schaefer and Yuksel 2007]) and commercial solutions exist. Most relevant here are methods that take a collection of meshes and infer the bones as well as the joints and blend weights (e.g. [Le and Deng 2014]). Such methods do not address the common problems of LBS models because they do not learn corrective blend shapes. Models created from sequences of meshes (e.g. [De Aguiar et al. 2008]) may not generalize well to new poses and motions. Here, we assume the kinematic structure is known, though the approach could be extended to also learn this using the methods above.

The key limitation of the above methods is that the models do not span a space of body shapes. Miller et al. [2010] partially address this by auto-rigging using a database of pre-rigged models. They formulate rigging and skinning as the process of transferring and adapting skinning weights from known models to a new model. Their method does not generate blend shapes, produces standard LBS artifacts, and does not minimize a clear objective function.

Blend shapes. To address the shortcomings of basic blend skinning, the pose space deformation model (PSD) [Lewis et al. 2000] defines deformations (as vertex displacements) relative to a base shape, where these deformations are a function of articulated pose. This is the key formulation that is largely followed by later approaches and is also referred to as "scattered data interpolation" and "corrective enveloping" [Rouet and Lewis 1999]. We take an approach more similar to weighted pose-space deformation (WPSD) [Kurihara and Miyata 2004; Rhee et al. 2006], which defines the corrections in a rest pose and then applies a standard skinning equation (e.g. LBS). The idea is to define corrective shapes (sculpts) for specific key poses, so that when added to the base shape and transformed by blend skinning, produce the right shape. Typically one finds the distance (in pose space) to the exemplar poses and uses a function, e.g. a Radial Basis (RBF) kernel [Lewis et al. 2000], to weight the exemplars non-linearly based on distance. The sculpted blend shapes are then weighted and linearly combined.

These approaches are all based on computing weighted distances to exemplar shapes. Consequently, these methods require computation of the distances and weights at runtime to obtain the corrective blend shape. For a given animation (e.g. in a video game) these weights are often defined in advance based on the poses and "baked" into the model. Game engines apply the baked-in weights to the blend shapes. The sculpting process is typically done by an artist and then only for poses that will be used in the animation.

Learning pose models. Allen et al. [2002] use this PSD approach but rather than hand-sculpt the corrections, learn them from registered 3D scans. Their work focuses primarily on modeling the torso and arms of individuals, rather than whole bodies of a population. They store deformations of key poses and interpolate between them. When at, or close to, a stored shape, these methods are effectively perfect. They do not tend to generalize well to new poses, requiring dense training data. It is not clear how many such shapes would be necessary to model the full range of articulated human pose. As the complexity of the model increases, so does the complexity of controlling all these shapes and how they interact.

To address this, Kry et al. [2002] learn a low-dimensional PCA basis for each joint's deformations. Pose-dependent deformations are described in terms of the coefficients of the basis vectors. Kavan et al. [2009] use example meshes generated using a non-linear skinning method to construct linear approximations. James and Twigg [2005] combine the idea of learning the bones (non-rigid, affine bones) and skinning weights directly from registered meshes. For blend shapes they use an approach similar to [Kry et al. 2002].

Another way to address the limitations of blend skinning is through "multi-weight enveloping" (MWE) [Wang and Phillips 2002]. Rather than weight each vertex by a weighted combination of the bone transformation matrices, MWE learns weights for the *elements* of these matrices. This increases the capacity of the model (more parameters). Like [James and Twigg 2005] they overparameterize the bone transformations to give more expressive power and then use PCA to remove unneeded degrees of freedom. Their experiments typically involve user interaction and the MWE approach is not supported by current game engines.

Merry et al. [2006] find MWE to be overparameterized, because it allows vertices to deform differently depending on rotation in the global coordinate system. Their Animation Space model reduces the parameterization at minimal loss of representational power, while also showing computational efficiency on par with LBS.

Another alternative is proposed by Mohr and Gleicher [2003] who learn an efficient linear and realistic model from example meshes. To deal with the problems of LBS, however, they introduce extra "bones" to capture effects like muscle bulging. These extra bones increase complexity, are non-physical, and are non-intuitive for artists. Our blend shapes are simpler, more intuitive, more practical, and offer greater realism. Similarly, Wang et al. [2007] introduce joints related to surface deformation. Their rotational regression approach uses deformation gradients, which then must be converted to a vertex representation.

Learning pose and shape models. The above methods focus on learning poseable single-shape models. Our goal, however, is to have realistic poseable models that cover the space of human shape variation. Early methods use PCA to characterize a space of human body shapes [Allen et al. 2003; Seo et al. 2003] but do not model how body shape changes with pose. The most successful class of models are based on SCAPE [Anguelov et al. 2005] and represent body shape and pose-dependent shape in terms of triangle deforma-

tions rather than vertex displacements [Chen et al. 2013; Freifeld and Black 2012; Hasler et al. 2009; Hirshberg et al. 2012; Pons-Moll et al. 2015]. These methods learn statistical models of shape variation from training scans containing different body shapes and poses. Triangle deformations provide allow the composition of different transformations such as body shape variation, rigid part rotation, and pose-dependent deformation. Weber et al. [2007] present an approach that has properties of SCAPE but blends this with example shapes. These models are not consistent with existing animation software.

Hasler et al. [2010] learn two linear blend rigs: one for pose and one for body shape. To represent shape change, they introduce abstract "bones" that control the shape change of the vertices. Animating a character of a particular shape involves manipulating the shape and pose bones. They learn a base mesh and blend weights but not blend shapes. Consequently the model lacks realism.

What we would like is a vertex-based model that has the expressive power of the triangle deformation models so that it can capture a whole range of natural shapes and poses. Allen et al. [2006] formulate such a model. For a given base body shape, they define a standard LBS model with scattered/exemplar PSD to model pose deformations (using RBFs). They greedily define "key angles" at which to represent corrective blend shapes and they hold these fixed across all body shapes. A given body shape is parameterized by the vertices of the rest pose, corrective blend shapes (at the key angles), and bone lengths; these comprise a "character vector." Given different character vectors for different bodies they learn a low-dimensional latent space that lets them generalize character vectors to new body shapes; they learn these parameters from data. Their model is more complex than ours, has fewer parameters, and is learned from much less data. A more detailed analysis of how this method compares to SMPL is presented in Sec. 7.

3 Model Formulation

Our Skinned Multi-Person Linear model (SMPL) is illustrated in Fig. 3. Like SCAPE, the SMPL model decomposes body shape into identity-dependent shape and non-rigid pose-dependent shape; unlike SCAPE, we take a vertex-based skinning approach that uses corrective blend shapes. A single blend shape is represented as a vector of concatenated vertex offsets. We begin with an artist-created mesh with $N = 6890$ vertices and $K = 23$ joints. The mesh has the same topology for men and women, spatially varying resolution, a clean quad structure, a segmentation into parts, initial blend weights, and a skeletal rig. The part segmentation and initial blend weights are shown in Fig. 6.

Following standard skinning practice, the model is defined by a mean template shape represented by a vector of N concatenated vertices $\bar{\mathbf{T}} \in \mathbb{R}^{3N}$ in the zero pose, $\vec{\theta}^*$; a set of blend weights, $\mathcal{W} \in \mathbb{R}^{N \times K}$, (Fig. 3(a)); a blend shape function, $B_S(\vec{\beta}) : \mathbb{R}^{|\vec{\beta}|} \mapsto \mathbb{R}^{3N}$, that takes as input a vector of shape parameters, $\vec{\beta}$, (Fig. 3(b)) and outputs a blend shape sculpting the subject identity; a function to predict K joint locations (white dots in Fig. 3(b)), $J(\vec{\beta}) : \mathbb{R}^{|\vec{\beta}|} \mapsto \mathbb{R}^{3K}$ as a function of shape parameters, $\vec{\beta}$; and a pose-dependent blend shape function, $B_P(\vec{\theta}) : \mathbb{R}^{|\vec{\theta}|} \mapsto \mathbb{R}^{3N}$, that takes as input a vector of pose parameters, $\vec{\theta}$, and accounts for the effects of pose-dependent deformations (Fig. 3(c)). The corrective blend shapes of these functions are added together in the rest pose as illustrated in (Fig. 3(c)). Finally, a standard blend skinning function $W(\cdot)$ (linear or dual-quaternion) is applied to rotate the vertices around the estimated joint centers with smoothing defined by the blend weights. The result is a model, $M(\vec{\beta}, \vec{\theta}; \Phi) : \mathbb{R}^{|\vec{\theta}| \times |\vec{\beta}|} \mapsto \mathbb{R}^{3N}$, that maps shape and pose parameters to vertices (Fig. 3(d)). Here Φ represents the learned model parameters described below.

Below we will use both LBS and DQBS skinning methods. In general the skinning method can be thought of as a generic black box. Given a particular skinning method our goal is to learn Φ to correct for limitations of the method so as to model training meshes. Note that the learned pose blend shapes both correct errors caused by the blend skinning function and static soft-tissue deformations caused by changes in pose.

Below we describe each term in the model. For convenience, a notational summary is provided in Table 1 in the Appendix.

Blend skinning. To fix ideas and define notation, we present the LBS version as it makes exposition clear (the DQBS version of SMPL only requires changing the skinning equation). Meshes and blend shapes are vectors of vertices represented by bold capital letters (e.g. $\mathbf{X}$) and lowercase bold letters (e.g. $\mathbf{x}_i \in \mathbb{R}^3$) are vectors representing a particular vertex. The vertices are sometimes represented in homogeneous coordinates. We use the same notation for a vertex whether it is in standard or homogeneous coordinates as it should always be clear from the context which form is needed.

The pose of the body is defined by a standard skeletal rig, where $\vec{\omega}_k \in \mathbb{R}^3$ denotes the axis-angle representation of the relative rotation of part k with respect to its parent in the kinematic tree. Our rig has $K = 23$ joints, hence a pose $\vec{\theta} = [\vec{\omega}_0^T, \ldots, \vec{\omega}_K^T]^T$ is defined by $|\vec{\theta}| = 3 \times 23 + 3 = 72$ parameters; i.e. 3 for each part plus 3 for the root orientation. Let $\bar{\omega} = \frac{\vec{\omega}}{\|\vec{\omega}\|}$ denote the unit norm axis of rotation. Then the axis angle for every joint j is transformed to a rotation matrix using the *Rodrigues formula*

$$\exp(\vec{\omega}_j) = \mathcal{I} + \widehat{\bar{\omega}}_j \sin(\|\vec{\omega}_j\|) + \widehat{\bar{\omega}}_j^2 \cos(\|\vec{\omega}_j\|) \tag{1}$$

where $\widehat{\bar{\omega}}$ is the skew symmetric matrix of the 3-vector $\bar{\omega}$ and $\mathcal{I}$ is the 3×3 identity matrix. Using this, the standard linear blend skinning function $W(\bar{\mathbf{T}}, \mathbf{J}, \vec{\theta}, \mathcal{W}) : \mathbb{R}^{3N \times 3K \times |\vec{\theta}| \times |\mathcal{W}|} \mapsto \mathbb{R}^{3N}$ takes vertices in the rest pose, $\bar{\mathbf{T}}$, joint locations, $\mathbf{J}$, a pose, $\vec{\theta}$, and the blend weights, $\mathcal{W}$, and returns the posed vertices. Each vertex $\bar{\mathbf{t}}_i$ in $\bar{\mathbf{T}}$ is transformed into $\bar{\mathbf{t}}'_i$ (both column vectors in homogeneous coordinates) as

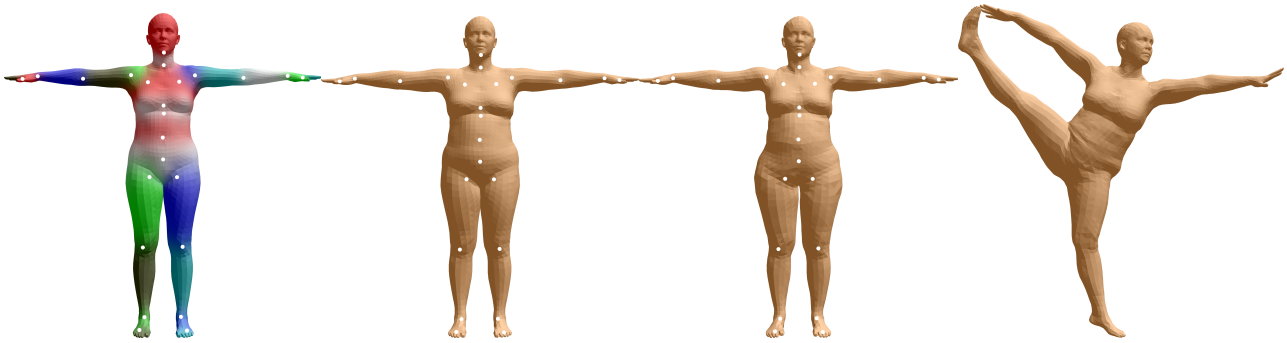
$$\bar{\mathbf{t}}'_i = \sum_{k=1}^{K} w_{k,i} G'_k(\vec{\theta}, \mathbf{J}) \bar{\mathbf{t}}_i \tag{2}$$

$$G'_k(\vec{\theta}, \mathbf{J}) = G_k(\vec{\theta}, \mathbf{J}) G_k(\vec{\theta}^*, \mathbf{J})^{-1} \tag{3}$$

$$G_k(\vec{\theta}, \mathbf{J}) = \prod_{j \in A(k)} \left[\begin{array}{c|c} \exp(\vec{\omega}_j) & \mathbf{j}_j \\ \hline \vec{0} & 1 \end{array} \right] \tag{4}$$

where $w_{k,i}$ is an element of the blend weight matrix $\mathcal{W}$, representing how much the rotation of part k effects the vertex i, $\exp(\vec{\theta}_j)$ is the local 3×3 rotation matrix corresponding to joint j, $G_k(\vec{\theta}, \mathbf{J})$ is the world transformation of joint k, and $G'_k(\vec{\theta}, \mathbf{J})$ is the same transformation after removing the transformation due to the rest pose, $\vec{\theta}^*$. Each 3-element vector in $\mathbf{J}$ corresponding to a single joint center, j, is denoted $\mathbf{j}_j$. Finally, $A(k)$ denotes the ordered set of joint ancestors of joint k. Note, for compatibility with existing rendering engines, we assume $\mathcal{W}$ is sparse and allow at most four parts to influence a vertex.

Many methods have modified equation (2) to make skinning more expressive. For example MWE [Wang and Phillips 2002] replaces $G_k(\vec{\theta}, \mathbf{J})$ with a more general affine transformation matrix and replaces the scalar weight with a separate weight for every element of



(a) $\bar{\mathbf{T}}, \mathcal{W}$ **(b)** $\bar{\mathbf{T}} + B_S(\vec{\beta}), J(\vec{\beta})$ **(c)** $T_P(\vec{\beta}, \vec{\theta}) = \bar{\mathbf{T}} + B_S(\vec{\beta}) + B_P(\vec{\theta})$ **(d)** $W(T_P(\vec{\beta}, \vec{\theta}), J(\vec{\beta}), \vec{\theta}, \mathcal{W})$

Figure 3: SMPL model. *(a) Template mesh with blend weights indicated by color and joints shown in white. (b) With identity-driven blendshape contribution only; vertex and joint locations are linear in shape vector $\vec{\beta}$. (c) With the addition of of pose blend shapes in preparation for the split pose; note the expansion of the hips. (d) Deformed vertices reposed by dual quaternion skinning for the split pose.*

the transformation matrix. Such changes are expressive but are not compatible with existing animation systems.

To maintain compatibility, we keep the basic skinning function and instead modify the template in an additive way and learn a function to predict joint locations. Our model, $M(\vec{\beta}, \vec{\theta}; \Phi)$ is then

$$M(\vec{\beta}, \vec{\theta}) = W(T_P(\vec{\beta}, \vec{\theta}), J(\vec{\beta}), \vec{\theta}, \mathcal{W}) \quad (5)$$

$$T_P(\vec{\beta}, \vec{\theta}) = \bar{\mathbf{T}} + B_S(\vec{\beta}) + B_P(\vec{\theta}) \quad (6)$$

where $B_S(\vec{\beta})$ and $B_P(\vec{\theta})$ are vectors of vertices representing offsets from the template. We refer to these as shape and pose blend shapes respectively.

Given this definition, a vertex $\bar{\mathbf{t}}_i$ is transformed according to

$$\bar{\mathbf{t}}'_i = \sum_{k=1}^{K} w_{k,i} G'_k(\vec{\theta}, J(\vec{\beta}))(\bar{\mathbf{t}}_i + \mathbf{b}_{S,i}(\vec{\beta}) + \mathbf{b}_{P,i}(\vec{\theta})) \quad (7)$$

where $\mathbf{b}_{S,i}(\vec{\beta}), \mathbf{b}_{P,i}(\vec{\theta})$ are vertices in $B_S(\vec{\beta})$ and $B_P(\vec{\theta})$ respectively and represent the shape and pose blend shape offsets for the vertex $\bar{\mathbf{t}}_i$. Hence, the joint centers are now a function of body shape and the template mesh that is deformed by blend skinning is now a function of both pose and shape. Below we describe each term in detail.

Shape blend shapes. The body shapes of different people are represented by a linear function B_S

$$B_S(\vec{\beta}; \mathcal{S}) = \sum_{n=1}^{|\vec{\beta}|} \beta_n \mathbf{S}_n \quad (8)$$

where $\vec{\beta} = [\beta_1, \ldots, \beta_{|\vec{\beta}|}]^T$, $|\vec{\beta}|$ is the number of linear shape coefficients, and the $\mathbf{S}_n \in \mathbb{R}^{3N}$ represent orthonormal principal components of shape displacements. Let $\mathcal{S} = [\mathbf{S}_1, \ldots, \mathbf{S}_{|\vec{\beta}|}] \in \mathbb{R}^{3N \times |\vec{\beta}|}$ be the matrix of all such shape displacements. Then the linear function $B_S(\vec{\beta}; \mathcal{S})$ is fully defined by the matrix $\mathcal{S}$, which is learned from registered training meshes, see Sec. 4.

Notationally, the values to the right of a semicolon represent learned parameters, while those on the left are parameters set by an animator. For notational convenience, we often omit the learned parameters when they are not explicitly being optimized in training.

Figure 3(b) illustrates the application of these shape blend shapes to the template $\bar{\mathbf{T}}$ to produce a new body shape.

Pose blend shapes. We denote as $R : \mathbb{R}^{|\vec{\theta}|} \mapsto \mathbb{R}^{9K}$ a function that maps a pose vector $\vec{\theta}$ to a vector of concatenated part relative rotation matrices, $\exp(\vec{\omega})$. Given that our rig has 23 joints, $R(\vec{\theta})$ is a vector of length $(23 \times 9 = 207)$. Elements of $R(\vec{\theta})$ are functions of sines and cosines (Eq. (1)) of joint angles and therefore $R(\vec{\theta})$ is non-linear with $\vec{\theta}$.

Our formulation differs from previous work in that we define the effect of the pose blend shapes to be linear in $R^*(\vec{\theta}) = (R(\vec{\theta}) - R(\vec{\theta}^*))$, where $\vec{\theta}^*$ denotes the rest pose. Let $R_n(\vec{\theta})$ denote the n^{th} element of $R(\vec{\theta})$, then the vertex deviations from the rest template are

$$B_P(\vec{\theta}; \mathcal{P}) = \sum_{n=1}^{9K} (R_n(\vec{\theta}) - R_n(\vec{\theta}^*))\mathbf{P}_n, \quad (9)$$

where the blend shapes, $\mathbf{P}_n \in \mathbb{R}^{3N}$, are again vectors of vertex displacements. Here $\mathcal{P} = [\mathbf{P}_1, \ldots, \mathbf{P}_{9K}] \in \mathbb{R}^{3N \times 9K}$ is a matrix of all 207 pose blend shapes. In this way, the pose blend shape function $B_P(\vec{\theta}; \mathcal{P})$ is fully defined by the matrix $\mathcal{P}$, which we learn in Sec. 4.

Note that subtracting the rest pose rotation vector, $R(\vec{\theta}^*)$, guarantees that the contribution of the pose blend shapes is zero in the rest pose, which is important for animation.

Joint locations. Different body shapes have different joint locations. Each joint is represented by its 3D location in the rest pose. It is critical that these are accurate, otherwise there will be artifacts when the model is posed using the skinning equation. For that reason, we define the joints as a function of the body shape, $\vec{\beta}$,

$$J(\vec{\beta}; \mathcal{J}, \bar{\mathbf{T}}, \mathcal{S}) = \mathcal{J}(\bar{\mathbf{T}} + B_S(\vec{\beta}; \mathcal{S})) \quad (10)$$

where $\mathcal{J}$ is a matrix that transforms rest vertices into rest joints. We learn the regression matrix, $\mathcal{J}$, from examples of different people in many poses, as part of our overall model learning in Sec. 4. This matrix models which mesh vertices are important and how to combine them to estimate the joint locations.

SMPL model. We can now specify the full set of model parameters of the SMPL model as $\Phi = \{\bar{\mathbf{T}}, \mathcal{W}, \mathcal{S}, \mathcal{J}, \mathcal{P}\}$. We describe how to learn these in Sec. 4. Once learned they are held fixed and new

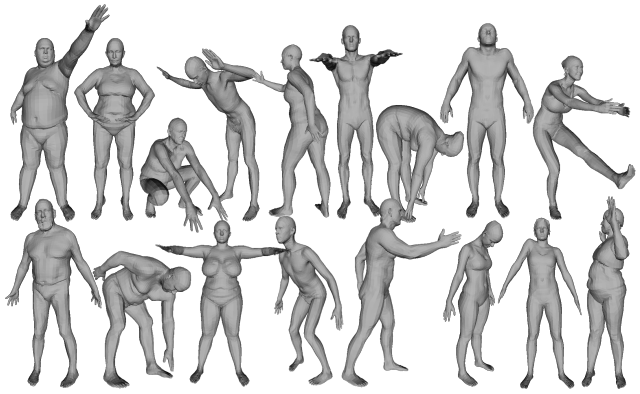


Figure 4: *Sample registrations from the multipose dataset.*

body shapes and poses are created and animated by varying $\vec{\beta}$ and $\vec{\theta}$ respectively.

Then the SMPL model is finally defined as $M(\vec{\beta}, \vec{\theta}; \Phi) =$

$$W\left(T_P(\vec{\beta}, \vec{\theta}; \bar{\mathbf{T}}, \mathcal{S}, \mathcal{P}), J(\vec{\beta}; \mathcal{J}, \bar{\mathbf{T}}, \mathcal{S}), \vec{\theta}, \mathcal{W}\right) \quad (11)$$

and hence each vertex is transformed as

$$\mathbf{t}'_i = \sum_{k=1}^{K} w_{k,i} G'_k(\vec{\theta}, J(\vec{\beta}; \mathcal{J}, \bar{\mathbf{T}}, \mathcal{S})) \mathbf{t}_{P,i}(\vec{\beta}, \vec{\theta}; \bar{\mathbf{T}}, \mathcal{S}, \mathcal{P}) \quad (12)$$

where $\mathbf{t}_{P,i}(\vec{\beta}, \vec{\theta}; \bar{\mathbf{T}}, \mathcal{S}, \mathcal{P}) =$

$$\bar{\mathbf{t}}_i + \sum_{m=1}^{|\vec{\beta}|} \beta_m \mathbf{s}_{m,i} + \sum_{n=1}^{9K} (R_n(\vec{\theta}) - R_n(\vec{\theta}^*)) \mathbf{p}_{n,i} \quad (13)$$

represents the vertex i after applying the blend shapes and where $\mathbf{s}_{m,i}, \mathbf{p}_{n,i} \in \mathbb{R}^3$ are the elements of the shape and pose blend shapes corresponding to template vertex $\bar{\mathbf{t}}_i$.

Below we experiment with both LBS and DQBS and train the parameters for each. We refer to these models as SMPL-LBS and SMPL-DQBS; SMPL-DQBS is our default model, and we use SMPL as shorthand to mean SMPL-DQBS.

4 Training

We train the SMPL model parameters to minimize reconstruction error on two datasets. Each dataset contains meshes with the same topology as our template that have been aligned to high-resolution 3D scans using [Bogo et al. 2014]; we call these aligned meshes "registrations." The *multi-pose* dataset consists of 1786 registrations of 40 individuals (891 registrations spanning 20 females, and 895 registrations spanning 20 males); a sampling is shown in Fig. 4. The *multi-shape* dataset consists of registrations to the CAESAR dataset [Robinette et al. 2002], totaling 1700 registrations for males and 2100 for females; a few examples are shown in Fig. 5. We denote the j^{th} mesh in the multi-pose dataset as $\mathbf{V}_j^P$ and the j^{th} mesh in the multi-shape dataset as $\mathbf{V}_j^S$.

Our goal is to train the parameters $\Phi = \left\{\bar{\mathbf{T}}, \mathcal{W}, \mathcal{S}, \mathcal{J}, \mathcal{P}\right\}$ to minimize vertex reconstruction error on the datasets. Because our model decomposes shape and pose, we train these separately, simplifying optimization. We first train $\{\mathcal{J}, \mathcal{W}, \mathcal{P}\}$ using our multi-pose dataset and then train $\{\bar{\mathbf{T}}, \mathcal{S}\}$ using our multi-shape dataset. We train separate models for men and women (i.e. Φ_m and Φ_f).

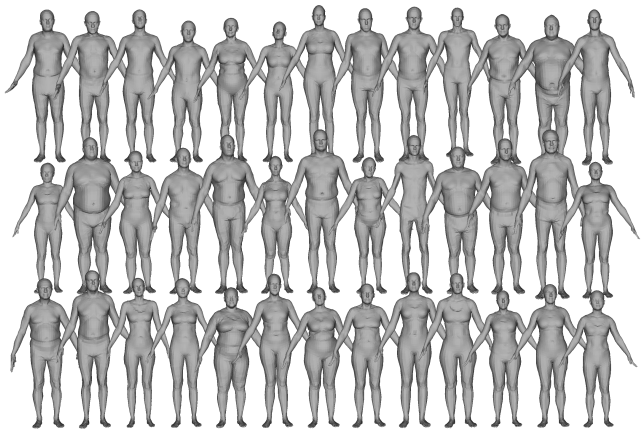


Figure 5: *Sample registrations from the multishape dataset.*

4.1 Pose Parameter Training

We first use the multi-pose dataset to train $\{\mathcal{J}, \mathcal{W}, \mathcal{P}\}$. To this end, we need to compute the rest templates, $\hat{\mathbf{T}}_i^P$, and joint locations, $\hat{\mathbf{J}}_i^P$, for each subject, i, as well as the pose parameters, $\vec{\theta}_j$, for each registration, j, in the dataset. We alternate between optimizing registration specific parameters $\vec{\theta}_j$, subject-specific parameters $\{\hat{\mathbf{T}}_i^P, \hat{\mathbf{J}}_i^P\}$, and global parameters $\{\mathcal{W}, \mathcal{P}\}$. We then learn the matrix, $\mathcal{J}$, to regress from subject-specific vertex locations, $\hat{\mathbf{T}}_i^P$, to subject-specific joint locations, $\hat{\mathbf{J}}_i^P$. To achieve all this, we minimize an objective function consisting of a data term, E_D, and several regularization terms $\{E_J, E_Y, E_P, E_W\}$ defined below.

The data term penalizes the squared Euclidean distance between registration vertices and model vertices

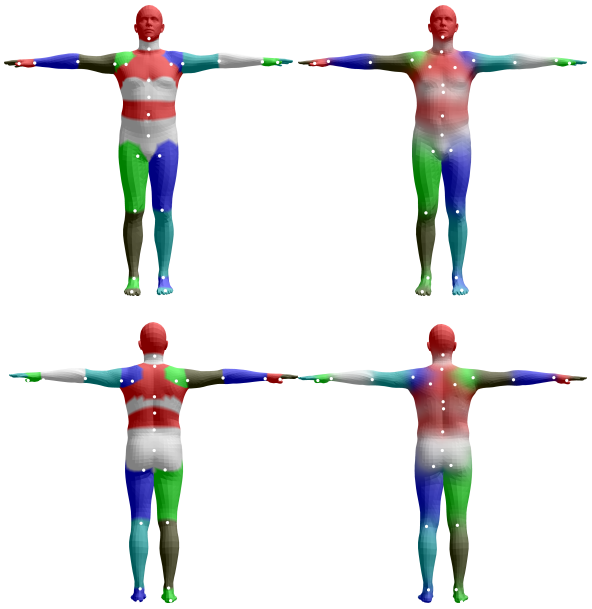
$$E_D(\hat{\mathbf{T}}^P, \hat{\mathbf{J}}^P, \mathcal{W}, \mathcal{P}, \Theta) = \sum_{j=1}^{P_{\text{reg}}} ||\mathbf{V}_j^P - W(\hat{\mathbf{T}}_{s(j)}^P + B_P(\vec{\theta}_j; \mathcal{P}), \hat{\mathbf{J}}_{s(j)}^P, \vec{\theta}_j, \mathcal{W})||^2$$

where $\Theta = \left\{\vec{\theta}_1, \ldots, \vec{\theta}_{P_{\text{reg}}}\right\}$, $s(j)$ is the subject index corresponding to registration j, P_{reg} are the number of meshes in the pose trainings set, $\hat{\mathbf{T}}^P = \{\hat{\mathbf{T}}_i^P\}_{i=1}^{P_{\text{subj}}}$, $\hat{\mathbf{J}}^P = \{\hat{\mathbf{J}}_i^P\}_{i=1}^{P_{\text{subj}}}$ are the sets of all rest poses and joints, and P_{subj} is the number of subjects in the pose training set.

We estimate $207 \times 3 \times 6890 = 4,278,690$ parameters for the pose blend shapes, $\mathcal{P}$, $4 \times 3 \times 6890 = 82,680$ parameters for the skinning weights, $\mathcal{W}$, and $3 \times 6890 \times 23 \times 3 = 1,426,230$ for the joint regressor matrix, $\mathcal{J}$. To make the estimation well behaved, we regularize by making several assumptions. A symmetry regularization term, E_Y, penalizes left-right asymmetry for $\hat{\mathbf{J}}^P$ and $\hat{\mathbf{T}}^P$

$$E_Y(\hat{\mathbf{J}}^P, \hat{\mathbf{T}}^P) = \sum_{i=1}^{P_{\text{subj}}} \lambda_U ||\hat{\mathbf{J}}_i^P - U(\hat{\mathbf{J}}_i^P)||^2 + ||\hat{\mathbf{T}}_i^P - U(\hat{\mathbf{T}}_i^P)||^2,$$

where $\lambda_U = 100$, and where $U(\mathbf{T})$ finds a mirror image of vertices $\mathbf{T}$, by flipping across the sagittal plane and swapping symmetric vertices. This term encourages symmetric template meshes and, more importantly, symmetric joint locations. Joints are unobserved variables and along the spine they are particularly difficult to localize. While models trained without the symmetry term produce



(a) *Segmentation* **(b)** *Initialization* $\mathcal{W}_I$

Figure 6: Initialization of joints and blend weights. *Discrete part segmentation in (a) is diffused to obtain initial blend weights, $\mathcal{W}_I$, in (b). Initial joint centers are shown as white dots.*

reasonable results, enforcing symmetry produces models that are visually more intuitive for animation.

Our model is hand segmented into 24 parts (Fig. 6). We use this segmentation to compute an initial estimate of the joint centers and a regressor $\mathcal{J}_I$ from vertices to these centers. This regressor computes the initial joints by taking the average of the ring of vertices connecting two parts. When estimating the joints for each subject we regularize them to be close to this initial prediction:

$$E_J(\hat{\mathbf{T}}^P, \hat{\mathbf{J}}^P) = \sum_{i=1}^{P_{\text{subj}}} ||\mathcal{J}_I \hat{\mathbf{T}}_i^P - \hat{\mathbf{J}}_i^P||^2.$$

To help prevent overfitting of the pose-dependent blend shapes, we regularize them towards zero

$$E_P(\mathcal{P}) = ||\mathcal{P}||_F^2,$$

where $\|\cdot\|_F$ denotes the Frobenius norm. Note that replacing the quadratic penalty with an L1 penalty would encourage greater sparsity of the blend shapes. We did not try this.

We also regularize the blend weights towards the initial weights, $\mathcal{W}_I$, shown in Fig. 6:

$$E_W(\mathcal{W}) = ||\mathcal{W} - \mathcal{W}_I||_F^2.$$

The initial weights are computed by simply diffusing the segmentation.

Altogether, the energy for training $\{\mathcal{W}, \mathcal{P}\}$ is as follows:

$$E_*(\hat{\mathbf{T}}^P, \hat{\mathbf{J}}^P, \Theta, \mathcal{W}, \mathcal{P}) = E_D + \lambda_Y E_Y + \lambda_J E_J + \lambda_P E_P + E_W, \quad (14)$$

where $\lambda_Y = 100$, $\lambda_J = 100$ and $\lambda_P = 25$. These weights were set empirically. Our model has a large number of parameters and the regularization helps prevent overfitting. As the size of the training set grows, so does the strength of the data term, effectively reducing the influence of the regularization terms. Our experiments below with held-out test data suggest that the learned models are not overfit to the data and generalize well.

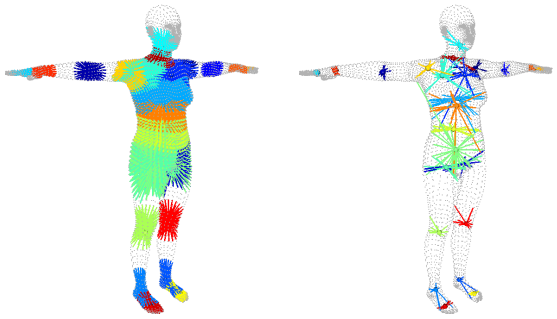


Figure 7: Joint regression. *(left) Initialization. Joint locations can be influenced by locations on the surface, indicated by the colored lines. We assume that these influences are somewhat local. (right) Optimized. After optimization we find a sparse set of vertices and associated weights influencing each joint.*

The overall optimization strategy is described in Sec. 4.3.

Joint regressor. Optimizing the above gives a template mesh and joint locations for each subject, but we want to predict joint locations for new subjects with new body shapes. To that end, we learn the regressor matrix $\mathcal{J}$ to predict the training joints from the training bodies. We tried several regression strategies; what we found to work best, was to compute $\mathcal{J}$ using non-negative least squares [Lawson and Hanson 1995] with the inclusion of a term that encourages the weights to add to one. This approach encourages sparsity of the vertices used to predict the joints. Making weights positive and add to one discourages predicting joints outside the surface. These constraints enforce the predictions to be in the convex hull of surface points. Figure 7 shows the non-zero elements of the regression matrix, illustrating that a sparse set of surface vertices are linearly combined to estimate the joint centers.

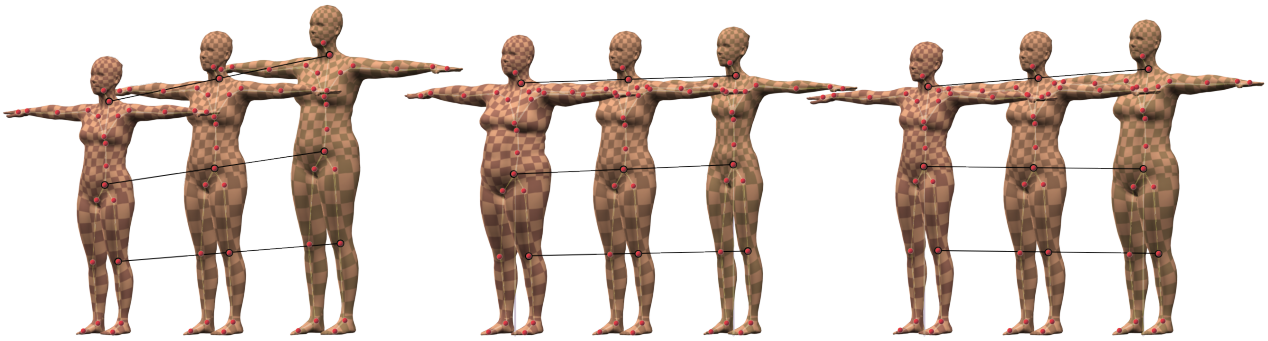
4.2 Shape Parameter Training

Our shape space is defined by a mean and principal shape directions $\{\bar{\mathbf{T}}, \mathcal{S}\}$. It is computed by running PCA on shape registrations from our multi-shape database after pose normalization. Pose normalization transforms a raw registration $\mathbf{V}_j^S$ into a registration, $\hat{\mathbf{T}}_j^S$, in the rest pose, $\vec{\theta}^*$. This normalization is critical to ensure that pose and shape are modeled separately.

To pose-normalize a registration, $\mathbf{V}_j^S$, we first have to estimate its pose. We denote $\hat{\mathbf{T}}_\mu^P$ and $\hat{\mathbf{J}}_\mu^P$ as the mean shape and mean joint locations from the multi-pose database respectively. Let $W_e(\hat{\mathbf{T}}_\mu^P, \hat{\mathbf{J}}_\mu^P, \vec{\theta}, \mathcal{W}), \mathbf{V}_{j,e}^S \in \mathbb{R}^3$ denote an edge of the model and of the registration respectively. An edge is obtained by subtracting a pair of neighboring vertices. To estimate the pose using an average generic shape $\hat{\mathbf{T}}_\mu^P$, we minimize the following sum of squared edge differences so that $\vec{\theta}_j =$

$$\arg\min_{\vec{\theta}} \sum_e ||W_e(\hat{\mathbf{T}}_\mu^P + B_P(\vec{\theta}; \mathcal{P}), \hat{\mathbf{J}}_\mu^P, \vec{\theta}, \mathcal{W}) - \mathbf{V}_{j,e}^S||^2, \quad (15)$$

where the sum is taken over all edges in the mesh. This allows us to get a good estimate of the pose without knowing the subject specific shape.



(a) *PC 1* **(b)** *PC 2* **(c)** *PC 3*

Figure 8: Shape blend shapes. *The first three shape principal components of body shape are shown. PC1 and PC2 vary from -2 to +2 standard deviations from left to right, while PC3 varies from -5 to +5 standard deviations to make the shape variation more visible. Joint locations (red dots) vary as a function of body shape and are predicted using the learned regressor,* $\mathcal{J}$.

Once the pose $\vec{\theta}_j$ is known we solve for $\hat{\mathbf{T}}_j^S$ by minimizing

$$\hat{\mathbf{T}}_j^S = \arg\min_{\hat{\mathbf{T}}} ||W(\hat{\mathbf{T}} + B_P(\vec{\theta}_j; \mathcal{P}), \mathcal{J}\hat{\mathbf{T}}, \vec{\theta}_j, \mathcal{W}) - \mathbf{V}_j^S||^2.$$

This computes the shape that, when posed, matches the training registration. This shape is the pose-normalized shape.

We then run PCA on $\{\hat{\mathbf{T}}_j^S\}_{j=1}^{S_{\text{subj}}}$ to obtain $\{\bar{\mathbf{T}}, \mathcal{S}\}$. This procedure is designed to maximize the explained variance of vertex offsets in the rest pose, given a limited number of shape directions.

Note that the optimization of pose is critically important when building a shape basis from vertices. Without this step, pose variations of the subjects in the shape training dataset would be captured in the shape blend shapes. The resulting model would not be decomposed properly into shape and pose. Note also that this approach constrasts with SCAPE or BlendSCAPE, where PCA is performed in the space of per-triangle deformations. Unlike vertices, triangle deformations do not live in a Euclidean space [Freifeld and Black 2012]. Hence PCA on vertices is more principled and is consistent with the registration data term, which consists of squared vertex disparities.

Figure 8 visualizes the first three shape components. The figure also shows how the joint locations change with the changes in body shape. The joint positions are shown by the spheres and are computed from the surface meshes using the learned joint regression function. The lines connecting the joints across the standard deviations illustrate how the joint positions vary linearly with shape.

Figure 9 shows the relative cumulative variance of SMPL and BlendSCAPE. While SMPL requires many fewer PCs to account for the same percentage of overall variance, the variance is different in the two cases: one is variance in vertex locations and the other is variance in triangle deformations. Explained variance in deformations does not directly translate into explained variance in vertex locations. While this makes the two models difficult to compare precisely, triangle deformations have many more degrees of freedom and it is likely that there are many deformations that produce visually similar shapes. A model requiring fewer components is generally preferable.

4.3 Optimization summary

Pose parameters $\vec{\theta}_j$ in Eq. (14) are first initialized by minimizing the difference between the model and the registration edges,

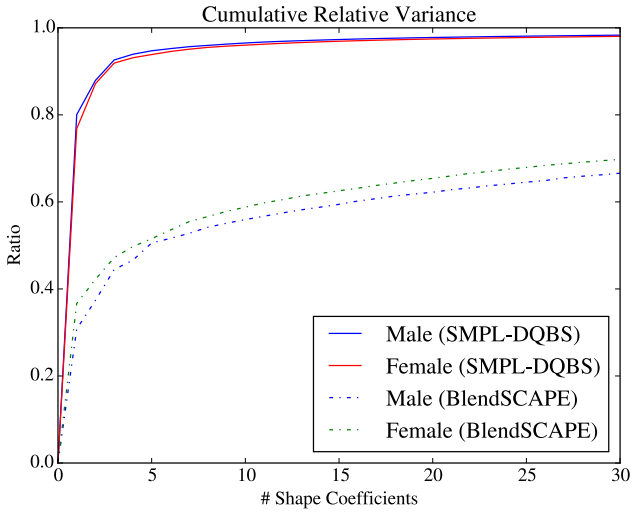


Figure 9: *Cumulative relative variance of the CAESAR dataset explained as a function of the number of shape coefficients. For SMPL the variance is in vertex locations, while for BlendSCAPE it is in triangle deformations.*

similar to Eq. (15) using an average template shape. Then $\{\hat{\mathbf{T}}^P, \hat{\mathbf{J}}^P, \mathcal{W}, \mathcal{P}, \Theta\}$ are estimated in an alternating manner to minimize Eq. 14. We proceed to estimate $\mathcal{J}$ from $\{\hat{\mathbf{J}}^P, \hat{\mathbf{T}}^P\}$. We then run PCA on pose normalized subjects $\{\hat{\mathbf{T}}_j^S\}_{j=1}^{S_{\text{subj}}}$ to obtain $\{\bar{\mathbf{T}}, \mathcal{S}\}$. The final model is defined by $\{\mathcal{J}, \mathcal{W}, \mathcal{P}, \bar{\mathbf{T}}, \mathcal{S}\}$. Note that all training parameters except for $\{\bar{\mathbf{T}}, \mathcal{S}\}$ are found with gradient-based dogleg minimization [Nocedal and Wright 2006]. Gradients are computed with automatic differentiation using the the Chumpy framework [Loper and Black 2014].

5 SMPL Evaluation

All training subjects gave prior informed written consent for their data to be used in creating statistical models for distribution. Registered meshes and identifiable subjects shown here are of professional models working under contract.

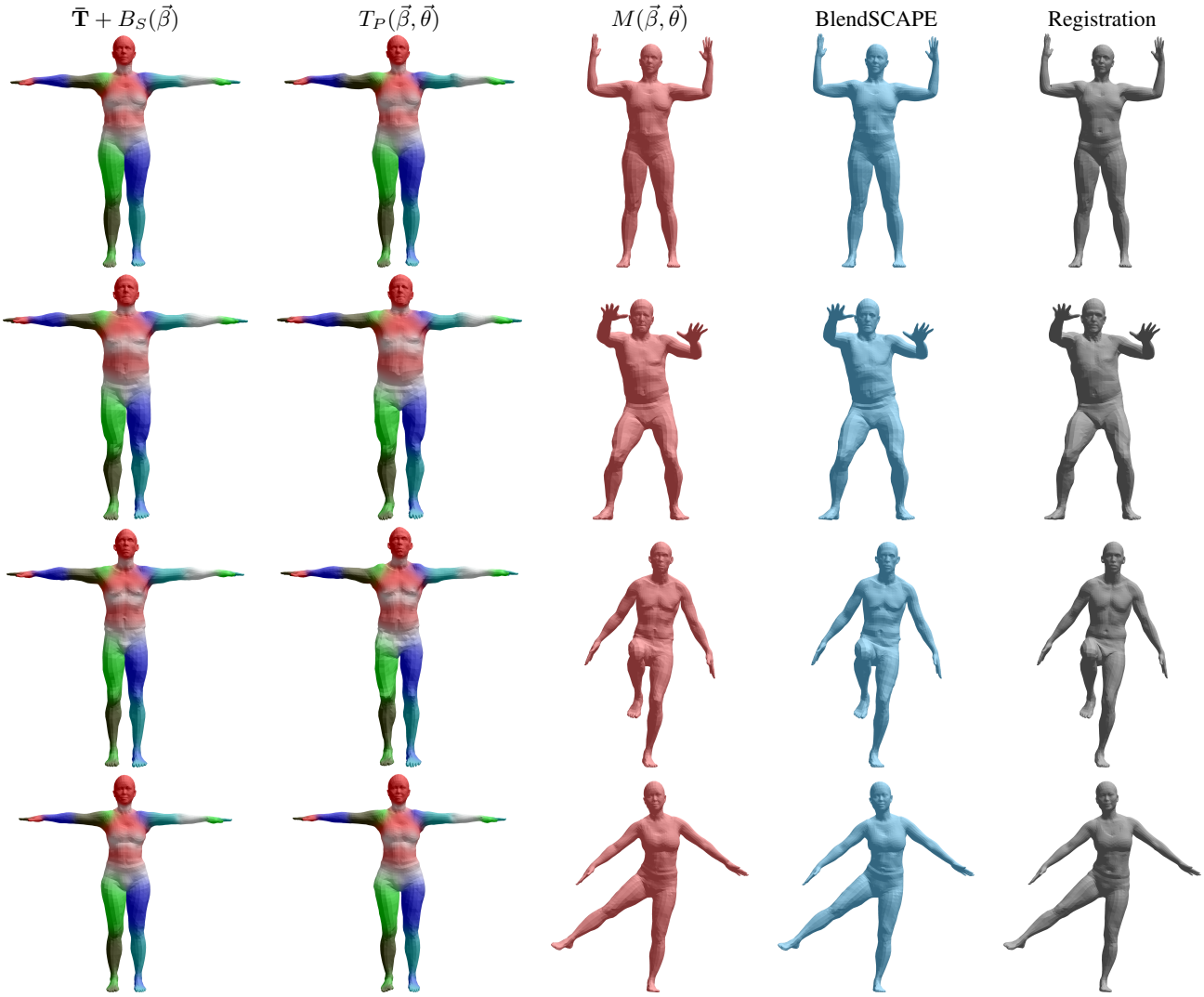


Figure 10: *Model fitting with intermediate stages. We fit both BlendSCAPE (blue) and SMPL-LBS, $M(\vec{\beta}, \vec{\theta})$, (red) to registered meshes by optimizing pose and shape. $\bar{\mathbf{T}} + B_S(\vec{\beta})$ shows the estimated body shape and $T_P(\vec{\beta}, \vec{\theta})$ shows the effects of pose-dependent blend shapes. Here we show SMPL-LBS, because T_P shows more variation due to pose than SMPL-DQBS.*

5.1 Quantitative Evaluation

We evaluate two types of error. *Model generalization* is the ability of the model to fit to meshes of new people and poses; this tests both shape and pose blend shapes. *Pose generalization* is the ability to generalize a shape of an individual to new poses of the same person; this primarily tests how well pose blend shapes correct skinning artifacts and pose-dependent deformations. Both are measured by mean absolute vertex-to-vertex distance between the model and test registrations. For this evaluation we use 120 registered meshes of four women and two men from the public Dyna dataset [Dyn 2015]. These meshes contain a variety of body shapes and poses. All meshes are in alignment with our template and none were used to train our models. Figure 10 (gray) shows four examples of these registered meshes.

We evaluate SMPL-LBS and SMPL-DQBS. We also compare these with a BlendSCAPE model [Hirshberg et al. 2012] trained from exactly the same data as the SMPL models. The kinematic tree structure for SMPL and the BlendSCAPE model are the same: therefore we have the same number of pose parameters. We also compare the models using the same number of shape parameters.

To measure model generalization we first fit each model to each registered mesh, optimizing over shape $\vec{\beta}$ and pose $\vec{\theta}$ to find the best fit in terms of squared vertex distances. Figure 10 shows fits of the SMPL-LBS (red) and BlendSCAPE (blue) models to the registered meshes. Both do a good job of fitting the data. The figure also shows how the model works. Illustrated are the estimated body shape, $\bar{\mathbf{T}}+B_S(\vec{\beta})$, and the effect of applying the pose blend shapes, $T_P(\vec{\beta}, \vec{\theta})$.

For pose generalization, we take each indvidual, select one of the estimated body shapes from the generalization task, and then optimize the pose, $\vec{\theta}$, for each of the other meshes of that subject, keeping the body shape fixed. The assumption behind pose generalization is that, if a model is properly decomposed into pose and shape, then the model should be able to fit the same subject in a different pose, without readjusting shape parameters. Note that the pose blend shapes are trained to fit observed registrations. As such,

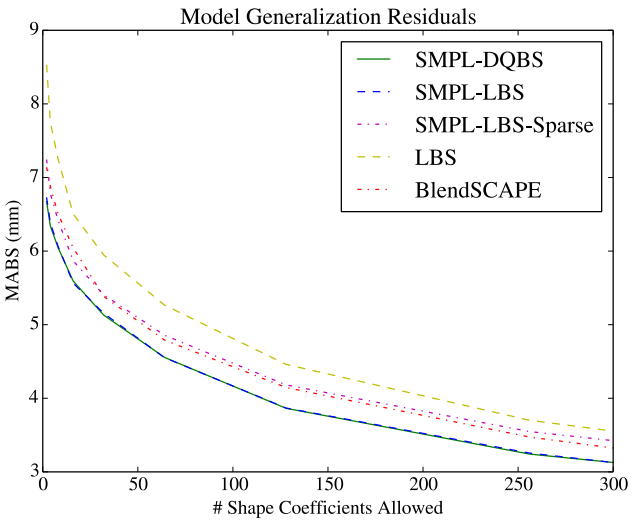


Figure 11: *Model generalization indicates how well we can fit an independent registration. Mean absolute vertex error versus the number of shape coefficients used.*

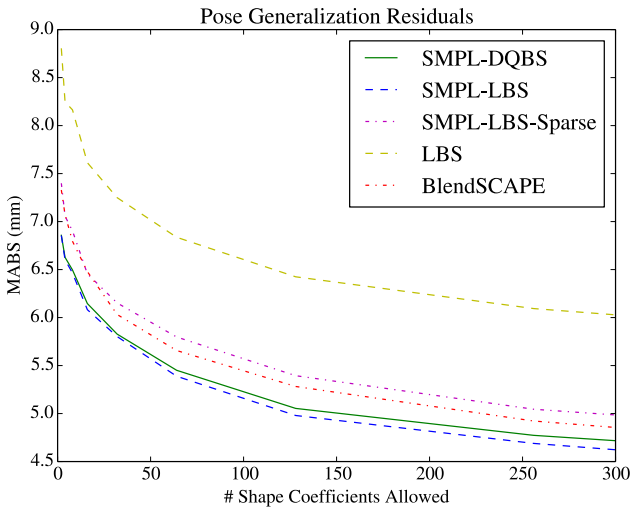


Figure 12: *Pose generalization error indicates how well a fitted shape generalizes to new poses.*

they correct for problems of blend skinning and try to capture pose-dependent deformations. Since the pose blend shapes are not dependent on body shape, they capture something about the average deformations in the training set.

Figures 11 and 12 show the error of the SMPL models and BlendSCAPE as a function of the number of body shape coefficients used. The differences between SMPL and BlendSCAPE are small (on the order of 0.5mm) but SMPL is more accurate on average. Remarkably, SMPL-LBS and SMPL-DQBS are essentially identical in model generalization and SMPL-LBS is actually slightly better at pose generalization. This is surprising because the pose blend shapes have much more to correct with LBS. Possibly the simplicity of LBS helps with generalization to new poses. This analysis is important because it says that users can choose the simpler and faster LBS model over the DQBS model.

The plots also show how well standard LBS fits the test data. This corresponds to the SMPL-LBS model with no pose blend shapes. Not surprisingly, LBS produces much higher error than either BlendSCAPE or SMPL. LBS is not as bad in Fig. 11 because here the model can vary body shape parameters, effectively using changes in identity to try to explain deformations due to pose. Figure 12 uses a fixed body shape and consequently illustrates how LBS does not model pose-dependent deformations realistically. Note that here we do not retrain a model specifically for LBS and expect such a model would be marginally more accurate.

5.2 Sparse SMPL

The pose blend shapes in SMPL are not sparse in the sense that a rotation of a part can influence any vertex of the mesh. With sufficient training data sparsity may emerge from data; *e.g.* the shoulder corrections will not be influenced by ankle motions. To make hand animation more intuitive, and to regularize the model to prevent long-range influences of joints, we can manually enforce sparsity. To this end, we trained a sparse version of SMPL by using the same sparsity pattern used for blend weights. In particular, we allow a vertex deviation to be influenced by at most 4 joints. Since every joint corresponds to a rotation matrix, the pose blend shape corresponding to any given vertex will be driven by 9×4 numbers as opposed to 9×23.

This model is referred to as SMPL-LBS-Sparse in Figs. 11 and 12. It is consistently less accurate than the regular SMPL-LBS model but may still be useful to animators. This suggests that SMPL-LBS is not overfit to the training data and that sparseness reduces the capacity of the model. The sparse model sometimes produces slight discontinuities at boundaries were vertices are influenced by different joint angles. Other strategies to enforce sparsity could be adopted, such as using an L1 prior or enforcing smoothness in the pose blend shapes. These approaches, however, would complicate the training process.

5.3 Visual Evaluation

Figure 13 illustrates the decomposition of shape parameters $\vec{\beta}$ and pose parameters $\vec{\theta}$ in SMPL. Pose is held constant from left to right across each row while varying the shape. Similarly, the shape of each person is held constant while varying the pose from top to bottom in each column. The bodies are reposed using poses from the CMU mocap database [CMU 2000]. Note that the pose-dependent deformations look natural through a wide range of poses, despite very different body shapes. This illustrates that the joint regression works well and that the blend shapes are general.

Please see the **Supplemental Video** for many more examples and animations comparing SMPL and BlendSCAPE, illustrating the pose blend shapes, and illustrating body shape and pose variation.

5.4 Run-time Performance

The run-time cost of SMPL is dominated by skinning and blend-shape multiplication. Many skinning implementations exist, and we do not claim to have the fastest. Performance of our own CPU-based implementation, and a comparison against BlendSCAPE, is shown in Fig. 14. The plot shows the time needed to generate the vertices. Note that our BlendSCAPE rendering makes use of multiple cores, while the SMPL rendering does not; this is why the system time for BlendSCAPE is higher than the wall-clock time. Note that here we are showing the cost of changing body shape. For most applications, this is done once and the shape is then held fixed. The cost of animating the mesh then comes from only the

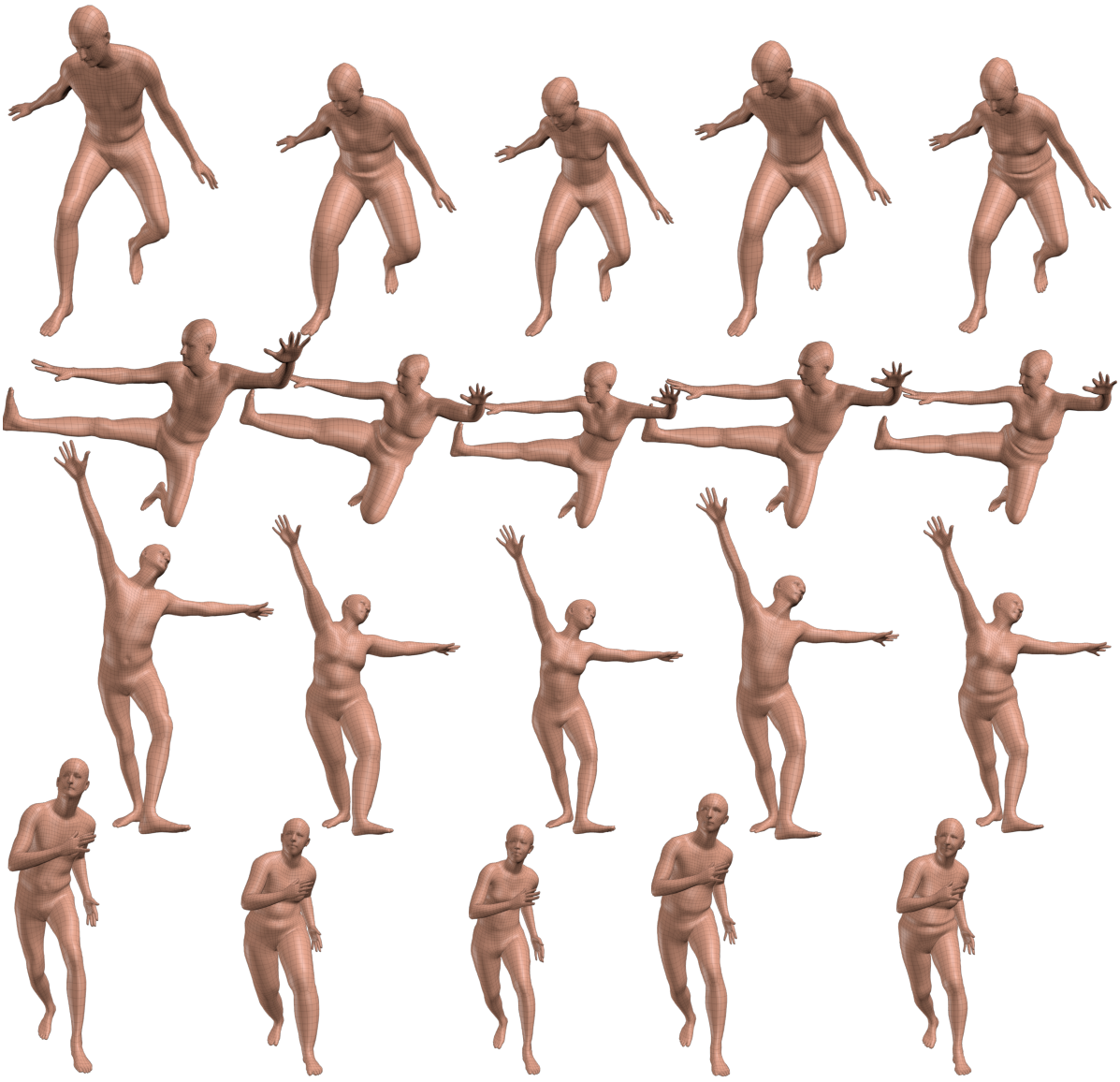


Figure 13: Animating SMPL. *Decomposition of SMPL parameters into pose and shape: Shape parameters, $\vec{\beta}$, vary across different subjects from left to right, while pose parameters, $\vec{\theta}$, vary from top to bottom for each subject.*

pose blend shapes; this cost corresponds to 0 shape coefficients.

For meshes with the same number of vertices, SCAPE will always be slower. In SMPL each blend shape is of size $3N$, requiring that many multiplications per shape. SCAPE uses triangle deformations with 9 elements per triangle and there are roughly twice as many triangles as vertices. This results in roughly a factor of 6 difference between SMPL and SCAPE in terms of basic multiplications.

5.5 Compatibility with Rendering Engines

Because SMPL is built on standard skinning, it is compatible with existing 3D animation software. In particular, for a given body shape, we generate the subject-specific rest-pose template mesh and skeleton (estimated joint locations) and we export SMPL as a rigged model with pose blend shapes in Autodesk's Filmbox (FBX) file format, giving cross-platform compatibility. The model loads as a typical rigged mesh and can be animated as usual in standard 3D animation software.

Pose blend weights can be precomputed, baked into the model, and exported as an animated FBX file. This kind of file can be loaded into animation packages and played directly. We have tested the animated FBX files in Maya, Unity, and Blender.

Pose blend weights can also be computed on the fly given the pose, $\vec{\theta}_t$, at time t. To enable this, we provide scripts that take the joint angles and compute the pose blend weights. We have tested loading and animating SMPL in Maya 2013, 2014 and 2015. The animator can animate the model using any of the conventional animation methods typically used in Maya. We will provide a Python script that runs inside Maya to apply blend-shape corrections to an animated SMPL model. The pose blend shape values can be viewed and/or edited manually within Maya if desired. We have also tested SMPL in Unity. In SMPL, the blend weights range from -1 to +1 while in Unity they range form 0 to 1. Consequently, we scale and recenter our weights for compatibility. For runtime computation of

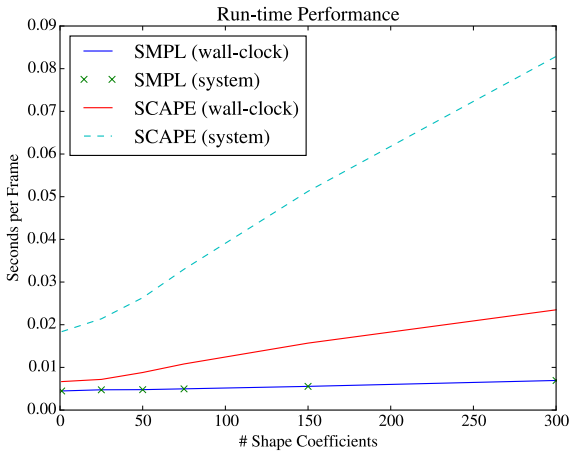


Figure 14: *Performance of SMPL and BlendSCAPE vary with the number of body shape coefficients used. Performance shown here is from a 2014 Macbook Pro.*

pose blend shape coefficients, we provide a C# script that the user can attach to SMPL's mesh game object.

The SMPL model with shape and pose blend shapes, and the evaluation meshes, are available for research purposes at `http://smpl.is.tue.mpg.de`.

6 DMPL: Dynamic SMPL

While SMPL models static soft-tissue deformations with pose it does not model dynamic deformations that occur due to body movement and impact forces with the ground. Given 4D registrations that contain soft-tissue dynamics, we fit them by optimizing only the pose of a SMPL model with a personalized template shape. Displacements between SMPL and the observed meshes correspond to dynamic soft-tissue motions. To model these, we introduce a new set of additive blend shapes that we call *dynamic blend shapes*. These additional displacements are correlated with velocities and accelerations of the body and limbs rather than with pose. We follow the approach of Dyna [Pons-Moll et al. 2015], using the same training data, and apply the ideas to our additive vertex-based model.

Let $\vec{\phi}_t = [\dot{\vec{\theta}}_t, \ddot{\vec{\theta}}_t, \mathbf{v}_t, \mathbf{a}_t, \vec{\delta}_{t-1}, \vec{\delta}_{t-2}]$ denote the dynamic control vector at time t. It is composed of pose velocities and accelerations $\dot{\vec{\theta}}_t, \ddot{\vec{\theta}}_t \in \mathbb{R}^{|\vec{\theta}|}$, root joint velocities and accelerations $\mathbf{v}_t, \mathbf{a}_t \in \mathbb{R}^3$ and a history of two vectors of predicted dynamic coefficients $\vec{\delta}_{t-1}, \vec{\delta}_{t-2} \in \mathbb{R}^{|\vec{\delta}|}$, described below.

We extend our linear formulation from Sec. 3 and simply add the dynamic blend shape function, $B_D(\vec{\phi}_t, \vec{\beta})$, to the other blend shapes in the rest pose before applying the skinning function. The shape in the zero pose becomes

$$T_D(\vec{\beta}, \vec{\theta}_t, \vec{\phi}_t) = \bar{\mathbf{T}} + B_S(\vec{\beta}) + B_P(\vec{\theta}_t) + B_D(\vec{\phi}_t, \vec{\beta}), \quad (16)$$

as illustrated in Fig. 15. Here, $B_D(\vec{\phi}_t, \vec{\beta})$ takes as input the dynamic control vector at time t, and shape coefficients $\vec{\beta}$, and predicts vertex offsets in the rest pose.

Whereas in [Pons-Moll et al. 2015] dynamic deformations are modeled using triangle deformations, DMPL models deformations in vertex space. We build male and female models using roughly $40,000$ registered male and female meshes from [Dyn 2015]. We compute the pose in each frame and the displacements between SMPL and the registration. Using PCA, we obtain a mean and the dynamic blend shapes, $\mu_D \in \mathbb{R}^{3N}$ and $\mathcal{D} \in \mathbb{R}^{3N \times |\vec{\delta}|}$ respectively. We take $|\vec{\delta}| = 300$ principal components as in Dyna. Dynamic deformations vary significantly between subjects based on their body shape and fat distribution. To capture this, we train a model that depends on the body shape parameters $\vec{\beta}$ as in Dyna.

Dynamic blendshapes are then predicted using

$$B_D(\vec{\phi}_t, \vec{\beta}; \mathcal{D}) = \mu_D + \mathcal{D} f(\vec{\phi}_t, \vec{\beta}) \quad (17)$$

analogous to Eq. (22) in [Pons-Moll et al. 2015] where $f(\cdot)$ is a function that takes as input a dynamic control vector, $\vec{\phi}_t$, and predicts the vector of dynamic shape coefficients, $\vec{\delta}_t$. This formulation of soft-tissue displacements in terms of dynamic blend shapes means that, unlike Dyna, our model remains compatible with current graphics software. To animate the model, we only need a script to compute the coefficients, $\vec{\delta}_t = f(\vec{\phi}_t, \vec{\beta})$, from the pose sequence and body shape. We observe that the DMPL model produces soft-tissue dynamics that appear more realistic than those of Dyna. See the **Supplemental Video** for visualizations of the training data, dynamic blend shapes, and resulting animations.

7 Discussion

Why does it work? First, good quality training data is important. Here we use thousands of high-quality registered template meshes. Importantly, the pose training data spans a range of body shapes enabling us to learn a good predictor of joint locations. Second, training all the parameters (template shape, blend weights, joint regressor, shape/pose/dynamic blend shapes) to minimize vertex reconstruction error is important to obtain a good model. Here the simplicity of the model is an advantage as it enables training everything with large amounts of data.

In contrast to the scattered-data interpolation methods, we learn the blend shapes from a large set of training meshes covering the space of possible poses and learn a simpler function relating pose to blend-shape weights. In particular, our function is linear in the elements of the part rotation matrices. The larger support of the learned linear functions as opposed to radial basis functions allows the model to generalize to arbitrary poses; in addition the simple linear form makes it fast to animate in a game engine without baking in the weights. Because elements of a rotation matrix are constrained, the model cannot "blow up" when generalizing outside the training set.

SMPL is an additive model in vertex space. In contrast, while SCAPE also factors deformations into shape and pose deformations, SCAPE multiplies the triangle deformations. With SCAPE a bigger person will have bigger pose-dependent deformations even though these deformations are not learned for different body shapes. Despite this, in our experiments, the SCAPE approach is less accurate at generalizing to new shapes. Ideally one would have enough pose data from enough different people to learn a true body-shape-dependent pose deformation space. Our work with DMPL, where deformations depend on body shape, suggests that this is possible.

Why is it more accurate than BlendSCAPE? Models based on the statistics of triangle deformations have dominated the recent literature [Anguelov et al. 2005; Chen et al. 2013; Freifeld and Black 2012; Hasler et al. 2009]. Such models are not trained to reproduce their training registrations directly. Instead, they are trained

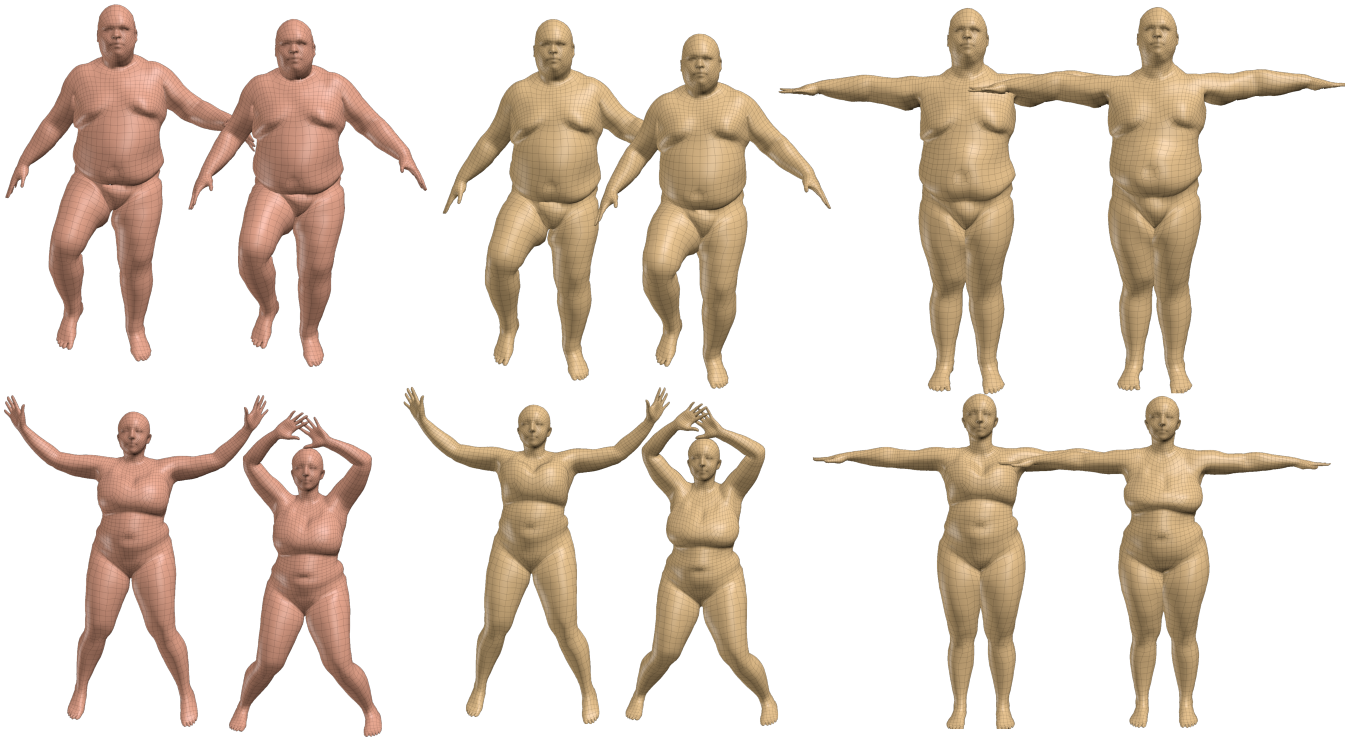
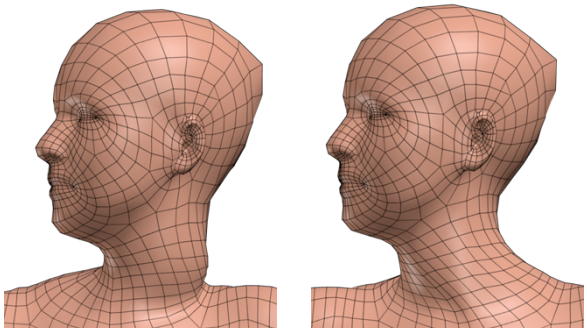


Figure 15: DMPL model of soft-tissue motion. *Above, two frames of a "running" sequence of a male subject from the Dyna dataset, below two frames of a "jumping jacks" sequence of a female subject from the Dyna dataset. From left to right: SMPL, DMPL, and the dynamic blend shapes added to the base body shape. While SMPL models deformations due to pose well it does not model dynamic deformations. DMPL predicts dynamic deformations from motion and body shape, resulting in more life like animations.*

to reproduce the local deformations that produced those registrations. Part of the tractability of training these models comes from the ability to train deformations independently across triangles. As a result, long range distances and relationships are not preserved as well as local relationships between vertices. We speculate that an advantage of vertex based models (such as SMPL and [Allen et al. 2006]) is that they can be trained to minimize the mean squared error between the model and training vertices. Theoretically one could train a SCAPE model to minimize vertex error in global coordinates, but the inner loop of the optimization would involve solving a least-squares problem to reconstruct vertices from the deformations. This would significantly increase the cost of optimization and make it difficult to train the model with large amounts of data.

Why has it not been done before? While we think the SMPL model is a natural way to extend blend skinning, we are unaware of any previous published versions. Unfortunately, the obvious implementation makes the pose blend shapes a linear function of $\vec{\theta}$. This does not work. The key to SMPL's performance is to make the blendshapes a linear function of the elements of $R^*(\vec{\theta})$. This formulation, sufficient training data, and a good optimization strategy make it possible to learn the model.

The closest work to ours is the pioneering work of Allen et al. [2006]. Their model is more complex than ours, using radial basis functions for scattered data interpolation, shape-dependent pose deformations, and a fixed set of carrying angles. Consequently training it is also complex and requires a good initialization. They had limited data and difficulty with overfitting so they restricted their body shape PCA space. As a result, the model did not generalize well to new shapes and poses. Our simpler model lets us learn it from large datasets and having more data makes the simple model perform well.



(a) *Euler-angles* **(b)** *SMPL parameterization*

Figure 16: Parameterizing pose blend shapes. *(a) Pose blend shapes parameterized by Euler angles cause significant problems. (b) our proposed parameterization allows the head to rotate in either direction with natural deformations.*

Other features for driving pose blend shapes. We experimented with driving pose blendshapes linearly from other features, such as raw $\vec{\theta}$, simple polynomials of $\vec{\theta}$, and trigonometric functions (sin, cos) of $\vec{\theta}$. None of these performed as well as our proposed formulation. Using raw $\vec{\theta}$ has serious limitations because the values vary between -π and π. Imagine a twist of the neck (Fig. 16), which produces negative and postive angles about the vertical axis. Standard LBS will cause the neck to shrink as it rotates in either direc-

tion. To counteract this, we need a blend shape that increases the neck volume no matter which direction it rotates. Unfortunately, if the blendshapes are trained to expand during rightwards rotation (to counteract LBS shrinkage), they would contract during leftward rotation.

In general one can think of replacing the raw rotations with any functions of rotations and using these to weight the blend shapes. An exhaustive search is impossible and other features may work as well as our method; for example, we did not experiment with normalized quaternions.

Our pose blend shapes function is also very different from scattered data interpolation methods like WPSD [Kurihara and Miyata 2004; Rhee et al. 2006], which use a discrete number of poses and associated corrections are interpolated between them using RBFs. In practice, a large number of poses might be needed to cover the pose space well. This makes animation slow since the closest key poses have to be found at run time.

Limitations. The pose-dependent offsets of SMPL are not dependent on body shape. It is surprising how well SMPL works without this, but the general approach would likely not work if we were to model a space of nonrealistic animated characters in which body part scales vary widely, or a space of humans that includes infants and adults.

This limitation could be addressed by training a more general function that would take elements of $R^*(\vec{\theta})$ together with $\vec{\beta}$ to predict the blend shape coefficients. Note that the dynamic blend shape coefficients do depend on body shape and therefore it should be possible to do the same for the pose blend shapes. This would not significantly complicate the model or run-time behavior but might require more training data.

As described, the SMPL model is a function of joint angles and shape parameters only: it does not model breathing, facial motion, muscle tension, or any changes independent of skeletal joint angles and overall shape. These could potentially be learned as additional additive blendshapes (as with DMPL) if the appropriate factored data is available (cf. [Tsoli et al. 2014]).

While we learn most model parameters, we do not learn them all. We manually define the segmentation of the template into parts, the topology of the mesh, and the zero pose. Theoretically these could also be learned but we expect only marginal improvements for significant effort.

Future work. SMPL uses 207 pose blend shapes. This could likely be reduced by performing PCA on the blend shapes. This would reduce the number of multiplications and consequently increase rendering speed. Also, our dynamic model uses PCA to learn the dynamic blend shapes but we could learn the elements of these blend shapes directly as we do for the pose blend shapes. Finally, here we fit our model to registered meshes but could fit SMPL to mocap marker data (cf. MoSh [Loper et al. 2014]), depth data, or video. We anticipate that optimizing the pose and shape of a SMPL-LBS model will be significantly faster than optimizing a SCAPE model of similar quality.

8 Conclusions

Our goal was to create a skeletally-driven human body model that could capture body shape and pose variation as well as, or better than, the best previous models while being compatible with existing graphics pipelines and software. To that end, SMPL uses standard skinning equations and defines body shape and pose blend shapes that modify the base mesh. We train the model on thousands of aligned scans of different people in different poses. The form of the model makes it possible to learn the parameters from large amounts of data while directly minimizing vertex reconstruction error. Specifically we learn the rest template, joint regressor, body shape model, pose blend shapes, and dynamic blend shapes. The surprising result is that, when BlendSCAPE and SMPL are trained on exactly the same data, the vertex-based model is more accurate and significantly more efficient to render than the deformation-based model. Also surprising is that a relatively small set of learned blend shapes do as good a job of correcting the errors of LBS as they do for DQBS. Using 4D registered meshes we extended SMPL to model dynamic soft-tissue deformations as a function of poses over time using an autoregressive model. SMPL can be exported as an FBX file and we make scripts available to animate the model in common rendering systems. This will allow anyone to realistically animate human bodies.

9 Acknowledgements

We thank F. Bogo for help with registration and 3D body modeling; B. Allen and B. Curless for information about their model; B. Mohler, J. Tesch, and N. Troje for technical discussion; A. Keller, E. Holderness, S. Polikovsky, and S. Streuber for help with data acquisition; J. Anning for voice recording and technical support; and A. Quiros Ramirez for web development.

Conflict-of-Interest Disclosure: MJB is a founder, shareholder, and member of the board of Body Labs Inc., which is commercializing body shape technology.

References

ALLEN, B., CURLESS, B., AND POPOVIĆ, Z. 2002. Articulated body deformation from range scan data. *ACM Trans. Graph. (Proc. SIGGRAPH) 21*, 3 (July), 612–619.

ALLEN, B., CURLESS, B., AND POPOVIĆ, Z. 2003. The space of human body shapes: Reconstruction and parameterization from range scans. *ACM Trans. Graph. (Proc. SIGGRAPH) 22*, 3, 587–594.

ALLEN, B., CURLESS, B., POPOVIĆ, Z., AND HERTZMANN, A. 2006. Learning a correlated model of identity and pose-dependent body shape variation for real-time synthesis. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SCA '06, 147–156.

ANGUELOV, D., SRINIVASAN, P., KOLLER, D., THRUN, S., RODGERS, J., AND DAVIS, J. 2005. SCAPE: Shape Completion and Animation of PEople. *ACM Trans. Graph. (Proc. SIGGRAPH 24*, 3, 408–416.

BARAN, I., AND POPOVIĆ, J. 2007. Automatic rigging and animation of 3D characters. *ACM Trans. Graph. (Proc. SIGGRAPH) 26*, 3 (July).

BOGO, F., ROMERO, J., LOPER, M., AND BLACK, M. J. 2014. FAUST: Dataset and evaluation for 3D mesh registration. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 3794 –3801.

CHANG, W., AND ZWICKER, M. 2009. Range scan registration using reduced deformable models. *Computer Graphics Forum 28*, 2, 447–456.

CHEN, Y., LIU, Z., AND ZHANG, Z. 2013. Tensor-based human body modeling. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 105–112.

2000. CMU graphics lab motion capture database. http://mocap.cs.cmu.edu. Accessed: 2012-12-11.

CORAZZA, S., AND GAMBARETTO, E., 2014. Automatic generation of 3D character animation from 3D meshes, Aug. 5. US Patent 8,797,328.

DE AGUIAR, E., THEOBALT, C., THRUN, S., AND SEIDEL, H.-P. 2008. Automatic conversion of mesh animations into skeleton-based animations. *Computer Graphics Forum 27*, 2, 389–397.

2015. Dyna dataset. http://dyna.is.tue.mpg.de/. Accessed: 2015-05-15.

FREIFELD, O., AND BLACK, M. J. 2012. Lie bodies: A manifold representation of 3D human shape. In *European Conf. on Computer Vision (ECCV)*, Springer-Verlag, A. Fitzgibbon et al. (Eds.), Ed., Part I, LNCS 7572, 1–14.

HASLER, N., STOLL, C., SUNKEL, M., ROSENHAHN, B., AND SEIDEL, H. 2009. A statistical model of human pose and body shape. *Computer Graphics Forum 28*, 2, 337–346.

HASLER, N., THORMÄHLEN, T., ROSENHAHN, B., AND SEIDEL, H.-P. 2010. Learning skeletons for shape and pose. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ACM, New York, NY, USA, I3D '10, 23–30.

HIRSHBERG, D., LOPER, M., RACHLIN, E., AND BLACK, M. 2012. Coregistration: Simultaneous alignment and modeling of articulated 3D shape. In *European Conf. on Computer Vision (ECCV)*, Springer-Verlag, A. F. et al. (Eds.), Ed., LNCS 7577, Part IV, 242–255.

JAMES, D. L., AND TWIGG, C. D. 2005. Skinning mesh animations. *ACM Trans. Graph. 24*, 3 (July), 399–407.

KAVAN, L., AND ŽÁRA, J. 2005. Spherical blend skinning: A real-time deformation of articulated models. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games*, ACM, New York, NY, USA, I3D '05, 9–16.

KAVAN, L., COLLINS, S., ŽÁRA, J., AND O'SULLIVAN, C. 2008. Geometric skinning with approximate dual quaternion blending. *ACM Transactions on Graphics (TOG) 27*, 4, 105:1–105:23.

KAVAN, L., COLLINS, S., AND O'SULLIVAN, C. 2009. Automatic linearization of nonlinear skinning. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games*, ACM, New York, NY, USA, I3D '09, 49–56.

KRY, P. G., JAMES, D. L., AND PAI, D. K. 2002. EigenSkin: Real time large deformation character skinning in hardware. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM, New York, NY, USA, SCA '02, 153–159.

KURIHARA, T., AND MIYATA, N. 2004. Modeling deformable human hands from medical images. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SCA '04, 355–363.

LAWSON, C. L., AND HANSON, R. J. 1995. *Solving least squares problems*. Classics in applied mathematics. SIAM, Philadelphia, PA. SIAM : Society of industrial and applied mathematics.

LE, B. H., AND DENG, Z. 2012. Smooth skinning decomposition with rigid bones. *ACM Trans. Graph. 31*, 6 (Nov.), 199:1–199:10.

LE, B. H., AND DENG, Z. 2014. Robust and accurate skeletal rigging from mesh sequences. *ACM Trans. Graph. 33*, 4 (July), 84:1–84:10.

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.

LOPER, M. M., AND BLACK, M. J. 2014. OpenDR: An approximate differentiable renderer. In *Computer Vision – ECCV 2014*, Springer, Heidelberg, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., vol. 8695 of *Lecture Notes in Computer Science*, 154–169.

LOPER, M. M., MAHMOOD, N., AND BLACK, M. J. 2014. MoSh: Motion and shape capture from sparse markers. *ACM Trans. Graph., (Proc. SIGGRAPH Asia) 33*, 6 (Nov.), 220:1–220:13.

MERRY, B., MARAIS, P., AND GAIN, J. 2006. Animation space: A truly linear framework for character animation. *ACM Trans. Graph. 25*, 4 (Oct.), 1400–1423.

MILLER, C., ARIKAN, O., AND FUSSELL, D. 2010. Frankenrigs: Building character rigs from multiple sources. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ACM, New York, NY, USA, I3D '10, 31–38.

MOHR, A., AND GLEICHER, M. 2003. Building efficient, accurate character skins from examples. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 562–568.

NOCEDAL, J., AND WRIGHT, S. J. 2006. *Numerical Optimization*, 2nd ed. Springer, New York.

PONS-MOLL, G., ROMERO, J., MAHMOOD, N., AND BLACK, M. J. 2015. Dyna: A model of dynamic human shape in motion. *ACM Transactions on Graphics, (Proc. SIGGRAPH) 34*, 4 (July), 120:1–120:14.

RHEE, T., LEWIS, J., AND NEUMANN, U. 2006. Real-time weighted pose-space deformation on the GPU. *EUROGRAPHICS 25*, 3.

ROBINETTE, K., BLACKWELL, S., DAANEN, H., BOEHMER, M., FLEMING, S., BRILL, T., HOEFERLIN, D., AND BURNSIDES, D. 2002. Civilian American and European Surface Anthropometry Resource (CAESAR) final report. Tech. Rep. AFRL-HE-WP-TR-2002-0169, US Air Force Research Laboratory.

ROUET, C., AND LEWIS, J., 1999. Method and apparatus for creating lifelike digital representations of computer animated objects by providing corrective enveloping, Mar. 16. US Patent 5,883,638.

SCHAEFER, S., AND YUKSEL, C. 2007. Example-based skeleton extraction. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SGP '07, 153–162.

SEO, H., CORDIER, F., AND MAGNENAT-THALMANN, N. 2003. Synthesizing animatable body models with parameterized shape modifications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SCA '03, 120–125.

TSOLI, A., MAHMOOD, N., AND BLACK, M. J. 2014. Breathing life into shape: Capturing, modeling and animating 3D human

breathing. *ACM Trans. Graph., (Proc. SIGGRAPH) 33*, 4 (July), 52:1–52:11.

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. (Proc. SIGGRAPH) 26*, 3 (July).

WEBER, O., SORKINE, O., LIPMAN, Y., AND GOTSMAN, C. 2007. Context-aware skeletal shape deformation. *Computer Graphics Forum 26*, 3 (Sept.), 265–274.

A Appendix

A.1 Mathematical Notation

We summarize our notation here and in Table 1. Matrices $\mathcal{A} \in \mathbb{R}^{n \times m}$ are denoted with math calligraphic typeface. Vectors $\mathbf{A} \in \mathbb{R}^m$ are denoted with uppercase boldface, expect for the special case of 3-vectors, which are denoted with lower case $\mathbf{a} \in \mathbb{R}^3$ to distinguish a particular vertex from a vector of concatenated vertices. The notation $A() : \mathbb{R}^m \mapsto \mathbb{R}^n$ is used to denote a function that maps vectors in an m-dimensional space to vectors in n-dimensional space. Typically, indices are used as follows: j iterates over mesh registrations, k iterates over joint angles and i iterates over subjects, and t denotes time.

Table 1: *Table of Notation*

		Model generation functions
W	$\triangleq$	Skinning function
M	$\triangleq$	SMPL function
B_P	$\triangleq$	Pose blendshapes function
B_S	$\triangleq$	Shape blendshapes function
B_D	$\triangleq$	Dynamic blendshapes function
J	$\triangleq$	Joint regressor: Predicts joints from surface
		Model input parameters (controls)
$\vec{\beta}$	$\triangleq$	Shape parameters
$\vec{\theta}$	$\triangleq$	Pose parameters
$\vec{\omega}$	$\triangleq$	Scaled axis of rotation; the 3 pose parameters corresponding to a particular joint
$\vec{\phi}$	$\triangleq$	Dynamic control vector
$\vec{\delta}$	$\triangleq$	Dynamic shape coefficients
$\vec{\theta}^*$	$\triangleq$	Zero pose or rest pose; the effect of the pose blendshapes is zero for that pose
		Model parameters (parameters learned)
$\mathcal{S}$	$\triangleq$	Shape blendshapes
$\mathcal{P}$	$\triangleq$	Pose blendshapes
$\mathcal{W}$	$\triangleq$	Blendweights
$\mathcal{J}$	$\triangleq$	Joint regressor matrix
$\bar{\mathbf{T}}$	$\triangleq$	Mean shape of the template
		Training data
$\mathbf{V}$	$\triangleq$	A registration
$\mathbf{V}^P$	$\triangleq$	Pose dataset registration
$\mathbf{V}^S$	$\triangleq$	Shape dataset registration
$\hat{\mathbf{T}}^P$	$\triangleq$	Pose dataset subject shape; body vertices in the template pose
$\hat{\mathbf{J}}^P$	$\triangleq$	Pose dataset subject joint locations in the template pose
$\hat{\mathbf{T}}^P_\mu$	$\triangleq$	Mean shape of a pose subject; body vertices in the template pose
$\hat{\mathbf{T}}^S$	$\triangleq$	Shape dataset subject shape; body vertices in the template pose
$\hat{\mathbf{T}}^S_\mu$	$\triangleq$	Mean shape of a subject in the shape dataset; body vertices in the template pose