Building Helper Bone Rigs from Examples

Tomohiko Mukai[*]
Tokai University

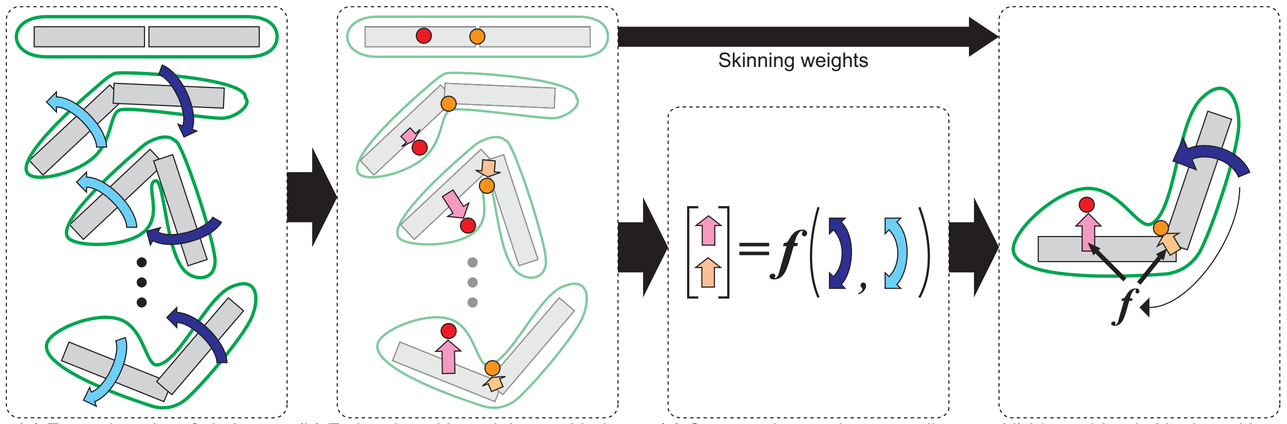

Skinning weights
(a) Example pairs of skeleton pose and skin shape
(b) Estimating skin weights and helper bone rotation and translation
(c) Constructing motion controller of helper bone
(d) Linear blend skinning with procedural helper bone control
f
f

Figure 1: *Example-based helper bone rigging for synthesizing a variety of skin deformations. (a) Multiple pairs of primary skeleton pose and desirable skin shapes are provided as example data. (b) Optimal rotation and translation of helper bones for each example are estimated using an iterative optimization. (c) The helper bone motion controller is constructed as a regression function that maps the primary skeleton pose to the helper bone transformations. (d) Artifact-free, stylized skin deformation can be synthesized in real-time using linear blend skinning (LBS) with the procedurally controlled helper bones.*

Abstract

Helper bone system has been widely used in real-time applications to synthesize high-quality skin deformation with linear blend skinning. Even though this technique provides a flexible yet efficient synthesis for a variety of expressive skin deformations, rigging with helper bones is still a labor-intensive process. In this study, we propose a novel method for building helper bone rigs from examples. We used multiple pairs of skeleton pose and desired skin shapes for our system. First, the system estimates the optimal skinning weights and helper bone transformations to reconstruct each example shape. Next, we construct a regression model which maps a primary skeleton pose to the helper bone transformations. The regression model enables a procedural control over the helper bones according to the primary skeleton. This is done at a lower computational cost and memory footprint. In addition, artists can edit the regression coefficient of the helper bone controller to modify deformation behavior. We demonstrate our system's potential by synthesizing stylized skin deformations in real-time.

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

Keywords: linear blend skinning, rigging, helper bone, skinning decomposition

[*]e-mail:tmki@acm.org

1 Introduction

Linear blend skinning (LBS) has been widely used for broad range of interactive applications such as games. Most real-time graphics engines support the LBS algorithm because of its simplicity and efficiency. However, this algorithm has so-called candy-wrapper and elbow-collapse artifacts, which cause an unwanted shrink in the skin mesh. One practical solution to minimize the LBS artifacts is to add extra bones, called helper bones. The helper bones work with the primary character skeleton to fix the collapse and crease in the skin mesh. An animation engine synthesizes the movement of the helper bones rather than a baked animation. This technique is used because the character movement is often synthesized during runtime process using some procedural techniques like inverse kinematics. The helper bone controller is designed as a polynomial function that maps the primary skeleton pose to the rotation, translation, and scale of the helper bone. This technique minimizes the LBS artifacts and stylizes skin deformations. Figure 1(a), shows an example of a limb model whose primary skeleton consists of a shoulder and an elbow. The two helper bones are inserted into the character rig, and are controlled with the shoulder and elbow motions. This minimizes the elbow-collapse artifact and emulates the behavior of the biceps as shown in Figure 1(d).

Even though the helper bone system requires computational cost to execute the bone controller, this extra cost is permissible for practical productions. This allowance is due to the significant improvement in the quality of the skin deformation while maintaining the real-time performance. In addition, since rig artists set up the helper bones, animators only have to create primary skeleton motions. They do not require prior knowledge about the helper bone system, which is compatible with common animation workflows.

The helper bone system poses one practical issue of unintuitive rigging process. As helper bones rarely have anatomical meaning, it is difficult to design their transformations and skinning weights. In addition, designing a helper bone controller is further unintuitive because the controller is often designed as a polynomial function. Rig artists have to specify the polynomial order, polynomial coefficients, and pose variables of the primary skeleton that must be used. Hence, several days of manual labor are required to build a production quality character rig with helper bones by a trial-and-error method.

In this study, we propose an example-based method to build character rigs with helper bones. We use a two-step algorithm to insert helper bones into a pre-designed primary skeleton rig using example pairs of primary skeletal pose and desirable skin shape. In the first step, our system estimates the optimal skinning weights and helper bone transformation for each example as shown in Figure 1(b). We used a modified version of the skinning decomposition algorithm [Le and Deng 2012; Le and Deng 2014] to incrementally insert rigid helper bones into the character rig. In the second step, the helper bone movement is calculated by a polynomial function of the primary skeleton movement. This is displayed in Figure 1(c). Our method allows artists to modify the polynomial coefficients for editing the skin deformation. To the best of our knowledge, this is the first method proposed to automate the helper bone rigging. We believe that the proposed system is compatible with common production workflows.

2 Related Work

Several methods have been proposed to synthesize high-quality skin deformation in real-time. The LBS model computes the deformed vertex position by transforming each vertex through a weighted combination of bone transformation matrices [Magnenat-Thalmann et al. 1988]. Multi-weight enveloping [Wang and Phillips 2002; Merry et al. 2006] extends the LBS model by adding a weight to each of the matrix elements. The non-linear skinning technique uses a dual quaternion instead of a transformation matrix to overcome the LBS artifacts. However, while bending, a side effect called the bulging artifact occurs [Kavan et al. 2007]. Several hybrid approaches have been proposed to blend bone transformations with lesser artifacts. Kavan et al.[2012] proposed a blending scheme that decomposes a bone rotation into a swing and twist component, and then separately blends each component using different algorithms. The stretchable and twistable bone model [Jacobson and Sorkine 2011] uses different weighting functions for scaling and bone twisting, respectively. These methods successfully synthesize the artifact-free skin deformation, and do not discuss the stylized skin deformation such as muscle-skin deformation.

Morh and Gleicher [2003] had introduced the basic concept of a helper bone system. In their work, helper bones were generated by subdividing primary bones. In this technique, the bone scaling was procedurally controlled according to the twist angle of the primary bone thus minimizing the candy-wrapper artifact. This technique has been extended into the gaming industry [Parks 2005; Kim and Kim 2011]. In these real-time works, the scale, rotation and translation of the helper bones are manipulated. This method minimizes the LBS artifacts while producing stylized skin deformations such as muscle bulging and skin sliding. Rig artists have to manually build the helper bone rig and its motion controller by trial-and-error. To avoid this manual effort, several methods have been proposed to optimize the skinning weights using a static shape [Baran and Popović 2007; Jacobson et al. 2011], or a set of multiple shapes [Wang and Phillips 2002; Mohr and Gleicher 2003; Miller et al. 2011]. However, there is no automated method to optimize the helper bone transformations.

Our method is closely related to the skinning decomposition technique. This method extracts optimal bone transformations and skinning weights from a set of example shapes. The skinning mesh animation model [James and Twigg 2005] introduced a basic concept for extracting bone transformations from a mesh animation. Kavan et al. [2010] used a dimensionality reduction technique to efficiently solve the skinning decomposition problem. The smooth skinning decomposition with a rigid bone (SSDR) model introduced a rigidity constraint on the bone transformation [Le and Deng 2012]. The SSDR model was later extended to extract hierarchically structured bone transformations from a mesh sequence [Le and Deng 2014]. Our method is different from these techniques. In this method, we assume that the hierarchical structure of the primary skeleton is designed by artists. We use several optimization techniques of the SSDR model to insert helper bones into a primary skeleton rig.

Scattered data interpolation, such as pose-space deformation (PSD) models, is another approach that can be used to synthesize skin deformation from example shapes [Lewis et al. 2000; Sloan et al. 2001; Kurihara and Miyata 2004]. The PSD model uses a radial basis function interpolation to blend example shapes according to the skeleton pose. This technique produces high-quality skin animations using an intuitive designing operation. However, the PSD model requires a runtime engine to store all example data in memory. In addition, the computational cost of the PSD increases proportionally to the number of examples. Hence, many example shapes cannot be used in a real-time system that has limited memory capacity.

The data-driven skinning method uses a regression model to map a skeleton pose to vertex positions of a skin mesh [Park and Hodgins 2008]. This method constructs a regression model from a set of example skeleton poses and skin shapes. However, the created model cannot be easily edited by hand. This limitation is due to the number of regression coefficients that are equal to the number of dense motion-capture markers. In comparison, our method constructs a fewer number of bone controllers as polynomial functions. This allows artists to manually modify the polynomial coefficients and edit the deformation behavior.

3 Overview

Given a primary skeleton with D bones, the global transformation matrix and bind pose matrix of d-th bone are denoted as 4×4 homogeneous matrices $\mathbf{G}_d$ and $\mathbf{B}_d$, $d = \{1, \cdots, D\}$, respectively. The skinning matrix is computed with a product of the inverse bind matrix and the global matrix as $\mathbf{S}_d = \mathbf{B}_d^{-1}\mathbf{G}_d$. The global transformation matrix $\mathbf{G}_d$ can be decomposed into a product of the local transformation matrix $\mathbf{L}_d$ and the parent's global transformation matrix as $\mathbf{G}_d = \mathbf{L}_d\mathbf{G}_{p(d)}$ where $p(d)$ is the parent of d-th bone. Let us indicate the bind position and skinning weights of j-th vertex as $\bar{\mathbf{v}}_j$ and $\{w_{j,d}\}$, respectively. The deformed vertex position $\mathbf{v}_j$ is computed as:

$$\mathbf{v}_j = \sum_d w_{j,d}\bar{\mathbf{v}}_j\mathbf{S}_d \tag{1}$$

where $\sum_d w_{j,d} = 1$ is satisfied. Now, we add H helper bones, where the skinning matrix of h-th helper bone are denoted as $\hat{\mathbf{S}}_h$, $h = \{1, \cdots, H\}$. The deformed vertex position $\mathbf{v}_j$ is computed with the skinning weights $\{\hat{w}_{j,h}\}$ as:

$$\mathbf{v}_j = \sum_d w_{j,d}\bar{\mathbf{v}}_j\mathbf{S}_d + \sum_h \hat{w}_{j,h}\bar{\mathbf{v}}_j\hat{\mathbf{S}}_h \tag{2}$$

where $\sum_d w_{j,d} + \sum_h \hat{w}_{j,h} = 1$ is satisfied. Given a set of N pairs of an example shape and a primary skeleton pose $\{\tilde{\mathbf{v}}_{j,n}\}$, $\{\tilde{\mathbf{G}}_{d,n}\}$, $n = \{1, \cdots, N\}$, our problem is formulated as a constrained least square problem that minimizes the squared reconstruction error between the example shape and skin mesh with respect to the skinning weights $\{w_{j,d}\}$, $\{\hat{w}_{j,h}\}$ and the skinning matrices $\{\hat{\mathbf{S}}_{h,n}\}$ as:

$$\begin{aligned}\min_{\{w_{j,d}\},\{\hat{w}_{j,h}\},\{\hat{\mathbf{S}}_{h,n}\}} \quad & \sum_n \sum_j \Big| \tilde{\mathbf{v}}_{j,n} - \sum_d w_{j,d}\bar{\mathbf{v}}_j \mathbf{B}_d^{-1}\tilde{\mathbf{G}}_{d,n} \\ & \qquad - \sum_h \hat{w}_{j,h}\bar{\mathbf{v}}_j\hat{\mathbf{S}}_{h,n}\Big|_2^2 \qquad (3)\\ \text{subject to} \quad & \hat{\mathbf{S}}_{h,n} = \hat{\mathbf{R}}_{h,n}\hat{\mathbf{T}}_{h,n}, \quad \forall h,n \\ & w_{j,d} \geq 0, \quad \hat{w}_{j,h} \geq 0, \quad \forall j,d,h \\ & \sum_d w_{j,d} + \sum_h \hat{w}_{j,h} = 1, \quad \forall j \\ & \sum_d |w_{j,d}|_0 + \sum_h |\hat{w}_{j,h}|_0 \leq K, \quad \forall j\end{aligned}$$

where $|\cdot|_\alpha$ denotes L^α-norm. The first constraint is the rigidity constraint that imposes a skinning matrix $\hat{\mathbf{S}}_{h,n}$ to be a product of a rotation matrix $\hat{\mathbf{R}}_{h,n}$ and translation matrix $\hat{\mathbf{T}}_{h,n}$. The rigidity constraint is applied because it is compatible with common animation engines, all of which support an efficient processing of both transformations. The second, third, and fourth constraints are non-negativity, partition of unity, and sparsity constraints on the skinning weights. The sparsity constraint limits the maximum number of bone transformations that can be blended to K bones. This constrained least square problem is solved with an iterative algorithm, which is explained in the next section.

We then constructed a regression function f_h that maps the primary skeleton pose $\{\mathbf{L}_d\}$ to the helper bone transformation $\hat{\mathbf{S}}_h$ as:

$$\hat{\mathbf{S}}_h \approx f_h\left(\mathbf{L}_1, \mathbf{L}_2, \cdots, \mathbf{L}_D\right). \qquad (4)$$

We used a simple polynomial function for f_h to allow artists to manually edit the deformation behavior. The construction of the regression function is explained in detail in Section 5.

During the run-time process, first the skinning matrix of each helper bone is computed according to the primary skeleton pose. Second, the deformed vertex position is efficiently computed in the standard graphics pipeline with linear blend skinning.

4 Per-example Optimization of Helper Bone Transformations

Optimal rigid transformations of helper bones are first estimated for each example shape using the optimization procedure that is summarized in Algorithm 1. Using the example data and the number of helper bones, our system inserts helper bones into the character rig in an incremental manner. Then, the helper bone transformations for each example and skinning weights are optimized using an iterative method. The overall procedure is similar to the SSDR model, where the skinning weights and bone transformations are alternately optimized by subdividing the optimization problem (Equation 3) into sub-problems of bone insertion, skinning weight optimization, and bone transformation optimization. We used the optimization techniques from the SSDR model to solve these three sub-problems. The main difference here is that the SSDR model does not have prior information about the transformable bones, but only information about the number. Hence, the SSDR model applies a clustering technique to simultaneously estimate an initial bone configuration. In our method, the primary skeleton and its example poses are given in the problem. This method inserts helper bones using the incremental optimization with a hard constraint on the primary bone transformation.

Algorithm 1 Optimization of helper bone transformations and skinning weights

```
Input: $\{\bar{\mathbf{v}}_j\}$, $\{\mathbf{B}_d\}$, $\{\tilde{\mathbf{v}}_{d,n}\}$, $\{\tilde{\mathbf{G}}_{d,n}\}$, $H$
Output: $\{\hat{\mathbf{S}}_{h,n}\}$, $\{w_{j,d}\}$, $\{\hat{w}_{j,h}\}$
 1: $\{\hat{\mathbf{S}}_{h,n}\} = I, \forall h,n, \quad \{\hat{w}_{j,h}\} = 0, \forall j,h$
 2: Initialize $\{w_{j,d}\}$
 3: repeat
 4:     Insert a new helper bone
 5:     Update helper bone transformations $\{\hat{\mathbf{S}}_{h,n}\}$
 6:     Update skinning weights $\{w_{j,d}\}$ and $\{\hat{w}_{j,h}\}$
 7:     Remove insignificant helper bones
 8: until The number of inserted helper bones is reached
 9: repeat
10:     Update helper bone transformations $\{\hat{\mathbf{S}}_{h,n}\}$
11:     Update skinning weights $\{w_{j,d}\}$ and $\{\hat{w}_{j,h}\}$
12: until The error threshold is reached
```

In the following sections, we will explain in detail the skinning weight update step (line 2, 6, and 11 in Algorithm 1) in Section 4.1, the transformation update step (line 5 and 10) in Section 4.2 and the bone insertion step (line 4) in Section 4.3.

4.1 Skinning Weight Optimization

The skinning weights $\{w_{j,d}\}$ and $\{\hat{w}_{j,h}\}$ are optimized by fixing all bone transformations $\{\hat{\mathbf{S}}_{h,n}\}$ in Equation 3. The resulting optimization problem is rewritten as following per-vertex constrained least square problem as:

$$\begin{aligned}\min_{\mathbf{w}_j} \quad & \left|[\tilde{\mathbf{A}}\ \hat{\mathbf{A}}]\mathbf{w}_j - \mathbf{b}\right|_2^2 \qquad (5)\\ \text{subject to} \quad & \mathbf{w}_j \geq 0, \quad |\mathbf{w}_j|_1 = 1, \quad |\mathbf{w}_j|_0 \leq K, \quad \forall j\end{aligned}$$

where

$$\begin{aligned}\mathbf{w}_j &= \begin{bmatrix} w_{j,1} & \cdots & w_{j,D} & \hat{w}_{j,1} & \cdots & \hat{w}_{j,H}\end{bmatrix}^T \in \Re^{(D+H)}\\ \tilde{\mathbf{A}} &= \begin{bmatrix} (\bar{\mathbf{v}}_j\tilde{\mathbf{S}}_{1,1})^T & \cdots & (\bar{\mathbf{v}}_j\tilde{\mathbf{S}}_{D,1})^T \\ \vdots & \ddots & \vdots \\ (\bar{\mathbf{v}}_j\tilde{\mathbf{S}}_{1,N})^T & \cdots & (\bar{\mathbf{v}}_j\tilde{\mathbf{S}}_{D,N})^T\end{bmatrix} \in \Re^{3N\times D}\\ \hat{\mathbf{A}} &= \begin{bmatrix} (\bar{\mathbf{v}}_j\hat{\mathbf{S}}_{1,1})^T & \cdots & (\bar{\mathbf{v}}_j\hat{\mathbf{S}}_{H,1})^T \\ \vdots & \ddots & \vdots \\ (\bar{\mathbf{v}}_j\hat{\mathbf{S}}_{1,N})^T & \cdots & (\bar{\mathbf{v}}_j\hat{\mathbf{S}}_{H,N})^T\end{bmatrix} \in \Re^{3N\times H}\\ \mathbf{b} &= \begin{bmatrix} \tilde{\mathbf{v}}_{j,1} & \cdots & \tilde{\mathbf{v}}_{j,N}\end{bmatrix}^T \in \Re^{3N}.\end{aligned}$$

This problem is NP-hard due to the L^0-norm constraint $|\mathbf{w}_j|_0 \leq K$. Hence, we use an approximation solution proposed in [Le and Deng 2012]. We exclude the L^0-norm constraint from Equation 5, and the resulting quadratic programming (QP) problem is solved using a stock numerical solver. When the solution does not satisfy the L^0-norm constraint, the most effort K bones are selected, and the weights for other bones are set to zero. The final solution is obtained by solving the QP problem again with the selected K bones.

At the initialization step (line 2 of Algorithm 1) we assume that there is no prior information about the skinning weight, even though we can assign an artist-painted weight map as an initial guess.

4.2 Bone Transformation Optimization

The per-example optimization of the helper bone transformations is formulated as a constrained least square problem that can be solved with a block coordinate descent algorithm [Le and Deng 2012]. This algorithm optimizes the objective function (Equation 3) over one helper bone transformations at each sub-iteration, while fixing the other variables. In addition, transformation of the h-th helper bone for each example shape is optimized while fixing the skinning weights, the transformations of remaining $H-1$ helper bones, and the primary bones, at each sub-iteration. Each sub-problem becomes a per-example weighted absolute orientation problem given by:

$$\begin{aligned}\min_{\hat{\mathbf{R}}_{h,n},\hat{\mathbf{T}}_{h,n}} \quad & \sum_j \Big| \tilde{\mathbf{v}}_{j,n} - \sum_d w_{j,d}\bar{\mathbf{v}}_j \tilde{\mathbf{S}}_{d,n} \\ & - \sum_{i,i\neq h} \hat{w}_{j,i}\bar{\mathbf{v}}_j \hat{\mathbf{R}}_{i,n}\hat{\mathbf{T}}_{i,n} \\ & - \hat{w}_{j,h}\bar{\mathbf{v}}_j \hat{\mathbf{R}}_{h,n}\hat{\mathbf{T}}_{h,n} \Big|_2^2 \qquad (6)\\ \text{subject to} \quad & \hat{\mathbf{R}}_{h,n}^T\hat{\mathbf{R}}_{h,n} = \mathbf{I}, \quad \det \hat{\mathbf{R}}_{h,n} = 1, \quad \forall h, n\end{aligned}$$

where the optimal $\hat{\mathbf{R}}_{h,n}$ and $\hat{\mathbf{T}}_{h,n}$ are obtained by the closed-form method proposed in [Le and Deng 2012].

4.3 Incremental Bone Insertion

Our technique uses an incremental method to insert a new helper bone into the region where the largest reconstruction errors occur. For example, if the new LBS causes an elbow-collapse artifact, a helper bone is generated around the elbow to minimize this artifact. First, our system searches for a vertex with the largest reconstruction error, which is computed as:

$$\arg\max_j \sum_n \Big| \tilde{\mathbf{v}}_{j,n} - \sum_d w_{j,d}\bar{\mathbf{v}}_j \tilde{\mathbf{S}}_{d,n} - \sum_h \hat{w}_{j,h}\bar{\mathbf{v}}_j \hat{\mathbf{S}}_{h,n} \Big|_2^2. \qquad (7)$$

Second, we compute a rigid transformation that closely approximates the displacement of the identified vertex and its one-ring neighbors, from their bind position by solving an absolute orientation problem [Horn 1987]. Then, a new helper bone is generated using the estimated transformation as its own transformation. Next, the skinning weights $\{w_{j,d}\}$ and $\{\hat{w}_{j,h}\}$, and the transformation matrix of all the helper bones are updated by solving the constrained least square problems. Finally, the system removes insignificant helper bones that have little influence on the skin deformation. Our current implementation removes the helper bones that influence less than four vertices. This process is repeated until the specified number of helper bones is achieved.

5 Helper Bone Controller Construction

The helper bone controller is constructed by learning a mapping from the primary bone transformations to the helper bone transformations. We use a linear regression model to represent the mapping. This simple representation allows artists to manually edit the bone controller in a post-processing step. In the following sections, we explain the transformation parameters that are used as variables for the regression model. The model construction algorithm that uses these parameters is explained in Section 5.3.

5.1 Selecting Coordinate System

The selection of coordinate systems to represent the bone transformation has a significant influence on the precision of the regression model. For example, we can learn a mapping from the local transformation matrices of primary bones $\{\mathbf{L}_d\}$ to the skinning matrix of helper bone $\hat{\mathbf{S}}_h$ as given in Equation 4. Alternatively, we can use a mapping from the skinning matrices of primary bones $\{\mathbf{S}_d\}$ to the global matrix of helper bones $\hat{\mathbf{G}}_h$. In most cases, we have observed that the local transformation shows simpler trajectories than the global or skinning transformation. For example, the helper bone inserted to offset the elbow-collapse artifact shows a linear trajectory in the elbow's local coordinate system. In comparison, if we use a global transformation to describe the helper bone movement, the motion trajectory becomes highly nonlinear. This is caused by a composite rotation of the ancestral bones in the skeleton hierarchy, which results in a significant decrease of the approximated accuracy. Hence, we use local transformation $\hat{\mathbf{L}}_h$ and $\mathbf{L}_d$ as both independent and dependent variables of the regression model.

The local transformation matrix $\hat{\mathbf{L}}_h$ is extracted from the skinning matrix $\hat{\mathbf{S}}_h$. This is the result of the per-example transformation optimization. By definition, the skinning matrix $\hat{\mathbf{S}}_h$ is decomposed into a product of transformation matrices as:

$$\hat{\mathbf{S}}_h = \hat{\mathbf{B}}_h^{-1}\hat{\mathbf{L}}_h\mathbf{G}_{p(h)} \qquad (8)$$

where $p(h) \in \{1, \cdots, D\}$ is the parent primary bone, and the bind pose matrix $\hat{\mathbf{B}}_h$ is an unknown rigid transformation matrix. Assuming that the helper bones local transformation is identity at the bind pose, the bind matrix $\hat{\mathbf{B}}_h$ is equal to that of the parent primary bone $\mathbf{B}_{p(h)}$ by the definition of forward kinematics. Therefore, we can uniquely extract the local transformation matrix by:

$$\hat{\mathbf{L}}_h = \mathbf{B}_{p(h)}\hat{\mathbf{S}}_h\mathbf{G}_{p(h)}^{-1}. \qquad (9)$$

The appropriate parent primary bone $p(h)$ is selected to minimize the approximation error. The detailed selection algorithm is further explained in Section 5.4.

5.2 Parameterizing Transformations

The extracted local matrix is parameterized with fewer variables to reduce the dimensionality of the regression problem. Using rigid transformation, the local transformation matrix $\hat{\mathbf{L}}_h$ can be parameterized using a combination of a translation vector $\hat{\mathbf{t}}_h \in \Re^3$ and bone rotation variables $\hat{\mathbf{r}}_h \in \Re^3$. We used exponential maps for the rotation variable $\hat{\mathbf{r}}_h$ [Grassia 1998]. This results in the transformation of $\hat{\mathbf{L}}_h$ to a six-dimensional pose vector form $[\hat{\mathbf{t}}_h \quad \hat{\mathbf{r}}_h] \in \Re^6$. In addition, the local transformation of primary bone $\mathbf{L}_d$ is parameterized by its animating variables. For example, when the primary bone has no translation or scale animation key, we can use three variables of the exponential map $\mathbf{r}_d$ to parameterize the bone transformation $\mathbf{L}_d$. Further, we can reduce the number of variables using the Euler angle representation for hinge and universal joints. However, to simplify this, we have assumed that each primary bone does not have a translation or scale key, and that a bone rotation is always represented by exponential maps.

5.3 Regression Model Construction

We have used a P-th order polynomial function as a regression model. The pose vector of each helper bone is approximated by:

$$\begin{aligned}\begin{bmatrix} \hat{\mathbf{t}}_h & \hat{\mathbf{r}}_h \end{bmatrix}^T &= f_h(\mathbf{L}_1, \mathbf{L}_2, \cdots, \mathbf{L}_D) \\ &= \mathbf{F}_h \begin{bmatrix} 1 & \mathbf{x}_1 & \cdots & \mathbf{x}_D \end{bmatrix}^T \qquad (10)\end{aligned}$$

where $\mathbf{x}_d \in \Re^{{}_4H_P - 1}$ is an independent variable vector that is composed of all variables of the P-th order polynomial of $\mathbf{r}_d$. For example, if we take $P = 2$, the independent variable vector from $\mathbf{r} = [r_1, r_2, r_3]$ is $\mathbf{x} = [r_1, r_2, r_3, r_1^2, r_2^2, r_3^2, r_1r_2, r_1r_3, r_2r_3]$. The regression matrix for h-th helper bone $\mathbf{F}_h \in \Re^{6\times(1+\sum_d \dim(\mathbf{x}_d))}$ is estimated from examples using the least square technique. In addition, we add a sparsity constraint to minimize the number of non-zero regression coefficient to generate a simpler model. The least square problem with the sparsity constraint can be formulated as a Lasso problem [Tibshirani 2011] given by:

$$\min_{\mathbf{F}_h} |\mathbf{Y}_h - \mathbf{F}_h \mathbf{X}|_2^2 + \lambda |\mathbf{F}_h|_1 \qquad (11)$$

where

$$\mathbf{X} = \begin{bmatrix} 1 & \mathbf{x}_{1,1} & \cdots & \mathbf{x}_{D,1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \mathbf{x}_{1,N} & \cdots & \mathbf{x}_{D,N} \end{bmatrix}^T \in \Re^{(1+\sum_d \dim(\mathbf{x}_d))\times N}$$

$$\mathbf{Y}_h = \begin{bmatrix} \hat{\mathbf{t}}_{h,1} & \hat{\mathbf{r}}_{h,1} \\ \vdots & \vdots \\ \hat{\mathbf{t}}_{h,N} & \hat{\mathbf{r}}_{h,N} \end{bmatrix}^T \in \Re^{6\times N}$$

and λ is the shrinkage parameter that controls the trade-off between model accuracy and the number of non-zero coefficients. Using a stock Lasso solver, we can efficiently solve this problem.

5.4 Parent Primary Bone Search

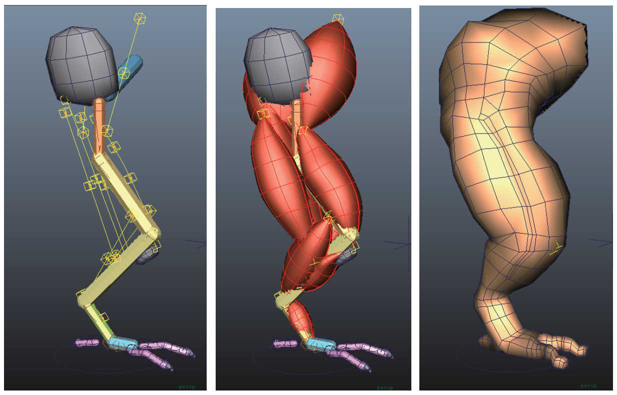
There is only one problem that remains: the selection of an appropriate parent bone $p(h)$ for each helper bone. This is a discrete optimization problem. Generally, since the number of primary bones is smaller, we can implement an exhaustive search to find the optimal one. Further, the best parent bone $p(h)$ can be identified by evaluating Equation 9, 10. Here, each primary bone can be used as $p(h)$, and the best one that minimizes the objective function (Equation 11) can be selected.

6 Experimental Results

We evaluated our system's approximation capability and computational performance using example datasets, which cannot be reconstructed with the LBS algorithm without helper bones. For all experiments, the parameter K, which is the maximum number of transformations to be blended, was fixed to a value equal to 4. The reconstruction error was evaluated using root mean squared (RMS) error of the vertex position. The optimization procedure of the helper bone transformation is parallelized over vertices, helper bones, or examples using Intel Threading Building Blocks. The computational timing was measured at 3.4 GHz on a Core i7-4770 CPU (8 logical processors) with 16 GB RAM.

6.1 Test Dataset

We used a muscle function from Autodesk Maya to synthesize an example skin shape from a skeleton pose. The muscle system emulates static muscle-skin deformation with skeletal pose. The muscle system also produces dynamic deformation which is caused by bone acceleration and inertia of muscles. For our experiment, we had used an earlier function because our method supports only static mapping from a skeleton pose to a skin shape. The test character model is a sample asset of a Maya tutorial [Autodesk] as shown in Figure 2. The leg model height is 200 cm at the bind pose. The skeleton has $D = 3$ animating bones and 5 degrees of freedom (DOFs) including hip swing and twist (3 DOFs), knee bend (1 DOF) and ankle bend (1 DOF). The eleven muscles expand and contract according to the movement of the primary skeleton. They drive the deformation of 663 vertices using the proprietary algorithm.



(a) Primary skeleton (b) Muscle (c) Skin mesh

Figure 2: *Character model used to create an example pose and skin shape. The skin deformation is driven by a primary skeleton and virtual muscle, which is a built-in function of Autodesk Maya.*

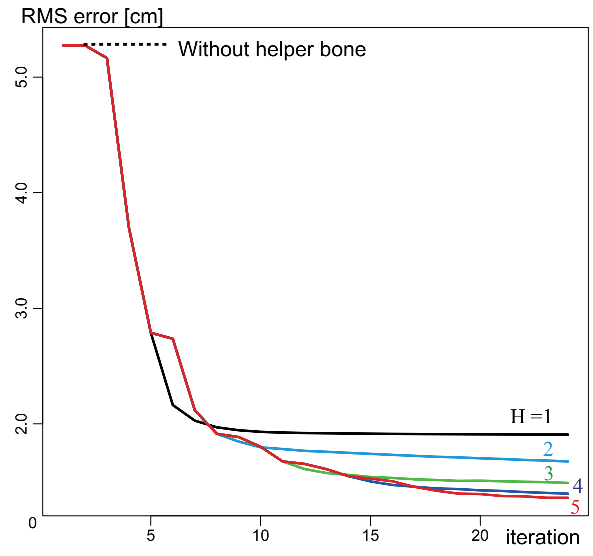
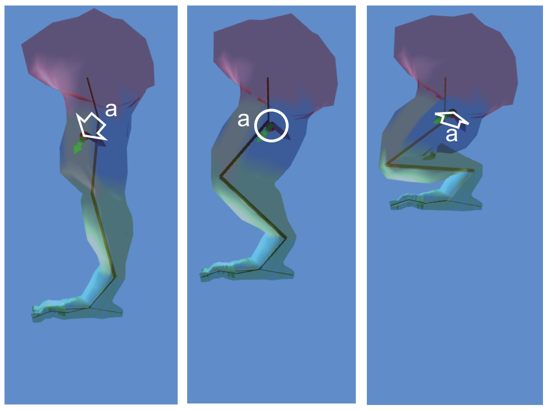


Figure 3: *Convergence of the reconstruction error according to the number of helper bones and iterations.*

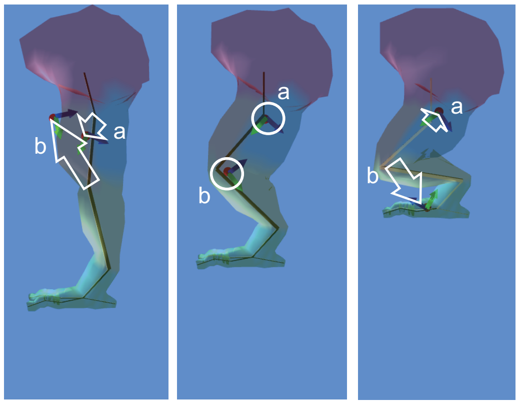
A test dataset was created by uniformly sampling the bone rotation of the primary skeleton every 20 degrees within each range of joint motion. Consequently, we created 6750 example pairs of skeleton pose and skin shape. This was done by discretizing the DOFs of the hip swing, hip twist, knee bend, and ankle bend into 6×6, 9, 5, and 5 levels, respectively.

6.2 Evaluating Optimized Bone Transformations

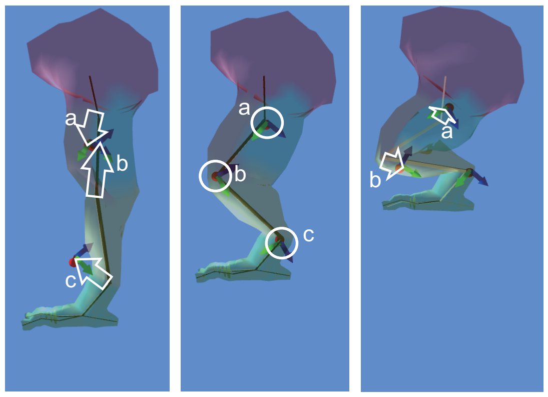
In the first experiment, different number of helper bones were inserted into the character rig while fixing the polynomial order $P = 2$ and the shrinkage parameter $\lambda = 0$. Figure 3 shows the convergence of the reconstruction error with the number of helper bones and the number of iterations. The reconstruction error decreased according to the number of helper bones. In addition, there were no significant differences between the reconstruction error of 4 helper bones and that of 5 helper bones. This result indicates that the approximation was almost converged at 4 helper bones. The reconstruction error monotonically decreased with the number of iterations, which demonstrates the stability of our system.


(a1) Leg extension (a2) Bind pose (a3) Leg bend

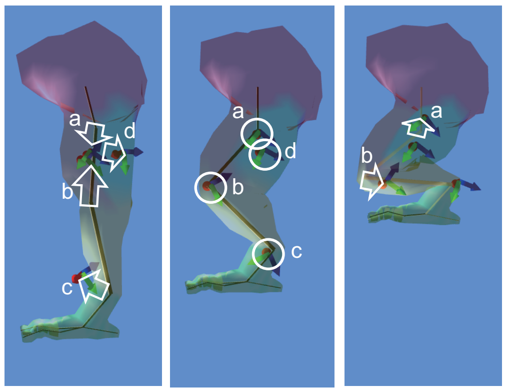
(a) 1 helper bone


(b1) Leg extension (b2) Bind pose (b3) Leg bend

(b) 2 helper bones


(c1) Leg extension (c2) Bind pose (c3) Leg bend

(c) 3 helper bones


(d1) Leg extension (d2) Bind pose (d3) Leg bend

(d) 4 helper bones

Figure 4: *Optimized character rigs using different number of helper bones. Each model shows a different helper bone behavior.*

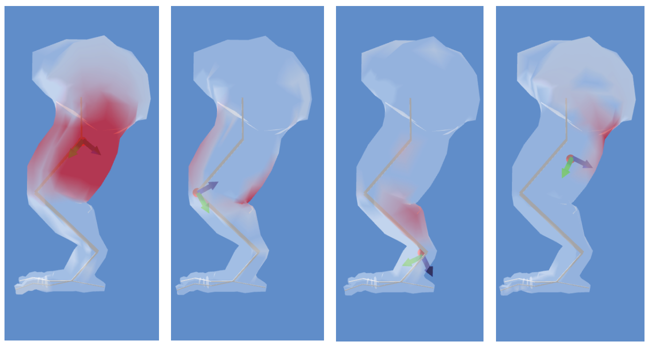


Figure 5: *Skinning weight map for each helper bone. The larger weight is indicated by a darker red area.*

Figure 4 shows optimized models using different number of helper bones. The center image of each screen shot shows the bind pose, and the left and right images show a leg stretching pose and bending pose, respectively. The helper bones *a*, *b* and *c* are located near the hip, knee, and ankle to minimize the LBS artifacts. The helper bone *d* is located in the thigh to emulate the muscle bulge. The skinning weight map for each helper bone is visualized in Figure 5. The helper bone *a* had a significant influence on a large area of the thigh, while the other helper bones had a lesser influence. This is the standard result of our incremental bone insertion algorithm where the first helper bone is inserted to offset the largest reconstruction error. This result would be useful for a level of detail (LOD) control for helper bones. We can suspend lesser effective bone controllers to reduce the redundant computational cost when a rendered character is small on the screen.

To build the rig with one helper bone, our system spent 0.17, 0.51 and 0.17 s per iteration for the bone insertion step, weight update step, and transformation update step, respectively. The total optimization time spent was about 15 s for 20 iterations. For the rig with 4 helper bones, the time recorded was 0.17, 0.82 and 0.72 s per iteration. The total time spent was about 32 s.

6.3 Evaluating Accuracy of Bone Controller

In the second experiment, we examined the approximation capability of a helper bone controller. We evaluated the increase of the

Table 1: *Statistics of the reconstruction error and the number of non- zero polynomial coefficients considering the polynomial order P and the shrinkage parameter λ.*

	Avg. # of non-zeros			RMS error [cm]		
λ	0	10	20	0	10	20
Linear	6	5.3	5.1	2.57	2.58	2.59
Quadratic	14	10.9	9.1	2.11	2.12	2.17
Cubic	26	18.4	15.1	2.03	2.07	2.11

RMS error that was caused by approximating the bone transformations with the regression model. In addition, we counted the number of non-zero polynomial coefficients using a different setting of the polynomial order P and the shrinkage parameter λ while fixing $H = 4$.

The results of the experiment are summarized in Table 1. The baseline RMS reconstruction error, which was measured after the per-example transformation optimization, was 1.36 cm. The increase ratio of the approximation error was within the range of 150% - 190%. The reconstruction error decreased according to the polynomial order, and there was no significant difference between the quadratic and cubic polynomials. Alternatively, the redundant polynomial terms were removed through the shrinkage parameter λ while minimizing the increase of the approximation error.

In this experiment, our system spent about 5 μs per frame to compute all the skinning matrices $\{\hat{\mathbf{S}}_h\}$ from the primary skeleton pose $\{\mathbf{L}_d\}$. In detail, 1 μs was spent to compose the independent variables $\{\mathbf{x}_d\}$ from the local transformation matrices $\{\mathbf{L}_d\}$. Using Equation 10, the computation of the regression model spent 1 μs for each helper bone. The former time increases proportional to the number of primary bones, and the latter time increases with the number of helper bones. We could further improve the performance by parallelizing the execution of bone controllers.

6.4 Test on Stylized Character Model

Our system was also evaluated on a more stylized character model as shown in Figure 6 (a). We edited the leg model by modifying the muscle parameter to amplify the muscle deformation. The skin mesh was smoothed to 8322 vertices by subdividing polygonal faces to evaluate detailed deformation. The example dataset was created by the same procedure as the first experiment. In this experimental setting, our system spent 2.6, 8.4, and 11.4 s per iteration for the bone insertion, weight update step, and transformation update step, respectively. The total time spent for 20 iterations was about 420 s. The increase in the computational time is due to the mesh smoothing.

First, we used 4 helper bones, and set $P = 2$ and $\lambda = 10.0$, respectively. After per-example transformation optimization, the RMS reconstruction error reconstruction error was 1.75 cm. After bone controller construction, the final RMS error was 2.90 cm. The results in Figure 6 (b), shows that the deformation required some more details of the example shape, such as the bulging of thigh muscles. Second, we used 8 helper bones expecting an improvement in the reconstruction accuracy. After per-example transformation optimization, the RMS error was improved to 1.23 cm. However, the final RMS error decreased slightly to 2.79 cm. As this number indicates, we cannot see a significant difference in the reconstructed skin shape, as depicted in Figure 6 (c).

This result shows that the regression approximation had canceled the accuracy improvement by the bone insertion. This occurs due to reconstruction accuracy, which is converged according to the number of helper bones as shown in Figure 3. On the contrary, the accumulated error of regression models, increases almost proportional to the same number. We might be able to minimize this problem by introducing an advanced regression model, such as Gaussian process models, to construct an accurate bone controller. However, the complicated regression model prevents intuitive manual editing and decreases efficiency. In our future work, we intend to investigate this exchange between the regression accuracy and edit ability of the bone controller.

7 Discussion and Future Work

This study proposed an example-based method to build a helper bone rig from example pairs of primary skeleton pose and skin shape. The optimization parameter includes the number of helper bones, polynomial order, and shrinkage parameter in the bone controller construction. This example-based method requires a lot of example data to construct a robust bone controller that can synthesize a wide variety of skin deformations, which still demands a lot of artist labor. However, several simulation methods have been proposed to generate physically valid skin deformations using heavy computations such as finite element methods [Li et al. 2013; Fan et al. 2014]. In addition, performance capture equipment has been developed to capture and construct a statistical skin deformation model [Neumann et al. 2013]. These state-of-the-art techniques will allow the mass production of many example skin shapes within a short time period.

Currently, we do not provide any guideline to create an example dataset. Even though, we have used the uniform sampling of joint DOFs to create example poses in the experiments, this simple method might generate many redundant examples. This method may even possibly fail to sample important poses and shapes. We plan to do further studies to identify a more artist-friendly workflow that can create a minimal example dataset. We believe that an active learning method [Cooper et al. 2007] could be a possible solution that allows artists to design example shapes using a step-by-step method.

Our future work includes an LOD control for the helper bones as mentioned in the previous section. The computation of the helper bone can be simplified, or skipped, when the rendered character is small on the screen. Although our bone controller uses less computational resources, such redundant computations should always be avoided in real-time applications. We intend to design an LOD control mechanism for helper bones.

In the future, we intend to produce a physically plausible secondary skin deformation, such as muscle jiggling, using the helper bone system. We believe that the helper bone is compatible for synthesizing such secondary motion at a lower computational cost. A simple solution is to use a physics simulation, but this might not be compatible for stable synthesis of stylized skin deformations. We will further explore an algorithm that can extract a controllable dynamic bone controller from example animations.

Acknowledgements

This study was supported in part by Research and Study Project of Tokai University Educational System Research Organization.

References

AUTODESK. *Maya Muscle Advanced Techniques*.

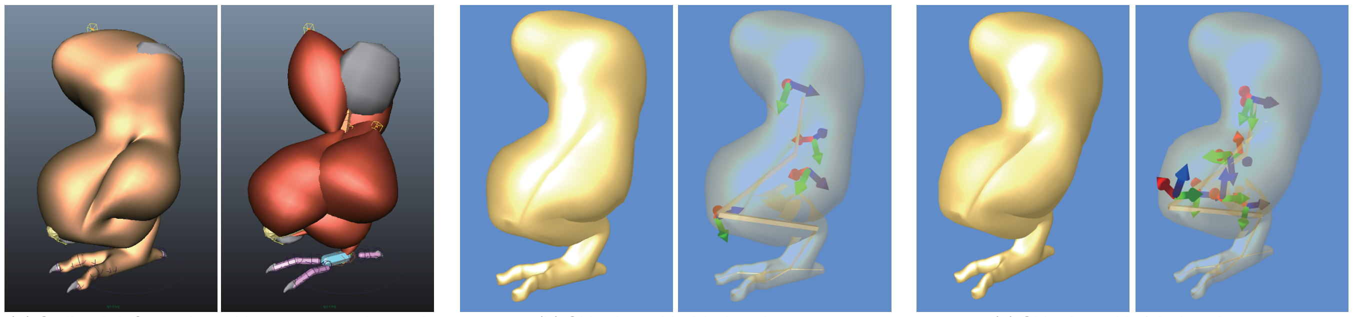

(a) Stylized deformation with exaggerated muscles
(b) Skinning with 4 helper bones
(c) Skinning with 8 helper bones

Figure 6: *Stylized character model with exaggerated muscles. We used 4 and 8 helper bones to approximate skin deformation.*

BARAN, I., AND POPOVIĆ, J. 2007. Automatic rigging and animation of 3d characters. *ACM Transactions on Graphics 26*, 3, Article 72.

COOPER, S., HERTZMANN, A., AND POPOVIĆ, Z. 2007. Active learning for real-time motion controllers. *ACM Transactions on Graphics 26*, 3, 5.

FAN, Y., LITVEN, J., AND PAI, D. K. 2014. Active volumetric musculoskeletal systems. *ACM Transactions on Graphics 33*, 4, 152.

GRASSIA, F. S. 1998. Practical parameterization of rotations using the exponential map. *Graphics Tools 3*, 3, 29–48.

HORN, B. K. P. 1987. Closed-form solution of absolute orientation using unit quaternions. *Journal of Optical Society of America A 4*, 4, 629–642.

JACOBSON, A., AND SORKINE, O. 2011. Stretchable and twistable bones for skeletal shape deformation. *ACM Transactions on Graphics 30*, 6, Article 165.

JACOBSON, A., BARAN, I., POPOVIĆ, J., AND SORKINE, O. 2011. Bounded biharmonic weights for real-time deformation. *ACM Transactions on Graphics 30*, 4, Article 78.

JAMES, D. L., AND TWIGG, C. D. 2005. Skinning mesh animations. *ACM Transactions on Graphics 24*, 3, 399–407.

KAVAN, L., AND SORKINE, O. 2012. Elasticity-inspired deformers for character articulation. *ACM Transactions on Graphics 31*, 6, Article 196.

KAVAN, L., COLLINS, S., ZARA, J., AND O'SULLIVAN, C. 2007. Skinning with dual quaternions. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics 2007*, 39–46.

KAVAN, L., SLOAN, P.-P., AND O'SULLIVAN, C. 2010. Fast and efficient skinning of animated meshes. *Computer Graphics Forum 29*, 2, 327–336.

KIM, J., AND KIM, C.-H. 2011. Implementation and application of the real-time helper-joint system. In *Game Developer Conference 2011*.

KURIHARA, T., AND MIYATA, N. 2004. Modeling deformable human hands from medical images. In *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2004*, 355–363.

LE, B. H., AND DENG, Z. 2012. Smooth skinning decomposition with rigid bones. *ACM Transactions on Graphics 31*, 6, Article 199.

LE, B. H., AND DENG, Z. 2014. Robust and accurate skeletal rigging from mesh sequences. *ACM Transactions on Graphics 33*, 4.

LEWIS, J. P., CORDNER, M., AND FONG, N. 2000. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proc. of SIGGRAPH 2000*, 165–172.

LI, D., SUEDA, S., NEOG, D. R., AND PAI, D. K. 2013. Thin skin elastodynamics. *ACM Transactions on Graphics 32*, 4, 49.

MAGNENAT-THALMANN, N., LAPERRIÈRE, R., AND THALMANN, D. 1988. Joint-dependent local deformations for hand animation and object grasping. In *Proceedings on Graphics interface '88*, 26–33.

MERRY, B., MARAIS, P., AND GAIN, J. 2006. Animation space: A truly linear framework for character animation. *ACM Transactions on Graphics 25*, 6, 1400–1423.

MILLER, C., ARIKAN, O., AND FUSSELL, D. S. 2011. Frankenrigs: Building character rigs from multiple sources. *IEEE Transactions on Visualization and Computer Graphics 17*, 8, 1060–1070.

MOHR, A., AND GLEICHER, M. 2003. Building efficient, accurate character skins from examples. *ACM Transactions on Graphics 22*, 3, 562–568.

NEUMANN, T., VARANASI, K., HASLER, N., WACKER, M., MAGNOR, M., AND THEOBALT, C. 2013. Capture and statistical modeling of arm-muscle deformations. *Computer Grahics Forum 32*, 2, 285–294.

PARK, S. I., AND HODGINS, J. K. 2008. Data-driven modeling of skin and muscle deformation. *ACM Transactions on Graphics 27*, 3, Article 96.

PARKS, J. 2005. Helper joints: Advanced deformations on runtime characters. In *Game Developers Conference 2005*.

SLOAN, P.-P. J., ROSE, C. F., AND COHEN, M. F. 2001. Shape by example. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics 2011*, 135–143.

TIBSHIRANI, R. 2011. Regression shrinkage and selection via the lasso: A retrospective. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 73*, 3, 273–282.

WANG, X. C., AND PHILLIPS, C. 2002. Multi-weight enveloping: Least-squares approximation techniques for skin animation. In *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 129–138.