Robust and Accurate Skeletal Rigging from Mesh Sequences

Binh Huy Le* Zhigang Deng†
University of Houston

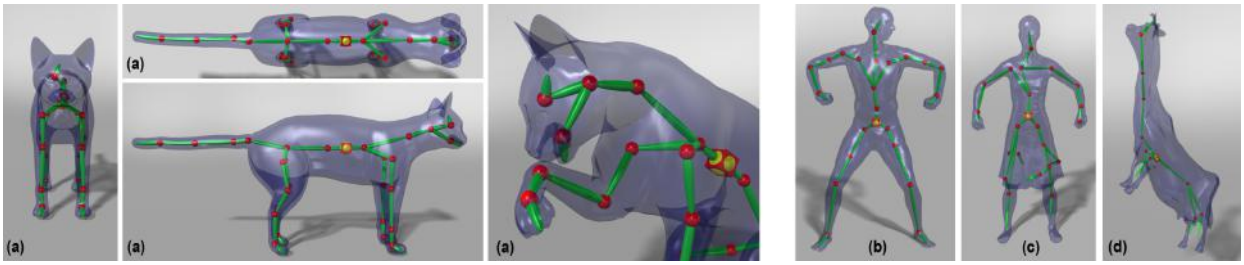

Figure 1: *Only using a single set of parameters, our example-based method can accurately rig various models such as quadrupled animals (a), humans (b, c), and highly deformable models (d). Our method can even generate bone structures for challenging parts, such as the mouth and the two ears of the cat (a), the skirt (b), and the elastic cow model (d). Our method is robust: using only 9 frames, it can generate a skeleton with 28 bones for the cat model (a); note that even though the given example poses of the cat model have asymmetric poses, it still can generate an almost symmetric skeleton without imposing any symmetry constraints.*

Abstract

We introduce an example-based rigging approach to automatically generate linear blend skinning models with skeletal structure. Based on a set of example poses, our approach can output its skeleton, joint positions, linear blend skinning weights, and corresponding bone transformations. The output can be directly used to set up skeleton-based animation in various 3D modeling and animation software as well as game engines. Specifically, we formulate the solving of a linear blend skinning model with a skeleton as an optimization with joint constraints and weight smoothness regularization, and solve it using an iterative rigging algorithm that (i) alternatively updates skinning weights, joint locations, and bone transformations, and (ii) automatically prunes redundant bones that can be generated by an over-estimated bone initialization. Due to the automatic redundant bone pruning, our approach is more robust than existing example-based rigging approaches. Furthermore, in terms of rigging accuracy, even with a single set of parameters, our approach can soundly outperform state of the art methods on various types of experimental datasets including humans, quadrupled animals, and highly deformable models.

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

Keywords: rigging, skeleton extraction, geometric deformation, skinning from examples, linear blend skinning

Links: DL PDF WEB

*e-mail: bhle2@cs.uh.edu
†e-mail:zdeng@cs.uh.edu

1 Introduction

Skeleton-based mesh deformation is a widely-used method for animating articulated creatures such as humans and animals. Setting up the skeleton-based animation (also known as *rigging*), however, often requires careful manual interventions in practice. Rigging a model currently consists of two main steps: building a hierarchical skeleton with rigid bones connected by joints, and skinning the 3D model to define how joint rotations and translations would propagate to the surface during animation. In practice, animators typically repeat the two steps many times to refine the model for best results. This trial-and-error method is costly and time-consuming, since its two steps are often done manually or semi-automatically.

The concept of using example poses (i.e., a sequence of deformed mesh frames) for rigging [Schaefer and Yuksel 2007; de Aguiar et al. 2008a; Hasler et al. 2010] has become increasingly practical and useful in recent years. In particular, deformed mesh sequences can be soundly reconstructed by performance capture [de Aguiar et al. 2008b; Vlasic et al. 2008; Vlasic et al. 2009; Stoll et al. 2010] or by dense motion capture based on commercial systems [Park and Hodgins 2006]. Since the skeleton extracted from example poses is typically compatible with game engines and popular animation software such as Autodesk Maya, it can be directly used for various animation editing, compression, and rendering applications, which helps to reduce production cost in industry practice assuming example poses are available.

The essential idea of these example-based rigging methods [Schaefer and Yuksel 2007; de Aguiar et al. 2008a; Hasler et al. 2010] is to first perform motion driven clustering to extract rigid bone transformations, then estimate joint locations and bone lengths using linear or non-linear least squares, and finally optimize the bone transformations and skinning weights. However, they have the following limitations: First, nearly-rigid parts cannot be identified perfectly since motion driven clustering algorithms model neither skin blending nor skeletal structure. As such, this step would either require non-trivial model-specific parameter tuning or result in an unrobust skeleton. Second, each step in this pipeline is performed on the output from the previous step; thus, each step does not model any constraints on the previous or next steps. For example, the clustering step does not model transformation blending. Likewise, after the joint locations are determined, joint constraints would change

the bone transformations generated at the previous step. As an end result, errors could be significantly accumulated (e.g., from the root joint to leaf joints). Due to these issues, these rigging methods have limited accuracy and robustness and therefore fall short of meeting the demand. Examples in Fig. 2 show two common limitations of these rigging methods.

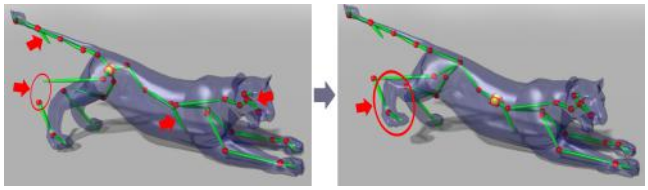

Figure 2: *Examples that show two common limitations of current rigging methods: (i) an over-estimated bone initialization may generate inaccurate and redundant bones (indicated by red arrows); (ii) an inaccurate estimation of bone transformations (indicated by red arrows with ellipse) causes noticeable deformation errors when the bones are connected by joints (see the change of the legs). The examples in this figure are generated using [Hasler et al. 2010].*

In this paper, we address the above limitations and introduce a robust and accurate rigging framework that takes a set of example poses as input and produces its corresponding *Skeleton-based Linear Blend Skinning* (LBS) model. The obtained LBS model includes skeletal structure, skinning weights, joint locations, and bone transformations corresponding to all the example poses. As shown in Fig. 1, our method can robustly generate high-quality rigging models from a small set of example poses. Compared to previous methods [Schaefer and Yuksel 2007; de Aguiar et al. 2008a; Hasler et al. 2010], our method offers the following two advantages:

- **Robustness.** By automatically pruning redundant bones, our approach is more robust than the previous methods which often suffer from an over-estimated bone initialization.
- **Accuracy.** By formulating the solving of a LBS model with skeleton as a constrained optimization, our iterative approach can obtain more accurate joint locations, bone lengths, corresponding joint rotations, as well as the reconstruction of the example poses without accumulation of fitting errors.

The key component of our method is an *Iterative Rigging* algorithm (§6), in which we alternatively update skinning weights (§6.1), joint locations, and bone transformations (§6.2); and automatically prune redundant bones (§6.3). Specifically, in this method we make the following three technical contributions:

1. **Joint constraints optimization.** We add soft constraints to the framework of *skinning decomposition without skeleton* [Le and Deng 2012], which models the common rotation between two rigid bone transformations. Using this new formulation, we convert unorganized bone transformations [Le and Deng 2012] to skeletal bones with a hierarchical structure, and propose an optimization algorithm (§6.2) to solve joint locations with a high accuracy.
2. **Skeleton pruning.** To address the robustness issue, instead of attempting to find the proper number of bones at the initialization step, we first generate an over-estimated bone estimation and then iteratively prune redundant bones (§6.3). The redundant bones are determined by utilizing the weight smoothness regularization below.
3. **Weight smoothness regularization.** To handle noisy inputs and solve the skinning fracture problem resulting from the commonly used weight sparseness constraint, we introduce a smoothness constraint into the skinning weight solver (§6.1).

2 Related Work

Skinning and rigging. The most widely used skinning model for character animation to date is *skeleton-based skinning*. A skeleton pose can be used to determine the skin deformation, with the consideration of physical properties including elasticity, contact, collision [McAdams et al. 2011; Hahn et al. 2012; Liu et al. 2013], and even the anatomy of creatures [Lee et al. 2009; Ali-Hamadi et al. 2013]. To achieve real time performance, geometric skinning methods can simply blend bone transformations to produce the skin using a set of blending weights for each vertex [Merry et al. 2006; Kavan et al. 2008; Jacobson and Sorkine 2011; Kavan and Sorkine 2012; Le and Deng 2013; Vaillant et al. 2013]. Many automatic methods have been proposed to generate the skinning weights from a static 3D model [Baran and Popović 2007; Jacobson et al. 2011]. On top of these skinning weights, artists can also make manual adjustments (weight painting). However, it is non-trivial to obtain the optimal set of weights for realistic skinning, especially for highly deformable models. One possible solution to this issue is to use a set of example poses to solve for the optimal skinning weights [Wang and Phillips 2002; Mohr and Gleicher 2003]. Recently, skinning decomposition techniques were also proposed to extract bone transformations from example poses [James and Twigg 2005; Kavan et al. 2010; Le and Deng 2012]. However, since their extracted bone transformations are not organized in any skeletal structures, they are not quite suitable for animation editing purposes.

Skeleton extraction from a single shape. Researchers have pursued two different directions to extract skeletons from a single static pose, serving different purposes. In one direction, curve skeleton extraction typically focuses on discovering the topology of the skeleton (e.g., handle or loop) rather than its exact shape [Au et al. 2008; Tagliasacchi et al. 2009; Livny et al. 2010; Huang et al. 2013]. In the other direction, some methods focus on generating skeletons for animation [Baran and Popović 2007]. Since the only input of these methods is a single pose, the quality of their output is limited.

Motion-based skeleton extraction. Using motion data, the skeleton and joint locations can be determined more easily than using a static pose alone. Anguelov et al. [2004] proposed an early work to deal with unorganized point cloud data. Due to the lack of temporal coherence, their method is primarily designed to identify and track rigid components of the model. Kirk et al. [2005] use marker-based mocap data as the input for skeleton extraction, exploiting the temporal coherence between mocap frames. Since both of the methods use low spatial resolution data, they only focus on solving the joint locations between two rigid body components, discarding the blending between bodies. Later, several methods have been proposed to work with a set of example poses [Schaefer and Yuksel 2007; de Aguiar et al. 2008a; Hasler et al. 2010]. Since the example poses offer much higher spatial resolutions than mocap data, they can often produce quality skeletons along with the LBS model. However, as mentioned in §1, besides requiring non-trivial model-specific parameter tuning, they have limited accuracy and robustness due to the significant accumulation of fitting errors.

3 Method Overview

The input data for our method is a set of example poses with temporal coherence, i.e., different poses (meshes) that share the same topology. Let F be the total number of example poses (frames), N be the number of vertices in each mesh, and $v_i^f \in \mathbb{R}^3$ be the

3D position of the i-th vertex in frame f. The input also includes the rest pose (a.k.a., the dressing pose), where the position of the i-th vertex in the rest pose is denoted as $u_i \in \mathbb{R}^3$. From the input data, our method generates skinning weights, skeletal structure, joint locations, and corresponding bone transformations using the following three main steps (Fig. 3).

Initialization (§4). From the given example poses, we first identify nearly-rigid parts and use each part to initialize one rigid bone transformation sequence. At this step, we favor production of an over-estimated number of bones. Redundant bones will be later removed at the skeleton pruning step (§6.3).

Topology reconstruction. Using the initialized bone transformations, we perform *Skinning Decomposition with rigid bones* [Le and Deng 2012] to obtain the LBS without skeleton. Then, we reconstruct the skeleton topology using minimum spanning tree algorithms on a weighted graph that is inferred from the LBS (§5).

Iterative rigging. This is the main component of our approach that employs block coordinate descent method (§6). In an iterative manner, we alternatively update all the skinning weights (§6.1), joint locations, and bone transformations (§6.2); and automatically prune redundant bones (§6.3). Our key idea of constraining two bone rotations around the joint is to add soft constraints to the bone transformations update.

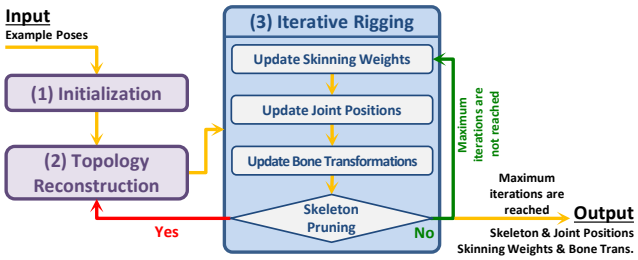

Figure 3: *The pipeline of our skeletal rigging approach*

4 Initialization

From the given deformed mesh sequence (i.e., example poses), we perform motion driven clustering to obtain B clusters, where vertices in the same cluster have similar rigid motions. For each cluster j, we fit a rigid bone transformation $[R_j^f|T_j^f]$ to relate the vertex positions at the rest pose to the vertex positions at frame f, where $R_j^f \in \mathbb{R}^{3\times3}$ is an orthogonal rotation matrix and $T_j^f \in \mathbb{R}^3$ is a translation vector.

Motion-driven vertex clustering. We apply the *Linde-Buzo-Gray* (LBG) algorithm [Linde et al. 1980] to cluster the vertices with similar rigid transformations. We first initialize one cluster to include all the vertices. Then, by extending the *cluster splitting-EM strategy* [Linde et al. 1980] to handle the mesh sequence, we divide this initial cluster into two clusters. Afterwards, the same cluster splitting-EM strategy is repeatedly called to further generate 4, 8, 16, and 32 clusters. We stop splitting at 32 clusters since all the datasets used in this paper have no more than 32 bones. However, the number of clusters can be more than 32 if needed (refer to Fig. 9), e.g., complex models are inputted.

Cluster splitting-EM. This strategy consists of two sequential steps: (i) splitting one cluster into two, and then (ii) refining the resulting clusters using Expectation-Maximization (EM). The details of the two steps are described below.

- *Cluster splitting.* Since we perform motion-driven clustering without explicitly computing the feature vector for each vertex, the feature-based cluster splitting step in the original LBG algorithm [Linde et al. 1980] cannot be directly applied. Instead, we design a new cluster splitting scheme by considering the following two criteria: (i) minimizing the approximation error, and (ii) keeping the vertices in one cluster close together. To split a cluster j, we first compute O, the rest-pose centroid of all the vertices belonging to j. Then, we find the seed vertex s belonging to j such that the product of its reconstruction error, $e(s)$, and the distance from its rest-pose position to the rest-pose centroid, $d(u_s, O)$, is maximum. Eq. (1a) shows its computing process, where $\mathcal{L}(i)$ denotes the cluster label of vertex i. Finally, for all the vertices belonging to j, we use the Euclidean distances from their rest-pose positions to u_s as the criterion to evenly split them into two sets. Conceptually, we want to use the seed s and its neighbors to initialize the new cluster centers (i.e., bone transformations). The seed s is chosen as the off-center vertex with a large reconstruction error so that the resulting new clusters can maximally reduce the overall reconstruction error, as illustrated in Fig. 4.

$$s = \arg\max_i \{d(u_i, O)e(s)\} \text{ s.t. } \mathcal{L}(i) = j \tag{1a}$$

$$\text{where: } d(u_i, O) = \|u_i - O\|_2$$

$$e(i) = \sqrt{\frac{1}{F}\sum_{f=1}^{F}\left\|[R_j^f|T_j^f]\begin{bmatrix}u_i\\1\end{bmatrix} - v_i^f\right\|_2^2} \tag{1b}$$

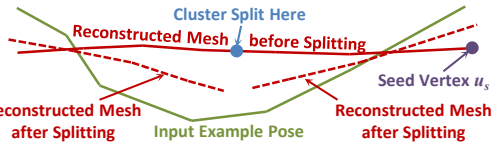

Figure 4: *An illustration of our cluster splitting strategy*

- *Cluster refining.* Similar to the work proposed by Le and Deng [2012], we represent each cluster by a rigid transformation, and EM is done by alternatively updating the cluster label $\mathcal{L}(i)$ for each vertex i at the E-step, and updating the center $[R_j^f|T_j^f]$ for each cluster j at the M-step. After the M-step, to ensure the robustness, we remove all the insignificant clusters, which are clusters with members (i.e., vertices) fewer than 0.1% of the total number of the mesh vertices.

Connected patches generation. The motion-driven clustering algorithm only assigns a vertex to the cluster using the smallest reconstruction error criterion, ignoring the mesh connectivity information. Thus, more than one rigid part might be assigned to a cluster if the parts have similar motions. To keep only one rigid part per bone, we simply find all the connected components (i.e., patches of vertices) for each cluster, and then initialize a new rigid bone transformation for each connected component.

The LBG algorithm is as fast and simple as K-means clustering [Le and Deng 2012], yet it is still sufficiently robust compared to more costly solutions such as Mean Shift clustering [James and Twigg 2005] and Agglomerative clustering [Schaefer and Yuksel 2007]. Having a few redundant clusters at the initialization step would affect our final result negligibly at most, thanks to our skeleton pruning step that removes redundant bones. Thus, an accurate estimation of the number of clusters is not required in the initialization step in our approach.

5 Topology Reconstruction

We reconstruct the skeleton topology using both bone transformations and mesh connectivity information. First, we run the skinning decomposition without a skeleton [Le and Deng 2012] for 10 iterations to refine the initialized bone transformations and generate the corresponding skinning weights. To improve the robustness, we start with only one non-zero weight per vertex and increase the number of weights-per-vertex by one every 3 iterations. The output of this step is bone transformations $[R_j^f|T_j^f] \in \mathbb{R}^{3\times 4}$ and skinning weights $w_{ij} \in \mathbb{R}$, where i, j, and f denote a vertex index, a bone index, and a frame index, respectively.

We construct a weighted graph $\mathbb{G}$ with B nodes (corresponding to B bones). The weight $g(j,k)$ between bone j and bone k is computed by Eq. (2), which puts a strong preference for having the joint (j,k) if the *joint location fitting error* (numerator) is small and the *blending of two bones* (denominator) is large. The joint fitting error is computed as the sum of squared difference of the *center of rotation* of two bones j and k after their bone transformations. Here, the center of rotation C_{jk} of two bones j and k is defined at the rest pose, and its position is computed by [Anguelov et al. 2004]. Intuitively, C_{jk} is the point such that its locations after both bone transformations in all frames are most similar; thus, the joint location fitting error (the numerator in Eq. (2)) measures the quality of having the joint between bone j and bone k. The blending of two bones (the denominator in Eq. (2)) measures the relative position of the two bones, in which a larger blending means the two bones are more likely to share a common joint; and vice versa. As the result, $g(j,k)$ is small if bone j and bone k have a high probability of sharing a common joint.

$$g(j,k) = \frac{\sum_{f=1}^{F} \left\| \left([R_j^f|T_j^f] - [R_k^f|T_k^f] \right) \begin{bmatrix} C_{jk} \\ 1 \end{bmatrix} \right\|_2^2}{\sum_{i=1}^{N} w_{ij} w_{ik}} \qquad (2)$$

Finally, we determine the skeleton $\mathbb{S}$ as the *Minimum Spanning Tree* (MST) of $\mathbb{G}$ [Kruskal 1956], which is similar to the idea of previous methods [Kirk et al. 2005; Schaefer and Yuksel 2007; de Aguiar et al. 2008a; Hasler et al. 2010]. For the sake of convenience, we write $(j,k) \in \mathbb{S}$ to denote (j,k) is an edge of the tree $\mathbb{S}$, or bone j and bone k share a common joint. This common joint is also denoted as (j,k), and the joint position (at the rest pose) is denoted as C_{jk}, which is also the center of rotation of two bones j and k.

In practice, the root of the skeleton is often manually specified. In this work, we simply set the root as the joint that is the closest to the centroid of the rest pose for visualization purpose. Given the root of the skeleton and each bone transformation, we can straightforwardly compute the joint rotations and bone lengths. However, to ensure the readability of this paper we only use the raw representation of the skeleton, that is, an unrooted skeleton with bone transformations.

For the sake of robustness, we suggest to compute the center of rotation between two bones using [Anguelov et al. 2004] rather than [Schaefer and Yuksel 2007]. We found that the pseudo-inverse solver in [Schaefer and Yuksel 2007] tends to find the wrong subspace if the bone rotations are degenerate. In contrast, the regularization term in [Anguelov et al. 2004] only depends on the centroid of two bones, which is more robust to estimate. As a trade-off, [Anguelov et al. 2004] gives a slightly larger approximation error than [Schaefer and Yuksel 2007]. Our implementation of [Anguelov et al. 2004] sets the regularization weight $\gamma = 10^{-2}F$.

6 Iterative Rigging

After initializing bone transformations and determining the skeleton topology, previous approaches directly compute the joint positions and skinning weights [Schaefer and Yuksel 2007; de Aguiar et al. 2008a; Hasler et al. 2010]. However, since the initialization of the bone transformations lacks both robustness and the consideration of joint rotation constraints, the output skinning model can be degraded or even inaccurate, as shown in the examples in Fig. 2. We solve this problem by imposing soft joint constraints to the optimization function and pruning redundant bones.

Problem formulation. We minimize the objective function E in Eq. (3a) to find the optimized LBS model with skeletal constraints. It includes the following three terms:

- *Data fitting term* E_D (Eq. (3b)) minimizes the mesh reconstruction error, similar to the work of [Le and Deng 2012].
- *Weight regularization term* E_S (Eq. (3c)) favors the smoothness of skinning weights (§6.1) and drives the removal of redundant bones (§6.3). We derive this term from the fairness and smoothness conditions on the manifold, which is similar to [Kim et al. 2010]. $w_j \in \mathbb{R}^N$ is the column vector of N skinning weights of bone j, and $L \in \mathbb{R}^{N\times N}$ is a discrete Laplacian matrix of the input mesh, where L_{ik} represents the similarity between the weights of two neighboring vertices i and k (refer to Eq. (4a) for detailed definition of L).
- *Joint constraint term* E_J (Eq. (3d)) keeps any two connected bone transformations $\in \mathbb{S}$ rotate around their common joint. This soft constraint is derived from [Anguelov et al. 2004; Schaefer and Yuksel 2007]. If two bones j and k share the common joint (j,k), this constraint favors the rest-pose position of the joint, C_{jk}, going to the same position after bone j transformation and bone k transformation (also refer to the explanation of Eq. (2) in §5).

The minimization of E is also subject to the same set of constraints as in [Le and Deng 2012]. It includes *convex constraints* (non-negativity and affinity) and *sparseness constraints* (no more than 4 non-zero weights per vertex) on the skinning weights w_{ij} (Eq. (3e)), and includes *orthogonal constraints* (Eq. (3f)) on the bone transformations R_j^f (all the bone transformations need to be rigid).

$$E = E_D + \omega E_S + \lambda E_J \qquad (3a)$$

$$\text{Where: } E_D = \frac{1}{NF} \sum_{i=1}^{N} \sum_{f=1}^{F} \left\| \sum_{j=1}^{B} w_{ij} [R_j^f|T_j^f] \begin{bmatrix} u_i \\ 1 \end{bmatrix} - v_i^f \right\|_2^2 \qquad (3b)$$

$$E_S = \sum_{j=1}^{B} {w_j}^{\mathsf{T}} L w_j \qquad (3c)$$

$$E_J = \frac{1}{F} \sum_{(j,k)\in\mathbb{S}} \sum_{f=1}^{F} \left\| \left([R_j^f|T_j^f] - [R_k^f|T_k^f] \right) \begin{bmatrix} C_{jk} \\ 1 \end{bmatrix} \right\|_2^2 \qquad (3d)$$

$$\text{Subject to: } w_{ij} \geq 0,\ \sum_{j=1}^{B} w_{ij} = 1,\ \|w_i\|_0 \leq 4, \forall i, j \qquad (3e)$$

$${R_j^f}^{\mathsf{T}} R_j^f = I,\ \det R_j^f = 1, \forall j, f \qquad (3f)$$

Optimization. We optimize the objective function Eq. (3a) using Block Coordinate Descent algorithm [Le and Deng 2012]. This

algorithm alternatively updates skinning weights, joint positions, and bone transformations while keeping the remaining blocks fixed (Fig. 3). Specifically, the weights update minimizes E with respect to the data fitting and weight regularization terms (§6.1); the joint positions update minimizes the joint constraint term using [Anguelov et al. 2004]; the bone transformations update minimizes the data fitting and joint constraint terms (§6.2). We repeat this alternative update process for a user-specified maximum number of times, which is empirically set to 20 in our experiments. During this process, we always prune redundant bones (§6.3) after each bone transformations update step. If a redundant bone is pruned, we will restart the iterative update process by resetting the iteration counter to zero. In Eq. (3a), ω is a constant determined by the Laplacian matrix (§6.1); λ starts with 1 and is multiplied by 1.5 after each iteration.

6.1 Skinning Weights Update

Conventional least squares (LS) skinning weights solvers [James and Twigg 2005; Schaefer and Yuksel 2007] are sensitive to noisy data. When the sparseness constraint is imposed, the LS solver might generate skinning fractures, i.e., visible discontinuities on the surface (examples are shown as the $\omega = 0$ case in Fig. 5), as some neighboring vertices are associated with different bones. Thus, we add a weight regularization term to make our algorithm robust.

Rigidness Laplacian regularization. Our Laplacian L is computed by Eq. (4a), where the rigidness weight d_{ik} penalizes the maximum change of the Euclidean distance between vertex i and vertex k with respect to the rest pose (Eq. (4b)). Intuitively, if vertex i and vertex k belong to the same nearly-rigid part, the distance between them should not change much, resulting in a large rigidness weight d_{ik} and highly similar skinning weights for vertex i and vertex k (i.e., w_i and w_k).

$$L_{ik} = \begin{cases} 1 & \text{if } k = i \\ -\frac{d_{ik}}{\sum_{h \in \mathcal{N}(i)} d_{ih}} & \text{if } k \in \mathcal{N}(i) \\ 0 & \text{otherwise.} \end{cases} \tag{4a}$$

Where: $\mathcal{N}(i)$ denotes all the 1-ring neighbors of vertex i

$$d_{ik} = \frac{1}{\sqrt{\frac{1}{F}\sum_{f=1}^{F}\left(\left\|v_i^f - v_k^f\right\|_2 - \left\|u_i - u_k\right\|_2\right)^2}} \tag{4b}$$

With the proposed matrix L, we empirically set $\omega = 10^{-3}$ for all the experiments in this paper. This parameter is determined by first resizing all the datasets to tightly fit their rest poses in a unit sphere and then observing the deformation smoothness for different ω values, examples of which are shown in Fig. 5.

Iterative local optimization. Without the smoothness term E_s, skinning weights can be updated per-vertex [James and Twigg 2005; Le and Deng 2012] since the weights of each vertex are independent of those of other vertices. In our case, the weights of all the vertices are related to each other in the smoothness term E_s (refer to Eq. (3c)). Unfortunately, a global weights update of all the vertices is impractical with a very large number of unknowns ($N \times 4$). Instead, we use an iterative local optimization strategy similar to the one used in [Landreneau and Schaefer 2010]. Specifically, we iterate through all the vertices one by one (with the given order in the input). For each vertex $\hat{i}$, we fix the weights of its one-ring neighbors and minimize the objective function, Eq. (3a), with respect to weights $w_{\hat{i}} \in \mathbb{R}^B$. Then, this problem becomes a linear least squares problem with convex and sparseness constraints as follows:

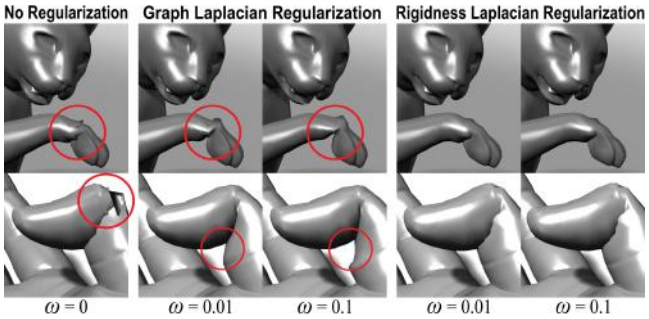

Figure 5: *The effect of different weight smoothness regularizations. Compared to the graph Laplacian [Merry et al. 2006; Kim et al. 2010], our rigidness Laplacian regularization offers better smoothness while still preserving the global shape. Note that how the graph Laplacian over-smooths the fracture while it starts to change the global shape (indicated by red circles). Our rigidness Laplacian is robust as demonstrated by the similar results with different ω values.*

$$\min_{w_{\hat{i}}} \frac{1}{NF}\sum_{f=1}^{F}\left\|\sum_{j=1}^{B} w_{\hat{i}j}[R_j^f|T_j^f]\begin{bmatrix}u_{\hat{i}}\\1\end{bmatrix} - v_{\hat{i}}^f\right\|_2^2 + \omega\sum_{i=1}^{N} L_{\hat{i}i}\left\|w_{\hat{i}} - w_i\right\|_2^2 \tag{5}$$

The above least squares problem can be solved in a similar way as [James and Twigg 2005]. Specifically, we handle the sparseness constraint by greedily selecting 4 weights with smallest residuals when used to individually estimate the objective function, Eq. (5). Then, we perform non-negative linear least squares with the affinity constraint on the set of 4 selected weights to compute their values. Note that its size is relatively small since L is a sparse matrix. As suggested by Landreneau and Schaefer [2010], we iterate the local solver a few times proportional to the number of mesh vertices. Since these iterations are mixed with the iterations of the global block coordinate descent, a small number of iterations are typically sufficient to reach the convergence in our experiments, e.g., 0.05 percent of the total number of the mesh vertices.

Note that similar solutions of using *graph Laplacian* for skinning weights regularization have been proposed [Merry et al. 2006; Kim et al. 2010]. However, in their methods, the Laplacian matrix is computed only based on the graph of a single mesh; it is insensitive to the deformation over the whole mesh sequence. In contrast, our rigidness Laplacian regularization takes the deformation of the whole mesh sequence into consideration (Fig. 5).

6.2 Bone Transformations Update

When updating the bone transformations, our main focus is to enforce the joint constraints while still achieving a low approximation error. The main idea of this step is to employ the solution to the Absolute Orientation problem with blending to relate two sets of points [Le and Deng 2012], while the joints are treated as additional points with a large weight, λ. When λ is increased, bones are constrained to rotate more strictly around the joints (Fig. 6).

At this bone transformations update step, we need to minimize the objective function (Eq. (3a)) with respect to the bone transformations $[R_j^f|T_j^f]$. To constrain the rotation matrix R_j^f to be orthog-

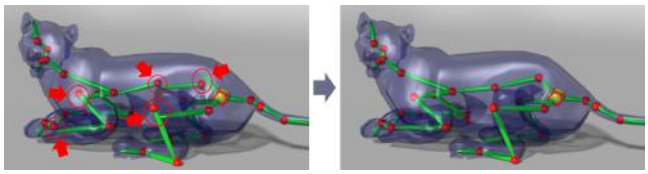

Figure 6: *(Left) Without joint constraints, the bone transformations generated by [Le and Deng 2012] do not always rotate around the joints. (Right) With the soft joint constraints, bones rotate more strictly around the joints. Since bone transformations are alternatively updated with joint locations, our model can converge to a local optimum with a good approximation of the input.*

onal, we employ the optimization strategy in [Le and Deng 2012], where bones are updated one by one (with the fixed order of the list of bones) while keeping the remaining bones fixed.

The transformation of bone $\hat{j}$ at frame f is updated by minimizing the objective function in Eq. (6a), which contains two parts. The first part corresponds to the data fitting term (Eq. (3b)) whose optimal solution brings the rest pose u_i to the residual q_i^f (Eq. (6b)). The second part corresponds to the joint constraints (Eq. (3d)) that enforce $[R_{\hat{j}}^f|T_{\hat{j}}^f]$ to bring every joint $C_{\hat{j}k}$ of bone $\hat{j}$ to its expected position $\Psi_k^f(C_{\hat{j}k})$ after bone k transformation. Note that the weight smoothness term E_S in Eq. (3c) can be dropped since $[R_j^f|T_j^f]$ is not involved in E_S.

$$\min E_{\hat{j}}^f = \frac{1}{N}\sum_{i=1}^{N}\left\| w_{i\hat{j}}[R_{\hat{j}}^f|T_{\hat{j}}^f]\begin{bmatrix}u_i\\1\end{bmatrix} - q_i^f\right\|_2^2 + \lambda\sum_{(\hat{j},k)\in\mathbb{S}}\left\|[R_{\hat{j}}^f|T_{\hat{j}}^f]\begin{bmatrix}C_{\hat{j}k}\\1\end{bmatrix} - \Psi_k^f(C_{\hat{j}k})\right\|_2^2 \quad (6a)$$

$$\text{Where: } q_i^f = v_i^f - \sum_{j=1,j\neq\hat{j}}^{B} w_{ij}[R_j^f|T_j^f]\begin{bmatrix}u_i\\1\end{bmatrix} \quad (6b)$$

$$\Psi_k^f(C_{\hat{j}k}) = [R_k^f|T_k^f]\begin{bmatrix}C_{\hat{j}k}\\1\end{bmatrix} \quad (6c)$$

The optimal solution to minimize the objective function Eq. (6a) is to combine the solution proposed by Le and Deng [2012] with the solution to the Weighted Absolute Orientation problem [Kabsch 1978]. We first need to compute the center of rotation p_* for the rest pose (Eq. (7a)) and the center of rotation q_*^f for frame f (Eq. (7b)). Note that, in Eqs. (7a) and (7b), $\left|(\hat{j},k)\in\mathbb{S}\right|$ denotes the number of the edges that are connected with $\hat{j}$ in $\mathbb{S}$.

$$p_* = \frac{\frac{1}{N}\sum_{i=1}^{N} w_{i\hat{j}}^2 u_i + \lambda\sum_{(\hat{j},k)\in\mathbb{S}} C_{\hat{j}k}}{\frac{1}{N}\sum_{i=1}^{N} w_{i\hat{j}}^2 + \lambda\left|(\hat{j},k)\in\mathbb{S}\right|} \quad (7a)$$

$$q_*^f = \frac{\frac{1}{N}\sum_{i=1}^{N} w_{i\hat{j}} q_i^f + \lambda\sum_{(\hat{j},k)\in\mathbb{S}} \Psi_k^f(C_{\hat{j}k})}{\frac{1}{N}\sum_{i=1}^{N} w_{i\hat{j}}^2 + \lambda\left|(\hat{j},k)\in\mathbb{S}\right|} \quad (7b)$$

Then, we subtract the center of rotation from each vertex and each joint as follows:

$$\overline{p}_i = u_i - p_*; \qquad \overline{C}_{\hat{j}k} = C_{\hat{j}k} - p_* \quad (8a)$$

$$\overline{q}_i^f = q_i^f - w_{i\hat{j}} q_*^f; \qquad \overline{\Psi}_k^f(C_{\hat{j}k}) = \Psi_k^f(C_{\hat{j}k}) - q_*^f \quad (8b)$$

The points after subtraction are concatenated into matrices $P, Q \in \mathbb{R}^{3\times(N+|(\hat{j},k)\in\mathbb{S}|)}$ as follows:

$$P = \left[\frac{w_{1\hat{j}}}{N}\overline{p}_1 \cdots \frac{w_{N\hat{j}}}{N}\overline{p}_N \;\middle|\; \forall_{(\hat{j},k)\in\mathbb{S}}\lambda\overline{C}_{\hat{j}k}\right] \quad (9a)$$

$$Q = \left[\frac{1}{N}\overline{q}_1^f \cdots \frac{1}{N}\overline{q}_N^f \;\middle|\; \forall_{(\hat{j},k)\in\mathbb{S}}\lambda\overline{\Psi}_k^f(C_{\hat{j}k})\right] \quad (9b)$$

Finally, the optimal transformation of bone $\hat{j}$ at frame f is computed by performing Singular Value Decomposition on PQ^T as follows (Eq. (10a)):

$$R_{\hat{j}}^f = \vartheta\mu^\mathsf{T}; \;\; T_{\hat{j}}^f = q_*^f - R_{\hat{j}}^f p_* \quad (10a)$$

$$\text{Where: } \mu\varsigma\vartheta^\mathsf{T} = PQ^\mathsf{T} \quad (10b)$$

6.3 Skeleton Pruning

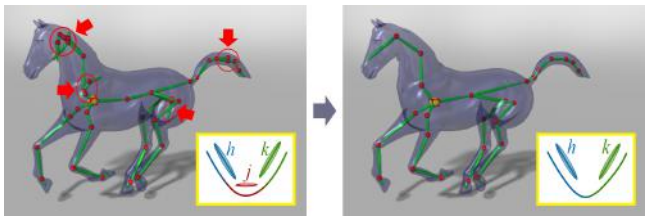

Figure 7: *Redundant bones in the left panel are pruned to achieve the neat skeleton in the right panel. As illustrated in the two yellow boxes, a redundant bone j is identified by utilizing the weight regularization term to force its weights degenerate.*

Since the initialization step (§4) considers neither transformation blending nor skeletal structure, it might generate some redundant bones. Typically, the redundant bones are located in highly deformable regions, e.g., around the joints, as they cause large approximation errors at the motion driven clustering step. Later, after the LBS resolves the highly deformable regions, the redundant bones are no longer needed and thus should be removed. Unfortunately, since the redundant bones do not violate any conditions in the LBS formulation, accurately identifying them is difficult. For this reason, the solution optimized by a general data fitting with alternative skinning weights update and bone transformations update [Le and Deng 2012] would still retain the redundant bones in order to minimize the LBS deformation error.

Instead of conducting a brute-force search for the redundant bones, we utilize the weight regularization term to force their weights degenerate. We illustrate an example of this strategy in two yellow boxes in Fig. 7. As illustrated in the yellow box in the left panel, the bone j (red) is initialized at a potential joint between the bone h (blue) and the bone k (green). Although the blending between h and k can closely approximate the deformation, having j is still a valid solution for the best approximation. By adding the weight regularization term, our minimized objective function (Eq. (3a)) makes

the weights of j very close to zero, in order to improve the smoothness of the skinning weights. As the result, the bone j becomes degenerate and can be removed. In our experiments, we found that the weights of the redundant bones converge to zero quite slowly. For this reason, we remove the redundant bone j if the sum of its squared weights is smaller than 1% of the largest sum of the squared weights among all the bones. Mathematically, we remove the bone j if its weights satisfy the condition described in the following Eq. (11).

$$\text{Remove bone } j \text{ if } \sum_{i=1}^{N} {w_{ij}}^2 < 10^{-2} \max_k \left\{ \sum_{i=1}^{N} {w_{ik}}^2 \right\} \quad (11)$$

7 Results and Comparisons

Datasets. We evaluated our approach on 9 test datasets (Table 1) obtained from various publicly available sources: the **cat-poses**, **horse-poses**, **lion-poses**, and **horse-gallop** were obtained from [Sumner and Popović 2004]; the **hand** was obtained from [Utah 2013]; the **dance** and **cow** were obtained from [Briceño et al. 2003]; the **scape** was obtained from [Anguelov et al. 2005]; and the **samba** was obtained from [Vlasic et al. 2008]. To test the robustness of our approach, we only used a *single* set of parameters for all the experiments on all the test datasets in this paper (see previous sections for parameter selection discussion). Please refer to the enclosed supplemental video for the animation results by our approach.

Fig. 8 shows the skeletons extracted from 3 test datasets by our approach as well as their corresponding cluster initialization results. Despite generating over-estimated clusters at the initialization step, our approach successfully pruned redundant bones, especially for the **dance** and **hand** models. Fig. 9 demonstrates the robustness and effectiveness of our skeleton pruning for handling different numbers of clusters from the initialization step. Our approach can produce similar final skeletons with consistent structure on the trunk and legs, by pruning most of the redundant bones. Note that minor differences only appear on certain highly deformable regions such as the tail and feet.

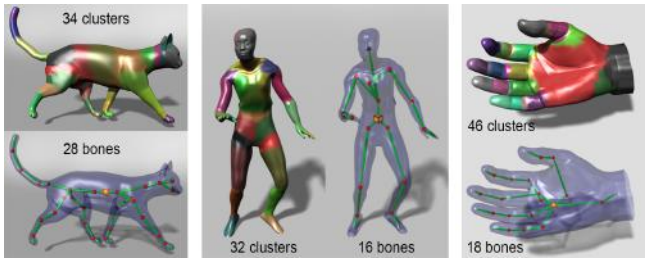

Figure 8: *The skeletons extracted from 3 datasets by our approach (from left to right):* ***cat-poses***, ***dance***, *and* ***hand****. The clustering results obtained at the initialization step are also illustrated via a color-coded scheme. Using the same set of parameters, our approach can robustly determine the optimal number of bones in the skeletons using skeleton pruning.*

In Fig. 10, we compare our approach with three state of the art approaches [Schaefer and Yuksel 2007; de Aguiar et al. 2008a; Hasler et al. 2010]. For a fair comparison, we use the number of bones generated by our approach as one of the input parameters to the three state of the art methods. In addition, other parameters in the three state of the art approaches are set as suggested by their authors. While our approach can generate sound results for all the

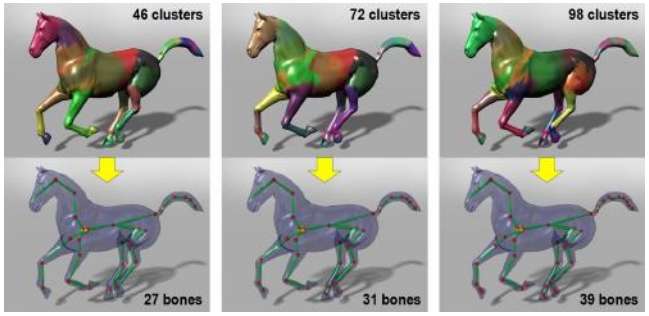

Figure 9: *The extracted* ***horse-gallop*** *skeletons with different cluster initializations. Despite very different numbers of initialized clusters, our approach can output similar final skeletons with consistent structure on the trunk and legs; minor differences on the tail and feet are due to the high deformations on these regions.*

test datasets, in general the other three approaches suffer from the following two issues:

1. *Redundant bones.* The **lion-poses**, **horse-gallop**, and **scape** generated by both [Schaefer and Yuksel 2007] and [Hasler et al. 2010] have many redundant bones, since the clustering algorithms in their approaches result in an over-estimated number of bones in some highly-deformable regions.

2. *Inaccurate joint locations.* The joints at the back legs of the **lion-poses** are inaccurately estimated by both [de Aguiar et al. 2008a] and [Hasler et al. 2010], and the joints at the legs of the **scape** are inaccurately estimated by [Schaefer and Yuksel 2007]. We suspect this issue is due to the inaccurate estimation of bone transformations in some highly deformable parts.

In Table 1, we show quantitative comparisons among all the four methods (our method, *method II* - [Schaefer and Yuksel 2007], *method III* - [de Aguiar et al. 2008a], and *method IV* - [Hasler et al. 2010]). In term of the approximation power (measured by lower RMSE), our method soundly outperforms all the other three methods thanks to its iterative rigging. However, our method is slower than both the method II and the method III because our method needs many iterations. Fig. 11 shows examples of visual distortion on the reconstructed **hand** poses with respect to different RMSE values.

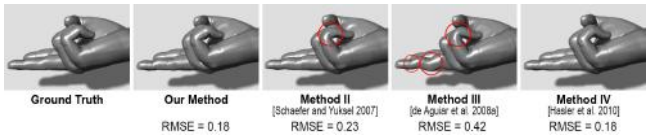

Figure 11: *Examples of visual distortion on the reconstructed* ***hand*** *poses with respect to different RMSE values (pay attention to the red circled areas). Fig. 10 shows the corresponding skeletons.*

8 Discussion

Performance. Our method can only prune a small number of bones at a time, and thus its iterative rigging may need to be repeated many times if the number of initialized bones is significantly over-estimated. We found a positive correlation between the running time of our method and the number of bones pruned. Therefore, if the number of initialized bones is much larger than the number of bones in the final skeleton, our method takes a significant amount of time to prune redundant bones. Fig. 12 visualizes an ex-

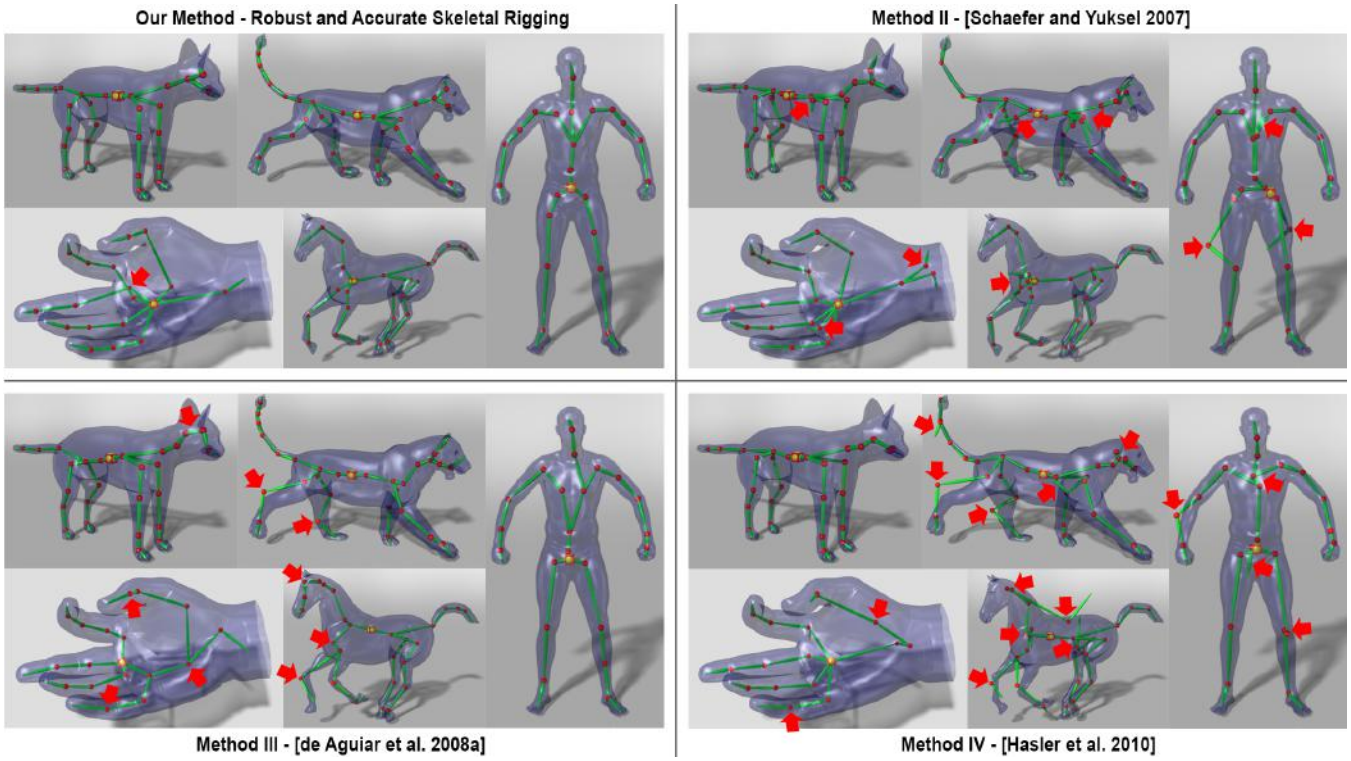

Figure 10: *Comparisons between our method and three state of the art approaches. The five test datasets shown in this figure are (in the clockwise direction starting from the top-left corner):* ***cat-poses***, ***lion-poses***, ***scape***, ***horse-gallop***, *and* ***hand****. For a fair comparison, we set the same number of bones for all the four methods. Only our method can generate sound outputs for all the 5 datasets. The issues in the results are indicated by red arrows.*

Dataset	N	F	B	Our method		Method II		Method III		Method IV	
				Time	RMSE	Time	RMSE	Time	RMSE	Time	RMSE
cat-poses	7207	9	28	5.8	**0.25**	0.1	0.68	6.9	1.04	17.2	0.63
horse-poses	8431	10	27	7.7	**0.21**	0.2	0.54	6.2	1.24	20.0	0.75
lion-poses	5000	9	30	4.1	**0.27**	0.1	0.83	4.0	1.62	11.7	1.14
horse-gallop	8431	48	27	41.9	**0.22**	0.8	0.44	33.3	1.10	80.3	0.88
hand	7997	43	18	65.1	**0.18**	0.6	0.23	20.0	0.42	41.9	**0.18**
dance	7061	201	16	148.7	**0.22**	2.5	0.76	61.8	0.78	168.0	0.53
scape	12500	70	23	252.1	**0.42**	1.7	1.03	60.7	1.18	410.4	1.24
samba	9971	175	22	348.2	**0.56**	3.3	1.29	95.1	1.57	296.0	1.79
cow	2904	204	11	72.3	**1.52**	1.0	5.41	16.0	5.61	47.9	5.58

Table 1: *Quantitative comparisons among all the four methods. The reported RMSE is normalized by the bounding volume diagonal [Schaefer and Yuksel 2007; Hasler et al. 2010]. Specifically, $RMSE = 100 \times \sqrt{E_D}/d$, where E_D is the data fitting error in Eq.* (3b)*, and d is the diagonal of the bounding box of the rest pose. The error is computed on the output using the joint rotation representation to strictly enforce the joint constraints. The running time (in minutes) was recorded on the same off-the-shelf computer with an Intel Xeon E5405 2.0GHz CPU. All the methods in this comparison were implemented in C++ with single thread.*

ample of its detailed computation breakdown, which shows that a more proper initialization of bones would significantly shorten the computational time of our method. Due to potentially long running time, our current method is only suitable for offline applications. A potential solution to speed up the performance is to incorporate skeleton templates or certain user interactions into our approach, which is a part of our future work.

Figure 12: *Detailed computation breakdown of our method on the* ***samba*** *model. The number inside a Skeleton-Reconstruction block denotes the number of bones at the current step. The number inside a Rigging-Iterations block denotes the number of iterations executed at the iterative rigging step. Each rigging iteration includes skinning weights update, joint positions update, bone transformations update, and skeleton pruning. The computational times of all the rigging iterations are approximately the same; one detailed breakdown is illustrated in the Last-Rigging-Iteration block.*

Data dependency. Due to its data-driven (i.e., example-based) nature, rigging results by our approach largely depend on the quality of input example poses. We found that the limited motion range and noise of input example poses could affect the outcome of our approach at some cases. For example, the asymmetric skeletons of the **dance** model (in Fig. 1 and Fig. 8) and the **samba** model (in Fig. 1 and Fig. 13) are due to noise, asymmetry, and the limited motion ranges of some joints in the example poses. The incorrect joint of the **hand** model (in Fig. 10) is due to the very similar motions of the pinkie and index fingers in the example poses.

Approximation power. Results in §7 demonstrate that our approach is robust and accurate in its handling of articulated, nearly-rigid models. We also tested our approach with several highly deformable models although this is not one of the targeted applications for any skeleton extraction or rigging methods. The re-

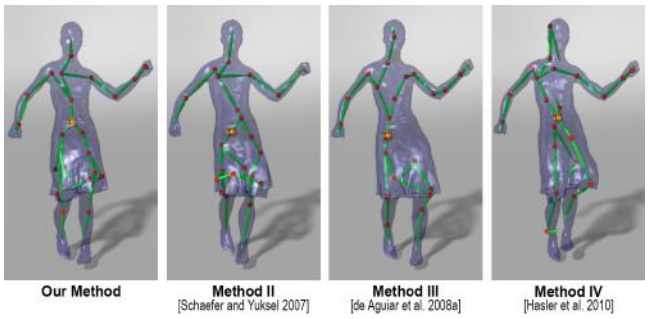

Figure 13: *Despite the high deformation on the skirt part of the **samba** model, our approach is still able to generate a reasonable skeletal structure where the skirt is rigged by some bones originated from the hip. Meanwhile, the three previous approaches cannot extract similar skeleton patterns.*

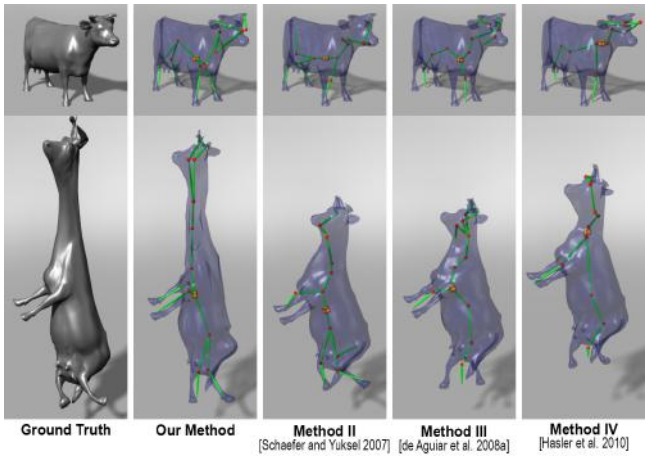

Figure 14: *Our method can even rig an elastic model such as this stretched **cow**. The top row shows the resulting skeletons in the rest pose. Compared with the reconstructed result by our method, the reconstructed poses by the three previous methods are not visually close to the ground-truth.*

sults in Fig. 13 and Fig. 14 show that, our method can generate skeleton-based LBS models with better overall approximations than the three previous methods [Schaefer and Yuksel 2007; de Aguiar et al. 2008a; Hasler et al. 2010], since our iterative rigging algorithm can effectively optimize the objective function in Eq. (3a). Notice that our method can even approximate the **cow** deformation reasonably well (Fig. 14), although a more suitable skinning model for this dataset is still the LBS model without skeletal structure [Le and Deng 2012]. Moreover, due to the limited approximation power of the LBS model, our current method cannot capture certain complicated non-linear deformations (one example is shown in Fig. 15).

9 Conclusions

We present a robust and accurate approach to automatically generate a skeleton-based LBS rigging model. Given a set of example poses, our approach can generate its skeletal structure, joint positions, skinning weights, and bone transformations corresponding to the example poses. The output of our method (i.e., skeleton-based LBS) can be straightforwardly incorporated into current animation pipeline in game engines and popular 3D modeling/animation software such as Autodesk Maya. As demonstrated in our experimental

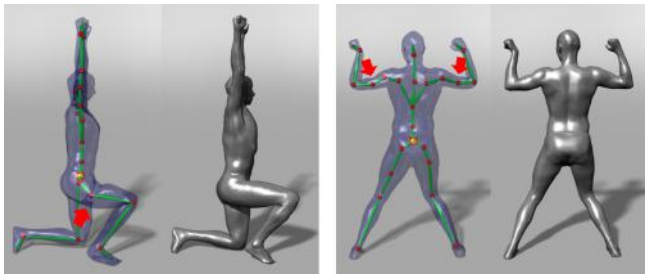

Figure 15: *Our approach fails to remove 4 redundant bones in the upper arms and upper legs of the **scape** model, since the LBS model need them for a better approximation of muscle bulging on these parts (indicated by red arrows). To the end, our approach keeps the 4 bones to capture the non-linear deformation effect. The gray models are the ground-truth.*

results, even only using a single set of parameters, our approach can generate high-quality skeletons and soundly outperform three state-of-the-art methods for all the test datasets in terms of robustness and accuracy.

Despite the achieved accuracy and robustness, our current approach has several limitations including the aforementioned low computational efficiency, example data dependency, and limited approximation power of the LBS model. We believe some potential solutions to tackle these issues would be to utilize skeleton templates, introduce certain user interventions, or design more sophisticated skinning models.

Acknowledgements

This work is supported in part by NSF IIS-0914965, NIH 1R21HD075048-01A1, and NSFC Grant 61328204. We would like to thank Daniel Vlasic, Jovan Popovic, James Davis, and Hugues Hoppe for providing mesh sequences used in this work, and thank JP Lewis and Omprakash Gnawali for their proofreading helps. Finally, we would like to thank the anonymous SIGGRAPH reviewers for their constructive comments.

References

ALI-HAMADI, D., LIU, T., GILLES, B., KAVAN, L., FAURE, F., PALOMBI, O., AND CANI, M.-P. 2013. Anatomy transfer. *ACM Trans. Graph. 32*, 6 (Nov.), 188:1–188:8.

ANGUELOV, D., KOLLER, D., PANG, H.-C., SRINIVASAN, P., AND THRUN, S. 2004. Recovering articulated object models from 3D range data. In *UAI'04: Proc. of Conf. on Uncertainty in Artificial Intelligence*, 18–26.

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

AU, O. K.-C., TAI, C.-L., CHU, H.-K., COHEN-OR, D., AND LEE, T.-Y. 2008. Skeleton extraction by mesh contraction. *ACM Trans. Graph. 27*, 3 (Aug.), 44:1–44:10.

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

BRICEÑO, H. M., SANDER, P. V., MCMILLAN, L., GORTLER, S., AND HOPPE, H. 2003. Geometry videos: A new representa-

tion for 3D animations. In *SCA'03: Proc. of Symp. on Computer Animation*, 136–146.

DE AGUIAR, E., THEOBALT, C., THRUN, S., AND SEIDEL, H.-P. 2008. Automatic conversion of mesh animations into skeleton-based animations. *Comput. Graph. Forum 27*, 2 (4), 389–397.

DE AGUIAR, E., STOLL, C., THEOBALT, C., AHMED, N., SEIDEL, H.-P., AND THRUN, S. 2008. Performance capture from sparse multi-view video. *ACM Trans. Graph. 27*, 3 (Aug.), 98:1–98:10.

HAHN, F., MARTIN, S., THOMASZEWSKI, B., SUMNER, R., COROS, S., AND GROSS, M. 2012. Rig-space physics. *ACM Trans. Graph. 31*, 4 (July), 72:1–72:8.

HASLER, N., THORMÄHLEN, T., ROSENHAHN, B., AND SEIDEL, H.-P. 2010. Learning skeletons for shape and pose. In *I3D'10: Proc. of Symp. on Interactive 3D Graphics and Games*, 23–30.

HUANG, H., WU, S., COHEN-OR, D., GONG, M., ZHANG, H., LI, G., AND CHEN, B. 2013. L1-medial skeleton of point cloud. *ACM Trans. Graph. 32*, 4 (July), 65:1–65:8.

JACOBSON, A., AND SORKINE, O. 2011. Stretchable and twistable bones for skeletal shape deformation. *ACM Trans. Graph. 30*, 6 (Dec.), 165:1–165:8.

JACOBSON, A., BARAN, I., POPOVIĆ, J., AND SORKINE, O. 2011. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph. 30*, 4 (July), 78:1–78:8.

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

KABSCH, W. 1978. A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A 34*, 827–828.

KAVAN, L., AND SORKINE, O. 2012. Elasticity-inspired deformers for character articulation. *ACM Trans. Graph. 31*, 6 (Nov.), 196:1–196:8.

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

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

KIM, B.-U., FENG, W.-W., AND YU, Y. 2010. Real-time data driven deformation with affine bones. *Vis. Comput. 26*, 6-8 (June), 487–495.

KIRK, A. G., O'BRIEN, J. F., AND FORSYTH, D. A. 2005. Skeletal parameter estimation from optical motion capture data. In *IEEE CVPR*, 782–788.

KRUSKAL, J. B. 1956. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. *Proceedings of the American Mathematical Society 7*, 1 (Feb.), 48–50.

LANDRENEAU, E., AND SCHAEFER, S. 2010. Poisson-based weight reduction of animated meshes. *Comput. Graph. Forum 29*, 6, 1945–1954.

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. 2013. Two-layer sparse compression of dense-weight blend skinning. *ACM Trans. Graph. 32*, 4 (July), 124:1–124:10.

LEE, S.-H., SIFAKIS, E., AND TERZOPOULOS, D. 2009. Comprehensive biomechanical modeling and simulation of the upper body. *ACM Trans. Graph. 28*, 4 (Sept.), 99:1–99:17.

LINDE, Y., BUZO, A., AND GRAY, R. 1980. An algorithm for vector quantizer design. *IEEE Trans. on Communication 28*, 1, 84–95.

LIU, L., YIN, K., WANG, B., AND GUO, B. 2013. Simulation and control of skeleton-driven soft body characters. *ACM Trans. Graph. 32*, 6 (Nov.), 215:1–215:8.

LIVNY, Y., YAN, F., OLSON, M., CHEN, B., ZHANG, H., AND EL-SANA, J. 2010. Automatic reconstruction of tree skeletal structures from point clouds. *ACM Trans. Graph. 29*, 6 (Dec.), 151:1–151:8.

MCADAMS, A., ZHU, Y., SELLE, A., EMPEY, M., TAMSTORF, R., TERAN, J., AND SIFAKIS, E. 2011. Efficient elasticity for character skinning with contact and collisions. *ACM Trans. Graph. 30*, 4 (July), 37:1–37:12.

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.

MOHR, A., AND GLEICHER, M. 2003. Building efficient, accurate character skins from examples. *ACM Trans. Graph. 22*, 3 (July), 562–568.

PARK, S. I., AND HODGINS, J. K. 2006. Capturing and animating skin deformation in human motion. *ACM Trans. Graph. 25*, 3 (July), 881–889.

SCHAEFER, S., AND YUKSEL, C. 2007. Example-based skeleton extraction. In *SGP'07: Proc. of Symp. on Geometry Processing*, 153–162.

STOLL, C., GALL, J., DE AGUIAR, E., THRUN, S., AND THEOBALT, C. 2010. Video-based reconstruction of animatable human characters. *ACM Trans. Graph. 29*, 6 (Dec.), 139:1–139:10.

SUMNER, R. W., AND POPOVIĆ, J. 2004. Deformation transfer for triangle meshes. *ACM Trans. Graph. 23*, 3 (Aug.), 399–405.

TAGLIASACCHI, A., ZHANG, H., AND COHEN-OR, D. 2009. Curve skeleton extraction from incomplete point cloud. *ACM Trans. Graph. 28*, 3 (July), 71:1–71:9.

UTAH, 2013. The utah 3D animation repository.

VAILLANT, R., BARTHE, L., GUENNEBAUD, G., CANI, M.-P., ROHMER, D., WYVILL, B., GOURMEL, O., AND PAULIN, M. 2013. Implicit skinning: Real-time skin deformation with contact modeling. *ACM Trans. Graph. 32*, 4 (July), 125:1–125:12.

VLASIC, D., BARAN, I., MATUSIK, W., AND POPOVIĆ, J. 2008. Articulated mesh animation from multi-view silhouettes. *ACM Trans. Graph. 27*, 3 (Aug.), 97:1–97:9.

VLASIC, D., PEERS, P., BARAN, I., DEBEVEC, P., POPOVIĆ, J., RUSINKIEWICZ, S., AND MATUSIK, W. 2009. Dynamic shape capture using multi-view photometric stereo. *ACM Trans. Graph. 28*, 5 (Dec.), 174:1–174:11.

WANG, X. C., AND PHILLIPS, C. 2002. Multi-weight enveloping: least-squares approximation techniques for skin animation. In *SCA'02: Proc. of Symp. on Computer animation*, 129–138.