

Smooth Skinning Decomposition with Rigid Bones

Binh Huy Le* Zhigang Deng†
University of Houston

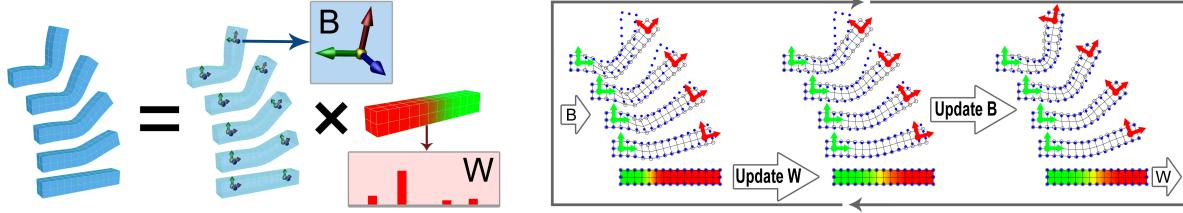


Figure 1: A set of example poses are decomposed into rigid bone transformations B and a sparse, convex bone-vertex weight map W (left hand side) by our block coordinate descent algorithm (right hand side). During the process, the example poses (indicated as blue dots) can be reconstructed more accurately by alternatively updating W and B while the other is kept fixed.

Abstract

This paper introduces the Smooth Skinning Decomposition with Rigid Bones (SSDR), an automated algorithm to extract the linear blend skinning (LBS) from a set of example poses. The SSDR model can effectively approximate the skin deformation of nearly articulated models as well as highly deformable models by a low number of rigid bones and a sparse, convex bone-vertex weight map. Formulated as a constrained optimization problem where the least squared error of the reconstructed vertices by LBS is minimized, the SSDR model can be solved by a block coordinate descent-based algorithm to iteratively update the weight map and the bone transformations. By employing the sparseness and convex constraints on the weight map, the SSDR model can be used for traditional skinning decomposition tasks such as animation compression and hardware-accelerated rendering. Moreover, by imposing the orthogonal constraints on the bone rotation matrices (rigid bones), the SSDR model can also be applied in motion editing, skeleton extraction, and collision detection tasks. Through qualitative and quantitative evaluations, we show the SSDR model can measurably outperform the state-of-the-art skinning decomposition schemes in terms of accuracy and applicability.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

Keywords: Skinning Decomposition, Linear Blend Skinning, Geometric Deformation, Block Coordinate Descent

Links:

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

†e-mail: zdeng@cs.uh.edu

1 Introduction

Numerous research efforts have been focused on efficiently skinning mesh animations. Among many proposed techniques, *linear blend skinning* (LBS) is widely known to be the most popular skinning computational model due to its effectiveness, simplicity, and efficiency [Lewis et al. 2000; Gain and Bechmann 2008]. It has many different names over the years including *skeleton subspace deformation*, *enveloping*, *vertex blending*, *smooth skinning* (Autodesk Maya), *bones skinning* (Autodesk 3D Studio Max), or *linear blend skinning* (the open-source Blender).

In the LBS model, skin deformation is driven by a set of bones. Every vertex is associated with the bones via a bone-vertex weight map which quantifies the influence of each bone to the vertices. The skin is deformed by transforming each vertex through a weighted combination of bone transformations from the rest pose. Assuming w_{ij} is the influence of j -th bone to the i -th vertex, p_i is the position of the i -th vertex at the rest pose, $|B|$ is the number of bones, and R_j^t and T_j^t are the rotation matrix and translation vector of the j -th bone at the t -th configuration, respectively, then the deformed i -th vertex, v_i^t , can be computed as follows:

$$v_i^t = \sum_{j=1}^{|B|} w_{ij} (R_j^t p_i + T_j^t) \quad (1)$$

Depending on specific applications, the above LBS model may impose certain constraints. The weight map w_{ij} is normally required to be convex, i.e., $w_{ij} \geq 0$ and $\sum_{j=1}^{|B|} w_{ij} = 1$. The first *non-negativity* constraint makes the transformation blending additive. The second *affinity* constraint normalizes the influences/weights to prevent over-fitting and deformation artifacts. The two constraints are critical for certain applications such as animation editing. In addition, the *sparseness* constraint on the weight map, which limits the number of non-zero bone weights per vertex, may be applied to take advantage of graphic hardware capabilities. Many applications also require R_j^t matrix to be orthogonal, e.g. animation editing, collision detection, and skeleton extraction (please refer to Section 6 for more details). The *orthogonal* constraint avoids any shearing or scaling effect on the bone transformations, thus put the transformations into the rigid group. For this reason, the bone transformation with orthogonal rotation matrix is also called the “rigid bone”.

In this work, we propose a novel example-based method (called

Smooth Skinning Decomposition with Rigid Bones (SSDR) to extract both rigid bone transformations and a sparse, convex bone-vertex weight map from a set of example poses (refer to Figure 1). Specifically, the skinning decomposition is formulated as a constrained optimization problem in which the sum-of-squared error of vertices reconstructed by the LBS model is minimized. The term *skinning decomposition* is borrowed from [Kavan et al. 2010] to refer to the inverse problem of linear blend skinning.

Compared with the state-of-the-art algorithms [James and Twigg 2005; Hasler et al. 2010] that typically treat both the orthogonal constraint for the bone transformations and the convex constraint for the bone-vertex weight map as *soft constraints*, the main contribution of this work is that our algorithm exactly enforces the two constraints as *hard constraints*. As such, our algorithm does not need to deal with the trade-off between exactly satisfying the constraints and sacrificing the reconstruction error. Specifically, our exact solution for the orthogonal bone constraint is linear, simple, and effective. Through extensive qualitative and quantitative comparisons, we show that the proposed SSDR algorithm can produce more accurate and meaningful skinning decompositions than the state of the art algorithms.

2 Related Work

Skinning Mesh Animations: Despite its wide uses, the LBS model has certain limitations such as collapsing elbow and candy-wrapper effects [Gain and Bechmann 2008] and failure of secondary deformation. Over the years, researchers proposed various techniques to overcome these limitations [Kry et al. 2002; Merry et al. 2006; Miller et al. 2010; Huang et al. 2011a]. For example, Lewis et al. [2000] pioneered the pose space deformation that treats the deformation problem as a shape interpolation problem from a set of example poses, and it allows animators intuitively add or manipulate the set of control poses to correct the deformation artifacts. Along with those skinning models, several user interfaces were proposed for interactive skinning tasks [Sloan et al. 2001; Mohr et al. 2003]. Skinning weights can also be calculated automatically by first extracting the skeleton from a static character mesh [Baran and Popović 2007] or from a set of example poses [Schaefer and Yuksel 2007; de Aguiar et al. 2008].

The intrinsic limitation of the LBS model can also be tackled by manipulating the transformation or weight matrices, including multi-weight enveloping [Wang and Phillips 2002] that modifies the transformation matrix (or called the enveloping matrix) by adding a weight to each of its entries, modifying the weight matrix by adding a scalar weight function to each bone in order to support bone stretching and twisting [Jacobson and Sorkine 2011], and non-linear skinning techniques by transforming matrices to a form of quaternion [Kavan and Žára 2005] or dual quaternion [Kavan et al. 2008]. In addition, the mesh skinning problem can also be reformulated as a regression problem, in which a regressor is learnt from examples to predict the deformation from a skeleton configuration [Wang et al. 2007; Feng et al. 2008; Kim and James 2011]. Also, physically based methods have been exploited for skinning applications [Capell et al. 2005; Kim and James 2011].

Skinning Decomposition: Several early related efforts were pursued to partially address the skinning decomposition problem. For example, Mohr and Gleicher [2003] employ linear least squares to calculate the weights for the LBS, given a set of examples poses and their corresponding bone transformations. Later, researchers [Kavan et al. 2007] proposed an efficient solution to compute the bone transformations in the dual quaternion form based on given example poses and weights. Kavan et al. [2010] officially coined the term skinning decomposition, and it refers to the problem of extracting

both the bone transformations (or the control points) and the bone-vertex influences (or the weights) from a set of example poses. In their model, the vertices of example poses, denoted as a matrix A , are decomposed to $A = TX$, where T represents the transformation and X represents the combination of the weight matrix and the rest pose. Their approach does not enforce any constraints on the transformations T and thus their approach is suitable (and often limited) to compression and GPU-accelerated high performance rendering applications.

The first reported solution to compute both bone rigid transformations and bone-vertex weights from example poses is the *skinning mesh animations* (SMA) method [James and Twigg 2005]. Although the SMA model provides a complete solution to the skinning decomposition problem; essentially, it is still a combination of two separate solutions of sub-problems rather than a unified framework. SMA is more effective for near-rigid (e.g., articulated) objects than non-rigid models since the triangle rotations can be clearly observed in the near-rigid objects. The SMA model is able to enforce hard constraints on bone rotation matrices; however, it can only enforce soft constraints on the bone-vertex weights (i.e., the affinity constraint). Due to this reason, it has to face a trade-off between guaranteeing the affinity constraint on the weights and sacrificing the reconstruction error.

Later, following this direction, Hasler et al. [2010] first segment the meshes of example poses into rigid parts by applying spectral clustering to the triangle rotations. Then, the bone transformations and bone-vertex weights are refined via an iterative process, where the bone transformations are optimized by linear least squares with soft orthogonal constraints enforced on rotation matrices, followed by a Levenberg-Marquardt optimization. The bone-vertex weights are solved by non-negative linear least squares with L1-norm minimization to achieve sparse solutions, followed by L1-norm normalization in order to enforce the affinity constraint. Since all the employed constraints are soft, the iterative process cannot guarantee its convergence in theory. In Section 5, we will compare our proposed SSDR model with both the [James and Twigg 2005] and [Hasler et al. 2010] methods.

3 Smooth Skinning Decomposition with Rigid Bones

The introduced *Smooth Skinning Decomposition with Rigid Bones* (SSDR) model of animated meshes aims to solve the inverse problem of the LBS model. Suppose we have $|t|$ example poses of a $|V|$ -vertices model, where the coordinate of the i -th vertex in the t -th example pose is denoted as v_i^t . Our SSDR takes the positions of vertices, $\{v_i^t : t = 1..|t|, i = 1..|V|\}$, as the input, and decomposes them to the bone transformations and the bone-vertex weight map, i.e. w_{ij} , R_j^t , and T_j^t in the right hand side of Eq. (1).

The example poses V can be considered as a $|V| \times |t|$ matrix, where each element is a 3×1 vector represents the 3D coordinate of a vertex in an example pose. Similarly, the rest pose P can be considered as a vector of $|V|$ vertices. The output of the algorithm is the bone-vertex influences $W = \{w_{ij}\}$ and the bone transformations $B = \{R_j^t, T_j^t\}$. The bone-vertex influences W is a sparse, non-negative $|V| \times |B|$ matrix with the sum of elements in any row equal to 1, where w_{ij} denotes the influence of the j -th bone on the i -th vertex. The transformation matrix of the j -th bone in the t -th example pose is $[R_j^t | T_j^t]$, where R_j^t denotes the 3×3 orthogonal rotation matrix and T_j^t denotes the 3×1 translation vector.

We reformulate our SSDR as a constrained least squares optimization problem of the example poses reconstruction error as follows:

$$\min_{w, R, T} E = \min_{w, R, T} \sum_{t=1}^{|t|} \sum_{i=1}^{|V|} \left\| v_i^t - \sum_{j=1}^{|B|} w_{ij} (R_j^t p_i + T_j^t) \right\|^2 \quad (2a)$$

$$\text{Subject to: } w_{ij} \geq 0, \forall i, j \quad (2b)$$

$$\sum_{j=1}^{|B|} w_{ij} = 1, \forall i \quad (2c)$$

$$|\{w_{ij} | w_{ij} \neq 0\}| \leq |K|, \forall i \quad (2d)$$

$$R_j^{t\top} R_j^t = I, \det R_j^t = 1, \forall t, j \quad (2e)$$

The objective function E in Eq. (2a) is the squared sum of the reconstruction errors for all the vertices for all the example poses. E is minimized subject to four hard constraints. The non-negativity constraint, Eq. (2b), the affinity constraint, Eq. (2c), and the sparseness constraint, Eq. (2d) are imposed on the bone-vertex influences. The orthogonal constraint Eq. (2e) ensures all the matrices, $\{R_j^t\}$, in the rotation group (i.e. the special orthogonal group $SO(3, \mathbf{R})$) in the 3D Euclidean space.

To solve our constrained least squares problem, we design a block coordinate descent algorithm [Bertsekas 1999; Huang et al. 2011b] with $|B| + 1$ blocks where one block is the bone-vertex influences W and the remaining blocks are $|B|$ bone transformations. In this method, a feasible solution is initialized at the first step and, for each iteration, the objective function is optimized with respect to one block while the other blocks are fixed. For convenience, we do the optimization of all the bone transformations sequentially and we group them as one big block update step. Through iterations the two blocks are alternatively updated until the objective function converges to a local minimum. Finally, we correct the rest pose by the linear least squares solver [Kavan et al. 2010]. Note that we do not put the rest pose correction in the iterations as done by Kavan et al. [2010] since vertex displacements in the correction are not well captured by rigid rotations. This reason may deviate the bone transformations from the true solution. The whole process is illustrated in the right hand side of Fig. 1 and detailed in Algorithm 1.

Algorithm 1 Smooth Skinning Decomposition with Rigid Bones

Input: $V = \{v_i^t\}$, $P = \{p_i\}$, $|B|$, $|K|$
Output: $W = \{w_{ij}\}$, $B = \{R_j^t | T_j^t\}$ s.t. Eq. (2)

- 1: Initialize bone transformations B
- 2: **repeat**
- 3: Update bone-vertex weight map W
- 4: Update bone transformations B
- 5: **until** Convergent OR Maximum iterations are reached
- 6: Correct the rest pose P
- 7: **return** W , B , and P

In the above block coordinate descent algorithm 1, each updating step must decrease the objective function in order to guarantee the algorithmic convergence. In fact, this is the main challenge for these updating steps, where approximate solutions are not suitable and closed-form solutions are required. Our proposed updating solutions will be detailed in Sections 3.2 and 3.3.

3.1 Initialization

Inspired by the work of [Alexa et al. 2000; Igarashi et al. 2005] that deformations should be *as-rigid-as-possible* in order to achieve realistic visual results, we assumes no bone-vertex weight blending

in the initialization step. With this assumption, each vertex is influenced by exactly one bone and its bone-vertex weight is exactly 1. Then, the initialization problem becomes clustering $|V|$ vertices into $|B|$ clusters, and all the vertices in one cluster follows the same rigid transformation.

We employ the K-means clustering algorithm for the initialization purpose, where each cluster is represented as a sequence of $|t|$ bone transformations (corresponding to the $|t|$ example poses). We first randomly initialize the clusters. Then, in the assignment step, we associate each vertex to the bone which has the smallest squared reconstruction error. In the updating step, we calculate the bone transformation based on all the associated vertices by the Kabsch algorithm [Kabsch 1978]. In this way, we find the best transformation for pose t to relate the set of the associated vertices in pose t and the rest pose. This assignment and updating process is repeated for several times (e.g., 5 times in our experiments) to achieve a sound skinning initialization.

Since the initialization step is modeled as a clustering problem, users can choose more advanced clustering algorithms such as mean shift clustering [Georgescu et al. 2003] and self-tuning spectral clustering [Zelnik-manor and Perona 2004]. Beside robustness improvement, the number of bones $|B|$ can even be automatically determined by employing these clustering algorithms. However, these solutions are more costly in terms of both time complexity and memory usage. For example, compared with K-means initialization that takes less than 3 seconds to complete, initialization by spectral clusterings could take up to an hour [James and Twigg 2005; Hasler et al. 2010].

3.2 Update Bone-Vertex Weight Map

We update the bone-vertex weight map W by fixing the bone transformations B and finding the optimized W subject to the non-negativity constraint, Eq. (2b), the affinity constraint, Eq. (2c), and the sparseness constraint, Eq. (2d). The weight map is optimized per vertex by solving the constrained least squares for each line W_i of the matrix W :

$$W_i^\top = \arg \min_x \|Ax - b\|^2 \quad (3)$$

$$\text{Subject to: } x \geq 0$$

$$\|x\|_1 = 1$$

$$\|x\|_0 \leq |K|$$

The problem is often solved by first addressing the sparseness constraint through associating a subset of bones (typically no more than $|K| = 4$ bones) with each vertex. A simple solution for the bone-vertex association is to select the bone prior to its approximation error [Mohr and Gleicher 2003; James and Twigg 2005; Kavan et al. 2010]. This greedy method only works well for nearly articulated models since the bone-vertex association only considers the effect of one bone transformation on the vertex. Additional information can also be used, for example, Schaefer and Yuksel [2007] select the associated bones from the shape and topology of the rest pose mesh or the skeleton of the subject. The sparseness constraint can also be traded off with the approximation error as a soft constraint in a sparse solver [Hasler et al. 2010].

In this work, we associate $|K|$ out of $|B|$ bones for each vertex by first solving the linear least squares with non-negativity constraint ($x \geq 0$) and affinity constraint ($\|x\|_1 = 1$), and then removing $|B| - |K|$ bone weights with least effect to the solution. Finally, the linear solver is performed one more time with the remaining $|K|$ bone weights.

The effect of the bone j on the solution W_i , e_{ij} , is calculated as the displacement caused by setting the corresponding weight w_{ij} to 0, that is, the contribution of the bone transformation j to the approximation of the vertex i :

$$e_{ij} = \|w_{ij}(R_j^t p_i + T_j^t)\|^2 \quad (4)$$

To solve the linear least squares problem with non-negativity and affinity constraints, we employ the Lawson and Hanson's *active set method* (ASM) with bound constraints, as proposed by Schaefer and Yuksel [2007]. In contrast with several solutions for calculating the skinning weights, the ASM solver does not impose any soft constraints (as used in [James and Twigg 2005] to treat the affinity constraint) and does not take exponential computational time with respect to the number of bones [Kavan et al. 2010].

We also accelerate the performance of the ASM solver by taking advantage of certain properties of our problem. Specifically, we do three modifications: (1) using the weight map in the previous iteration to initialize the ASM solver, (2) pre-computing the cross products $A^\top A$ and $A^\top b$ with the corresponding LU decomposition [Bro and De Jong 1997], and (3) pre-computing the QR decomposition of the matrix $[1 \ 1 \ \cdots \ 1]^\top$. Modification (1) takes the fact that during the iterative update process, the weight map converges to a local optimum solution; thus, the solution in each ASM solver step does not substantially change the weight map, especially the active set (zero bone weights). Modification (3) computes and stores the QR decomposition of the left hand side matrix $[1 \ 1 \ \cdots \ 1]^\top$ in the equality constraint.

3.3 Update Bone Transformations

In this step, we need to fix the bone-vertex influences and minimize the objective function, Eq. (2a), with respect to the bone rotations R and the bone translations T over the set of example poses. Since the bone transformations for every pose are independent, we can solve the minimization for each pose individually. For the pose t , the problem then becomes finding the set of bone transformations to associate the vertices in the rest pose $\{p_i\}$ to the vertices $\{v_i^t\}$ in pose t through minimization of the following objective function:

$$\min_{R^t, T^t} E^t = \min_{R^t, T^t} \sum_{i=1}^{|V|} \left\| v_i^t - \sum_{j=1}^{|B|} w_{ij}(R_j^t p_i + T_j^t) \right\|^2 \quad (5)$$

Eq. (5) can be solved by the Levenberg-Marquardt algorithm [Marquardt 1963] as proposed in [Hasler et al. 2010]. However, this algorithm is inefficient since it requires many iterations as well as an initial guess that is sufficiently close to the solution. Many other approaches have been proposed to approximate the result or tackle similar problems. For example, researchers proposed closed-form solutions to the *Absolute Orientation* problem [Horn 1987; Kabsch 1978], which is a particular case of Eq. (5) with only one bone transformation to be solved. Multiple bone transformations can be solved approximately in a similar way by adding the bone weights for each vertex [Müller et al. 2005; Zhu and Gortler 2007; Schaefer and Yuksel 2007]. However, this Weighted Absolute Orientation solution cannot provide the exact solution to the original problem (Eq. (5)) as illustrated in the left hand side of Fig. 2. For this reason, employing Weighted Absolute Orientation cannot guarantee for the convergence of our main algorithm 1. Recently, Kavan et al. [2007] proposed a closed-form solution by transforming Eq. (5) to the Dual Quaternion space. Unfortunately, due to the non-linear nature of this transformation, it cannot preserve the error metric in the original space.

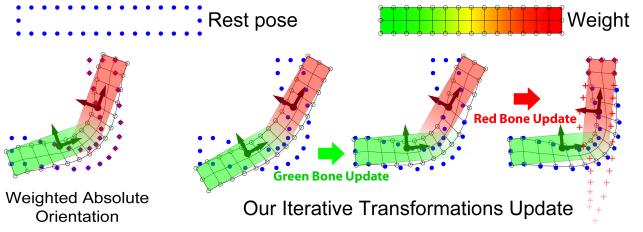


Figure 2: Comparison of the Weighted Absolute Orientation solution [Horn 1987; Kabsch 1978] on the left and our iterative bone transformations update on the right. The blue dots indicate vertices in the example pose. The red plus signs (+) indicate the target positions for the red bone transformation fitting. While the Weighted Absolute Orientation only provides an approximate solution due to the deformation of the target positions in the example pose, our iterative converges to the true optimum solution by calculating the target positions as the residual of the deformation caused by the remaining bones (the green bone).

To guarantee the non-increasing of the global objective function, Eq. (2a), we update the bone transformations one by one instead of updating all of them at once. For the example pose t , updating transformation of bone \hat{j} while keeping the remaining $|B|-1$ bones can be solved by minimizing the following equation:

$$E_{\hat{j}}^t = \sum_{i=1}^{|V|} \left\| v_i^t - \sum_{j=1, j \neq \hat{j}}^{|B|} w_{ij}(R_j^t p_i + T_j^t) - w_{i\hat{j}}(R_{\hat{j}}^t p_i + T_{\hat{j}}^t) \right\|^2$$

Let q_i^t be the deformation residual of vertex i in example pose t caused by the remaining $|B|-1$ bones,

$$q_i^t = v_i^t - \sum_{j=1, j \neq \hat{j}}^{|B|} w_{ij}(R_j^t p_i + T_j^t) \quad (6)$$

The problem of finding the optimal transformation then becomes:

$$\{R_{\hat{j}}^t, T_{\hat{j}}^t\}^* = \arg \min_{R_{\hat{j}}^t, T_{\hat{j}}^t} \sum_{i=1}^{|V|} \left\| q_i^t - w_{i\hat{j}}(R_{\hat{j}}^t p_i + T_{\hat{j}}^t) \right\|^2 \quad (7)$$

$$\text{Subject to: } R_{\hat{j}}^{t\top} R_{\hat{j}}^t = I, \det R_{\hat{j}}^t = 1$$

The problem, Eq. (7), is similar to the Weighted Absolute Orientation problem except the difference in the scope of the weights term, i.e. the weights only affect p_i but not q_i^t . Inspired by the work of [Horn 1987; Kabsch 1978], we can first remove the translation $T_{\hat{j}}^t$ and then solve for the optimum rotation. This is done by translating all the vertices to bring the center of rotation (CoR) to the origin for each set of vertices as computed in Eq. (8). Note that our CoR is different from the CoR computed in [Horn 1987; Kabsch 1978], since the latter is simply the centroid of the two sets of vertices.

$$\bar{p}_i = p_i - p_*, \quad \bar{q}_i^t = q_i^t - w_{i\hat{j}} q_*^t \quad (8a)$$

$$\text{Where: } p_* = \frac{\sum_{i=1}^{|V|} w_{ij}^2 p_i}{\sum_{i=1}^{|V|} w_{ij}^2}, \quad q_*^t = \frac{\sum_{i=1}^{|V|} w_{ij} q_i^t}{\sum_{i=1}^{|V|} w_{ij}^2} \quad (8b)$$

The above translation removes T_j^t from the optimization problem and leaves only the rotation R_j^t to be minimized. Similar to [Kabsch 1978], we find the optimal rotation using Singular Value Decomposition (SVD). First, we construct two matrices $P = [w_{1j}\bar{p}_1^t \dots w_{|V|j}\bar{p}_{|V|}^t]$ and $Q = [\bar{q}_1^t \dots \bar{q}_{|V|}^t]$ ($P, Q \in \mathbb{R}^{3 \times |V|}$). Then, SVD is performed on PQ^\top as follows:

$$PQ^\top = \sum_{i=1}^{|V|} w_{ij}\bar{p}_i\bar{q}_i^\top = \mu\Sigma\vartheta^\top \quad (9)$$

Finally, we obtain the optimal rotation and translation as follows:

$$R_j^t = \vartheta\mu^\top, \quad T_j^t = q_*^t - R_j^t p_* \quad (10)$$

The proof for the optimal CoR in Eq. (8) and the optimal transformation in Eq. (10) are detailed in **Appendices A & B**, respectively. The summary of our bone transformation updating process is described in Algorithm 2.

3.3.1 Bone Transformation Re-Initialization

At the bone-vertex influence update step, some bones maybe do not have major influence on any vertices (called “*insignificant bones*” in this writing). In such a case, their bone transformations cannot be accurately estimated. This is similar to the case of having less than 3 points in the Absolute Orientation problem, where the rotation cannot be uniquely determined. In addition, having those insignificant bones does not have major impact on the reconstruction error; thus, it is unnecessary to keep them. For this reason, instead of estimating its rotation and translation, we randomly re-initialize its bone transformation if a bone is determined as an insignificant bone. Moreover, the re-initialization can also prevent the iteration process to get stuck at a local minimum.

Specifically, we determine the j -th bone is an insignificant bone if $\sum_{i=1}^{|V|} w_{ij}^2 < \epsilon$. Here, the sum $\sum_{i=1}^{|V|} w_{ij}^2$ is the denominator in equation (8b). If this sum is small, estimation of rotation center p_* and q_*^t and estimation of PQ^\top in equation (9) will be non-robust. We empirically set the threshold $\epsilon = 3$ since this is the minimum number of vertices required in the Absolute Orientation problem. If the j -th bone needs to be re-initialized, we assign it to the vertex i with the largest reconstruction error. Then, we find 20 nearest vertices of i in the rest pose, denoted as $N(i)$. Finally, we use $N(i)$ to re-initialize the bone transformation $\{R_j^t | T_j^t\}$ for all the example poses t by using the Kabsch algorihtm [Kabsch 1978]. Re-initialization does not guarantee strict decreasing of the objective function so it could create a loop. We avoid this by limiting the maximum number of re-initializations. In our experiments, the maximum number of re-initializations is empirically set to 10.

4 Results

The test datasets: To demonstrate and evaluate our SSDR model, we used publicly available triangle mesh datasets, including 12 datasets from the “Deformation Transfer” [Sumner and Popović 2004], 2 datasets from the “Geometry Videos” [Briceño et al. 2003], 1 dataset (chickenCrossing) from the “Skinning Mesh Animations” [James and Twigg 2005], and 1 dataset (pjump) from the “Wavelet Compression” [Guskov and Khodakovsky 2004]. Based on properties of these datasets, we roughly divide the datasets into two categories: *articulated* (near-rigid) models and *elastic* (highly-deformable) models. Details of the used 16 datasets are described in Table 1.

Algorithm 2 Update bone transformations B

```

Input:  $V = \{v_i^t\}$ ,  $P = \{p_i\}$ ,  $W = \{w_{ij}\}$ 
Output:  $B = \{R_j^t | T_j^t\}$  s.t. Eq. (2e)
1: for  $t = 1 \rightarrow |t|$  do
2:   for  $j = 1 \rightarrow |B|$  do
3:     if  $\sum_{i=1}^{|V|} w_{ij}^2 \geq \epsilon$  then
4:       Calculate  $q_i, \forall i$  by equation (6)
5:       Calculate  $p^*, q^*, \bar{p}_i$  and  $\bar{q}_i \forall i$ , by equation (8)
6:       Perform SVD by equations (9)
7:       Calculate rotation  $R_j^t$  and by equations (10)
8:     else
9:       Re-initialize the bone transformation
10:    end if
11:   end for
12: end for

```

Name	V	t	Category
camel-collapse	21887	53	Elastic
camel-gallop	21887	48	Articulated
camel-poses	21887	10	Articulated
cat-poses	7207	9	Articulated
chickenCrossing	3030	400	Articulated
elephant-gallop	42321	48	Articulated
elephant-poses	42321	10	Articulated
face-poses	29299	9	Elastic
flamingo-poses	26907	10	Articulated
horse-collapse	8431	53	Elastic
horse-gallop	8431	48	Articulated
horse-poses	8431	10	Articulated
lion-poses	5000	9	Articulated
pcow	2904	204	Elastic
pdance	7061	201	Articulated
pjump	15830	222	Articulated

Table 1: The test datasets used in this work. $|V|$ denotes the number of vertices and $|t|$ denotes the number of example poses.

Error metric: To quantitatively evaluate our SSDR model we choose the error metric proposed in [Kavan et al. 2010] over the one in [James and Twigg 2005], because the former is less sensitive to global motions of the models and thus more robust than the latter, as reported in [Kavan et al. 2010]. Specifically, we first resize all the datasets so that the rest poses are tightly enclosed by a unit sphere. Then, the error metric is calculated from the value E of the objective function in Eq. (2a) as follows:

$$E_{RMS} = 1000 \sqrt{\frac{E}{3 \cdot |V| \cdot |t|}}$$

Configurations: For each dataset obtained from the “Deformation Transfer” [Sumner and Popović 2004], the rest pose P is set to be the provided reference pose. For each of the other datasets, the rest pose P is chosen to be the first example pose. The number of bones, $|B|$, is manually determined for experiments. The maximum number of non-zero weights per vertex $|K|$ is set to 4 as commonly used in [James and Twigg 2005]. In the experiments, we denote this number $|B|$ as a subscript of the dataset name. In our implementation, we stop the algorithm if within one iteration the the objective function E is not improved by 1% and the bone transformation re-initialization is not performed.

Figure 3 shows our results on four articulated models and two elastic models. We use the skin colors to illustrate the bone-vertex influence map. At the rest pose, we also draw a coordinate system for each bone to illustrate its bone transformation, and the origin of the coordinate system is put at the vertex that is most influenced by the bone. In each example pose, the coordinate system for each bone is deformed in accordance with its bone transformation.

Figure 4 shows the changes of the error metric during the iteration process. We can see that the error does not increase, except after each bone transformation re-initialization; however, it is quickly converged back to the optimum solution. We also notice that the

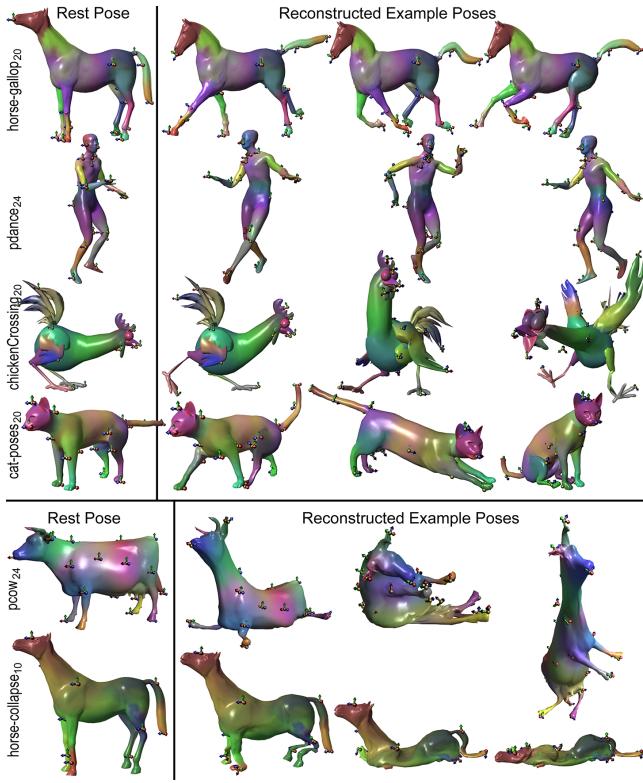


Figure 3: Results of our Smooth Skinning Decomposition with Rigid Bones (SSDR) on articulated models (the top four models) and elastic models (the bottom two models).

bone transformation re-initialization happens more frequently if the number of bones is set to an overlarge value. For example, since the human model in the pdance dataset has much fewer than 24 rigid components, i.e. much less than 24 rotational joints in the input sequence, re-initialization happened 3 times in pdance₂₄. Meanwhile, there is no re-initialization in the other 5 test cases. We also compare the rate of convergence (RoC) of our bone transformation update with the well-known Levenberg-Marquardt (LM) algorithm [Marquardt 1963]. Specifically, we use the Levmar open-source package [Lourakis 2004] as the implementation of the LM optimization. For each transformation update step with LM, we initialize the solution using the result from the previous step and then perform 20 iterations. From the error curves (Fig. 4), we can see that our algorithm can reduce the error as good as the LM optimization. However, since our algorithm takes much less execution time (shown in Fig. 4), the RoC of our algorithm is much higher than the RoC of LM. In addition, the similarity in changes of the error shows that a single-pass bone transformation update can provide a sufficient RoC.

5 Comparisons

We compared our proposed SSDR model with two state-of-the-art skinning decomposition approaches: the *Skinned Mesh Animations* (SMA) with rigid bones [James and Twigg 2005] and the *Learning Skeletons for Shape and Pose* (LSSP) [Hasler et al. 2010]. Both of them impose orthogonal constraints on the bone rotation matrices (rigid bones), convex constraints and sparseness constraints on the bone-vertex weight map. To ensure a fair comparison, all the three methods ran on a single thread, without GPU-

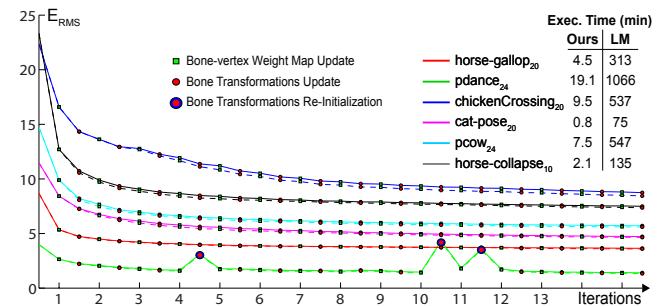


Figure 4: Changes of the error metric in the first 15 iterations of our algorithm (solid lines) versus Levenberg-Marquardt (LM) algorithm (dashed lines). While the changes of the error are almost the same for the two methods, the execution time of our algorithm is much less than that of LM. Thus, the convergence rate of our algorithm is much higher.

based acceleration. We also used all the parameter settings published by the original authors in our implementations. It is noteworthy that the work of [Kavan et al. 2010] was not included in this comparison because it can only handle the case of non-rigid bones, while the focus of this comparison experiment is skinning decomposition with rigid bones.

The SMA was implemented in C++. We used the open source library *Mean Shift Clustering* with locality sensitive hashing (LSH) [Georgescu et al. 2003]. The two parameters for LSH, i.e., K and L , were set to 70 and 200, respectively. The bandwidth h was set to $h = 9|t|\epsilon$, where $|t|$ is the number of example poses and $\epsilon = 0.05$. To estimate the bone-vertex weights, we used non-negative least squares.

The LSSP was implemented in MATLAB. We used the *Self-Tuning Spectral Clustering* [Zelnik-manor and Perona 2004] for the initialization step with automatic local scaling on distance matrix. On the weight matrix optimization step, we employed the $L1/L2$ + minimizer [Zhang 2009], where the parameter ρ was set to 0.1 and the A matrix was normalized. At the factorization step, we performed 10 iterations.

Figure 5 shows the comparison results on three models generated by SSDR, SMA, and LSSP. Due to the high deformability of the models, SMA fails to associate vertices into rigid bones and hefty distortion can be observed on the reconstructed poses, especially on the camel-collapse₁₁ model. Although LSSP can estimate the global deformation well, certain details are not correctly reconstructed, e.g., on the horse-collapse₁₀ model.

We also evaluated the accuracy and performance of the three methods (SSDR, SMA, and LSSP) in a quantitative way including the error metric, the number of required bones, execution time, etc. (shown in Table 2). In Table 2, the number of bones is denoted by the subscript of the dataset name. The error E_{RMS} as well as the error after rank-5 EigenSkin corrections [Kry et al. 2002] are reported where the numbers in the parentheses are the EigenSkin correction errors. All the running times were measured on the same computer with a 2GHz single core CPU. Note that the number of bones is automatically estimated in SMA, thus its results are not available for many specific numbers of bones. The results of LSSP are also not available for models with more than 10K vertices due to the large memory requirement in the Self-Tuning Spectral Clustering algorithm at the initialization step.

From Table 2, we can see that SSDR clearly outperforms SMA

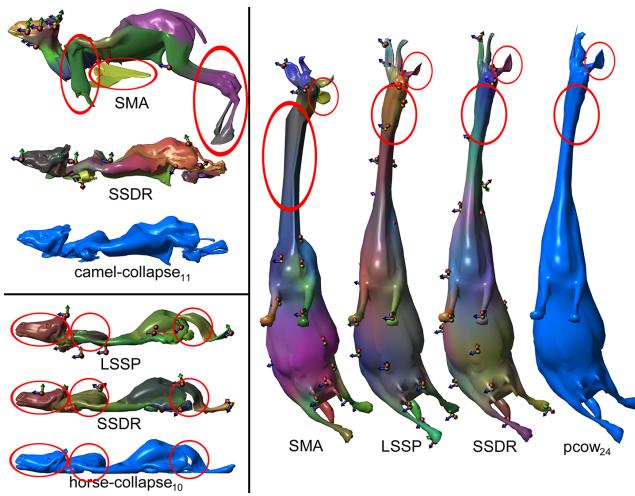


Figure 5: Comparisons of the skinning decomposition results among the proposed SSDR, SMA [James and Twigg 2005], and LSSP [Hasler et al. 2010]. LSSP is unable to run on the camel-collapse due to the large size of this dataset and SMA is unable to configure with 10 bones for the horse-collapse dataset. The example poses are rendered in blue. Significant distortion areas are indicated in red circles.

Dataset[# of bones]	Approximation error E_{RMS}			Execution time (minutes)		
	SMA	LSSP	SSDR	SMA	LSSP	SSDR
camel-collapse ₁₁	125.3 (4)	-	5.4(1.7)	13.8	-	7.4
camel-collapse ₂₀	-	-	4(1.4)	-	-	15.1
camel-gallop ₂₀	-	-	3.7(1.5)	-	-	13.9
camel-gallop ₂₉	17.6 (1.9)	-	3(1.3)	25.2	-	21.9
camel-poses ₁₀	-	-	10.1(3.9)	-	-	1
camel-poses ₂₅	8.3 (1.8)	-	2.7(1.2)	9.4	-	4.6
cat-poses ₁₅	-	10.7(6.1)	6.5(2.7)	-	302.8	0.7
cat-poses ₂₀	-	-	4.7(2)	-	-	1
cat-poses ₂₅	8.5 (3.1)	6.2(3.3)	3.4(1.4)	0.7	371.7	1.5
chickenCrossing ₂₀	-	10.1(9.7)	8.7(5.7)	-	1128.9	14.8
chickenCrossing ₂₈	12.5 (4.2)	6.2(5.1)	8.1(5.4)	14.1	1165.4	24
elephant-gallop ₂₀	-	-	4.3(2.3)	-	-	27.5
elephant-gallop ₂₇	5.6 (1.9)	-	2.7(1.3)	56.1	-	53.6
elephant-poses ₁₀	-	-	8.2(4.2)	-	-	3.2
elephant-poses ₂₁	5.8 (2.2)	-	3.2(1.5)	29.4	-	8.1
face-poses ₂₇	-	-	7(3.6)	-	-	7
face-poses ₃₆	37.6 (8.5)	-	-	3.6	-	-
flamingo-poses ₁₀	-	-	4.8(2.2)	-	-	2
flamingo-poses ₂₃	5.7 (1.7)	-	1.7(0.8)	16.3	-	5.1
horse-collapse ₃	139.5 (5.1)	51(3.1)	26.5(4.1)	3.3	856.6	0.8
horse-collapse ₁₀	-	15.5(2.6)	7.4(2.1)	-	802	2.6
horse-collapse ₂₀	-	6(2)	5(1.6)	-	1088.5	5.7
horse-gallop ₂₀	-	15.7(5.3)	3.6(1.8)	-	859.9	5.4
horse-gallop ₃₃	9.5 (1.5)	12.5(4.6)	2.2(1.1)	3.8	911	9.8
horse-poses ₂₀	-	20.6(7.8)	3.8(1.8)	-	461.8	1.4
horse-poses ₄₂	4.7 (1.4)	2.2(1.2)	-	2.1	475.1	-
lion-poses ₁₀	-	22.1(11)	11.3(5)	-	201.3	0.2
lion-poses ₂₁	62.8 (5.7)	7.7(3.9)	4.4(2.2)	0.6	360.2	0.8
pcow ₁₀	-	16.5(13.2)	14.4(11.9)	-	549.2	2.4
pcow ₂₄	24.8 (13.2)	7.2(6.7)	5.7(4.8)	3.8	564.5	8.9
pdance ₁₀	-	8(4.9)	6.3(3.4)	-	2177.7	5.8
pdance ₂₄	3.8 (1.6)	3.4(2.3)	1.3(0.8)	22	2446.8	28.3
pjump ₂₀	-	-	6.7(4.7)	-	-	42.7
pjump ₄₀	15.3 (6.7)	-	4.5(3.4)	30.5	-	104.1

Table 2: Rigid bone skinning decomposition results of Skinning Mesh Animation (SMA) with rigid bones [James and Twigg 2005], Learning Skeletons for Shape and Pose (LSSP) [Hasler et al. 2010], and our SSDR algorithm. The results after rank-5 EigenSkin correction [Kry et al. 2002] are also reported in the parentheses.

and LSSP. Especially on the elastic models (i.e., camel-collapse₁₁, face-poses, horse-collapse₃, and pcow₂₄), SSDR generates significantly smaller errors than SMA. The reason is that SMA computes the bone transformations by first clustering the triangle rotation sequences into rigid transformation groups. Thus, it only works well if the model can be divided into nearly rigid parts. Our results also outperform LSSP results due to the following two limitations of LSSP: (1) LSSP employs the soft constraint for sparseness; and (2) LSSP does not consider the affinity constraint during the optimization, and the affinity constraint is only enforced through post-normalization. Also, even LSSP supports for the sparseness weight map, it does not guarantee the maximum number of non-zero weights per vertex, which could be an issue for the implementation of hardware accelerated skinning.

Among the three approaches (SSDR, SMA, and LSSP), LSSP has the lowest performance due to its MATLAB implementation and the complexity of its clustering and optimization algorithm. In general, SSDR performs slightly slower than SMA. This is one limitation of the current SSDR model. However, SSDR can be efficiently accelerated via parallel implementations such as exploiting multi-core CPUs or GPUs. This can be done by parallelizing certain high computational cost operations such as calculating vertex transformations or matrix multiplications. Fortunately, most of those calculations are matrix operations, which are relatively easy to be parallelized. By contrast, parallelizing SMA and LSSP will be more challenging since it requires to parallelize spectral clustering and non-linear optimizers.

6 Discussion

6.1 Rigid Bones versus Flexible Bones

In many skinning decomposition methods, the rotation part of the bone transformation is not subject to the orthogonal constraint [James and Twigg 2005; Kavan et al. 2010]. This type of bone is referred to as a *Flexible Bone*. In general, solving for the flexible bone transformations is easier than solving for the rigid bone transformations. The flexible bones can also approximate the deformation better than the rigid bones since the rigid bones do not have shearing and scaling transformations. In Fig. 6, we show an example of skinning with rigid bones versus flexible bones. In this example, the skinning decomposition with flexible bones is solved by replacing the bone transformation update in our SSDR algorithm (line 4 in Algorithm 1) with the linear least squares solver proposed in [James and Twigg 2005]. Due to the advantages in accuracy and simplicity, skinning with flexible bones can be applied for hardware accelerated rendering and animation compression [James and Twigg 2005; Kavan et al. 2010].

However, rigid bones provide better support for applications of skinning decomposition such as animation editing, collision detection, and skeleton extraction, as illustrated in Fig. 7. In animation editing tasks, the most intuitive way for users is to drag manipulators in a graphical interface. While the rotation manipulator and the translation manipulator can be used to edit the bone transformations; it is non-trivial to directly edit the flexible bone transformations, especially editing the shearing effect. Due to the shearing effect, bounding spheres embedded with flexible bones are distorted, which requires more complicated solutions for collision detection. In addition, since there is no center of rotation for non-rigid transformations, the joint between two bones does not exist. For this reason, skeleton extraction algorithms [Schaefer and Yuksel 2007; Hasler et al. 2010] cannot work. In addition, rigid bone transformations can be represented in more compact forms, e.g. Euler angles or quaternions, than flexible bone transformations.

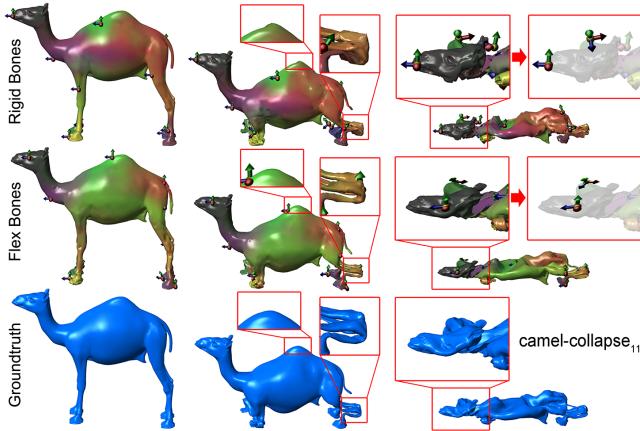


Figure 6: Skinning decomposition comparisons of *camel-collapse₁₁* with rigid bones versus flexible bones. The skinning with flexible bones approximates the example poses slightly better than the skinning with rigid bones (magnified in red rectangles) since flexible bones could have shearing and scaling transformations (shown on the right hand side).

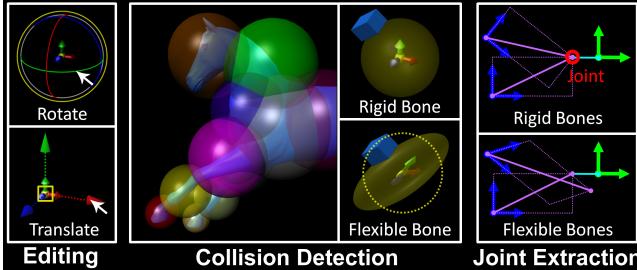


Figure 7: Selected applications that can be better supported by rigid bone transformations. **Editing:** Graphical manipulators can be used to edit rigid bone transformations, while it is non-trivial and difficult to directly edit flexible bone transformations. **Collision Detection:** Additional solutions are needed to handle the distorted objects within the bounding spheres of the flexible bone transformations. **Joint Extraction:** Since there is no center of rotation, the joint between two bones does not exist.

6.2 Skinning Weight Solver

In Section 3.2, we describe a new method (solver) to solve the bone-vertex weight map with the convex constraint and the sparseness constraint (Eq. (3)). Therefore, we also compared the performance of the proposed skinning weight solver with the solution used by Kavan et al. [2010]. We use the linear least squares to solve the flexible bone transformations for both the methods since [Kavan et al. 2010] only supports flexible bones. We also apply the rest pose optimization [Kavan et al. 2010] before all the bone-vertex weight map update steps (line 3 in Algorithm 1). As shown in Table 3, our solution is slower since it requires to solve the dense least squares for associating bones to vertices. However, we can observe that our proposed weight solver can measurably reduce the estimation error, compared with the solver used in [Kavan et al. 2010].

7 Conclusions

In this paper, we propose a novel smooth skinning decomposition with rigid bones (SSDR) model to find rigid bone transformations

Dataset[No. of bones] [Kavan et al. 2010]	Approximation error E_{RMS} Our method	Execution time (minutes) [Kavan et al. 2010]	Our method	
camel-collapse ₁₁	4.1 (1.6)	3.4(1.5)	1.3	6.5
camel-collapse ₂₀	3 (2.8)	2.7(1.2)	2.1	14.2
camel-gallop ₁₀	3 (1.7)	2.8(2.6)	1.1	5.5
camel-gallop ₂₀	1.7 (1.5)	1.7(1.5)	1.7	15
camel-gallop ₂₉	1.3 (1)	1.2(0.8)	1.7	24.2
camel-poses ₁₀	4.1 (1.9)	4.1(2.4)	0.7	2.3
camel-poses ₂₅	1.9 (0.9)	1.7(0.9)	0.9	8.4
cat-poses ₁₀	7.1 (4.2)	6(3.7)	0.2	0.5
cat-poses ₁₅	4.7 (2.6)	4.2(2.8)	0.2	0.8
cat-poses ₂₀	3.5 (2.1)	3.3(1.9)	0.2	1.3
cat-poses ₂₅	2.8 (1.8)	2.5(1.5)	0.3	1.9
chickenCrossing ₂₀	6.2 (6.5)	7(9.2)	1	11.8
chickenCrossing ₂₈	5.3 (10.3)	4.2(13.4)	1.2	24.6
elephant-gallop ₁₀	5.4 (4.2)	3.9(3.4)	2.7	12.6
elephant-gallop ₂₀	3 (2.7)	2.2(1.9)	3.1	31.7
elephant-gallop ₂₇	2.3 (1.6)	1.6(1.4)	3.4	50.5
elephant-poses ₁₀	7.7 (5.1)	6.6(3.3)	1.2	5
elephant-poses ₂₁	3 (1.4)	2.8(1.6)	1.7	17
face-poses ₂₇	3.9 (2.3)	3.2(2)	1.2	13.9
flamingo-poses ₁₀	3.5 (2.5)	3(2)	0.8	2.8
flamingo-poses ₂₃	1.7 (0.6)	1.2(0.8)	1	11.2
horse-collapse ₁₀	5.7 (2.2)	4.6(1.7)	0.5	2.1
horse-collapse ₂₀	3.8 (3.6)	3.1(2.8)	0.6	4.7
horse-gallop ₁₀	5.4 (3.4)	4.8(5)	0.4	1.9
horse-gallop ₂₀	2.6 (1.5)	2.3(1.6)	0.5	6
horse-gallop ₃₃	1.6 (1)	1.6(1.4)	0.8	10.4
horse-poses ₁₀	6.2 (3)	5.5(3.4)	0.3	0.6
horse-poses ₂₀	2.6 (1.2)	2.6(1.5)	0.3	1.6
lion-poses ₁₀	6.9(4)	5.8(3.2)	0.1	0.3
lion-poses ₂₁	3.1 (1.4)	3(1.8)	0.2	0.9
pcow ₁₀	7.4 (7.6)	7.3(15.6)	0.9	2.5
pcow ₂₄	3.4 (3.3)	3.3(3.8)	1.3	7.4
pdance ₁₀	3.2 (2)	4.2(4.2)	0.7	6
pdance ₂₄	1.1 (0.7)	1.4(1.5)	1	22.8
pjump ₂₀	4.5 (4.4)	4.2(3.3)	9.3	36.1
pjump ₄₀	2.8 (3.3)	2.6(2.4)	13	94.2

Table 3: Comparisons between our proposed skinning weight solver and the solver used in [Kavan et al. 2010]. Both the methods only find the flexible bone transformations. The results after rank-5 EigenSkin correction [Kry et al. 2002] are also reported in the parentheses.

and a sparse, convex bone-vertex weight map from a set of example poses. Specifically, we formulate the problem as a least squares optimization with hard constraints. Then, this optimization problem is further solved by a block coordinate descent method that involves an iterative process of updating the weight map and the bone transformations alternatively.

The major contribution of this work is to introduce a closed-form solution to update the bone transformations with orthogonal constraint (rigid bones). The bone transformation update step in our approach is linear, simple, and effective. The orthogonal constraint on the rotation matrices can play a crucial role in many skinning decomposition applications such as animation editing, collision detection, and skeleton extraction. We also propose a bone-vertex weight map solver that is more robust for highly deformable models. Through qualitative and quantitative comparisons and evaluations, we demonstrate that the introduced SSDR model can significantly outperform the state-of-the-art skinning decomposition algorithms in terms of accuracy and applicability.

However, the current SSDR model still has certain limitations. The first limitation is its relatively high computational cost, although it can be alleviated, to a certain extent, via GPUs or multi-core CPU based parallel implementation. The second limitation is, although in the current SSDR model users can automatically estimate an initial number of bones using the method proposed in [James and Twigg 2005]; for some cases, the automatically estimated bone

number may not lead to the desired decomposition results. In these cases, like the previous work [Hasler et al. 2010; Kavan et al. 2010], users still need to manually specify an appropriate number of bones as an input parameter, which often depends on the prior knowledge of the mesh model.

Acknowledgements

This research is supported in part by NSF IIS-0915965 and unrestricted research gifts from Google and Nokia. We would like to thank Robert Sumner, Jovan Popovic, Hugues Hoppe, Doug James, and Igor Guskov for providing the mesh sequences used in this work. “The chicken character was created by Andrew Glassner, Tom McClure, Scott Benza, and Mark Van Langeveld. This short sequence of connectivity and vertex position data is distributed solely for the purpose of comparison of geometry compression techniques.” Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the agencies.

References

- ALEXA, M., COHEN-OR, D., AND LEVIN, D. 2000. As-rigid-as-possible shape interpolation. In *Proc. of ACM SIGGRAPH'00*, 157–164.
- BARAN, I., AND POPOVIĆ, J. 2007. Automatic rigging and animation of 3D characters. *ACM Trans. Graph.* 26 (July).
- BERTSEKAS, D. P. 1999. *Nonlinear Programming*, 2nd ed. Athena Scientific, Sept.
- BRICEÑO, H. M., SANDER, P. V., McMILLAN, L., GORTLER, S., AND HOPPE, H. 2003. Geometry videos: a new representation for 3d animations. In *SCA'03: Proc. of Symposium on Computer Animation*, 136–146.
- BRO, R., AND DE JONG, S. 1997. A fast non-negativity-constrained least squares algorithm. *Journal of Chemometrics* 11, 5, 393–401.
- CAPELL, S., BURKHART, M., CURLESS, B., DUCHAMP, T., AND POPOVIĆ, Z. 2005. Physically based rigging for deformable characters. In *SCA'05: Proc. of Symposium on Computer Animation 2005*, 301–310.
- 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 (April).
- FENG, W.-W., KIM, B.-U., AND YU, Y. 2008. Real-time data driven deformation using kernel canonical correlation analysis. *ACM Trans. Graph.* 27 (August), 91:1–91:9.
- GAIN, J., AND BECHMANN, D. 2008. A survey of spatial deformation from a user-centered perspective. *ACM Trans. Graph.* 27 (November), 107:1–107:21.
- GEORGESCU, B., SHIMSHONI, I., AND MEER, P. 2003. Mean shift based clustering in high dimensions: a texture classification example. In *ICCV'03*, 456–463.
- GUSKOV, I., AND KHODAKOVSKY, A. 2004. Wavelet compression of parametrically coherent mesh sequences. In *SCA'04: Proc. of Symposium on Computer Animation*, 183–192.
- 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.
- HORN, B. K. P. 1987. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A* 4, 4, 629–642.
- HUANG, H., ZHAO, L., YIN, K., QI, Y., YU, Y., AND TONG, X. 2011. Controllable hand deformation from sparse examples with rich details. In *SCA'11: Proc. of Symposium on Computer Animation 2011*, 73–82.
- HUANG, Q., KOLTUN, V., AND GUIBAS, L. 2011. Joint shape segmentation with linear programming. *ACM Trans. Graph.* 30, 125:1–125:12.
- IGARASHI, T., MOSCOVICH, T., AND HUGHES, J. F. 2005. As-rigid-as-possible shape manipulation. In *ACM SIGGRAPH 2005 Papers*, 1134–1141.
- JACOBSON, A., AND SORKINE, O. 2011. Stretchable and twistable bones for skeletal shape deformation. *ACM Trans. Graph.* 30 (Dec.), 165:1–165:8.
- JAMES, D. L., AND TWIGG, C. D. 2005. Skinning mesh animations. *ACM Trans. Graph.* 24 (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 ŽÁRA, J. 2005. Spherical blend skinning: a real-time deformation of articulated models. In *I3D'05: Proc. of Symp. on Interactive 3D Graphics and Games*, 9–16.
- KAVAN, L., McDONNELL, R., DOBBYN, S., ŽÁRA, J., AND O’SULLIVAN, C. 2007. Skinning arbitrary deformations. In *I3D'07: Proc. of Symp. on Interactive 3D Graphics and Games*, 53–60.
- KAVAN, L., COLLINS, S., ŽÁRA, J., AND O’SULLIVAN, C. 2008. Geometric skinning with approximate dual quaternion blending. *ACM Trans. Graph.* 27 (November), 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, T., AND JAMES, D. L. 2011. Physics-based character skinning using multi-domain subspace deformations. In *SCA'11: Proc. of Symposium on Computer Animation*, 63–72.
- KRY, P. G., JAMES, D. L., AND PAI, D. K. 2002. EigenSkin: real time large deformation character skinning in hardware. In *SCA'02: Proc. of Symposium on Computer Animation*, 153–159.
- LEWIS, J. P., CORDNER, M., AND FONG, N. 2000. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proc. of SIGGRAPH'00*, 165–172.
- LOURAKIS, M., 2004. levmar: Levenberg-marquardt nonlinear least squares algorithms in C/C++. [web page] <http://www.ics.forth.gr/~lourakis/levmar/>, July. [Accessed on 31 Jan. 2005].
- MARQUARDT, D. W. 1963. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *SIAM Journal on Applied Mathematics* 11, 2, 431–441.
- MERRY, B., MARAIS, P., AND GAIN, J. 2006. Animation space: A truly linear framework for character animation. *ACM Trans. Graph.* 25 (October), 1400–1423.

- MILLER, C., ARIKAN, O., AND FUSSELL, D. 2010. Frankenrigs: building character rigs from multiple sources. In *I3D'10: Proc. of Symp. on Interactive 3D Graphics and Games*, 31–38.
- MOHR, A., AND GLEICHER, M. 2003. Building efficient, accurate character skins from examples. *ACM Trans. Graph.* 22 (July), 562–568.
- MOHR, A., TOKHEIM, L., AND GLEICHER, M. 2003. Direct manipulation of interactive character skins. In *I3D'03: Proc. of Symp. on Interactive 3D Graphics*, 27–30.
- MÜLLER, M., HEIDELBERGER, B., TESCHNER, M., AND GROSS, M. 2005. Meshless deformations based on shape matching. *ACM Trans. Graph.* 24, 3 (July), 471–478.
- SCHAEFER, S., AND YUKSEL, C. 2007. Example-based skeleton extraction. In *SGP'07: Proc. of Eurographics Symposium on Geometry Processing*, 153–162.
- SLOAN, P.-P. J., ROSE, III, C. F., AND COHEN, M. F. 2001. Shape by example. In *I3D'01: Proc. of Symp. on Interactive 3D Graphics*, 135–143.
- SUMNER, R. W., AND POPOVIĆ, J. 2004. Deformation transfer for triangle meshes. *ACM Trans. Graph.* 23 (August), 399–405.
- WANG, X. C., AND PHILLIPS, C. 2002. Multi-weight enveloping: least-squares approximation techniques for skin animation. In *SCA'02: Proc. of Symposium on Computer Animation*, 129–138.
- WANG, R. Y., PULLI, K., AND POPOVIĆ, J. 2007. Real-time enveloping with rotational regression. *ACM Trans. Graph.* 26 (July).
- ZELNIK-MANOR, L., AND PERONA, P. 2004. Self-tuning spectral clustering. In *Advances in Neural Information Processing Systems 17*, MIT Press, 1601–1608.
- ZHANG, Y. 2009. User's guide for YALL1: Your ALgorithms for L1 optimization. Tech. rep., Rice University, May.
- ZHU, Y., AND GORTLER, S. J. 2007. 3D deformation using moving least squares. harvard computer science technical report: TR-10-07. Tech. rep., Cambridge, MA.

Appendices

A Finding the Center of Bone Rotation

In this appendix, we will prove the translation in Eq. (8) can remove the translation parameter from the optimization problem (7). Expanding the objective function (7) and substituting \bar{q}_i^t, \bar{p}_i from Eq. (8a) yields:

$$\begin{aligned} E_j^t &= \sum_{i=1}^{|V|} \|q_i^t - w_{ij}(R_j^t p_i + T_j^t)\|^2 \\ &= \sum_{i=1}^{|V|} \|q_i^t - w_{ij}q_*^t + w_{ij}q_*^t - w_{ij}[(R_j^t(p_i - p_* + p_*) + T_j^t)]\|^2 \\ &= \sum_{i=1}^{|V|} \|\bar{q}_i^t + w_{ij}q_*^t - w_{ij}[R_j^t(\bar{p}_i + p_*) + T_j^t]\|^2 \\ &= \sum_{i=1}^{|V|} \|\bar{q}_i^t - w_{ij}R_j^t \bar{p}_i - w_{ij}(T_j^t - q_*^t + R_j^t p_*)\|^2 \end{aligned} \quad (11)$$

Let $\tau = T_j^t - q_*^t + R_j^t p_*$, Eq. (11) yields:

$$\begin{aligned} E_j^t &= \sum_{i=1}^{|V|} \|\bar{q}_i^t - w_{ij}R_j^t \bar{p}_i - w_{ij}\tau\|^2 = \sum_{i=1}^{|V|} \|\bar{q}_i^t - w_{ij}R_j^t \bar{p}_i\|^2 \\ &\quad - 2\tau \sum_{i=1}^{|V|} (w_{ij}\bar{q}_i^t - w_{ij}^2 R_j^t \bar{p}_i) + \sum_{i=1}^{|V|} \|w_{ij}\tau\|^2 \end{aligned} \quad (12)$$

From Eq. (8), we have $\sum_{i=1}^{|V|} w_{ij}\bar{q}_i^t = 0$ and $\sum_{i=1}^{|V|} w_{ij}^2 R_j^t \bar{p}_i = 0$. Simplifying equation (12) yields:

$$E_j^t \rightarrow \min \Leftrightarrow \begin{cases} \sum_{i=1}^{|V|} \|\bar{q}_i^t - w_{ij}R_j^t \bar{p}_i\|^2 \rightarrow \min & (13) \\ \sum_{i=1}^{|V|} \|w_{ij}\tau\|^2 \rightarrow \min \Leftrightarrow \tau = 0 & (14) \end{cases}$$

We can minimize E_j^t by first solving the optimum rotation R_j^t in Eq. (13) without considering the translation (detailed in Appendix B). Thus, p_* and q_*^t are the optimum centers of rotations. After solving the optimum rotation matrix R_j^t , the translation T_j^t in Eq. (10) is obtained by solving Eq. (14):

$$\tau = T_j^t - q_*^t + R_j^t p_* = 0 \Leftrightarrow T_j^t = q_*^t - R_j^t p_*$$

B Finding the Bone Rotation Matrix

Substituting $P = [w_{1j}\bar{p}_1^t \dots w_{|V|j}\bar{p}_{|V|}^t]$ and $Q = [\bar{q}_1^t \dots \bar{q}_{|V|}^t]$ ($P, Q \in \mathbb{R}^{3 \times |V|}$) to the objective function in Eq. (13) yields:

$$\begin{aligned} \sum_{i=1}^{|V|} \|\bar{q}_i^t - w_{ij}R_j^t \bar{p}_i\|^2 &= \sum_{i=1}^{|V|} \|\bar{q}_i^t - R_j^t(w_{ij}\bar{p}_i)\|^2 \\ &= \|Q - R_j^t P\|_F^2 = \text{tr}((Q - R_j^t P)^T(Q - R_j^t P)) \\ &= \text{tr}(Q^T Q) + \text{tr}(P^T R_j^T R_j^t P) - 2\text{tr}(Q^T R_j^t P) \end{aligned}$$

Where $\|\cdot\|_F$ denotes the Frobenius norm and $\text{tr}(\cdot)$ denotes the matrix trace. Since P and Q are constant matrices and $R_j^t R_j^T = I$, this minimization problem then becomes maximizing

$$\zeta = \text{tr}(Q^T R_j^t P) = \text{tr}(P Q^T R_j^t) \rightarrow \max \quad (15)$$

Substituting the SVD decomposition $P Q^T = \mu \Sigma \vartheta^T$ in Eq. (9) yields:

$$\zeta = \text{tr}(\mu \Sigma \vartheta^T R_j^t) = \text{tr}(\Sigma \vartheta^T R_j^t \mu)$$

Since ϑ , R_j^t , and μ are orthogonal matrices, $\vartheta^T R_j^t \mu$ is an orthogonal matrix as well. In addition, since Σ is a diagonal matrix, we have $\text{tr}(\Sigma \vartheta^T R_j^t \mu) \leq \text{tr}(\Sigma)$. Thus, the rotation in Eq. (10) is obtained by solving:

$$\zeta \rightarrow \max \Leftrightarrow \vartheta^T R_j^t \mu = I \Leftrightarrow R_j^t = \vartheta \mu^T$$