

Transplanting and Editing Animations on Skinned Meshes

Yuntao Jia
yjia3@uiuc.edu

Wei-wen Feng
wfeng2@uiuc.edu

Yizhou Yu
yyz@uiuc.edu

Department of Computer Science
University of Illinois at Urbana-Champaign

Abstract

Skinned Mesh Animation (SMA) well approximates a mesh animation with extracted bones and their transformations. However, unlike skeleton, bones in SMA are not organized in hierarchies, thus they need mesh dependent translation vectors which prevent other sources of motion (i.e. skeletal animations, MoCAP, SMAs etc) from being applied to the skinned mesh. In this paper, we propose a new and fast method to transplant motion to skinned meshes. By efficiently solving a linear least-squares system, we can compute new translation vectors which enable the motion to work on the skinned mesh. Based on the same idea, we have also devised a SMA editing tool which allows users to edit frames of the SMA interactively. Furthermore, the editing can be propagated to all subsequent frames.

1. Introduction

Skeletal animation is one of the most important problems in computer graphics and it has received a great deal of attention from the animation community. It is the standard way to do large scale character animations. As a result, it is commonly used in computer game programming and the movie industry. The strength of skeletal animation is that once the skeleton is available, the animation can be easily manipulated and many nice approaches for skinning can be applied. Common skinning approaches include SSD, smooth skinning, linear blend skinning [7], enveloping [12], example based methods [5, 11, 4, 9] and so on. A good summary of research in this area can be found in [5, 9].

James and Twigg extended approaches for skinning characters to the general setting of skinning mesh animations (SMA)[2]. With an input mesh animation, their method can automatically extract a fixed number of bones from these meshes, as well as the transformations of these bones in each frame. Each mesh vertex is weighted by several bones. With these weights and the bones' per-frame transformations, a skinned mesh animation can be constructed, which

well approximates the original mesh animation.

Different from the hierarchical bone structure of skeletal characters, in skinned meshes, there is no structure defined among the extracted bones. As a result each bone needs its own rotation matrix and translation vector. The translation vectors make SMA mesh dependent. Thus, we can neither apply other sources of motion to the skinned mesh nor edit the SMA [1]. This limits the use of the SMA technique. Within our knowledge, there has been no research addressing this problem yet.

In this paper, we propose a new and fast method to transplant different sources of motion to skinned meshes. This is accomplished in three steps. First, the rotation matrices of the bones are extracted from the motion. Second, the rest pose of the motion is aligned with that of the skinned mesh's animation, which makes the rotation matrices applicable to the skinned mesh. Finally, new bone translation vectors are computed by efficiently solving a linear least-squares problem. Based on the same idea, we also implemented a SMA editing tool which allows users to edit the poses in frames of the SMA interactively, and the editing can be propagated to all subsequent frames.

The rest of the paper is organized as follows: in Section 2, we will explain how to transplant motion to skinned meshes, then we will show how to apply the same idea to SMA editing in Section 3, results and discussion are given in Section 4 and the conclusions are in Section 5.

2. Transplant motion to skinned meshes

2.1. Problem definition

Suppose we are having two different animations A_a and A_b , where A_b is a skinned mesh animation (SMA) defined on skinned mesh M_b . If we want to apply the motion of A_a to M_b , we call the problem transplanting motion to skinned meshes. Here, A_a can be any kind of animation, as long as it has bones and transformations defined on the bones. For instance, A_a can be a skeletal animation, MoCAP, SMA and

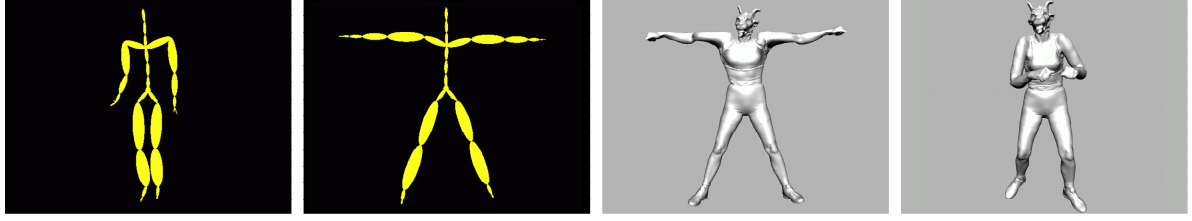


Figure 2. Aligning rest poses to the same (approximately) bind pose.

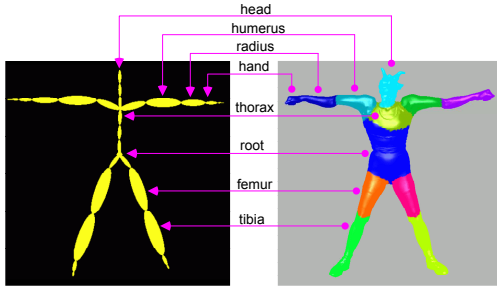


Figure 1. Bones mapping.

so on. For simplicity, in the rest of the paper, we will illustrate our method with A_a defined as a skeletal animation, and the method can be extended to other cases similarly.

Suppose the bone set of M_a and M_b are B_a and B_b respectively. We assume that B_a and B_b are similar to each other, which is a reasonable assumption for transplanting motion. Thus, we can manually build a bone mapping from B_b to B_a . The mapping does not need to be a one-to-one mapping, as long as the mapping is reasonable. One example that we are using is shown in Figure 1. The left skeleton, which has 31 bones, is from the CMU motion capture database¹. The right skinned mesh has 11 bones. An injective mapping from the skinned mesh to the skeleton can be built.

After setting up the bones mapping, for each bone B_b^i in B_b , we find its corresponding bone in B_a , suppose it is B_a^j . Then we extract the motion from B_a^j 's transformation matrices, and fit the motion to B_b^i . This is done in three steps, extracting rotations, aligning the rest poses, and solving translations using a linear least-squares method.

2.2. Rotation extraction

A transformation matrix is a four by four matrix. It contains rotation, translation and scaling of a bone in one frame of the animation. Rotation is defined in the upper left three by three block of the transformation matrix and it controls

¹<http://mocap.cs.cmu.edu/>, Carnegie Mellon University (CMU) Graphics Lab Motion Capture Database

the angle of the bone. So the motion of the animation is determined by the rotation. There are several methods to extract a rotation matrix from a general matrix. In our implementation, we used the polar decomposition method [10].

With polar decomposition, we can extract the rotation from transformation matrices of B_a , we denote them as R_a .

2.3. Rest pose alignment

The rotation matrices R_a cannot be applied to B_b directly, because they are defined relative to the rest pose of bones B_a in animation A_a . The rest pose here means the pose in the first frame of the animation. An example of a rest pose is shown in the rightmost image of Figure 2.

In order to apply the rotations R_a to B_b , we need to define an intermediate pose, through which we can align the rest poses of two animations. The intermediate pose is a configuration of bones, which both rest poses of two animations can be transformed to very easily. A good choice of an intermediate pose is the "bind pose"[6], which is just the initial configuration of the bones. Figure 2 gives an example of two animations aligned at bind poses. The first and last images are the rest poses of two animations. The second and third images are their bind poses which are approximately the same. Thus we can select the bind pose as intermediate pose and align both rest poses through it.

If there is no predetermined transformation to transform the rest pose to the intermediate pose (like the cases where "bind pose" is not defined), we can use the SMA editing technique (Section 3) to align the rest pose to the intermediate pose interactively. For more discussions on this, please see Section 3.

The equation that makes rotations R_a applicable to bones B_b with intermediate pose I can be written as:

$$R_a * R_{I \rightarrow a} * R_{b \rightarrow I} \quad (1)$$

Where $R_{b \rightarrow I}$ is the rotation that transforms a bone in B_b from the rest pose to the intermediate pose, and $R_{I \rightarrow a}$ is the inverse rotation of $R_{a \rightarrow I}$, which transforms a bone from the intermediate pose to the rest pose of B_a . After that we can apply the rotations in R_a on the bone set.

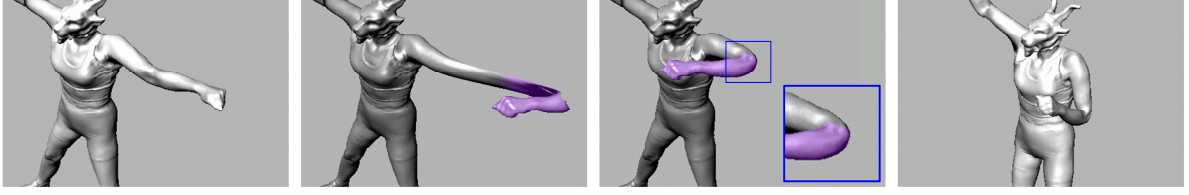


Figure 3. SMA interactive editing.

2.4. Solving translation

Once we can apply rotations R_a on B_b , we need to solve new translations to work with the rotations. For simplicity of illustration, we assume that each vertex in the skinned mesh M_b is only affected by one bone. It can be easily extended to cases where each vertex is associated with a weighted sum of multiple bones.

Suppose two vertices P_1 affected by bone B_b^1 and P_2 affected by bone B_b^2 are connected in the skinned mesh M_b , as shown in Figure 4. We define the vector between P_1 and P_2 in the rest pose as V . After applying B_b^2 's new rotation matrix R to P_2 , without translation, the vector between P_1 and P_2 becomes V_R , which is quite different from V in terms of length. Notice that rotation R is defined in the global coordinate system with respect to point O . We need to solve B_b^2 's new translation V_T that can translate V_R to V' , where V' has the same length as V and similar rotation as V_R . The equation for translation V_T is:

$$V' = V_R + V_T = RP_2 - P_1 + V_T \quad (2)$$

However, before solving the equation, V' is not known yet. A practical approximation to V' can be obtained by applying half rotation of R to V . The half rotation $R_{\frac{1}{2}}$ can be approximated by polar decomposition of matrix $\frac{R+I}{2}$ [2, 8]. I is the identity matrix, which has zero rotation. By moving all knowns to the right hand side, the equation becomes:

$$V_T = R_{\frac{1}{2}}V - RP_2 + P_1 \quad (3)$$

If at the same time B_b^1 has a new rotation R_1 which is applied on P_1 , then P_1 will become R_1P_1 , $R_{\frac{1}{2}}$ will become $(R + R_1)_{\frac{1}{2}}$ and V_T will become $V_T - V_T^1$, where V_T^1 is the translation vector of B_b^1 . For cases where each vertex is associated with a weighted sum of multiple bones, both sides of the equation will become weighted sums.

As long as two vertices are connected and their major weighted bones are different, we can build an equation for them. Suppose we have n_p pairs of vertices like this. In the end, we will obtain a large sparse linear system $\mathbb{A}x \simeq b$. \mathbb{A} has $3 * n_p$ rows and $3 * n_b$ columns where n_b is the number of bones in skinned mesh M_b and both 3 stand for three components of translation vectors. Notice that the matrix \mathbb{A}

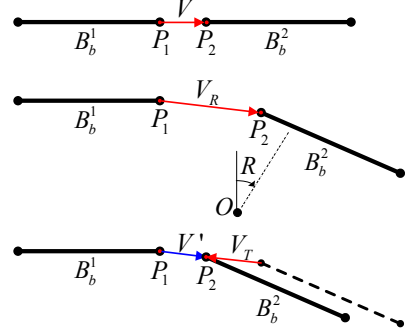


Figure 4. Solving translation.

will only depend on vertices' bone weights, and it will not change for different rotations. Thus to solve the system, we only need to obtain a QR factorization on matrix \mathbb{A} once, and solve x (translations V_T) by substitutions, which can be done very efficiently. An example of solving translations is shown in Figure 3. The second image shows that the rotation has been edited and the correct translation is computed in the third image. We can see that a good approximation of new translation vectors can be obtained by our method.

In actual problems, all bones are being rotated. To solve for the translations, we need to fix one bone and compute the translations of other bones with respect to it.

3. SMA editing

The idea of solving for translations also enables us to implement a SMA editing tool. Basically with the SMA editing tool, users can edit the transformation of a bone by applying an additional rotation to it. After that, a new translation is solved to make the editing work on the skinned mesh. The editing can also be propagated to all other frames.

An example of SMA editing is shown in the Figure 3. The first image shows the initial pose. The second image shows that users can select one bone, and apply an additional rotation to it. After that, we run the translation solver to obtain the correct translation for the rotation, as shown in the third image. The last image shows that users can obtain nice customized poses after several interactions.

This technique for SMA editing can also help transplant-

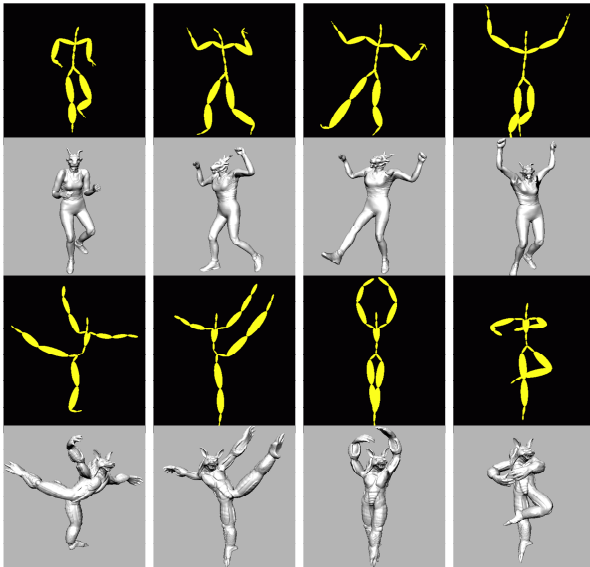


Figure 5. Results.

ing motion to skinned meshes. When there is no "bind pose" defined for either of two animations A_a and A_b , we can select the rest pose of the skeletal animation A_a as the intermediate pose, and use our SMA editing tool to align the rest pose of A_b to it interactively. This gives reasonable solutions for cases where there is insufficient information.

4. Results

The results of transplanted skinned mesh animations are shown in Figure 5. The first two rows in Figure 5 show a walking motion from the CMU motion capture database transplanted to the skinned mesh "female", which has 11 bones and 55,999 vertices. The last two rows show a ballet dancing skeletal animation transplanted to the skinned mesh "armadillo", which has 19 bones and 65,999 vertices. In both skinned meshes, each vertex is weighted by four bones and the time for solving translations per frame is less than 0.1s. Thus our SMA editing system can achieve interactive frame rates, which is very important for user interactions.

We have provided the full skinned animations of Figure 5 in our video demo, as well as two more results, including a lambada dance transplanted to the skinned mesh "female" and another dance transplanted to the skinned mesh "armadillo".

5. Conclusions

We propose a new and fast method for transplanting different motion to skinned meshes. The key problem of our work is to compute new translation vectors for the motion

based on the bone structure of the skinned mesh. This is done by solving a linear least-squares problem very efficiently. Based on the same idea, we also implemented a SMA editing tool, which allows users to edit skinned animations interactively.

With our method, users can apply various sources of motion to skinned meshes, including skeletal animations, motion capture data, SMA and so on. For example, the motion in the CMU motion capture database can be applied to skinned meshes now with our method.

When transplanting motion that are quite different from the skinned mesh's original animation, defects may occur at skeletal joints, such as "collapsing elbow" and the "candy-wrapper" effect. In future, we are considering integrating skin deformation research [13] with our work. We are also considering applying our method to other kinds of skinned animations, such as those reported by Kavan et al. [3].

References

- [1] K. G. Der, R. W. Sumner, and J. Popović. Inverse kinematics for reduced deformable models. *ACM Trans. Graph.*, 25(3):1174–1179, 2006.
- [2] D. L. James and C. D. Twigg. Skinning mesh animations. *ACM TOG*, 24(3), Aug. 2005.
- [3] L. Kavan, R. McDonnell, S. Dobbyn, J. Žára, and C. O'Sullivan. Skinning arbitrary deformations. In *I3D '07*, pages 53–60, 2007.
- [4] P. Kry, D. James, and D. Pai. Eigenskin: Real time large deformation character skinning in hardware. In *ACM SIGGRAPH Symp. on Computer Animation*, pages 153–159, 2002.
- [5] J. P. Lewis, M. Cordner, and N. Fong. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *SIGGRAPH '00*, pages 165–172, 2000.
- [6] F. Luna. *Skinned Mesh Character Animation with Direct3D 9.0c*. 2004.
- [7] N. Magnenat-Thalmann, R. Laperrière, and D. Thalmann. Joint-dependent local deformations for hand animation and object grasping. In *Graphics Interface*, pages 26–33, 1988.
- [8] M. Moakher. Means and averaging in the group of rotations. *SIAM J. Matrix Anal. Appl.*, 24(1):1–16, 2002.
- [9] A. Mohr and M. Gleicher. Building efficient, accurate character skins from examples. In *SIGGRAPH '03*, pages 562–568, 2003.
- [10] K. Shoemake and T. Duff. Matrix animation and polar decomposition. In *Proceedings of the conference on Graphics interface '92*, pages 258–264, 1992.
- [11] P.-P. J. Sloan, I. Charles F. Rose, and M. F. Cohen. Shape by example. In *I3D '01*, pages 135–143, 2001.
- [12] X. Wang and C. Phillips. Multi-weight enveloping: Least-squares approximation techniques for skin animation. In *ACM SIGGRAPH Symp. on Computer Animation*, pages 129–138, 2002.
- [13] X. S. Yang and J. J. Zhang. Realistic skeleton driven skin deformation. In *ICCSA (3)*, pages 1109–1118, 2005.