A Powell Optimization Approach for Example-Based Skinning in a Production Animation Environment

Xiao Xian [*]
Nanyang Technological University

John P. Lewis
Graphic Primitives

Seah Hock Soon
Nanyang Technological University

Nickson Fong
EggStoryCP

Tian Feng
Nanyang Technological University

Abstract

We propose a layered framework for incorporating example-based skinning algorithms such as Pose Space Deformation or Shape-by-Example into an existing character animation system. The challenge in implementing example-based skinning in an existing system lies in the fact that it is generally believed that the interpolation of the examples is best performed before doing other skinning deformations (although there has been no analysis as to why this is the case), whereas the examples are specified by the user after the other deformations are performed. It is therefore necessary to *invert* the operation of these skinning and deformation operators. Existing systems typically allow layering of both basic skinning methods such as Skeleton Subspace Deformation (SSD) and other deformations such as lattices, etc., and commercial systems may further allow additional proprietary deformation algorithms as part of the character skinning. Unfortunately, understanding and accessing their various parameters can be laborious at best, and we do not have access to the algorithms in the case of commercial packages. The contributions of this paper are 1) a detailed analysis showing how inverting the skinning operations leads to better example interpolation, and 2) a demonstration that the black-box inverse can be accomplished in practice using Powell optimization, resulting in an improved example-based skinning capability for existing large animation systems.

Keywords: skinning, Powell optimization, computer animation

1 Introduction

With the help of modelling tools or capture devices, complicated 3D character models are widely used in fields of entertainment, virtual reality, medicine etc. The range of breathtaking realistic 3D models is only limited by the creativity of artists and resolution of devices. Driving 3D models in a natural and believable manner is not trivial, especially when the model is very detailed and playback of animation becomes quite heavy and time consuming. Each time when a frame goes wrong, a production cannot afford major revisions such as re-sculpting models or re-rigging skeletons. Therefore, providing a flexible and efficient solution to animation remains an open problem.

Articulated character animation is a process of deforming the skin surface by manipulating influencing objects such as skeletons, IK, wire frames and Nurbs curves etc. Skeleton Subspace Deformation (SSD) is the predominant approach

[*]xiao0003@ntu.edu.sg

to character skinning at present. A nice review of SSD is given in [1]. SSD is widely used in games, virtual reality and other realtime applications due to its ease of implementation and low cost of computing. It provides the relation between characters and their underlying skeletons. Normally this relation is defined in the rest pose, and determines how characters move according to their skeletons thereafter. Sometimes, artists will edit the geometry of characters in the rest pose to fine-tune animations. This approach is not commonly applied, however, since editing in the rest pose will influence most other poses. On the other hand SSD is also notorious for artifacts at rotating elbows and extreme poses. For those applications that require visual fidelity, such as movies, SSD serves only as a basic framework, on which lots of more complicated deformation approaches are built as a compensation.

Example based skinning methods such as Pose Space Deformation (PSD) are candidates for correcting SSD limitations. Example geometric models paired with underlying skeletons in different poses are provided by artists with carefully sculpting and posing. PSD smoothly interpolates these meshes in pose space and produces visually attractive animations. However, although PSD may be used as a compensation to the underlying SSD, and the animator specifies the PSD examples *after* the SSD has been performed, it is generally believed that the examples are best interpolated in the rest pose, *before* the SSD has been applied. Therefore the action of the SSD and any other deformations must be "inverted" in order to push the example compensation before these operations. Besides SSD, other skinning approaches such as rigid skinning, Free Form Deformation etc. can also be applied. Our goal is to incorporate example-based skinning into a system having a variety of other skinning and deformation operations, and to be able to invert these operations regardless of their nature.

Since SSD is the most representative in the family of basic skinning, we will discuss how it performs in the inverse operation of PSD scheme. For a simplified condition where only one joint rotation and two example poses are considered, we demonstrate this inverse strategy has a better performance than the same framework without it.

The rest of this paper is organized as follows. After a review of related work, we will study details of SSD and PSD respectively in the third section. Then discussion of the inverse operation is presented in the fourth section including its implementation and reasons why this approach will improve the quality of deformation. In the fifth section, we propose a unified skinning framework by applying minimization theory with respect to other basic skinning schemes. Conclusion and some discussions of future work are presented in the last section.

2 Related Work

Besides the geometric solutions mentioned in the previous section, physical modelling and animation is another field providing realistic character simulations. Given physical principles, this category can generate more believable animation effects compared to its geometric counterpart. But they are seldom applied to interactive applications because of the high cost of computing and complicated algorithms. This paper is mainly dedicated to geometric solutions.

Pose Space Deformation [1] combines shape blending and Skeleton Subspace Deformation by formulating a scattered data interpolation problem over sculpted (or otherwise obtained) example poses. Character geometries in problematic poses will be re-sculpted by animators and then resulting displacement (referred as delta values in this paper) from the original geometries will be stored as "scattered data" for interpolation phase. The interpolation is performed in the pose space which consists of skeleton joints, or other potentially abstract controllers. Their values such as rotation degrees can be chosen as coordinates of the abstract pose space. After a model is posed and resculpted in different example poses, a multidimensional linear system is built by implementing an interpolation scheme using Radial Basis Functions (RBF), and the output of this system is the weights of each example pose. The final animation can be synthesized by linearly blending RBF functions with the solved weights.

Related research efforts have improved the speed and power of example-based skinning. [2]

incorporate linear elements into RBF to produce constant changes between examples. [3] precompute principal components of the deformation influences for individual kinematic joints instead of storing displacements for key poses, thereby enabling realtime rendering large non-linear finite element models of human hands. [4] introduce weighted pose space deformation for deforming realistic models of human hand. The latest work [5] identifies statistically relevant bones and approximates bone transforms from example mesh animations.

Using established terminology from statistical modeling, these example-based approaches can be considered as non-parametric skin deformation methods. The data needed for these methods grows with the number of examples, but arbitrary deformations can be approximated as a result. Simpler parametric skinning approaches (of which SSD is the prototype) have a fixed number of parameters; these have also seen some development in recent years [6], [7].

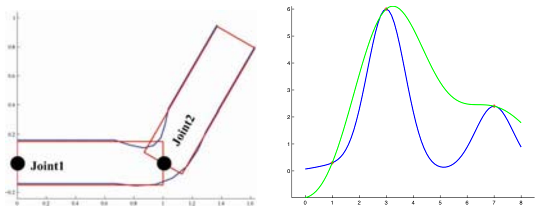
Skinning using free form lattices [8], [9] or NURBS curves [10] instead of skeletons to drive character surface are also common practices in the entertainment production. Our framework implements existing PSD theory and the distinction is that we insert an optimization module into the PSD pipeline by applying a unified inverse approach assuming the knowledge of basic skinning is unavailable.

2.1 *Our Contribution*

We provide detailed reasons why and how the inverse operation can improve the results. For a simplified case, we show that the direction of deformed vertices from inverse skinning is a linear function of joint rotation, while in the forward approach, that direction is kept as a constant. This demonstration provides for the first time a clear theoretical reason why inverse operation is required.

We formulate editing geometry in rest pose as an optimization problem and propose a unified framework which can be implemented on high-end commercial packages while allowing any proprietary skinning operators to be incorporated.

3 Skeleton Sub-Space Deformation



(a). (b).

Figure 1: *(a). Skeleton Subspace Deformation; (b). Radial Basis Interpolation*

Skeleton Subspace Deformation (SSD) is a basic algorithm that is used to define how the character surface deforms following movements of its underlying skeletons. The main idea is introduced by [11], and is also known as soft skinning, linear blending or Single Weight Enveloping (SWE). Due to its simplicity and efficiency, SSD is widely applied to interactive applications such as games and virtual reality, and it is implemented in most commercial animation packages.

A skeleton should be rigged to a character surface beforehand, roughly based on the anatomy of the character and kinetic rules. The pose in which the skeleton is rigged normally is referred to as the rest pose. The basic relationship between surfaces and skeletons is defined at the rest pose, and all motions of the character will be influenced thereafter. If SSD is adopted to define this relation, each vertex or control point of the character surface is provided with a list of joints, that will influence it, along with the weight indicating the amount of influence. When the character is animated, the position of a vertex in the animated pose is the result of weighted linear blending of its transformation by each associated joint. We formulate SSD as:

$$v_p = SSD_p(v^r) = \sum_{k=1}^{M} \omega_k T_{pk} v^r \tag{1}$$

For a vertex in rest pose v^r, its transformed position in pose p is v_p. T_{pk} means the kth joint's transformation from rest pose to pose p. Readers can find details on how to compute T_{pk} in [1]. ω_k is the corresponding weight. This weight is usually a function of distance between v^r and its associated joints, and is defined when we apply SSD to the rigged character. Figure 1 (a). is a simple illustration of SSD with only two joint frames. Rectangles represent animated sections in each of two frames and the curve shows the blended result of both frames. Since vertex transformations can be easily implemented in the graphic card, SSD is very popular in circumstances that require animating a number of characters in real time. Some opportunities for control are provided to the animators. When a character goes wrong in some pose, animators can adjust joint influence weights. But the domain of adjusting one vertex in this way is strictly limited to the linear subspace formed by the vertex as transformed by joints influencing this vertex.

3.1 Pose Space Deformation

The famous SSD problem of "collapsed elbow" is recognized in [1] as being due to the fact that deforming is limited to a linear subspace. Because of this limitation SSD cannot synthesize many parts of a character skin involving complicated joint structures. Built on the SSD scheme, the Pose Space Deformation (PSD) is proposed by [1] as a combination of SSD and shape blending providing nice solution to above mentioned problems.

PSD can be performed in the following steps:

- example pose setup: move the character to problematic poses, sculpt and store pose information (joint configuration x) including the amount of movement of each sculpted vertex (delta values d).
- In the inverse PSD approach in this paper, we transform example models (d) to the rest pose (d^r): This step is trivial if the basic skinning, say SSD, is explicit. In our proposed framework, this step will be replaced by an optimization routine.
- solving a linear system: we setup an interpolation scheme for delta $\Phi(x)\omega = d$ and solve it to obtain weights ω for all example poses.
- realtime synthesis: for an intermediate pose, we obtain the delta in the rest pose by interpolating example poses at runtime. We add this delta to the original character surface and then let SSD or any other skinning scheme finish the final transformation.

For a vertex v, if sculpted in N example poses, then there are N delta $d_i, i = 0, \ldots, N-1$ corresponding to each pose $x_i, i = 0, \ldots, N-1$. These are converted to rest pose displacements using $d_i^r = SSD^{-1}(d_i)$. We adopt Gaussian Radial Basis functions to interpolate d_i^r. First a $N * N$ matrix Φ is built with the $(i, j)th$ element as $\phi(\| x_i - x_j \|)$, where $\| x_i - x_j \|$ means the Euclidean distance between pose x_i and pose x_j, then we have a linear system:

$$W = \Phi^{-1} D^r \tag{2}$$

Here W and D^r are column vectors with ith element ω_i and d_i^r respectively. In the synthesis phase, for an intermediate pose x, we can obtain the delta d for this vertex by:

$$d = \sum_{i=0}^{N-1} \omega_i \phi(\| x_i - x \|) \tag{3}$$

For the Gaussian function $\phi(x) = e^{-\frac{x^2}{\sigma^2}}$, σ is used to control the "fall-off". In Figure 1 (b)., we use Gaussian Radial Basis functions to interpolate 3 points. The blue and green curve represent $\sigma = 1.0$ and $\sigma = 2.0$ respectively. Other basis functions also can be candidates.

Although PSD and improved example-based schemes have been discussed in many publications [2], [3], [4], the reason why the inverse should be performed is still ambiguous. In the next section we will analyze this issue and demonstrate why inverting the SSD (and other deformations) in order to interpolate the examples in the rest pose is a the right choice.

4 Inverse Operation

This section will describe the implementation of our inverse algorithm and why it is an improvement. We still study SSD as the underlying skinning, since an explicit form of basic skinning can help to simplify our task of explanation. We call the PSD scheme without the inverse operation as "forward PSD", and comparison to it will be used to demonstrate the superiority of the inverse method.

4.1 Implementation

In this sub-section, we will explain how inverse PSD works. For N examples, a vertex v is first transformed from rest pose by SSD to positions $v_i, i = 0, \ldots, N-1$, then animators move it to example positions to obtain delta values $d_i, i = 0, \ldots, N-1$. The final positions of v in example poses are $v_i + d_i, i = 0, \ldots, N-1$, and we call them target positions v_i^t. We summarize above operations as:

$$v_i^t = v_i + d_i = SSD_i(v^r) + d_i \tag{4}$$

where v^r means the rest position of v and $SSD_i(*)$ represents the equation 1. The "forward PSD" approach then concludes by interpolating d_i as a function of pose.

In the inverse approach we instead apply the inverse of $SSD_i(*)$ to v_i^t to obtain a modified rest pose vertex v_i^r. The difference of v_i^r and v^r produces new delta value d_i^r, which will be the input of linear system (equation 2) introduced in the previous section.

$$d_i^r = SSD_i^{-1}(v_i^t) - v^r \tag{5}$$

In this step we need implement the inverse skinning operator SSD^{-1}. Since SSD is a 3D transformation, SSD^{-1} simply is the inverse transformation matrix generated by SSD. For the situation where other unknown skinning operations are adopted, we propose a unified framework which will be discussed in the following section. Next we build a new delta vector D^r with ith element as d_i^r, and replace D in equation 2 with d_i^r to get a new weight vector W^r.

$$W^r = \Phi^{-1} D^r \tag{6}$$

In the synthesis phase, for an intermediate pose x we have:

$$d_x^r = \sum_{i=0}^{N-1} \omega_i^r \phi(\| x_i - x \|) \tag{7}$$

then we add this d_x^r to v^r and let SSD finish the rest of the job:

$$v_x = SSD_x(v^r + d_x^r) \tag{8}$$

v_x represents the final position of vertex v in pose x.

4.2 Deformation Direction Analysis

In this section we will analyze and compare the deforming effects of both forward PSD and inverse PSD, explaining why the inverse method is superior..

Given two examples as shown in Figure 2 (a) and (b) respectively, vertex v with the position v^r in the rest pose (0 degrees) is sculpted to a "target position" v_i^t in an example pose (90 degrees). The delta value in the first pose is zero. Then

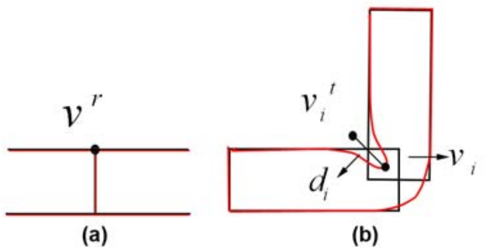


Figure 2: *(a). rest pose; (b). example pose with rotation of 90 degrees*

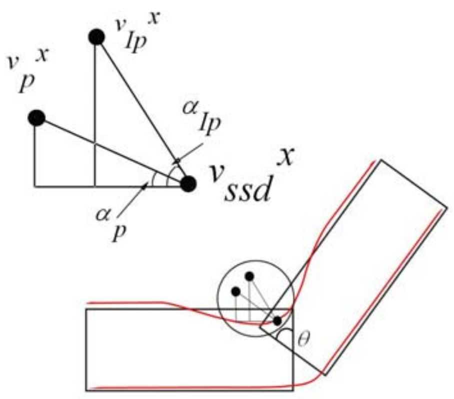


Figure 3: *deformation direction*

we apply forward and inverse PSD respectively to interpolate these two poses. For an intermediate pose x, we have two distinct deforming vertices resulting from two algorithms, as illustrated in Figure 3, $v_{ssd}^x, v_p^x, v_{Ip}^x$ are the deformed positions from SSD, forward and inverse PSD in an intermediate pose x. We use two angles α_p and α_{Ip} to analyze how directions of a deformed vertex change with the pose. In the forward case, α_p is formed by the vector (v_{ssd}^x, v_p^x) and the line $y = Y_{v_{ssd}^x}$, where $Y_{v_{ssd}^x}$ is the y coordinate of v_{ssd}^x. For two examples shown in Figure 2 (a) and (b), we have delta values $\mathbf{d}_1 = [d_{1x}, d_{1y}]$ and $\mathbf{d}_2 = [d_{2x}, d_{2y}]$. For equation 2:

$$\begin{vmatrix} \omega_{1x} & \omega_{1y} \\ \omega_{2x} & \omega_{2y} \end{vmatrix} = \begin{vmatrix} \phi_{11} & \phi_{12} \\ \phi_{21} & \phi_{22} \end{vmatrix}^{-1} \begin{vmatrix} d_{1x} & d_{1y} \\ d_{2x} & d_{2y} \end{vmatrix}$$

Because in the rest pose, we didn't generate any movement for v , then $\mathbf{d}_1 = [d_{1x}, d_{1y}] = [0, 0]$. Taking the model in rest pose as an example is a common practice when applying shape interpolation, since interpolating effects from other examples should not change the original model in rest pose. Therefore, by solving above equation we have:

$$\begin{aligned} \omega_{1x} &= \phi_{11}^{-1} d_{1x} + \phi_{12}^{-1} d_{2x} = \phi_{12}^{-1} d_{2x} \\ \omega_{2x} &= \phi_{21}^{-1} d_{1x} + \phi_{22}^{-1} d_{2x} = \phi_{22}^{-1} d_{2x} = d_{2x} \end{aligned}$$

$$\begin{aligned}\omega_{1y} &= \phi_{11}^{-1}d_{1y}+\phi_{12}^{-1}d_{2y}=\phi_{12}^{-1}d_{2y}\\ \omega_{2y} &= \phi_{21}^{-1}d_{1y}+\phi_{22}^{-1}d_{2y}=\phi_{22}^{-1}d_{2y}=d_{2y}\end{aligned}$$

where ϕ_{ij}^{-1} is the $(i,j)th$ element of Φ^{-1}, and if $i=j$, $\phi_{ij}^{-1}=1$. Then in an intermediate pose x for α_p, we have $\tan\alpha_p=\frac{d_y}{d_x}$. d_y and d_x are delta values in x,y coordinates computed from equation 3. With the above weight values, we have:

$$\begin{aligned}\tan\alpha_p=\frac{d_y}{d_x} &= \frac{\omega_{1y}\phi(x-x_1)+\omega_{2y}\phi(x-x_2)}{\omega_{1x}\phi(x-x_1)+\omega_{2x}\phi(x-x_2)}\\ &= \frac{\phi_{12}^{-1}d_{2y}\phi(x-x_1)+d_{2y}\phi(x-x_2)}{\phi_{12}^{-1}d_{2x}\phi(x-x_1)+d_{2x}\phi(x-x_2)}\\ &= \frac{d_{2y}}{d_{2x}}\end{aligned}$$

We can see this angle α_p is a constant and depends only on the value of delta in the second pose $\mathbf{d}_2=[d_{2x},d_{2y}]$. Then we take a look at α_{Ip} in the inverse case. We transform two examples to rest pose to obtain delta values: $\mathbf{d}_1^r=[d_{1x}',d_{1y}']=[0,0]$ and $\mathbf{d}_2^r=[d_{2x}',d_{2y}']$. Since only the second joint is rotating, we simplify SSD as a rotation transformation ignoring other issues such as accumulating effects from the first joint:

$$SSD_\theta=\begin{pmatrix}\cos\theta & \sin\theta & 0\\ -\sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{pmatrix}.$$

where θ is the rotation angle of the second joint. Then for the vertex $v^r=[v_{0x},v_{0y}]$, $SSD_\theta(v^r)$ transforms v from rest pose to $[v_{SSD_x},v_{SSD_y}]=[v_{0x}\cos\theta-v_{0y}\sin\theta,v_{0x}\sin\theta+v_{0y}\cos\theta]$. In an intermediate pose x, we have its corresponding rest position as $v_x^r=[v_{0x}+d_x',v_{0y}+d_y']$, and here the $[d_x',d_y']$ are interpolated result computing from equation 3. We just apply the simplified SSD to v_x^r to obtain v_{Ip}^x: $v_{Inp_x}=(v_{0x}+d_x')\cos\theta-(v_{0y}+d_y')\sin\theta$ and $v_{Inp_y}=(v_{0x}+d_x')\sin\theta+(v_{0y}+d_y')\cos\theta$. Similarly, we compute the tangent of α_{Ip}:

$$\begin{aligned}\tan\alpha_{Ip} &= -\frac{v_{Inp_y}-v_{SSD_y}}{v_{Inp_x}-v_{SSD_x}}\\ &= -\frac{d_x'\sin\theta+d_y'\cos\theta}{d_x'\cos\theta-d_y'\sin\theta}=-\tan(\beta+\theta)\end{aligned}$$

where $\tan\beta=\frac{d_y'}{d_x'}=\frac{d_{2y}'}{d_{2x}'}$. Then we can see $\alpha_{Ip}=-(\theta+\beta)$, which is linearly proportional to the pose rotation θ.

And now we take a look at a real cylinder model with one vertex sculpted in the second pose, shown in the Figure 4. Forward PSD and the corresponding inverse PSD in the same poses (30, 45 and 60 degree of one rotated joint) are illustrated respectively in Figure 5. We can see that in forward case, the direction of deformed vertex always keeps the same with the example cylinder (figure 4). For inverse PSD however, that direction is changed along with the rotation of the joint. The case described above is quite common in practice when animating shoulder, elbow, knee, hip-bone, neck, etc. All these parts would rotate from the rest pose with some angle to other poses. On the other hand, as a matter of experience, PSD is supposed to be a method as "local" correction, which means pose space should not be extended to a whole space that has to incorporate all influenced objects. Otherwise, large amount of unnecessary works of building examples will be required, and the distance between different poses is also meaningless. For example how to measure the distance between differing poses such as "lying down" and "pitching"?

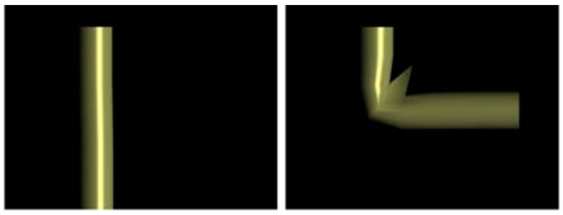


Figure 4: *A simple test case: two example poses with one vertex sculpted*

5 A Unified Framework for Inverse Skinning Model

The above discussions assume that the basic skinning algorithm is SSD, but in many circumstances, other deformation schemes will be adopted [9], [10], most of which have been implemented in most animation packages. Therefore we propose a unified framework in which no explicit inverse operation is necessitated.

Given a basic skinning method supported by animation packages we can deform the original character model from rest pose to another specific pose. In the more general case, we need to replace SSD with SKINNING in the inverse skinning algorithm as such for equation 4:

$$v_i^t=v_i+d_i=SKINNING_i(v^r)+d_i$$

But this time we don't implement the inverse of SKINNING as in equation 5. To find delta d_i^r

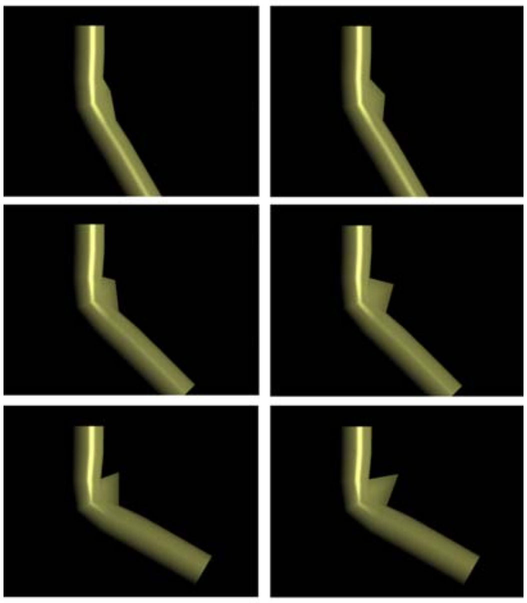


Figure 5: Interpolated Poses: *the rotated joint is at* 30, 45 *and* 60 *degree. left column: PSD; right column: inverse PSD.*

in the rest pose:

$$v_i = SKINNING_i(v^r) + d_i = SKINNING_i(v^r + d_i')$$

we can setup a minimization problem to minimize the function:

$$f(d_i') = \|v_i - SKINNING_i(v^r + d_i')\|^2 \tag{9}$$

This function can be given to Powell's method to find d_i' at the minimum of $f(d')$. For each example pose P_i, we have a d_i', then we can apply radial basis function to $d_i'(i = 0 \ldots n-1)$ in pose space to obtain $\omega_i (i = 0 \ldots n-1)$. In synthesis phase, a d_x' in an intermediate pose x can be computed by equation 3 based on its position x in pose space $d_x' = \sum_{i=0}^{n-1} \omega_i \phi(||x - x_i||)$. Then we have the final synthesis result:

$$v_x = SKINNING(v^r + d_x') \tag{10}$$

5.1 Direction Set (Powell's) Methods for Minimization

For a minimization problem, there are many candidate algorithms according to the form of function, knowledge of the derivative, computing capacity, and requirements for the rate of convergence, etc. In our situation, the function form is not explicit, and the computing burden increases with the number of example poses increases. We will adopt Powell's method as the solution to this minimization problem.

One advantage of Powell's classic method is that it does not need explicit computation of the function's gradient [12]. Because we are treating the skinning operations as a "black box", their gradient is not available, so Powell's method is suitable. Minimizing the function $f(d')$ in a particular direction is actually minimization problem of one variable, which is also called *line minimization*. Powell's idea is trying to find each minimum of function $f(d')$ in different direction until $f(d')$ stops decreasing. How to choose the next direction is the main concern of Powell's method, and it has been proved that after repeated cycles of M line minimizations on conjugate directions, the optimization will in due course converge to the minimum [12]. M is the dimensionality of the function$f()$.

5.2 System Overview

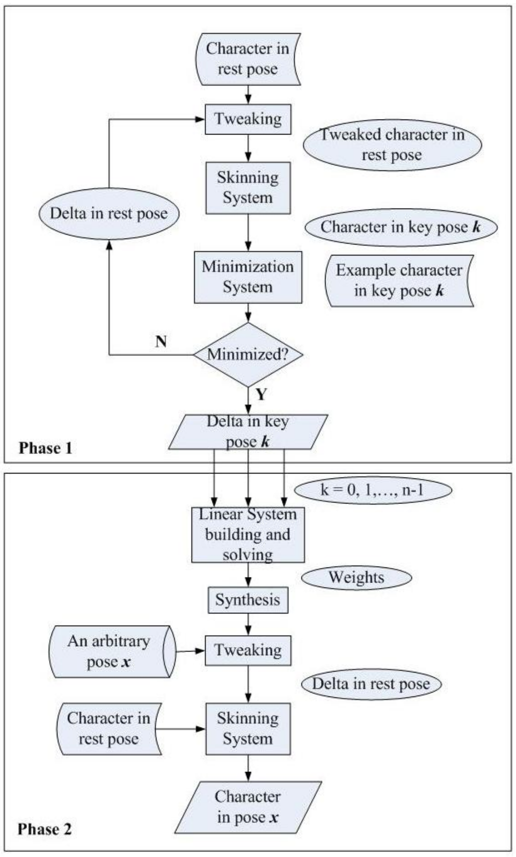


Figure 6: *System Pipeline for Unified Inverse Skinning.*

We implement this unified approach as a Maya plug-in. In Maya, "tweaking" is a procedure adding delta values to original surface vertices before any deformations. It is actually Maya's form of rest-pose editing for their built-in deformation operators. As presented in Figure 6, the whole system is divided into two phases. The first phase is to find each delta in the rest pose corresponding to each example pose. Basic skinning provided by Maya is called in the loop of minimization scheme.

The output of the first phase, the delta in the rest pose, is input to into the second phase that is a linear system performing RBF interpolation to obtain the PSD weights. In the synthesis process, for an intermediate pose x, a delta d'_x (or d^r_x) is synthesized by equation 7. The final deformed vertex is computed by Maya skinning as in equation 10.

5.3 *Singular Poses*

If the SSD transformation in equation 1 is singular, some types of inverse PSD deformation will not be possible, because any component of the desired deformation that lies in the null space of the SSD matrix will be ignored. Although singular cases are rare (one example is a joint with 180^o rotation and equal $\frac{1}{2}, \frac{1}{2}$ weights on the two joint frames, which is an unrealistic case of self-intersection), it is possible to handle these cases with a small rearrangement of the inverse PSD approach. We reformulate the problem as

$$f(y_i) = \|v_i + w_i - SKINNING_i(v^r + d'_i)\|^2 + \lambda\|w_i\|^2$$

where y_i is a concatenated vector $y_i = [d'_i, w_i]$ and λ is an arbitrary small number. The final synthesis is then

$$v_x = SKINNING(v^r + d'_x) + w_x$$

where w_x is interpolated after SKINNING by applying the same RBF scheme as used for d'_x (thus, only minimal code changed are required). The idea here is that, since w_i is being minimized, it will be generally be zero, and will be non-zero only if it is not possible to obtain the desired deformation v_i using $SKINNING_i(v^r + d'_i)$. In the case where the SSD transform is near-singular, the solved d'_i can be much large than other d'_k, which can result in poorly posed interpolation. To address this case, we further modify the objective function as

$$f(y_i) = \|v_i + w_i - SKINNING_i(v^r + d'_i)\|^2 + \lambda\|w_i\|^2 + \quad \|d'_i\|^2$$

where 0.0001 is a sufficient value for both λ and .

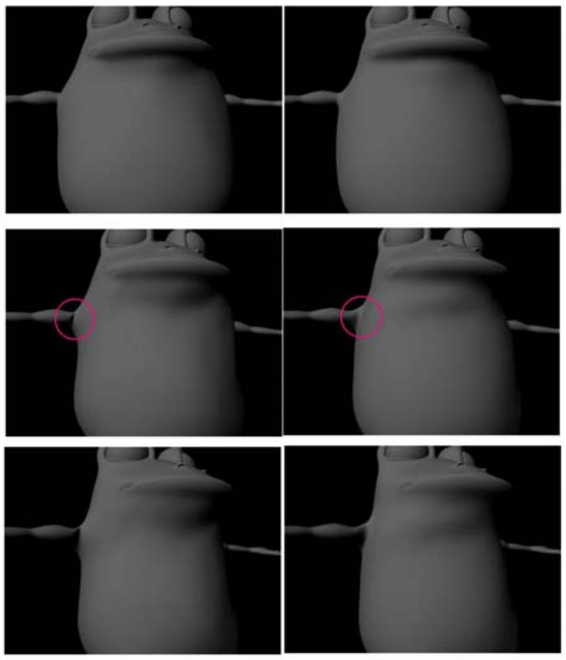


Figure 7: **toad:** *a sequence of animated frames. left: PSD; right: inverse PSD.*

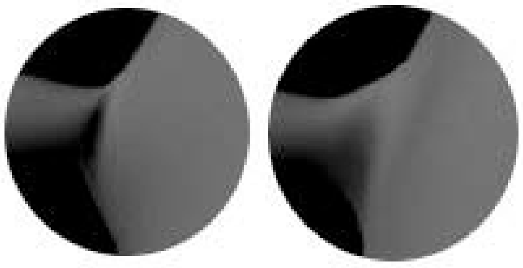


Figure 8: **toad:** *closeup of circled part from figure 9. left: PSD; right: inverse PSD.*

6 Conclusions and Discussions

Inverse skinning integrates SSD and shape interpolation more firmly than its forward rival. We demonstrate the direction of deformed vertex in inverse skinning is linearly proportional to joint rotations in a simplified example, while the forward PSD does not incorporate the direction information. Therefore the inverse approach presents better performance and more consistent interpolation (Figure 7 to Figure 10).

By formulating the inverse process as a minimization problem we propose a unified model not only for SSD but also for other skinning schemes, into which shape interpolation can be incorporated. But the minimizing process will introduce more cost. This cost depends on the size of deformed character, parameters of minimization methods (Powell) such as convergence precision, and the number of example poses. In addition the cost of the animation software must be considered (for example, the Maya API implements a run-time type interpretation system on all operations). The cost of the inverse operation is not critical, however, since it is a one time "setup" cost, and the compute time is insignificant compared to the human time required to sculpt the desired deformations. Once the linear system is solved, the synthesis is potentially realtime since no extra computing is involved in this process compared to the forward PSD. We implement this unified example-based approach as a Maya plugin. It interoperates with their closed-source "Smooth Skinning" deformation.

Acknowledgements

Authors would like to thank artists from EggStory Creative Production. We also thank several of the reviewers for comments which improved this work

References

[1] J. P. Lewis, Matt Cordner, and Nickson Fong. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *SIGGRAPH 2000: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 165–172, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[2] Peter-Pike J. Sloan, III Charles F. Rose, and Michael F. Cohen. Shape by example. In *SI3D 2001: Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 135–143, New York, NY, USA, 2001. ACM Press.

[3] Paul G. Kry, Doug L. James, and Dinesh K. Pai. Eigenskin: real time large deformation character skinning in hardware. In *SCA 2002: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 153–159, New York, NY, USA, 2002. ACM Press.

[4] Tsuneya Kurihara and Natsuki Miyata. Modeling deformable human hands from medical images. In *SCA 2004: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 355–363, New York, NY, USA, 2004. ACM Press.

[5] Doug L. James and Christopher D. Twigg. Skinning mesh animations. *ACM Trans. Graph.*, 24(3):399–407, 2005.

[6] Xiaohuan Corina Wang and Cary Phillips. Multi-weight enveloping: least-squares approximation techniques for skin animation. In *SCA 2002: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 129–138, New York, NY, USA, 2002. ACM Press.

[7] Alex Mohr and Michael Gleicher. Building efficient, accurate character skins from examples. *ACM Trans. Graph.*, 22(3):562–568, 2003.

[8] Karan Singh and Evangelos Kokkevis. Skinning characters using surface oriented free-form deformations. In *Graphics Interface*, pages 35–42, 2000.

[9] Ron MacCracken and Kenneth I. Joy. Free-form deformations with lattices of arbitrary topology. In *SIGGRAPH 96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 181–188, New York, NY, USA, 1996. ACM Press.

[10] Karan Singh and Eugene Fiume. Wires: a geometric deformation technique. In *SIGGRAPH 98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 405–414, New York, NY, USA, 1998. ACM Press.

[11] N. Magnenat-Thalmann, R. Laperrire, and D. Thalmann. Joint-dependent local deformations for hand animation and object grasping. In *Proceedings on Graphics interface '88*, pages 26–33, Toronto, Ont., Canada, Canada, 1988. Canadian Information Processing Society.

[12] William H. Press, William T. Vetterling, Saul A. Teukolsky, and Brian P. Flannery. *Numerical Recipes in C++: the art of scientific computing*. 2002.

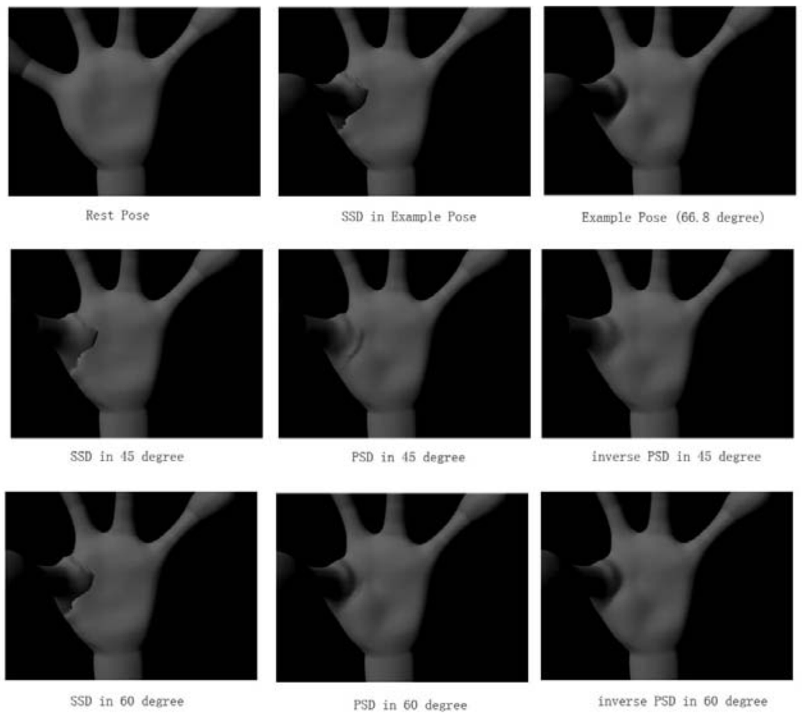
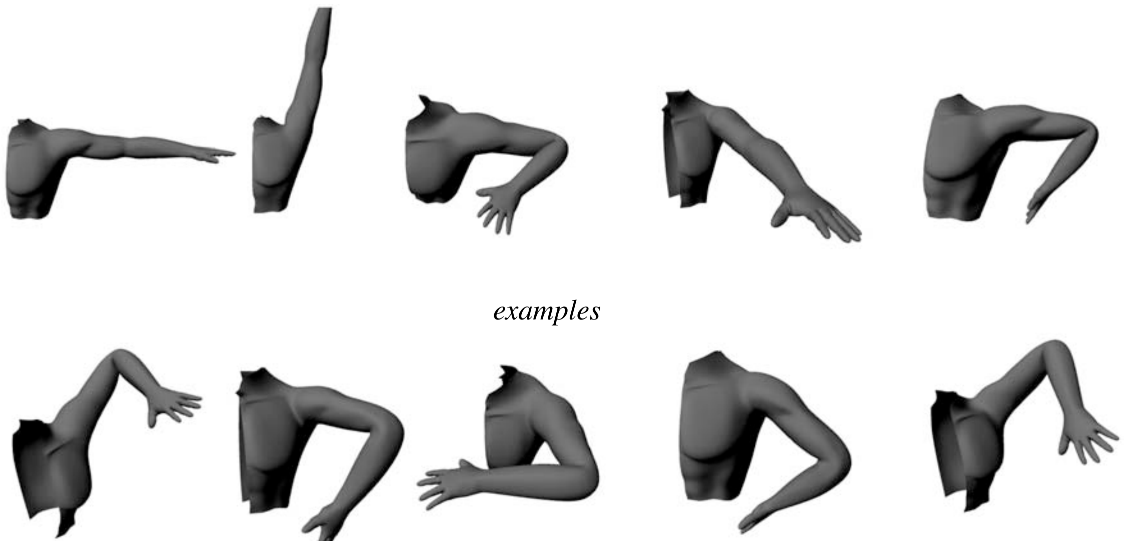


Figure 9: PSD vs. Inverse PSD: *we have two examples, one in rest pose and the other is sculpted when the thumb is rotated down* 66.8 *degrees. SSD shows an obvious collapse. The intermediate poses from SSD, PSD and inverse PSD are shown in the second (45 degrees) and third (60 degrees) row respectively.*



interpolated models computed by Powell optimization approach

Figure 10: *human arm*