Pose Space Deformation

Veronica Lee Yifan Wang

Abstract

Skeletal animation is a technique used for character animation in computer graphics, where a character is represented as a hierarchical model consisting of bones and skins enveloping the bones. Skinning methods focus on how to appropriately envelope the bones with a skin mesh. In this paper, we discuss the implementation and comparison of two example-based skinning methods: skeletal subspace deformation (SSD) and pose space deformation (PSD). We also discuss our attempt to compute joint weights using the pose samples.

1. Introduction

Skeletal animation is a technique used for character animation in computer graphics, where a character is represented as a hierarchical model consisting of bones and skins enveloping the bones. Skinning methods focus on how to appropriately envelope the bones with a skin mesh. There are three common approaches to skinning: 1) algorithmic, 2) physically-based, and 3) example-based. Purely algorithmic or physically-based skinning methods can have shortcomings because it is difficult to accurately model the complex biomechanics of skin deformation. Example-based methods require first creating a set of examples, but the results are likely to be more realistic.

Skeletal subspace deformation (SSD) [**?**] is a commonly used example-based method for skinning that uses a weighted transformation from the bind pose. Pose space deformation (PSD) [**?**] is another example-based skinning method that performs better than SSD, especially for poses at sharp angles. In this paper, we will discuss the implementation and comparison of SSD and PSD, primarily by following the implementation as described in [**?**]. We will also discuss our attempt to compute joint weights using the pose samples defined for PSD [**?**, **?**].

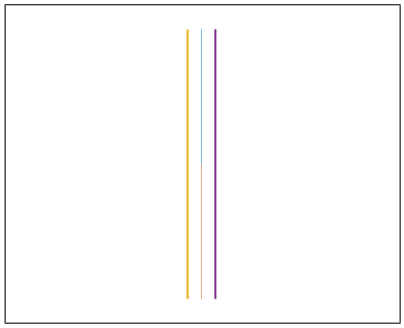

Figure 1. The bind pose for SSD.

2. Background Work

SSD is a method of linear blend skinning described by Thalmann et al [**?**] that is now used widely in many commercial software packages. This method is popular because the computations are simple and fast. However, it encounters strange behavior at sharp angles, including collapsing skin at bend elbows and rotated shoulders. We used the knowledge acquired from 6.837 lectures and homework assignment 2 to reimplement SSD in this paper.

More advanced example-based skinning methods like PSD tackle these issues by using a set of arbitrary poses defined where the strange deformations happen. The implementations in other papers [**?**, **?**] have been shown to handle complex skinning effects like muscle bulges and major wrinkles. We referenced the PSD algorithm in [**?**, **?**] to reimplement PSD in this paper.

Both SSD and PSD rely on joint weights to be defined for each vertex on the skin, in order to bind the skin mesh to the bone hierarchy. A method of computing joint weights from samples using a least-squares based estimation has been shown in [**?**, **?**]. We referenced the least-squares estimation as described in

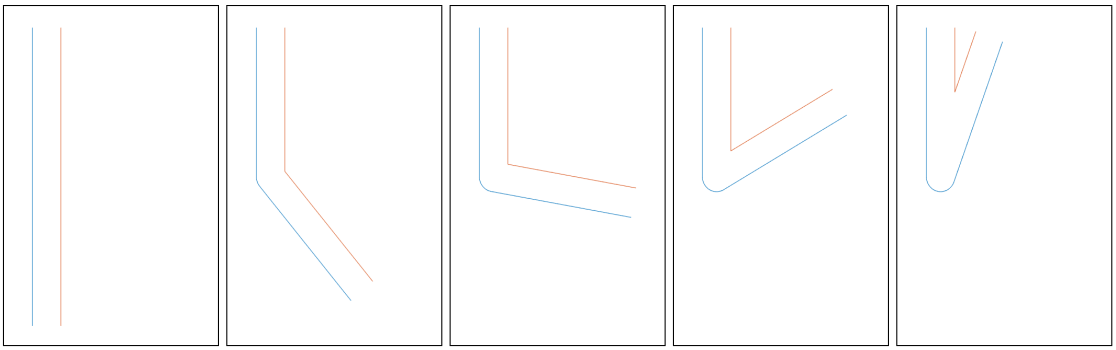

Figure 2. The five example poses defined at $\theta = 180°, 140°, 100°, 60°, 20°$.

[**?**, **?**, **?**] to reimplement the method in this paper.

3. Related Work

Weighted pose space deformation (WPSD) is an even more advanced example-based skinning method, which were derived from a hand model using medical images [**?**]. WPSD is able to generate realistic skinning even with a limited number of poses. Lewis [**?**] expands on this method by implementing parallel WPSD using fragment processors. This approach only works for poses with more than one degree of freedom, so we were not able to implement WPSD with the examples used for this paper.

4. Methodology

We used MATLAB implement in code and generate images.

For the implementations of SSD and PSD in this paper, we created a simple bone structure consisting of two bones and one degree of freedom. The resulting bind pose is shown in Figure **??**. Let θ be the angle between the two bones, so that $\theta = 180°$ in the bind pose. To test the implementations of SSD and PSD, we varied the value of θ. We kept the upper bone (the blue line in Figure **??**) fixed, while moving the lower bone (the red line Figure **??**).

Joint weights needed to be defined for both SSD and PSD. Let $w(v) = (w_u, w_l)$ be defined as the weight distribution for a vertex v on the skin, where w_u is the weight corresponding to the upper bone and w_l is the weight corresponding to the lower bone. We defined the weights to be as follows: $w(v) = (1, 0)$ if v is on the upper half of the upper bone, $w(v) = (0, 1)$ if v is on the lower half of the lower bone, $w(v)$ is a linear distribution from $(1, 0)$ to $(0.5, 0.5)$ for the vertices on the lower half of the upper bone, and $w(v)$ is a linear distribution from $(0.5, 0.5)$ to $(0, 1)$ for the vertices on the upper half of the lower bone, so that the vertex connecting the two bones has weight $(0.5, 0.5)$. This setup also ensures that $w_u + w_l = 1$ for every vertex on the skin.

5. Skeletal Subspace Deformation (SSD)

The general equation for SSD is

$$v(p_a) = S(v_0) \tag{1}$$

where p_a is an arbitrary pose, $v(p_a)$ is a vertex of a deformed target of p_a, v_0 is the corresponding vertex in the bind pose, and S is the SSD function.

Using the bind pose and joint weights as defined in the methodology, SSD computes the position of each skin vertex by using the weighted blending of an affine transformation of each joint from the bind pose. Then we can define S to be

$$S(v_0) = (\sum_{j=1}^{n_{joint}} w_j T_j) v_0 \tag{2}$$

where n_{joint} is the number of joints, w_j is the predefined joint weight, and T_j is joint j's transformation to the current vertex.

6. Pose Space Deformation (PSD)

The general equation for PSD is

$$v(p_a) = S(v_0 + D(p_a)) \tag{3}$$

where $D(p_a)$ is the displacement as a function of the arbitrary pose.

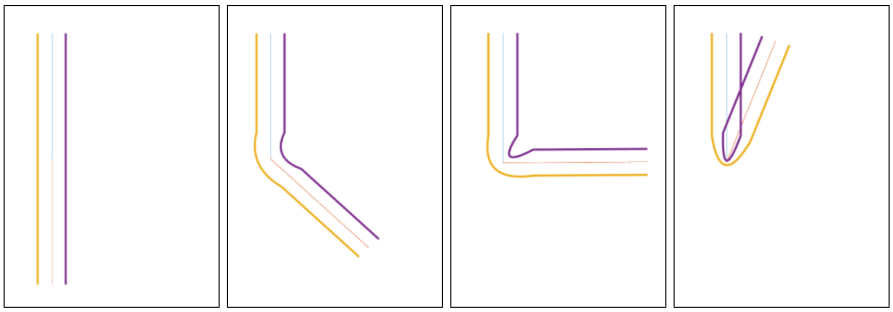

Figure 3. The results for SSD at $\theta = 180°, 135°, 90°, 20°$.

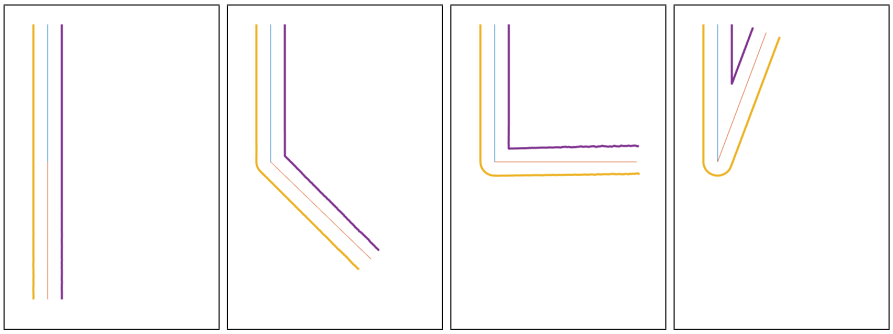

Figure 4. The results for PSD at $\theta = 180°, 135°, 90°, 20°$.

PSD requires a set of arbitrary poses to first be defined where the strange deformations happen. We defined four example poses at $\theta = 140°, 100°, 60°, 20°$. These predefined poses along with the bind pose are shown in Figure **??**. Given this set of example poses, we can interpolate to calculate $D(p_a)$ in the "pose space" [**?**].

We first need to compute the displacements of the example poses in the rest coordinates to be able to interpolate on them. We can do this by using inverse SSD on each example pose k:

$$d_k = (\sum_{j=1}^{n_{joint}} w_j T_j)^{-1} v_k - v_0 \tag{4}$$

We used the radial basis function [**?**] to interpolate the displacement for the arbitrary pose:

$$D(p_a) = \sum_{k=1}^{n_{pose}} \lambda_k \phi(\|\mathbf{x} - \mathbf{x_k}\|) \tag{5}$$

We use Gaussian radial basis function as following

$$\phi(x) = e^{-(\epsilon x)^2} \tag{6}$$

where x is the distance. In our implementation, we use the difference in degrees.

Gaussian function converges to 0 very quickly, which ensures that displacements at an angle θ will not be affected greatly by displacements at extreme angles, e.g. at 20°. At the same time, we set ϵ to be fairly small so that it would not converge to 0 too quickly. By doing so, we were able to interpolate displacements for any arbitrary angle smoothly.

In our implementation, we calculated λ_k on the fly. This is done by solving a linear system

$$D = \boldsymbol{\phi} \cdot \lambda \tag{7}$$

where displacements vector for each predefined pose D is pre-computed and $\boldsymbol{\phi}$ depends on those poses. Then displacements for any arbitrary pose can be interpolated using equation 5.

7. Computing joint weights from samples

The joint weights of each vertex are important to generate accurate skinning in SSD (equation 2). The joint weights of each vertex can be automatically calculated from the sample poses to enhance the accuracy

of the weight value. This should result in better skinning and reduces the elaborate manual work required to create weight maps.

In each sample pose p_k, we have following equation based on SSD:

$$\tilde{v_k} - e_k = (\sum_{j=1}^{n_{joint}} w_j T_j) v_0 \tag{8}$$

where $\tilde{v_k}$ is a particular vertex from skin sample k, the right hand side is the SSD deformation of vertex v_0 from the rest pose, and e_k is a displacement between the SSD deformation and $\tilde{v_k}$. If we have sufficient examples involving the same set of n_{joint} joints, we have n_{pose} equations of the form:

$$\tilde{v_k} - e_k = \sum_{j=1}^{n_{joint}} v_j w_j \tag{9}$$

where $v_j$is v_0 transformed by T_j. We can minimize the error e_k using least square method

$$\|\boldsymbol{v} - \boldsymbol{A}\boldsymbol{w}\|^2 \tag{10}$$

where $\boldsymbol{w}$ is a weight vector, $\boldsymbol{v}$ is the predefined pose vertex vector, and $\boldsymbol{A}$ is a transformed vertex matrix. Finally, weights are scaled to ensure that they sum up to 1.

8. Results

The results of SSD are shown in Figure **??**. As expected, the result at $\theta = 180°$ is perfect since that is the bind pose. However, for the other angles, it is very apparently that the skin collapses where the bones meet. At extreme angles like $\theta = 20°$, the inner skin even collapses past the bone and folds into itself.

The results of PSD are shown in Figure **??**. (A YouTube video showing the results of PSD for $15° \leq \theta \leq 190°$ is also available here). Even though some results are computed at angles that were not defined in the example poses (besides at $\theta = 180°$), the skin does not collapse, has very smooth transitions, and stays close along the bones. This shows that PSD can deal with extreme angles and our interpolation is smooth.

The results of the computed joint weights are shown in Figure **??**. The results do not look good because the least squares by subspace projection can produce

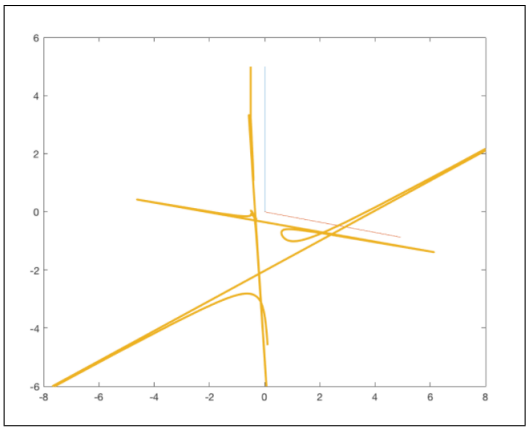

Figure 5. The results with computed joint weights using least squares estimation.

weights that are less than 0 or greater than 1, in which case the outcome will not converge.

Another disadvantage of SSD is it cannot handle rotation in 3D. The skin collapses while rotating. However, by defining poses for different rotation angles, PSD can solve this problem easily.

PSD also is not computationally expensive. In our system, coefficients for the radial basis function are computed on the fly while it only takes 20% more time than SSD. Lewis [**?**] tried to do real-time PSD for large meshed on GPU, which makes it even faster.

9. Conclusion

We were able to show the shortcomings of SSD at sharp angles and also demonstrate that, with appropriately defined example poses and appropriately chosen parameters for the radial basis function, PSD can fix those issues and produce smooth skinning.

In our simple two-bone system there was only one degree of freedom (DOF) so we could not implement WPSD [**?**]. We can expand our system to 3D and add more DOFs. Then we can try to interpolate facial expression or to skin more arbitrary poses for characters.

We could also try to implement the non-negative least square (NNLS) method instead of subspace projection, which may produce converged SSD and WPSD outcome.

PSD unifies two techniques that have been common in computer graphics. It uses a relatively simple algorithm and can handle a wide variety of deformations from a simple elbow in our example to secondary an-

imation. The setup cost of the algorithm is insignificant, and the synthesis cost is only slightly more than that of shape interpolation.

Acknowledgements

We would like to thank Professors Wojciech Matusik and Justin Solomon for their guidance throughout this project.