Real-time Character Animation Techniques – A Comprehensive Survey

CMPS 619 – Advanced Topics in Computer Science
Spring 2008, Technical Report

Jan-Phillip Tiesel
University of Louisiana at Lafayette
jpt4246@cacs.louisiana.edu

> Character animation isn't the fact that an object looks like a character or has a face or hands. Character animation is when an object moves like it is alive, when it looks like it is thinking and all of its movements are generated by its own thought process. It is the change of shape that shows that a character is thinking. It is the thinking that gives the illusion of life.
> [Lasseter 1994]

1. HISTORY AND MOTIVATION

Character animation was first recognized as an art form by cartoonist Winsor McCay in the early twentieth century. He showed that it was possible to convey emotions of an artificial character through change in shape with his hand-drawn short film "Gertie the dinosaur" which was published in 1917 [Johnston and Thomas 1981, p. 22]. About 20 years later, the art form was taken to a new level by the ground-breaking work of artists at the Walt Disney Animation Studios. Their impact on the field of character animation can hardly be overestimated.

While increasing the range of artistic expression through the exploration of unique animation principles, they were also continuously advancing the technology that brought animated characters to the screen. Some examples of their technological innovations include the multiplane camera, use of xerography for drawing transfer, or advanced composition techniques to combine live-action film and hand-drawn animation. Those improvements helped to streamline the process of animated filmmaking and led to the release of the first full-length animated film in 1937.

The fact that the visual appearance of an artificial character is often only of minor importance for its believability and range of expressions is widely recognized among artists working in animation. The basic principles of animation, developed by Disney animators in the early 1930s, can be applied to a wide range of shapes in order to create the illusion of a thinking character driving the deformation and change in shape of an artificial object. If done properly, actions and emotions of virtual characters can be easily read by a general audience almost regardless of the complexity of the animated shape. A good example is *Luxo Jr.*, the first short film released by *Pixar* (see Figure 1). Its director – John Lasseter – was one of the first "traditional" animators to show that most of the principles of animation can be applied to computer animation, as well [Lasseter 1987].

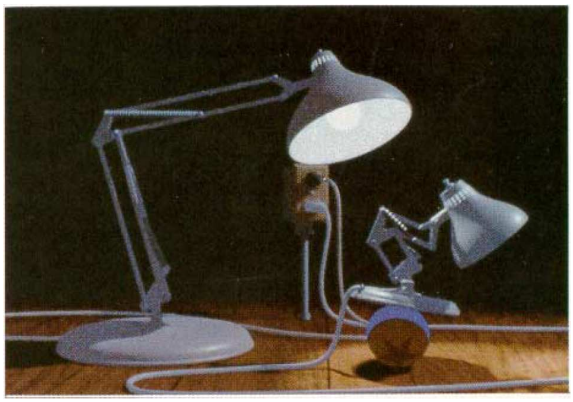

Figure 1: *Luxo Jr.*, an example of conveying emotion and personality despite the use of simple geometry. Source: [Lasseter 1987]

An extensive amount of research in psychology of perception proved a strong sensitivity of human perception to motion clues provided by living organisms. It was shown that human subjects – and even some animals – are able to spontaneously organize a sparse number of those clues into the percept of a moving figure. Johansson used point light displays showing a small number of moving markers attached to distinct features of the human body (typically joints) to demonstrate the general ability of recognizing common motion patterns despite a very reduced type of presentation [Johansson 1973].

In psychology, these types of patterns are commonly referred to as *biological motion* and it has been shown that subjects can even recognize gender, certain emotional states

or known individuals exclusively from point light displays which withhold the presentation of the person's shape from the subjects. Johansson gave an initial analytical model for describing *biological motion* and it was later shown by Troje that certain patterns can be synthesized while maintaining their perceptive quality [Troje 2002].

Both researchers and artists working in the field of computer animation have to take into account this sensitivity of human perception to a variety of organic motion patterns when trying to create convincing artificial characters. Ever after the release of the first full-length feature film entirely generated by artists using computer animation in 1995, there has been a growing demand for sophisticated computer animated characters – not only in the movie industry, but also for use in video games, virtual avatar applications, etc. Alongside, an increasing need developed for sophisticated tools and algorithms that aid computer animators in making the transfer of an imaginative motion sequence into a computational model more intuitive and efficient.

With the availability of sophisticated graphics hardware, many animation techniques that have been suggested are now applicable for real-time applications running at interactive frame rates. The artistic potential of the medium of animation paired with the ongoing technical advances will make believable virtual characters much more likely in the future. Documented applications in the field of *Virtual Reality* making use of character animation created by professional animators are rare. One exception is the *Aladdin* VR attraction created by *Walt Disney Imagineering* for one of their theme parks [Pausch et al. 1996].

This document attempts to provide an overview of the most influential and commonly used real-time character animation techniques – with an emphasis on the animation of articulated structures.

1.1 Approaches

The major goal of most surface deformation algorithms is to minimize the effort and complexity needed to represent a certain character pose while providing realistic deformation behavior for a large variety of configurations.

Even though other approaches exist and are used for certain applications, most character animation is created using key-frame systems. While the term *key-frame* was coined in the traditional animation industry, it has been generalized in the field of computer animation to apply to any variable for which certain key values are defined and any intermediate values can be computed according to some computational procedure [Parent 2002, p. 116]. The individual key-frames are typically created by an artist although this is not necessary in general – any computational model can potentially serve as a provider of key-frames to "bake" its results to a less computationally expensive key-frame discretization. The key-frames and their interpolated in-betweens are then used as input to the respective deformation algorithm.

Physical simulation models exist that provide highly realistic deformation results by employing a complex model of bones, muscles, and tendons and solving the interaction of all its elements with the deformed surface for every frame of the animation. Generally, those models are the preferred choice when rendering at interactive rates is not necessary and very realistic results are required (e.g. in the movie industry). While the computational complexity of such models is too high for most real-time animation applications, they can still indirectly benefit from the results produced by this method (see 3.4).

Procedural methods have been used for automatic generation of character animation sequences in computer games and crowd simulations, but are used primarily to blend existing motion sequences in a realistic way and are usually an extension to the underlying deformation algorithm.

2. FACIAL ANIMATION

The *blend shape face model* takes a collection of n surface meshes of identical topology (e.g. defined by vertices of a polygon mesh or control points of a parameterized surface) and linearly blends their individual attributes. A resulting vertex position $\mathbf{v}$ is calculated as

$$\mathbf{v} = \sum_{i=1}^{n} w_i \mathbf{v}_i$$

with w_i being the weight of blend shape i and $\mathbf{v}_i$ being the position of the corresponding vertex or control point in the ith blend shape. Similarly, additional attributes – such as texture coordinates or surface normal – are constructed using a linear combination of the attribute values of the respective blend shapes. Generally, the range of blended shapes is restricted to be the convex hull of the n blend shapes:

$$\sum_{i=1}^{n} w_i = 1, w_i \geq 0 \, \forall \, i$$

After the blend shapes are sculpted, semantically meaningful controls are defined that allow the animator to actuate certain features or expressions defined by one or a combination of several blend shapes (e.g. raising an eyebrow). Key-frames are created by the artist for each individual control over the course of an animated sequence. The configuration of all controls establishes the weights w_i needed to determine the deformed surface. An example of a few typical blend shapes used to constitute facial expressions is given in Figure 2.

In order to create the desired range of facial expressions for a digital character, the number of blend shapes that have to be created by an artist may increase drastically. To create the facial animation for the character *Gollum* in the feature film *Lord of the Rings: The Two Towers*, a total number of 675 blend shapes was defined [Fordham 2003]. One reason for this high number of necessary blend shape lays in one of the major limitations of the approach: the interpolated motion is not always smooth. Although the interpolation of key-frames driving the blend shape controls can be of arbitrary continuity, the interpolated motion path of a vertex is only piecewise linear. One commonly used approach to hide the linear nature of this technique is sculpting additional key shapes for specific transitions between facial expressions.

With an increasing number of blend shapes, the key-framed

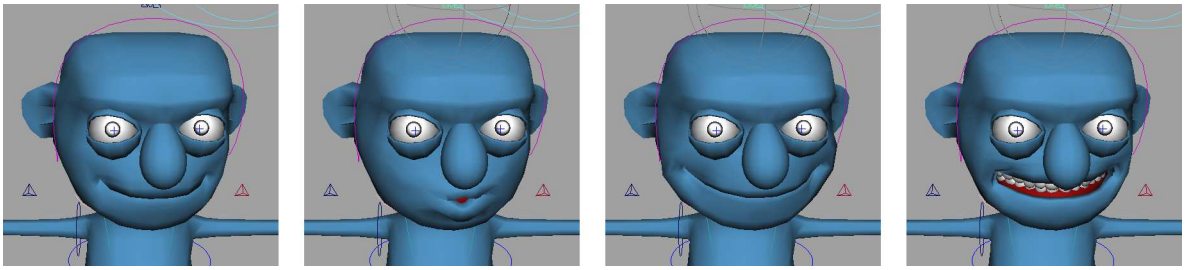

Figure 2: An example of different blend shapes for facial expressions

controls exposed to the animator that define the weights used in the blending equation also become more complex and harder to interpret semantically. Joshi et al. suggested an attempt at automating the process of defining such blend shape controls through a physically-motivated segmentation algorithm [Joshi et al. 2003].

A thorough overview over influential facial animation research and an outlook of the field can be found in [Radovan and Pretorius 2006].

3. ANIMATION OF ARTICULATED STRUCTURES

While shape blending is the preferred way to create an almost unlimited range of detailed facial expressions, the approach is infeasible for animating the whole body of a character. The growth in the number of blend shapes and controls necessary to compose all desired body poses is unacceptable both in terms of creation time and usability. For this reason, most algorithms used to deform surfaces representing a character's body are based on the assumption of an underlying articulated structure – consisting of rigid or almost rigid bones connected by joints – which drives the deformation.

3.1 Skeleton-subspace deformation

This commonly used algorithm is also referred to as *Linear Blend Skinning*, *Vertex Blending*, or *Enveloping* in the literature. For simplicity, the algorithm will be referred to as *SSD* for the remainder of this text.

3.1.1 Assumptions

SSD is based on the assumption that the deformation of surface points can be *fully* described by the pose of an underlying articulated structure. This structure is typically modeled to resemble an imaginary skeleton which drives the deformation of the surface. The full skeleton pose is then defined by a description of the hierarchical structure of its bones and the local transformations of its joints.

The second assumption that is inherent to the SSD algorithm – which is in turn one of its major drawbacks – is that the deformation of any control point or vertex on the skinned surface can be described as a linear combination of the transforms of its influencing joints. While this observation seems valid, it turns out to pose several severe restrictions on the expressiveness of the surface deformation (details in 3.1.3).

3.1.2 Algorithm

While it can operate on control points of other surface representations – such as a NURBS surface –, we will use the notion of vertices of a polygonal mesh being transformed by the SSD algorithm. Also, we assume that 4x4 matrices are used to represent transformations; different representations have been used to represent skeleton configuration (e.g. [Kavan et al. 2007]), but are not instrumental in the understanding of SSD.

SSD introduces the *rest pose* of a character's body to define the geometric information of all mesh vertices with respect to a certain skeleton configuration. Typically, the *T-pose* (a character standing on its feet with its arms extended) is used to define the rest pose and "bind" the skin to the bones of the skeleton.

The following notation will be used: the position of a particular vertex in its rest pose is denoted as $\hat{\mathbf{v}}$. The skeleton consists of b individual bones; each of the bones is defined by its constant transformation matrix in the rest pose ${}^{\text{model}}_{\text{bone}}\hat{T}_i$ and its current transform ${}^{\text{model}}_{\text{bone}}T_i$, which may vary over time. Both ${}^{\text{model}}_{\text{bone}}\hat{T}_i$ and ${}^{\text{model}}_{\text{bone}}T_i$ are found by traversing the bone hierarchy from the skeleton's root node and accumulating the individual bone transforms. Let $1 \leq i \leq b$.

The displacement of a vertex when moving rigidly with bone i is then constructed as follows: the vertex position has to be transformed from model space to bone space. This is done by employing the inverse of the rest pose matrix of the bone ${}^{\text{model}}_{\text{bone}}\hat{T}_i$:

$$\hat{\mathbf{v}}_i = {}^{\text{model}}_{\text{bone}}\hat{T}_i^{-1}\hat{\mathbf{v}} = {}^{\text{bone}}_{\text{model}}\hat{T}_i\hat{\mathbf{v}}$$

The varying transformation of bone i – describing the altered skeletal configuration – is then applied to the result to transform the vertex back into the model coordinate frame:

$$\mathbf{v}_i = {}^{\text{model}}_{\text{bone}}T_i\hat{\mathbf{v}}_i = {}^{\text{model}}_{\text{bone}}T_i\,{}^{\text{bone}}_{\text{model}}\hat{T}_i\hat{\mathbf{v}}$$

This equation is the basis for SSD; in fact, the above equation is the trivial case for the SSD algorithm, which describes the position $\mathbf{v}$ of a particular vertex as a linear combination of several different $\mathbf{v}_i$.

During the "binding" stage, an additional step is added: scalar weights for the influence of each bone on the displacement of a single vertex are calculated. This is typically done automatically based on different attributes of the vertex, such as distance to the bone. Also, a maximum number of bones influencing the position of a vertex are usually defined to minimize computational cost. A maximum number of four influential bones per vertex is often used and

seems to be a sufficient choice for many applications. The calculated weights can be edited by the artist (e.g. by painting a weight map corresponding to a single bone onto the character's surface). Techniques to automatically determine optimal bone weights according to a set of "ground truth" training examples were presented by several authors in the past [Merry et al. 2006, Mohr and Gleicher 2003, Wang and Phillips 2002].

The weighted sum of displaced the gives the blended vertex position that partially defines the deformed surface:

$$\mathbf{v} = \sum_{i=1}^{b} w_i T_i \hat{T}_i^{-1} \hat{\mathbf{v}} \quad (1)$$

where $\sum_{i=1}^{b} w_i = 1$. For bones that do not have an influence on the displacement of a particular vertex, the assigned weight would be 0. In practice, the vertex position is usually calculated as follows (assuming a maximum number of 4 influential joints):

$$\mathbf{v} = \sum_{i=1}^{4} w_i T_{b_i} \hat{T}_{b_i}^{-1} \hat{\mathbf{v}}$$

where b_i is the index of the ith bone influencing the displacement of the vertex. As $\hat{T}_i^{-1}$ and $\hat{\mathbf{v}}$ are constant, their product can be precalculated to reduce the cost of the per-vertex displacement calculation. The description of the algorithm is based on [Jacka et al. 2007].

The computation can be made more efficient by providing the transformation matrices for all bones and the per-vertex bone indices and weights to a custom vertex shader that is running on the GPU. The required per-vertex computations can easily be parallelized by modern GPUs without additional implementation effort. The varying bone transformations can be passed to the vertex shader by using texture memory for storage and shader access. The bone indices and weights are per-vertex attributes and can be assembled along with geometric information in a vertex array that provides the necessary information to the shading unit.

In order to create an animated sequence of a moving character's body, the artist would define the binding parameters as described above and continuously create key-frames that define $T_i(t)$ – the transformation matrix of bone i with respect to the model (body) coordinate frame at time t.

3.1.3 Problems and limitations of SSD

Despite its straight-forward implementation and good real-time performance, there are several shortcomings and limitations that may have a negative impact on the quality of the resulting deformation. The most significant artifact shows when joints are rotated to extreme angles: the deformed surface "suffers" a loss of volume that results in very unnatural looking deformations. This problem is referred to as *collapsing joints* for extreme bending angles and called *candy-wrapper effect* for twisting operations (see Figure 3).

These drawbacks are due to the linear nature of the algorithm: the linear interpolation of the transformation matrices is not equivalent to the linear interpolation of their rotations. The interpolated transformation matrix becomes

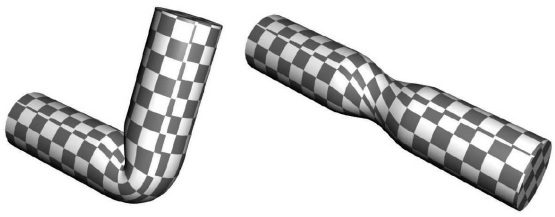

Figure 3: Collapsing joints (left) and candy-wrapper effect (right) showing loss of volume. Source: [Jacka et al. 2007]

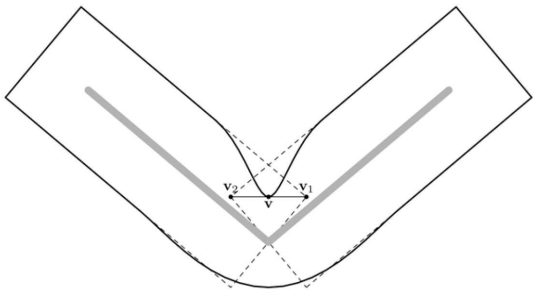

Figure 4: The resulting vertex is a linear interpolation of two rigidly displaced vertices. Source: [Merry et al. 2006]

degenerate at extreme angles and collapses the skin geometry [Mohr and Gleicher 2003]. This relationship can easyly be observed in Figure 4.

Inherently, the deformations that the algorithm produces are linear combinations of the bone transformations. Therefore, typical skin deformation effects – such as muscle bulging – that are not directly proportional to bone rotation, cannot be modeled using SSD.

Despite its inherent failure to produce deformations of consistent quality, the algorithm is supported by most commercial animation software and remains popular in video games and virtual environments – mainly due to its superior real-time performance. Numerous techniques have been suggested to overcome the visual artifacts of SSD. Some of the most common approaches are reviewed in the following sections.

3.2 Augmenting the SSD model

In order to cope with the volume loss problems apparent in the SSD algorithm, [Wang and Phillips 2002] expand the weighted blending of bone transformations (Equation (1)) applied to a vertex by replacing the single scalar value w_i. Instead, they introduce a weight matrix which is a combination of the bone transformation matrix and its inverse rest pose matrix – substituted here as $M_i = T_i \hat{T}_i^{-1}$ – and an increased number of weights:

$$\mathbf{v} = \sum_{i=1}^{b} w_i T_i \hat{T}_i^{-1} \hat{\mathbf{v}} = \sum_{i=1}^{b} w_i M_i \hat{\mathbf{v}}$$

$$\mathbf{v} = \sum_{i=1}^{b} \begin{pmatrix} w_{i,11}m_{i,11} & w_{i,12}m_{i,12} & w_{i,13}m_{i,13} & w_{i,14}m_{i,14} \\ w_{i,21}m_{i,21} & w_{i,22}m_{i,22} & w_{i,23}m_{i,23} & w_{i,24}m_{i,24} \\ w_{i,31}m_{i,31} & w_{i,32}m_{i,32} & w_{i,33}m_{i,33} & w_{i,34}m_{i,34} \\ 0 & 0 & 0 & \frac{1}{b} \end{pmatrix} \hat{\mathbf{v}}$$

By adding additional weights, the components of $\mathbf{v}$ can now be influenced independently by any component of a bone's transformation matrix. While this gives the animator a lot more flexibility in order to overcome volume loss artifacts, it also drastically increases the complexity of defining proper weight factors for every vertex. Instead of one weight per bone for every vertex of the mesh, now 12 weights per bone per vertex have to be determined in order to profit from the added flexibility. Wang and Phillips report that *Multi-Weight Enveloping* introduces greater geometrical error in their test cases than SSD – an observation which the authors blame on overfitting during the training process which was employed to generate the bone weights.

Animation space [Merry et al. 2006] is based on the idea of combining the vertex attributes $\hat{\mathbf{v}}$ (position) and w_i (bone weight) into a single attribute in order to incorporate the blending process into a simple matrix-vector multiplication. Combining the weight, inverse rest-pose matrix, and vertex into a single vector, we can write

$$\mathbf{v} = \sum_{i=1}^{b} T_i \mathbf{p}_i,$$

with $\mathbf{p}_i = w_i \hat{T_i}^{-1} \hat{\mathbf{v}}$. The sum can be rearranged as a matrix-vector product:

$$\mathbf{v} = (T_1 T_2 \dots T_b) \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \dots \\ \mathbf{p}_b \end{pmatrix} = T\mathbf{p}$$

This increases the number of variable weights per bone to the four components of the $\mathbf{p}_i$ vector. Using the *Animation Space* algorithm, it is also not viable anymore to determine those weights manually. Typically, a training set of example shapes and their respective skeleton poses are used to determine the weights that satisfy certain optimization constraints.

One option for the generation of realistic example shapes is to employ a deformation model that is too complex for real-time evaluation – such as a physical simulation modeling the interaction of bones, muscles, and tendons with the surrounding skin tissue. If the creation of a sufficient number of example shapes is not a feasible option, both *Multi-Weight Enveloping* and [Merry et al. 2006] are not adequate replacements of the SSD technique.

An overview and quality comparison of SSD, Animation Space and Multi-Weight Enveloping is given in [Jacka et al. 2007]. The authors conclude from their findings that Animation Space consistently outperforms the other two algorithms with respect to both objective and subjective quality measures. However, the significance of their results is debatable as they fail to provide information on the generative technique of the "ground truth" deformed surfaces that were used for objective quality comparisons.

It has been suggested to replace the 4x4 matrix representation of a concatenated transform for an individual bone coordinate frame with a dual quaternion representation [Kavan et al. 2007]. While this model is able to overcome some of the limitations of SSD and generates reliable blending of rotations, it also cannot produce realistic muscle bulging or dynamic effects that are not directly related to the rigid transformations represented by the dual quaternions.

3.3 Correctional models for SSD

Presented in this section are techniques that use SSD as the underlying deformation algorithm but employ additional methods to correct for the errors and artifacts introduced by the SSD algorithm.

[Mohr and Gleicher 2003] attempt to correct errors introduced by SSD by identifying critical deformation regions within the mesh and creating additional joints in the existing skeleton. The influences and transformations of the new joints are optimized to minimize volume loss and to create surface effects that can't be achieved with the previous joint configuration. For example, additional joints are added to create muscle bulges by scaling nearby vertices according to the rotation of an adjacent joint.

[Lewis et al. 2000] published their influential work on the *Pose Space Deformation* (PSD) technique, which lets the user interactively correct vertex positions on the deformed surface. This is repeated for poses that exhibit the typical SSD deformation artifacts. Once the user has "tweaked" the mesh deformation for several poses, the algorithm computes correctional shapes as a function of skeleton pose, blends them using a radial basis function and applies the result to the SSD deformation. Therefore, PSD can be seen as a combination of SSD and linear shape blending. [T. Rhee 2006] demonstrated how PSD can be efficiently implemented using the GPU.

[Kry et al. 2002] introduce the *EigenSkin* construct and present a complementary algorithm to SSD that has a similar approach to correcting SSD errors as PSD. However, rather than storing large displacement fields for user-defined key poses and then interpolating between them at runtime, as in PSD, *EigenSkin* converts vertex displacements into "eigendisplacements" with much smaller dimensions using principal component analysis (PCA) in order to decrease the cost for real-time shape synthesis. The eigendisplacement basis functions do not represent correctional fields for the entire surface, but are optimized for local domains learned from joint influences. Therefore, the storage requirements of the *EigenSkin* technique are much lower compared to PSD. The authors show that the algorithm greatly benefits from implementation on programmable graphics hardware.

[Sloan et al. 2001] represent deformations in an abstract space whose dimensions describe semantically meaningful properties of the desired shape (e.g. gender, age, or specifics about skeleton configuration like amount of bend at an elbow). After the artist tags example shapes with a set of adjectives (which in turn define the abstract space), the algorithm uses radial basis and linear interpolation techniques to generate new shapes on-the-fly and thereby enables the user to blend between abstract shape attributes over time.

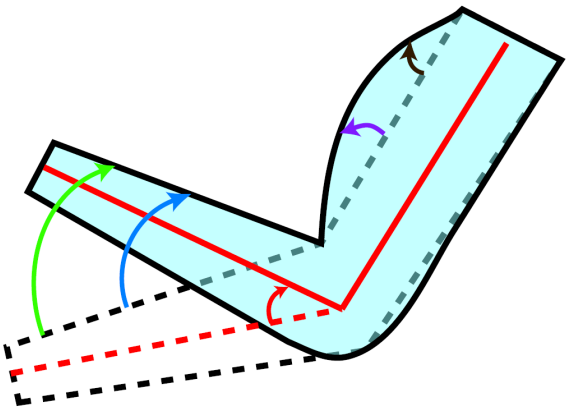

Figure 5: The illustration depicts a typical deformation that SSD cannot capture: most of the forearm rotates about the same axis and with the same amount as the bone, while parts of the bicep rotate about different axes and with different amounts. Source: [Wang et al. 2007]

3.4 Replacement for SSD in sight?

A promising approach to replace SSD with a similarly flexible but less error-prone technique was presented recently by [Wang et al. 2007]. The authors suggest to replace the SSD model with a new paradigm which still allows for the application of complementary methods such as those described in 3.3.

Their technique extracts sequences of bone rotations and triangle deformation gradients from example skeletal poses and corresponding example meshes (e.g. created using a physical simulation model). After employing regression models for rotation and scale/shear, this information is used to form a deformation gradient predictor. For real-time deformation, the deformation gradient predictions are pieced together using Poisson mesh reconstruction.

The results shown by the authors are very promising; the technique is capable of reliably avoiding SSD artifacts (even those that are still apparent in the *EigenSkin* and PSD methods). Combined with the fact that the algorithm shows good real-time performance (worst-case performance about half as fast as SSD) might render the method a potential candidate to replace the long-serving SSD algorithm.

3.5 Other approaches

One example of a more "exotic" character animation techniques is the use of artificial neural networks to learn proper deformation of a character skin from a given set of example shapes paired with their respective skeleton configuration [Guo and Wong 2004]. The deformation can be transfered to similar characters without a need for repeating the learning process. In contrast to approaches that require the generation and storage of a radial basis function per example shape (like Pose Space Deformation, EigenSkin, Shape by example), the *neuroEnveloping* algorithm takes more time and memory during the *fitting* stage with an increasing number of examples, but has constant computational cost for *synthesizing* new shapes.

Instead of trying to find a skinning algorithm that consistently creates realistic deformations for *any* type of surface material, [Forstmann et al. 2007] suggest to define reusable deformation styles for *different types* of material. Their algorithm – which allows for fast GPU evaluation – determines the deformation behavior of a surface based on radial scale textures which can be interactively created and edited by an artist. Once optimized to resemble the deformation behavior of a certain class of surface materials, the scale textures can be applied to any existing geometry.

[James and Twigg 2005] present a technique that creates mesh deformation sequences without the necessity of a predefined skeleton hierarchy. By analyzing triangle rotation sequences of the example meshes, a bone hierarchy and respective influences on the surface vertices are fitted to the character. The resulting model is capable of producing realistic deformations at very high frame rates even for scenes with several thousand animated characters. The authors also show how it can be used for integrating a character undergoing complex mesh deformations into a physical simulation.

The vast majority of research on real-time character animation techniques focuses on creating models that designed to be used by experts – mostly computer science professionals or professional computer animators. In this context, the work of [Baran and Popović 2007] is remarkable as it presents an animation framework that is targeted at non-professionals and novice users. It allows automatic embedding of an optimized skeleton into a user-defined character mesh and employs existing motion data to drive the deformation of the now animated character.

4. REFERENCES

Baran, I. and Popović, J. 2007. Automatic rigging and animation of 3d characters. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*. ACM, New York, NY, USA, 72.

Fordham, J. 2003. Middle earth strikes back. *Cinefex 92*, 71–142.

Forstmann, S., Ohya, J., Krohn-Grimberghe, A., and McDougall, R. 2007. Deformation styles for spline-based skeletal animation. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 141–150.

Guo, Z. and Wong, K. C. 2004. Neuroenveloping: A transferable character skin deformation technique. In *PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference*. IEEE Computer Society, Washington, DC, USA, 77–86.

Jacka, D., Reid, A., Merry, B., and Gain, J. 2007. A comparison of linear skinning techniques for character animation. In *AFRIGRAPH '07: Proceedings of the 5th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*. ACM, New York, NY, USA, 177–186.

James, D. L. and Twigg, C. D. 2005. Skinning mesh animations. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*. ACM, New York, NY, USA, 399–407.

Johansson, G. 1973. Visual perception of biological

motion and a model for its analysis. *Perception & Psychophysics*, 201–211.

Johnston, O. and Thomas, F. 1981. *The Illusion of Life. Disney Animation.*, 1st ed. ed. Abbeville Press, New York.

Joshi, P., Tien, W. C., Desbrun, M., and Pighin, F. 2003. Learning controls for blend shape based realistic facial animation. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 187–192.

Kavan, L., Collins, S., Žára, J., and O'Sullivan, C. 2007. Skinning with dual quaternions. In *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games*. ACM, New York, NY, USA, 39–46.

Kry, P. G., James, D. L., and Pai, D. K. 2002. Eigenskin: real time large deformation character skinning in hardware. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*. ACM, New York, NY, USA, 153–159.

Lasseter, J. 1987. Principles of traditional animation applied to 3d computer animation. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*. ACM, New York, NY, USA, 35–44.

Lasseter, J. 1994. Principles of animation: Tricks to animating characters with a computer. Course notes from SIGGRAPH 1994, "Animation Tricks". Available at `http://www.siggraph.org/education/materials/HyperGraph/animation/character_animation/principles/lasseter_s94.htm`. Accessed 26 May 2008.

Lewis, J. P., Cordner, M., and Fong, N. 2000. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 165–172.

Merry, B., Marais, P., and Gain, J. 2006. Animation space: A truly linear framework for character animation. *ACM Trans. Graph. 25,* 4, 1400–1423.

Mohr, A. and Gleicher, M. 2003. Building efficient, accurate character skins from examples. *ACM Trans. Graph. 22,* 3, 562–568.

Parent, R. 2002. *Computer Animation: Algorithms and Techniques*, 2nd ed. ed. Morgan Kaufmann Publishers, San Francisco.

Pausch, R., Snoddy, J., Taylor, R., Watson, S., and Haseltine, E. 1996. Disney's aladdin: first steps toward storytelling in virtual reality. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, New York, NY, USA, 193–203.

Radovan, M. and Pretorius, L. 2006. Facial animation in a nutshell: past, present and future. In *SAICSIT '06: Proceedings of the 2006 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries*. South African Institute for Computer Scientists and Information Technologists, Somerset West, South Africa, 71–79.

Sloan, P.-P. J., Charles F. Rose, I., and Cohen, M. F. 2001. Shape by example. In *I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*. ACM, New York, NY, USA, 135–143.

T. Rhee, J.P. Lewis, U. N. 2006. Real-time weighted pose-space deformation on the gpu. *Computer Graphics Forum. Proceedings of Eurographics 2006 25,* 3, 439–448.

Troje, N. F. 2002. Decomposing biological motion: A framework for analysis and synthesis of human gait patterns. *Journal of Vision 2,* 5, 371–387.

Wang, R. Y., Pulli, K., and Popović, J. 2007. Real-time enveloping with rotational regression. *ACM Trans. Graph. 26,* 3, 73.

Wang, X. C. and Phillips, C. 2002. Multi-weight enveloping: least-squares approximation techniques for skin animation. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*. ACM, New York, NY, USA, 129–138.