Reducing Blendshape Interference by Selected Motion Attenuation

J.P. Lewis[*], Jonathan Mooser[†], Zhigang Deng, and Ulrich Neumann
Computer Graphics and Immersive Technology Lab
University of Southern California

Fig. 1: (a) We attempt to mimic the "Jack Nicholson" expression of partially closed eyes with an arched eyebrow. First the eyelids are partially closed.

(b) The model has three controls over eyebrow shape. The desired arched eyebrow is easily obtained, but the eyelid is changed as a side effect.

(c) The model is capable of approximating the desired expression however, by readjusting the eyelid control (or, by using our technique).

Abstract

Blendshapes (linear shape interpolation models) are perhaps the most commonly employed technique in facial animation practice. A major problem in creating blendshape animation is that of blendshape interference: the adjustment of a single blendshape "slider" may degrade the effects obtained with previous slider movements, because the blendshapes have overlapping, non-orthogonal effects. Because models used in commercial practice may have 100 or more individual blendshapes, the interference problem is the subject of considerable manual effort. Modelers iteratively resculpt models to reduce interference where possible, and animators must compensate for those interference effects that remain. In this short paper we consider the blendshape interference problem from a linear algebra point of view. We find that while full orthogonality is not desirable, the goal of preserving previous adjustments to the model can be effectively approached by allowing the user to temporarily designate a set of points as representative of the previous (desired) adjustments. We then simply solve for blendshape slider values that mimic desired new movement while moving these "tagged" points as little as possible. The resulting algorithm is easy to implement and demonstrably reduces cases of blendshape interference found in existing models.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid and object modeling I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

Keywords: facial animation, user interaction

[*]zilla@computer.org
[†]jonathan@perry-street.com

1 Introduction

Blendshapes are a standard approach to computer facial animation. The technique was popularized in the pioneering character animation Tony de Peltrie [Bergeron and Lachapelle 1985], and it continues to be used in projects such as the *Stuart Little, Star Wars,* and *Lord of the Rings* movies. The technique is described by other names including morph targets and shape interpolation.

A blendshape model is simply the linear weighted sum of a number of topologically conforming shape primitives,

$$\mathbf{f}_j = \sum w_k \mathbf{b}_{kj} \tag{1}$$

where $\mathbf{f}_j$ is the jth vertex of the resulting animated model, w_k are the blending weights, and $\mathbf{b}_{kj}$ is the jth vertex of the kth blendshape. The weighted sum can be applied to the vertices of polygonal models, or to the control vertices of spline models. The weights w_k are manipulated by the animator in the form of sliders, with one slider for each weight. Weight values (slider positions) are keyframed to produce animation over time.

One of the major issues in constructing and using blendshape models is that of *blendshape interference*. The problem is seen in definition (1): the individual blend shapes $\mathbf{b}_k$ often have overlapping (competing or reinforcing) effects. For example (see Fig. 1), the animator may initially adjust the eyelid by moving one slider, but by adjusting other sliders (eyebrows, forehead, etc.) the eyelid may be raised or further lowered from its desired position. The animator then has to go back and readjust the first slider.

In practice the interference problem is minimized by sculpting the individual blendshapes to be as independent in effect as possible, and by iteratively refining these shapes when interference is found. Blendshape interference cannot be entirely eliminated, however, because desirable blendshape targets naturally have overlapping effects. For example, the blendshapes to effect a smile, raise the corner of the mouth, and produce the vowel "A" all affect the corner of the mouth region. Interference is thus considered by the animator as an expected cost of using the blendshape approach.

That cost is considerable, however. Blendshape models used in entertainment practice may take as long as a year or more to construct and can have more than 100 individual blendshapes [Jenkins 2004], resulting in thousands of potential pairwise interference effects. An animator working with a poorly designed model may

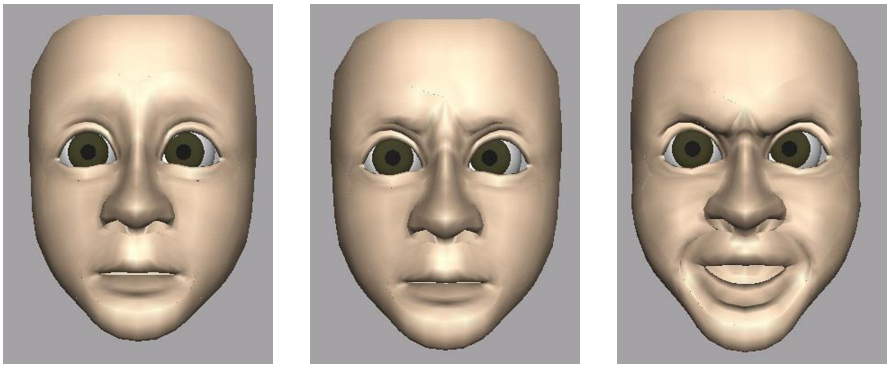

Fig. 2 a,b,c: Another example, using a well-known model [Landreth 1998]. a) Neutral pose of the blendshape model. b) The effect of moving the fourth slider, `Furrow`, to a value of 0.4. c) Further slider moves, mostly affecting the mouth, but the middle of the brow has continued to move downward as well, below the desired position.

spend much more time re-adjusting previously adjusted sliders than doing "new" animation. In the "Gollum" model used on the recent *Lord of the Rings* movies, the pairwise effects of many blendshapes were explicitly corrected in the model construction. The result was a model with a total of 946 blendshapes, the majority of which were used to correct for the behavior of an original set of 64 shapes (the correction shapes were automatically invoked, with only 64 sliders being exposed to the animator). More than 10,000 blendshapes were sculpted in the process of developing the final model [Raitt 2004].

Figure 2 shows another example of an interference problem in a professionally developed model [Landreth 1998]. Fig. 2a) is the unaltered neutral pose – all slider weights are set to zero. Fig. 2b) shows the effect of a single slider move. Presumably, an animator making such an adjustment has positioned the brow exactly as desired. But as further moves are made (Fig. 2c), the brow continues to furrow, requiring readjustment of the first slider. It should be emphasized that

- although the geometric movement being discussed is sometimes subtle, the character animator's art requires them to produce and control such subtle movement in order to distinguish geometry similar facial expressions (such as "worried" and "angry").

In this paper we show that the interference problems in a given model can be greatly reduced during animation without resculpting the model. We present a technique that allows the animator to temporarily designate portions of the model that should not be altered during a set of subsequent editing operations.

2 Related work

Despite its popularity the blendshape technique has had relatively little development since the 1980s, though it is often used as a representation in research on facial animation and tracking, e.g. [Pighin et al. 1998; Choe et al. 2001]. Prior to [Bergeron and Lachapelle 1985], Parke demonstrated cross-fading between whole-face models in his well known early work [Parke 1972]. In the 'delta' blendshape form introduced in the late 1980s (see section 3) the individual blendshapes are all offsets from a common 'neutral' face. Delta blendshapes were implemented at Pacific Data Images [Beier 2005], and the Symbolics animation system also had a sophisticated implementation [Bergeron 1987; Elson 1990]. [Kleiser 1989] described segmenting the face into separate regions (e.g. upper and lower face regions) that are blended independently, thus reducing memory usage and improving performance (though, segmenting does not by itself improve the power of the model, because a blendshape that affects a local region is no different than a blendshape with global support that is zero outside the local region).

An exception to the lack of attention to blendshapes in the research community is the recent paper [Joshi et al. 2003]. In this paper the segmentation of the face into regions is obtained automatically using a physically motivated approach. Segmenting prevents interference across segments but does not eliminate the problem, because many interference effects result from overlapping blendshapes that effect the same local region (c.f. Fig. 4). Principal component (PCA) face models derived from data [Blanz and Vetter 1999] can strictly be regarded as blendshapes, though they are outside the spirit of the term: blendshapes are understood as a representation suitable for manual animation, with the individual blendshapes having intuitive meanings such as `raise-left-eyebrow`. Typically only the first few eigenvectors of a PCA model have any intuitive interpretations, and doing keyframe animation in a PCA representation would be very difficult.

Blendshapes are a standard component of commercial animation packages such as Maya and Softimage. Although the algorithms used in these packages are unpublished, in the case of a simple technique such as blendshapes it is easy to surmise the underlying approach from the documentation, available controls, and the behavior of the controls. Maya, for example, implements the standard delta blendshape scheme, with one extension: multiple 'intermediate' blendshape targets can be situated at points along a particular slider. The system crossfades among these to produce the ultimate blendshape for that slider, which then takes part in the blendshape weighted sum.

Our solution to the interference problem resembles inverse kinematics (IK) [Yamane and Nakamura 2003; Grochow et al. 2004] in that we constrain particular points. It differs, however, in that IK *automatically* determines the pose of unconstrained parts of the character given the constraint of a *moved* part, whereas in our technique the user manually explores facial poses while the system keeps particular points stationary. On the other hand, the IK 'pin-and-drag' editing presented in [Yamane and Nakamura 2003] directly anticipates the working style needed to use our technique. Blendshapes can be considered a form of skinning; pointers to the general literature on this subject include the popular SSD or linear-blend skinning technique [Magnenat-Thalmann et al. 1988] and recent improvements [Mohr et al. 2003]. Recent example-based approaches [Kurihara and Miyata 2004] to skinning have produced excellent results, but have not yet been applied in facial animation.

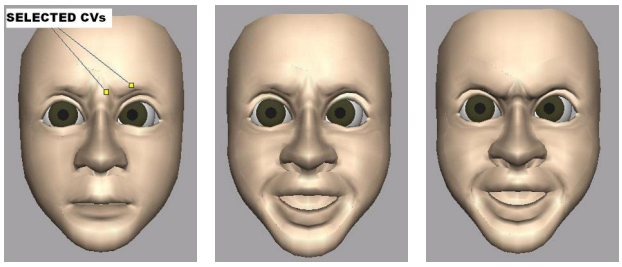

Fig. 3 a,b,c: a) The user selects two vertices whose motion is to be attenuated. The algorithm operates at the level of coordinates rather than vertices, so in effect 6 coordinates are selected. b) The results of the same slider moves shown in Fig. 3c, but with coordinate movement attenuation (compare Fig. 3c.). c) Repeated from Fig. 2c for comparison.

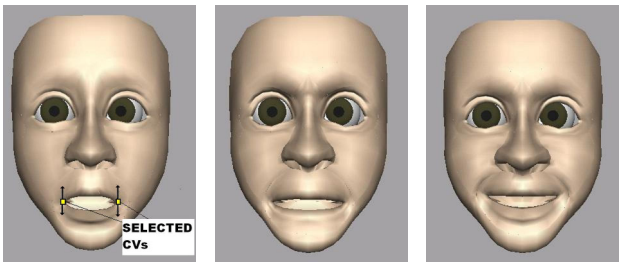

Fig. 4 a,b,c: a) The y-component of two vertices are selected. b) Sliders `SmirkLeft` and `SmirkRight` are increased to 0.7. We see their normal effects of stretching out the mouth and pushing back the cheeks, but the selected vertices maintain their vertical positions. c) The same slider moves with no attenuation. The corners of the mouth move significantly in the y-direction.

3 A Linear Algebra View of the Problem

To simplify notation, a model will be expressed as a vector of length $3n$ (for n vertices), by packing the individual vertices in the vector in some arbitrary but consistent order such as xyzxyzxyz... or xxxx...yyyy...zzzz.... The vectors $\mathbf{b}_k$ representing each blendshape are gathered in the columns of a blendshape basis matrix $\mathbf{B}$. The blendshape sum for the complete model is

$$\mathbf{f} = \mathbf{B}\mathbf{w}$$

where $\mathbf{f}$ is a $3n$ x 1 vector containing the resulting model, $\mathbf{B}$ is a $3n$ x m matrix containing m blendshapes, and $\mathbf{w}$ is the m x 1 vector of weights.

Two variants of the blendshape idea are the "whole-face" formulation and "delta" blendshapes. In the former, the vectors $\mathbf{b}_k$ represent the complete face in some pose such as a smile, or with the mouth posed to produce a particular vowel. In the delta form, the individual blendshapes are added to a "neutral" face model $\mathbf{f}_0$,

$$\mathbf{f} = \mathbf{f}_0 + \mathbf{D}\mathbf{w}$$

where columns of the delta blendshape basis $\mathbf{D}$ are simply the corresponding columns of the original basis $\mathbf{B}$ with the neutral shape subtracted: $\mathbf{d}_k = \mathbf{b}_k - \mathbf{f}_0$, and $\mathbf{D}$ has $m-1$ rather than m columns.

The whole-face formulation is preferable for the modeler because sculpting delta shapes is difficult, and it has been used to guarantee that particularly important expressions appear (by sculpting that exact expression and adding it to the blendshape basis) [Zhao 2001]. The delta form is often preferred by animators because (with appropriately sculpted blendshapes) it can allow localized control, such as blendshapes that affect only one eyebrow, raise one corner of the mouth, etc., or roughly mimic the effect of individual facial muscles [Choe et al. 2001].

Although they present different advantages to the user, the whole-face and delta formulations are identical in power because any posed blendshape model in one formulation can be represented exactly in the other form. For example to convert a delta model to the whole face form, set $\mathbf{b}_1 = \mathbf{f}_0$, and set $\mathbf{b}_k$ to $\mathbf{f}_0 + \mathbf{d}_{k-1}$ for $k = 2 \ldots m$ ($\mathbf{d}_k$ is the kth column of $\mathbf{D}$). A particular pose in the delta model represented by the weight vector $\mathbf{w}_{1\ldots m-1}$ is then represented in the whole-face model by the weights $(1 - \sum w_k, w_1, w_2, \cdots)$. In the whole-face formulation the weights should sum to one, a constraint that prevents scaling of the model. With one fewer weight, the delta form does not have this 'barycentric' constraint. Because the algorithm we present below transforms the weight vector such that its sum may change, we assume the use of the delta form throughout this paper. This does not limit the results, because the two forms are equivalent in power and are easily exchanged.

An obvious approach to reducing the interference of new slider movements on a recently obtained facial pose might be to find all rows in the blendshape matrix $\mathbf{D}$ that correspond to vertices that have recently been moved and set those rows to zero. With this change, further adjustments made to the sliders are guaranteed to not move those vertices. Such a brute force approach will generally produce poor results, however. There are cases where a single blendshape affects most of the vertices in the model, at least slightly. If we freeze the position of all those vertices, the model cannot be animated. More importantly, an underlying assumption of blendshapes is that the columns of $\mathbf{D}$ span the desired and allowable movement of the model. Zeroing rows of $\mathbf{D}$ allows facial poses that depart from the designed subspace in undesirable ways. For example, suppose slider 1 affects vertex 1, while slider 2 affects both vertex 1 and its neighbor vertex 2. If vertex 1 is frozen, the adjustment of slider 2 will move vertex 2 away from its neighbor vertex 1, disrupting the smoothness of the surface.

A more subtle approach is to enforce vector orthogonality between some of the blendshape vectors. This would allow an animator to select a set of one or more blendshapes to be "locked." Then, the effect of the remaining blendshape vectors would be altered by projecting them onto a space orthogonal to the locked vectors. This ensures orthogonality between previous changes and any future changes.

Although this avoids problems introduced by the first approach, in our experience the results are still not as desired. The main problem arises from the fact that orthogonality between high-dimensional vectors representing the entire face does not sufficiently constrain the movement of particular vertices. Further, even at a local level orthogonality is not always what is desired. Imagine a blendshape that moves a particular vertex up and to the right and another blendshape that move the same vertex up and to the left. Those two movements might be orthogonal, but arguably they disrupt each other.

4 Coordinate Motion Attenuation

The technique we present overcomes these problems by allowing the user to select a subset of coordinates to remain as stationary as possible. Those coordinates may ultimately move as the sliders are adjusted, but their movements will be relatively attenuated.

We treat this as a minimization problem. For any weight vector $\mathbf{w}_1$, we want to find a new weight vector $\mathbf{w}_2$ such that the resulting coordinate changes, $\mathbf{D}\mathbf{w}_2$, are as close as possible, in Euclidean distance, to 0 for the selected coordinates and as close as possible

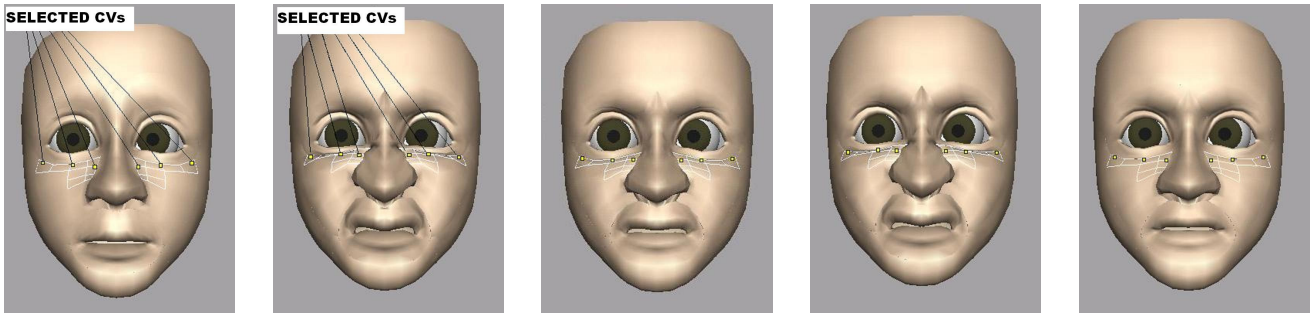

Fig. 5, a-e: a) Six vertices (18 coordinates) are selected below the eye. b) Slider moves without attenuation: `Wince`, `SneerLeft`, and `SneerRight` are all set to 0.7. The selected vertices move significantly as the face is deformed c) The same slider moves with attenuation applied. Mouth movement is similar to Fig. 5b, but the movement of the selected vertices is attenuated. α is left at its default initialization value here, $\alpha = 335$. d) The user adjusts α to 25. Now the mouth moves exactly as in Fig. 5b, but so do the selected vertices. e) $\alpha = 3000$. The algorithm favors keeping the selected vertices stationary, but at the cost of not moving unselected vertices either. The animator should reduce α.

to $\mathbf{D}\mathbf{w}_1$ for the unselected coordinates. In other words, we want to minimize

$$\begin{aligned} & \|\mathbf{S}\mathbf{w}_2\|^2 + \|\bar{\mathbf{S}}(\mathbf{w}_2 - \mathbf{w}_1)\|^2 \\ = \quad & \mathbf{w}_2^T \mathbf{S}^T \mathbf{S}\mathbf{w}_2 + (\mathbf{w}_2 - \mathbf{w}_1)^T \bar{\mathbf{S}}^T \bar{\mathbf{S}}(\mathbf{w}_2 - \mathbf{w}_1) \end{aligned}$$

where $\mathbf{S}$ is a matrix made up of the rows of $\mathbf{D}$ corresponding to the selected coordinates, and $\bar{\mathbf{S}}$ is a matrix made up of every other row of $\mathbf{D}$.

We can simplify this expression by defining

$$\begin{aligned} \mathbf{P} &= \bar{\mathbf{S}}^T \bar{\mathbf{S}} \\ \mathbf{Q} &= \mathbf{S}^T \mathbf{S} \end{aligned}$$

so the expression to be minimized becomes

$$\mathbf{w}_2^T \mathbf{Q}\mathbf{w}_2 + (\mathbf{w}_2 - \mathbf{w}_1)^T \mathbf{P}(\mathbf{w}_2 - \mathbf{w}_1)$$

Minimizing this expression strikes a balance between keeping the selected coordinates motionless and letting the other coordinates move freely. A user should be able to control the relative weight of each of these goals, so we multiply $\mathbf{Q}$ by a user controlled scaling factor, α,

$$\mathbf{w}_2^T (\alpha\mathbf{Q})\mathbf{w}_2 + (\mathbf{w}_2 - \mathbf{w}_1)^T \mathbf{P}(\mathbf{w}_2 - \mathbf{w}_1) \tag{2}$$

(α will be discussed further below).

Taking the gradient of (2) and setting to 0, we get the desired weight vector $\mathbf{w}_2$:

$$\begin{aligned} \nabla \left[\mathbf{w}_2^T (\alpha\mathbf{Q})\mathbf{w}_2 + (\mathbf{w}_2 - \mathbf{w}_1)^T \mathbf{P}(\mathbf{w}_2 - \mathbf{w}_1) \right] &= 0 \\ 2\alpha\mathbf{Q}\mathbf{w}_2 + 2\mathbf{P}\mathbf{w}_2 - 2\mathbf{P}\mathbf{w}_1 &= 0 \end{aligned}$$

$$\mathbf{w}_2 = (\mathbf{P} + \alpha\mathbf{Q})^{-1} \mathbf{P}\mathbf{w}_1 \tag{3}$$

5 Results

We implemented the weight correction (3) in Matlab, making use of a commercial animation package to perform coordinate selection interaction and render the results. Figure 3 illustrates the effect of applying the correction algorithm to the case shown in Fig. 2. Note that the mouth is sculpted as desired, but the position of the brow is undisturbed.

The algorithm operates on coordinates, allowing the animator to independently select the x, y, and/or z components of a particular vertex, thus attenuating its movement along particular axes. Figure 4a shows a pose with only two coordinates selected: the y-components of the corners of the mouth. After further slider moves, Figure 4b shows a new pose where the corners of the mouth have moved along the x-axis, but not along the y-axis. Figure 4c shows the effect of the same slider moves without coordinate attenuation.

The computations required are simple and fast enough to be performed at interactive speeds on a modern desktop machine. As such, selected motion attenuation would be relatively easy to add to an existing blendshape animation package. In addition to the usual slider interface, there would be controls to invoke motion attenuation, select vertices, and adjust the value of α. Until motion attenuation is turned off, weights would be automatically transformed as the sliders move. Using motion attenuation in conjunction with a one-level undo is a particularly simple strategy: if any slider has undesired side-effects, the animator simply undoes the slider move, applies motion attenuation to the area(s) exhibiting the interference, and releases the attenuation after adjusting the slider as desired.

The scaling factor α

The factor α controls the balance between attenuating the movement of selected coordinates and allowing the unselected coordinates to move freely. If α is very small,

$$\begin{aligned} \alpha\mathbf{Q} &\approx 0 \\ \mathbf{w}_2 &\approx (\mathbf{P} + 0)^{-1} \mathbf{P}\mathbf{w}_1 = \mathbf{w}_1 \end{aligned}$$

so $\mathbf{w}_2$ will be close to the original values $\mathbf{w}_1$, which is to say the algorithm favors allowing the unselected vertices to move. As α takes on increasingly large values, (2) will be dominated by $\mathbf{w}_2^T \mathbf{Q}\mathbf{w}_2$, which is minimized when $\mathbf{w}_2$ is in the nullspace of $\mathbf{Q}$, and therefore is in the nullspace of $\mathbf{S}$:

$$\mathbf{S}\mathbf{w}_2 = 0$$

In this case the selected vertices will not move at all. It is possible to move coordinates in $\bar{\mathbf{S}}$ without significantly displacing coordinates in $\mathbf{S}$ to the extent that $\mathbf{S}$ has small (even if not strictly zero) singular values. We see this in practice in Figs. 3 and 4.

The two terms in (2) have unequal contributions, with typically fewer selected than unselected coordinates. We compensate for this by initializing α to the ratio of the number of unselected and selected vertices.

Figure 5 shows an example of poses created using the same slider weights but different values of α. At very high values, the results are almost indistinguishable from the attenuated pose. At very low values, none of the vertices, selected or unselected, move very much.

6 Limitations and Future Work

This paper has demonstrated an interactive technique that provides animators improved control over blendshape interference. There does exists some risk that the addition of new controls would complicate the animation process rather than simplify it. In order to take advantage of this technique, an animator needs to be aware of the new controls, understand them, and make ongoing decisions about when to use them. While a similar 'pin-and-drag' style of interaction has also been proposed for inverse kinematics [Yamane and Nakamura 2003], the usefulness of our technique will ultimately need to be judged by an animator who has the opportunity to use a full implementation and test it on a real-world animation project.

The need for judgment on the animator's part is ultimately not a disadvantage; animators rely on human judgment and necessarily become intimately familiar with the models they use. In fact, the use of our technique reduces the need for animators to learn and remember the interactions between blendshapes. Reducing the need for repeated correction of blendshape interference has the potential to save valuable modeling and animation effort. The video accompanying this paper, viewable at [Video 2005], gives an impression of this potential and the interaction technique required to achieve it.

It is also worth investigating the possibility of incorporating these techniques into the modeling process. A modeler would apply constraints to the blendshape system, dictating that a given vertex should only be moved by certain sliders or that it should remain within specific bounds. How these controls might be defined and presented to a modeler is the subject of future work.

Acknowledgments

We thank Hiroki Itokazu and Bret St. Clair for model preparation, Fred Parke, Craig Reynolds, and Lance Williams for information on the early history of blendshapes, and Fred Pighin for additional discussion of these topics.

References

BEIER, T., 2005. personal communication. Hammerhead Productions.

BERGERON, P., AND LACHAPELLE, P. 1985. Controlling facial expression and body movements in the computer generated short 'Tony de Peltrie'. In *SIGGRAPH Tutorial Notes*. ACM.

BERGERON, P. 1987. 3-d character animation on the symbolics system. *SIGGRAPH Course Notes: 3-D Character Animation by Computer* (July).

BLANZ, T., AND VETTER, T. 1999. A morphable model for the synthesis of 3d faces. In *Proceedings of ACM SIGGRAPH 1999*, ACM SIGGRAPH, 187–194.

CHOE, B., LEE, H., AND KO, H.-S. 2001. Performance-driven muscle-based facial animation. *J. Visualization and Computer Animation 12*, 2, 67–79.

ELSON, M. 1990. "Displacement" facial animation techniques. *SIGGRAPH Course Notes: State of the Art in Facial Animation*.

GROCHOW, K., MARTIN, S. L., HERTZMANN, A., AND POPOVIC, Z. 2004. Style-based inverse kinematics. *ACM Trans. Graph. 23*, 3, 522–531.

JENKINS, S., 2004. personal communication. Disney Feature Animation.

JOSHI, P., TIEN, W. C., DESBRUN, M., AND PIGHIN, F. 2003. Learning controls for blend shape based realistic facial animation. *Eurographics/SIGGRAPH Symposium on Computer Animation (SCA)*.

KLEISER, J. 1989. A fast, efficient, accurate way to represent the human face. In *SIGGRAPH '89 Course Notes 22: State of the Art in Facial Animation*.

KURIHARA, T., AND MIYATA, N. 2004. Modeling deformable human hands from medical images. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM, 355–363.

LANDRETH, C., 1998. Bingo. Computer generated short film directed by Chris Landreth.

MAGNENAT-THALMANN, N., LAPERRIERE, R., AND THALMANN, D. 1988. Joint-dependent local deformations for hand animation and object grasping. In *Proc. Graphics Interface*, 26–33.

MOHR, A., TOKHEIM, L., AND GLEICHER, M. 2003. Direct manipulation of interactive character skins. In *Symposium on Interactive 3D Graphics*, ACM.

PARKE, F. 1972. Computer generated animation of faces. *Proceedings ACM annual conference.* (Aug.).

PIGHIN, F., HECKER, J., LISCHINSKI, D., SZELISKI, R., AND SALESIN, D. 1998. Synthesizing realistic facial expressions from photographs. In *Proceedings of ACM SIGGRAPH 1998*, ACM SIGGRAPH, 75–84.

RAITT, B., 2004. The making of Gollum. Presentation at U. Southern California Institute for Creative Technologies's *Frontiers of Facial Animation* Workshop, August.

VIDEO, 2005. Accompanying video for "Reducing Blendshape Interference by Selected Motion Attenuation". http://graphics.usc.edu/∼noisebrain/I3D2005.mov.

YAMANE, K., AND NAKAMURA, Y. 2003. Natural motion animation through constraining and deconstraining at will. *IEEE Tran. Visualization and Computer Graphics 9*, 3, 352–360.

ZHAO, X., 2001. personal communication. Disney Feature Animation.