# Direct-Manipulation Blendshapes

**J.P. Lewis** ■ *Weta Digital*

**Ken Anjyo** ■ *OLM Digital*

**Although direct manipulation for figure animation has long been possible using inverse kinematics, there has been no similar "inverse kinematics" approach for blendshape models. In both problems the system must infer unknown degrees of freedom during each edit. We solve this problem by minimizing changes in facial expression.**

Animators use blendshape facial models in animation and film projects such as *Stuart Little*, *Lord of Rings*, and *King Kong*. The blendshape technique is popular because it's a simple linear representation that defines a semantic model of faces (see Figure 1). Unlike generic mesh-editing tools, blendshapes approximate a space of valid facial expressions. Unlike linear models constructed using principal component analysis (PCA), the individual blendshape basis shapes have clear, interpretable meanings.

To skillfully and efficiently use a blendshape model, animators must memorize the function of 50 to 100 commonly used sliders—and locating a desired slider isn't immediate (see Figure 2). As a ballpark figure, professional 3D animation requires about one hour of labor to produce one second of animation. As in inverse kinematics for figure animation, a direct-manipulation interface could significantly accelerate this work. Although direct manipulation for figure animation has existed for years, no existing approach addresses directly manipulating blendshapes. (See the "Related Work in Facial Animation" sidebar for details on other approaches.)

Generally, interfaces should offer both direct manipulation and editing of underlying parameters. Although direct manipulation usually is more natural and efficient, parameter editing can be more exact and reproducible, so animators might prefer it in certain cases. Most professional graphics software provides both interaction modes. For example, 2D compositing packages such as Shake let animators position images by either dragging or adjusting translation values or animation curves. Three-dimensional figure animation widely uses direct manipulation by inverse kinematics. Forward kinematics is still used for some purposes, however; for example, you can most easily mimic the periodic flapping of a bird's wing by animating the appropriate joint angles. Additionally, taking the forward-kinematics or parametric viewpoint often is useful in operations such as filtering on a particular parameter's graph over time.[1,2] Parameter editing has been the only possibility for blendshapes until now.

Here, we introduce a direct-manipulation approach for blendshapes. Because a realistic blendshape model can have 100 degrees of freedom (DOF) or more, direct manipulation requires inferring many parameters for each known DOF. We regularize this inverse problem by exploiting the fact that facial-pose changes are proportional to slider position changes. The resulting approach is simple to implement, efficient (involving only linear system solvers), and compatible with existing blendshape models. Crucially, it also interoperates with the traditional parameter-based keyframe editing that current animation practices use. As we'll show, a single direct-manipulation edit is usually equivalent to a number of edits using the underlying sliders.

## Manipulators

*Manipulators* are user interface elements indicating the desired location of a particular corresponding vertex. Our implementation automatically creates a small sphere to represent each manipulator. The selected manipulator (the *active manipulator*) is the one being moved; the other manipulators serve as constraints. Because the manipulator locations are initialized from the locations of the corresponding vertices, we sometimes use the two terms interchangeably.

 Published by the IEEE Computer Society 

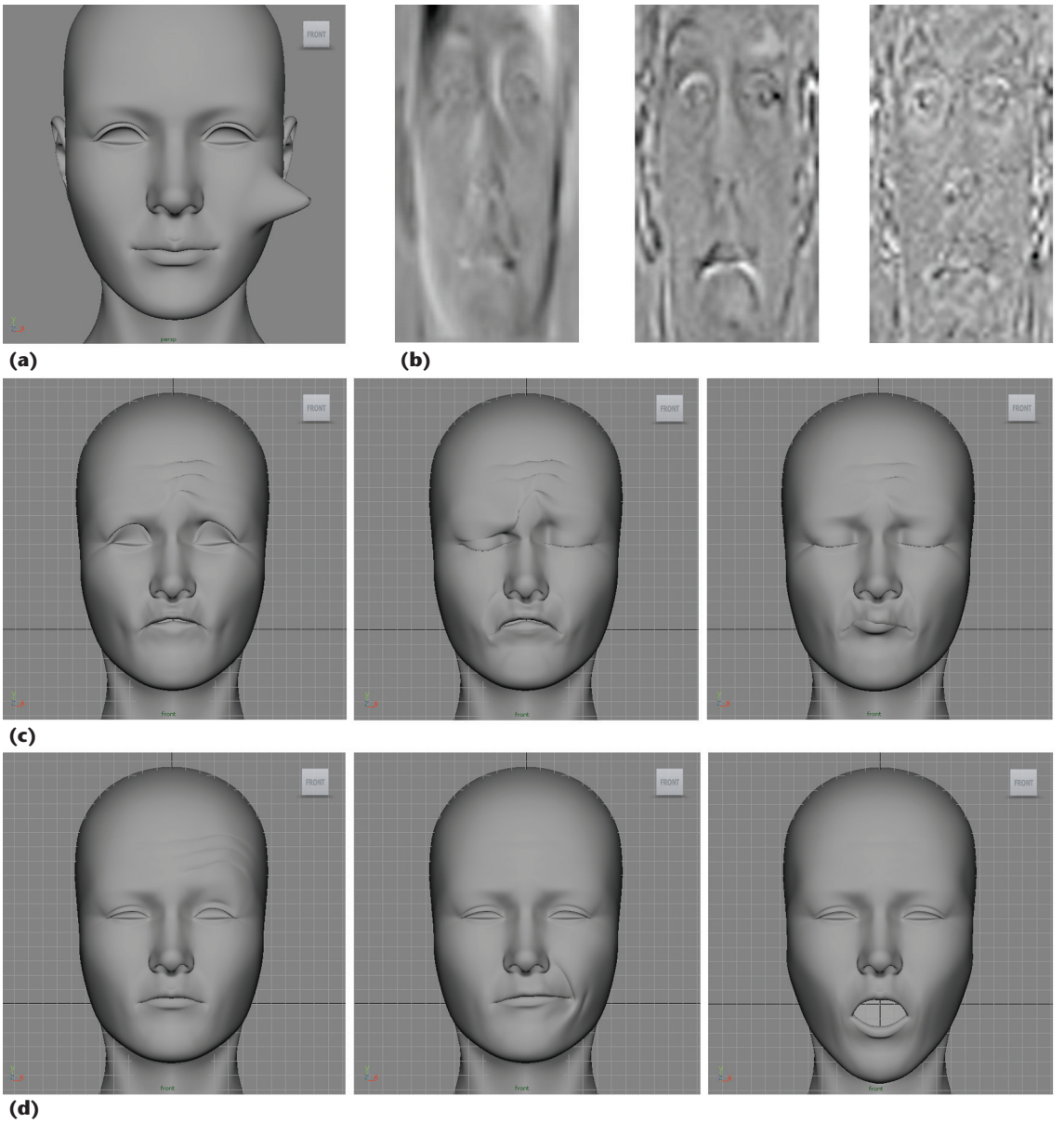



**Figure 1. Different facial-editing approaches. (a) Generic surface-editing tools aren't suitable for facial animation because they don't respect the space of valid face shapes. (b) The 10th, 50th, and 100th eigenvectors of a facial image collection. A principal component analysis (PCA) basis isn't a semantic or human-editable representation; it's difficult to know how much of these basis vectors to add to match a particular face. (c) A PCA basis generated from a professional animation is similarly uninterpretable. (d) Blendshape targets are suitable for human editing because they have clear, identifiable roles such as `RAISE-BROW`, `SMIRK`, and `JAW-OPEN` (from left to right).**

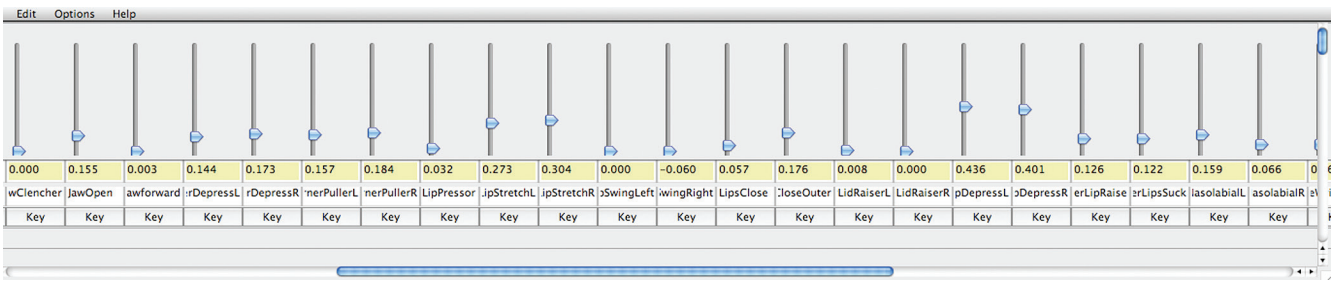



**Figure 2. A portion of the blendshape slider interface for a professionally created model (the complete set of sliders doesn't fit on the computer display). The model has more than 100 sliders.**

## Related Work in Facial Animation

The facial-animation and facial-editing literature covers various approaches, but our article discusses only a few representative facial-editing approaches. The "Blendshape Facial Animation" sidebar briefly discusses such models' definition and history; another great general facial-animation reference is *Computer Facial Animation*.[1]

One possible approach is to edit face meshes with standard geometric deformers such as free-form deformation lattices. Radial basis functions are an attractive editing approach because you can place the control points to coincide with important locations such as the mouth and eye corners.[2] Bernd Bickel and his colleagues propagated the influence of marker points across the face mesh by minimizing energies defined by the iterated Laplace-Beltrami operator on the mesh.[3] Li Zhang and his colleagues used radial basis functions to locally blend facial basis shapes.[4] Wei-Wen Feng and his colleagues used kernel canonical correlation to identify a mapping between surface points and "bones" driving the skin movement and used regression to generate the bone movement from the manipulators.[5] This system can generate plausible facial deformations from a sparse, underdetermined set of control points.

Facial animation is a broad area; several other direct-manipulation approaches for faces exist, including some that involve linear models. For example, Manfred Lau and his colleagues' approach learned a prior on coefficients of an underlying principal component analysis (PCA) model using the mixtures-of-factor-analyzers approach.[6] The direct-manipulation inverse problem can be considered in statistical terms as data imputation and can be solved with methods such as PCA imputation.[7] However, this requires an iterative solution and degrades when the basis isn't orthogonal. Other approaches have included facial manipulation through an underlying physical simulation.

Previous researchers often avoid the inverse problem by driving the face mesh with a full set of manipulators (see Figure A1).[2,3,8] This is appropriate for performance-driven facial animation, in which the positions of the full set of markers are available. When a single manipulator or marker moves, these approaches superficially resemble the direct manipulation our system offers, with a crucial difference. The equivalent in our system is to place a full set of constraining pins on the face. However, our system permits facial movement with any number of constraining pins (including none), which permits larger, less constrained edits (see Figure A2).

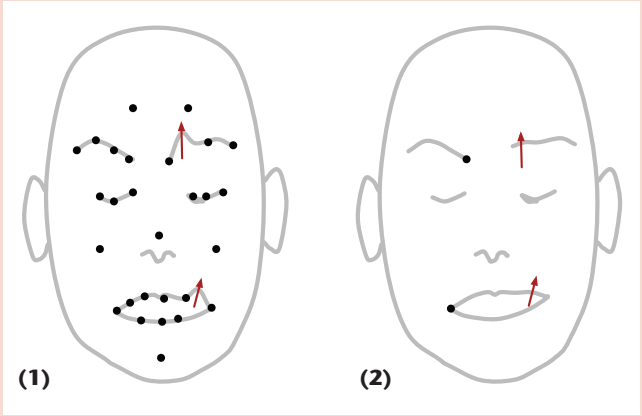



**Figure A. A comparison of performance-driven facial movement with our direct-manipulation approach: (1) Approaches that drive a face model with motion capture can be adapted for direct manipulation. However, because the model is fully constrained, you must specify the position of every vertex or motion capture marker. (2) Our approach resembles inverse kinematics: it can constrain any desired subset of vertices and automatically infer the remaining points' positions.**

The approaches closest to ours drive blendshapes with motion-capture data; for example, one such approach automatically determines localized blendshape regions.[9] Such approaches provide direct manipulation as a side effect if they let users interactively move the individual markers' positions. However, they fully constrain the facial motion, limiting the power of individual edits (see Figure A1).

Unlike previous research, our approach targets manually sculpted and semantically parameterized blendshape models common in current animation practice. It doesn't require a PCA basis or a custom model; it works with any existing blendshape model. The approach reflects direct-manipulation edits in the blendshape sliders' positions and vice versa, so

## Method

Figure 3 shows how our system (built using a commercial animation software package) operates. The animator simply selects control points or vertices and drags them to desired locations. Previously moved vertices act as constraints when subsequent vertices move.[3] Vertex configurations that the system can't reach from the blendshape subspace are approximated.

### *Directly Manipulating a Given Model*

As the animator selects and drags a particular vertex on the face model, the system must determine the other vertices' positions. Our solution acknowledges that a skilled artist created the blendshape model that inherently defines the character's desired range of movement. This leads to two observations.

First, a face model with $m$ vertices has potentially $3m$ DOF. However, if the face is a blendshape model with $n$ targets, the face movement is restricted to a subspace of dimension $n$. This is the space of valid shapes as defined by the model's creator; the direct-manipulation solution should remain in this space.

Second, the space of valid shapes is adequately

the animator can switch between direct manipulation and blendshape parameter editing as appropriate. This isn't possible with other approaches, such as those relying on an underlying PCA basis (see Figure 1 in the main article). Our approach is simple to implement and involves only a constrained linear-system solver.


**References**

1. F.I. Parke and K. Waters, *Computer Facial Animation*, AK Peters, 2008.
2. J. Noh, D. Fidaleo, and U. Neumann, "Animated Deformations with Radial Basis Functions," *Proc. ACM Symp. Virtual Reality Software and Technology* (VRST 00), ACM Press, 2000, pp. 166–174.
3. B. Bickel et al., "Multi-scale Capture of Facial Geometry and Motion," *ACM Trans. Graphics*, vol. 26, no. 3, 2007, p. 33.
4. L. Zhang et al., "Spacetime Faces: High Resolution Capture for Modeling and Animation," *ACM Trans. Graphics*, vol. 23, no. 3, 2004, pp. 548–558.
5. W.-W. Feng, B.-U. Kim, and Y. Yu, "Real-Time Data Driven Deformation Using Kernel Canonical Correlation Analysis," *ACM Trans. Graphics*, vol. 27, no. 3, 2008, article 91.
6. M. Lau et al., "Face Poser: Interactive Modeling of 3D Facial Expressions Using Facial Priors," *ACM Trans. Graphics*, vol. 29, no. 1, 2009, pp. 1–17.
7. R. Dear, *A Principle-Component Missing Data Method for Multiple Regression Models*, tech. report SP-86, Systems Development Corp., 1959.
8. B. Choe, H. Lee, and H.-S. Ko, "Performance-Driven Muscle-Based Facial Animation," *J. Visualization and Computer Animation*, vol. 12, no. 2, 2001, pp. 67–79.
9. P. Joshi et al., "Learning Controls for Blend Shape Based Realistic Facial Animation," *ACM Siggraph 2006 Courses*, ACM Press, 2006, article 17.


parameterized by the blendshape weights (sliders). In turn, this implies that Euclidean distance in slider space is an approximation of the semantic distance between face poses.

With these observations in mind, we can evaluate possible solutions to the direct-manipulation problem. One would be to require all other vertices to move as little as possible. Although this has some appeal, the animator's goal isn't to manipulate the face as little as possible—he or she would rather take large steps, provided they're predictable and controllable.

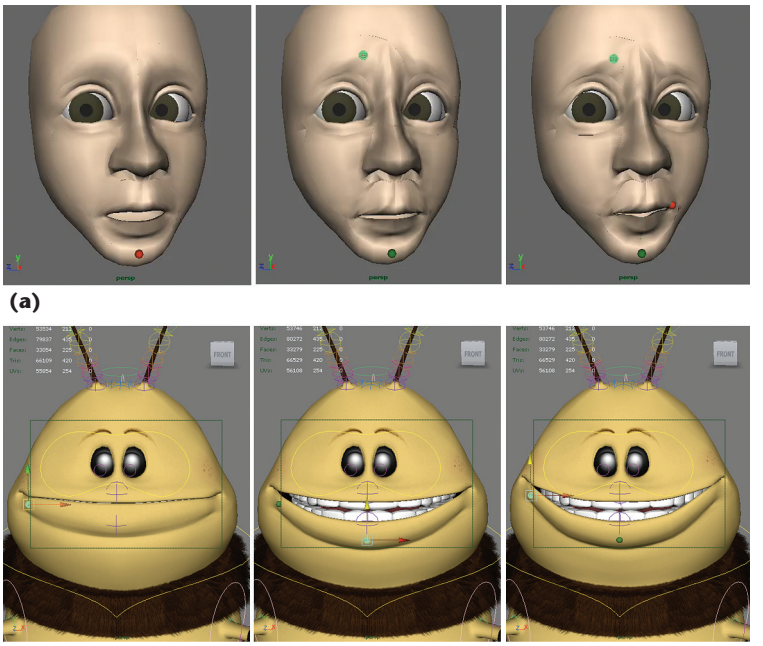


**Figure 3. Screenshots of our system in operation. (a) A blendshape model posed by direct manipulation rather than traditional slider weight editing. (b) Control over a smile's shape by direct manipulation of several vertices. The figures are screen captures of the Maya program interface and are low resolution owing to hardware rendering.**

Instead, we find the movement that causes the smallest change in sliders, subject to interpolating the active vertex. This solution embodies the second observation, that the sliders are an appropriate parameterization of the model.

Using Equation B in the "Blendshape Facial Animation" sidebar, we could achieve a direct implementation of this approach by solving

$$\frac{1}{2}\min_{\mathbf{w}}\left\|\mathbf{w}-\mathbf{w}_0\right\|^2+\boldsymbol{\lambda}^T\left(\mathbf{m}-\bar{\mathbf{B}}\mathbf{w}\right), \tag{1}$$

where $\mathbf{w}$ are the current blendshape weights, $\mathbf{w}_0$ are the previous weights, $T$ denotes transpose, $\mathbf{m}$ is the vector containing the moved manipulators, $\bar{\mathbf{B}}$ are the corresponding rows of the $\mathbf{B}$ matrix, and $\boldsymbol{\lambda}$ is a Lagrange multiplier vector.

Although this approach works, we instead minimize the distance between the model and the moved manipulator using least squares,

$$\min_{\mathbf{w}}\left\|\bar{\mathbf{B}}\mathbf{w}-\mathbf{m}\right\|^2+\alpha\left\|\mathbf{w}-\mathbf{w}_0\right\|^2, \tag{2}$$

rather than a hard constraint. (In our system, we set the regularization parameter $\alpha$ to a small number such as 0.1.) We explain the motivation for using Equation 2 rather than Equation 1 in the next two sections.

### *Pinning*

After the animator drags a vertex, it is desirable

## Blendshape Facial Animation

A blendshape model is a linear weighted sum of a number of blendshape "targets," which typically represent individual facial expressions (such as a smile or frown) or shapes based on approximations of facial-muscle actions or FACS (Facial Action Coding System) motions.[1,2] To simplify notation, we represent the face model as a column vector **f** containing all the model vertex coordinates in some arbitrary but fixed order, such as *xyzxyzxyz*. Similarly, we denote the blendshape targets as vectors $\mathbf{b}_k$, so the blendshape model is

$$\mathbf{f} = \sum_k w_k \mathbf{b}_k . \quad \text{(A)}$$

The carefully sculpted blendshape targets serve as interpretable controls; the span of these targets strictly defines the valid range of expressions for the modeled face (in practice, not all combinations produce equally plausible results). These characteristics differentiate the blendshape approach from those that involve linear combinations of uninterpretable shapes or algorithmically recombine the target shapes using a method other than that in Equation A.

We can place the blendshape targets in columns of a matrix **B** and gather the weights (also called blendshape sliders) in a vector **w** so that the blendshape model can be denoted as

$$\mathbf{f} = \mathbf{Bw}. \quad \text{(B)}$$

The model's geometric resolution (one-third of **B**'s row dimension) is independent of the animation resolution (the number of blendshape targets, or **B**'s column dimension). For example, a model with low geometric resolution nevertheless might have many targets, resulting in a rich space of deformations. In our system, any subset of vertex components (corresponding to particular rows of **f**) can be constraints, and direct manipulation occurs in these constraints' null space.

A well-known early animation popularized the blendshape approach,[3] and commercial animation packages were incorporating the approach by the late 1980s. The approach isn't associated with any single publication, however, and it received scant attention from researchers until the publication of papers by Frederic Pighin and his colleagues[4] and Pushkar Joshi and his colleagues.[5] Most research has focused on the problems of constructing a high-quality blendshape model from data,[4] driving blendshape models from motion-capture data,[6] or finding an automatic segmentation into independent, localized blendshape regions.[5]

Early on, animators debated the relative advantages of whole-face and delta blendshapes. In the whole-face scheme, each column of **B** represents an entire facial expression, such as a smile. In the delta form, the targets are offsets from a neutral face $\mathbf{f}_0$ that's added as $\mathbf{f} = \mathbf{Bw} + \mathbf{f}_0$, so many entries in each target vector are zero or nearly zero. In both cases, the weights are usually nonnegative (and reflect a convex combination of the basis shapes in the nondelta form)—although allowing mildly negative weights is occasionally useful. The whole-face approach makes it easier to reproduce the chosen expressions and can be efficient for modeling speech if the basis includes a set of visemes. On the other hand, this approach makes controlling local adjustments more difficult. Current industry practice most often uses delta blendshapes modeled on facial muscles.

Principal-component models resemble blendshape models in that they're also a linear sum of basis vectors. However, unlike blendshapes, principal-component models are orthogonal, and the individual basis vectors generally lack semantic interpretations.[7] Humans can sometimes interpret the first couple of principal components. However, the remaining components are incoherent and do not have clear semantic interpretations such as `RAISE-BROW`. (see the upper and middle rows in Figure 1 in the main article). It is difficult to estimate how much of a component to add to produce some desired expression. Another difference is that the PCA model's weights are zero-mean, not nonnegative.


**References**

1. P. Ekman and W. Friesen, *Manual for Facial Action Coding System*, Consulting Psychologists Press, 1978.
2. M. Sagar, "Facial Performance Capture and Expressive Translation for *King Kong*," *Siggraph Sketches*, ACM Press, 2006, article 26.
3. P. Bergeron and P. Lachapelle, "Controlling Facial Expression and Body Movements in the Computer-Generated Short 'Tony de Peltrie,'" *Siggraph Tutorial Notes*, ACM Press, 1985.
4. F. Pighin et al., "Synthesizing Realistic Facial Expressions from Photographs," *Proc. Siggraph*, ACM Press, 1998, pp. 75–84.
5. P. Joshi et al., "Learning Controls for Blend Shape Based Realistic Facial Animation," *ACM Siggraph 2006 Courses*, ACM Press, 2006, article 17.
6. B. Choe, H. Lee, and H.-S. Ko, "Performance-Driven Muscle-Based Facial Animation," *J. Visualization and Computer Animation*, vol. 12, no. 2, 2001, pp. 67–79.
7. D. Lee and H.S. Seung, "Learning the Parts of Objects by Non-negative Matrix Factorization," *Nature*, vol. 401, no. 6755, 1999, pp. 788–791.


to "pin" that vertex, forcing it to remain stationary as he or she drags and pins other vertices.[3] We easily accomplish this by simply appending the pinned vertices to **m** and the corresponding rows to $\bar{\mathbf{B}}$ (see Equation 2). So, the implementation doesn't distinguish between dragged and pinned

vertices; both are simply constraints. Each pinned vertex adds three rows to $\bar{\mathbf{B}}$ and $\mathbf{m}$, but partially constraining a vertex is straightforward—for example, by including only the rows corresponding to the $x$ and $y$ dimensions.

In a particularly simple model, it's possible (but unlikely) that when a sufficient number of vertices are constrained, $\bar{\mathbf{B}}$ will become full rank. In this case, the right-hand term in Equation 1 would fully constrain the facial movement. More generally, Equation 1 permits movement only in the constrained vertices' null space.

However, in practice, some additional directions are "almost" in the null space, corresponding to singular values that are almost but not quite zero (see Figure 4). Equation 2 allows movement in these directions, providing additional freedom with only a slight violation of the constraints. We obtain this flexibility at the expense of treating the manipulators as soft rather than hard constraints (we consider this a useful, natural trade-off). If needed, we can easily obtain more precise control over particular manipulators' influence by incorporating weights on each manipulator and solving Equation 2 using weighted least squares.

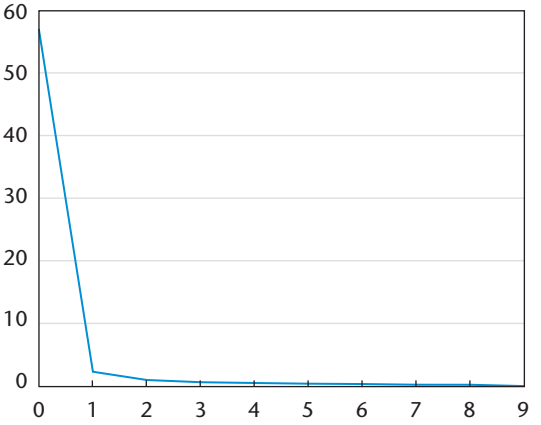



**Figure 4. The first 10 singular values of the blendshape basis of the model in Figure 5. (Small singular values correspond to weight directions that have little effect on the model.) Although the basis has full rank, many directions "almost" define a null space.**

### *Haphazard Movements*

An animator could drag vertices in directions and to extents that the model can't reasonably accommodate (see Figure 5). Blendshapes controlled by a standard slider interface have a similar problem (the animator can drag the sliders too far). This isn't a problem for experienced animators. However, we also cope with the problem by constraining the sliders to a range $[0, t]$ with $t$ set to 1.

Additionally, we incorporate a mild weight decay regularization term $\mu\|\mathbf{w}\|^2$, with $\mu$ = 0.001. We can understand this term's rationale by imagining a manipulator that repeatedly moves from one location to another and back again. In this scenario, after extensive manipulation, components of $\mathbf{w}$ in the null space of $\bar{\mathbf{B}}$ can grow owing simply to numerical error. The weight decay regularization opposes this possibility and biases the results away from extreme poses.

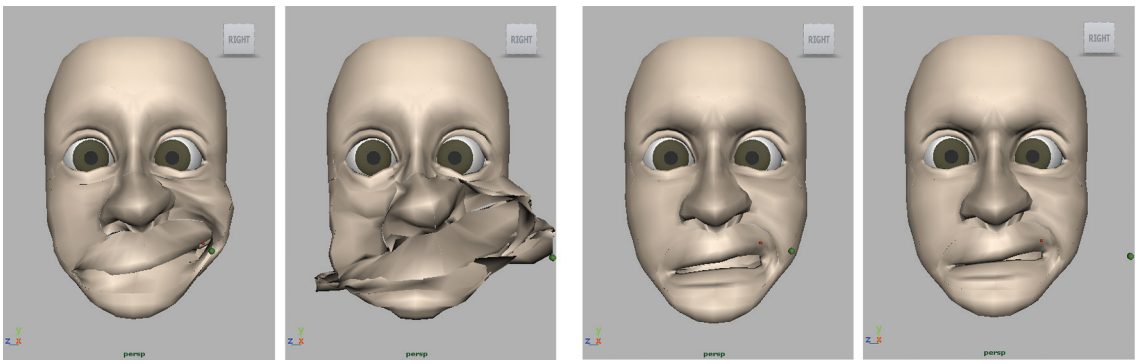
Putting these together, the general form of our objective is

$$\min_{\mathbf{w}} \|\bar{\mathbf{B}}\mathbf{w} - \mathbf{m}\|^2 + \alpha\|\mathbf{w} - \mathbf{w}_0\|^2 + \mu\|\mathbf{w}\|^2, \quad (3)$$

subject to $\mathbf{w} \in [0, t]$, giving the corresponding quadratic program:

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{S}\mathbf{w} + \mathbf{q}^T\mathbf{w}$$

subject to $\mathbf{G}\mathbf{w} \preceq \mathbf{h}$




**(a)** **(b)**

**Figure 5. Dealing with haphazard animator movements. (a) Model behavior with unconstrained weights as the animator drags the active manipulator to an unreasonable location. The green sphere is the animator's desired active-manipulator location. (b) The corresponding constrained solutions to this haphazard drag.**

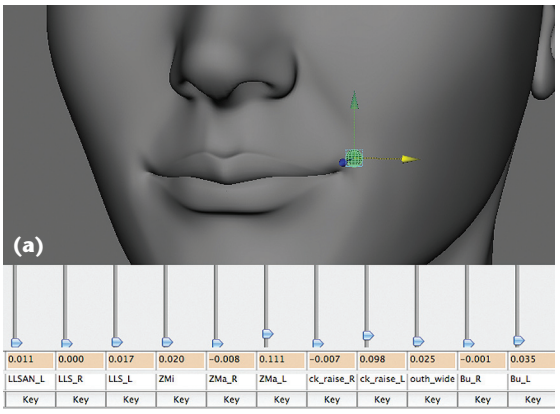



**Figure 6. The relationship between direct movements and slider movements. (a) Direct movement of a single vertex creates (b) many subtle slider movements. However, the converse is also true: a single slider movement in some cases equals multiple direct manipulations. Both approaches are desirable.**

with $\mathbf{S} = 2\left(\bar{\mathbf{B}}^T\bar{\mathbf{B}} + (\alpha+\mu)\mathbf{I}\right)$

$$\mathbf{q} = -2\bar{\mathbf{B}}^T\mathbf{m} + \alpha\mathbf{w}_0$$

$$\mathbf{G} = \left[\mathbf{I}; -\mathbf{I}\right]^T$$

$$\mathbf{h} = \left[t\mathbf{1}_n; \mathbf{0}_n\right]^T,$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{0}_n$ and $\mathbf{1}_n$ are zero- and one-vectors of length $n$. Because the matrix $\mathbf{S}$ is $m \times m$ (with $m \approx 100$), we easily solve this at interactive rates on current machines. Figure 5 shows the results of Equation 3 versus Equation 2 on an unreasonable movement.

### Implementation Assumptions

Our work targets professional animators rather than novice users. Current animation practice employs large software packages such as Maya that have reusable components implementing most of our needed functionality, including basic blendshapes, spline surfaces, constraining of points to surfaces, and interaction in a perspective window. We further assume that the animator is skilled in this software. In particular, in the case of spline models we provide direct manipulation of the spline control vertices. This is standard practice in manipulating spline surfaces and not a challenge for skilled animators.

## Results

Direct-manipulation editing is a familiar concept and presents few surprises. Figure 3a shows a blendshape model posed by our direct manipulation at interactive rates, using three constraints ("moves"). Producing such an expression using traditional slider editing would typically require several iterations of weight adjustment, as well as trial and error. Figure 3b shows how direct manipulation can control a cartoon character's smile. A companion video (available at http://scribblethink.org) shows additional examples of interactive direct manipulation of standard blendshape models.

### Direct Manipulation versus Parametric Control

In facial animation, we can illustrate the benefits of offering both direct manipulation and indirect parameter control in terms of the effort required for different facial edits. As we mentioned before, a single direct-manipulation vertex move generally equals many slider movements.

To see this mathematically, consider solving Equation 2 for the weights

$$\mathbf{w} = \left(\bar{\mathbf{B}}^T\bar{\mathbf{B}} + \alpha\mathbf{I}\right)^{-1}\left(\bar{\mathbf{B}}^T\mathbf{m} + \alpha\mathbf{w}_0\right).$$

This equation is of the form

$$\mathbf{w} = \mathbf{X}\mathbf{m} + \mathbf{k}, \tag{4}$$

with $\mathbf{X} = \left(\bar{\mathbf{B}}^T\bar{\mathbf{B}} + \alpha\mathbf{I}\right)^{-1}\bar{\mathbf{B}}^T$. Consider the effect of $\mathbf{m}$ containing a single pinned vertex. The $\mathbf{X}$ matrix is denser than $\bar{\mathbf{B}}$ owing to the convolution effect of the matrix multiplication and the bandwidth increase due to the matrix inverse. So, the direct-manipulation move will produce many nonzero weights (see Figure 6).

Although it might seem contradictory, the converse is also true: a single slider movement can equal multiple direct manipulations. We can analyze the situation by considering

$$\mathbf{m} = \bar{\mathbf{B}}\mathbf{w}.$$

With the weight change regularization $\min\|\mathbf{w} - \mathbf{w}_0\|$ in Equation 2, columns of $\bar{\mathbf{B}}$ corresponding to unaltered sliders have little effect. We can imagine the system containing only columns corresponding to the altered sliders in $\mathbf{w}$. In this case, the constraints provided by a single pinned 3D vertex provide information to solve up to three unknowns in $\mathbf{w}$. However, the vertex corresponding to $\mathbf{m}$ might be nonzero in a number of blendshapes. If more than three blendshape targets affect the vertex, a full specification of the state of the face will require additional directly manipulated vertices. For example, if the animator has moved six sliders, obtaining the equivalent facial shape with direct manipulation will require pinning at least two vertices.

So, both direct manipulation and slider control are advantageous in different circumstances, and both should be provided. Slider control is more effective when the desired action directly corresponds to a particular slider's effect or when the edits must be reproducible and numerically accurate. On the other hand, direct manipulation is more intuitive and generally more powerful, because of the matrix's action in Equation 4 and because each direct-manipulation edit specifies several DOF.

### *Unifying Whole-Expression and Delta Blendshape Views*

As we discuss in the "Blendshape Facial Animation" sidebar, using blendshape targets that represent complete expressions, such as a viseme or a smile, can save time over recreating an expression from muscle-based targets each time we require it. On the other hand, it's difficult to make exact, local adjustments by blending a set of targets with global influence. As a result, many animators avoid whole-expression blendshapes when subtle, local control is required.

Our technique lets us easily combine localized adjustments with whole-expression control. The animator first creates any desired set of whole-expression blendshapes. Then, the system saves the weight vectors expressing these shapes in terms of the original local blendshape targets.

The animator can blend these expressions as desired. Internally, the result is a new weight vector that's itself a weighted sum of the $\mathbf{w}_e$ vectors. To allow localized adjustment, we simply use this vector in the role of $\mathbf{w}_0$ in Equation 3. In other words, direct manipulation will now stay as close as possible to the previously edited whole expression while respecting the desired local edit (see Figure 7). The $\mathbf{w}_e$ specifications of whole-expression blendshape targets can optionally include only a subset of sliders. For example, one could specify a viseme library using only the sliders affecting the mouth region.

### *Animator Feedback*

Comments from professional animators have been generally positive. One animator said that the direct-manipulation capability resembles clusters, but with the advantage of working within the space of the blendshape targets. (Clusters are a weighted fall-off skin deformation technique that lets animators directly move surface vertices.)

Another comment was that the technique is advantageous when you're adjusting animations that another animator created. This is a common situation owing to the increasing use of motion capture in movies such as *Beowulf* and *Avatar*. In these projects, animators animate the nonhumanoid characters, but they also make small alterations to the original actors' performances due to needs that become apparent only after the motion is captured.

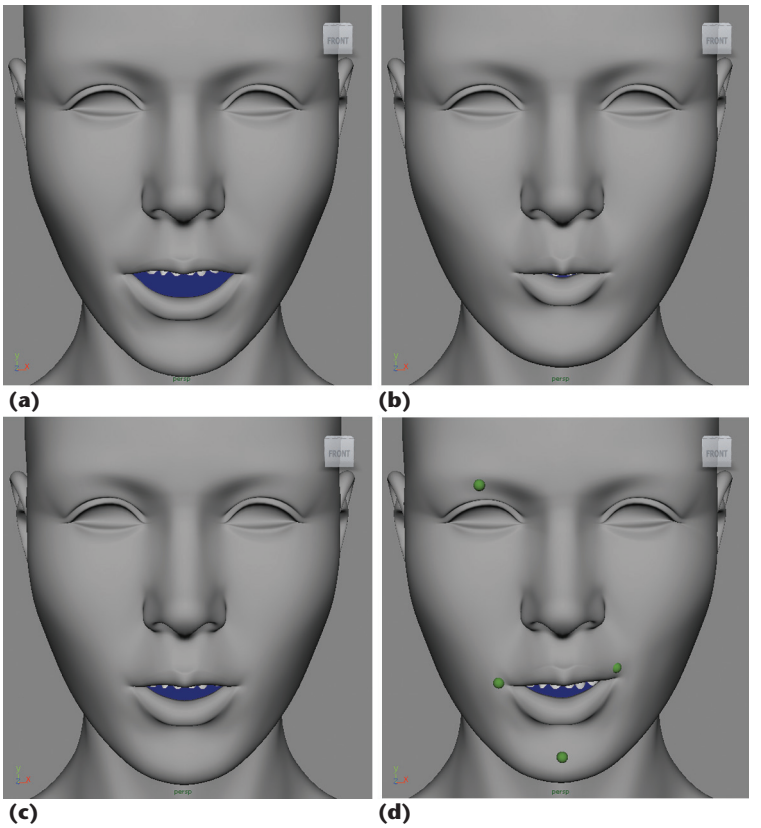



**Figure 7. Using direct manipulation to make local adjustments on whole-face expressions. The whole-face (a) "ah" and (b) "oo" visemes created in advance. (c) A weighted combination of these two expressions. (d) The result of local direct manipulation.**

To understand the animator's comment, remember that producing original facial animation is laborious. Over the course of days and weeks, the animator forms a strong memory of both the model and the logic of the slider movements used to create the animation. Animators are also trained to prefer a particular logic and discipline of slider movements—for example, some animators always move sliders corresponding to particular major muscles first. When animators must adjust an animation they didn't create, however, they don't have any memory of the slider combinations that created particular movements. In this situation, adjusting the model using only sliders can involve much trial and error.

Negative comments focused on aspects of our admittedly prototypical software implementation. For example, when animators drag direct-manipulation pins, the corresponding sliders don't update until they release the mouse. However, this is simply a

limitation of our current implementation rather than of the method itself.

Initial feedback confirms that animators regard our direct-manipulation framework as natural and obvious, and we believe it will become a practical, effective technique. The direct-manipulation problem's underconstrained nature could also permit future extensions using a statistical prior on the facial expression. Similar statistically based approaches have proved successful in other face-editing research as well as style-based inverse-kinematics animation of the body.


## Acknowledgments

*Hiroki Itokazu, Tatsuo Yotsukura, Ng Jiahao, Nickson Fong and Egg Story Creative Productions, Dejan Momcilovic, Kyle Labad, Raymond Chan, and Gray Hodgkinson provided models and animation data. We thank Yosuke Katsura, Ayumi Kimura, Luca Fascione, Nebojsa Dragosavac, Andy Jones, Ben Forster, Pravin Bhat, James Jacobs, Sebastian Sylwan, and Joe Letteri for discussions and help. The Japan Science and Technology Agency's Core Research of Evolutional Science and Technology project partially supported this research.*


## References


1. M. Unuma, K. Anjyo, and R. Takeuchi, "Fourier Principles for Emotion-Based Human Figure Animation," *Proc. Siggraph*, ACM Press, 1995, pp. 91–96.
2. A. Bruderlin and L. Williams, "Motion Signal Processing," *Proc. Siggraph*, ACM Press, 1995, pp. 97–104.
3. K. Yamane and Y. Nakamura, "Natural Motion Animation through Constraining and Deconstraining at Will," *IEEE Trans. Visualization and Computer Graphics*, vol. 9, no. 3, 2003, pp. 352–360.



***J.P. Lewis*** *is a member of Weta Digital's research group and a senior lecturer at Victoria University in New Zealand. His research interests include facial animation and computer vision. Lewis has published more than 40 papers on a variety of topics and has several patents and film credits. Contact him at zilla@computer.org.*

***Ken Anjyo*** *is the digital-effects and R&D supervisor at OLM Digital and an affiliate professor at Kyushu University. His research focus is on the construction of mathematical and computationally tractable models for computer graphics. Anjyo has a PhD in information engineering from Nagoya University. Contact him at anjyo@acm.org.*