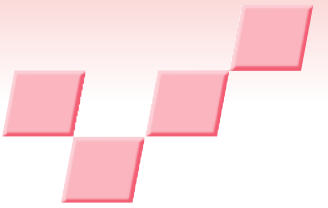


Orthogonal-Blendshape-Based Editing System for Facial Motion Capture Data

Qing Li and Zhigang Deng ■ *University of Houston*

A new data-driven system combines the automated construction of an orthogonal blendshape face model with constrained weight propagation to intuitively and efficiently edit facial motion capture sequences.

The accuracy and robustness of motion capture has made it a popular technique for acquiring preplanned natural human movements. But owing to its high cost, the computer animation community is searching for ways to efficiently and intuitively transform prerecorded motion capture data for novel applications, such as retargeting motion capture data to people with different body proportions. Although researchers have pursued various methods of editing body movement, relatively little research exists on editing facial motion capture data. In contrast to body motion, facial motion capture data don't have an obvious internal structure, so animators can't directly apply inverse-kinematic models to this scenario without considerable effort.

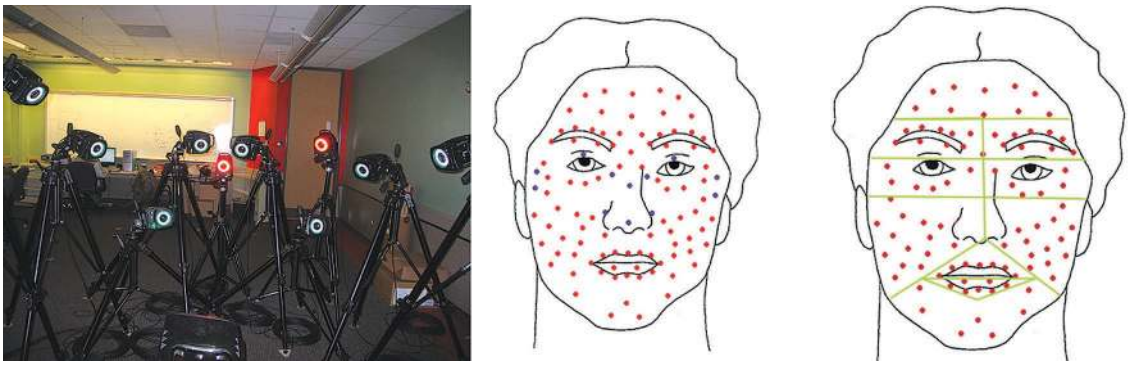
In this work, we present a data-driven 3D facial motion capture editing system that uses the automated construction of an orthogonal-blendshape face model and a constrained weight propagation. Given a collected facial motion capture data set, we start by performing a region-based principal component analysis (PCA) decomposition and constructing a truncated PCA space spanned by the largest eigenvectors for each anatomical region of the human face (for example, the left eyebrow). We then construct an orthogonal-blendshape face model as follows: each eigenvector of an anatomical region corresponds to a blendshape basis, so we regard its PCA coefficient as its blendshape weight. Our orthogonal model also minimizes the blendshape interference issue[1] that can affect the efficiency of the overall approach. Because this interference issue is due mainly to the nonorthogonality of blendshape bases, our approach can minimize the issue by automatically constructing an orthogonal-blendshape model at the beginning. Finally, we transform each frame of a new facial motion capture sequence into blendshape weights by projecting it into the PCA spaces mentioned earlier on a per-region basis. As such, modifying the blendshape weights (PCA coefficients) is equivalent to editing the underlying motion capture sequence.

Facial Motion Capture

In the data collection stage, we captured high-quality 3D facial motions by using a Vicon motion capture system (see the left panel of Figure 1a). We directed an actress with 102 markers on her face to speak a delicately designed corpus (composed of hundreds of sentences) four times, with each repetition spoken wearing a different facial expression. In this data recording, we considered a total of four basic facial expressions: neutral, happiness, anger, and sadness.

We recorded the facial-motion data at a 120Hz sampling rate and collected approximately 135 minutes of motion capture data. Owing to tracking errors caused by rapid, large head movements and the removal of unnecessary facial markers, we used only 90 of 102 markers for this work. After data capture, we normalized the data by removing head motion from the motion capture frames as follows: we first translated all the markers so that a specific marker was at the local coordinate center of each frame and then used method based on a

 Published by the IEEE Computer Society



(a) (b) (c)

Figure 1. Facial motion capture and facial region-dividing scheme. (a) Our facial motion capture system, (b) the facial marker layout we used, and (c) our region-dividing scheme. Blue and red points together represent the total 102 captured markers; the red points are the 90 markers we used for this work.

singular value decomposition (SVD)[2] to calculate head motion. Figure 1b shows the 102 facial marker layout and the 90 markers we kept.

Region-Based Principal Component Analysis

We concatenated the 3D positions of each of the 90 valid markers of a single facial-motion capture frame to form a 270-dimensional facial motion vector. However, to keep more than 90 percent of the possible variations of these vectors in a single truncated PCA space, a retained dimensionality of greater than 20 would result. Furthermore, because PCA is essentially a global transformation/reduction, there is no explicit and clear correspondence between PCA eigenvectors and facial movements. In other words, if users want to edit a specific facial region, it's extremely difficult for them to know how and which PCA coefficients to adjust. As such, facial-motion editing in the single truncated PCA space isn't intuitive for animators.

To automatically maintain the 3D face's natural appearance while a user edits the facial region, we extend the motion propagation algorithm proposed by Qingshan Zhang and his colleagues[3] into a constrained weight propagation technique that lets animators optionally specify which blendshape controls (and their weights) shouldn't be affected in later editing. Figure 2 shows the schematic overview of this approach.

To provide local controls, we adopted an anatomical segmentation scheme to divide the whole face (using its facial markers) into 10 different regions (see Figure 1b): forehead, left eyebrow, right eyebrow, left eye, right eye, left cheek, right cheek, upper lip, lower lip, and jaw. For the markers in each of the 10 regions, we constructed a separate PCA space. We experimentally set the reduced dimensionality to three because this retains more than 90 percent of the motion variation for every facial region.

In our experiments, we found that because we applied PCA to a specific facial region's motions, increasing or decreasing the weights of its largest three eigenvectors often corresponds to perceptual movements of that region. For example, when users increase the weight (PCA coefficient) of the largest eigenvector for the left eyebrow region, the left eyebrow is accordingly raised. Figures 3a and 3b (see next page) show two examples of how these retained eigenvectors approximately correspond to perceptual movements for certain facial regions.

Automated Orthogonal Blendshape Construction

From the region-based PCA decompositions we've just described, our approach automatically constructs an orthogonal-blendshape face model with 30 blendshape bases (each region has three blendshape bases and 10 total regions) as follows: a retained eigenvector of each facial region corresponds

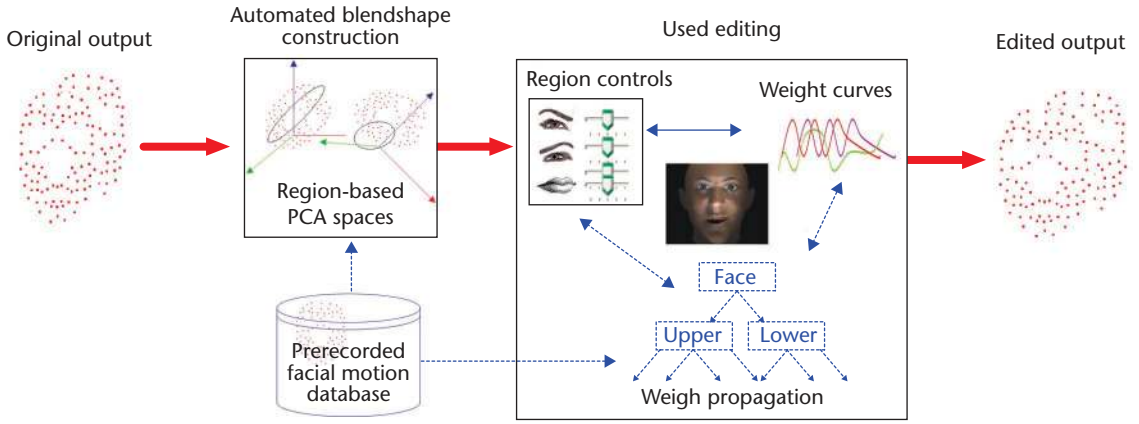


Figure 2. Schematic overview. Our system consists of two main components: after an orthogonal blendshape face model is automatically constructed (the first stage), users can perform various editing operations on the constructed blendshape face model (the second stage).

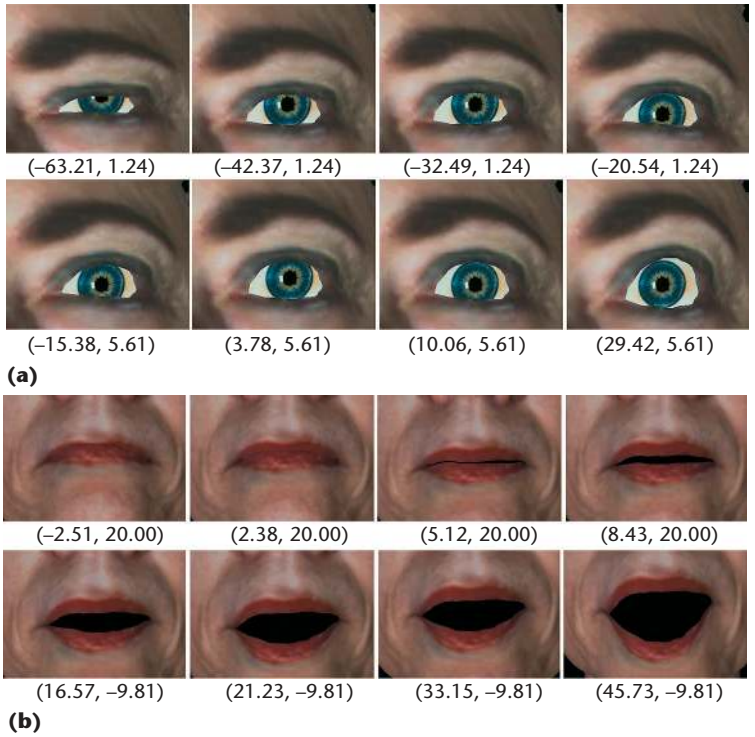


Figure 3. Examples of perceptual correspondences between the retained eigenvectors and facial movements: (a) Dominant principal component analysis (PCA) coefficient controls for the left eyebrow region. Duples show the PCA coefficients; when the first PCA coefficient is increased, the left eyebrow is raised. (b) Dominant PCA coefficient controls for the lower lip region. Duples in the figure show the PCA coefficients. When the first PCA coefficient is increased, the lower lip opens more.

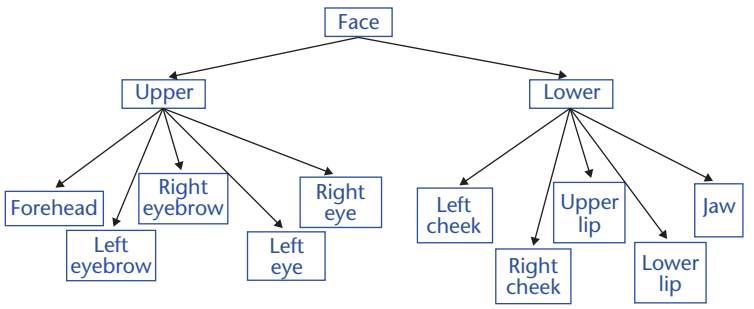


Figure 4. Face hierarchy for weight propagation. The whole face (root) has two sublevels (the upper face and the lower face), and has ten leaf nodes that correspond to separate facial regions.

to a blendshape basis, with its projected PCA coefficient serving as its weight. Equation 1 describes the constructed blendshape model:

$$MrkMotion = MeanMotion + \sum_{i=1}^{10}\sum_{j=1}^{3} C_{i,j} * EigV_{i,j} \quad (1)$$

Changing these retained PCA coefficients (equivalent to moving the sliders of blendshape weights) leads to a change of the underlying facial motion capture frame. Throughout the remainder of this article, we use weights and PCA coefficients, as well as blendshape bases and region-based PCA eigenvectors, interchangeably.

The blendshape approach often suffers from an interference problem due to nonorthogonal blendshape bases.[1] In this work, the constructed blendshape bases are orthogonal because motion markers in different facial regions don't intersect each other, and retained eigenvectors in a facial region are orthogonal to each other. As such, our system avoids blendshape interference or at least minimizes it.

Given a facial motion capture sequence as the input of an editing task, we can now project every frame of the sequence to the region-based truncated PCA spaces and obtain corresponding weights for all blendshape bases. Animators can then further edit these continuous weight curves in the time domain as shown in the next section.

Automated Weight Propagation

As illustrated in Figure 3, changing the weights of individual blendshape bases directly affects the movement of certain facial regions. However, as discussed previously, the motions of different facial regions are intrinsically correlated to each other. On the basis of on the hierarchical PCA-based motion propagation algorithm proposed by Zhang and his colleagues,[3] we developed a constrained weight propagation technique to adjust weights automatically while offering flexible user controls.

Specifically, we use a hierarchical PCA[3] for blendshape weight propagation. Initially, we divide the face into 10 leaf nodes (each corresponding to a divided region in Figure 2b), and then construct two intermediate nodes (the upper face and the lower face). We consider the whole face as the root of this hierarchy; Figure 4 shows the constructed hierarchical face structure.

For each node in Figure 4, we construct a truncated PCA space. We retain the 20 most dominant eigenvectors (corresponding to the largest 20 eigenvalues) for the root node in this work, the most dominant 10 eigenvectors for the two intermediate nodes (the upper face and the lower face), and the three most dominant eigenvectors for the 10 leaf nodes.

This weight propagation procedure follows two rules: it chooses to move up prior to moving down in the hierarchy, and it visits each node only once. The propagation works as follows:

- When users change a blendshape basis's weight, the propagation starts at the lowest hierarchy, which is one of the 10 left nodes.
- The change of blendshape weights propagates up to the middle of the hierarchy (the upper or lower face node) and then moves up to the root node (the entire face).
- Finally, the propagation moves down again to the middle node that it hasn't yet visited and keeps going up and down until it visits all the nodes.

(a) (b) (c)

Figure 5. The first example of weight propagation results: (a) The initial face (before editing), (b) the edited 3D face (the editing operation is the lower lip stretching, just before the weight propagation starts), and (c) its propagated results (after weight propagation). Comparing (b) and (c), we clearly see that the weight propagation process properly adjusted the weights to make the whole face look more natural.

(a) (b) (c)

Figure 6. The second example of weight propagation results: (a) The initial face (before editing), (b) the edited 3D face (the editing operation is the right eyebrow raising, just before weight propagation starts), and (c) the propagated results (after weight propagation). Note that after weight propagation, not only is the right eyebrow properly adjusted, but the left eyebrow is also automatically adjusted accordingly.

```
 1: set h to the hierarchy level of F*.;
 2: if hasBeenProcessed(F*) then
 3:   return;
 4: end if
 5: Compute Proj(δV,F*);
 6: Update δV with Proj(δV,F*);
 7: Set hasBeenProcessed(F*) to be true;
 8: for ∀ F ⊆ (level(F) = h − 1 and F ∩ F* = NonEmpty) do
 9:   WeightPropagation(F);
10: end for
11: for ∀ F ⊆ (level(F) = h + 1 and F ∩ F* = NonEmpty) do
12:   WeightPropagation(F);
13: end for
```

Figure 7. The weight propagation algorithm. Input: *F, the selected node in the hierarchy.**

For each node it visits, our approach projects the facial markers contained in the node to the PCA subspace spanned by the node's retained principal components. In other words, the projection is the best approximation of the propagated motions in the node's PCA subspace. For more details about this propagation algorithm, please refer to the work of Zhang and his colleagues.[3] Figures 5 and 6 show two examples of how this weight propagation technique automatically adjusts weights to make the edited 3D face appear more natural.

To illustrate the process, we briefly summarize the motion propagation algorithm's basic steps in Figure 7. Here, F represents any node in the hierarchy, δV represents the displacement vector of all markers, and $Proj(\delta V, F*)$ denotes the projection

Related Work

Geometrically deforming a 3D face model is a natural way to generate facial animation. However, manually sculpting faces every two to three frames is a painstaking process, with repetitive manual work needed to produce realistic facial animation with appropriately fine details. Owing to its efficiency and simplicity, the blendshape approach is widely used for keyframing facial animations. Recent research efforts focus on ways to both reduce blendshape interference by selected motion attenuation[1] and map facial motion capture sequences to blendshape face models.[2]

Essentially, geometric deformation and blendshape approaches are designed to simultaneously move and edit a group of relevant vertices of facial geometry. However, human-facial motion is the consequence of subtle skin deformation supported by the relaxation and contraction of hidden facial muscles, where the motions of various facial regions are intrinsically correlated to each other. Local region-based or global deformation approaches often require users to switch editing operations in different facial regions to produce 3D facial animation with a fine level of detail. Furthermore, it's extremely difficult to judge which facial pose is closer to a real human face.

Several data-driven facial animation editing techniques attempt to address this issue by exploiting rich correlations in collected data sets.[1,3–7] Erika S. Chuang and her colleagues apply the bilinear model to transform an input 2D facial animation sequence to a new one with a different expression.[4] Most recently, Daniel Vlasic and his colleagues successfully applied the multilinear model for the purpose of facial animation transferring.[5]

Yong Cao and his colleagues apply the independent component analysis (ICA) algorithm to recorded expressive facial motion capture data, and then interpret certain ICA components as expression or speech components.[6] Because their technique is essentially performed on a whole-face basis, each ICA component is not purely local and often affects facial movements nonlocally. As such, making local adjustments on specific facial regions is not well addressed in their work.

Based on a blendshape representation for 3D face models, Pushkar Joshi and his colleagues present an interactive tool for editing 3D face geometry by learning controls through physically motivated face segmentation. Their approach presents a rendering algorithm for preserving visual realism in the editing.[7] However, it also requires a processed blendshape face model as an input, while our approach does not and adaptively constructs a blendshape face model from a static 3D face model. Furthermore, in order to output edited facial motion capture data (sequences) in Joshi's work, a mapping algorithm from the weights of the blendshape model to 3D marker motions is additionally required.

The geometry-driven face-image-editing technique generates expression details on 2D images by constructing a PCA-based hierarchical representation from a selected number of training 2D face images. When users move one or several points on the 2D image, a motion-propagation algorithm automatically computes the movements of other facial control points.[3] Most recently, Edwin Chang and Odest Chadwicke Jenkins presented a 2D sketch interface for posing 3D faces. In their work, users can intuitively draw 2D strokes to search for the optimal facial pose.[8]

References

1. J.P. Lewis et al., "Reducing Blendshape Interference by Selected Motion Attenuation," *Proc. ACM Siggraph Symp. Interactive 3D Graphics and Games*, ACM Press, 2005, pp. 25–29.
2. Z. Deng et al., "Animating Blendshape Faces by Cross Mapping Motion Capture Data," *Proc. ACM Siggraph Symp. Interactive 3D Graphics and Games*, ACM Press, 2006, pp. 43–48.
3. Q. Zhang et al., "Geometry-Driven Photorealistic Facial Expression Synthesis," *Proc. Symp. Computer Animation*, Eurographics Assoc., 2003, pp. 177–186.
4. E.S. Chuang, H. Deshpande, and C. Bregler, "Facial Expression Space Learning," *Proc. Pacific Graphics 2002*, IEEE CS Press 2002, pp. 68–76.
5. D. Vlasic et al, "Face Transfer with Multilinear Models," *ACM Trans. Graphics*, vol. 24, no. 3, 2005, pp. 426–433.
6. Y. Cao, P. Faloutsos, and F. Pighin, "Unsupervised Learning for Speech Motion Editing," *Proc. Symp. Computer Animation*, Eurographics Assoc., 2003, pp. 225–231.
7. P. Joshi et al., "Learning Controls for Blend Shape Based Realistic Facial Animation," *Proc. Symp. Computer Animation*, Eurographics Assoc., 2003, pp. 35–42.
8. E. Chang and O.C. Jenkins, "Sketching Articulation and Pose for Facial Animation," *Proc. Symp. Computer Animation*, Eurographics Assoc., 2006, pp. 271–280.

of the $F*$ part of δV to the truncated PCA space of the node $F*$.

Constrained Weight Propagation

To automatically maintain the face's natural appearance, the weight propagation algorithm in Figure 7 will affect all facial regions when a specific region's weights (controls) are being edited. However, in some scenarios, automated propagation results might not be exactly what users need—for instance, when they want to exaggerate the edited 3D face's expressiveness or keep current configurations of certain facial regions as stationary as possible and not have them spoiled by later edit-

(a) (b) (c)

Figure 8. Constrained weight propagation example 1: (a) An original face, (b) the edited result with weight propagation to raise left and right cheeks, and (c) the edited result with constrained weight propagation so that upper lip, lower lip, and jaw region remain untouched.

(a) (b) (c)

Figure 9. Constrained weight propagation example 2: (a) An original face, (b) the edited result with weight propagation, resulting in effects to eyebrows and eyes, and (c) the edited result with constrained weight propagation so that forehead, left eyebrow, right eyebrow, left eye and right eye remain untouched.

ing. Here's a specific example: animators put the mouth to the correct shape but want to edit the cheeks' expressiveness to make the face look happier. In this case, if they move the cheek controls, the weight propagation algorithm will automatically adjust the mouth shape, which isn't what the animators want.

Our system uses a constrained weight propagation technique to achieve this flexibility. It works as follows. First, animators specify which facial regions are constrained (for example, left/right eyebrows) and won't be affected by later weight propagations. On the basis of the predefined mappings between markers and regions, our algorithm sets special flags on the markers of these constrained facial regions. Next, we need to modify the weight propagation procedure (Figure 7). The leaf nodes of the constrained facial regions (in this example, left/right eyebrow nodes) are exempt from processing, so when our algorithm processes their parent nodes (in this example, the upper and whole face node), markers with special flags won't be updated.

Constrained weight propagation provides an accessible tool for animators to balance the trade-off between user controls and automation; Figures 8 and 9 show two specific examples. Figure 8a shows an original face (deformed on the basis of a 3D facial motion capture frame), and Figure 8b shows the edited results when users raise the left or right cheeks. Notice that the weight propagation procedure automatically opens the mouth here without constraints. Figure 8c shows the result of constrained weight propagation—three regions (upper lip, lower lip, and jaw) are the constrained regions, so the mouth and jaw remain untouched while the cheeks are moved to the right place.

Figure 9 shows another application of constrained weight propagation. Notice that in Figure 9b, the weight propagation procedure without constraints affects the eyebrows and eyes (from an angry expression to a neutral one). Figure 9c shows the result of constrained weight propagation—five facial regions (forehead, left eyebrow, right eyebrow, left eye, and right eye) are constrained, so the eyebrows and eyes remain untouched while the mouth is moved to the intended closed position.

Besides modifying blendshape weights frame by frame, animators can also edit a sequence of frames by directly manipulating corresponding weight curves: we first fit a Catmull-Rom spline based on the weight sequence of each blendshape control, which lets the animators edit control points on the Catmull-Rom spline. We chose this spline primarily because of its smooth interpolation and local control properties.

Results and Experiments

We developed our 3D facial motion capture data

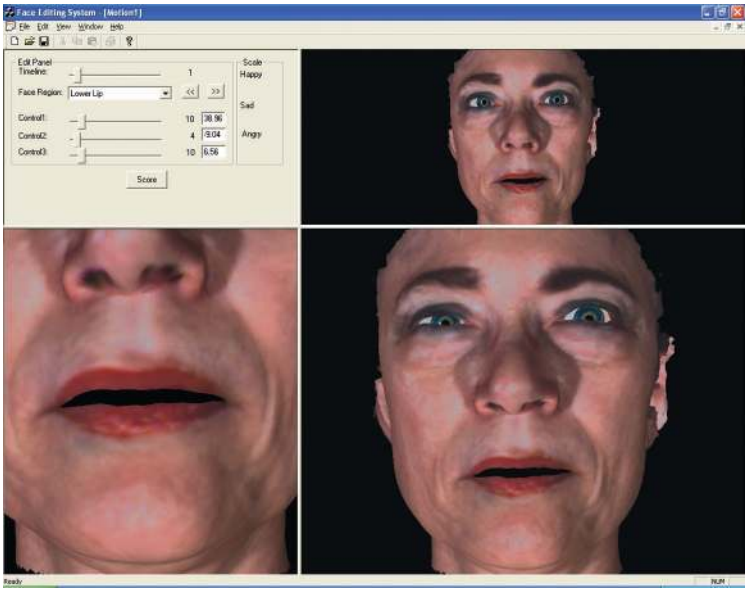


Figure 10. Our system in action. The top-left panel is a blendshape control window in which users can select a facial region and move its weight sliders, the bottom-left panel displays the edited results without weight propagation, the top-right panel displays the original facial motion capture sequence, and the bottom-right panel displays the edited facial-motion sequence (the final output) after constrained weight propagation is applied.

editing system using VC++ on a Microsoft Windows XP system. Figure 10 shows a snapshot of the running system. As shown, the interface has four panels: Three of these panels (bottom left, top right, and bottom right) can display facial motion capture data in two different modes: point rendered (each marker is rendered as a 3D point) or deformed. In the deformed 3D face display, we use a feature-point-based deformation technique[5] to distort a static 3D face model based on 3D facial motion capture data.

We've conducted more than 50 experiments using our system, including improving the animation quality of some initial synthesized or captured facial-motion sequences and exaggerating the expressiveness of some existing facial-motion sequences.

In future work, we plan to extend our data-driven facial motion editing system in two different directions. For example, given the mappings between predesigned blendshape face models and automatically constructed ones in this work, we can efficiently retarget any facial motion capture sequence to predesigned blendshape faces. As such, animators can conveniently animate and edit predesigned blendshape face models. We would also like to explore enhancing this facial motion editing technique with high-level feedback. For instance, when animators are performing an editing operation (such as expression exaggeration) on facial motion data, our system can intelligently compute the exaggerated expressiveness of the edited motion and provide instant feedback for further editing.

Acknowledgments

A new faculty research startup fund at the University of Houston funded this research. Special thanks go to Jose Baez-Franceschi, George Toderici, and Xiaohan Ma for face model preparation, Ulrich Neumann, J.P. Lewis, Joy Nash, Murtaza Bulut, and Carlos Busso for facial motion data capture and post-processing, and Tanasai Sucontphunt for writing the facial deformation code.

References

1. J.P. Lewis et al., "Reducing Blendshape Interference by Selected Motion Attenuation," *Proc. ACM Siggraph Symp. Interactive 3D Graphics and Games* (I3DG 05), ACM Press, 2005, pp. 25–29.
2. C. Busso et al., "Natural Head Motion Synthesis Driven by Acoustic Prosody Features," *Computer Animation and Virtual Worlds*, vol. 16, nos. 3–4, 2005, pp. 283–290.
3. Q. Zhang et al., "Geometry-Driven Photorealistic Facial Expression Synthesis," *Proc. Symp. Computer Animation*, Eurographics Assoc., 2003, pp. 177–186.
4. P. Joshi et al., "Learning Controls for Blendshape Based Realistic Facial Animation," *Proc. Symp. Computer Animation*, Eurographics Assoc., 2003, pp. 35–42.
5. S. Kshirsagar, S. Garchery, and N.M. Thalmann, "Feature Point Based Mesh Deformation Applied to MPEG-4 Facial Animation," *Proc. Deform* 2000, Kluwer, 2000, pp. 23–34.

Qing Li *is a PhD student in the Department of Computer Science at the University of Houston. Her research interests include computer graphics, computer animation, virtual-human modeling, and animation. Li received her BS in computer science from the Beijing University of Chemical Technology. Contact her at qingli12@cs.uh.edu.*

Zhigang Deng *is an assistant professor of computer science and the founding director of the Computer Graphics and Interactive Media Lab at the University of Houston (http://graphics.cs.uh.edu). His research interests include computer graphics, computer animation, and visualization. Deng received his PhD in computer science from the University of Southern California. Contact him at zdeng@cs.uh.edu.*