Eurographics/SIGGRAPH Symposium on Computer Animation (2003)
D. Breen, M. Lin (Editors)

Vision-based Control of 3D Facial Animation

Jin-xiang Chai,[1†] Jing Xiao[1] and Jessica Hodgins[1]

[1] The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Abstract

Controlling and animating the facial expression of a computer-generated 3D character is a difficult problem because the face has many degrees of freedom while most available input devices have few. In this paper, we show that a rich set of lifelike facial actions can be created from a preprocessed motion capture database and that a user can control these actions by acting out the desired motions in front of a video camera. We develop a real-time facial tracking system to extract a small set of animation control parameters from video. Because of the nature of video data, these parameters may be noisy, low-resolution, and contain errors. The system uses the knowledge embedded in motion capture data to translate these low-quality 2D animation control signals into high-quality 3D facial expressions. To adapt the synthesized motion to a new character model, we introduce an efficient expression retargeting technique whose run-time computation is constant independent of the complexity of the character model. We demonstrate the power of this approach through two users who control and animate a wide range of 3D facial expressions of different avatars.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Animation; I.3.6 [Computer Graphics]: Interaction techniques; I.4.8 [Image Processing and Computer Vision]: Tracking

1. Introduction

Providing intuitive control over three-dimensional facial expressions is an important problem in the context of computer graphics, virtual reality, and human computer interaction. Such a system would be useful in a variety of applications such as video games, electronically mediated communication, user interface agents and teleconferencing. Two difficulties, however, arise in animating a computer-generated face model: designing a rich set of believable actions for the virtual character and giving the user intuitive control over these actions.

One approach to animating motion is to simulate the dynamics of muscle and skin. Building and simulating such a model has proven to be an extremely difficult task because of the subtlety of facial skin motions[40, 34]. An alternative solution is to use motion capture data[20, 30]. Recent technological advances in motion capture equipment make it possible to record three-dimensional facial expressions with high fidelity, resolution, and consistency. Although motion capture data are a reliable way to capture the detail and nuance of

† http://graphics.cs.cmu.edu/projects/face-animation

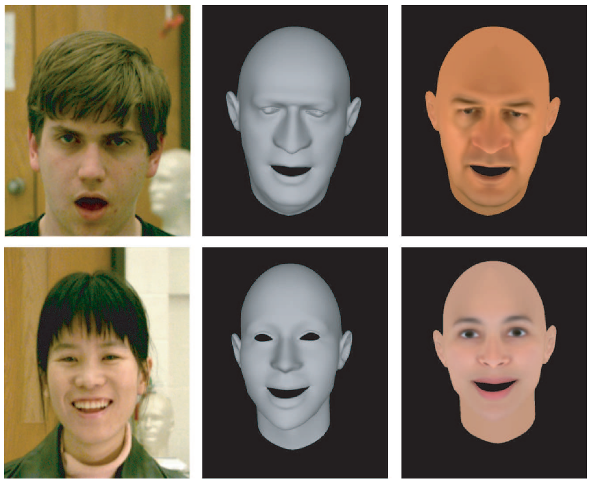
Figure 1: *Interactive Expression Control : a user can control 3D facial expressions of an avatar interactively. (Left) The users act out the motion in front of a single-view camera. (Middle) The controlled facial movement of the avatars with gray masks. (Right) The controlled facial movement of the avatars with texture mapped models.*

live motion, reuse and modification for a different purpose remains a challenging task.

Providing the user with an intuitive interface to control a broad range of facial expressions is difficult because character expressions are high dimensional but most available input devices are not. Intuitive control of individual degrees of freedom is currently not possible for interactive environments unless the user can use his or her own face to act out the motion using online motion capture. However, accurate motion capture requires costly hardware and extensive instrumenting of the subject and is therefore not widely available or practical. A vision-based interface would offer an inexpensive and nonintrusive alternative to controlling the avatar interactively, though accurate, high-resolution, and real-time facial tracking systems have not yet been developed.

In this paper, we propose to combine the strengths of a vision-based interface with those of motion capture data for interactive control of 3D facial animations. We show that a rich set of lifelike facial actions can be created from a motion capture database and that the user can control these actions interactively by acting out the desired motions in front of a video camera (figure 1). The control parameters derived from a vision-based interface are often noisy and frequently contain errors[24]. The key insight in our approach is to use the knowledge embedded in motion capture data to translate these low-quality control signals into high-quality facial expressions.

In the process, we face several challenges. First, we must map low-quality visual signals to high-quality motion data. This problem is particularly difficult because the mapping between the two spaces is not one-to-one, so a direct frame-by-frame mapping will not work. Second, in order to control facial actions interactively via a vision-based interface, we need to extract meaningful animation control signals from the video sequence of a live performer in real time. Third, we want to animate any 3D character model by reusing and interpolating motion capture data, but motion capture data record only the motion of a finite number of facial markers on a source model. Thus we need to adapt the motion capture data of a source model to all the vertices of the character model to be animated. Finally, if the system is to allow *any user* to control *any* 3D face model, we must correct differences between users because each person has somewhat different facial proportions and geometry.

1.1. Related Work

Facial animation has been driven by keyframe interpolation[31, 36, 27], direct parameterization[32, 33, 11], the user's performance[13, 43, 41, 14], pseudomuscle-based models[42, 9, 23], muscle-based simulation[40, 26], 2D facial data for speech[6, 5, 15] and full 3D motion capture data[20, 30]. Among these approaches, our work is most closely related to the performance-driven approach and motion capture, and we therefore review the research related to these two approaches in greater detail. Parke and Waters offer an excellent survey of the entire field[34].

A number of researchers have described techniques for recovering facial motions directly from video. Williams tracked expressions of a live performer with special makeup and then mapped 2D tracking data onto the surface of a scanned 3D face model[43]. Facial trackers without special markers can be based on deformable patches[4], edge or feature detectors[41, 25], 3D face models[14, 37, 12] or data-driven models[19]. For example, Terzpoloulos and Waters tracked the contour features on eyebrows and lips for automatic estimation of the face muscle contraction parameters from a video sequence, and these muscle parameters were then used to animate the physically based muscle structure of a synthetic character[41]. Essa and his colleagues tracked facial expressions using optical flow in an estimation and control framework coupled with a physical model describing the skin and muscle structure of the face[14]. More recently, Gokturk and his colleagues applied PCA on stereo tracking data to learn a deformable model and then incorporated the learned deformation model into an optical flow estimation framework to simultaneously track the head and a small set of facial features[19].

The direct use of tracking motion for animation requires that the face model of a live performer have similar proportions and geometry as those of the animated model. Recently, however, the vision-based tracking approach has been combined with blendshape interpolation techniques[32, 27] to create facial animations for a new target model[7, 10, 16]. The target model could be a 2D drawing or any 3D model. Generally, this approach requires that an artist generate a set of key expressions for the target model. These key expressions are correlated to the different values of facial features in the labelled images. Vision-based tracking can then extract the values of the facial expressions in the video image, and the examples can be interpolated appropriately.

Buck and his colleagues introduced a 2D hand-drawn animation system in which a small set of 2D facial parameters are tracked in the video images of a live performer and then used to blend a set of hand-drawn faces for various expressions[7]. The FaceStation system automatically located and tracked 2D facial expressions in real time and then animated a 3D virtual character by morphing among 16 predefined 3D morphs[16]. Chuang and Bregler explored a similar approach for animating a 3D face model by interpolating a set of 3D key facial shapes created by an artist[10]. The primary strength of such animation systems is the flexibility to animate a target face model that is different from the source model in the video. These systems, however, rely on the labor of skilled animators to produce the key poses for each animated model. The resolution and accuracy of the final animation remains highly sensitive to that of the visual control signal due to the direct frame-by-frame mappings adopted in these systems. Any jitter in the tracking system results in an unsatisfactory animation. A simple filtering technique like a Kalman filter might be used to reduce the noise and the jumping artifacts, but it would also remove the details and high frequencies in the visual signals.

Like vision-based animation, motion capture also uses measured human motion data to animate facial expressions. Motion capture data, though, have more subtle details than vision tracking data because an accurate hardware setup is used in a controlled capture environment. Guenter and his colleagues created an impressive system for capturing human facial expressions using special facial markers and multiple calibrated cameras and replaying them as a highly realistic 3D "talking head"[20]. Recently, Noh and Neumann presented an expression cloning technique to adapt existing motion capture data of a 3D source facial model onto a new 3D target model[30]. A recent notable example of motion capture data is the movie *The Lord of the Rings: the Two Towers* where prerecorded movement and facial expressions were used to animate the synthetic character "Gollum."

Both the vision-based approach and motion capture have advantages and disadvantages. The vision-based approach gives us an intuitive and inexpensive way to control a wide range of actions, but current results are disappointing for animation applications[18]. In contrast, motion capture data generate high quality animation but are expensive to collect, and once they have been collected, may not be exactly what the animator needs, particularly for interactive applications in which the required motions cannot be precisely or completely predicted in advance. Our goal is to obtain the advantage of each method while avoiding the disadvantages. In particular, we use a vision-based interface to extract a small set of animation control parameters from a single-camera video and then use the embedded knowledge in the motion capture data to translate it into high quality facial expressions. The result is an animation that does what an animator wants it to do, but has the same high quality as motion capture data.

An alternative approach to performance-based animation is to use high degree-of-freedom (DOF) input devices to control and animate facial expressions directly. For example, DeGraf demonstrated a real-time facial animation system that used a special purpose interactive device called a "waldo" to achieve real-time control[13]. Faceworks is another high DOF device that gives an animator direct control of a character's facial expressions using motorized sliders[17]. These systems are appropriate for creating animation interactively by trained puppeteers but not for occasional users of video games or teleconferencing systems because an untrained user cannot learn to simultaneously manipulate a large number of DOFs independently in a reasonable period of time.

Computer vision researchers have explored another

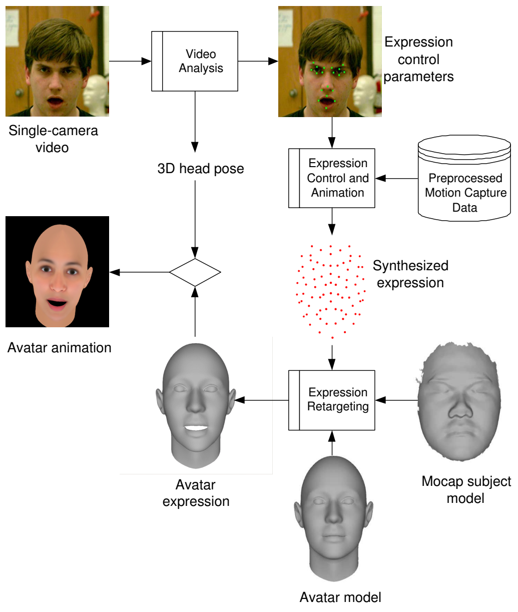

Figure 2: *System overview diagram. At run time, the video images from a single-view camera are fed into the Video Analysis component, which simultaneously extracts two types of animation control parameters: expression control parameters and 3D pose control parameters. The Expression Control and Animation component uses the expression control parameters as well as a preprocessed motion capture database to synthesize the facial expression, which describes only the movement of the motion capture markers on the surface of the motion capture subject. The Expression Retargeting component uses the synthesized expression, together with the scanned surface model of the motion capture subject and input avatar surface model, to produce the facial expression for an avatar. The avatar expression is then combined with the avatar pose, which is directly derived from pose control parameters, to generate the final animation.*

way to combine motion capture data and vision-based tracking[21, 35, 39]. They use motion capture data to build a prior model describing the movement of the human body and then incorporate this model into a probabilistic Bayesian framework to constrain the search. The main difference between our work and this approach is that our goal is to animate and control the movement of any character model rather than reconstruct and recognize the motion of a specific subject. What matters in our work is how the motion looks, how well the character responds to the user's input, and the latency. To this end, we not only want to remove the jitter and wobbles from the vision-based interface but also to retain as much as possible the small details and high frequencies in motion capture data. In contrast, tracking applications require that the prior model from motion capture data is able to predict the real world behaviors well so that it can provide a powerful cue for the reconstructed motion in the presence of occlusion and measurement noise. The details and high frequencies of the reconstructed motion are not the primary concern of these vision systems.

1.2. Overview

Our system transforms low-quality tracking motion into high-quality animation by using the knowledge of human facial motion that is embedded in motion capture data. The input to our system consists of a single video stream recording the user's facial movement, a preprocessed motion capture database, a 3D source surface scanned from the motion capture subject, and a 3D avatar surface model to be animated.

By acting out a desired expression in front of a camera, any user has interactive control over the facial expressions of any 3D character model. Our system is organized into four major components (figure 2):

- **Video analysis.** Simultaneously track the 3D position and orientation of the head and a small group of important facial features in video and then automatically translate them into two sets of high-level animation control parameters: expression control parameters and head pose parameters.
- **Motion capture data preprocessing.** Automatically separate head motion from facial deformations in the motion capture data and extract the expression control parameters from the decoupled motion capture data.
- **Expression control and animation.** Efficiently transform the noisy and low-resolution expression control signals to high-quality motion in the context of the motion capture database. Degrees of freedom that were noisy and corrupted are filtered and then mapped to the motion capture data; missing degrees of freedom and details are synthesized using the information contained in the motion capture data.
- **Expression retargeting.** Adapt the synthesized motion to animate all the vertices of a different character model at run time.

The motion capture data preprocessing is done off-line; the other stages are completed online based on input from the user. We describe each of these components in more detail in the next four sections of this paper.

2. Video Analysis

An ideal vision-based animation system would accurately track both 3D head motion and deformations of the face in video images of a performer. If the system is to allow *any* user to control the actions of a character model, the system must be user-independent. In the computer vision literature, extensive research has been done in the area of vision-based facial tracking. Many algorithms exist but the performance of facial tracking systems, particularly user-independent expression tracking, is not very good for animation applications because of the direct use of vision-based tracking data for animation. Consequently, we do not attempt to track all the details of the facial expression. Instead, we choose to robustly track a small set of distinctive facial features in real time and then translate these low-quality tracking data into high-quality animation using the information contained in the motion capture database.

Section 2.1 describes how the system simultaneously tracks the pose and expression deformation from video. The pose tracker recovers the position and orientation of the head, whereas the expression tracker tracks the position of 19 distinctive facial features. Section 2.2 explains how to extract a set of high-level animation control parameters from the tracking data.

2.1. Facial tracking

Our system tracks the 6 DOFs of head motion (yaw, pitch, roll and 3D position). We use a generic cylinder model to approximate the head geometry of the user and then apply a model-based tracking technique to recover the head pose of the user in the monocular video stream[45, 8]. The expression tracking step involves tracking 19 2D features on the face: one for each mid-point of the upper and lower lip, one for each mouth corner, two for each eyebrow, four for each eye, and three for the nose, as shown in figure 3. We choose these facial features because they have high contrast textures in the local feature window and because they record the movement of important facial areas.

Initialization: The facial tracking algorithm is based on a hand-initialized first frame. Pose tracking needs the initialization of the pose and the parameters of the cylinder model, including the radius, height and center position whereas expression tracking requires the initial position of 2D features. By default, the system starts with the neutral expression in the frontal-parallel pose. A user clicks on the 2D positions of 19 points in the first frame. After the features are identified in the first frame, the system automatically computes the cylindrical model parameters. After the initialization step, the system builds a texture-mapped reference head model for use in tracking by projecting the first image onto the surface of the initialized cylinder model.

Pose tracking: During tracking, the system dynamically updates the position and orientation of the reference head model. The texture of the model is updated by projecting the current image onto the surface of the cylinder. When the next frame is captured, the new head pose is automatically computed by minimizing the sum of the squared intensity differences between the projected reference head model and the new frame. The dynamically updated reference head model can deal with gradual lighting changes and self-occlusion,

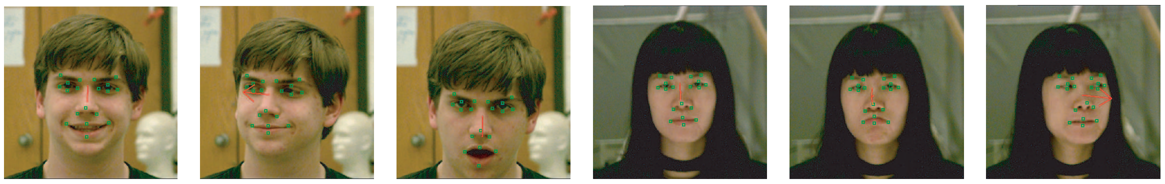

Figure 3: *User-independent facial tracking: the red arrow denotes the position and orientation of the head and the green dots show the positions of the tracked points.*

which allows us to recover the head motion even when most of the face is not visible.

Because the reference head model is dynamically updated, the tracking errors will accumulate over time. We use a re-registration technique to prevent this problem. At run time, the system automatically stores several texture-mapped head models in key poses, and chooses to register the new video image with the closest example rather than with the reference head model when the difference between the current pose and its closest pose among the examples falls below a user-defined threshold.

Some pixels in the processed images may disappear or become distorted or corrupted because of occlusion, non-rigid motion, and sensor noise. Those pixels should contribute less to motion estimation than other pixels. The system uses a robust technique, compensated iteratively re-weighted least squares (IRLS), to reduce the contribution of these noisy and corrupted pixels[3].

Expression tracking: After the head poses have been recovered, the system uses the poses and head model to warp the images into the fronto-parallel view. The positions of facial features are then estimated in the warped current image, given their locations in the warped reference image. Warping removes the effects of global poses in the 2D tracking features by associating the movements of features only with expression deformation. It also improves the accuracy and robustness of our feature tracking because the movement of features becomes smaller after warping.

To track the 2D position of a facial feature, we define a small square window centered at the feature's position. We make the simplifying assumption that the movement of pixels in the feature window can be approximated as the affine motion of a plane centered at the 3D coordinate of the feature. Affine deformations have 6 degrees of freedom and can be inferred using optical flow in the feature window. We employ a gradient-based motion estimation method to find the affine motion parameters thereby minimizing the sum of the squared intensity difference in the feature window between the current frame and the reference frame[2, 38, 1].

Even in warped images, tracking single features such as the eyelids based on intensity only is unreliable. To improve tracking robustness, we incorporate several prior geometric constraints into the feature tracking system. For example, we assign a small threshold to limit the horizontal displacement of features located on the eyelids because the eyelids deform almost vertically. The system uses a re-registration technique similar to the one used in pose tracking to handle the error accumulation of the expression tracking. In particular, the system stores the warped facial images and feature locations in example expressions at run time and re-registers the facial features of the new images with those in the closest example rather than with those in the reference image when the difference between current expression and its closest template falls below a small threshold.

Our facial tracking system runs in real time at 20 fps. The system is user-independent and can track the facial movement of different human subjects. Figure 3 shows the results of our pose tracking and expression tracking for two performers.

2.2. Control Parameters

To build a common interface between the motion capture data and the vision-based interface, we derive a small set of parameters from the tracked facial features as a robust and discriminative control signal for facial animation. We want the control parameters extracted from feature tracking to correspond in a meaningful and intuitive way with the expression movement qualities they control. In total, the system automatically extracts 15 control parameters that describe the facial expression of the observed actor:

- **Mouth (6):** The system extracts six scalar quantities describing the movement of the mouth based on the positions of four tracking features around the mouth: left corner, right corner, upper lip and lower lip. More precisely, these six control parameters include the parameters measuring the distance between the lower and upper lips (1), the distance between the left and right corners of the mouth (1), the center of the mouth (2), the angle of the line segment connecting the lower lip and upper lip with respect to the vertical line (1), and the angle of the line segment connecting the left and right corners of the mouth with respect to the horizontal line (1).
- **Nose (2):** Based on three tracked features on the nose, we compute two control parameters describing the movement of the nose: the distance between the left and right corners

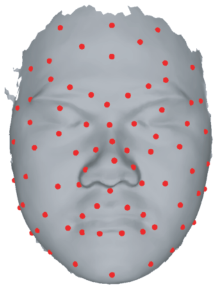

Figure 4: *The scanned head surface model of the motion capture subject aligned with 76 motion capture markers.*

of the nose and the distance between the top point of the nose and the line segment connecting the left and right corners (2).

- **Eye (2)**: The control parameters describing the eye movements are the distance between the upper and lower eyelids of each eye (2).
- **Eyebrow (5)**: The control parameters for the eyebrow actions consist of the angle of each eyebrow relative to a horizontal line (2), the distance between each eyebrow and eye (2), and the distance between the left and right eyebrow (1).

The expression control signal describes the evolution of $M = 15$ control parameters derived from tracking data. We use the notation $\tilde{Z}_i \equiv \{\tilde{z}_{a,i} | a = 1,...,15\}$ to denote the control signal at time i. Here a is the index for the individual control parameter. These 15 parameters are used to control the expression deformation of the character model. In addition, the head pose derived from video gives us 6 additional animation control parameters, which are used to drive the avatar's position and orientation.

3. Motion Capture Data Preprocessing

We used a Minolta Vivid 700 laser scanner to build the surface model of a motion capture subject[22]. We set up a Vicon motion capture system to record facial movement by attaching 76 reflective markers onto the face of the motion capture subject. The scanned surface model with the attached motion capture markers is shown in figure 4. The markers were arranged so that their movement captures the subtle nuances of the facial expressions. During capture sessions, the subject must be allowed to move his head freely because head motion is involved in almost all natural expressions. As a result, the head motion and the facial expressions are coupled in the motion capture data, and we need to separate the head motion from the facial expression accurately in order to reuse and modify the motion capture data. Section 3.1 describes a factorization algorithm to separate the head motion from the facial expression in the motion capture data by utilizing the inherent rank constraints in the motion capture data.

3.1. Decoupling Pose and Expression

Assuming the facial expression deforms with L independent modes of variation, then its shape can be represented as a linear combination of a deformation basis set $S_1, S_2, ..., S_L$. Each deformation basis S_i is a $3 \times P$ matrix describing the deformation mode of P points. The recorded facial motion capture data X_f combines the effects of 3D head pose and local expression deformation:

$$X_f = R_f \cdot (\sum_{i=1}^{L} c_{fi} \cdot S_i) + T_f \tag{1}$$

where R_f is a 3×3 head rotation matrix and T_f is a 3×1 head translation in frame f. c_{fi} is the weight corresponding to the *ith* deformation basis S_i. Our goal is to separate the head pose R_f and T_f from the motion capture data X_f so that the motion capture data record only expression deformations.

We eliminate T_f from X_f by subtracting the mean of all 3D points. We then represent the resulting motion capture data in matrix notation:

$$\underbrace{\begin{pmatrix} X_1 \\ \vdots \\ X_F \end{pmatrix}}_{M} = \underbrace{\begin{pmatrix} c_{11}R_1 & \dots & c_{1L}R_1 \\ \vdots & \vdots & \vdots \\ c_{F1}R_F & \dots & c_{FL}R_F \end{pmatrix}}_{Q} \cdot \underbrace{\begin{pmatrix} S_1 \\ \vdots \\ S_L \end{pmatrix}}_{B} \tag{2}$$

where F is the number of frames of motion capture data, M is a $3F \times P$ matrix storing the 3D coordinates of all the motion capture marker locations, Q is a $3F \times 3L$ scaled rotation matrix recording the head orientations and the weight for each deformation basis in every frame, B is a $3L \times P$ matrix containing all deformation bases. Equation (2) shows that without noise, the rank of the data matrix M is at most $3L$. Therefore, we can automatically determine the number of deformation bases by computing the rank of matrix M.

When noise corrupts the motion capture data, the data matrix M will not be exactly of rank $3L$. However, we can perform singular value decomposition (SVD) on the data matrix M such that $M = USV^T$, and then get the best possible rank $3L$ approximation of the data matrix, factoring it into two matrices:

$$\tilde{Q} = U_{3F,3L} S_{3L,3L}{}^{\frac{1}{2}}, \quad \tilde{B} = S_{3L,3L}{}^{\frac{1}{2}} V_{P,3L}{}^{T} \tag{3}$$

The rank of data matrix M, that is $3L$, is automatically determined by keeping a specific amount of original data energy. In our experiment, we found that 25 deformation bases were sufficient to capture 99.6% of the deformation variations in the motion capture database.

The decomposition of Equation (3) is determined up to a linear transformation. Any non-singular $3L \times 3L$ matrix G and its inverse could be inserted between $\tilde{Q}$ and $\tilde{B}$ and their product would still be the same. Thus the actual scaled rotation matrix Q and basis matrix B are given by

$$Q = \tilde{Q} \cdot G \quad B = G^{-1} \cdot \tilde{B} \tag{4}$$

with the appropriate $3L \times 3L$ invertible matrix G selected.

To recover the appropriate linear transformation matrix G, we introduce two different sets of linear constraints: *rotation constraints* and *basis constraints* on the matrix GG^T (For details, please see our technical report[44]). Rotation constraints utilize the orthogonal property of the rotation matrices to impose the linear constraints on the matrix GG^T. Given 3D motion capture data, the deformation bases representing the facial expression are not unique. Any non-singular transformation of deformation bases are still valid to describe the facial expressions. To remove the ambiguity, our algorithm automatically finds the L appropriate frames in the motion capture data that cover all the deformation variations in the data. We choose these L frames as the deformation basis. The specific form of the deformation basis provides another set of linear constraints on the matrix GG^T. These two sets of constraints allow us to uniquely recover the matrix GG^T via standard least-square techniques. We use the SVD technique again to factor the matrix GG^T into the matrix G and its transpose G^T, a step that leads to the actual rotation R_f, configuration coefficients c_{fi}, and deformation bases $S_1, ..., S_L$.

After we separate the poses from the motion capture data, we project each frame of the 3D motion capture data in the fronto-parallel view and extract the expression control parameters for each motion capture frame much as we extracted expression control parameters from the visual tracking data. Let $X_i \equiv \{\mathbf{x}_{b,i} | b = 1, ..., 76\}$ be 3D positions of the motion capture markers in frame i and $Z_i \equiv \{z_{a,i} | a = 1, ..., 15\}$ be the control parameters derived from frame i. Here $\mathbf{x}_{b,i}$ is the 3D coordinate of the bth motion capture marker corresponding to frame i. In this way, each motion capture frame X_i is automatically associated with animation control parameters Z_i.

4. Expression Control and Animation

Given the control parameters derived from a vision-based interface, controlling the head motion of a virtual character is straightforward. The system directly maps the orientation of the performer to the virtual character. The position parameters derived from video need to be appropriately scaled before they are used to control the position of an avatar. This scale is computed as the ratio of the mouth width between the user and the avatar.

Controlling the expression deformation requires integrating the information in the expression control parameters and the motion capture data. In this section, we present a novel data-driven approach for motion synthesis that translates the noisy and lower-resolution expression control signals to the high-resolution motion data using the information contained in motion capture database. Previous work in this area can be classified into two categories: synthesis by examples and synthesis by a parametric or probabilistic model. Both approaches have advantages: the former allows details of the original motion to be retained for synthesis; the latter creates a simpler structure for representing the data. In order to keep the details and high frequency of the motion capture data, we choose to synthesize the facial animation by examples. The system finds the K closest examples in the motion capture database using the low-resolution and noisy query control parameters from the vision-based interface and then linearly interpolates the corresponding high-quality motion examples in the database with a local regression technique. Because the mapping from the control parameters space to the motion data space is not one to one, the query control signal is based on multiple frames rather than a single frame thereby eliminating the mapping ambiguity by integrating the query evidence forward and backward over a window of a short fixed length.

The motion synthesis process begins with a normalization step that corrects for the difference between the animation control parameters in the tracking data and the motion capture data. We then describe a data-driven approach to filter the noisy expression control signals derived from the vision-based interface. Next, we introduce a data-driven expression synthesis approach that transforms the filtered control signals into high quality motion data. Finally, we describe a new data structure and an efficient K nearest points search technique that we use to speed up the synthesis process.

4.1. Control Parameter Normalization

The expression control parameters from the tracking data are inconsistent with those in the motion capture data because the user and the motion capture subject have different facial geometry and proportions. We use a normalization step that automatically scales the measured control parameters according to the control parameters of the neutral expression to approximately remove these differences. By scaling the control parameters, we ensure that the control parameters extracted from the user have approximately the same magnitude as those extracted from motion capture data when both are in the same expression.

4.2. Data-driven Filtering

Control signals in a vision-based interface are often noisy. We divide them into segments of a fixed, short temporal length W at run time and then use the prior knowledge embedded in the motion capture database to sequentially filter the control signals segment by segment. The prior model of the expression control signals is a local linear model learned at run time. When a new segment arrives, the motion capture database is searched for the examples that are most relevant to the segment. These examples are then used to build a local linear dynamical model that captures the dynamical behavior of the control signals over a fixed length sequence.

We collect the set of neighboring frames with the same temporal window W for each frame in the motion capture database and treat them as a data point. Conceptually, all the motion segments of the facial control signals form a nonlinear manifold embedded in a high-dimensional configuration space. Each motion segment in the motion capture database is a point sample on the manifold. The motion segments of control signals from a vision-based interface are noisy samples of this manifold. The key idea of our filtering technique is that we can use a low-dimensional linear subspace to approximate the local region of the high-dimensional nonlinear manifold. For each noisy sample, we apply Principal Component Analysis (PCA) to learn a linear subspace using the data points that lie within the local region and then reconstruct them using the low-dimensional linear subspaces.

If we choose too many frames for the motion segments, the samples from the motion capture data might not be dense enough to learn an accurate local linear model in the high-dimensional space. If we choose too few frames, the model will not capture enough motion regularities. The length of the motion segments will determine the response time of the system, or the action delay between the user and the avatar. From our experiments, we found 20 to be a reasonable number of frames for each segment. The delay of our system is then 0.33s because the frame rate of our video camera is 60fps.

Let $\tilde{\phi}_t \equiv [\tilde{Z}_1, ..., \tilde{Z_W}]$ be a fragment of input control parameters. The filter step is

- Find the K closest slices in the motion capture database.
- Compute the principal components of the K closest slices. We keep the M largest eigenvectors $U_1, ..., U_M$ as the filter basis, and M is automatically determined by retaining 99% of the variation of the original data.
- Project $\tilde{\phi}_t$ into a local linear space spanned by $U_1, ..., U_M$ and reconstruct the control signal $\bar{\phi}_t = [\bar{Z}_1, ..., \bar{Z}_b]$ using the projection coefficients.

The principal components, which can be interpreted as the major sources of dynamical variation in the local region of the input control signals, naturally captures the dynamical behavior of the animation control parameters in the local region of the input control signal $\tilde{\phi}_t$. We can use this low dimensional linear space to reduce the noise and error in the sensed control signal. The performance of our data-driven filtering algorithm depends on the number of closest examples. As the number of closest examples K increases, the dimension of subspace M becomes higher so that it is capable of representing a large range of local dynamical behavior. The high dimensional subspace will provide fewer constraints on the sensed motion and the subspace constraints might be insufficient to remove the noise in data. If fewer examples are used, they might not particularly similar to the online noisy motion and the subspace might not fit the motion very well. In this way, the filtered motion might be over smoothed and distorted. The specific choice of K depends on the properties of a given motion capture database and control data extracted from video. In our experiments, we found that a K between 50 to 150 gave us a good filtering result.

4.3. Data-driven Expression Synthesis

Suppose we have N frames of motion capture data $X_1, ..., X_N$ and its associated control parameters $Z_1, ..., Z_N$, then our expression control problem can be stated as follows: given a new segment of control signals $\bar{\phi}_t = [\bar{Z}_{t_1}, ..., \bar{Z}_{t_W}]$, synthesize the corresponding motion capture example $\bar{M}_t = [\bar{X}_{t_1}, ..., \bar{X}_{t_W}]$.

The simplest solution to this problem might be K-nearest-neighbor interpolation. For each frame $\bar{Z}_i$ such that $i = t_1, ..., t_W$, the system can choose the K closest example points of the control parameters, denoted as $Z_{i_1}, ..., Z_{i_K}$, in the motion capture data and assign each of them a weight ω_i based on their distance to the query $\bar{Z}_i$. These weights could then be applied to synthesize the 3D motion data by interpolating the corresponding motion capture data $X_{i_1}, ..., X_{i_K}$. The downside of this approach is that the generated motion might not be smooth because the mapping from control parameter space to motion configuration space is not one to one.

Instead, we develop a segment-based motion synthesis technique to remove the mapping ambiguity from the control parameter space to the motion configuration space. Given the query segment $\bar{\phi}_t$, we compute the interpolation weights based on its distance to the K closest segments in the motion capture database. The distance is measured as the Euclidean distance in the local linear subspace. We can then synthesize the segment of motion data via a linear combination of the motion data in the K closest segments. Synthesizing the motion in this way allows us to handle the mapping ambiguity by integrating expression control parameters forwards and backwards over the whole segment.

Both the dynamical filtering and the motion mapping procedure work on a short temporal window of 20 frames. The system automatically breaks the video stream into the segments at run time, and then sequentially transforms the visual control signals into high-quality motion data segment by segment. Because a small discontinuity might occur at the transition between segments, we introduce some overlap between neighboring segments so that we can move smoothly from one segment to another segment by blending the overlap. In our experiment, the blend interval is set to 5 frames. Two synthesized fragments are blended by fading out one while fading in the other using a sigmoid-like function, $\alpha = 0.5cos(\beta\pi) + 0.5$. Over the transition duration, β moves linearly from 0 to 1. The transitional motion is determined by linearly interpolating similar segments with weights α and $1 - \alpha$.

4.4. Data Structure

Because a large number of the K-nearest-neighbor queries may need to be conducted over the same data set S, the computational cost can be reduced if we preprocess S to create a data structure that allows fast nearest-point search. Many such data structures have been proposed, and we refer the reader to[28] for a more complete reference. However, most of these algorithms assume generic inputs and do not attempt to take advantage of special structures. We present a data structure and efficient K nearest neighbor search technique that takes advantage of the temporal coherence of our query data, that is, neighboring query examples are within a small distance in the high-dimensional space. The K closest point search can be described as follows:

- Construct neighborhood graph G. We collect the set of neighboring frames with the temporal window of W for each frame in the motion capture database and treat them as a data point. Then we compute the standard Euclidean distance $d_x(i, j)$ for every pair i and j of data points in the database. A graph G is defined over all data points by connecting points i and j if they are closer than a specific threshold ε, or if i is one of the K nearest neighbors of j. The edge lengths of G are set to $d_x(i, j)$. The generated graph is used as a data structure for efficient nearest-point queries.
- K-nearest-neighbor search. Rather than search the full database, we consider only the data points that are within a particular distance of the last query because of the temporal coherence in the expression control signals. When a new motion query arrives, we first find the closest example E among the K closest points of the last query in the buffer. To find the K nearest points of the current query motion, the graph is then traversed from E in a best first order by comparing the query with the children, and then following the ones that have a smaller distance to the query. This process terminates when the number of examples exceeds a pre-selected size or the largest distance is higher than a given threshold.

A single search with the database size $|S|$ can be approximately achieved in time O(K), which is independent on the size of database $|S|$ and is much faster than exhaustive search with linear time complexity $O(|S|)$.

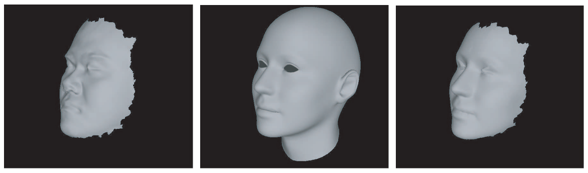

Figure 5: *Dense surface correspondence. (Left) The scanned source surface model. (Middle) The animated surface model. (Right) The morphed model from the source surface to the target surface using the surface correspondence.*

5. Expression Retargeting

The synthesized motion described in the previous section specifies the movement of a finite set of points on the source surface; however, animating a 3D character model requires moving all the vertices on the animated surface. Noh and Neumann introduced an expression cloning technique to map the expression of a source model to the surface of a target model[30]. Their method modifies the magnitudes and directions of the source using the local shape of two models. It produces good results but the run time computation cost depends on the complexity of the animated model because the motion vector of each vertex on the target surface is adapted individually at run time.

In this section, we present an efficient expression retargeting method whose run-time computation is constant independent of the complexity of the character model. The basic idea of our expression retargeting method is to precompute all deformation bases of the target model so that the run-time operation involves only blending together these deformation bases appropriately.

As input to this process, we take the scanned source surface model, the input target surface model, and the deformation bases of the motion capture database. We require both models be in the neutral expression. The system first builds a surface correspondence between the two models and then adapts the deformation bases of the source model to the target model based on the deformation relationship derived from the local surface correspondence. At run time, the system operates on the synthesized motion to generate the weight for each deformation basis. The output animation is created by blending together the target deformation bases using the weights. In particular, the expression retargeting process consists of four stages: motion vector interpolation, dense surface correspondences, motion vector transfer, and target motion synthesis.

Motion Vector Interpolation: Given the deformation vector of the key points on the source surface for each deformation mode, the goal of this step is to deform the remaining vertices on the source surface by linearly interpolating the movement of the key points using barycentric coordinates. First, the system generates a mesh model based on the 3D

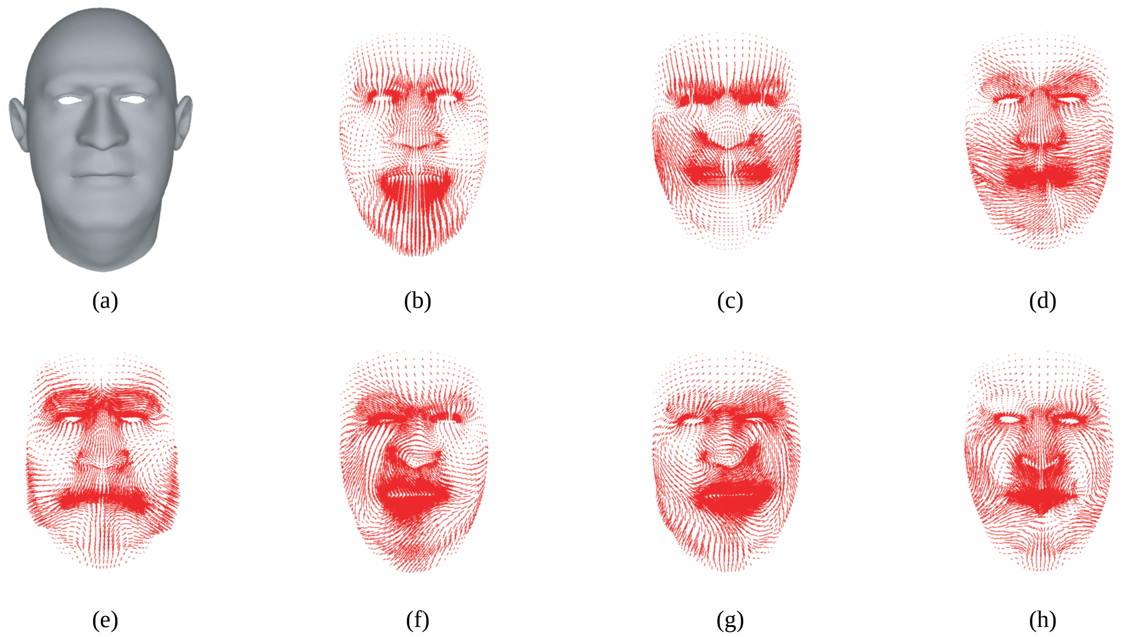

Figure 6: *The top seven deformation bases for a target surface model. (a) The gray mask is the target surface model in the neutral expression. (b-h) The needles show the scale and direction of the 3D deformation vector on each vertex.*

positions of the motion capture markers on the source model. For each vertex, the system determines the face on which the vertex is located and the corresponding barycentric coordinate for interpolation. Then the deformation vector of the remaining vertices is interpolated accordingly.

Dense Surface Correspondences: Beginning with a small set of manually established correspondences between two surfaces, a dense surface correspondence is computed by volume morphing with a Radial Basis Function followed by a cylindrical projection[29] (figure 5). This gives us a homeomorphic mapping (one to one and onto) between the two surfaces. Then we use Radial Basis Functions once again to learn continuous homeomorphic mapping functions $\mathbf{f}(\mathbf{x}_s) = (f_1(\mathbf{x}_s), f_2(\mathbf{x}_s), f_3(\mathbf{x}_s))$ from the neutral expression of the source surface $\mathbf{x}_s = (x_s, y_s, z_s)$ to the neutral expression of the target surface $\mathbf{x}_t = (x_t, y_t, z_t)$, such that $\mathbf{x}_t = \mathbf{f}(\mathbf{x}_s)$.

Motion Vector Transfer: The deformation on the source surface can not simply be transferred to a target model without adjusting the direction and scale of each motion vector, because facial proportions and geometry vary between models. For any point $\mathbf{x}_s$ on the source surface, we can compute the corresponding point $\mathbf{x}_t$ on the target surface by using the mapping function $\mathbf{f}(\mathbf{x}_s)$. Given a small deformation $\delta\mathbf{x}_s = (\delta x_s, \delta y_s, \delta z_s)$ for a source point $\mathbf{x}_s$, the deformation $\delta\mathbf{x}_t$ of its corresponding target point $\mathbf{x}_t$ is computed by the Jacobian matrices $\mathbf{Jf}(\mathbf{x}_s) = (\mathbf{Jf}_1(\mathbf{x}_s), \mathbf{Jf}_2(\mathbf{x}_s), \mathbf{Jf}_3(\mathbf{x}_s))^T$:

$$\delta\mathbf{x}_t = \mathbf{Jf}(\mathbf{x}_s) \cdot \delta\mathbf{x}_s \tag{5}$$

We use the learnt RBF function $\mathbf{f}(\mathbf{x}_s)$ to compute the Jacobian matrix at $\mathbf{x}_s$ numerically:

$$\mathbf{Jf}_i(\mathbf{x}_s) = \begin{pmatrix} \frac{f_i(x_s+\delta x_s, y_s, z_s) - f_i(x_s, y_s, z_s)}{\delta x_s} \\ \frac{f_i(x_s, y_s+\delta y_s, z_s) - f_i(x_s, y_s, z_s)}{\delta y_s} \\ \frac{f_i(x_s, y_s, z_s+\delta z_s) - f_i x_s, y_s, z_s)}{\delta z_s} \end{pmatrix} \quad i = 1,2,3 \tag{6}$$

Geometrically, the Jacobian matrix adjusts the direction and magnitude of the source motion vector according to the local surface correspondence between two models. Because the deformation of the source motion is represented as a linear combination of a set of small deformation bases, the deformation $\delta\mathbf{x}_t$ can be computed as:

$$\begin{aligned} \delta\mathbf{x}_t &= \mathbf{Jf}(\mathbf{x}_s) \cdot \Sigma_{i=1}^{L} \lambda_i \delta\mathbf{x}_{s,i} \\ &= \Sigma_{i=1}^{L} \lambda_i (\mathbf{Jf}(\mathbf{x}_s) \cdot \delta\mathbf{x}_{s,i}) \end{aligned} \tag{7}$$

where $\delta\mathbf{x}_{s,i}$ represents the small deformation corresponding to the *ith* deformation base, and λ_i is the combination weight. Equation (7) shows that we can precompute the deformation bases of the target surface $\delta\mathbf{x}_{t,i}$ such that $\delta\mathbf{x}_{t,i} = \mathbf{Jf}(\mathbf{x}_s) \cdot \delta\mathbf{x}_{s,i}$. Figure 6 shows the top seven deformation bases on a target surface model.

Target Motion Synthesis: After the motion data are synthesized from the motion capture database via a vision-based interface, the system projects them into the deformation basis space of the source model $S_1, ... S_L$ to compute the combination weights $\lambda_1, ..., \lambda_L$. The deformation of the target surface $\delta\mathbf{x}_t$ are generated by blending together the deformation bases of the target surface $\delta\mathbf{x}_{t,i}$ using the combination weights λ_i according to Equation (7).

The target motion synthesis is done online and the other three steps are completed off-line. One strength of our expression retargeting method is speed. The run-time computational cost of our expression retargeting depends only on the number of deformation bases for motion capture database L rather than the number of the vertices on the animated model. In our implementation, the expression retargeting cost is far less than the rendering cost.

6. Results

All of the motion data in our experiments was from one subject and collected as the rate of 120 frames/second. In total, the motion capture database contains about 70000 frames (approximately 10 minutes). We captured the subject performing various sets of facial actions, including the six basic facial expressions: anger, fear, surprise, sadness, joy, and disgust and such other common facial actions as eating, yawning, and snoring. During the capture, the subject was instructed to repeat the same facial action at least 6 times in order to capture the different styles of the same facial action. We also recorded a small amount of motion data related to speaking (about 6000 frames), but the amount of speaking data was not sufficient to cover all the variations of the facial movements related to speaking.

We tested our facial animation system on different users. We use a single video camera to record the facial expression of a live performer. The frame rate of the video camera is about 60 frames/second. The users were not told which facial actions were contained in the motion capture database. We also did not give specific instructions to the users about the kind of expressions they should perform and how they should perform them. The user was instructed to start from the neutral expression under the frontal-parallel view.

At run time, the system is completely automatic except that the positions of the tracking features in the first frame must be initialized. The facial tracking system then runs in real time (about 20 frames/second). Dynamical filtering, expression synthesis, and expression retargeting do not require user intervention and the resulting animation is synthesized in real time. Currently, the system has a 0.33s delay to remove the mapping ambiguity between the visual tracking data and the motion capture data. Figure 7 and Figure 8 show several sample frames from the single-camera video of two subjects and the animated 3D facial expressions of two different virtual characters. The virtual characters exhibit a wide range of facial expressions and capture the detailed muscle movement in untracked areas like the lower cheek. Although the tracked features are noisy and suffer from the typical jitter in vision-based tracking systems, the controlled expression animations are pleasingly smooth and high quality.

7. Discussion

We have demonstrated how a preprocessed motion capture database, in conjunction with a real-time facial tracking system, can be used to create a performance-based facial animation in which a live performer effectively controls the expressions and facial actions of a 3D virtual character. In particular, we have developed an end to end facial animation system for tracking real-time facial movements in video, preprocessing the motion capture database, controlling and animating the facial actions using the preprocessed motion capture database and the extracted animation control parameters, and retargeting the synthesized expressions to a new target model.

We would like to do a user study on the quality of the synthesized animation. We can generate animations both from our system and from "ground truth" motion capture data and then present all segments in random order to users. The subjects will be asked to select the "more natural" animation so that we can evaluate the quality of synthesized animation based on feedback from users. We also would like to test the performance of our system on a larger set of users and character models.

The output animation might not be exactly similar to the facial movement in the video because the synthesized motion is interpolated and adapted from the motion capture data. If the database does not include the exact facial movements in the video, the animation is synthesized by appropriately interpolating the closest motion examples in the database. In addition, the same facial action from a user and an avatar will differ because they generally have different facial geometry and proportions.

One limitation of the current system is that sometimes it loses details of the lip movement. This problem arises because the motion capture database does not include sufficient samples related to speaking. Alternatively, the amount of tracking data in the vision-based interface might not be enough to capture the subtle movement of the lips. With a larger database and more tracking features around the mouth, this limitation might be eliminated.

Currently, our system does not consider speech as an input; however, previous work on speech animation shows there is a good deal of mutual information between vocal and facial gestures[6, 5, 15]. The combination of a speech-interface and a vision-based interface would improve the quality of the final animation and increase the controllability of the facial movements. We would like to record the facial move-

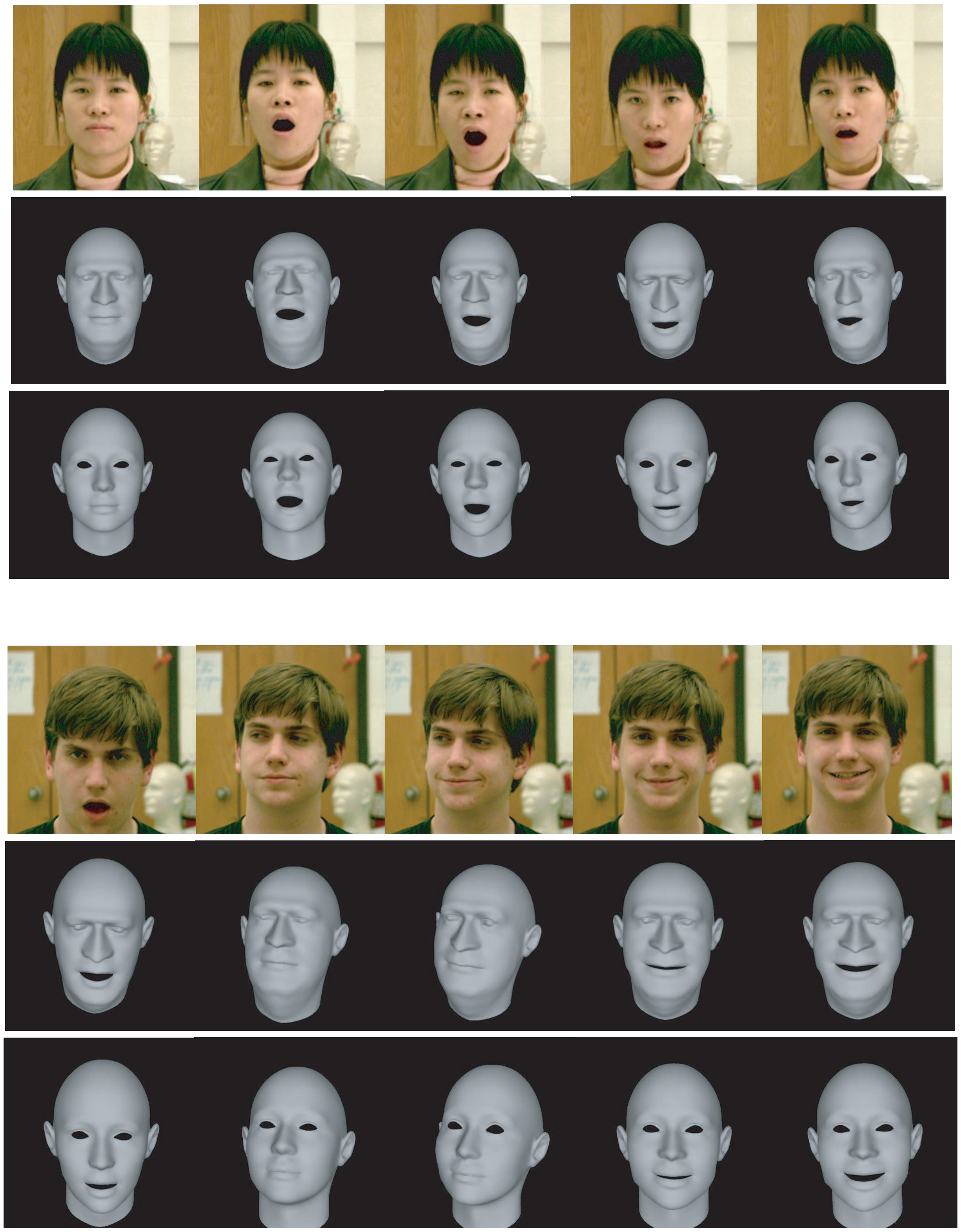

Figure 7: *Results of two users controlling and animating 3D facial expressions of two different target surface models.*

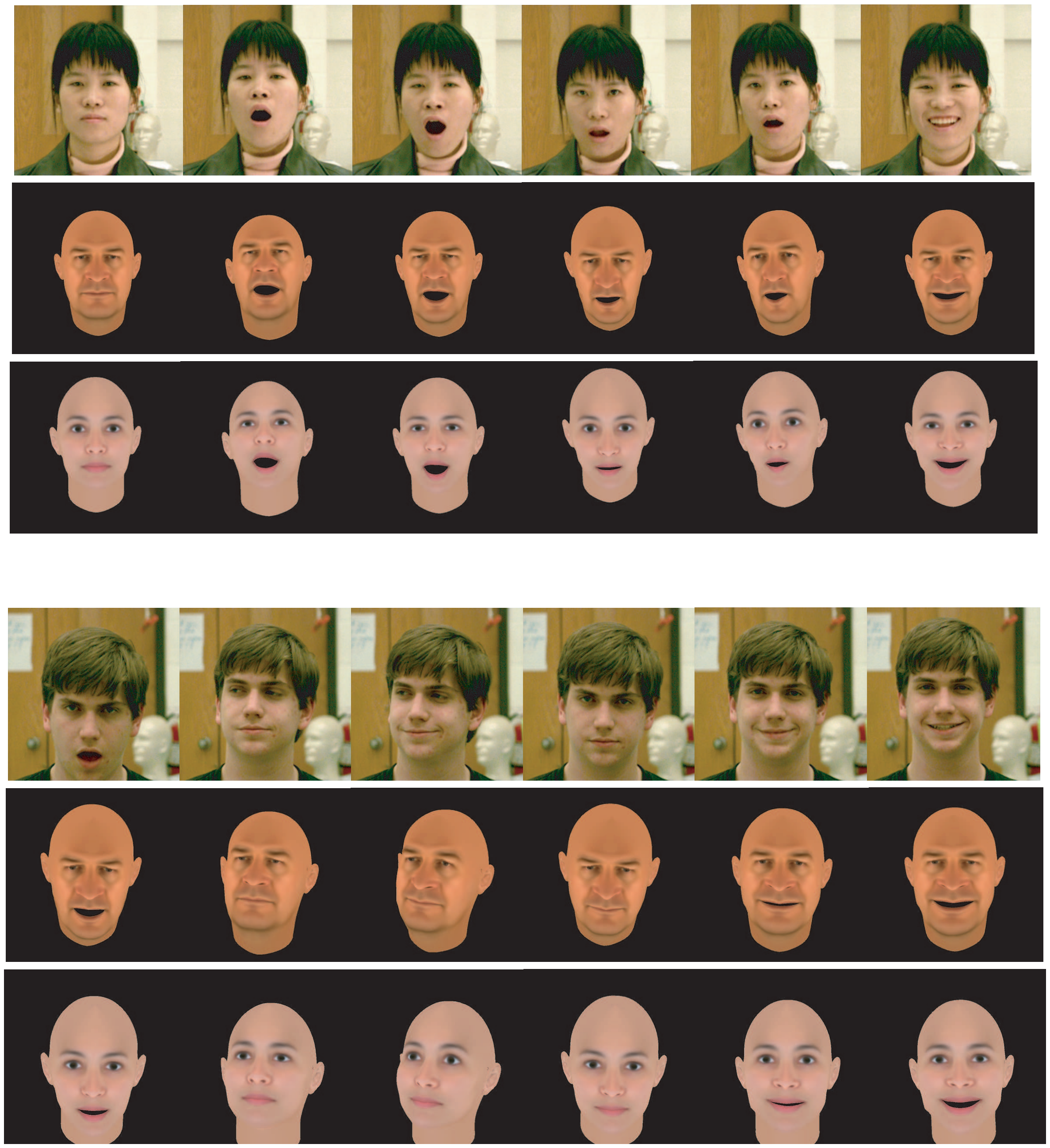

Figure 8: *Results of two users controlling and animating 3D facial expressions of two different texture-mapped avatar models.*

ment and the audio from the motion capture subject simultaneously, and increase the size of the database by capturing more speaking data. We would like to explore how to control the facial movement of an avatar using a multimodal interface, a combination of the vision-based interface and speaking interface.

We are also interested in exploring other intuitive interfaces for data-driven facial animation. For example, a keyframing interpolation system might use the information contained in a motion capture database to add the details and nuances of live motion to the degrees of freedom that are not specified by animator.

Acknowledgements

The authors would like to thank Daniel Huber for his help in scanning and processing the source surface model of the motion capture subject. We also would like to acknowledge the support of the NSF through EIA 0196217 and IIS 0205224.

References

1. S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework part 1: The quantity approximated, the warp update rule, and the gradient descent approximation. In *International Journal of Computer Vision*. 2003.
2. J. R. Bergen, P. Anandan, and K. J. Hanna. Hierarchical model-based motion estimation. In *Proceedings of European Conference on Computer Vision*. 1992. 237-252.
3. M. Black. Robust incremental optical flow. PhD thesis, Yale University, 1992.
4. M.J. Black and Y. Yacoob. Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motion. In *International Conference on Computer Vision*. 1995. 374-381.
5. M.E. Brand. Voice puppetry. In *Proceedings of SIGGRAPH 99*. Computer Graphics Proceedings, Annual Conference Series, 1999. 21-28.
6. C. Bregler, M. Covell, and M. Slaney. Video rewrite: Driving visual speech with audio. In *Proceedings of SIGGRAPH 97*. Computer Graphics Proceedings, Annual Conference Series, 1997. 353-360.
7. I. Buck, A. Finkelstein, C. Jacobs, A. Klein, D. H. Salesin, J. Seims, R. Szeliski, and K. Toyama. Performance-driven hand-drawn animation. In *Symposium on Non Photorealistic Animation and Rendering 2000*. Annecy, France, 2000. 101-108.
8. M. La Cascia and S. Sclaroff. Fast, reliable head tracking under varying illumination. In *IEEE Conf. on Computer Vision and Pattern Recognition*. 1999. 604-610.
9. J. E. Chadwick, D. R. Haumann, and R. E. Parent. Layered construction for deformable animated characters. In *Proceedings of SIGGRAPH 89*. Computer Graphics Proceedings, Annual Conference Series, 1989. 243-252.
10. E. Chuang and C. Bregler. Performance driven facial animation using blendshape interpolation. Computer Science Technical Report, Stanford University, 2002. CS-TR-2002-02.
11. M. N. Cohen and D. W. Massaro. Modeling coarticulation in synthetic visual speech. In *Models and Techniques in Computer Animation*. Springer-Verlag, Tokyo, Japan, 1992. edited by Thalmann N. M. and Thalmann, D.
12. D. DeCarlo and D. Metaxas. Optical flow constraints on deformable models with applications to face tracking. In *International Journal of Computer Vision*. 2000. 38(2):99-127.
13. B. DeGraf. Notes on facial animation. In *SIGGRAPH 89 Tutorial Notes: State of the Art in Facial Animation*. 1989. 10-11.
14. I. Essa, S. Basu, T. Darrell, and A. Pentland. Modeling, tracking and interactive animation of faces and heads using input from video. In *Proceedings of Computer Animation Conference*. 1996. 68-79.
15. T. Ezzat, G. Geiger, and T. Poggio. Trainable videorealistic speech animation. In *ACM Transaction on Graphics (SIGGRAPH 2002)*. San Antonio, Texas, 2002. 21(3):388-398.
16. FaceStation. *URL: http://www.eyematic.com*.
17. Faceworks. *URL: http://www.puppetworks.com*.
18. M. Gleicher and N. Ferrier. Evaluating video-based motion capture. In *Proceedings of Computer Animation 2002*. 2002.
19. S.B. Gokturk, J.Y. Bouguet, and R. Grzeszczuk. A data driven model for monocular face tracking. In *IEEE International conference on computer vision*. 2001. 701-708.
20. B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin. Making faces. In *Proceedings of SIGGRAPH 98*. Computer Graphics Proceedings, Annual Conference Series, 1998. 55-66.
21. N. Howe, M. Leventon, and W. Freeman. Bayesian reconstruction of 3d human motion from single-camera video. In *Neural Information Processing Systems 1999 (NIPS 12)*. 1999.
22. D.F. Huber and M. Hebert. Fully automatic registration of multiple 3d data sets. In *IEEE Computer Society Workshop on Computer Vision Beyond the Visible Spectrum (CVBVS 2001)*. December, 2001. 19:989-1003.

23. P. Kalra, A. Mangili, N. Magnenat Thalmann, and D. Thalmann. Simulation of facial muscle actions based on rational free form deformations. In *Computer Graphics Forum (EUROGRAPHICS '92 Proceedings)*. Cambridge, 1992. 59-69.

24. R. Kjeldsen and J. Hartman. Design issues for vision-based computer interaction systems. Proceedings of the 2001 Workshop on Perceptive User Interfaces, 2001.

25. A. Lanitis, C.J.Taylor, and T.F.Cootes. Automatic interpretation and coding of face images using flexible models. In *IEEE Pattern Analysis and Machine Intelligence*. 1997. 19(7):743-756.

26. Y. Lee, D. Terzopoulos, and K. Waters. Realistic modeling for facial animation. In *Proceedings of SIGGRAPH 95*. Computer Graphics Proceedings, Annual Conference Series, 1995. 55-62.

27. J. P. Lewis, M. Cordner, and N. Fong. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of ACM SIGGRAPH 2000*. Computer Graphics Proceedings, Annual Conference Series, New Orleans, LO, 2000. 165-172.

28. S. Nene and S. Nayar. A simple algorithm for nearest neighbor search in high dimensions. In *IEEE transations on Pattern Analysis and machine Intelligence*. 1997. 19:989-1003.

29. G. M. Nielson. Scattered data modeling. In *IEEE Computer Graphics and Applications*. 1993. 13(1):60-70.

30. J. Noh and U. Neumann. Expression cloning. In *Proceedings of ACM SIGGRAPH 2001*. Computer Graphics Proceedings, Annual Conference Series, Los Angeles CA, 2001. 277-288.

31. F. I. Parke. Computer generated animation of faces. In *Proc. ACM National Conference*, 1972. 1:451-457.

32. F. I. Parke. A parametric model for human faces. Salt Lake City, Utah, 1974. Ph.D thesis, University of Utah. UTEC-CSc-75-047.

33. F. I. Parke. Parameterized models for facial animation revisited. In *ACM Siggraph Facial Animation Tutorial Notes*. 1989. 53-56.

34. F. I. Parke and K. Waters. Computer facial animation. A.K. Peter, Wellesley, MA, 1996.

35. V. Pavlovic, J.M.Rehg, T.-J.Cham, and K.Murphy. A dynamic baysian network approach to figure tracking using learned dynamic models. In *IEEE International Conference on Computer Vision*. 1999. 94-101.

36. F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D. Salesin. Synthesizing realistic facial expressions from photographs. In *Proceedings of SIGGRAPH 98*. Computer Graphics Proceedings, Annual Conference Series, San Antonio, TX, 1998. 75-84.

37. F. Pighin, R. Szeliski, and D. Salesin. Resynthesizing facial animation through 3d model-based tracking. In *International Conference on Computer Vision*. 1999. 143-150.

38. J. Shi and C. Tomasi. Good features to track. In *Proceedins of IEEE Conference on Computer Vision and Pattern Recognition*. 1994. 593-600.

39. H. Sidenbladh, M.J. Black, and L. Sigal. Implicit probabilistic models of human motion for synthesis and tracking. In *European Conference on Computer Vision*. Springer-Verlag, 2002. 784-800.

40. D. Terzopoulos and K. Waters. Physically-based facial modelling, analysis, and animation. In *The Journal of Visualization and Computer Animation*. 1990. 1(2):73-80.

41. D. Terzopoulos and K. Waters. Analysis and synthesis of facial image sequences using physical and anatomical models. In *IEEE Trans. on Pattern Analysis and Machine Intelligence*. 1993. 15:569-579.

42. N. M. Thalmann, N.E. Primeau, and D. Thalmann. Abstract muscle actions procedures for human face animation. In *Visual Computer*. 1988. 3(5):290-297.

43. L. Williams. Performance driven facial animation. In *Proceedings of SIGGRAPH 90*. Computer Graphics Proceedings, Annual Conference Series, 1990. 235-242.

44. J. Xiao and J. Chai. A linear solution to nonrigid structure from motion. In *Robotics Institute Technical Report, Carnegie Mellon University*. 2003. CMU-RI-TR-03-16.

45. J. Xiao, T. Kanade, and J. Cohn. Robust full motion recovery of head by dynamic templates and re-registration techniques. In *Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition*. 2002. 156-162.