Expression Cloning

Jun-yong Noh
noh@usc.edu

Ulrich Neumann
uneumann@usc.edu

Computer Graphics and Immesive Technologies Laboratory
Computer Science Department
Integrated Media Systems Center
University of Southern California

Sample expressions cloned onto Yoda from a model with different geometric proportion and mesh structure
The top row of figure 11 shows the source model.

Abstract

We present a novel approach to producing facial expression animations for new models. Instead of creating new facial animations from scratch for each new model created, we take advantage of existing animation data in the form of vertex motion vectors. Our method allows animations created by any tools or methods to be easily retargeted to new models. We call this process *expression cloning* and it provides a new alternative for creating facial animations for character models. Expression cloning makes it meaningful to compile a high-quality facial animation library since this data can be reused for new models. Our method transfers vertex motion vectors from a source face model to a target model having different geometric proportions and mesh structure (vertex number and connectivity). With the aid of an automated heuristic correspondence search, expression cloning typically requires a user to select fewer than ten points in the model. Cloned expression animations preserve the relative motions, dynamics, and character of the original facial animations.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism – Animation; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling – Geometric Algorithms; I.2.9 [Artificial Intelligence]: Robotics – Kinematics and dynamics

Keywords: Deformations, Facial animation, Morphing, Neural Nets

1 Introduction

Facial animation aims at producing expressive and plausible animations of a 3D face model. Some approaches model the anatomy of the face, deriving facial animations from the physical behaviors of the bone and muscle structures [16, 24, 30, 31]. Others focus only on the surface of the face, using smooth surface deformation mechanisms to create facial expressions [11, 12, 23]. In general, these approaches make little use of existing data for animating a new model. Each time a new model is created for animation, a method-specific tuning is inevitable or the animation is produced from scratch. Animation parameters do not simply transfer between models. If manual tuning or computational costs are high in creating animations for one model, creating similar animations for new models will take similar efforts.

A parametric approach associates the motion of a group of vertices to a specific parameter [22]. This manual association must be repeated for models with different mesh structures. Vector based muscle models place the heuristic muscles under the surface of the face [30, 31]. This process is repeated for each new model and no automatic placement strategy has been reported except for the case where a new model has the same mesh structure. Muscle contraction values are transferable between models only when the involved models are equipped with properly positioned muscles. Even then, problems still arise when muscle structures or surface shapes are inherently different between two models, e.g., a human and a cat face. A three-layer mass-spring-muscle system requires extensive computation [16] and the final computed parameters are only useful for one model. Free-form deformation manipulates control points to create facial expressions [12], but there is no automatic method for mapping the control points from one model to another. Expression synthesis from photographs can capture accurate geometry as well as textures with a painstaking model fitting process for each key frame [23]. In practice, animators often sculpt key-frame facial expressions for every three to five frames to achieve the best quality animations [17]. Obviously, those fitting or sculpting processes must be repeated for a new model even if the desired expression sequences are available for other models.

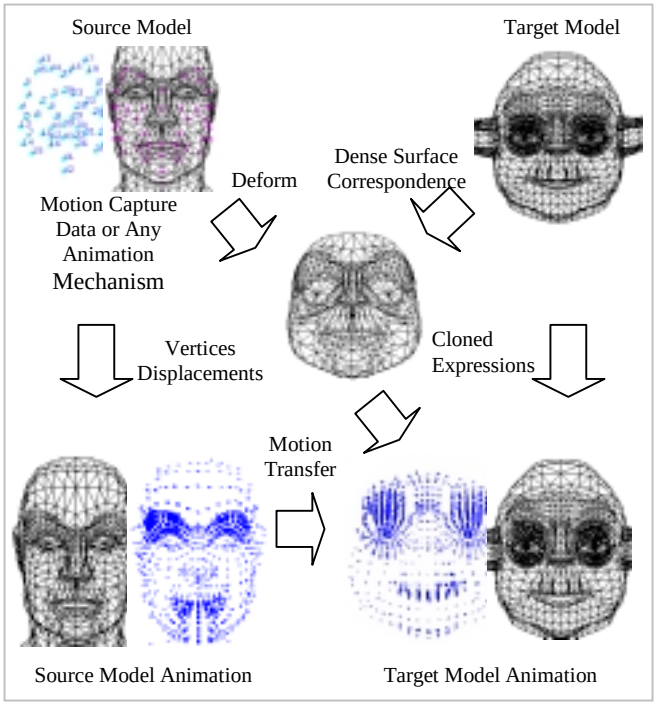


Figure 1 The expression cloning system

Our goal is to produce facial animations by reusing motion data. Once high-quality facial animations are created for any model by any available mechanisms, *expression cloning* (EC) reuses the dense 3D motion vectors of the vertices of the source model to create similar animations on a new target model. Animations of completely new characters can be based on existing libraries of high-quality animations created for many different models. If the animations of the source are smooth and expressive, the animations of the target model will also have the same qualities. Another advantage of EC is the speed of the algorithm; source animations created by computationally intensive physical simulations can be quickly cloned to new target models. After some preprocessing, target model animations are produced in real time, making EC also useful for interactive control of varied target models driven from one generic model, e.g., for text-to-speech applications [21].

Similar to EC, performance driven facial animation (PDFA) and MPEG-4 both use measured motion data [1, 7, 8, 11, 21, 32]. In PDFA, 2D or 3D motion vectors are recovered by tracking a live actor in front of a camera to drive the facial animation. With this approach, the quality of the animation depends on the quality of feature tracking and correspondences between the observed face and target model. MPEG-4 specifies eighty-four feature points. Accurately identifying corresponding feature points is difficult and a daunting manual task. Degraded animation is expected if only a subset of feature points is identified or tracked. In contrast, EC reuses animations already containing precise dense 3D motion data. A sophisticated mechanism identifies dense surface correspondences from a small initial set of correspondences. For models with typical human facial structure, a completely automated correspondence search is described in Section 3.

Expression cloning also relates to 3D metamorphosis research where establishing correspondences between two different shapes is an important issue [13]. Harmonic mapping is a popular approach for recovering dense surface correspondences [4]. Difficulty arises, however, when specific points need to be matched between models. For instance, a naïve harmonic mapping could easily flip the polygons if a user wanted to match the tip of the noses or lip corners between the source and target models. Proposed methods to overcome this problem include partitioning models into smaller regions [13] or model simplification [15] before applying harmonic mapping. A spherical mapping followed by image warping is used in the case of star shaped models [14]. Our approach to finding dense correspondences starts with specific feature matches, followed by a volume morphing and a cylindrical projection.

Our work is also motivated by techniques for retargeting full body animations from one character to another [9]. While we consign the creative decisions (how does a cat smile?) to the user's choice of the source animation as in [9], our technique of cloning a facial animation is significantly different in approach from that dealing with articulated body motions.

In section 2, we detail the methods used to create a cloned expression animation, followed by the heuristic rules to automate the correspondence search in section 3. Implementation specifics and results are shown in section 4. We discuss general issues and possible extensions in section 5.

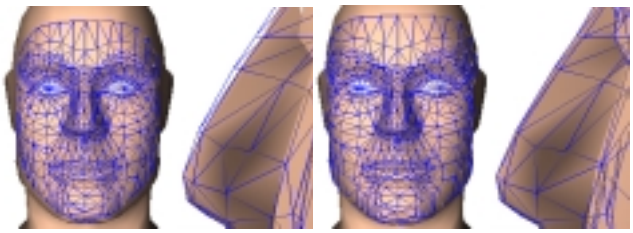
2 Expression Cloning

Expression cloning directly maps an expression of the source model onto the surface of the target model (figure 1). The first step determines which surface points in the target correspond to vertices in the source model. No assumptions are made about the number of vertices or their connectivity in either model. We compute dense correspondences between the models by using a small set of initial correspondences to establish an approximate relationship. Identifying initial correspondences requires manual selection of fewer than ten (possibly zero) vertices after an automated search is applied. Without the automated search, experiments show that fifteen to thirty-five manually selected vertices are sufficient, depending on the shape and the complexity of the model. The automatic correspondence search bootstraps the whole EC process, and heuristic rules are given in section 3.

The second step transfers motion vectors from source model vertices to target model vertices. The magnitude and direction of the transferred motions are properly adjusted to account for the local shape of the model. Using the dense correspondences computed in the first step, motion transfers are well defined by linear interpolation using barycentric coordinates.

2.1 Dense Surface Correspondences

Assuming we have n sparse correspondences, dense surface correspondences are computed by volume morphing with Radial Basis Functions (RBF) followed by a cylindrical projection. Volume morphing roughly aligns features of the two models such as eye sockets, nose ridge, lip corners, and chin points. As shown in figure 2a, volume morphing with a small set of initial correspondences does not produce a perfect surface match. A cylindrical projection of the morphed source model onto the target model ensures that all the source model vertices are truly



(a) After morphing the generic model to itself with 23 initial correspondences, some features are aligned. However, off-surface edges also arise like these blue edges over the nose.

(b) Morphing followed by a cylindrical projection achieves a complete surface match between two models.

Figure 2 Surface correspondences by morphing and projection

embedded in the target model surface , as shown in figure 2b. See figure 12 for more examples.

2.1.1 Radial Basis Functions

The family of radial basis functions (RBF) is well known for its powerful interpolation capability and it is often used for face model fitting [6, 23, 29]. The network of RBF is of the form

$$f(\vec{x}_i) = \sum_{j=1}^{n} w_j h_j(\vec{x}_i) \tag{1}$$

We employ Hardy multi-quadrics for the basis function, $h_j(\vec{x}_i) = \sqrt{\| \vec{x}_i - \vec{x}_j \|^2 + s_j^2}$. The variables w_j denote the weight to be computed, n the number of training inputs, $\vec{x}$ the input vector, and $f(\vec{x})$ the estimated output. The distance s_j is measured between $\vec{x}_j$ and the nearest $\vec{x}_i$, $s_j = \min_{i \neq j} \| \vec{x}_i - \vec{x}_j \|$, leading to smaller deformations for widely scattered feature points and larger deformations for closely located points [5]. This network is trained three times with the 3D coordinates of source correspondences as $\vec{x}_i$, and the x, y, or z values of target correspondences as y_i ($i = 1,2,...n$). Use of a regularization term λ minimizes the cost function

$$C(\vec{w}) = \vec{e}^T \vec{e} + \lambda \vec{w}^T \vec{w} \tag{2}$$

where $\vec{e}$ is the error vector of the difference between the actual value and the estimated value, $\vec{e} = \vec{y} - H\vec{w}$, and $H_{ij} = h_j(\vec{x}_i)$. The regularization parameter is added to avoid overfitting by penalizing large weights. Plugging $\vec{e}$ into equation (2) and differentiating $C(\vec{w})$ with respect to $\vec{w}$ yields

$$\vec{w} = A^{-1} H^T y \tag{3}$$

where $A = H^T H + \lambda I$ and I is the identity matrix.

Generalized cross-validation (GCV) [10], a tool for measuring prediction error, can be differentiated with respect to λ and set to zero to derive the iterative estimation formula for λ [20].

$$GCV = \frac{n \vec{e}^T \vec{e}}{(n-\gamma)^2} \tag{4}$$

$$\lambda = \frac{\eta}{n-\gamma} \frac{\vec{e}^T \vec{e}}{\vec{w}^T A^{-1} \vec{w}} \tag{5}$$

The number of correspondence inputs is n , $\eta = tr(A^{-1} - \lambda A^{-2})$, and γ is the effective number of parameters [19], $\gamma = m - \lambda tr(A^{-1})$. The number of basis functions m is the same as n in our case. Each term in equation (5) can be represented by the eigenvalues u_i , eigenvectors $\vec{u}_i$ of HH^T , and the projection of the target vectors $\vec{y}$ onto the eigenvectors, $z_i = \vec{y}^T \vec{u}_i$ [20].

$$\eta = \sum_{i=1}^{n} \frac{u_i}{(u_i + \lambda)^2} \tag{6}$$

$$\vec{e}^T \vec{e} = \sum_{i=1}^{n} \frac{\lambda^2 z_i^2}{(u_i + \lambda)^2} \tag{7}$$

$$\vec{w}^T A^{-1} \vec{w} = \sum_{i=1}^{n} \frac{u_i z_i^2}{(u_i + \lambda)^3} \tag{8}$$

$$n - \gamma = \sum_{i=1}^{n} \frac{\lambda}{u_i + \lambda} \tag{9}$$

The iteration stops when GCV converges, i.e. the difference between the previous GCV value and the current value becomes less than 0.000001. Once the unknowns are computed, the RBF network smoothly interpolates the remaining non-corresponding points, mapping the source model onto the target model's shape.

2.1.2 Cylindrical Projections

After the RBF deformation, each vertex in the source model is projected onto the target model's surface to ensure a complete surface match. A cylindrical projection centerline is established as a vertical line through the centroid of the head. A ray perpendicular to the projection centerline is passed through each vertex in the source model and intersected with triangles in the target model. The first intersection found is used in cases of multiple valid intersections. Although this could cause a potential problem, visual artifacts are not observed with various models in practice. A reason may be that motions are similar for any of the valid intersections due to their regional proximity.

To test for intersections within a triangle, compute the barycentric coordinates of the intersection point with respect to the vertices of the target triangle. Computing barycentric coordinates is equivalent to solving a 3 x 3 linear system.

$$\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \tag{10}$$

By a property of barycentric coordinate systems, if $0 \leq b_1, b_2, b_3 \leq 1$, then the intersection lies inside the triangle. In reality, because of numerical precision limits, we subtract and add 0.005 from zero and one, respectively.

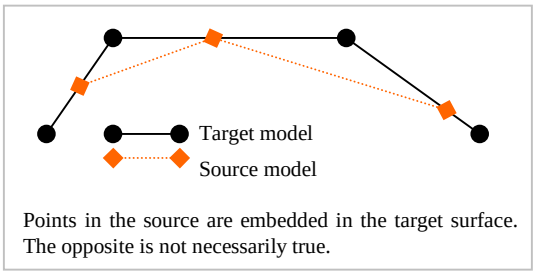


Figure 3 Side view of the two models after the projection

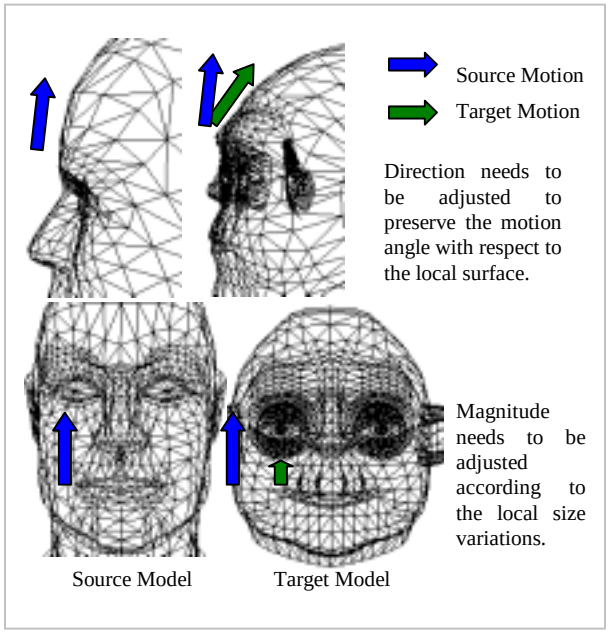


Figure 4 The direction and the magnitude adjustment of motion vectors

2.2 Animation with Motion Vectors

A cloned expression animation displaces each target vertex to match the motion of a corresponding source-model surface point. Since we have dense source motion vectors, linear interpolation with barycentric coordinates is sufficient to determine the motion vectors of the target vertices from the enclosing source triangle vertices.

Note that although the RBF morphing and cylindrical projection embed the source model vertices in the target model surface , the opposite is not necessarily true (figure 3). To obtain the barycentric coordinates needed for motion interpolation, we also project the target model vertices onto the source model triangles. In other words, we do the same operation described in section 2.1.2, but this time reversing the source and target models. The barycentric coordinates of each target vertex determine both the enclosing source model triangle and the motion interpolation coefficients.

Since facial geometry and proportions can vary greatly between models, source motions cannot simply be transferred without

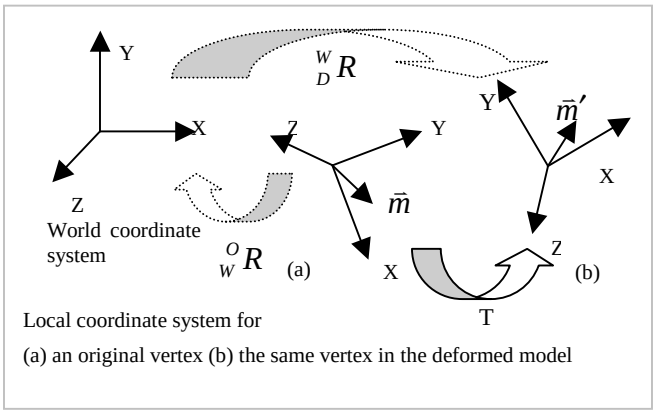


Figure 5 Transformation matrix as a means to adjust a motion vector direction

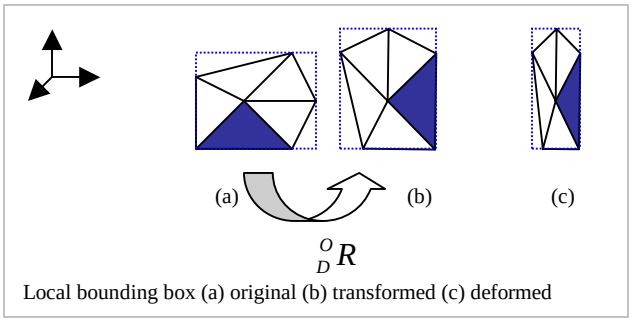


Figure 6 Local bounding box produces scaling factors.

adjusting the direction and magnitude of each motion vector. As shown in figure 4, the direction of a source motion vector must be altered to maintain its angle with the local surface when applied to the target model. Similarly, the magnitude of a motion vector must be scaled by the local size variations. Examples are shown in figure 13.

2.2.1 Motion Vector Direction Adjustment

To facilitate motion vector transfer while preserving the relationship with the local surface, a local coordinate system is attached to each vertex in both the original and deformed source model[1]. The transformation between these local coordinate systems defines the motion vector direction adjustment (figure 5). The local coordinate system is constructed as follows. First, the X-axis is determined by the average of the surface normals of all the polygons sharing a vertex. To ensure continuous normal (X-axis) variations across the surface, a noise filter [25] is applied by averaging neighbor vertex normals. Second, the Y-axis is defined by the projection of any edge connected to the vertex onto the tangent plane whose normal is the just-determined X-axis. Lastly, the Z-axis is the cross product of the X and Y-axes. To obtain the deformed motion vector $\vec{m}'$ for a given source vector $\vec{m}$ (figure 5), the transformation matrices are computed between the two local coordinate systems and the world coordinate system.

[1] A deformed source model is the source model after the morphing and projection described in section 2.1

$$ {}^{O}_{W}R = \begin{bmatrix} \vec{x}_w \bullet \vec{x}_o & \vec{y}_w \bullet \vec{x}_o & \vec{z}_w \bullet \vec{x}_o \\ \vec{x}_w \bullet \vec{y}_o & \vec{y}_w \bullet \vec{y}_o & \vec{z}_w \bullet \vec{y}_o \\ \vec{x}_w \bullet \vec{z}_o & \vec{y}_w \bullet \vec{z}_o & \vec{z}_w \bullet \vec{z}_o \end{bmatrix} \quad (11) $$

$$ {}^{W}_{D}R = \begin{bmatrix} \vec{x}_d \bullet \vec{x}_w & \vec{y}_d \bullet \vec{x}_w & \vec{z}_d \bullet \vec{x}_w \\ \vec{x}_d \bullet \vec{y}_w & \vec{y}_d \bullet \vec{y}_w & \vec{z}_d \bullet \vec{y}_w \\ \vec{x}_d \bullet \vec{z}_w & \vec{y}_d \bullet \vec{z}_w & \vec{z}_d \bullet \vec{z}_w \end{bmatrix} \quad (12) $$

The matrix ${}^{O}_{W}R$ denotes the rotation from a local source vertex coordinate axes to the world coordinate axes, and ${}^{W}_{D}R$ is the rotation from world axes to the local deformed model axes. Prior to the dot product computation in equation (11) and (12), each component denoting the direction of X, Y, and Z-axes is normalized. Finally, the transformation from source to target motion directions is

$$ {}^{O}_{D}R = {}^{W}_{D}R\,{}^{O}_{W}R \quad (13) $$

This mapping at each vertex determines the directions of the deformed source model motion vectors given the source model motion vectors.

2.2.2 Motion Vector Magnitude Adjustment

If the source and target face models have similar proportions, the motion vectors may simply be scaled in proportion to the model sizes. However, to preserve the character of animations for models with large geometry differences (e.g. the unusually big ears of Yoda), the magnitude of each motion vector is adjusted by a local scale factor constrained within a global threshold. Local scale at a vertex is determined by a bounding box (BB) around the polygons sharing the vertex. In deforming a source model to fit a target model, the local geometry around a vertex is often scaled and rotated. Rotations are eliminated to facilitate a fair comparison of local scale. The source BB is transformed by the rotation matrix of equation (13). For each source model vertex in a BB, we compute its rotated position due to model deformation

$$ \vec{v}' = {}^{O}_{D}R\vec{v} \quad (14) $$

The local scale change due to deformation is the ratio of the rotated source BB and the deformed BB (between b and c in figure 6)

$$ \vec{S}_{x,y,z} = \frac{size_{x,y,z}(DeformedSourceModelLocalBoudingBox)}{size_{x,y,z}(SourceModelLocalBoundingBox)} \quad (15) $$

A protrusion or noise in the local geometry (e.g., a bump on the face in either model) can exaggerate motion vector scaling, making the scaling unnecessarily large or small. One solution is to limit scale factors by a global threshold such as the standard deviation of all scale factors. Scale factors greater than the standard deviation are discarded and replaced by the results of a noise filter [25] that averages neighboring values. The filter is then applied over the whole face to ensure smooth continuous scale factors.

The transformation matrix that accounts both for the direction and magnitude adjustments of a motion vector is given by

$$ T = S\,{}^{O}_{D}R \quad (16) $$

where $S = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{bmatrix}$ from equation (15).

During animation, the motion vector for each deformed model vertex is obtained by

$$ \vec{m}' = T\vec{m} \quad (17) $$

where $\vec{m}$ is the vertex motion of the source model and $\vec{m}'$ is the vertex motion of the deformed model. Finally, a vertex in the target model $\vec{v}_t$ is displaced by the following equation

$$ \vec{m}_t = b_1\vec{m}'_1 + b_2\vec{m}'_2 + b_3\vec{m}'_3 \quad (18) $$

where $b_{1,2,3}$ denotes the barycentric coordinates, $\vec{m}_t$ the target vertex motion vector, and $\vec{m}'_{1,2,3}$ the enclosing source triangle motion vectors.

2.3 Lip Contact Line

Our models have lips that touch at a contact line. This contact line between the upper and lower lips requires special attention. Although they are closely positioned, motion directions are usually opposite for upper and lower lip vertices. Severe visual artifacts occur when a vertex belonging to the lower lip happens to be controlled by an upper lip triangle, or vice versa. Therefore, careful alignment of the lip contact lines between the two models is very important. Misalignment results in misidentification of the enclosing triangles and subsequent lip vertex motions in the wrong direction.

Specific processes are followed to produce artifact-free mouth animations. First, include all the source-model lip contact line vertices in the initial correspondence set for the RBF morphing step. Since source vertices do not usually coincide with target vertices (figure 7a), it is necessary to compute corresponding points in the target model. Compute the sum of the piecewise distances between the left and right corners of the lip contact line and normalize each length to the range [0, 1] for both models. Corresponding locations on the target lip-line are found at normalized parameters matching those of the source lip-line vertices. Label each vertex parameter in the lip contact line as $s_{1,2,3...}$ and $t_{1,2,3...}$ for the source and target model, respectively (figure 7). If parameter s_m falls between t_n and t_{n+1}, the corresponding 3D coordinate c on the target lip is interpolated by

$$ c = 3D(t_{n+1})\frac{s_m - t_n}{t_{n+1} - t_n} + 3D(t_n)\frac{t_{n+1} - s_m}{t_{n+1} - t_n} \quad (19) $$

With the above correspondences, the RBF morphing in section 2.1.1 brings the source lip vertices into the target model's surface as shown in figure 7a. Note that there are duplicate vertices at each point – one for the upper lip and one for the lower lip. If we perform the cylindrical projection in section 2.1.2, the duplicate points represented by t_2, t_3, or t_4 in figure 7a will be controlled by upper-lip source-model triangles since these points are located above the source-model lip-contact line. Therefore, another step is necessary to completely align the lip contact lines of the two models. Temporarily move the vertices of the target-model lip contact line onto corresponding source-model lip contact points. These corresponding positions are computed with normalized

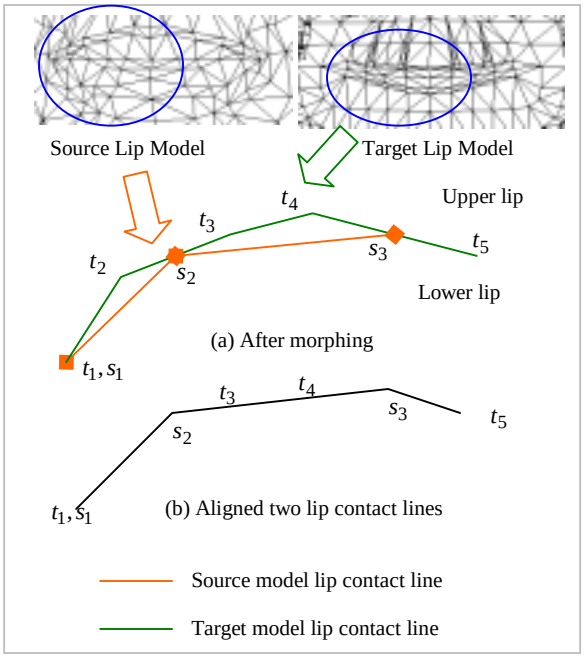


Figure 7 Lip contact line alignment

parameters and equation (19), as before, but this time the target vertices are moved onto the source-lip contact line as opposed to the source vertices moving onto the target-lip contact line. Figure 7b shows final aligned lip lines.

Two issues are noteworthy. First, there is no actual degradation of the fidelity of the target model from aligning its lip-line vertices with the source model. Lip-line alignment is only temporary to facilitate determining the enclosing source-model triangles. The original target-model lip-vertex coordinates are used for animation. Second, by manipulating the contact line vertices for alignment, there may be cases where triangles flip if only the vertices on the lip contact line move. We recursively propagate the same displacements in the contact line neighborhood until no more triangle flipping is detected.

The next step determines which vertex at the lip contact points belongs to the upper and lower lip so that each can be assigned to the appropriate enclosing triangle. A naïve barycentric coordinate test may indicate both the upper and lower-lip triangles as the enclosing triangles for both points on a lip contact line. We check the neighborhood of each vertex to see if neighbor vertices are located above or below the vertex.

Motion-vector transformations also require special attention at the lip contact line. The matrices could easily be different for each of the duplicate vertices at a lip contact point due to their different local neighborhoods. This would cause the two vertices to move to different positions when driven with the same source motion vector. To ensure the same transformation matrices for both vertices on a lip contact point, consider the upper and lower lips connected. Specifically, the normal computations and local BB comparisons include neighbors from the upper and lower lips.

3 Automated Correspondence Selection

A small set of correspondences is needed for the RBF morphing. Since all other EC steps are *fully* automated, automatic initial correspondence selection would completely automate expression cloning. Automatic correspondences not only reduce tedious manual selection, but also remove the errors and variations produced by mouse clicking and judgment. We present fifteen heuristic rules that identify more than twenty correspondences when applied to most human faces. In some cases, we find that up to ten additional manual correspondences may be added to improve the animation quality. In all cases, an animator can simply edit erroneous automatic correspondences, substituting or adding their own selections.

Orient the face model to look in the positive z-direction. The y-axis points through the top of the head, and the x-axis points through the right ear. The model is assumed to have a neutral expression initially with the lips together and the contact line defined by duplicate vertices. For robust behavior during the heuristic correspondence searches, we skip (ignore) degenerate triangles that have one very short edge compared to the other two edges.

Heuristic rules

1. Tip of the nose: Find the vertex with the highest z-value.
2. Top of the head: Find the vertex with the highest y-value.
3. Right side of the face (right ear): Find the vertex with the highest x-value.
4. Left side of the face (left ear): Find the vertex with the lowest x-value.
5. Top of the nose (between two eyes): From the tip of the nose, search upward along the ridge of the nose for the vertex with the local minimum z-value.
6. Left eye socket (near nose): From the top of the nose, search down to the left side of the nose for the vertex with the local minimum z-value.
7. Right eye socket (near nose): From the top of the nose, search down to the right side of the nose for the vertex with the local minimum z-value.
8. Bottom of the nose (top of the furrow): From the tip of the nose, search downward to the center of the lips until reaching the vertex with the local minimum z-value. The vertex with the biggest angle formed by two neighbors is the bottom of the nose.
9. Bottom left of the nose: From the tip of the nose, search downward to the left side of the nose until reaching the vertex with the local minimum z-value. The vertex with the biggest angle formed by two neighbors is the bottom left of the nose.
10. Bottom right of the nose: From the tip of the nose, search downward to the right side of the nose until reaching the vertex with the local minimum z-value. The vertex with the biggest angle formed by two neighbors is the bottom right of the nose.

11. Lip contact line: Find the set of duplicated vertices.
12. Top of the lip: From the center of the upper lip contact line, search upward along the centerline for the vertex with the local maximum z-value.
13. Bottom of the lip: From the center of the lower lip, search downward along the centerline for the vertex with the local minimum z-value after passing the vertex with the local maximum z-value.
14. Chin: From the bottom of the lip, search downward along the centerline for the vertex with the local maximum z-value.
15. Throat: From the chin, search downward along the centerline until reaching the vertex with the local minim z-value. Along the search, find two vertices with two maximum angles. The one with smaller z value is the throat (The other one should be near the chin point).

The labels given to these points may not be precise and they are not important. We only seek to locate corresponding geometric points in both models. Figure 8 shows the correspondences automatically found with the above rules.

4 Results

The specifications of the test models are summarized in table 1. The "source man" model is used as the animation source for all the expressions that are cloned onto the other models. Source animations are created by a) an interactive design system for creating facial animations and b) motion capture data embedded into the source man model (figure 9). An algorithm similar to [11] is implemented to animate the source model with the motion capture data.

For expression cloning onto the woman and man models, only the twenty-three correspondences from the automated search are used. This means that the whole EC process is *fully automated* for these models. The Yoda model has large eyes and ears. We manually add three additional points on each eye socket and two points on each side of the face. The monkey model is handled similarly. The dog and cat model do not have anything close to human face geometry. Twelve and eighteen points are manually selected for the dog and cat, respectively, to replace erroneous automatic search results. Figure 12 shows the deformed source models produced to determine dense surface correspondences from these initial sets of points. The deformations closely approximate each target model. For example, the bumps on the Yoda eyebrows are faithfully reproduced on the deformed source model. The source model cheek is also smoothly bulged for the monkey model. The eyes are properly positioned for the man and woman model. Motion vector adjustments are depicted in figure 13. The monkey model has different local geometry from the source model. Motions are widely distributed (column 5) and more horizontal (column 2) in the mouth region. Finer geometry of the forehead produces denser but smaller motions (column 3).

Figure 11 and 14 show sample expressions from cloned animation sequences. Although the models have different geometric proportions and mesh structures, the expressions are well scaled to fit each model. For instance, the smile and nervous expressions are effectively transferred to the woman model (column 3 and 4 in figure 11). Frown and surprise expressions are shown on the cat model (column 5 and 6). Moderate intensity expressions cause mostly small motions and these are sometimes hardly distinguishable from neutral expressions in static images. Exaggerated expressions are tested in figure 14. A big round open mouth source expression creates a rectangular mouth shape for the monkey due to its much longer lip line. An asymmetric mouth shape is reproduced on the target models and variations arise from differences in the initial target mesh expressions (column 4). The use of human source animations creates many human-like mouth shapes for the dog model rather than expressions more typical of a real dog (last row).

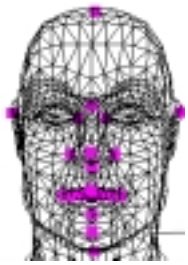
Model	Polygons	Vertices
Source Man	1954	988
Woman	5416	2859
Man	4314	2227
Rick	927	476
Yoda	3740	1945
Cat	5405	2801
Monkey	2334	1227
Dog	927	476
Baby	1253	2300

Table 1 Models used for the experiments

Assessing the emotional quality of the expressions produced by EC is clearly subjective, but we can validate the quantitative accuracy of the algorithm by using the "source man" model as both the source and target model. The EC algorithm is applied to find the surface correspondences and adjust the motion vectors to any local geometry variation. Ideally, the target vertex displacement should be identical to that of the source model. Table 2 and figure 10 show error measures for sample expressions. Staring with the automatically found twenty-three points, an additional ten points are included for this test, three on each eye socket and two on each side of the face. These points produce a more accurate surface match that reduces quantitative errors. The error measure is defined as the size ratio between the position error and the size of the motion vector.

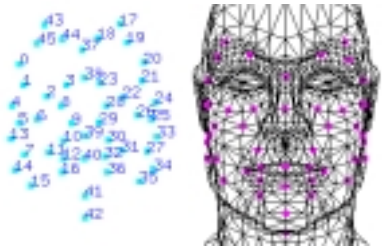
$$\%Error = 100\frac{size(PositionError)}{size(MotionVector)} \tag{20}$$

Figure 10 visually depicts displacement errors such that a vertex with zero error is yellow and a vertex position error one-tenth of its motion vector length (10%) is red. Errors between 0 and 10% are colored by interpolation. Vertices with no motion are colored blue. Figure 10 shows that central face areas where most expression motions occur have small errors and boundary regions generally have higher errors. The larger boundary-area error percentage occurs because motions are relatively small at the boundary, making the denominator in equation (20) small. With very small motions, even numerical errors can adversely affect this error measure. Table 2 shows the average errors of all the vertices with motions. To better quantify the visual significance of the errors, the position error is also measured relative to an absolute reference, in this case the size of the model.



23 automatically found feature points including 9 lip contact points

Figure 8 The automated search results



46 motion-capture data points

Motion capture data embedded into the source man model

Figure 9 The motion capture data and its association with the source model

$$\%Error_{x,y,z} = 100\frac{size_{x,y,z}(PositionError)}{size_{x,y,z}(FaceRegionBoundingBox)} \quad (21)$$

Note that in this case the error is computed separately along the x, y, and z directions. Table 3 indicates that the average errors relative to the size of the model are negligible. Since the motion vectors are dense over the whole face, and their errors are small, visual artifacts are very difficult to perceive, even at high resolutions.

The experiments are performed on a 550 MHz Pentium-III PC. Except for the actual animations, all other processes are performed offline. The automated search takes $O(n)$ to find the tip of the nose, the top of the head, and other extreme points. Once those initial points are found, the search of other points (i.e. the chin) only requires a local search of neighborhood vertices. Therefore, the feature search is fast, taking only a few seconds in our experience.

The RBF morphing involves solving for Eigen systems needed for the regularization parameter and the matrix inversion needed for the weight vectors. The size of the matrix is typically less than 30x30, so the morphing is also fast. A naive cylindrical projection to find the correspondence between n source vertices and m target triangles takes $O(nm)$. Even with this brute-force approach, projection takes less than a minute for our models. This time could be reduced, by using smarter search exploiting, for instance, spatial coherence. Unnecessary tests in the back of the head could be prevented by limiting the search to the frontal face. The transformation matrix to adjust the motion vector magnitude and direction is constructed per vertex, $O(n)$. Finally, the actual animation using already-computed barycentric coordinates is performed in real time (>30Hz) including rendering time.

5 Issues and Extensions

The manual intervention required for expression cloning is minimal, involving at most the selection of a small set of correspondences. We show that correspondences search can be at least partially automated by a heuristic analysis of the geometry. There are some regions, however, for which geometric

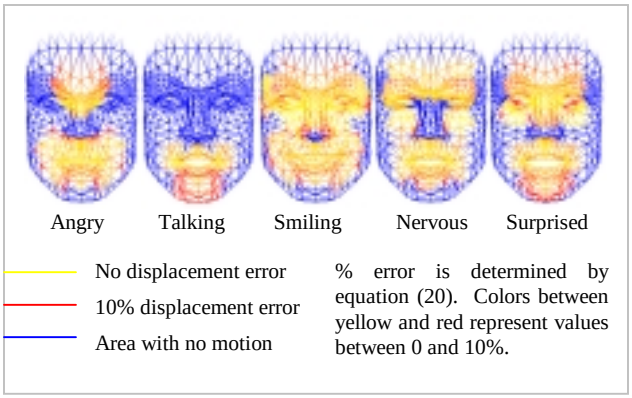


Figure 10 Visually depicted displacement errors

Angry	Talking	Smiling	Nervous	Surprised
5.28%	8.56%	4.77%	4.07%	4.56%

Table 2 Average errors relative to the motion vector size

	Angry	Talking	Smiling	Nervous	Surprised
x	0.22%	0.14%	0.13%	0.14%	0.16%
y	0.18%	0.26%	0.16%	0.11%	0.12%
z	0.09%	0.23%	0.06%	0.05%	0.05%

Table 3 Average errors relative to the model size

descriptions are not practical. For example, locating the boundary of the face and finding detailed eye features appear difficult using only geometry. As an extension, automatic search may be expanded to use textures. Additional rules or methods would help identify a greater set of correspondences [18, 27]. This could further automate facial animation cloning and reduce quantitative errors. The EC method currently transfers only motion vectors, but it seems possible to include color or texture changes as well [8].

Our goal is to easily create quality animations and we assume that dense surface motion vectors are available. However, we also observe that stick figures and cartoons can convey rich expressions from a sparse representation. Future research could explore how sparse source data can become without loss of expressive animation quality. The issue may be addressed by locating the points with the most salient information for conveying the animation while the dense data field is algorithmically decimated. This knowledge may be useful for collecting motion capture data, and at that point EC may also be suitable for applications in compression.

Currently, our efforts are focused on transferring exactly the same expressions from a source to targets. It would be useful to put control knobs that amplify or reduce a certain expression on all or part of a face. The control knobs would directly modulate the sizes of the motion vectors. The expression motions could also be transformed to Fourier space where its coefficients could be

manipulated [2]. It may also be possible to mix the motions of a set of expressions to produce a variety of speech and emotion combinations for any target model. Clearly, the flexibility provided by control knobs could provide varied target animations from just a few source animations.

Tongue and teeth model manipulations are not handled by EC at this point. If the source model includes tongue animation, we believe that the EC technique can generate animations for target tongue models [3, 28]. Similarly, teeth models can be rotated from source animations providing jaw rotation angles or just motion vectors for the teeth. Finally, assuming an eyeball as a separate model, an eyelid could be treated similar to the lip contact line, or eyelids could be rotated if the rotation angle is provided.

6 Conclusion

The concept of expression cloning provides an alternative to creating animations from scratch. We take advantage of the dense 3D data in (possibly painstakingly created) source model animations to produce animations of different models with similar expressions. Cloning can be completely automatic, yet animators can easily alter or add correspondences. Cloning effectively hides unintuitive low-level parameters from animators while allowing high-level control through correspondence selection. To naïve operators, selecting a small number of correspondences is likely to be much more intuitive and easier than dealing with muscles or sculpting. Since EC starts with ground truth data spatially (each frame) and temporally (a sequence of frames), the quality of output animation is very predictable. Because animations use pre-computed barycentric weights and transformations to determine the motion vector of each vertex, the method is fast and produces real time animations.

7 Acknowledgements

This work received funding from DARPA and the Annenberg Center at USC. Funding and research facilities are also provided by the NSF through its ERC funding of the Integrated Media Systems Center. Other support came from Intel, HP, and Motorola. We recognize the contributions to this work from all our colleagues in the USC CGIT laboratory. Special thanks go to Albin Cheenath for model preparations and Doug Fidaleo for video editing. We also appreciate J.P. Lewis for his assistance in providing motion capture data and his many valuable comments.

References

[1] S. Basu, N. Oliver, A. Pentland, 3D Modeling and Tracking of Human Lip Motions, ICCV, 1998, 337-343

[2] A. Bruderlin, L. Williams, Motion Signal Processing, SIGGRAPH 95 Proceedings, 1995, 97-104

[3] M. Cohen, D. Massaro, Modeling Co-articulation in Synthetic Visual Speech. In N. Magnenat-Thalmann, and D. Thalmann Editors, Model and Technique in Computer Animation, 1993, 139–156, Springer-Verlag, Tokyo

[4] M. Eck, T. DeRose, T. Duchamp, Multiresolution Analysis of Arbitrary Meshes, SIGGRAPH 95 Proceedings, 1995, 173-182

[5] M. Eck, Interpolation Methods for Reconstruction of 3D Surfaces from Sequences of Planar Slices, CAD und Computergraphik, Vol. 13, No. 5, Feb. 1991, 109 – 120

[6] R. Enciso, J. Li, D. Fidaleo, T-Y. Kim, J-Y.Noh, U. Neumann, Synthesis of 3D Faces, International Workshop on Digital and Computational Video, 2000

[7] M. Escher, I. Pandzic, N. Thalmann, Facial Deformations for MPEG-4, IEEE Computer Animation, 1998, 56 - 62

[8] D. Fidaleo, J-Y. Noh, T. Kim, R. Enciso, U.Neumann, Classification and Volume Morphing for Performance-Driven Facial Animation, International Workshop on Digital and Computational Video, 2000

[9] M. Gleicher, Retargetting Motion to New Characters, SIGGRAPH 98 Proceedings, 1998, 33 – 42

[10] G.H. Golub, M. Heath, G. Wahba. Generalized Cross-validation as a Method for Choosing a Good Ridge Parameter. Technometrics, 21(2):215-223, 1979

[11] B. Guenter, C. Grimm, D. Wood, H. Malvar, F. Pighin, Making Faces, SIGGRAPH 98 Proceedings, 1998, 55 – 66

[12] P. Kalra, A. Mangili, N. M. Thalmann, D. Thalmann, Simulation of Facial Muscle Actions Based on Rational Free From Deformations, Eurographics 1992, vol. 11(3) 59–69

[13] T. Kanai, H. Suzuki, F. Kimura, Metamorphosis of Arbitrary Triangular Meshes, Computer Graphics and Applications, March 2000, 62-75

[14] J.R. Kent, W.E. Carlson, R.E. Parent, Shape Transformation for Polyhedral Objects, SIGGRAPH 92 Proceedings, 1992, 47-54

[15] A.W. F. Lee, D. Dobkin, W. Sweldens, P. Schroder, Multiresolution Mesh Morphing, SIGGRAPH 99 Proceedings, 1999, 343-350

[16] Y.C. Lee, D. Terzopoulos, K. Waters, Realistic Face Modeling for Animation. SIGGRAPH 95 Proceedings, 1995, 55-62

[17] J.P. Lewis, M. Cordner, N. Fong, Pose Space Deformation: A Unified Approach to Shape Interpolation and Skeleton-Drive Deformation, SIGGRAPH 00 Proceedings, 2000, 165-172

[18] T. Maurer, C. Malsburg, "Tracking and Learning Graphs and Pose in Image Sequences of Faces", ICAFGR 1996, 242-247

[19] J.E. Moody, The Effective Number of Parameters: An Analysis of Generalization and Regularization in Nonlinear Learning Systems, Neural Information Processing Systems 4, 847-854, Morgan Kaufmann, California, 1992

[20] M. J. L. Orr, Optimizing the Widths of RBFs, Fifth Brazilian Symposium on Neural Networks, Brazil, 1998

[21] J. Ostermann, Animation of Synthetic Faces in MPEG-4, IEEE Computer Animation, 1998, 49 – 55

[22] F.I. Parke, Parameterized Models for Facial Animation. IEEE Computer Graphics and Applications, 1982, vol. 2(9) 61 – 68

[23] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, D.H. Salesin, Synthesizing Realistic Facial Expressions from Photographs, SIGGRAPH 98 Proceedings, 1998, 75-84

[24] S. Platt, N. Badler, Animating facial expression. Computer Graphics, 1981, vol. 15(3), 245-252

[25] W. Pratt, Digital Image Processing, Second Edition, A Wiley-Interscience Publication, ISBN 0-471-85766-1, 1991

[26] F. Preparata, M. Shamos, Computational Geometry-An Introduction. Springer-Verlag, New York, 1985

[27] Y. Shinagawa, T.L. Kunii, Unconstrained Automatic Image Matching Using Multiresolution Critical-Point Filters, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 9, 1998, 994 – 1010

[28] M. Stone, Toward a Model of Three-Dimensional Tongue Movement. Journal of Phonetics, 1991, Vol. 19, 309-320

[29] F. Ulgen, A Step Toward Universal Facial Animation via Volume Morphing, 6th IEEE International Workshop on Robot and Human communication, 1997, 358-363

[30] K. Waters, J. Frisbie, A Coordinated Muscle Model for Speech Animation, Graphics Interface, 1995, 163 – 170

[31] K. Waters. A Muscle Model for Animating Three-Dimensional Facial Expression. In Maureen C. Stone, editor, Computer Graphics, SIGGRAPH 87 Proceedings, 1987, vol. 21, 17-24

[32] L. Williams, Performance Driven Facial Animation, SIGGRAPH 90 Proceedings, 1990, 235 – 242

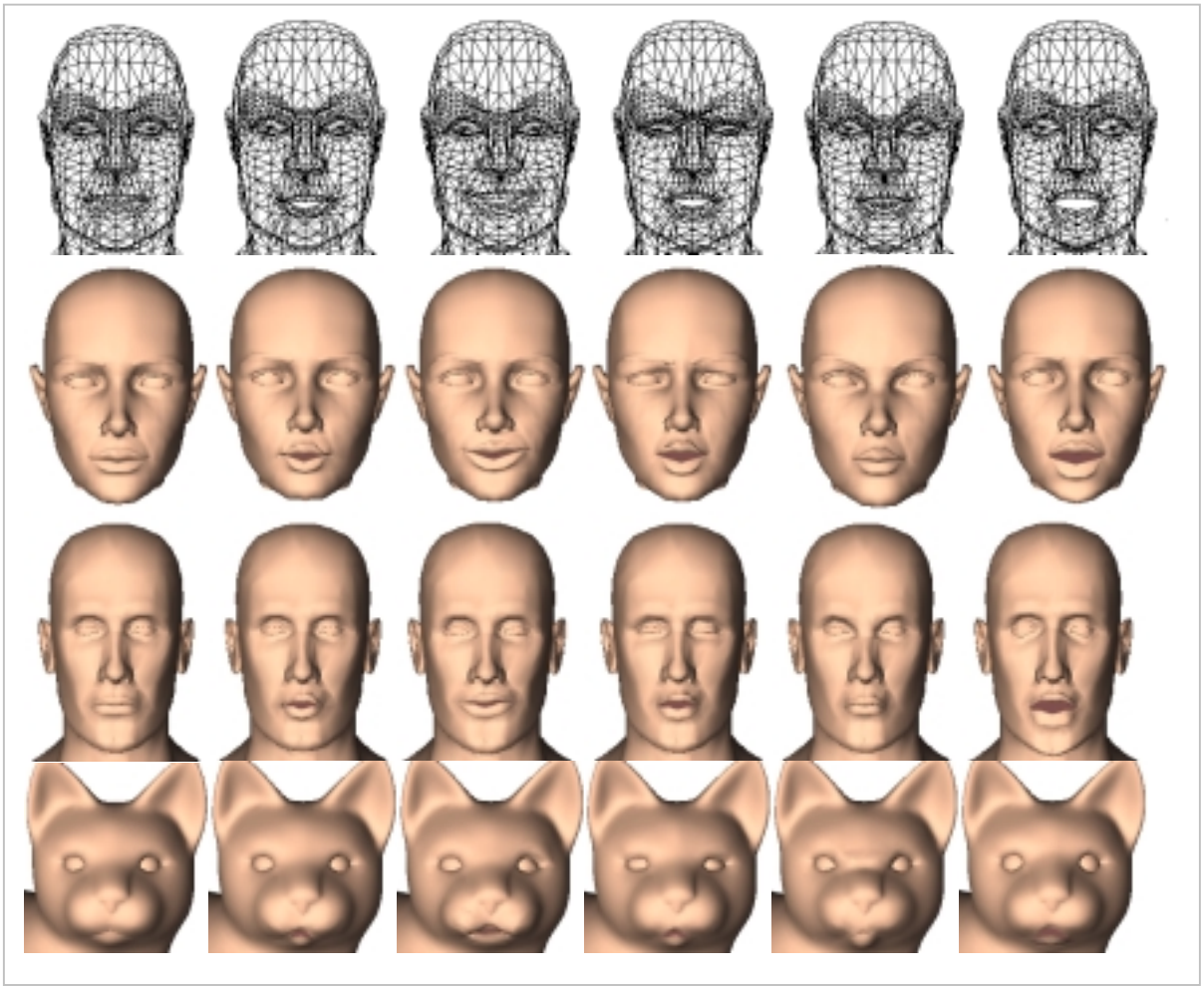


Figure 11 Cloned expressions on various models

First row: The source model and expressions. Rows two, three, and four: Cloned expressions on target models. The target models have different shapes but the expressions are well proportioned to fit each model.

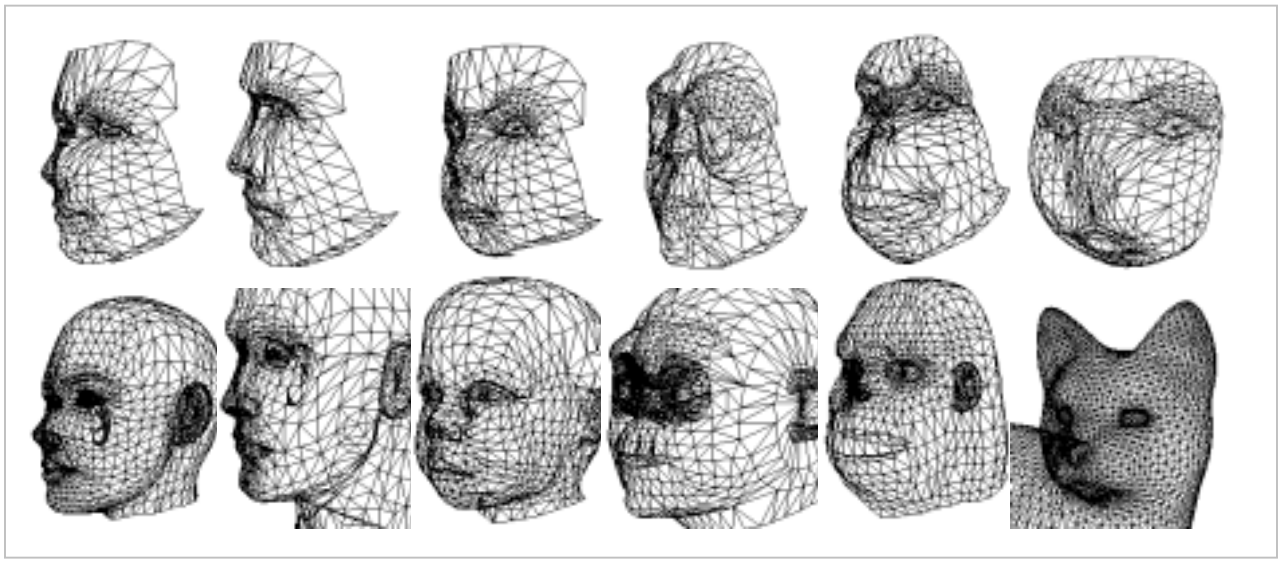


Figure 12 Deformed models produce dense surface correspondences.

First row: The source model after the RBF morphing followed by the cylindrical projection. Second row: Target models. The source model is shown in figure 13. Note that although all the source model vertices are embedded in the target model, different tessellation makes the deformed cat model wireframe appear different from the source. In general, deformed source models closely reproduce the target model features. For example, look at Yoda's eyebrows and mouth.

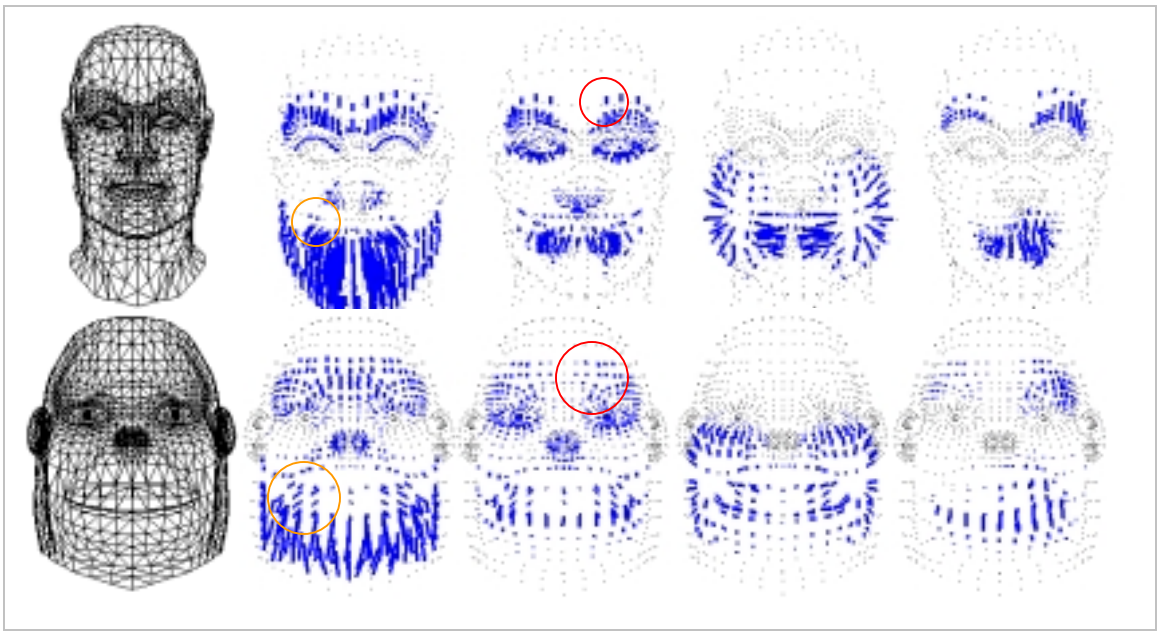


Figure 13 The direction and magnitude adjustments for the motion vector transfer

First row: Source model motions. Second row: Monkey model motions. The left four expressions in figure 14 are used. The monkey's wide and bulged mouth has more horizontal motions compared to the source model (orange circle). Finer geometry of the monkey forehead leads to denser smaller motions (red circle).

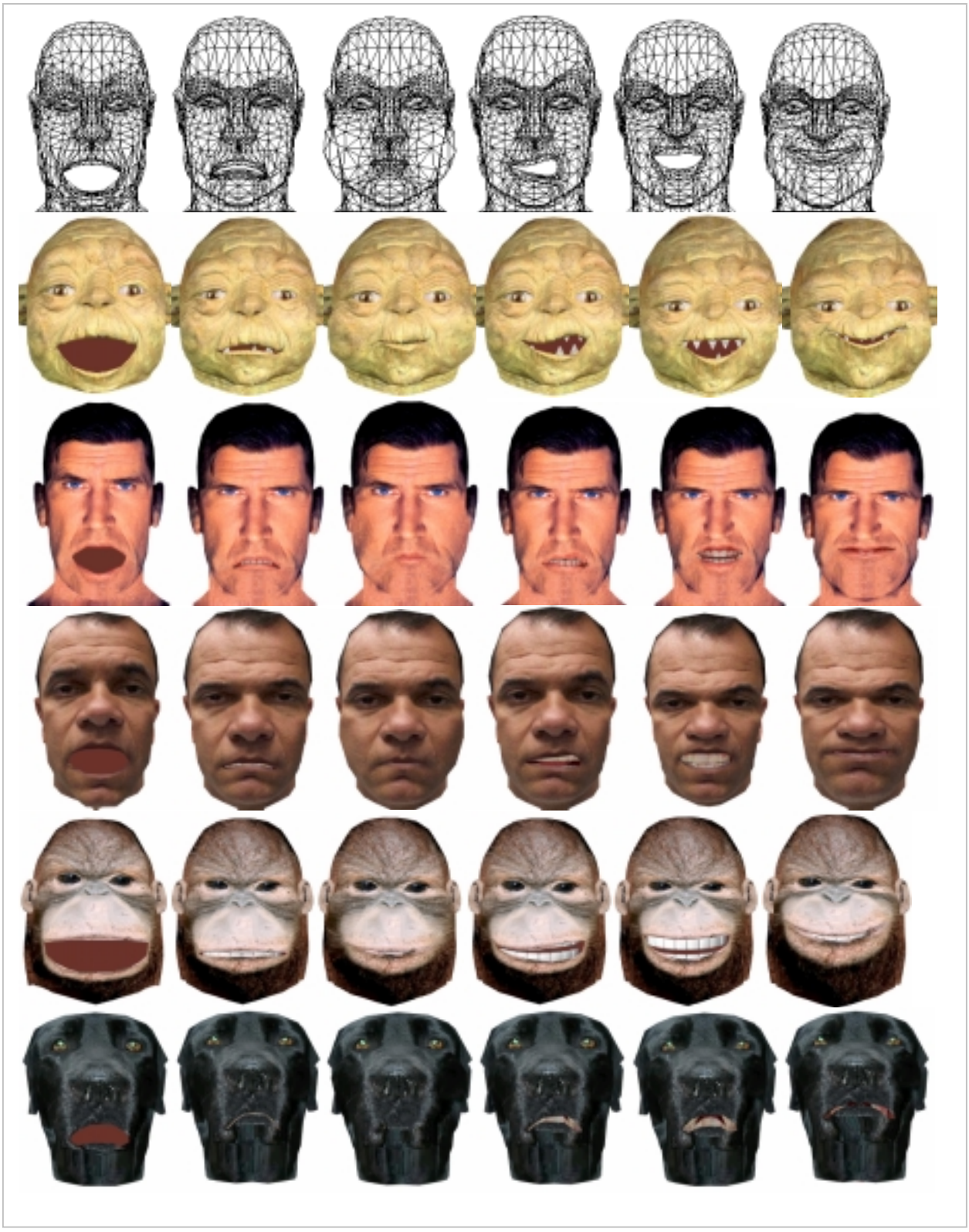


Figure 14 Exaggerated expressions cloned on a wide variety of texture-mapped target models

The Yoda model is provided courtesy of Harry Change, http://Avalon.viewpoint.com.