Performance-Driven Facial Animation

SIGGRAPH 2006

Frederic Pighin — Industrial Light + Magic
J.P. Lewis — Stanford University

George Borshukov — Electronic Arts
Chris Bregler — New York University
Parag Havaldar — Sony Pictures Imageworks
Thomas Kang — Softimage
Jim Radford — Moving Picture Company
Mark Sagar — Weta Digital
Steve Sullivan — Industrial Light + Magic
Tom Tolles — House of Moves
Li Zhang — Columbia University

fpighin a t ilm d o t com

zilla a t computer d o t org

Schedule

8:30	Introduction and Overview Fred Pighin, Industrial Light + Magic
9:00	Facial Motion Capture in Production Parag Havaldar, Sony, and Tom Tolles, House of Moves
10:00	Break
10:15	Facial Retargeting Fred Pighin, ILM, and J.P. Lewis, Stanford
11:15	Markerless Face Capture and Automatic Model Construction Chris Bregler, NYU, and Li Zhang, Columbia
12:15	Lunch
1:30	Performance Driven Facial Animation at ILM Steve Sullivan and Christophe Hery, ILM
2:15	Monster House Parag Havaldar, Sony
3:00	King Kong Mark Sagar, Weta
4:00	Virtual History - Jim Radford, Moving Picture Company, Face Robot - Thomas Kang, Softimage
4:45	Playable Universal Capture at Electronic Arts George Borshukov, EA
5:15	Panel on the future of performance-driven animation all speakers

Contents

Notes

Slide Presentations

Markerless Face Capture and Automatic Model Derivation
Chris Bregler, NYU

Markerless Face Capture using Structured Light
Li Zhang, Columbia

Playable Universal Capture at Electronic Arts
George Borshukov, Electronic Arts

Interactive UCap Sequencing with Leanne Adachi
Jefferson Montgomery, Electronic Arts

Papers

Performance-Driven Facial Animation
L. Williams

Learning Controls for Blend Shape Based Realistic Facial Animation
P. Joshi, W. Tien, M. Desbrun, and F. Pighin

Making Faces
B. Guentery, C. Grimmy, D. Woodz, H. Malvary, F. Pighinz

Synthesizing Realistic Facial Expressions from Photographs
F. Pighin, J. Hecker, D. Lischinski,, R. Szeliski, and D. Salesin

Universal Capture – Image-based Facial Animation for "The Matrix Reloaded"
G. Borshukov, D. Piponi, O. Larsen, J. Lewis, C. Tempelaar-Lietz

Analysis and Synthesis of Facial Expressions with Hand-Generated Muscle Actuation Basis
B. Choe and H.-S. Ko

An Example-Based Approach for Facial Expression Cloning
H. Pyun1, Y. Kim2, W. Chae1, H.-W. Kang1, and S.-Y. Shin1

Face Transfer with Multilinear Models
D. Vlasic, M. Brand, H. Pfister, J. Popovic

Performance-Driven Hand-Drawn Animation
I. Buck, A. Finkelstein, C. Jacobs, A. Klein, D. Salesin, J. Seims, R. Szeliski, K. Toyama

Siggraph 2006 course notes
Performance-driven Facial Animation

Introduction

Frédéric Pighin J.P. Lewis

Industrial Light + Magic Stanford University

Overview

Creating an animation of a realistic and expressive human face is clearly one of greatest challenges in computer graphics. The human face is an extremely complex biomechanical system that is very difficult to model. Human skin has unique reflectance properties that are challenging to simulate accurately. Moreover the face can convey subtle emotions through minute motions. We do not know the control mechanism of these motions.

This course focuses on animating the face, ignoring rendering issues. Although the subject is contentious, some people feel that manual keyframe animation may never capture the subtleties of real facial movement. A more practical viewpoint is that while some animators may be able to produce realistic facial animation, the consistent production of large amounts of flawless animation is expensive and probably not practical.

On the other hand, the ideal "input device" for driving facial animation is probably the face itself – simply mimicking the desired expression is far faster, easier, and more natural than adjusting dozens of sliders. Thus comes the idea of *performance-driven* facial animation: drive the animation directly from a person's captured facial performance.

History

Williams: the genesis of performance driven facial animation. Lance William's paper "Performance-Driven Facial Animation" [13] in Siggraph 1990 introduced the term to the computer graphics community. Although there were a few earlier works on automated face tracking, they generally targeted the idea of model-based video compression in an era when long-distance transmission of video was impractical. In William's groundbreaking demonstration, a static scan of the face was warped by a set of markers tracked via video (with frontal plane X-Y motion only). Each marker had a related "warp field".

Hardware systems. Motion capture systems date from the 1980s or earlier. One of the first prominent facial performance animation demonstrations is that of SimGraphics, shown at Siggraph in 1992. This system used a custom face-tracking helmet with physically attached sensors, rather that passively tracked markers. Only a few gross regions of the face were tracked. Systems such as Acclaim and Ascention capable of tracking up to a hundred markers or so date from the early 1990s, though they were largely used for body rather than face motion capture, however, by the mid- to late- 90s the idea of face tracking with hardware motion capture systems was commonplace and regularly seen in short demonstrations and animations. While current

systems such as the Vicon can track more than 100 markers, it is difficult and time consuming to physically place large numbers of markers on a face, and dense motion capture systems are arguably preferable if very high fidelity is required.

Face tracking approaches

Software based systems offer the promise of moving beyond markers to track all regions of the face at the pixel level. There have been literally hundreds of independent published papers on software face tracking, so we will only mention some of the main approaches. Further details on some of these approaches appear in other sessions of the course, particularly in the session on Dense Motion Capture.

Stereo, photogrammetry, and structured light. Stereo provides potentially the most accurate approach to face tracking. Photogrammetry has been applied to reconstructing accurate animated faces from several cameras [8]. Unguided stereo reconstruction is quite difficult, however, and successful systems have tended to use additional information, such as

- disambiguating markers [8],
- projected structured light patterns [7, 15],
- manual hints and correspondence establishment, and
- manual correction of mistakes [3].

Producing an animated mesh with a stereo technique also requires solving for frame to frame correspondances throughout the shot.

Feature tracking. Early work in computer vision proposed a breakdown in terms of feature-based and appearance-based approaches. Feature based approaches to face tracking would include custom approaches that look for prominant features (the eyes, nose, etc.), whereas appearance based approaches capture the appearance of these regions from training images. Although the appearance-based approach is intuitively appealing, it has not been as successful as other approaches.

Appearance-based tracking. The active appearance model (AAM) [5] (and related independently invented approaches) is the dominant appearance-based face-tracking approach at present.[1] The active appearance model consists of two principal component (PCA) models. The first is trained on variations in face shape (in the case of tracking, typically this means changes in the appearance of a single face as it rotates and changes expression), expressed as the locations of a set of invisible "virtual markers" on the face. The second principal component model is trained on the variations in face skin texture, however, this variation is captured *after* the shape data is identified and removed by morphing the texture to a normalized face shape.

The "gabor-jet" approach introduced by Wiskott, von der Malsburg and collaborators [14] might be considered to lie somewhere in between feature- and appearance- based approaches. This system represents the appearance of important features and their surroundings with a multi-resolution set of Gabor wavelets. The large scale wavelets provide context to help locate and disambiguate specific features. This approach is also used in a commercial face tracking product (Eyematic/NevenVision).

[1] An earlier version of these notes stated that the system developed by Image Metrics uses AAMs; this is incorrect.

Model-based tracking. One of the oldest approaches to face tracking is to construct a 3D model of the subject and adjust its orientation and parameters until it best matches the video frame (in the robotics community this approach is called "visual servo", though the "model" there is a physical robot rather than a CG face). This approach dates to the late 1980s [1], and pioneers of the approach include Li, Roivainen, and Forchheimer in the early 1990s [10] and more recently, DeCarlo and Metaxas [4] and Eisert and collaborators [6].

Automatic model-derivation. Recent research has shown that a deformable model can potentially be derived automatically from only two-dimensional information across an number of frames. Chris Bregler is a pioneer in this area and he surveys this work later in this course.

Performance-driven facial animation in entertainment

Applications of facial motion capture in entertainment have only appeared only in the last several years.

Movie Tests

In 1998 a number of graphics facilities did tests for a potential remake of *The Incredible Mr. Limpet*. Several of these used facial motion capture (the Centropolis test used the Eyetronics sytem, for example), and some of the technology from the PacTitle effort led to the LifeFx system several years later. LifeFx had a groundbreaking short animation in the Siggraph 2000 electronic theater – marking perhaps the first time that a CG actor actually fooled a few people (mainly people at the back of the rather large theater).

In 1999 ILM revealed their *Hugo* test of an alien-like character driven by retargeted performance capture. One writer remarked in 2002 that Hugo was the animated character "that most shakes my faith in the ...impossibility of fabricating synthetic souls" [12]. The technology behind this test has not been published.

In 2002 Disney showed their Human Face Project in the Siggraph Electronic Theater. This test animation showed cross-mapping from an older actor to a CG younger actor.

Combinations of Motion Capture and Traditional Animation

Both *Final Fantasy* (2000) and an early test for *Shrek* chose to use motion capture for the body animation but manual animation for the face, so these are not examples of performance-driven facial animation.

Although the facial motion for Gollum in the *Lord of the Rings* trilogy was done with traditional keyframe animation, it was heavily guided by reference video of the actor Andy Serkis who "played" Gollum. Thus the Gollum character might be considered to be almost "roto (rotoscope)-driven".

Motion Capture arrives at the movies: *The Matrix* sequels and *The Polar Express*

The Matrix sequels used performance-driven virtual actors in a number of special effects shots. Their system [3] applied markerless dense motion capture, but required manual intervention. The system used optical flow in each of 5 HD cameras, and then applied stereo to reconstruct the 3D motion of the face mesh. Because optical flow "drifts" or accumulates errors over time, the system needed to be re-initialized with a manually corrected face mesh at frequent intervals (every few seconds, depending on the amount of motion).

George Borshukov will describe an evolution of this system in his presentation later in the course.

The Polar Express

The Polar Express in 2004 was the breakthrough movie for performance-driven facial animation, with most or all characters being driven by marker-based motion capture [2].

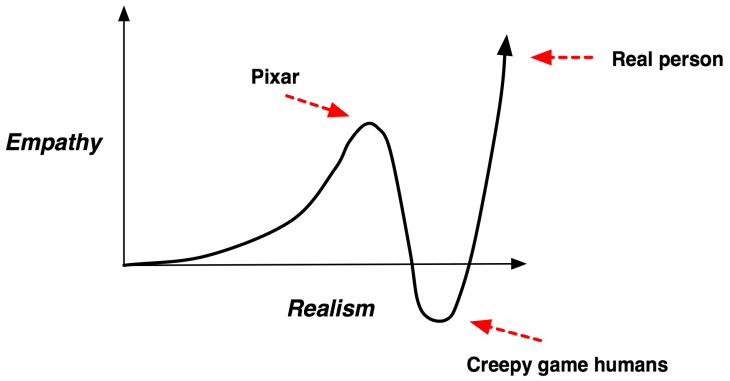


Figure 1: The “uncanney valley”.

Sony Imageworks has further developed their use of motion capture for movies such as *Monster House*, and Parag Haldavar describes the use of a FACS (Facial Action Coding System)-based intermediate representation for retargeting in his session of this course.

The Creepy Factor

Although performance-driven facial animation is now an established and somewhat successful technique, there is even further to go. Current technology is suitable for animated films where the characters have a somewhat “cartoony” feel, but attempts to produce realistic digital clones rarely fool people for more that a few seconds. Moreover, several large scale film industry attempts to produce a CG ”lead” character have been attempted and subsequently abandoned. These include the previously mentioned proposed remake of *The Incredible Mr. Limpet* in 1998 and Disney’s human face attempt in 2000-2002.

Worse still, some CG humans have been described as “creepy”, “the living dead”, etc. This phenomena is now widely described as *the Uncanny Valley* (see Fig. 1), a term introduced roboticist Masahiro Mori in 1970 [11]. Although we can all subjectively identify this effect, it’s cause is a subject of speculation. For example, while several people in an online discussion blamed the lifeless characters in *The Polar Express* on its use of motion capture, the blog author demonstrates (or argues, at least) that *still* images of the characters can be considerably improved with photoshop editing – suggesting that the problem may have nothing to do with motion [9].

References

[1] K. Aizawa, H. Harashima, and T. Saito. Model-based analysis synthesis coding system for a person’s face. *Signal Processing: Image Communication*, 1:139–152, 1989.

[2] David Bennett. The faces of *The Polar Express*. notes in SIGGRAPH course #9: Digital Face Cloning, 2005.

[3] George Borshukov, Dan Piponi, Oystein Larsen, J. P. Lewis, and Christina Tempelaar-Lietz. Universal capture: image-based facial animation for ”the matrix reloaded”. In *Proceedings of the SIGGRAPH 2003 conference on Sketches & applications*, pages 1–1. ACM Press, 2003.

[4] D. Decarlo and D. Metaxas. Deformable model-based shape and motion analysis from images using motion residual error. In *Proceedings, First International Conference on Computer Vision*, pages 113–119, 1998.

[5] G.J. Edwards, C.J. Taylor, and T.F. Cootes. Learning to identify and track faces in image sequences. In *Proceedings, 8th British Machine Vision Conference*, pages 130–139, 1998.

[6] P. Eisert, T. Wiegand, and B. Girod. Model-aided coding: A new approach to incorporate facial animation into motion-compensated video coding, 2000.

[7] EYETRONICS. Shapesnatcher. http://www.eyetronics.com.

[8] B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin. Making faces. In *SIGGRAPH 98 Conference Proceedings*, pages 55–66. ACM SIGGRAPH, July 1998.

[9] Ward Jenkins. *The Polar Express:* a virtual train wreck. `wardomatic.blogspot.com/2004/12/polar-express-virtual-train-wreck_18.html`, 2004.

[10] H. Li, P. Roivainen, and R. Forcheimer. 3-d motion estimation in model-based facial image coding. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(6):545–555, 1993.

[11] Masahiro Mori. On the uncanny valley. In *Proceedings of the Humanoids-2005 workshop: Views of the Uncanny Valley*, December 2005.

[12] Lawrence Weschler. Why is this man smiling? *Wired*, June 2002.

[13] L. Williams. Performance-driven facial animation. In *SIGGRAPH 90 Conference Proceedings*, volume 24, pages 235–242, August 1990.

[14] Laurenz Wiskott, Jean-Marc Fellous, Norbert Krüger, and Christoph von der Malsburg. Face recognition by elastic bunch graph matching. In Gerald Sommer, Kostas Daniilidis, and Josef Pauli, editors, *Proc. 7th Intern. Conf. on Computer Analysis of Images and Patterns, CAIP'97, Kiel*, number 1296, pages 456–463, Heidelberg, 1997. Springer-Verlag.

[15] Li Zhang, Noah Snavely, Brian Curless, and Steven M. Seitz. Spacetime faces: high resolution capture for modeling and animation. *ACM Trans. Graph.*, 23(3):548–558, 2004.

Siggraph 2006 course notes
Performance-driven Facial Animation

Facial Motion Retargeting

Frédéric Pighin J.P. Lewis

Industrial Light + Magic Stanford University

1 Introduction

When done correctly, a digitally recorded facial performance is an accurate measurement of the performer's motions. As such it reflects all the idiosyncrasies of the performer. However, often the digital character that needs to be animated is not a digital replica of the performer. In this case, the decision to use performance capture might be motivated by cost issues, the desire to use a favorite actor regardless of the intended character, or the desire to portray an older, younger, or otherwise altered version of the actor. The many incarnations of Tom Hanks in *Polar Express* illustrate several of these scenarios.

In this scenario, the recorded (source) performance has to be adapted to the target character. This process is called motion retargeting or cross-mapping. In this section, we examine different techniqques for retargeting a recorded facial performance onto a digital face.

The important issues for cross-mapping are the choice of facial model parameterization (or "rig" in the industry parlance) and the nature of the chosen cross mapping (linear or more general).

A rig provides a parameterization of the facial expressions of a digital face. The semantics of the parameters depends on the system. The parameters could be a set of values representing muscle activations (or actuations), or these value could describe the relative contribution of facial expressions. A rig provides a way of describing facial expressions with a small number of parameters and limits the range of expressions to the allowed range of these parameters. In particular, by using a rig for the target character we can solve for a retargeted motion with the rig as a constraint. Using a rig we can make sure that the expressions generated for the target face are valid for this character. Each frame requires solving for the parameters in the rig.

Note that we will not emphasize the distinction between spline and polygon models, because for the most part the cross-mapping algorithms are identical (or the required changes between the two cases will be evident to any experienced graphics person). Similarly, we will also not distinguish between geometric source models (as constructed manually or obtained from a dense motion capture system) and simple collections of points (as provided by hardware motion capture systems such as the Vicon). This is again because the difference between marker points and vertices has little effect on many algorithms, and one format can be transformed into the other by triangulating the markers or considering each vertex in the mesh as a marker. Instead, we underline when necessary when the technique is more appropriate for dense source animation (animated mesh or dense mocap). Lastly, for a couple of the techniques covered ([2, 5]) either the source or target (or both) is video rather than 3D data. Here the distinction is mentioned but not emphasized, because (at least in the

techniques surveyed below) it should be relatively easy to imagine a corresponding algorithm operating on 3D data.

1.1 Parameterization

Blendshape parameterization. Blendshape animation is one of the most widespread facial animation techniques. In blendshape animation, a rig is a set of linearly combined facial expressions each controlled by a scalar weight. The blendshapes provide a linear parameterization of the face deformations. The space of potential faces is the linear space spanned by the blendshapes (or a portion of this space if the weights are bounded). In this framework retargeting is reduced to estimating a set of blending weights for the target face at each frame of the source animation.

There are several approaches to choosing the individual blendshape targets:

- **whole-face** targets, such as a target that represents a smile expression or a particular vowel,
- **delta** targets, where each target is a displacement from the neutral face, and the displacement is typically localized, such as 'raise the right eyebrow',
- **local** blendshapes [12], where separate blendshape systems cover particular regions, typically the upper and lower face,
- linear **motion capture bases** obtained by selecting frames of motion capture, and
- blendshapes derived from **principal component analysis**, as described below.

Although the whole-face, delta, and local approaches all have their adherents, the systems have equivalent expressive power, as can be easily seen by considering the simplicity of converting a model to any of the alternate approaches. Given a delta model, a whole-face model with an identical range motion can be obtained simply by adding the neutral face to each delta target, and likewise, a whole face model can be converted to a delta model by defining and subtracting a "neutral" face from each of the other targets. Similarly, a local model can be converted to a delta model by extending each local target with zeros over the remainder of the whole face – resulting in a less efficient but expressively equivalent model.

Note that one way of automatically obtaining a whole-face basis is to simply select a sufficient set of frames from motion capture; we term this a 'motion capture basis'. This requires some care to obtain targets that both span the full expression space and are not redundant. An approach to this problem developed in [5] will be described later in these notes.

Blendshape models created by principal component analysis are distinctly different from the the whole-face, delta, and local approaches, as will be described next.

PCA parameterization. Principal component analysis (PCA) of motion capture data (dense or marker) automatically produces a blendshape model of sorts. The advantages of a PCA model are that it is the most accurate blendshape model possible with a given number of blendshapes, and that it is obtained automatically. A disadvantage is that while the resulting model has nice mathematical properties (the blendshapes are orthogonal, so fitting is particularly efficient), the model is quite poorly suited for human manipulation. This is because the individual targets resulting from PCA tend to have widespread effects that are difficult to describe and remember. In a production environment, it is often desirable to allow for the possibility of "emergency" human editing of the animation. Thus, PCA models may be ideal as a low-cost and accurate representation of the source performance, but they are perhaps a risky choice for the target. As well, motion capture of the target is often not available by definition.

Raw mesh parameterization. In some cases the source or target may be represented simply as the underlying (polygon or spline) face geometry. In the case of detailed geometry, this representation has many degrees of freedom that are not present in an actual face, and special care is required to avoid invoking these motions .

1.2 Mapping

In general, the retargeting problem can be posed as a function estimation problem where the goal is to create a function [1] that produces a target expression for each source expression. This scattered interpolation formulation encompasses retargeting mappings that are non-linear. In the case where the target face is parameterized by a rig, the function maps source expressions into target parameters. Again, the advantage here is that the target parameters always describe a valid target expression. Without a target rig, a mechanism has to be implemented that guarantees that the retargeting function produces valid target expressions.

For the source face, the issue of the rig is not fundamental since we assume that the source animation is valid. A more important issue for the source is to determine which components of the source animation is affecting the target face. By default, one can assume that every single vertex of the source face affects the target face. In practice this is often not desirable and we might decide to only consider the motion of a subset of the source geometry. This could be done for simplifying the mapping estimation process specially in the case where the target face has fewer degrees of freedom than source face (e.g. a live performer driving a cartoon character). Also, this could help in speeding up the retargeting process.

Linear mapping. A linear mapping is of course the simplest approximation to any mapping, and is a common (though not exclusive) choice for cross mapping. Linear mapping has obvious potential deficiencies, however. Suppose that the target is a dragon that "smiles" by first widening its mouth, and then curling the mouth corners upward. Doing performance-driven animation of a blendshape model using linear cross-mapping cannot produce this subtle non-linear motion, however, it could be obtained by either choosing non-linear mapping technique or using linear mapping in conjunction with a more sophisticated rig that has a built-in control for the desired effect.

Scattered data interpolation. In general, the retargeting problem can be posed as a scattered data interpolation problem, where a function is estimated that maps the source parameter space onto the target parameter face. In this case, the estimation is performed by constraining the function on specific data points in the source space. Igarashi *et al.* [10] call this technique *Spatial Keyframing* in a more general performance-driven animation context. There are many ways of solving scattered data interpolation problems. For instance Buck *et al.* [2] uses partitioning of the target space via a Delaunay triangulation. The mapping is set at each node in the triangulation and linearly interpolated withing each triangle. Partitioning quickly becomes impractical in higher dimensions. Fortunately kernel-based techniques such as Radial basis function (RBF) interpolation [3] provide excellent alternatives. A basic introduction to RBF interpolation is given in the introduction of the course.

Art direction. Regardless of the underlying representation of the retargeting function, it is often desirable to let a user drive the retargeting process. The need for user input is clear when the source and the target character are very different. A user needs to specify how the motions of the source maps into motions for the target. This is often done at the facial expression or muscle activation level depending on the underlying facial parameterization (or rig). The scattered data interpolation framework supports user input quite naturally since the correspondances between source and target expressions can be determined by a user. The need to support

[1] This assumes that only one target expression can be associated to each source expression. This might not be true if for instance the target expression depends as well from the dynamics of the source face.

user input and the corresponding interface depends very much on the application. It could be critical for animating a hero character in a feature film but not necessary for animating chatroom avatars.

2 Survey of Techniques

Parallel Blendshape Models. If we have a set of blendshapes for the performer and one for the target character that correspond to the same expressions, then once the performance is mapped onto the source blendshapes, it can be easily mapped onto the target character by simply reusing the same weights [17]. Unfortunately, this simple situation rarely occurs in practice. Often the blendshapes are not available for the source or they do not correspond to the target blendshapes. Although a custom model representing the source performance could be manually constructed, this can take as much as a year of manual effort to do well [11], which is a lot of effort considering that this model will never be seen or used other than as a representation of the source performance. Fortunately, many of the techniques described below make the manual construction of a source model unnecessary.

Constructing the (or source) by warping). If the target does not differ too much from the source, the unavailable model (whether source or target) can be constructed from the available blendshape model using a simple volume warp such as a free-form deformation. This was one of the early approaches to cross-mapping [16]. This approach is not suitable when the unavailable model has detailed features (such as wrinkles) that cannot be easily introduced with a global warp.

Posed-Correspondence Blendshape Models. In many cases one may have a target blendshape model and a source model (or motion capture basis) with differing controls. In fact, this is probably the first scenario to arise in many cross-mapping efforts: an artist has constructed the target model, and motion capture of the source is available (either in the direct form of a motion capture basis, or through PCA of the motion capture data). One easy approach to this scenario was demonstrated in [14]: given blendshape models with N targets, an animator must simply pose the target model to match at least N distinct frames of the source. The linear mapping from source to target is then simply a high-dimensional coordinate conversion that can be solved by least-squares.

Choe and Ko: refining a generic source model. Choe and Ko [4] approach the problem by assuming that blendshapes are available for the target face. To animate these shapes, they first create a corresponding set of blendshapes (or actuation basis) for the source face. Once this is done, the blending weights can simply be transfered from the source blendshapes to the target blendshapes. The main contribution of their approach is that it refines the source blendshapes as a function of the recorded performance. In this sense, it is tolerant to approximate modeling.

This refinement is performed as follows. This technique starts by manually assigning a location (corresponding point) on the target model for each recorded marker. From these correspondences, a transformation (rigid transformation and scaling) is computed that maps the source coordinate system onto the model coordinate system. This transformation takes care of the difference in orientation and scale between the source and target models. It is estimated on the first frame and applied to all the frames of the performance. The following procedure is then applied for each frame. If a frame in the animation is considered as a vector, it can be written as a linear combination of the corresponding points in the blendshapes where the weights are the blending weights. This provides a set of linear equations where the blending weights are the unknowns. Augmented with a convexity constraint (i.e. all weights have to be non-negative and sum up to one), this system can be solved using quadratic programming. This approach assumes that the source blendshapes can exactly represent the performance, which is generally not true of manually sculpted blendshape models. To address this issue, a geometric correction is performed by solving the same system for the position of the

corresponding points. These two steps (blend weight estimation and geometric correction) are iterated until convergence. Finally, the displacement of the corresponding points are propagated to the model vertices using radial basis functions [3].

This work is presented in a muscle actuation framework where each blendshape corresponds to the actuation of a muscle or muscle group. However, it should equally apply to sets of blendshapes constructed with different philosophies.

Chuang and Bregler: robust mapping from a motion capture basis. The work by Chuang and Bregler [5] starts with a different set of assumptions. The source and target faces do not have blendshapes. They proceed by extracting shapes from the animated source and sculpting corresponding ones for the target face. The source in their technique is video, but similar thinking could be applied in mapping from three-dimensional motion capture.

With this background, they present two ideas that lead to a robust retargeting. First, they show how to choose a basis from among the source frames. After experimenting with several plausible approaches it was found that the best basis (least error in representing the source performance) resulted from taking the source frame point vectors that result in the smallest and largest projection on the leading eigenvectors of the source performance principal component model. Secondly, they point out that reusing the source weights on a target model does not work well when the basis is large and does not exactly represent the source performance. In this situation errors in representing a novel source frame can sometimes be reduced with large counterbalanced positive and negative weights, which results in a poor shape when the weights are reused on the target. Instead, they require the weights to be non-negative. This prevents the previously described situation because negative weights are not available, and results in demonstrably better retargeting even though the error in representing the source is somewhat higher.

This approach has limitations. In particular since the target shapes are derived from the source shapes, using a different performer requires resculpting the target shapes. Also, since the source shapes are extracted using statistical analysis it is unlikely that the target shapes can be reused for correcting the target animation through key-framing.

Kuratate *et al.*: general linear mapping. The work by Kuratate *et al.* [13] assumes that both source and target ranges of facial expressions are defined by a set of scans. The scans are parameterized using PCA and transfer between the two PCA basis is performed using a linear estimator. Non-corresponding PCA shapes are handled using an exchange matrix that allows reordering or blocking off PCA in the estimation of the linear estimator. To handle a more general case, the exchange matrix could be generalized to express the PCA of the target face as linear combinations of the PCA of the source space. An issue with this technique is that the blendshape rigs derived from PCA are very poorly suited for manual editing, so the application must be fully automatic.

Vlasic *et al.*: multi-linear mapping. Freeman and Tenenbaum [8] popularized bilinear (and subsequently multilinear) repesentations in the vision community. While there are several "flavors" of bilinar and multilinear (also called tensor) models, they can generally be considered as extensions to principal component analysis. Whereas PCA captures all the variations in a single set of data, multilinear models can be used to capture independent high-level "meta-dimensions" of the data (not to be confused with the dimesions indentified by PCA). Vlasic *et al.*[20] applied this idea to facial cross-mapping, with one "dimension" capturing the variation in appearance across individuals (between source and target in particular) and a second dimension covering facial expression variation.

Buck *et al.*: local blendshape mapping. Buck *et al.* [2] developed a system for mapping 2D facial motion onto cartoon drawings that uses a piece-wise linear representation for facial expressions. The input motion

is estimated from video by tracking a sparse set of features whose configuration provides a simplified facial expression.

Their system is build on top of a library of cartoon drawings that represent key poses for different facial areas of the target character (e.g. mouth, forehead). These key drawings are blended together, much like a set of blendshapes, to create an animation. Their mapping algorithm relies on associating each key drawing with a particular configuration of the tracked features. This association is then generalized using a scattered data interpolation algorithm. The interpolation is performed using a partition of the space of potential feature configuration (i.e. simplified facial expression). The partition they use is a 2D Delaunay triangulation. To map a frame of input motion, first the triangle that contains that frame is determined; second the barycentric coordinates within the triangles are computed; finally these coordinates are used as blending weights to compute the combination of key drawings. To provide a 2D parameterization of the input space, a Principal Component Analysis is performed on some test data. The two first principal components (maximum variance) determine the reduced dimensionality space.

Tim Hawkins et al. [9] use the same technique to animate facial reflectance fields with a higher dimensional space.

Pyun et. al: radial basis mapping. The work by Pyun *et al.* [18] takes a different approach to retargeting with corresponding blendshapes. Instead of reusing the weights between corresponding shapes, they use an intermediary parameterization of the source motion. A small number of feature vertices are selected on the source mesh. Then the dimensionality of the vector space spanned by these vertices is reduced using PCA, so that each source frame can be projected onto a small set of parameters. The target weights are then estimated from these parameters. The mapping is performed using radial basis functions set to interpolate the corresponding source and target shapes. The advantage of using an intermediate parameterization is to speed-up the mapping algorithm and to add a level of control through the selection of the feature vertices.

Deng *et al.* [6] similarly use a radial basis mapping. Their source model is obtained by PCA of motion capture.

Expression Cloning. Using a blendshape system is not the only way to drive a synthetic face through performance capture. For instance, good results can also be achieved using radial basis functions [15]. Noh and Neumann [22] propose a different approach, called "Expression Cloning", that does not rely on blendshapes. Their technique assumes that the source performance is given as an animated mesh (i.e. the topology and motion curves for every vertex).

The first step of the algorithm is to find geometric correspondences between the source and target models. This is done by computing a sparse set of correspondences that are propagated to the rest of the mesh using scattered data interpolation (using radial basis functions). The sparse correspondences are determined either manually or using some face-specific heuristics.

Once the two models are brought into dense correspondence the motion vectors (offsets from the initial or rest expression) can be transferred. This transfer is performed by assigning a local coordinate system to each vertex in the source and target models. These coordinates systems are determined by the normal of the mesh at that vertex. Transferring a motion vector can then be done by changing local coordinate systems. The motion can also be locally scaled by using the ratio of locally defined bounding boxes in the two models. An additional procedure takes care of the special case of the lip contact line and prevents any spurious interactions between the two lips.

Summer and Popovic: global expression cloning. The work by Summer and Popovic [19] proposes a different solution to the same problem. Where Noh and Neumann [22] estimate local deformations independently at each vertex, Summer and Popovic [19] estimate them using a global optimization. They describe the source motion as a set of per triangle affine deformations from the rest configuration. The motion is transfered by estimating a corresponding set of transformations for the target mesh. Vertex continuity on the target

mesh imposes a set of constraints. The position of the target vertices can be estimated elegantly using a sparse least-squares solver. Although based on a sounder mathematical justification, the global approach adopted by this technique has its drawbacks. In particular, the optimization can amplify small mesh imperfections and noise.

Expression/Style learning. Wang et. al [21] describe an ambitious machine-learning based system for cross-mapping. A data reduction manifold learning technique (local linear embedding, LLE) is first used to derive a mapping from animated expression geometry over time to a one dimensional manifold (curve) embedded in a low-dimensional (e.g. 3D) space. They then establish correspondences between curves for a given expression over different individuals (this includes different monotonic reparameterizations of cumulative length along the curve). Once the correspondences are established, the registered curves are averaged to produce a mean manifold for the particular expression. Evolution of the expression over time now corresponds to movement along this curve.

Next a mapping from the curve back to the facial geometry is constructed. First a mapping from points on the expression curve back to the actor's changing facial geometry is obtained using an approximating variant of radial basis scattered interpolation. This mapping conflates the different "styles" of facial expressions of different people. Lastly, a bilinear decomposition is used to factor the changing expression geometry into components of facial expression and individual identity (thus, for a particular facial expression there is a linear model of how various individuals effect that frozen expression, and for any individual the evolution of the expression over time is also a linear combination of models).

Although the system deals with each facial expression in isolation, it hints at future advances in deriving useful higher level models from data.

Hardware systems Lastly, for historical completeness various hardware-based systems should be mentioned. A prominent example is the SimGraphics "face waldo" face tracking helmet in the early 1990s, which used sensors that were physically attached to a few important regions of the face such as the lips and eyebrows. Since these systems were both proprietary and rather limited compared to current approaches, we will not cover them in depth.

Summary. The source and target model construction and parameterizations and the cross-mapping chosen in each of the above approaches are summarized in the following table. RBF and PCA refer to radial basis function mapping and principal component (analysis) respectively.

approach	source model	mapping	target model
parallel blendshapes	manual blendshape	linear	blendshape
Choe and Ko	adapted linear	linear	blendshape
Chuang and Bregler	selected mocap basis	positive linear	blendshape
Buck *et al.*	PCA on (image) mocap	Delaunay scattered interpolation	local blendshape (image domain)
Kuratate	automatic PCA	linear	automatic PCA
Vlasic *et al.*	scans	multilinear	multilinear blendshape
Pyun et. al	PCA of mocap	radial basis	blendshape
Noh and Neumann	arbitrary	RBF on heuristic correspondences	arbitrary
Summer and Popovic	arbitrary	global optimization	arbitrary
Deng et. al	PCA mocap basis	radial basis	blendshape
Wang et. al	model (driven from structured light data)	linear manifold, RBF	bilinear expression/person basis

Unexplored issues

Expression vs. motion cross-mapping. The different techniques we have described treat the cross-mapping issue as a geometric problem where each frame from the recorded performance is deformed to match the target character. Unfortunately, this might not respect the dynamics of the target character. To go beyond a straight per frame cross-mapping requires an approach that takes timing into account. There are basically two ways this can be tackled: using a physical approach or a data-driven approach.

A physically-based animation system could provide physical constraints for the target face. The cross-mapping algorithm would have to satisfy two types of constraints: matching the source performance but also respecting the physics of the target face. By weighting these constraints an animator could control how much of the source performer versus how much of the target character appears in the final animation.

Any data-driven approach must be carefully designed to minimize the "curse of dimensionality" issues introduced by additional dimensions of timing and expression. One approach might involve building a (small) database of motions for the target character. The performer could then act these same motions to create a corresponding source database. Using machine learning or interpolation techniques these matching motions could provide a time-dependent mapping from the source motion space to the target motion space.

Non-linear rigs. The only rigs that are considered in the research covered in this document are linear or piecewise linear blendshapes. While a model that is automatically derived using PCA can by definition express the desired motion, some experts doubt that the simple blendshape models constructed by human modelers are sufficient to accurately express the subtle motion of real faces [1]. In general, the relationship between the blending weights and the deformed geometry should be non-linear to allow the greatest generality and fidelity, although this consideration must be balanced against criteria that favor simpler models – for example, a non-linear mapping will typically (though not necessarily) have more parameters and if so will require much more data to avoid falling on the variance (overfitting) side of the bias/variance dilemma. A non-linear relationship is clearly required for the motion of the jaw or the motion of the eyelids. Retargeting with such rigs require a different set of techniques.

Facial puppeteering. The human face can express a wide gamut of emotions and expressions that can vary widely both in intensity and meaning. The issue of cross-mapping raises the more general issue of using the human face as an animation input device not only for animating digital faces but any expressive digital object (e.g. a pen character does not have a face). This immediately raises the issue of "mapping" the facial expressions of the performer onto meaningful poses of the target character. Dontcheva *et al.* [7] tackles this issue in the context of mapping body gesture onto articulated character animations.

References

[1] George Borshukov. Face Cloning in the *Matrix* Sequels, notes in SIGGRAPH course #9: Digital Face Cloning, 2005.

[2] Ian Buck, Adam Finkelstein, Charles Jacobs, Allison Klein, David H. Salesin, Joshua Seims, Richard Szeliski, and Kentaro Toyama. Performance-driven hand-drawn animation. In *NPAR 2000 : First International Symposium on Non Photorealistic Animation and Rendering*, pages 101–108, June 2000.

[3] Martin D. Buhmann. *Radial Basis Functions : Theory and Implementations*. Cambridge University Press, 2003.

[4] Byoungwon Choe, Hanook Lee, and Hyeong-Seok Ko. Performance-driven muscle-based facial animation. *The Journal of Visualization and Computer Animation*, 12(2):67–79, May 2001.

[5] E. Chuang and C. Bregler. Performance driven facial animation using blendshapes interpolation. Technical Report CS-TR-2002-02, Stanford University, 2002.

[6] Z. Deng, P.-Y. Chiang, P. Fox, and U. Neumann. Animating blendshape faces by cross-mapping motion capture data. In *ACM I3D*, 2006.

[7] Mira Dontcheva, Gary Yngve, and Zoran Popović. Layered acting for character animation. *ACM Transactions on Graphics*, 22(3):409–416, July 2003.

[8] William T. Freeman and Joshua B. Tenenbaum. Learning bilinear models for two-factor problems in vision. In *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, page 554, Washington, DC, USA, 1997. IEEE Computer Society.

[9] Tim Hawkins, Andreas Wenger, Chris Tchou, Andrew Gardner, Fredrik Göransson, and Paul Debevec. Animatable facial reflectance fields. In *Rendering Techniques 2004: 15th Eurographics Workshop on Rendering*, pages 309–320, June 2004.

[10] T. Igarashi, T. Moscovich, and J.F. Hugues. Spatial keyframing for performance-driven animation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 2005.

[11] Hiroki Itokazu. *personal communication*, Disney Feature Animation, 2001.

[12] Pushkar Joshi, Wen C. Tien, Mathieu Desbrun, and Frdric Pighin. Learning controls for blend shape based realistic facial animation. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 187–192. Eurographics Association, 2003.

[13] T. Kuratate, E. Vatikiotis-Bateson, and H. Yehia. Cross-subject face animation driven by facial motion mapping. In *Proceeedings of 10th ISPE international Conference on Concurrent Engineering: Research and Applications*. Swets and Zeitlinger, 2003.

[14] J. P. Lewis and F. Pighin. Cross-Mapping notes in SIGGRAPH course #9: Digital Face Cloning, 2005.

[15] J. Noh, D. Fidaleo, and U. Neumann. Animated deformations with radial basis functions. In *ACM Symposium on Virtual Realisty Software and Technology*, pages 166–174, 2000.

[16] E. C. Patterson, P. C. Litwinowicz, and N. Greene. Facial animation by spatial mapping. In Nadia Magnenat Thalmann and Daniel Thalmann, editors, *Computer Animation 91*, pages 31–44. Springer-Verlag, Tokyo, 1991.

[17] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D.H. Salesin. Synthesizing realistic facial expressions from photographs. In *SIGGRAPH 98 Conference Proceedings*, pages 75–84. ACM SIGGRAPH, July 1998.

[18] Hyewon Pyun, Yejin Kim, Wonseok Chae, Hyung Woo Kang, and Sung Yong Shin. An example-based approach for facial expression cloning. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 167–176. Eurographics Association, 2003.

[19] R. W. Summer and J. Popovic. Deformation transfer for triangle meshes. In *Proceedings of ACM SIGGRAPH 2004*, 2004.

[20] Daniel Vlasic, Matthew Brand, Hanspeter Pfister, and Jovan Popović. Face transfer with multilinear models. *ACM Trans. Graph.*, 24(3):426–433, 2005.

[21] Y. Wang, X. Huang, C.-S. Lee, S. Zhang, D. Samaras, D. Metaxas, A. Elgammal, and P. Huang. High resolution acquisition, learning, and transfer of dynamic 3-d facial expressions. In *Eurographics*, 2004.

Slides and Notes for SIGGRAPH Course 30: Performance-Driven Facial Animation

Section: Markerless Face Capture and Automatic Model Construction

Chris Bregler chris.bregler@nyu.edu

Latest Version: http://cs.nyu.edu/~bregler/sig-course-06-face/

Slides click here: bregler-slides.ppt

In a way, "Makerless Face Capture" for animation is as old as the trade of traditional animation, taught by Disney and others in the early 20th century. Animators always look at example recordings, especially for facial animations. For example, the idiosyncrasies and mannerisms of the Step-Mother's facial expressions in Cinderella were based on filmed recordings of actress Eleanor Audley. In some cases, the animations were even a direct copy of the hand-traced motions from film, like in scenes of Snow White. Performing such tracing and tracking automatically by a computer has been the topic of computer vision research over the past 20 years. Just recently new techniques have become available that have good enough quality for animation in the entertainment industry.

In this part of the course we will survey these visual tracking techniques and modeling techniques, with a focus on non-rigid or deformable tracking and models. This includes a review of methods that incorporate non-rigid model constraints into optical-flow based methods and appearance/image based techniques, and blend-shape models or PCA models that are trained from example data. In addition, we will focus on a new family of techniques that evolved over the past five years, so-called “rank-constrained” techniques. These techniques combine the tracking and the 3D non-rigid reconstruction problem into one global XYT optimization problem. We will review applications of all these methods to lip-shape, eye-gaze, head-motion, and full-face tracking and modeling.

Snakes / Active Contours:

- http://www.cs.ucla.edu/~dt/papers/ijcv88/ijcv88.pdf
- http://www.robots.ox.ac.uk/~contours/

Optical Flow Basics:

- http://www.ri.cmu.edu/projects/project_515.html
- http://mrl.nyu.edu/~bregler/classes/vision_spring06/shi_tomasi_94.pdf
- http://mrl.nyu.edu/~bregler/classes/vision_spring06/bouget00.pdf

3D Model Based:

- http://www.cs.rutgers.edu/~decarlo/facetrack.html
- Disney's Human Face Project (link TBD)

Appearance Based:

- http://www.face-rec.org/algorithms/
- http://www.isbe.man.ac.uk/~bim/refs.html
- http://www.cs.brown.edu/people/black/appear.html
- http://www.mpi-sb.mpg.de/~blanz/html/projects_01.htm

Factorization Based Face Capture and Modelling:

- http://movement.stanford.edu/nonrig/
- http://people.csail.mit.edu/jovan/assets/papers/vlasic-2005-ftm.pdf
- http://movement.nyu.edu/projects/face.html

Vision Based Face Animation (so far incomplete!):

- http://mrl.nyu.edu/~bregler/videorewrite/
- http://research.microsoft.com/MSRSIGGRAPH/1998/makingfaces.htm
- http://www.cs.washington.edu/homes/pighin/work/face/
- http://www.merl.com/projects/puppets/
- http://cerboli.mit.edu:8000/research/mary101/mary101.html

Slides and Notes for SIGGRAPH Course 30: Performance-Driven Facial Animation

Section: Markerless Face Capture and Automatic Model Construction

Part II: Automatic Model Construction

Li Zhang lizhangATcsDOTcolumbiaDOTedu

Latest Version: http://www.cs.columbia.edu/~lizhang/sig-course-06-face

Slides click here: zhang-slides.ppt

Complementary to the first part on "markerless face catpure" taught by Chris, this part of the section is focused on constructing realistic 3D face models. The constructed models can be used for facial animation directly, or they can be used in the model-based face tracking algorithms.

Specifically, the key technical problem we will be discussing is the fundamental computer vision problem of *3D reconstruction from 2D images*. In particular, we will focus on the methods that will capture highly realistic 3D face models, and more importantly, the way how faces change from one pose to another.

To construct the face models, two important procedures are involved: *reconstructing the shapes* and *computing the motion between shapes*. For shape reconstruction, we will discuss triangulation based methods as well as non triangulation based ones. The former includes laser scanners, structured light sensors, and stereo systems; the latter includes defocus sensors and time-of-flight sensors. For motion estimation, we will discuss marker based approach and template fitting approach.

Laser Scanners:

- http://www.ri.cmu.edu/pubs/pub_2453.html
- http://graphics.stanford.edu/papers/spacetime/
- http://www.mos.t.u-tokyo.ac.jp/~y-oike/research.html

Structured Light Sensors:

- http://metrology.eng.sunysb.edu/holoimage/index.htm
- http://homes.esat.kuleuven.be/~tkoninck/publications/cvpr_sceneadaptedstructuredlight.pdf
- http://grail.cs.washington.edu/projects/moscan/

Stereo Systems:

- http://www.ri.cmu.edu/pubs/pub_1751.html
- http://grail.cs.washington.edu/projects/ststereo/
- http://graphics.stanford.edu/papers/SpacetimeStereo/

Time-of-Flight Sensors:

- http://www.ri.cmu.edu/pubs/pub_2522.html
- http://graphics.stanford.edu/~jedavis/papers/GonzalesBanosDavis_CVPR2004.pdf

Defocus Sensors:

- http://www1.cs.columbia.edu/CAVE/projects/depth_defocus/depth_defocus.php

Marker-based Motion Estimation:

- http://research.microsoft.com/research/pubs/view.aspx?pubid=290

Template-Fitting for Motion Estimation:

- http://grail.cs.washington.edu/projects/stfaces/
- http://www.cs.sunysb.edu/~ial/expressionModeling.html#pubs1

Commercial Systems:

- http://www.cyberware.com/
- http://www.xyzrgb.com/
- http://www.eyetronics.com/
- http://www.3q.com/home.asp
- http://www.3dvsystems.com/
- http://www.canesta.com/
- http://www.swissranger.ch/

SIGGRAPH 2006

Course Notes:
Performance Driven Facial Animation

Parag Havaldar - *havaldar@imageworks.com*

Modeling a face and rendering it in a manner that appears realistic is a hard problem in itself, and remarkable progress to achieve realistic looking faces has been made from a modeling perspective [1, 6, 13, 15, 16, 2] as well as a rendering perspective [5, 11, 12]. At last years Siggraph 2005, the course of Digital Face Cloning described relevant material to this end. An even bigger problem is animating the digital face in a realistic and believable manner that stands up to close scrutiny, where even the slightest incorrectness in the animated performance becomes egregiously unacceptable. While good facial animation (stylized and realistic) can be attempted via traditional key frame techniques by skilled animators, it is complicated and often a time consuming task especially as the desired results approach realistic imagery. When an exact replica of an actor's performance is desired, many processes today work by tracking features on the actor face and using information derived from these tracked features to directly drive the digital character. These features, range from a few marker samples [3], curves or contours [15] on the face and even a deforming surface of the face [2, 16]. This may seem like a one stop process where the derived data of the performance of an act can be made to programmatically translate to animations on a digital CG face. On the contrary, given today's technologies in capture, retargeting and animation, this can turn out to be a rather involved process depending on the quality of data, the exactness/realness required in the final animation, facial calibration and often requires expertise of both artists (trackers, facial riggers, technical animators) and software technology to make the end product happen. Also, setting up a facial pipeline that involves many actors' performances captured simultaneously to ultimately produce hundreds of shots, with the need to embrace inputs and controls from artists/animators can be quite a challenge. This course documents attempts to explain some of the processes that we have understood and by which we have gained experience by working on Columbia's *Monster House* and other motion capture-reliant shows at Sony Pictures Imageworks.

The document is organized by first explaining general ideas on what constitutes a performance in section 1. Section 2 explains how facial performance is captured using motion capture technologies at Imageworks. The next section 3 explains the background research that forms the basis of our facial system at Imageworks – FACS, which was initially devised by Paul Eckman *et al.* Although FACS has been used widely in research and literature [7], at Sony Pictures Imageworks we have used it on motion captured facial data to drive character faces. The following sections 4, 5, 6 explain how motion captured facial data is processed, stabilized, cleaned and finally retargeted onto a digital face. Finally, we conclude with a motivating discussion that relates to artistic versus software problems in driving a digital face with a performance.

1 What Constitutes a Performance?

A performance, in most cases, is understood to be a visual capture of an actor face talking and emoting either individually or in a group with other actors. This is often done by capturing a video performance of the actor and using video frames either purely for reference by an animator or further processing them to extract point samples and even deforming 3D surfaces which are then used to retarget onto a digital character. There are

of course a variety of technological hurdles to overcome before the 2D or 3D reconstructed data can be used – camera calibration, tracking points, and reconstructed 3D information. Apart from video media, other media types such as audio have also been used to capture a voice performance and drive digital faces. Most of the work here [4, 9] approximates the lip and mouth movement of lines of speech but does not have any explicit information that relates to other areas of the face such as brows, eyes and the overall emotion of the character. These have to be either implicitly derived or added on as a post process. Another medium of driving a digital face has been to use facial puppeteering, where a control device such as a cyber glove is used to drive a face. In such cases finger movements are retargeted on to the face.

While all these forms of capture to drive a face have yielded interesting results, the most common by far has been optical data that is used to reconstruct certain facial feature points that then are retargeted onto and drive a digital face.

2 Facial Motion Capture at Imageworks

Imageworks has had a fair amount of success in producing full-length CG animated movies based entirely on motion capture – both body and face. It began with *The Polar Express,* where the Imageworks Imagemotion™ facial and body motion capture acquisition, processing and animation pipeline was setup and successfully tried. The feedback from this experience helped to significantly enhance the pipeline for *Monster House* and now *Beowulf*. Motion capture has also been used to drive faces of digital characters in *Spider-Man 3*. Each productions needs are different, for example in *Monster House*, the motion capture system captured data of body and face together. The facial data had to be ultimately targeted onto characters whose faces were stylized and did not conform to the actual actor faces. For *Beowulf*, the requirements are more geared to producing realistic animation on characters that are supposed to look real and faces perform real. On *Spider-Man 3*, the facial motion captured data was acquired separately in a sit down position and the facial animation generated was blended in key framed body shots. The different types of systems used, the number of people simultaneously captured and facial only vs. face and body capture -- all result in varying data quality, creating many challenges to make data driven facial animation work well. At Imageworks, the artists and engineers are constantly involved to devise a universal system, which can adapt to these different production requirements.

On *Monster House*, face and body was captured simultaneously with two hundred cameras creating a capture volume of 20 feet x 20 feet x 16 feet (length, breadth, height). Eighty infrared markers were used on an actors face to capture the actor's performance. The data was captured and reconstructed in 3D using the multiple-camera system and post processed using, among a variety of tools, Imageworks proprietary Imagemotion technology which is adapted to capturing and processing motion data. On an average, there were about four actors in the volume, with a maximum of six at any given time.

In actual use, during any take, all the actors are made to stand apart in a standard T pose position – a position where the legs are together, hands are stretched out and the face is

relaxed. This position is tremendously useful for a variety of search and standardization during the mocap data processing both for the body and face. Also, each take ends in all the actors returning to a standard T-pose in the volume. The T-pose is used by the facial pipeline in the normalization process to ensure that marker placements are made to correspond (as far as possible) to those on the day of the calibration (master T-pose). More on this is explained in section 4 and illustrated in Figure 1 below.

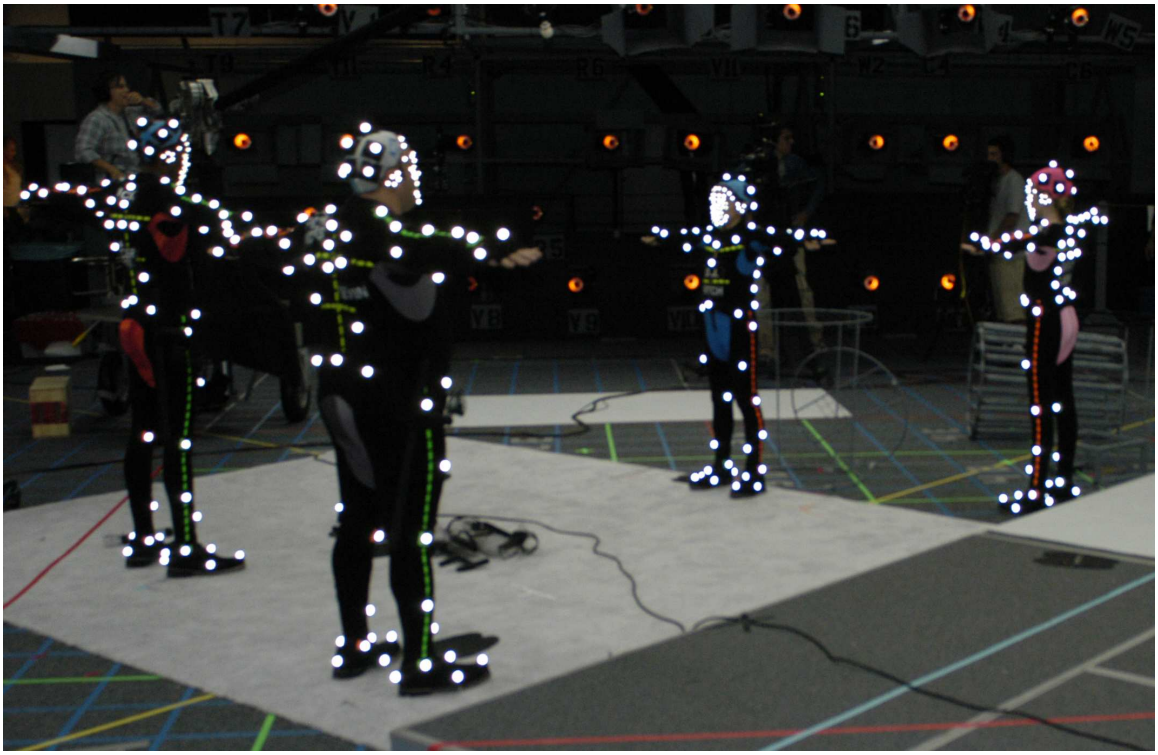


Figure 1. Motion capturing actors on "Monster House." Each actor starts the take and also ends the take in a standard T-pose position with a relaxed face. This neutral facial pose is used for normalizing the day-to-day changes in marker placements.

3 Facial Action Coding System (FACS)

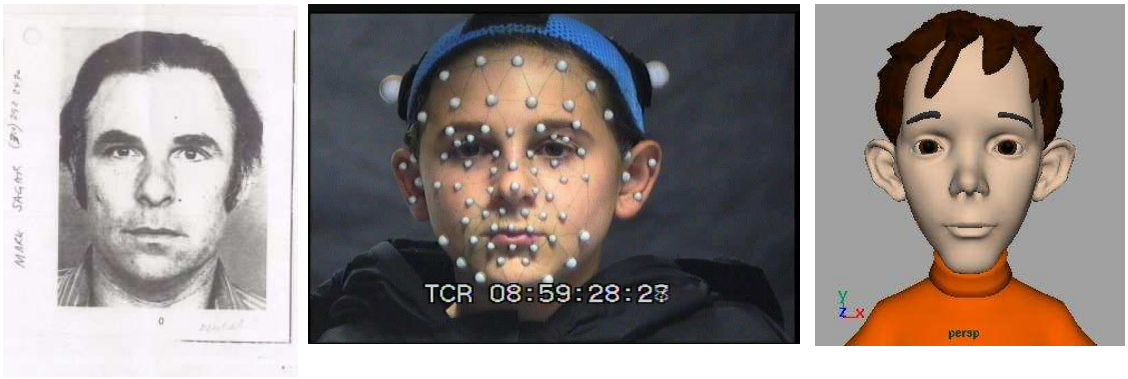
The Facial Action Coding System or FACS was originally designed in the 1970s by Paul Eckman, Wallace Friesen and Joseph Hager [7]. Given that the face has muscles that work together in groups called Actions Units (AUs), FACS teaches how to understand when these action units are triggered and give a relative score to them. Although initially designed for psychologists and behavioral scientists to understand facial expressiveness and behavior, it has also been recently adapted in visual communication, teleconferencing, computer graphics and CG animation [8, 15].

Eckman *et al.* [7], categorized facial expressions into 72 distinct Action Units. Each Action Unit represents muscular activity that produces momentary changes in facial appearance. This change in facial appearance of course varies from person to person

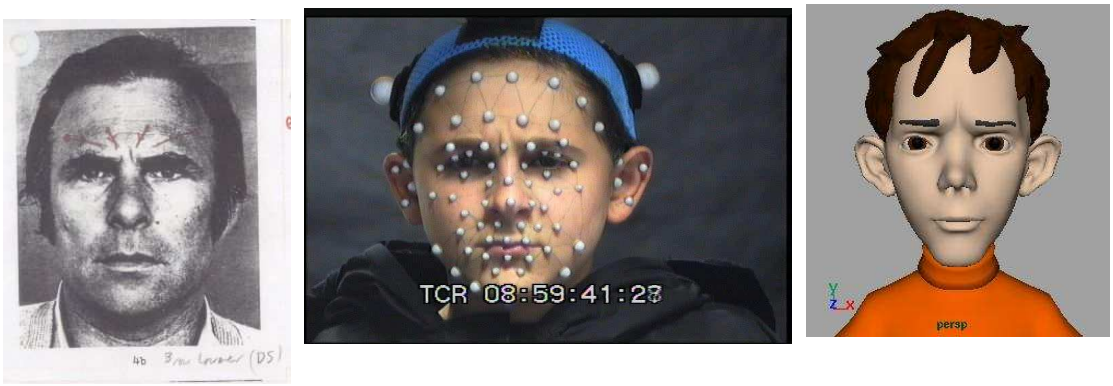
depending on facial anatomy, e.g., bone structure, fatty deposits, wrinkles, shape of features etc. However, certain commonalities can be seen across people as these action units are triggered. The action units in FACS are based on the location on the face and the type of facial action involved. For example, the upper face has muscles that affect the eyebrows, forehead, and eyelids; the lower muscles around the mouth and lips form another group. Each of these muscles works in groups to form action units. The action units can further be broken down into left and right areas of the face, which can get triggered asymmetrically and independently of each other. In general, all the action units suggested by FACS give a broad basis for providing a good foundation for dynamic facial expressions that can be used in CG animation.

On *Monster House*, the facial system made use of FACS as a foundation to base the capture and retarget motion captured data on the characters' faces. Prior to acting, each actor went through a calibration phase where the actor is made to perform all the action units. The 3D reconstructed facial pose data that corresponded to one-action unit captures the extreme positions of how the actors perform a certain action. We used 64 poses, some of which were split into left and right positions along with 18 phoneme poses which described how actors articulated a phoneme. A complete list of these 64 facial expressions is provided in reference [7]. A few of the expressions are described below. In each case, the actual FACS reference, the actor's performance, and the retargeted expression on the character are shown.

Neutral Pose



Brow Lowerer Pose



Lip Corner Puller Pose

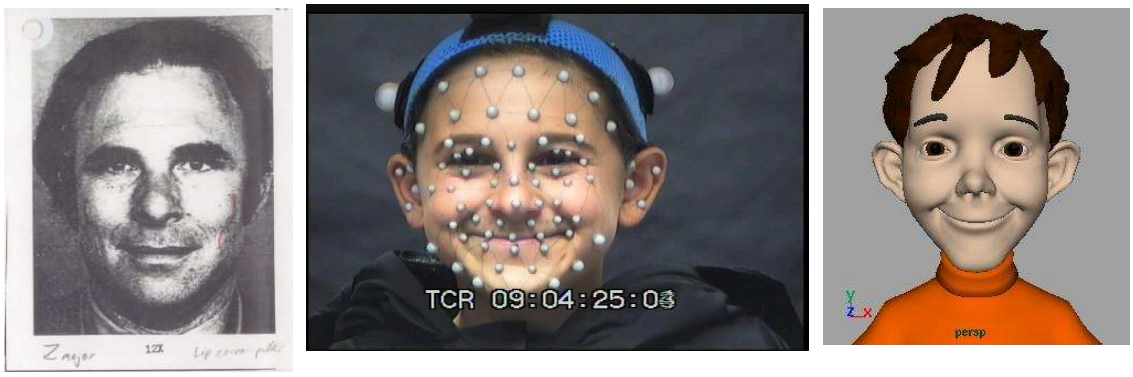


Figure 2. Example poses in a FACS based facial system. The reference action unit image is shown on the left, the actor's performance of the pose is shown in the middle and the artistic interpretation on the character is shown on the right.

Mouth Stretch Pose

Lip Stretcher Pose

Figure 3. More Example poses in a FACS based facial system. The mouth stretch and lip stretcher pose are shown. On "Monster House," the facial system used a total of 64 poses including 18 phoneme positions.

4 Data Capture and Processing

Data capture on *Monster House* was performed using an optical system with 200 infrared cameras capturing both body and face together. The capture volume was 20x20x16 feet and, on average, the performance of four actors was simultaneously captured. The number of facial markers used was 80. The capture system is a passive optical system that uses infrared cameras to capture infrared light reflected by the markers. The image from such cameras is a low entropy image consisting of mostly black areas where no infrared light was seen, with a few white dots whose size in the image varies depending on body/face marker, distance of the actor from the camera, occlusion caused by self and other actors, etc. The low entropy images have two main advantages:

- Due to low entropy, the cameras can capture and record images at higher definitions and at higher frame rates, typically at 60 Hz
- The 3D reconstruction problem triangulates points from multiple images at different viewpoints to get the marker location in space and needs to associate point correspondences automatically from the images. This correspondence problem in general is difficult to resolve, but is greatly improved by having only white dots to work with.

After 3D reconstruction, we now have (x,y,z) positions for markers in frames. However the data is bound to be noisy, does not have temporal associativity (i.e., consistent labeling) across all the frames, and may have gaps. A few typical data frames for the facial markers are shown in the Figure 4 below. These problems are sorted by using a learning based approach that learns both from a facial data model as well as the temporal quality of the data.

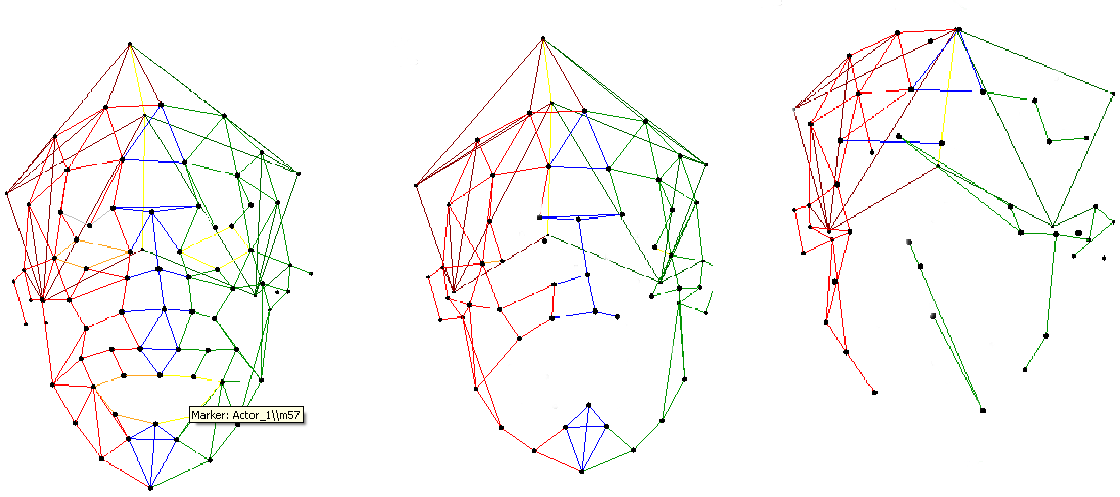


Figure 4. Facial data quality. On a large volume, multiple-person capture stage, the facial data capture quality varies depending on the number of people, self and inter-actor occlusions, etc. Good quality data is shown on the left, but at times this may significantly degrade as shown on the right.

4.1 Segmenting and Labeling Facial Data

The markers reconstructed for each frame have both body markers and face markers. Prior to any processing of facial data, the markers need to be labeled. Labelling is a process by which 3D reconstructed markers in all frames are consistently marked to have the same label. Labeling all markers (face and body) is normally based on trajectories can prove to be a very cumbersome and prone to error, especially with the number of markers visible in the volume and frequently required manual labor. We accomplished this task in a two step process by making use of the fact that body marker sizes are considerably larger than the facial markers. The first step involved tuning the 3D reconstruction so that only body markers are reconstructed (no facial markers) and consistently labeling the

body markers using velocity-based constraints. Next, the 3D reconstruction can be tuned to acquire facial markers, which will also get body markers. The body markers can then be removed by using labeled data from the first step, resulting in only facial data.

4.2 Stabilization

During a performance, the actor is moving around in the volume resulting in the face markers being translated along with body, while the actor is speaking and emoting. However, for the purposes of retargeting the facial marker data onto the face, the facial data needs to be made stationary without the effect of body and head movement. The nullification of head rotations and body movement from the facial marker data is known as stabilization. Stabilization can be a tricky problem to solve because the face markers do not necessarily undergo a rigid transform to a standard position as the actor continues acting. There are rigid movements caused by head rotations and the actor's motion, but when the actor starts emoting and speaking, all the facial markers change positions away from their rigid predictions. Typically, to solve for the inverse transformation, a few stable point correspondences should be enough but it is not always easy to detect, on a frame-by-frame basis, which markers are stable and have only undergone a change due to the rigid transformation. The noise in the markers 3D reconstructed position also adds to the problem further.

To solve this problem, the Imageworks artists and engineers developed tools that solve the problem in a hierarchical manner by first doing a *global* or *gross stabilization* choosing selective markers, and further refining the output by *local* or *fine stabilization* by analyzing the marker movements relative to a facial surface model. The gross stabilization involves using markers that mostly do not move due to facial expressions – such as markers on the head, ears and the nose bone.

4.3 Facial Cleaning

Although the facial data has been stabilized, the data does contain missing markers due to occlusions and lack of visibility in the cameras, noise caused by errors in 3D reconstructions and occasional mislabeled markers. The cleaning and filtering process developed on *Monster House* made use of a learning system based on good facial model data that helped estimate missing markers, could remove noise and in general ensured the legality of all markers. The system was made scalable enough to handle wide ranges in facial expression as well as be tuned to ensure that the dynamics of the facial data.

These cleaning tools used the basic, underlying FACS theory of muscle groups to group markers together and organize them into groups of *muscles*. Muscle movements can be used to estimate probabilistically where the likely positions of missing markers should be. The predicted location is based spatially on the neighborhood points as well as temporally by studying the range of motion of the markers. It was our experience the probabilistic model and the marker muscle grouping had to be tuned to work for with each actor.

Once all the markers were predicted, standard frequency transforms were used to remove the noise in the data. However, care should be taken to understand the level of signal-to-noise ratio, which can drastically change over different frames for each marker. This is because high frequency content, which is normally categorized as noise, need not be so when the actor moves muscles and creates expression quickly.

4.4 Normalization

When capturing a long performance, such as a movie that spans over more than one day, actors need to remove motion captured markers and reapply them on a regular basis. Although appropriate steps can be taken to ensure that markers are placed at the same position on the face, it is common to have small differences between markers placements at every day positions. These differences can significantly affect the retargeting solutions described in the next section. Normalization is the process of adjusting the marker placements so that the differences in the positions do not compromise the extent of facial expression that the actor has performed and faithfully transfer that onto the character's face. Normalization is accomplished in two steps.

- Each take starts and ends with the actors performing a T-pose (see Figure 1). The T-pose of each actor in a specific take is oriented to the master T-pose of the actor, which was computed during the calibration phase mentioned above. This relies on certain relaxed landmark markers such as the corners of the eyes and mouth, which are expected to change very little from day to day. This alignment enables the computation of offset vectors for each marker.
- The offset vectors can now be applied to the take's T-position so that each marker in the T-pose is identical to the master T-pose. The offsets are then propagated through the actors performance so as to *normalize* all the frames.

5 Retargeting Motion Captured Data Using FACS

As seen in the previous section, the FACS theory gives a set of action units or poses which the authors [7] deem as a complete set for most facial expression. During a calibration phase, we capture mocap frames of the actor that relate to his facial expression corresponding to a FACS pose. Some of these poses are broken into left and right sides in order to capture the asymmetry that an actor face might undergo. Every incoming frame of the actor is then analyzed in the space of all these FACS captured frames. As such, these action unit-triggered poses correspond to facial basis vectors, and each one's activation needs to be computed for an incoming data frame. An illustration of this process is shown in Figure 5. The activations get transferred onto a digital face, which has been rigged using a facial muscle system. On *Monster House*, the facial poses on a character, which corresponded to FACS poses, were generated by an artist, requiring a facial rig. The facial setup on *Monster House* was based on Imageworks' proprietary character facial system (CFS). The system helps pull and nudge vertices on a 3D facial model so that the resulting deformations are consistent with what happens on a human face. It consists of four fascia layers each of which are blended together to create the final facial deformation.

- Muscle Layer– This is the main layer that consists of skull patches with muscled controls that deform the face. The muscle controls are activated by motion captured data.
- Jaw Layer – helps control the jaw movements.
- Volume Layer – helps add volume to the deformations occurring on the face. This helps to model wrinkles and other facial deformation, which can then be triggered by motion captured data.
- Artic Layer – This layer relates to the shape of the lips as they deform – in particular, the roll and volume of lips, essential when the character's lips thin out or pucker in facial expressions.

Also important are the eye positions. The facial rigging system starts by using a high resolution model of the face and filling in the various fascia layers which are driven by the motion captured data. The next step then is to define a way to map or retarget the motion captured data onto the face. This is accomplished by analyzing an incoming mocap frame in the space of all the facial basis vectors as captured using the FACS theory. The weighted values give the relative percentage by which the FACS action units or poses are triggered. These weights are computed using mathematical linear and nonlinear optimization techniques. However, we have found that the weights obtained by such rigorous analysis might not always correspond to good aesthetic deformations on the characters' face and sometimes require an artist's or animator's feedback to tune the system to perform as desired.

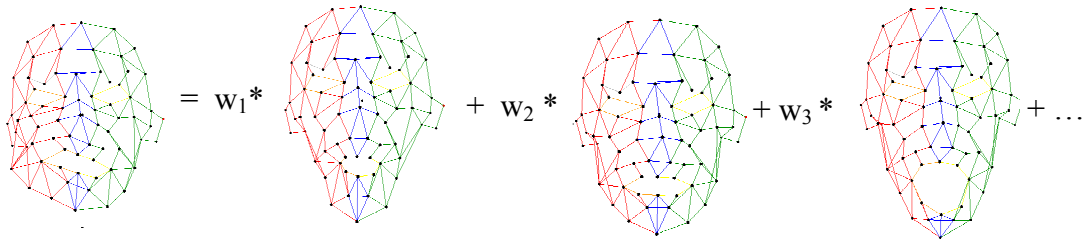


Figure 5. Computing w_1, w_2 … w_n gives the activation of each FACS action unit. The computation of these weights needs mathematical linear / non-linear optimization techniques.

The output of the whole system is shown, for example, in the *Monster House* still images below.

6 Multidimensional Tuning

Although the facial retargeting achieved by various mathematical optimization processes may seem correct mathematically, they do not always necessarily conform to the requirements of a finalized animation shots. There may be a variety of reasons for why this happens:

- Actors not performing wholly according to the actors calibration
- Retargeting issues arising from mapping mathematically correct marker data to an artistically designed face
- Facial model not conforming to an actor's face
- Marker placements differ from day to day
- Attempting to produce an animation contrary to what the actor performed – either the expression is not there in the motion captured data or needs to be exaggerated

Most of these problems can be corrected by a tuning system, which recalibrates the facial [bais??] library based on feedback from an animator. At Imageworks, the artists and software engineers developed a "Multidimensional Tuning System," which takes an artist's input to reduce the effects of incorrect mathematical solves. In this system, post a FACS solve, the animator adjusts a few frames (typically five to ten, among the many thousands) to "look correct." This is accomplished by modifying the weights of poses on a "few" culprit frames that have resulted from a FACS solve. The tuning system exports this changed data, analyzes it and creates a more optimized FACS library. The new library generated is based on the marker range of motion as well as the changed weighting and uses non-linear mathematical optimization tools. This changed library when used in a new solve now attempts to hit the weighting defined by the animator on the few tuned frames at the chosen poses *and also* incorporate this change programmatically on to various other frames causing the whole animation to look better.

An Example of the tuned outputs on the facial library is shown in Figure 6 below

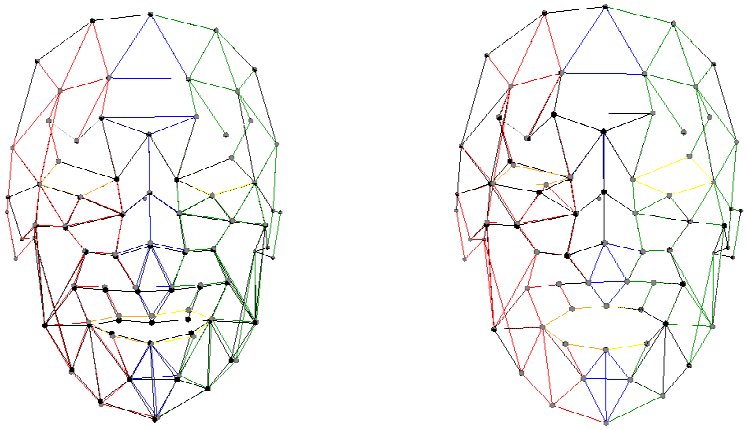


Figure 6. The images above show FACS poses overlaid before and after the tuning phase. The left image shows the lip shut phoneme position overlaid before and after, while the right image shows the same for the lid tightener pose.

In Figure 6, one can see that the new marker positions (in black) have been adjusted to an optimized location based on the animators corrected weighting values over a few tuned

frames. This change is shown on two poses but occurs on more depending on the animator's input. Furthermore, the changes are done in such a way, when the FACS system resolves the solution with the new "tuned FACS matrix," not only does the solve hit close to the weights on the tuned frames, but also on all other frames creating a more correct animation. The effect of this on the retargeting solution on a character is shown Figure 7 and Figure 8.

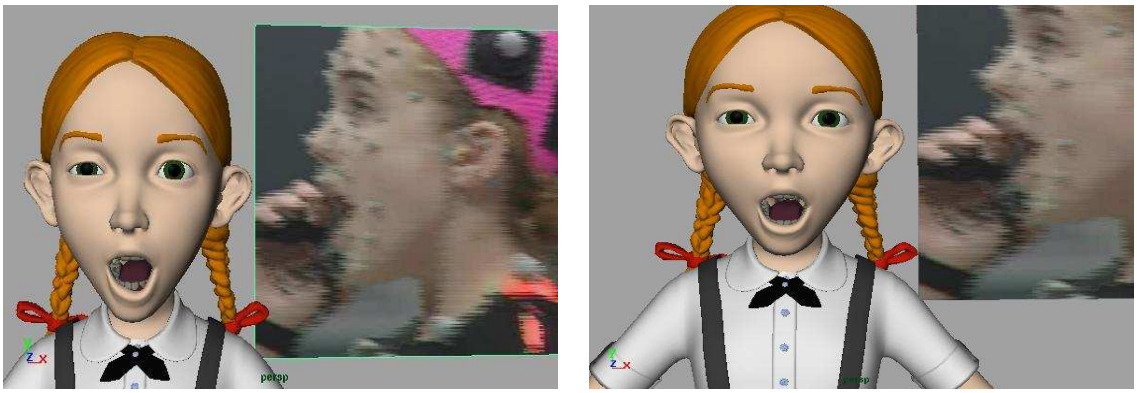


Figure 7. The left image above shows a solved frame with the original FACS calibrated matrix, the right one shows the solved frame with the tuned matrix. The effect of the tuned right lid tightener pose is evident.

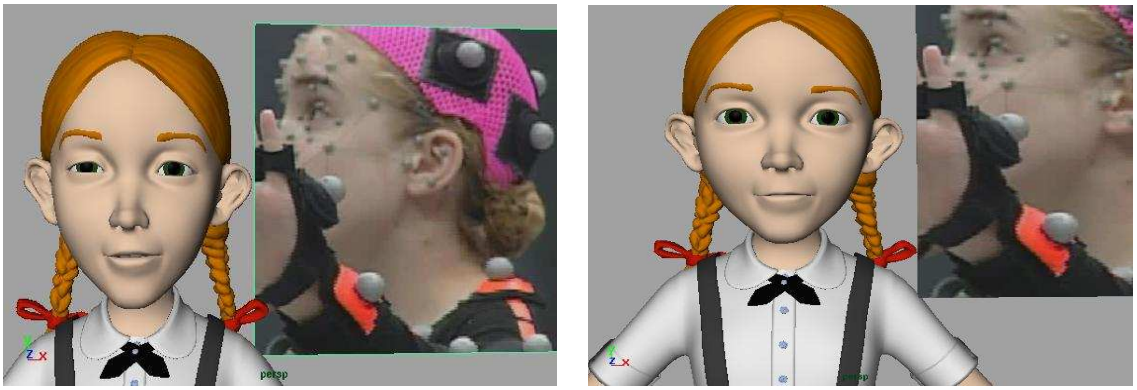
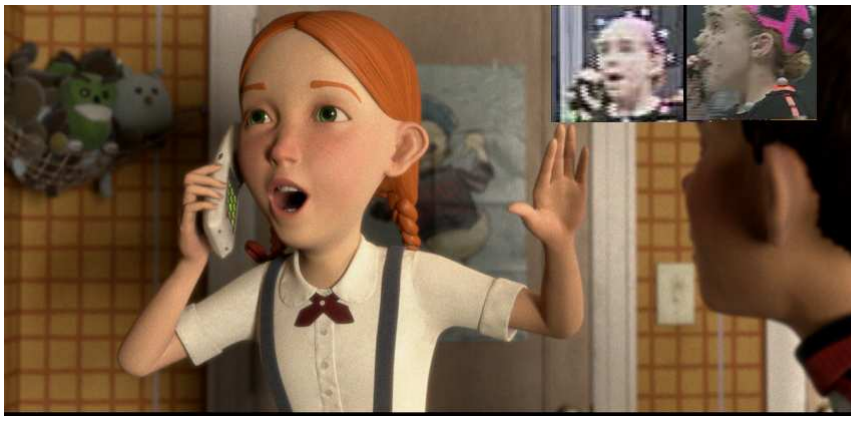


*Figure 8. The left image above shows a solved frame with the original FACS calibrated matrix, the right one shows the solved frame with the tuned matrix. The actor is saying the beginning of the word "**pl**ease." The original matrix solve does not show the lips closed to say the first syllable while the tuned matrix solve does.*

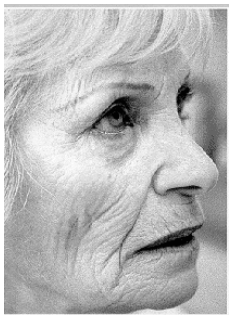
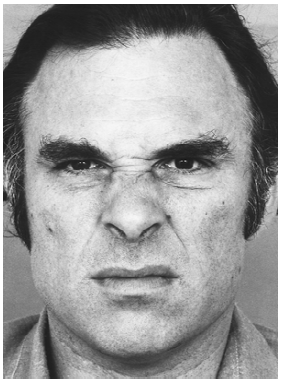


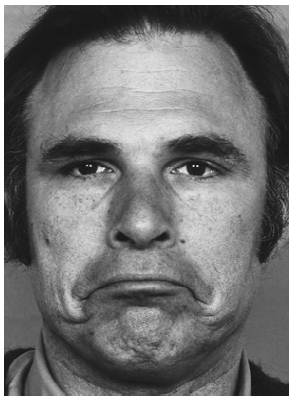
7 Discussion

We have attempted to describe the overall foundations of representation and analysis. The whole process is a combination of artistic input and mathematical processes to model a face and have data drive it via retargeting solutions. Here are a few interesting questions which should provoke good discussion.

- The retargeting solutions though mathematically correct, need not always work aesthetically where an animator's feedback is crucial and final touch ups necessary. How do programmers design tools that encapsulate the fidelity achieved by motion capture data, as well as, allow animators to massage or add on more artistic content additively – ***in such a manner that the two do not conflict***, and consistency is preserved temporarily across a shot(s) and for the entire production? To that end, the mathematical approaches in literature show how to get exact solves in some space (PCA, retargetted blend shapes, etc.), but having user assisted guidance to go towards mathematical minimizations in optimization that make sense aesthetically is not easy.

- Acting in a large volume with body and face, with many actors is what a director and a true performance capture system really desires. By being in the midst of the actual people, an actor is more in sync with his/her role and thereby emotes performances well -- this when compared to a sit down single scenario in front of a system. However, the data capture technologies from both cases yield substantial differences in fidelity – large volume captures with many people are not as stable as a sit down, one person capture. What course should be taken during solving, retargeting and animation such that the cost associated with data processing, clean up and the final integration and animation is reduced?
- Is facial rigging necessary? What does a rig get you in terms of being able to be driven by performance captured data sets. Rigging a face can be an expensive setup increasing production costs, but when and how does rigging solve these problems.
- Retargeting is not always an easy problem to solve, especially when the actor's facial feature and geometry do not coincide with the 3D facial model and rig – for example in case of a stylized character. How do we make universal systems in which the retargeting works effectively between real faces as well as stylized, humanoid characters that may also need exaggerated cartoon like movements?
- Are there any specific "rules" that you can use, whether mathematically founded, heuristically arrived at or even aesthetically pursued for when a person talks vs. muscle and skin movements on the rest of the face? To that end, how do you formalize a model for facial dynamics that are evident in small/large vibrations, jerks intonations when a persons face undergoes fast motion while acting an emoting e.g., moving head quickly from side to side, talking while running?
- Most performance driven techniques for facial animation including ours use visual optical data – video, infrared imagery, etc. What do other performance captures media give you and how can one use them more effectively to drive facial animations. There is various literature on usage of audio [4, 9], but how does a system use both?
- Wrinkles can have a lot to do with facial expression. How do you model wrinkles, and how does a programmatic system trigger their appearance and the extent to which they should be present in an expression? Do you need to physically model them in the 3D facial model, can their effect be achieved through shading and bump maps during the rendering process? A few wrinkles are shown below.

- Every CG face is modeled, textured, animated and rendered. Texturing and rendering are indeed necessary to make a face look real. But in order for the performance of an animated face to look real, to what extent does rendering help in getting close to a real performance?

8 Acknowledgments

All large production projects require artists, animators, integrators, software engineers and management. Cooperative work ultimately manifests in the final product on screen. Here is a list of people involved on the project in Monster House and other related shows that helped create this document.

Management – *George Joblove, Bill Villareal, Brian Keeney, Leslie Picardo, Jerome Chen, Jay Redd*
Motion Capture and Tracking – *Demian Gordon, Dennis J Huack, Darin Velarde, Nancy Ong*
Motion Capture Integration – *Albert Hastings, , Josh Ochoa David Bennett, Ron Fisher, Brian Doman*
Animation – *Troy Saliba, T. Daniel Hofstedt, Kenn Mcdonald, Remington Scott*
Facial Rigging – *JJ Blumenkranz, Michael Laubach, Brian Thompson, Rick Grandy Arthur Gregory*
Software – *Mark Sagar, Armin Bruderlin, Sosh Mirsepassi.*
Training – *Sande Scoredos*
Marketing – *Carlye Archibeque*

9 References

1. V. Blanz, C. Basso, T. Poggio, and T. Vetter. Reanimating faces in images and video. In *Proc. Of Eurographics*, 2003.
2. George Borshukov, Dan Piponi, Oystein Larsen, J. P. Lewis, and Christina Tempelaar Lietz. Universal capture: image-based facial animation for ”the matrix reloaded”. In *Proceedings of SIGGRAPH Conference on Sketches & applications*. ACM Press, 2003.
3. E. Chuang and C. Bregler. Performance driven facial animation using blendshape interpolation. *CSTR- 2002-02, Department of Computer Science, Stanford University*, 2002.
4. Cosker, D.P., Marshall, A.D., Rosin, P.L., Hicks, Y.A., Speech-driven facial animation using a hierarchical model, *VISP(151), No. 4,* August 2004, pp. 314-321
5. Paul Debevec, Tim Hawkins, Chris Tchou, Haarm-Pieter Duiker, Westley Sarokin, and Mark Sagar. Acquiring the reflectance field of a human face. In *SIGGRAPH 2000 Conference Proceedings*, pages 35–42. ACM SIGGRAPH, July 2000.
6. Eisert, P., and Girod, B. 1997. Model-based facial expression parameters from image sequences. In *Proceedings of the IEEE International Conference on ImageProcessing* (ICIP-97), 418-421.
7. P. Ekman and W.V. Friesen, *Manual for the facial action coding system,* Consulting Psychologists Press, Palo Alto, 1977.
8. I. A. Essa and A. P. Pentland Facial expression recognition using a dynamic model and motion energy. *Proc. IEEE Int’l Conference on Computer Vision*, pages 360–367, 1995.
9. Theobald, B.J., Kruse, S.M., Bangham, J.A., Cawley, G.C., Towards a low bandwidth talking face using appearance models, *IVC(21), No. 12-13*, December 2003, pp. 1117-1124.
10. Tim Hawkins, Andreas Wenger, Chris Tchou, Andrew Gardner, Fredrik G¨oransson, and Paul Debevec. Animatable facial reflectance fields. In *Rendering Techniques 2004: 15th Eurographics Workshop on Rendering*, pages 309–320, June 2004.
11. Jensen H. W., Marshiner, S., Levoy, M., and Hanrahan, P. 2001. A practical model for subsurface light transport. *Proceedings of SIGGRAPH ’2001*, 511–518.
12. Jensen H. W. J Buhler, A Rapid Hierarchical Rendering Technique for translucent materials. *In Proceedings of SIGGRAPH 2005.*
13. Jun Yong Noh and Ulrich Neumann. Expression cloning. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 277–288, August 2001.
14. Mark Sagar, Reflectance Field Rendering of Human Faces for “Spiderman 2”. SIGGRAPH 2004.
15. D. Terzopoulos and K. Waters. Techniques for realistic facial modeling and animation. In Nadia Magnenat Thalmann and Daniel Thalmann, editors, *Computer Animation 91*, pages 59–74. Springer-Verlag, Tokyo, 1991.
16. Li Zhang, Noah Snavely, Brian Curless, and Steven M. Seitz. Spacetime faces: high resolution capture for modeling and animation. *ACM Trans. Graph.*, 23(3):548–558, 2004.

“Virtual History : The Secret Plot to Kill Hitler”

Jim Radford

The Moving Picture Company

Abstract

This paper outlines the process the MPC Visual Effects team used to recreate in CG the faces of WWII leaders for Discovery’s TV special *Virtual History*, particularly the techniques employed in order to model and animate the leaders’ faces.

The main areas that will be covered:

- **Casting** the actor
- **Modeling** the face
- **Tracking** into contemporary footage
- **Facial Motion Capture**
- **Post Production** – inserting the CG into the shot
- **Grading** the result to resemble authentic archive material

1 Introduction

Virtual History was groundbreaking in several ways. It was the first TV program to attempt to create realistic historical figures using CGI and Facial Motion Capture, and the first to attempt to create imagery indistinguishable from 1940’s colour archive footage. Three WWII leaders were chosen : Hitler, Roosevelt (FDR) and Churchill. Not only were realistic CG human faces to be created, and not only faces resembling these figures, but faces matching the leaders as they appeared at that stage in their lives.
In addition to this, we portrayed the leaders in situations never before seen – Hitler with his doctor, FDR suffering a heart attack, Churchill discussing chemical warfare – the premise being that this was newly-discovered “unofficial” footage shot at that moment in history, 20th July 1944.
Virtual History was delivered in July 2004, and was first aired in October 2004.

Figure 1. Hitler: 1940's color archive footage

2 Casting

The process started with the actor.
Since we were recreating only the historical figure's face (we had decided against replicating the whole head, in order to limit the volume of CG shots), it was important that the actor's head dimensions matched the historical figures as closely as possible. We obviously did not have access to the WWII leaders, so the actor's compatibility in this respect was established by comparing digital stills of the actor with archive material of the historical figure.

Figure 2. Hitler: Facial area selected for CG reproduction.

3 Modeling

Once the actor was chosen, we took a plaster cast of his head. The actor/CG demarcation line was established, and the historical figure's face was sculpted directly on the cast. The resulting maquette was laser-scanned to obtain a CG model.
The face area was then extracted, and refined to create a workable mesh.

Figure 3. FDR: Actor's head cast, Sculpted maquette.

The static CG face model was then textured and shaded.
Our reference for this was archive material, and observation of faces that matched the age and physiology of the leaders. This included colour, reaction to light (diffuse, specular, translucence), and skin detail (wrinkles, pores, distinguishing marks). Facial hair (eyebrows, moustache), and glasses were also modelled.

As we knew that the viewer's focus would be drawn to the eyes of our characters, particular attention was paid to this area. This was not only true for the textures and shading model, but also for the animation rig being prepared for facial motion capture. Our characters would have to deliver dialogue, and convey emotion whilst doing so.

For Hitler and FDR it was felt that, with the aid of makeup, their appearances were convincing enough to appear in non-hero (ie. non-CG) shots. For Churchill, this was not the case, and a prosthetic layer was added to the actor's jaw and chin.

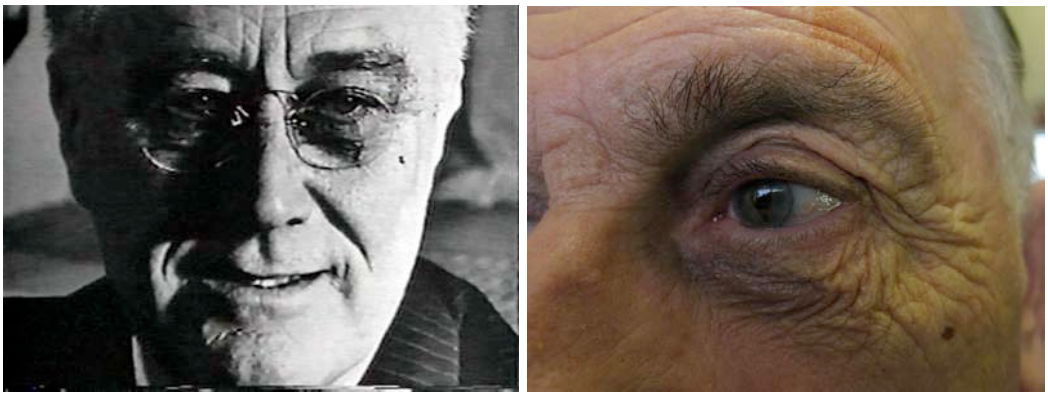


Figure 4. FDR: Actual 1940's archive, and actor's eye detail

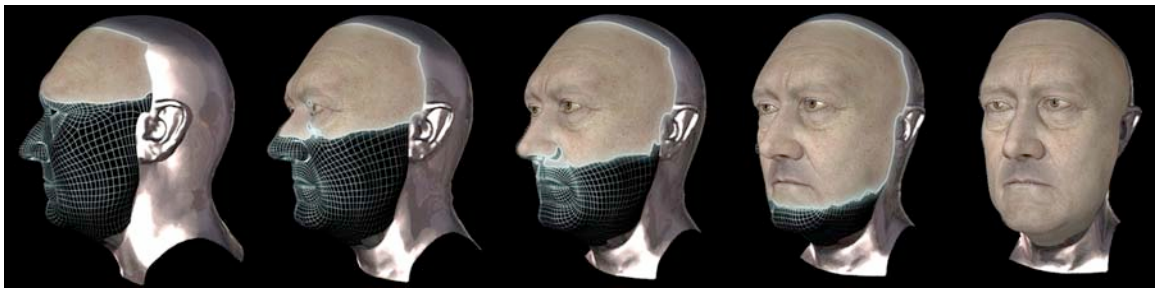


Figure 5. Hitler: Wireframe mesh to textured face (minus the facial hair !)

4 Tracking

During the live action shoot, the actor wore a custom-made facial rig with markers attached, which facilitated the subsequent tracking of his head within the shot. One of the challenges during production was the volume of setups to be completed within the schedule. Shots featuring the leaders were divided into two types: those destined for CG replacement, and those where - due to their being far from the camera, or obscured by other characters – no CG was required. The facial tracking rig therefore needed to be easy and quick to apply and take off, without disturbing the actor's makeup. Whether or not the shot was destined for CG face replacement, the actor would deliver his performance. This was important for direction and the supporting cast, and was invaluable as reference during the facial motion capture shoot and for the VFX team when working on the shots.

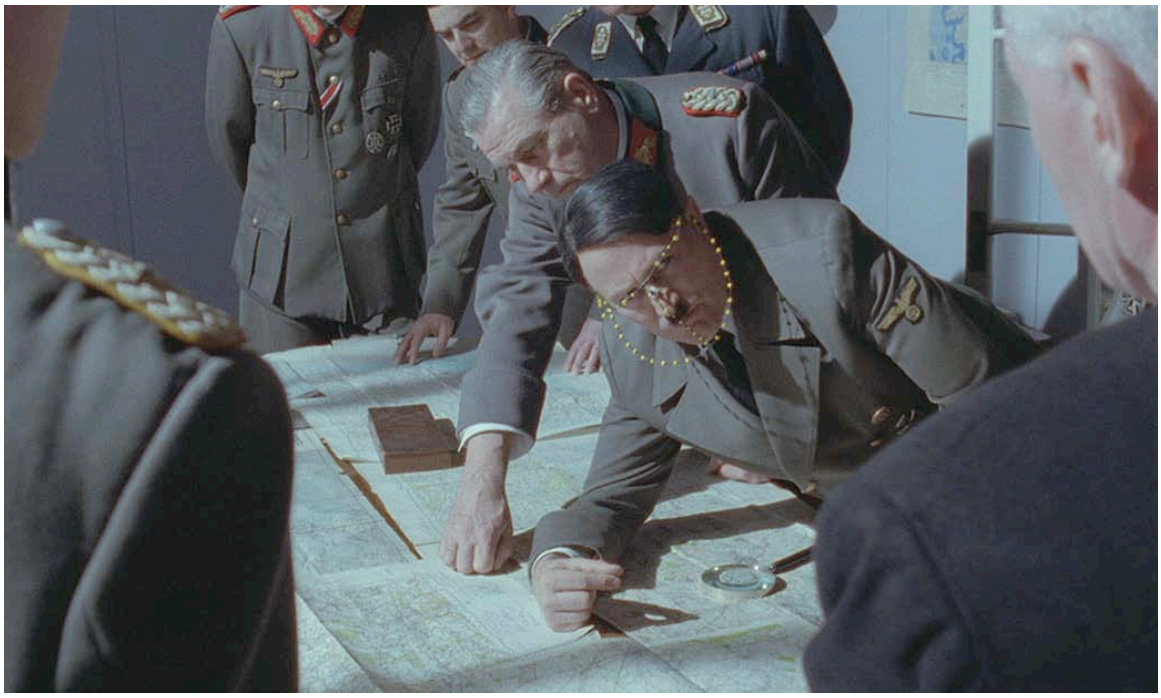


Figure 6. Hitler: Live action plate showing the actor wearing the facial tracking rig.

5 Facial Motion Capture

After Production wrapped, and takes were selected, the Facial Motion Capture shoot took place over three days - one day each for Hitler, FDR, and Churchill. Since Hitler was to be speaking German, it was decided to use a German speaker, rather than the actor who played him in the live action scenes. For the other two characters, the same actors were used.

As in the live action scenes, Churchill performed with his prosthetics applied, as we felt it was preferable to capture him with his facial form as close to the target mesh as possible, rather than relying more heavily on the retargeting part of the data processing pipeline.

Eyetronics (www.eyetronics.com), a company specialising in static and dynamic scanning, supervised the motion capture shoot.
First dots were placed on the actor's face to trace specific movements. As each actor spoke, a grid was projected onto his face as a reference guide to correlate movement from frame to frame. For each frame, about 30,000 points were captured – 100 times more than with traditional motion capture techniques of that time.

The shoot proved challenging for the actors. Not only were they required to assume their character without the aid of wardrobe, supporting cast etc, but they also had to deliver their dialogue accurately and "with feeling", whilst trying to keep their head still, and all the time staring into a bright light !

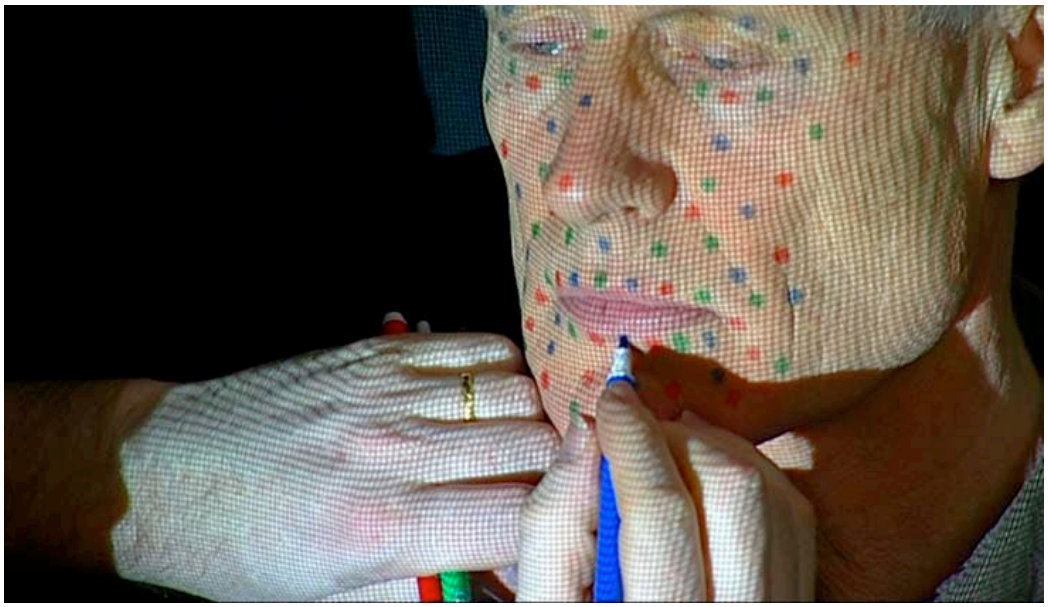


Figure 7. Hitler: Applying the dots to the motion capture actor.

Close attention was given to the duration of the delivered dialogue. Time is money, as the saying goes, and since the project had a finite budget set aside for data processing, a rolling total was kept to ensure that we were on track. Sometimes a good take was discarded, not because the delivery was poor, but because it had to be quicker for budgetary reasons.

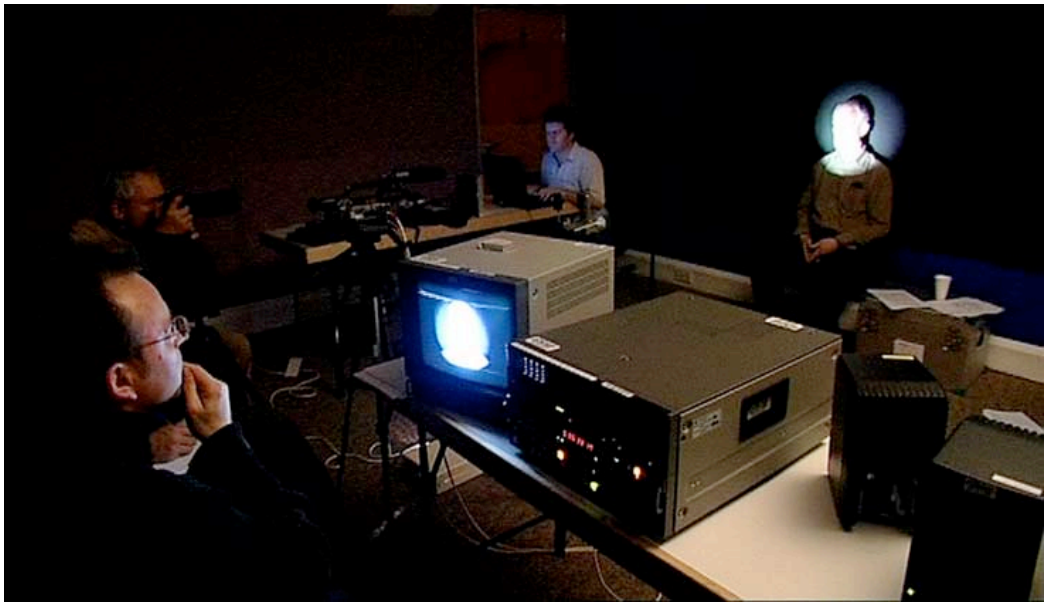


Figure 8. The motion capture shoot. Note the actor under the glare of the projected grid.

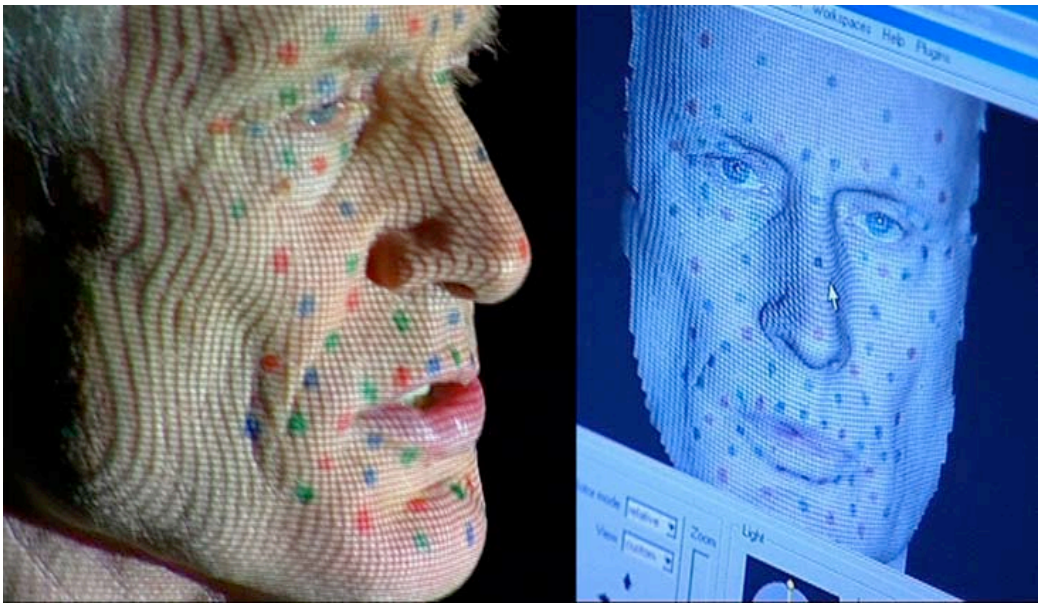


Figure 9. Hitler: The actor during the shoot, and the captured data.

6 Post Production

The static textured face meshes were tracked onto the live action actors in their respective shots, and lighting began.
Eyetronics delivered the motion capture data to us as Maya scenes, along with clips of the captured footage with audio and corresponding driven CG meshes (Figure 10). These clips were important as it allowed to us to accurately position the dialogue within the shots. Certain live action shots had their dialogue slipped or partially replaced, in order to better tell the story.

There were, of course, occasions where shots did not work immediately, and a certain amount of time was spent trying to find our way out of “Uncanny Alley” !
Sometimes it was remarkably difficult to establish where the problem lay, and we helped ourselves by using a checklist on a per-shot basis :

- Does the face fit the actor’s head properly (position, scale, orientation) ?
- Is the track good ?
- Does the face resemble the historical character ?
- As he would look on this day ?
- Is the lighting correct ?
- Is the eye animation correct ?
- Etc …

The motion capture data gave us an animated facial mesh, but certain elements were rigged and animated directly. The eyes were not captured, so these were animated by hand, which worked well, since this allowed us to direct them according to the dialogue

and emotion in the shot. We also “painted out” the motion in the areas immediately around the eyes, and animated these using blendshape techniques.

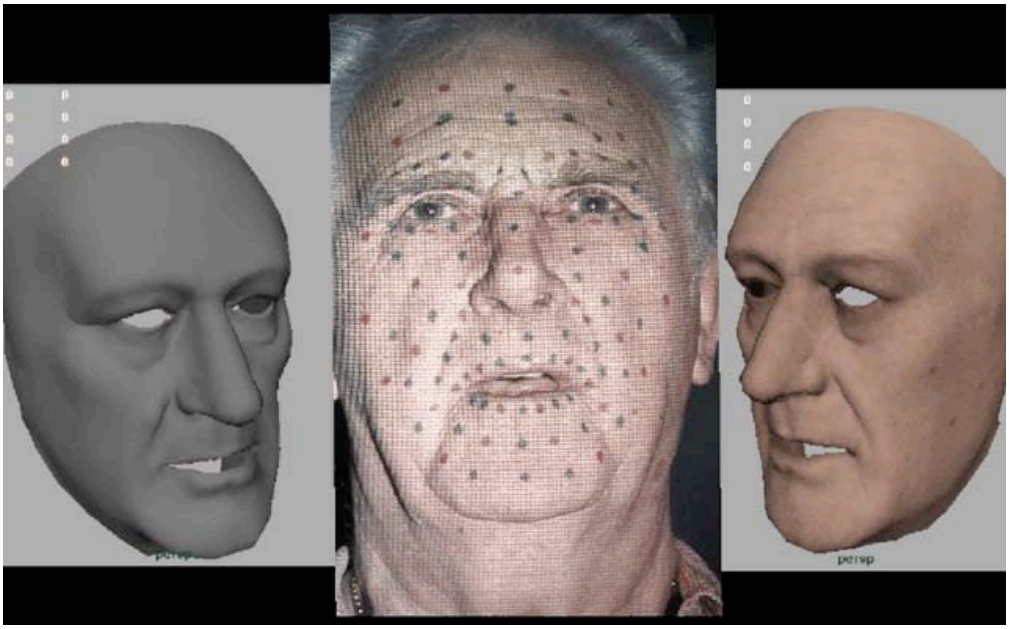


Figure 10. FDR: The actor, and the driven realtime CG meshes

One particular problem we encountered was the noise / signal ratio within the data. Eyetronics’ technique captured very fine motion, which we wanted to keep as much of this as possible, especially in scenes where the emotion was conveyed by movement that was felt rather than seen. However, sometimes we found that in removing the noise from a clip, the signal (movement) disappeared too ! We concluded also that the relatively unusual conditions that the actors were asked to perform under during the motion capture shoot probably contributed to the sometimes-muted facial performance.

The animated and lit faces were composited into the live action footage using standard 3D and 2D techniques.

7 Grading

Once we were happy with the technical composite, the footage was manipulated to resemble authentic 1940’s color archive material.
Essentially, we created a “life story” for each of the three character’s film footage, from initial exposure in-camera, through to the present day, when the footage was supposedly “discovered”. We employed a range of techniques towards this end, including film-stock characteristics, lens imperfections, camera frame-rates, chemical reactions during developing, and damage incurred during storage.

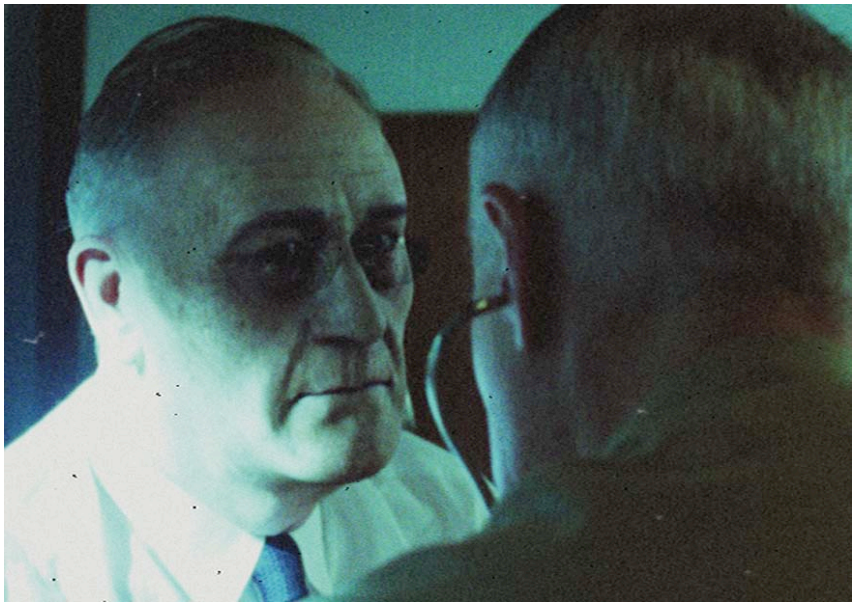
Figure 11. Hitler : Live action plate, final graded image

The three strands – German, American, English – were given different resultant "looks" based on the considerations described above. These also helped the viewer better differentiate between them in-program, especially when different leaders' scenes were cut back-to-back.

8 Conclusion

Creating realistic CG human faces remains a very ambitious task, especially within the time and budgetary constraints common in TV broadcasting. The completion of Virtual History left us with as many questions as answers, but this body of work is also testament to the talent and dedication of the VFX team assembled for this project.
We look forward to contributing further in future work, and are excited by the challenges that lay ahead.

9 Final Imagery



10 Acknowledgements

Since 2004, some of those I mention here have moved on to pastures new.
I'm including them here in the positions they held at the time of the project.
I'd like to thank the following :

David Abraham, Discovery UK
Dunja Noack, and the team at Tiger Aspect Productions
David McNab, Director
Eyetronics for Facial Motion Capture

The VFX team at MPC :
Klaudija Cermak, Loraine Cooper, Christophe Damiano, Dean Koonjul, Oliver Money, David Mucci, Minh Nguyen-Ba, Tom Phillips, Glen Swetez, Kim Taylor, Simon Thomas.

Facial Performance Capture and Expressive Translation for King Kong (sketches_0077)

Mark Sagar[1]
Weta Digital

To successfully convey and translate the most ferocious to the most subtle and soulful moments (often expressed purely through the eyes) from actor Andy Serkis's performance to the face of King Kong required applying innovative motion capture analysis and mapping techniques.

Standard facial motion capture techniques used in film production involve some form of direct drive between the marker position and the facial geometry control mesh with some form of geometric transformation if retargeting is necessary.

Typical problems with this approach include susceptibility to noise, and the difficulty to edit if a facial performance change is required. Also although critical to the performance, detailed eye motion is not usually captured, and it is usually animated later from video reference which can be a challenging task due to the high frequency components of eye motion.

For many reasons, including the complexity of King Kong's facial geometry, the subtlety required being close to the range of marker displacement error and the desire to work seamlessly with animation meant standard motion capture techniques were unsuitable.

Expression Space

The technique used for King Kong breaks the direct geometric connection by first fitting the motion capture data to an *expression space*, which is then mapped to another expression space defined by animation control presets on the creature facial puppet.

The expression space is effectively a Gorilla specific superset of Ekman and Friesen's Facial Action Coding System (FACS) which groups facial muscles which work as units. It represents an effective alphabet of facial expression; any expression can be formed by combinations of the individual elements.

Expression space has several benefits. It provides a transcription of the communicative content of the performance, and is ideal to map in an art-directable way to a topologically different character. It provides an intuitive format for motion editing and making performance changes. Because the face is constrained in its motion, fitting to an expression space significantly reduces noise and allows for intelligent filtering.

Expressive Translation

For King Kong, actor Andy Serkis closely studied gorilla behaviour and facial expression and mimicked it as much as possible. However while there is much in common between the facial structure of humans and gorillas, there are characteristic expressions a gorilla can make that a human cannot unambiguously mimic (and vice versa).

Since humans and gorillas have similar musculature it was possible to map most of the expression space in a muscle to muscle way especially in the upper part of the face. To enable the actor to perform the unique gorilla expressions, a mapping was made between a particular human mouth configurations and distinct gorilla mouth shapes.

Indirect Eye tracking

The eyes are the window of the soul. Eye motion is complex and driven by internal and external stimuli and reflexes. It has many very high frequency components making it difficult to keyframe animate. Although very subtle the relationship of the pupil and eyelid position in facial expression is a critical element of the performance.

When the eyeballs move the eyelids are affected mostly due to displacement caused by the cornea. However the actions of the surrounding eye muscles such as the oris orbicularis also affect the eyelids so expressions and corneal deformations are not independent. When the expressions are fitted the contribution of eye gaze direction is also solved for enabling indirect yet accurate tracking of eye movement, including high speed saccades.

Motion editing

The output of the system is a set animation curves which control muscle groups and gorilla expressions and are easy to manipulate for motion editing and combining with animation if performance changes are called for; For example brow motion could be toned down without affecting the rest of the face, or the jaw could be closed while keeping the action of muscles adding expression to the mouth.

A special GUI was created which allowed direct manipulation and visualization of the muscle groups directly in Maya, with sliders in 3D space moving with the face. These worked in conjunction with a rich set of curve editing tools providing for a fast intuitive workflow.

Contributors: Rudy Grossman, Weta Digital

References: P. Ekman and W.V. Friesen. *Facial Action Coding System: Investigator's Guide*. Consulting Psychologists Press, 1978

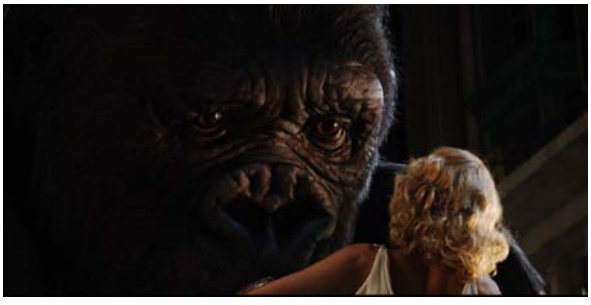


Figure 1: Detailed motion capture of the eyes and surrounding muscles helped give Kong a sense of soul.

[1] Email: marks@wetafx.co.nz

Playable Universal Capture: Compression and Real-time Sequencing of Image-based Facial Animation

George Borshukov* Jefferson Montgomery* Witek Werner*

Worldwide Visualization Group
Electronic Arts, Inc.

Figure 1: Frames of Leanne Adachi's performance that were captured, processed, and rendered using the techniques described in this paper.

Abstract

This paper describes a facial animation playback scheme based on simultaneous encoding of captured face textures and geometric positions. The high quality facial data is acquired using a more robust variant of the Universal Capture (UCap) technique that incorporates state-of-the-art marker-based motion capture. A novel encoding, based on Principal Component Analysis, is used to encode the data and uses more components in areas of the face that require them yielding finer control over the visual quality of the reconstruction. Furthermore, a frame interpolation technique is demonstrated that operates within this component sub-space and provides smooth transitions between facial expression clips allowing us to effectively apply a motion graph approach to facial animation. The method runs in real-time for tens of high-fidelity characters on current hardware.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.3.6 [Computer Graphics]: Methodology and Techniques; I.3.6 [Computer Graphics]: Computational Geometry and Object Modeling; E.4 [Coding and Information Theory]: Data compaction and compression

Keywords: Performance capture, Facial Modeling, Facial Animation, Image-based Techniques, Data Compression, Real-time Implementation, GPU Rendering

*e-mail: {gborshukov, jmontgomery, wwerner}@ea.com

1 Introduction

There are a vast number of factors involved in reproducing the subtleties of realistic facial animation. So many, in fact, that it is near impossible for an animator or a computer simulation to achieve genuinely realistic results. One reason for this is simply a lack of sufficient models for the important characteristics and subtle dynamics of facial expressions expected by human observers, who are all ultimate experts at watching faces! The *Uncanny Valley* has recently become the standard term for the unintentional creepy appearance of near-photoreal, computer generated, animated faces. This phenomenon, first predicted by [Mori 1982] in the context of robotics, is being increasingly seen in human characters in video games. The work described in this paper can be viewed as a direct attempt to build a bridge across the Uncanny Valley and start climbing up the other side for the most challenging applications of all for digital humans — those requiring real-time interactivity.

The best feasible solution currently available for achieving the highest possible realism and believability is to simply capture all aspects of a facial performance and use the captured data to reconstruct the performance. This paper discusses a new capture process and outlines some improvements introduced to ensure a more robust, production-ready pipeline that preserves important subtleties.

Such capture techniques produce a large amount of data which prohibits recreation at interactive rates. In order to reduce the storage and memory access burdens, a compressed representation that retains the important captured subtlety is desired. This paper introduces a novel compression technique based on multivariate analysis that accomplishes this with the following benefits:

- the encoding is an optimal approximation in a meaningful sense,
- the small per-frame representation can yield a very high compression ratio,
- useful operations can be performed in the compressed space, reducing required computation, and
- the decoding algorithm can decode frames in random order.

Ultimately, these new techniques allow us to unleash the richness of the high-fidelity data content we capture. Our compression allows us to encode numerous expression clips and then sequence short constituent expressions with minimal blending across the transitions. This paper shows how to use this approach in a real-time interactive application that delivers compelling visual experiences and allows

- interactive switching between different captured clips at any point, triggering and blending appropriate facial motions from a state graph,
- complete freedom of camera movement,
- control over the character's shading, lighting, and environment,
- stylization either by capturing made-up actors or by adjusting the shaders for additional artistic effect,
- application of deformations to the mesh geometry after the data has been processed and layering damage or sweat effects through textures or shader parameter adjustments, as well as
- removal of captured rigid head movement and application of new overall head movement which can come from body motion capture, hand animation, or procedural techniques.

Section 2 of this paper introduces a variety of research that enables this work or addresses the problem in a different way. Section 3 describes our data capture process and Section 4 covers the data compression. Section 5 shows how to combine performances into new sequences. Section 6 presents qualitative and quantitative results while Section 7 discusses concrete steps and ideas for future work and Section 8 offers some concluding remarks.

2 Related Work

One of the earliest efforts in the field of facial cloning was the pioneering system developed by [Parke 1972; Parke 1974] who made use of two orthogonal photographs and patterns painted on a performers face to recover 3D facial geometry. With advancements in range scanning, researchers have explored its use to automatically model faces. The resulting data can be fitted with a structured face mesh and additionally augmented with a physically based model of skin and muscles [Lee et al. 1993; Lee et al. 1995; Terzopoulos and Waters 1991; Waters 1987].

The idea of recovering facial geometry from images led naturally to motion estimation from video. An early system by [Williams 1990] tracks the 2D face motion of a performer from a single video stream. Later, [Guenter et al. 1998] extended this approach to 3D recovery from multiple video streams. Other researchers have explored marker-less tracking techniques, often relying on more sophisticated models such as linear blend shape models [Blanz et al. 2003; Pighin et al. 1999], bi-linear models [Vlasic et al. 2005], or physically-based models [Terzopoulos and Waters 1991]. Tracking from video using more complex models has also been discussed by [Essa et al. 1996; Reveret and Essa 2001]. Recently, [Hyneman et al. 2005] described a very successful tracking approach developed in the context of Disney's *Gemini Man* film attempt in 2000.

A promising new research direction is the recovery of dense geometry from video. The system developed by [Zhang et al. 2004] uses six video cameras and two structured light projectors to recover the facial geometry completely automatically, whereas [Borshukov et al. 2003] used optical flow calculated from five hi-def cameras to recover the facial geometry and texture at every frame which required some manual assistance, but no structured light. [Bregler et al. 2000] have also done pioneering work on recovering deformable models from single video streams.

[Lee et al. 1995; Waters 1987] designed systems to make it relatively easy to animate facial expression manually. The techniques in this paper do not synthesize animations using a physical or procedural model of the face. Instead, they capture facial movements in three dimensions and then replay them in real-time, faithfully reconstructing a particular person's facial expression. Hence our work is also unlike that of [Essa and Pentland 1997] which attempts to recognize expressions or that of [DeCarlo and Metaxas 1996] which can track a limited set of facial expressions. It differs also from recent work by [Sifakis et al. 2005] which combines a highly complex non-linear model of muscle activations with motion capture.

[Pighin et al. 1998] captured different facial expressions consisting of geometry and texture from an arbitrary number of images. They used these for creating smooth transitions between expressions by morphing. However, their library of eight fundamental expressions limited the space of possible transitions, and their shape construction required manual guidance.

Our work is similar to the work of [Guenter et al. 1998], but differs in important ways. The capture system and process is streamlined, modernized, and tested in large scale production. We deploy a state-of-the-art motion capture system with 8 infrared cameras, 3 high-definition cameras, and only 70 small retroreflective facial markers, instead of 6 cameras and 186 medium sized florescent facial markers. Our system never loses track of a marker and has sub-millimeter resolution. Our key contribution is extending their compression approach from a PCA on skeletons and MPEG-4 on textures to a new Variable PCA for both the full mesh and textures allowing us to compress detailed motions and achieve interactive facial animation. Arguably, this has less objectionable compression artifacts and potentially higher compression ratios (due to its ability to capitalize on longer-term temporal correlations) but the real win is the simplicity of decompression.

3 Data Capture

In an effort to generate photo-realistic believable human performances, we endeavored to capture and reconstruct data of very high fidelity. For this we have developed a more robust production level version of the Universal Capture (UCap) system introduced by [Borshukov et al. 2003]. A pipeline overview can be seen in Figure 2. Instead of doing a full markerless tracking approach using optical flow we deploy a facial motion capture system consisting of (see Figure 3): 8 IR cameras plus 3 synchronized, high-definition, color cameras framed on the actor's face as well as an ambient lighting setup. Around 70 small retroreflective markers are placed on the actor's face. The IR cameras are used to track reflected light from the facial markers in the infrared domain at a rate of 120 frames per second (fps). Simultaneously at 30 fps, the hi-def color cameras record all the subtleties of the actors performance including color variations in blood flow due to emotional state changes, skin compression in the areas of wrinkle formation, self-shadowing, and ambient occlusion effects. As part of the process the two data sources are aligned in both space and time. Mocap facial data is tracked in VICON iQ or House of Moves Diva. The facial markers are isolated and stabilized in Alias MotionBuilder. IO Industries Streams 5 is used to extract image sequences (see Figure 4) from the recorded video. Eos Systems PhotoModeler is used to recover the color camera positions in 3d space with respect to the recovered facial marker 3d locations.

The 3d motion path of the face markers is then used as the direct source of deformation for the face geometry. We scan using XYZRGB's technology and convert to a mesh surface using Cyslice. We achieve the deformation with a facial *bone* rig in Maya. The number of bones is equal to the number of markers on the face. Each bone is weighted to the face mesh using a standard skinning process. The vertex weights which describe how the movement of a bone (joint) affects the movement of a vertex are adjusted by an artist. What make our skinning process successful, and qualitatively different from other approaches, is the availability of a reference image sequence which the artist uses to achieve an optimal weighting. Once this one-time weighting setup is perfected using one captured move it is then reused for the remaining captured moves with

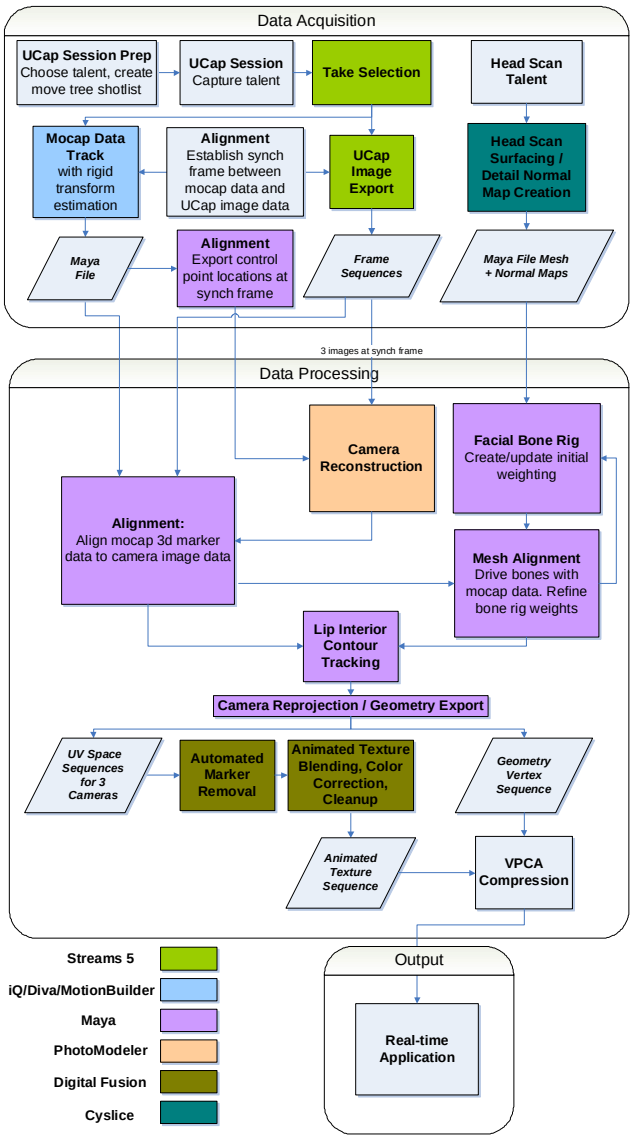


Figure 2: Pipeline diagram.

the same actor.

This deformation approach achieves excellent results in accurately reconstructing the facial shape and motion in all areas of the face; except for the interior contour of the lips for obvious reasons of not being able to place markers there. Currently, we solve the lip contour problem with the addition of 8 bones which control the shape of the lip interior and whose placement over time is determined by an animator using the background images as a reference. Another problem area where we have a similar solution is the tongue, which is of crucial importance in reproducing speech. In Section 7 we describe the approach we have taken to fully automate this process. Figure 5 shows the skinned mesh with the facial bone rig.

Once we have achieved successful reconstruction of the facial deformations as shown in Figure 6 then we can project the input image sequences from each camera for every frame to produce animated facial texture maps. The presence of facial markers would at first glance prevent us from using the captured color images as textures. However we exploit the fact that current generation motion capture systems need very small markers (1.5 mm in diameter) which are also highly retroreflective. By placing ring lights around each

Figure 3: Capture setup: above, the hi-def cameras with ring lights, mocap IR cameras, and ambient lighting setup. Below, make-up application (purely for artistic effect) and the use of a talent director which is key to good actor performances.

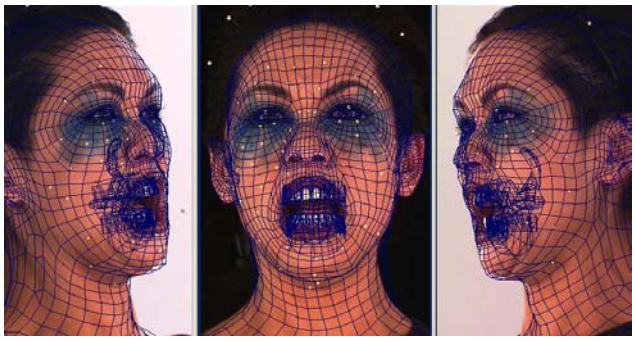


Figure 4: Input camera views: notice the nice marker separation which makes removal straight-forward.

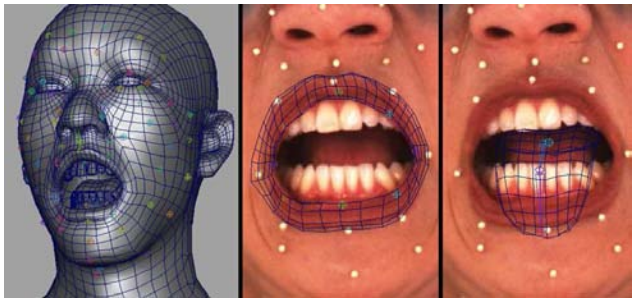


Figure 5: Facial bone rig: one bone per marker and additional bones for mouth/tongue tracking.

of the hi-def cameras we guarantee that the markers in each of the color images appear at very high intensity which makes them easy to key out and replace with adequate skin textures. We perform this marker removal and “hole-filling” process in the final stage of animated texture generation where the re-projected (in UV space) camera views are blended to produce a final marker-free composite image sequence as shown in Figure 7. This process is accomplished in the Eyeon Digital Fusion compositing package.

The combination of accurate mesh deformation, ambiently lit animated textures, and advanced shading techniques can achieve high realism of facial animation and rendering in challenging applications [Borshukov et al. 2004]. One of the contributions of this paper is in coming up with a more robust and efficient way of acquiring

Figure 6: Geometry reconstruction: the process results in accurate and detailed shape recovery for every frame.

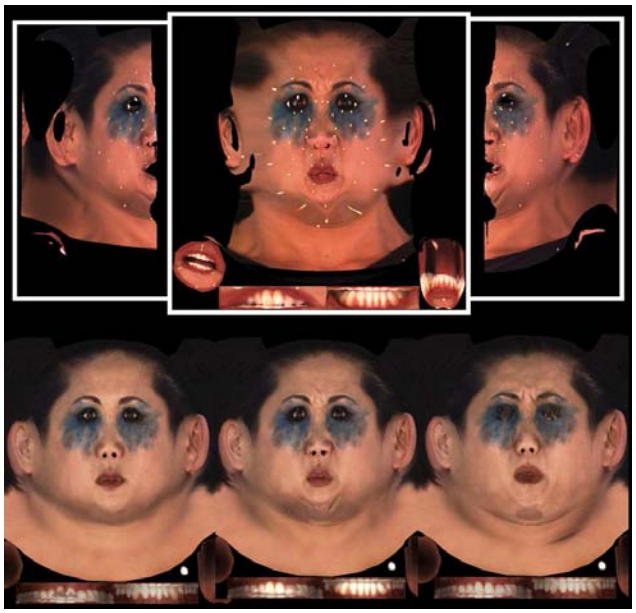


Figure 7: Texture generation: combining images in UV space with data from each camera.

data of this quality. The most important contribution however is the ability to aggressively compress the extremely large data sets associated with this technique while preserving their quality and allowing triggering, blending, and decompression at interactive speed. The following sections discusses the compression approach.

4 Data Compression

For our application, the captured performance is organized into two matrices, X_{img} and X_{geo}, containing image and geometry data respectively. For each captured frame, color channels and geometry attributes each contribute a single column vector, whose length is the number of pixels and the number of vertices respectively, to their respective matrix[1]. For example, one frame of a facial performance may specify 3 column vectors (representing the red, green, and blue color channels) of X_{img} and 9 vectors (representing the position, normal, and binormal attributes) of X_{geo}. In this formulation, the performance is analyzed, compressed, scheduled, and decompressed (as described in the following sections) in a process that is performed separately — but identically — on both the image and geometry data (X_{img} and X_{geo}).

4.1 Principal Component Analysis

For each of the data matrices (hereafter simply labeled X) assume that the n column vectors represent independent and identically distributed samples from an m-variate exponential family (e.g., Gaussian) distribution. Under such an assumption, sufficient statistics are provided by the sample mean and covariances, and the statement is implicitly made that directions out from the mean with largest variance contain the most relevant information content.

Principal Component Analysis (PCA) is a non-parametric optimization procedure for extracting information content from measured data, ranking its level of interest, and describing it using orthogonal components. As such, it is extremely useful; not only for identifying structure in measured data, but also for discovering lower-dimensional descriptions while optimally retaining relevant information.

The first step of PCA is to center the measured data by removing the sample mean. Define the matrix A such that each column equals the sample mean subtracted from the corresponding column of X (as shown in Equation 1, where the notation x_{ij} denotes the element of matrix X at row i and column j).

$$A : a_{ij} = x_{ij} - \frac{1}{n}\sum_{k=1}^{n} x_{ik} \tag{1}$$

The original vector basis for this representation corresponds to the experimenter's interpretation of the measurements, but PCA reveals a more useful, orthonormal basis with which to describe the data. The new basis is derived from the structure of the data itself through the *singular value decomposition* of the centered data:

$$a_{ij} = \sum_{k=1}^{r} u_{ik} s_k v_{jk} \tag{2}$$

U and V are both orthonormal matrices whose column vectors are the *left* and *right singular vectors* respectively, and S is a diagonal matrix with $r = \text{rank}(A)$ decreasing, but positive, *singular values*. The left singular vectors not only span the column space of A, thereby specifying an orthonormal basis for the data, but also have axes that are aligned along variance extrema and are independent (i.e., the transformed data has a diagonal covariance matrix). These axes are called the *principal components* of the data set.

When $r < \min(m,n)$, there is redundancy in the original basis description, and PCA reveals which components can be removed from the data without any loss of information yielding a compression ratio of $\rho_{\text{rank}} = \frac{1}{mn}(r(m+n)+m)$. In practice, however, a large number of low-variance components can also be disregarded without significant degradation in the data, as shown in Equation 3.

$$a_{ij} \approx \tilde{a}_{ij} = \sum_{k=1}^{c} u_{ik} s_k v_{jk}, \qquad c < r \tag{3}$$

Although lossy, removal of the least-principal components in this way achieves further compression ($\rho_{\text{pca}} = \frac{c}{r}\rho_{\text{rank}}$) while guaranteeing optimal retention of the original variances. Specifically, no other rank-c matrix can do better, in Frobenius norm[2], to this approximation of A.

[1] This assumes, for example, that each image color component is a separate sample from an identical distribution, which is useful when colors vary in similar ways.

[2] The *Frobenius matrix norm* is the discrete version of the L_2 norm, and is defined by the square root of the sum of squares of the matrix elements. The approximation error is given by:

$$\|\tilde{A} - A\|_F^2 = \sum_{i=1}^{m}\sum_{j=1}^{n}\left(\sum_{k=c+1}^{r} u_{ik} s_k v_{jk}\right)^2 \tag{4}$$

Typically, both the geometry and texture data captured using the techniques of Section 3 both exhibit highly correlated structure over long sequences. This can be observed in the distribution of singular values in Figure 8, which are directly related to the error induced by a corresponding dimensionality reduction. Consequently, this important structure can be accurately represented using a surprisingly small number of principal components. Several captures that we have analyzed, although representing drastically different actors and performances, share this important property.

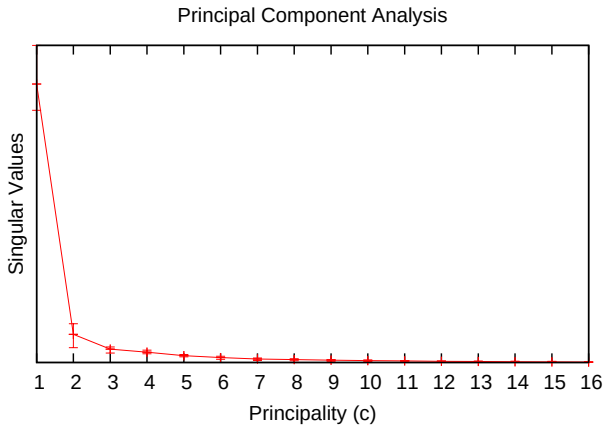


Figure 8: The first 16 singular values recovered from several different performance captures. A significant portion of the data variance is representable by a small number of principal components.

4.2 Relaxing The Assumptions: "Variable PCA"

Despite the optimality of the approximation, standard PCA compression produces poor results if the data does not satisfy the inherent assumptions. Specifically,

- the captured data may not be generated by a linear superposition of underlying components — in which case, PCA linearly approximates the non-linearity which can be inefficient, requiring many more principal components than the inherent structure of the data dictates; or
- the captured data may not be generated from an exponential distribution — in which case, the sufficient statistics considered may not determine the data and the Frobenius norm may not be the appropriate error metric.

The consequences of non-conformance tend to be localized compression artifacts. In this application, you might observe excessive relative blurring in areas such as the eyes which can be very detrimental to qualitative acceptance of the render due to a human observers tendency to scrutinize such areas.

Despite the fact that these assumptions may introduce compression artifacts, they also engender simple and efficient algorithms, which is key for an interactive implementation. Instead of abandoning PCA, note that the number of components representing information is measurement-invariant[3] and the computation required for reconstruction is also measurement-invariant[4]. At the same time for some data sets the error contribution of each measurement *does* vary, as shown in Figure 9.

This is a qualitative inefficiency that can be addressed by balancing the representational cost of each data measurement using a *basis configuration*: a different number of components for each dimension, $\{c_i\}$, as shown in Figure 10. By increasing the dimensionality of the basis representation for measurements with poor

[3] A PCA-compressed data set holds $nc + c + 1$ elements per measurement.

[4] Decoding a PCA-compressed data set requires nc multiply-and-add operations per measurement.

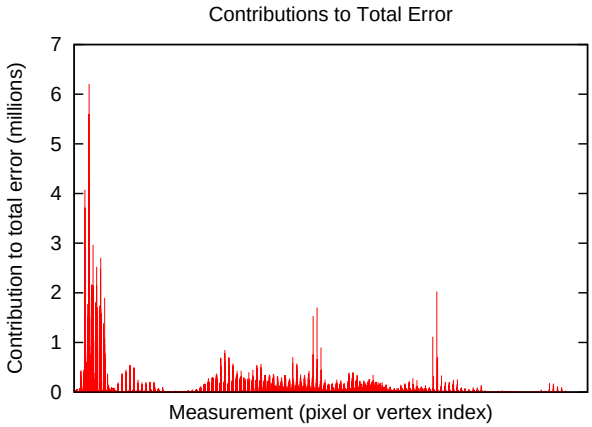


Figure 9: For a fixed number of components, the principal component sub-space minimizes the total Frobenius-norm error. However, each measurement's contribution to that error is different as shown in this example data set.

approximations, while equivalently decreasing the dimensionality for well-approximated measurements, this approach effectively increases the rank of the approximation without increasing the storage size, thereby increasing the compression quality.

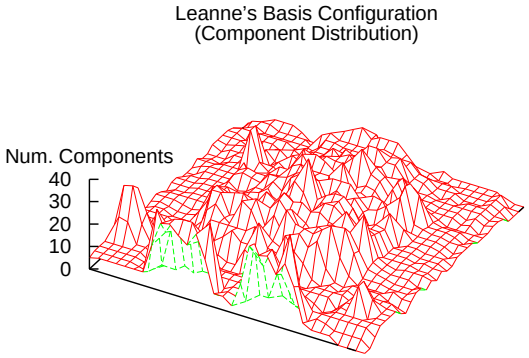


Figure 10: The basis configuration found for the texture data in Leanne Adachi's performance (in UV space as shown in Figure 7). Each pixel is reconstructed using a different number of components.

Although the altered basis is generally *not orthonormal* and the basis axes are *not independent* as in standard PCA, reconstruction is still a linear operation:

$$a_{ij} \approx \tilde{\tilde{a}}_{ij} = \sum_{k=1}^{c_i} u_{ik} s_k v_{jk} \tag{5}$$

Finding the best basis configuration is a combinatorial, NP-hard optimization problem. The goal is to alter the principal component basis by setting $u_{ij} = 0$ for all $j > c_i$ such that the Frobenius norm approximation is minimized while simultaneously maintaining a constant compression ratio, given by Equation 6[5].

$$\rho_{\text{vpca}} = \frac{1}{mn}\left(\sum_{i=1}^{m} c_i + n \max_{i \in [1,m]} c_i + m\right) \tag{6}$$

[5] The zeroed component values are not stored by the compressed representation.

This is an Integer Programming problem and can be solved using a standard technique such as Branch and Bound using Linear Programming regressions. However, because the dimensionality of this application is so high and since the value of each basis configuration can be evaluated, we elected to use a local search starting from the PCA configuration $\{c_i\} = \text{constant}$. This is modified by trading δ components from the best-approximated measurement to the worst-approximated measurement; a process which is repeated until such a trade no longer reduces the total error. To avoid local minima, it is also helpful to cycle δ through $[1, \max_{i\in[1,m]} c_i]$ until no trades of any amount can reduce the error.

4.3 Implementation

Besides providing very high potential compression ratios with relatively inoffensive artifacts, a major advantage of the codec described above is that decoding (using Equation 5) is algorithmically simple and particularly amenable to the fast parallel-processing SIMD processors made available by recent trends. In particular, the decoding of any two data elements is computationally independent (although they may require read access to some of the same data) and requires simple operations (multiplies and adds). This has several useful consequences:

1. Data vectors can be decoded in any order. For this application, animation clips can be played forwards or backwards, can be retimed, looped, have frames skipped, et cetera, all with no penalty.
2. Data vector elements can be decoded by different processors and/or at different times. This flexibility allows different faces (or different parts of different faces) to be decoded by whatever processors happen to have available cycles.
3. Partial data vectors can be decoded. Because texture or geometry elements not visible on the screen need not be decoded, extra benefit is obtained by delaying the decode until this information is known (e.g., when decoding in a shader program in a GPU).

The multiple vector processors available on current desktop and game consoles as well as modern graphics chips all contain specialized hardware for performing these decoding operations in bulk. Implementations targeting these platforms benefit from certain restrictions on the general code such as:

1. The number of principal components used for estimation of each measurement should be a multiple of the SIMD vector length of the target processor. Such a restriction increases compression, simplifies code, and ultimately utilizes the processor more efficiently.
2. A disadvantage of this codec is that the decoding of *any* data vector requires *all* the principal components which can be a bandwidth hardship. For some systems this is handled more easily than others, but in general, careful memory and cache management is important. Additionally, the principal components (u_i) can be further compressed, typically through a simple technique such as quantization, since that is typically not the precision bottleneck.

This compression algorithm enables, besides smaller storage size, quicker transfer rates (less data to communicate) less cache misses (less data to access) and less computation of linear operations.

5 Sequencing Performances

The previous section described a novel means to compress and decompress high-fidelity facial motions at interactive rates. The fact that this technique preserves the captured subtlety and variety allows us to apply *motion graphs* to facial animation. Motions graphs (a.k.a *move trees*) are common practice in video game development and have been applied successfully to produce rich and realistic body movement from both captured and hand-animated clips. They are similar in concept to a body of graphics research (see [Kovar et al. 2002; Arikan and Forsyth 2002]). A move tree is a graph of animation clips, usually representing motions in certain contexts (like playing a particular sports game) and consists of idle loops and domain-specific actions (such as punching, running, and kicking). The graph can be arbitrarily complex and denotes allowable transitions which are triggered either by an AI sub-system or directly by the user through a game controller.

To apply the motion graph approach to facial animation

1. a library of facial expression clips, fundamental to a certain context, are captured;
2. a motion graph is designed to connect the expressions, and
3. the graph can be traversed, rendering the clips and interpolating both textures and geometry across the transitions.

5.1 Specifying Leanne's Motion Graph

Each expression clip in the library is captured in context and can become a node in a motion graph. For example, Leanne Adachi's performance was directed in the context of a kung fu fight, resulting in a library of essential related facial motions (part of which is shown in Figure 11).

Figure 11: Single frames from the expression clip (facial motion) library we captured for Leanne Adachi.

To create our motion library we used extensive movie references, came up with classifications for different kinds of moves contrasting typical fighting facial expressions for attack and defense in different mental states with fight openings and idle breathing loops. Within this classification we apply a layered approach to achieve additional variety. Each move category is represented with multiple intensities corresponding to different stages of a fight. The capture session with Leanne lasted about 3 hours with around 17min of actual footage captured. Out of this about 2 minutes of footage was selected for use which was ultimately split into 55 separate clips for inclusion in the state graph. The average number of frames/move was 65, while the median was 47 frames. The shortest move was only 27 frames and the longest 220 frames.

5.2 Interpolating Across Transitions

In general, the boundaries of two captured performances cannot be seamlessly spliced together in a trivial way. This is due to such reasons as choosing performance clips that do not both start and end in a neutral pose as well as lighting changes due to the actor's orientation changes (despite the consistent lighting within the capture environment).

To alleviate the abruptness of this artifact, a transition window (typically on the order of $1/3$ of a second long) can be introduced within which both geometry and texture contributions from either clip are linearly interpolated. If reconstructing a frame that is 100α percent through the transition, and using frame f of the original clip and frame t of the new clip, the i^{th} data element is reconstructed using the function $\text{LERP}(A, i, f, t, \alpha)$ defined in Equation 7:

$$\text{LERP}(A, i, f, t, \alpha) = (1-\alpha)\ a_{if} + (\alpha)\ a_{it} \qquad (7)$$

Figure 12: Real-time GPU shading including subsurface scattering and soft shadow approximations.

This represents considerable computation: Equation 7 is computed for every color of every pixel and every attribute of every vertex. However, because of the linearity of the compression basis change, this operation can be equivalently performed in the dimension-reduced principal component sub-space (providing both clips use the same principal components, U) as shown in Equations 8–10 (reducing the number of elements to blend from m to $\max_{i\in[1,m]} c_i$).

$$\text{LERP}(A,i,f,t,\alpha) = (1-\alpha)\sum_{k=0}^{c_i} u_{ik}s_i v_{fk} + \alpha\sum_{k=0}^{c_i} u_{ik}s_i v_{tk} \quad (8)$$

$$= \sum_{k=0}^{c_i} u_{ik}s_i\left((1-\alpha)v_{fk} + \alpha v_{tk}\right) \quad (9)$$

$$= \sum_{k=0}^{c_i} u_{ik}s_i\ \text{LERP}(V^T,k,f,t,\alpha) \quad (10)$$

Beyond computation savings, this kind of linear blending achieves surprisingly reasonable results even when we allow transitions from any state to any state, at any moment in time. We were prepared to apply techniques discussed in [Schödl et al. 2000] to find and limit the possible set of transitions to places in the clips that satisfy requirements of similar appearance and direction of movement. However, we found this to be of fairly low priority as users enjoyed the variety, freedom, and responsiveness offered by a complete motion graph.

6 Results

Several subjects and performances were captured using the techniques described in this paper. The captured data was analyzed, compressed, and ultimately rendered using a custom prototyping engine based on OpenGL ES and Cg. The following images are all direct captures from our engine where you can interactively trigger the facial expression clips from the move libraries we have captured for each character.

The engine relies on compressed data, and the Variable PCA technique described in Section 4.2 improves the final results. Ultimately, this improvement can be made both quantitative (by measuring some global error metric such as the Frobenius norm, as in Figure 13). and qualitative (by observing the effect of equivalent compression on identified trouble areas, as in Figure 14)

Although re-lighting of faces is a difficult problem beyond the subject of this paper, our technique enables two solutions: the performance can be captured in ambient lighting and the high-detail normals from the head scan can be used to light and shade using a style suitable to the environment, or the problem can be avoided by capturing the data in the desired lighting. To handle the re-lighting issue using a combination of advanced shading techniques suitable to modern GPU hardware, we treat the captured color textures as diffuse albedo maps which already contain baked-in ambient occlusion. For reflectance we rely on our highly detailed normal maps perturbing fairly simple, but carefully tuned analytical model with Lambertian diffuse component and a modified Blinn-like specular component with a Fresnel-like effect. This is similar to the approach of [Borshukov and Lewis 2003] who also introduced the subsurface scattering approximation we use (see [Sander et al. 2004; Fernando 2004] for real-time implementation), which is crucial for close-ups of realistic skin as shown by [Jensen et al. 2001]. Shadows from "key" light sources are another important effect and we found that the efficient soft-edged shadows using pixel shader branching described in [Pharr 2005] worked well in our case. For the "fill" component of the lighting we take advantage of pre-blurred diffuse and specular high dynamic range maps.

Figure 12 shows images rendered at interactive (real-time) rates on modern hardware that demonstrate the success of these approaches in combination with our hi-resolution data assets. Figures 15 and 16 show stills from different subjects we have captured and processed including sports celebrities.

7 Future Work

One of the areas of future work we are pursuing actively with very promising results is fully automating the mouth and tongue tracking by estimation of bone positions thru minimizing the difference error between the input images and the rendered in each camera view model with static texture.

Another area of research is the change of eye gaze direction which would add another dimension of flexibility to our approach. We want to do this without sacrificing any visual fidelity, because the eyes are the most important part of the face and we owe a large part of the success of our results to deciding not to separate the eyes

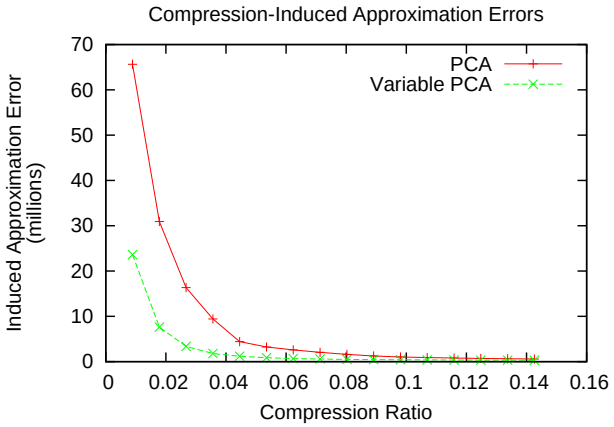


Figure 13: Approximation errors (in Frobenius norm) comparing the standard and Variable PCA representations for several performances and compression levels.

Figure 14: There are locations where violated PCA assumptions have resulted in compression artifacts. Each row above shows one such area in one performance (with left to right columns representing the captured data, PCA-compressed data, and Variable PCA-compressed data respectively).

from the captured data. The approach we like to pursue will still take advantage of captured geometry and texture data (from certain eye movement exercises) but parameterize it in such a way that we can drive it by procedural techniques or intuitive artistic controls.

Currently, although we have captured and processed visemes, we have not yet experimented with interactive speech synthesis by blending these short viseme clips, but we believe this approach will produce excellent results as demonstrated by [Bregler et al. 1997] and [Ezzat et al. 2002].

8 Conclusion

This paper introduced a production-level animated facial geometry and texture capture system and showed how to use the high-fidelity captured assets in compelling real-time interactive applications. This achievement relied upon a novel variant of PCA compression that varies the number of components used to represent each data measurement. The new compression technique achieves better results for the same compression ratio than standard PCA, and the linear reconstruction of compressed data is particularly amenable to current SIMD architectures achieving an acceptable trade-off between bandwidth, computation, and final visual fidelity.

The quality of data this provides when rendering in real-time, combined with advanced GPU shading techniques, achieves realistic and engaging visual experiences. This enabled the application of motion graphs to the domain of facial animation. The paper also demonstrated techniques for blending across arbitrary transitions of such motion graphs. Ultimately, this collection of techniques will

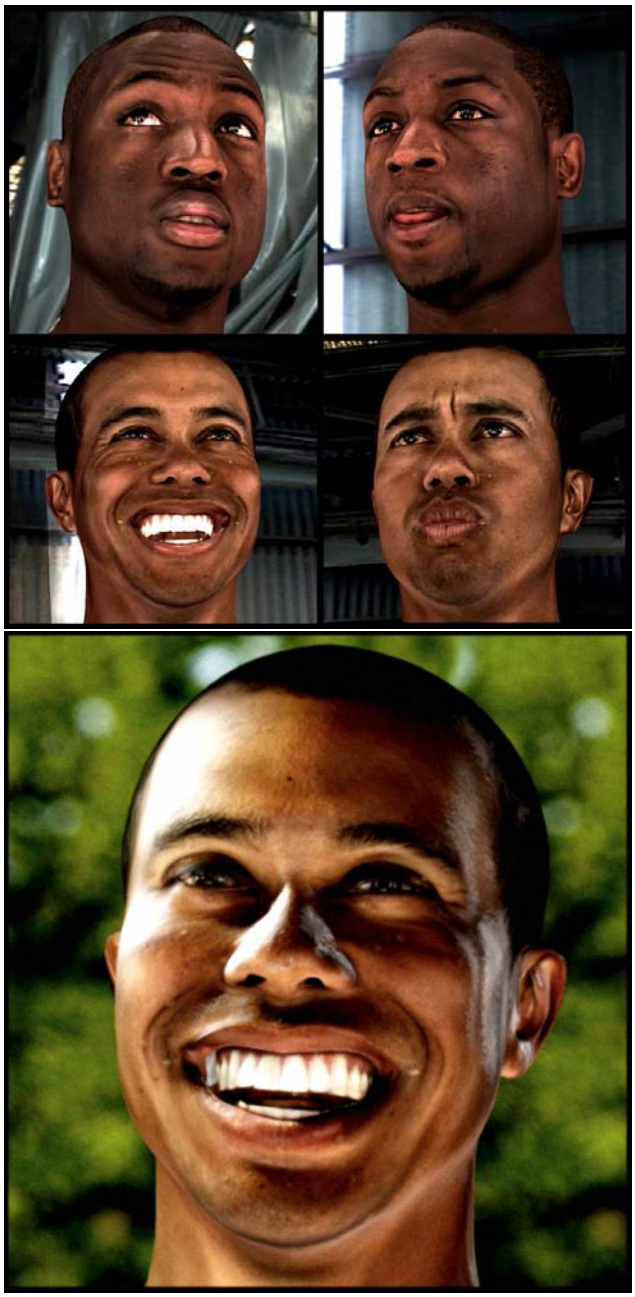


Figure 15: Dwyane Wade (top) was captured in specific lighting and is shown here with no additional shading (only the captured texture); and Tiger Woods was captured and is presented in more neutral lighting (middle) and outdoor lighting (bottom).

help human characters in video games and other real-time applications to approach the believability and emotional impact offered by their film counterparts.

9 Acknowledgements

We would like recognize the following people who made this work possible: James Lau and Kevin Noone who build the models for the results presented in the paper, Paul Thuriot for his important contributions to the facial rigging process, Barry Ruff for his work on the marker removal and shading techniques, Patrick Mooney who handled the organizational aspects of the project, Dave Raposo who

helped with processing the data, as well as Stefan Van Niekerk and the staff at EA Mocap. Special thanks go to James Grieve and Paul Lalonde for their work on an early prototype of the PCA compression and Glenn Entis for his continual support of this work.

References

ARIKAN, O., AND FORSYTH, D. 2002. Interactive motion generation from examples. In *Proceedings of SIGGRAPH 2002*, 483 – 490.

BLANZ, V., AND VETTER, T. 1999. A morphable model for the synthesis of 3d faces. In *Proceedings of SIGGRAPH 99*, 187 – 194.

BLANZ, V., BOSSO, C., POGGIO, T., AND VETTER, T. 2003. Reanimating faces in images and video. In *Computer Graphics Forum, Vol. 22, No. 3 EUROGRAPHICS 2003*, 641 – 650.

BORSHUKOV, G., AND LEWIS, J. P. 2003. Realistic human face rendering for "The Matrix Reloaded". In *Proceedings of SIGGRAPH 2003 Sketches and Applications*.

BORSHUKOV, G., PIPONI, D., LARSEN, O., LEWIS, J. P., AND TEMPELAAR-LIETZ, C. 2003. Universal Capture: Image-based facial animation for "The Matrix Reloaded". In *Proceedings of SIGGRAPH 2003 Sketches and Applications*.

BORSHUKOV, G., SABOURIN, K., SUZUKI, M., LARSEN, O., MIHASHI, T., FAIMAN, K., SCHINDERMAN, S., JAMES, O., AND JACK, J. 2004. Making of The Superpunch. In *Proceedings of SIGGRAPH 2004 Sketches and Applications*.

BREGLER, C., COVELL, M., AND SLANEY, M. 1997. Video rewrite: Driving visual speech with audio. In *Proceedings of SIGGRAPH 97*, 353 – 360.

BREGLER, C., HERTZMANN, A., AND BIERMANN, H. 2000. Recovering non-rigid 3d shape from image streams. In *Proceedings of IEEE CVPR 2000*.

DECARLO, D., AND METAXAS, D. 1996. The integration of optical flow and deformable models with applications to human face shape and motion estimation. In *Proceedings of CVPR*, 231 – 238.

ESSA, I., AND PENTLAND, A. 1997. Coding, analysis, interpretation and recognition of facial expressions. *IEEE Transactions on Pattern Analysis and Machine Intelligence 19*, 7, 757 – 763.

ESSA, I., BASU, S., DARRELL, T., AND PENTLAND, A. 1996. Modeling, tracking and interactive animation of faces and heads using input from video. In *Proceedings of the Computer Animation, IEEE Computer Society*, 68 – 79.

EZZAT, T., GEIGER, G., AND POGGIO, T. 2002. Trainable videorealistic speech animation. In *Proceedings of SIGGRAPH 2002*, 388 – 398.

FERNANDO, R. 2004. *GPU Gems*. Addison-Wesley.

GUENTER, B., GRIMM, C., WOOD, D., MALVAR, H., AND PIGHIN, F. 1998. Making faces. In *Proceedings of SIGGRAPH 98*, 55 – 66.

HOTELLING, H. 1933. Analysis of a complex of statistical variables with principle components. *Journal of Educational Psychology 24*, 498 – 520.

HYNEMAN, W., ITOKAZU, H., WILLIAMS, L., AND ZHAO, X. 2005. Human face project. In *Proceedings of SIGGRAPH 2005 Courses: Digital Face Cloning*, 29 – 46.

JENSEN, H. W., MARSCHNER, S. R., LEVOY, M., AND HANRAHAN, P. 2001. A practical model for subsurface light transport. In *Proceedings of SIGGRAPH 2001*, 511 – 518.

KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion graphs. In *Proceedings of SIGGRAPH 2002*, 473 – 482.

LEE, Y., TERZOPOULOS, D., AND WATERS, K. 1993. Constructing physics-based facial models of individuals. In *Proceedings of Graphics Interface 93*, 1 – 8.

LEE, Y., TERZOPOULOS, D., AND WATERS, K. 1995. Realistic modeling for facial animation. In *Proceedings of SIGGRAPH 95*, 55 – 62.

MORI, M. 1982. *The Buddha in the Robot*. Tuttle.

PARKE, F. 1972. Computer generated animation of faces. In *Proceedings ACM annual conference*.

PARKE, F. 1974. *A parametric model for human faces*. PhD thesis, University of Utah, Salt Lake City, Utah.

PHARR, M. 2005. *GPU Gems 2*. Addison-Wesley.

PIGHIN, F., HECKER, J., LISCHINSKI, D., SZELISKI, R., AND SALESIN, D. 1998. Synthesizing realistic facial expressions from photographs. In *Proceedings of SIGGRAPH 98*, 75 – 84.

PIGHIN, F., SZELISKI, R., AND SALESIN, D. 1999. Resynthesizing facial animation through 3d model-based tracking. In *Internation Conference on Computer Vision (ICCV)*, 143 – 150.

REVERET, L., AND ESSA, I. 2001. Visual coding and tracking of speech related facial motion. In *Proceedings of the IEEE CVPR International Workshop on Cues in Communication*.

SANDER, P. V., GOSSELIN, D., AND MITCHELL, J. L. 2004. Real-time skin rendering on graphics hardware. In *Proceedings of SIGGRAPH 2004 Sketches and Applications*.

SCHÖDL, A., SZELISKI, R., SALESIN, D. H., AND ESSA, I. 2000. Video textures. In *Proceedings of SIGGRAPH 2000*, 489 – 498.

SIFAKIS, E., NEVEROV, I., AND FEDKIW, R. 2005. Automatic determination of facial muscle activations from sparse motion capture marker data. In *Proceedings of SIGGRAPH 2005*, 417 – 425.

SIROVICH, L., AND KIRBY, M. 1987. Low-dimensional procedure for the characterization of human faces. *Journal of Optical Society of America 4*, 3 (March), 519 – 524.

STRANG, G. 1980. *Linear Algebra and its Applications*, 2 ed. Academic Press, New York.

TERZOPOULOS, D., AND WATERS, K. 1991. Techniques for realistic facial modeling and animation. In *Computer Animation 91*, Springer-Verlag, 59 – 74.

TURK, M., AND PENTLAND, A. 1991. Face recognition using eigenfaces. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 586 – 591.

VLASIC, D., BRAND, M., PFISTER, H., AND POPOVIC, J. 2005. Face transfer with multilinear models. In *Proceedings of SIGGRAPH 2005*, 426 – 433.

WATERS, K. 1987. A muscle model for animating three-dimensional facial expression. In *Proceedings of SIGGRAPH 87*, 17 – 24.

WILLIAMS, L. 1990. Performance-driven facial animation. In *Proceedings of SIGGRAPH 90*, 235 – 242.

ZHANG, L., SNAVELY, N., CURLESS, B., AND SEITZ, S. M. 2004. Spacetime faces: High-resolution capture for modeling and animation. In *Proceedings of SIGGRAPH 2004*, 548 – 558.

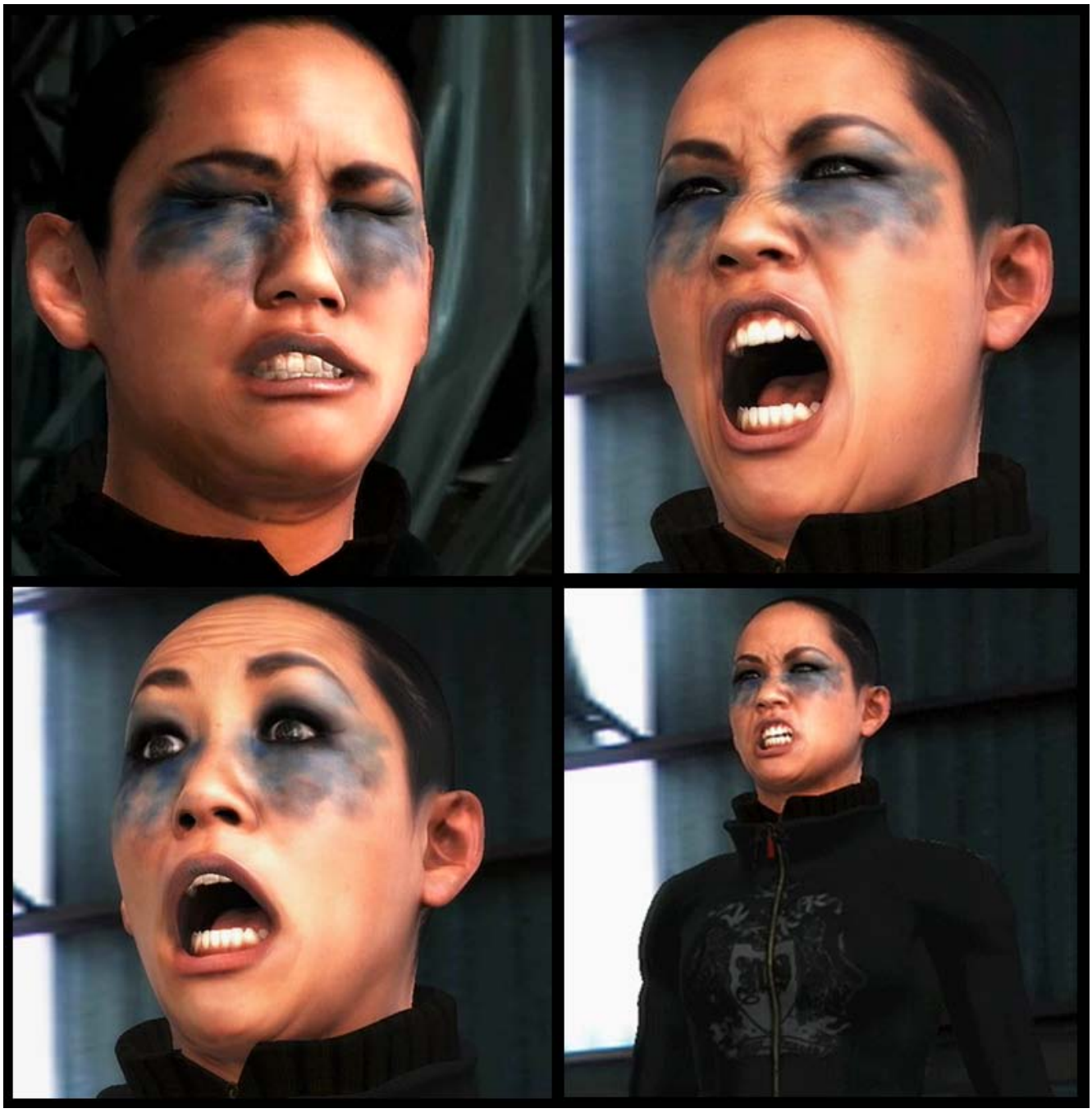


Figure 16: Leanne Adachi was captured in make-up and demonstrates a highly shaded, somewhat stylized look.

Performance-Driven Facial Animation

Chris Bregler
George Borshukov
Parag Havaldar
J.P.Lewis
Fred Pighin
Jim Radford
Mark Sagar
Steve Sullivan
Li Zhang

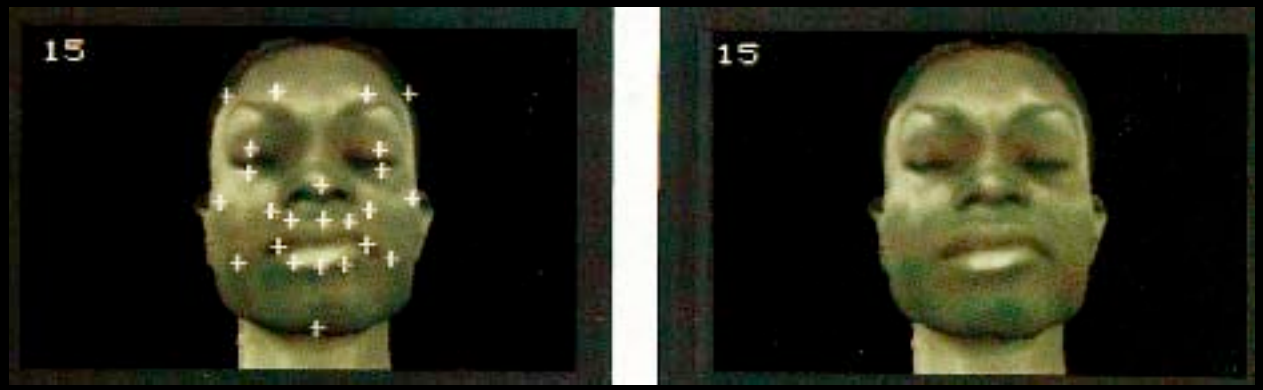
…and Thomas Kang!

Performance-Driven Facial Animation

- Use the human face itself as the “input device”
- Motivations:
  - Much easier to produce a facial expression than to adjust sliders
  - Some people believe that keyframe animation cannot consistently capture the subtleties of human motion
  - More people believe that some animators can achieve realistic motion, but that getting consistent results is time consuming and expensive.

Early History

- Lance Williams, Performance-Driven Facial Animation, SIGGRAPH 1990

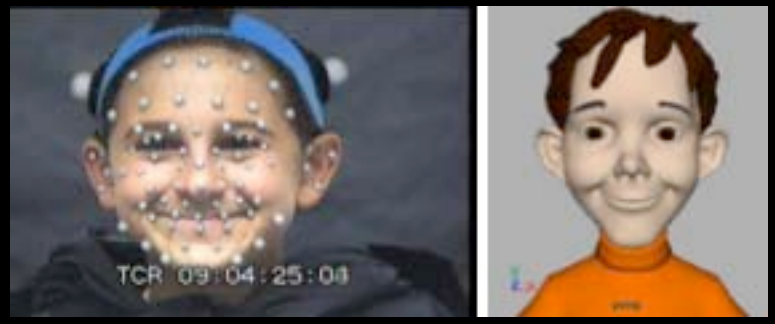


- SimGraphics “face waldo” demonstration at SIGGRAPH 1992

Face Tracking Approaches

- Photogrammetry, stereo

Face Tracking Approaches



- Marker-based hardware motion capture systems
- Tom Tolles (House of Moves) presentation 9:00 (next)
- Parag Havaldar (Sony Pictures Imageworks) presentation at 2:15 pm

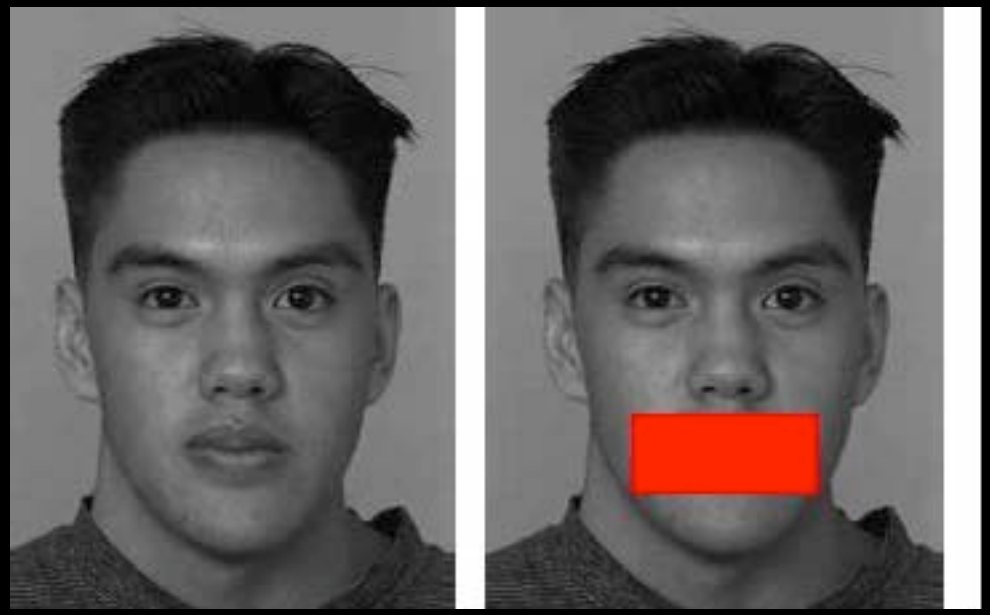
Structured Light

SIGGRAPH2006

- Eyetronics - used in Discovery Channel's *Virtual History* (Jim Radford presentation at 4:00 pm)

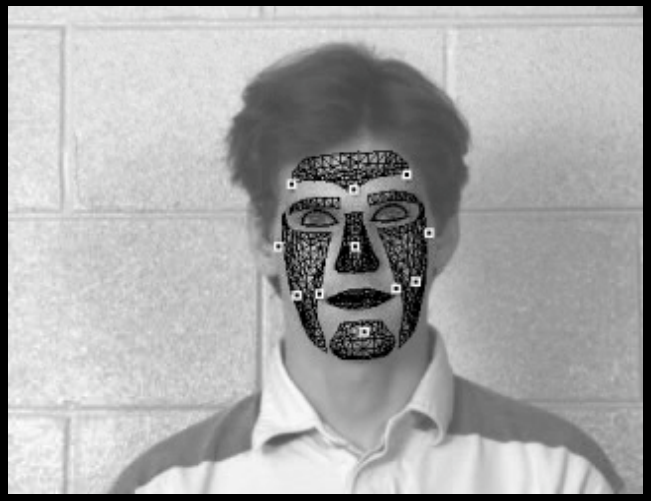
- Li Zhang - Space-time stereo (presentation at 11:15)

Appearance Models

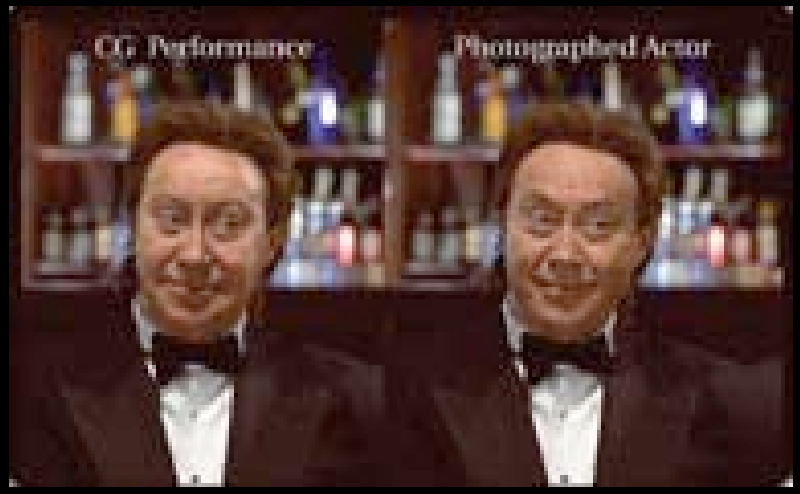



- Discussed in Chris Bregler's presentation, 11:15 am

Model-Based Tracking

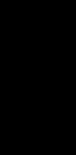


DeCarlo, Metaxas, 1999

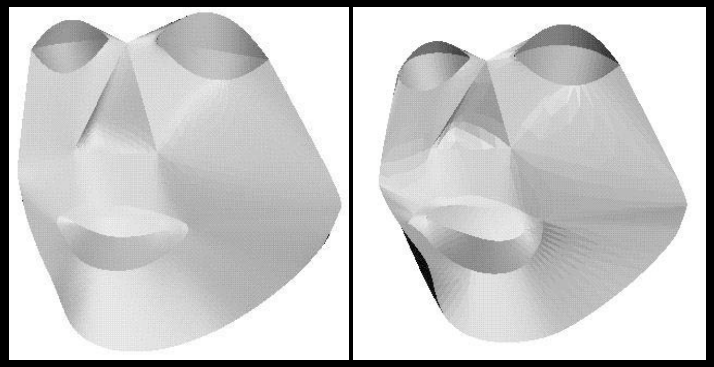


Disney Facial Test, 2002

Automatic derivation of models from video



- Chris Bregler's presentation, 11:15 am



PFDA in Entertainment

PFDA in Entertainment

- Movie industry tests: LifeFX, ILM's Hugo,

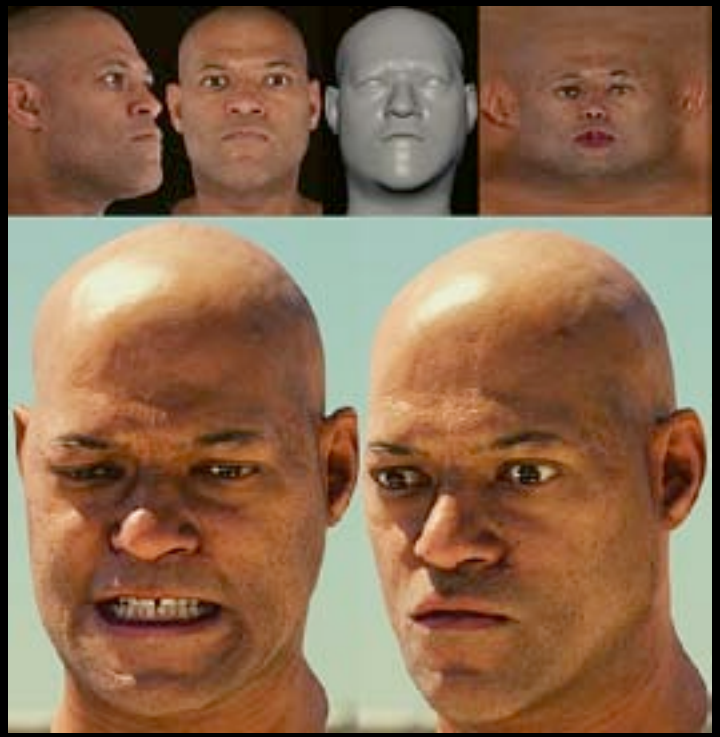
Did not use PDFA…

- *Final Fantasy* (2000): bodys were mocap, faces were manually animated
- *Lord of the Rings*: Gollum was manually animated, but based heavily on video reference -- “roto PFDA”?

… see Mark Sagar’s presentation on PFDA in King Kong at 3:00pm

The Matrix Reloaded (2003)

- Further development this “Ucap” approach at Electronic Arts: see George Borshukov’s presentation at 4:45



The Polar Express (2004)

- First project where PFDA used exclusively for main characters
- See Parag Havaldar's presentation on PFDA at Sony (*Monster House*, …) at 2:15 pm

Schedule

8:30	Introduction and Overview Fred Pighin, Industrial Light + Magic
9:00	Facial Motion Capture in Production Parag Havaldar, Sony, and Tom Tolles, House of Moves
10:00	Break
10:15	Facial Retargeting J.P. Lewis, Stanford, and Fred Pighin, ILM
11:15	Markerless Face Capture and Automatic Model Construction Chris Bregler, NYU, and Li Zhang, Columbia
12:15	Lunch
1:30	Performance Driven Facial Animation at ILM Steve Sullivan and Christophe Hery, ILM
2:15	Monster House Parag Havaldar, Sony
3:00	King Kong Mark Sagar, Weta
4:00	Virtual History - Jim Radford, Moving Picture Company, Face Robot - Thomas Kang, Softimage
4:45	Playable Universal Capture at Electronic Arts George Borshukov, EA
5:15	Panel on the future of performance-driven animation all speakers

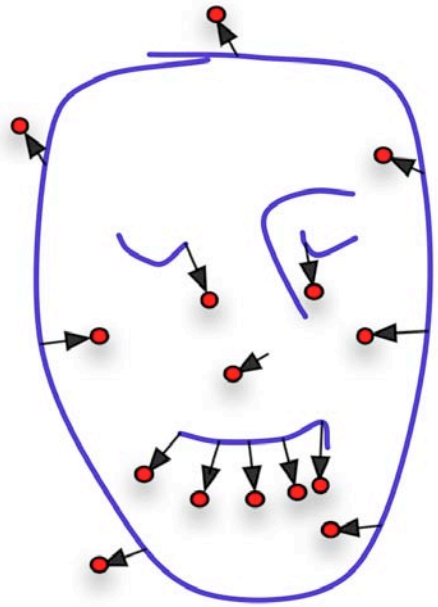
Performance-driven facial animation: background mathematics

J.P. Lewis and Fred Pighin

Background

Scattered Data Interpolation

- Interpolate data at arbitrary (irregularly spaced) locations
- Standard application: warp generic face mesh to fit mocap markers

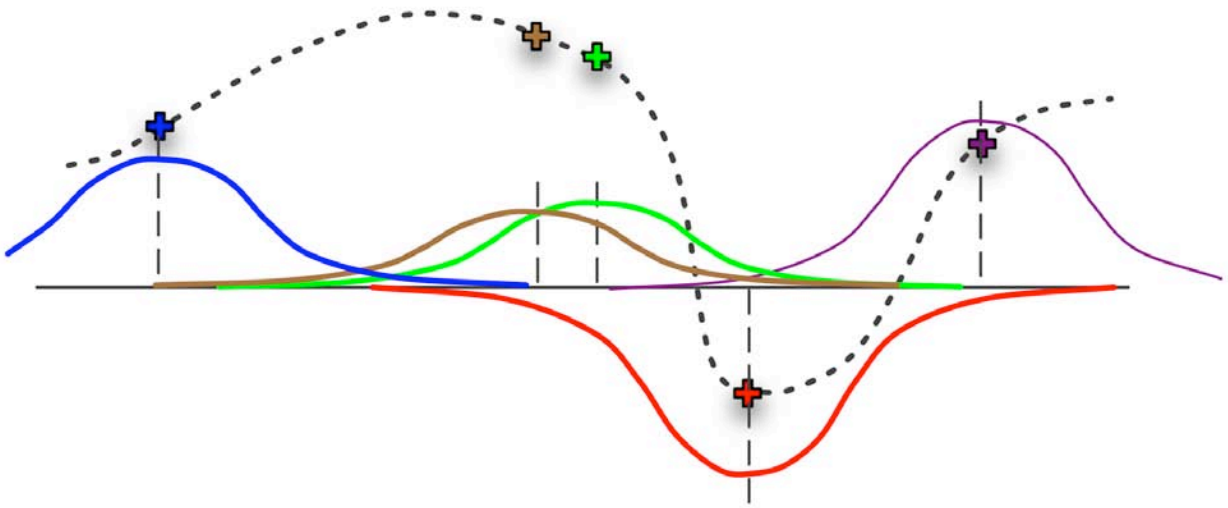


Background

Radial Basis Functions

- Data at arbitrary (irregularly spaced) locations can be interpolated with a weighted sum of radial functions situated at each data point.

$$\hat{d}(\mathbf{x}) = \sum_{k}^{N} w_k R(\|\mathbf{x} - \mathbf{x}_k\|)$$



Background

Radial Basis Functions

- *Any* function other than constant can be used!
- Common choices:
  - Gaussian: $R(r) = \exp(-r^2/\sigma^2)$
  - Thin plate spline: $R(r) = r^2 \log r$ (in 2D)
  - Hardy Multiquadratic: $R(r) = \sqrt{(r^2 + c^2)}, c > 0$
- Notice: the last two *increase* as a function of radius

Warping Facial Geometry

$$d(\mathbf{x}) = \sum_{k}^{N} w_k R(\|\mathbf{x} - \mathbf{x}_k\|)$$

For warping facial geometry:

d(x) is the x,y, or z displacement between a point and where it should go, e.g. between a model point and the corresponding mocap marker location

Background

Radial Basis Functions

- Given data points $d_k(x_k)$ to interpolate, solving for the weights is just a matrix inverse:

$$d(\mathbf{x}) = \sum_k^N w_k R(\|\mathbf{x} - \mathbf{x}_k\|)$$

$$e = ||(\mathbf{d} - \mathbf{R}\mathbf{w})||^2$$

$$e = (\mathbf{d} - \mathbf{R}\mathbf{w})^T(\mathbf{d} - \mathbf{R}\mathbf{w})$$

$$\frac{de}{d\mathbf{w}} = 0 = -\mathbf{R}^T(\mathbf{d} - \mathbf{R}\mathbf{w})$$

$$\mathbf{R}^T\mathbf{d} = \mathbf{R}^T\mathbf{R}\mathbf{w}$$

$$\mathbf{w} = (\mathbf{R}^T\mathbf{R})^{-1}\mathbf{R}^T\mathbf{d} = \mathbf{R}^{-1}\mathbf{d}$$

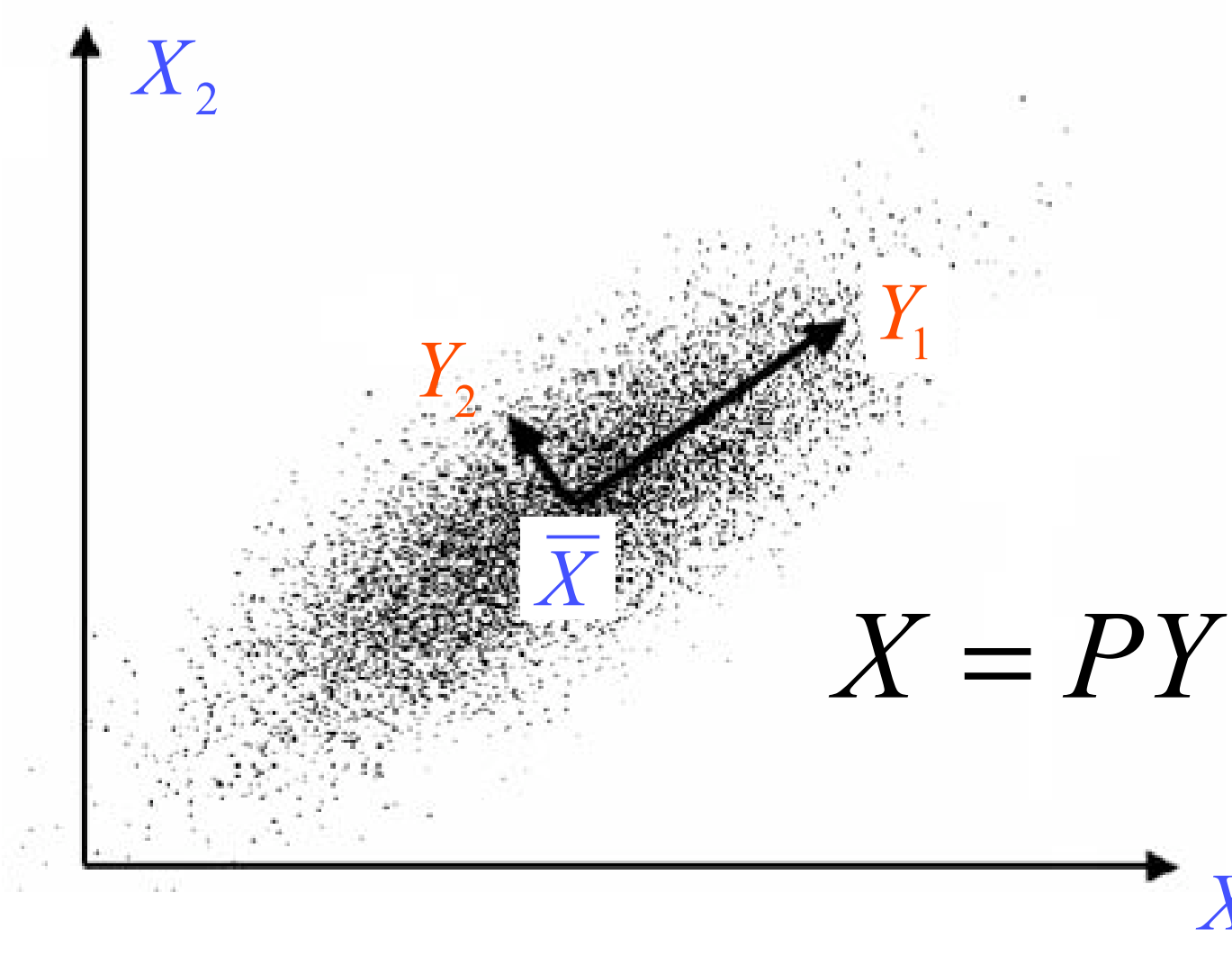
Background

Principal Component Analysis

Principal Component Analyis (PCA):

- Gives a best cartesian coordinate system for your data
- Is a method for dimensionality reduction
- Automatically makes a linear (blendshape-like) model of data

Example



$$X = PY + \overline{X}$$

Mathematical Formulation

1. Data centering $X' = X - \overline{X}$
2. Computing new basis

$$\text{Covariance matrix}: \; Cov = [c_{ij}]$$

$$\text{where } c_{ij} = \sum_{\text{all samples}} x'_i x'_j$$

The eigenvectors of *Cov* are the vector of the PCA basis $\{Y\}$

Limitations of PCA

- Linear model (linear meaning the subspace consists of a line, or a plane, or …)
- Not ideal if data is not Gaussian distributed

Manifold Learning

- Generalize away from the linear (line, plane, …subspace) restriction of PCA
- "Manifold": a subspace with continuity and smoothness properties. Essentially, *a surface in a higher dimensional space.*

Manifold Learning

Dimensional Thinking:

- A “dimension” is anything that you can independently change
- 3D space: every point describe by 3 numbers, (x,y,z).
- Image space: 1000x1000 “megapixel” image has one million pixels. A particular image is described by one million numbers (ignore RGB, say a greyscale image)

→ An image is a point in a million-dimensional space.

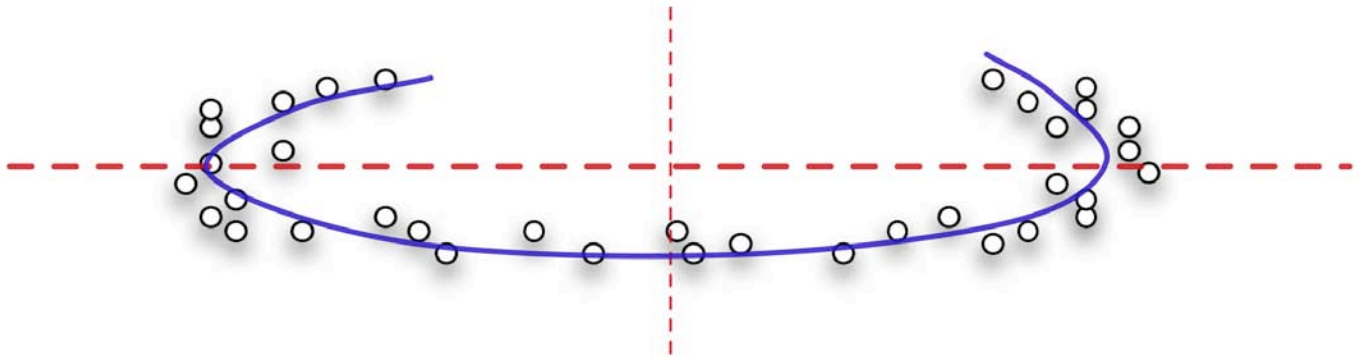
Manifold Learning

- (recall) An image is a point in a (e.g.) million-dimensional space
- Animate an image slightly and the “point” moves.
- Image sequence of a moving head (for example): corresponds to movement along a one-dimensional curve in the million dimensional space.
- Manifold learning: discover this curve
- See Expression/Style mapping section of Cross-Mapping session this morning

Manifold Learning

Red dashed: principal component analysis,

Blue solid: manifold learning



Manifold Learning

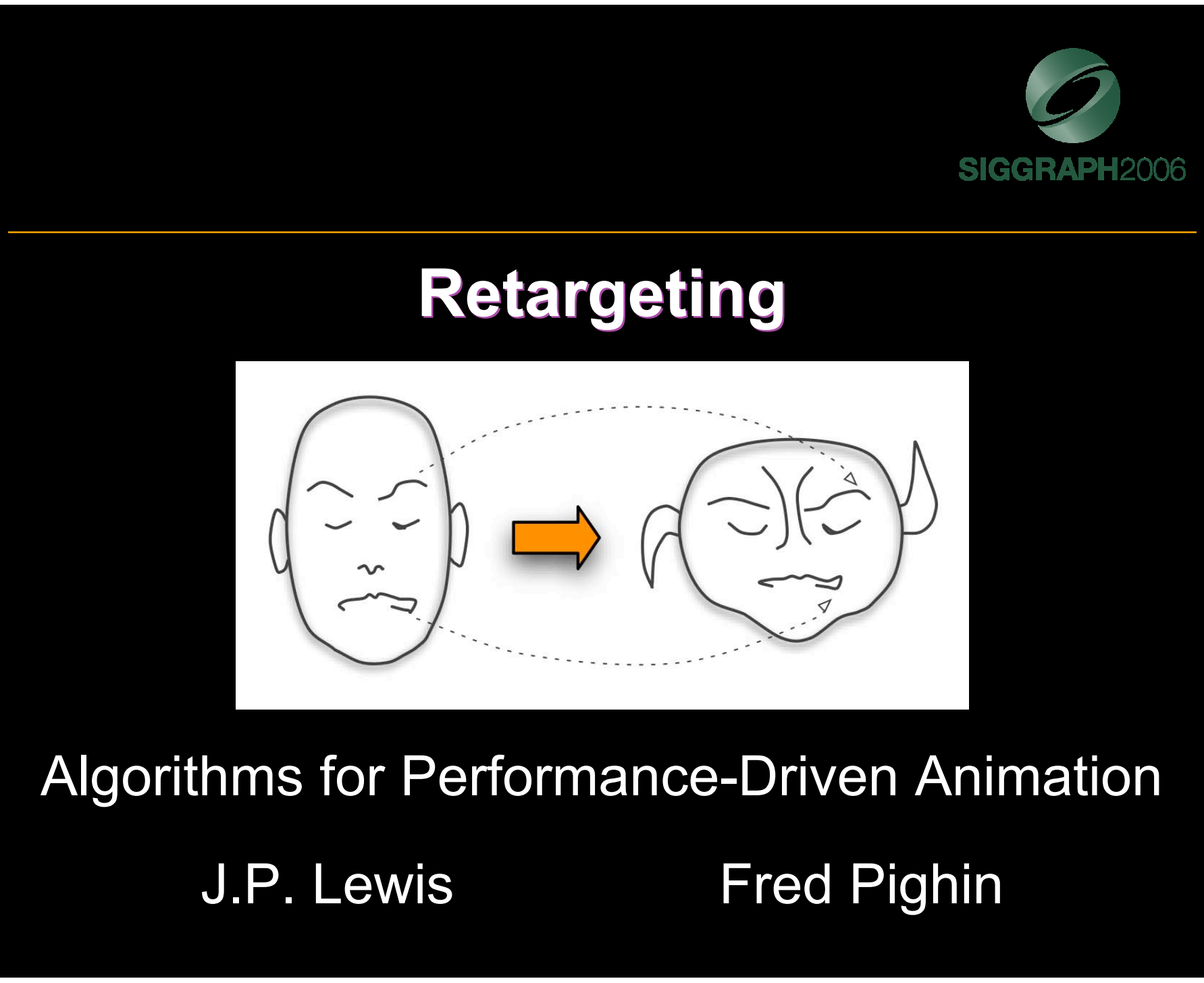
Multidimensional Scaling (MDS): a linear predecessor

- Algorithms:
  - Local Linear Embedding (LLE),
  - Isomap (non-linear version of MDS)

  } The two original algorithms

  - Laplacian Eigenmaps: related to LLE, clarified and simpler
  - Hessian Eigenmaps
  - Kernel PCA
  - Gaussian Process latent variable model …

- General approach: find a placement of points in a low-dimensional space (the manifold) such that the distance between points is proportional to the distance between the original points in the high dimensional space.


SIGGRAPH2006
Retargeting
Algorithms for Performance-Driven Animation
J.P. Lewis
Fred Pighin

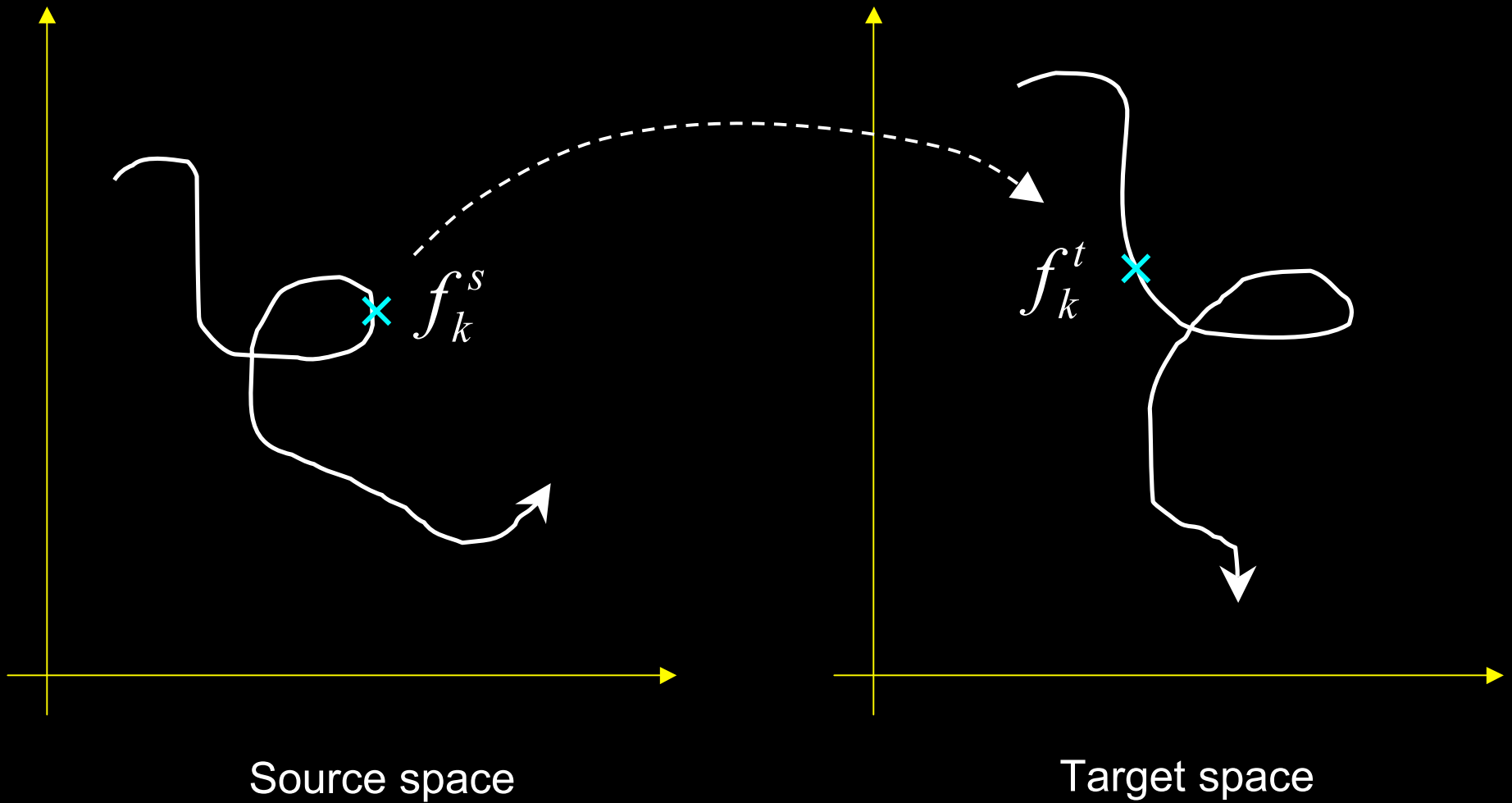
"Don't cross the streams." (Ghostbusters)

- Why cross-mapping?
  - Different character
  - Imperfect source model
- Also known as:
  - Performance-driven animation
  - Motion retargeting

Performance Cloning History

- L. Williams, Performance-driven Facial Animation, *SIGGRAPH* 1990
- SimGraphics systems, 1992-present
- LifeFX “Young at Heart” in Siggraph 2000 theater
- J.-Y. Noh and U. Neumann, Expression Cloning, *SIGGRAPH* 2001
- B. Choe and H. Ko, “Muscle Actuation Basis”, *Computer Animation* 2001 (used in Korean TV series)
- Wang et. al., *EUROGRAPHICS* 2003
- *Polar Express* movie, 2004

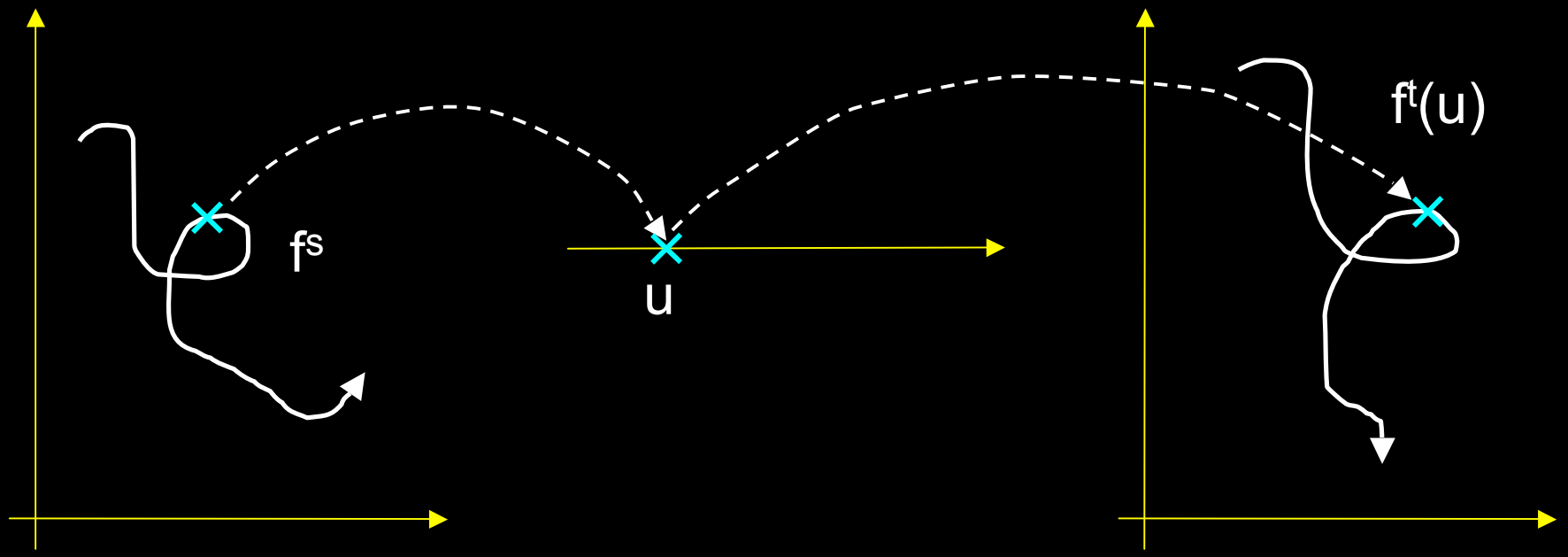
Face space mapping



Parameterizing the target space: a rig

- A facial rig defines a set of parameters/controllers for the face
- Interpolation in parameter space generates “valid” expressions

Face space mapping with rig

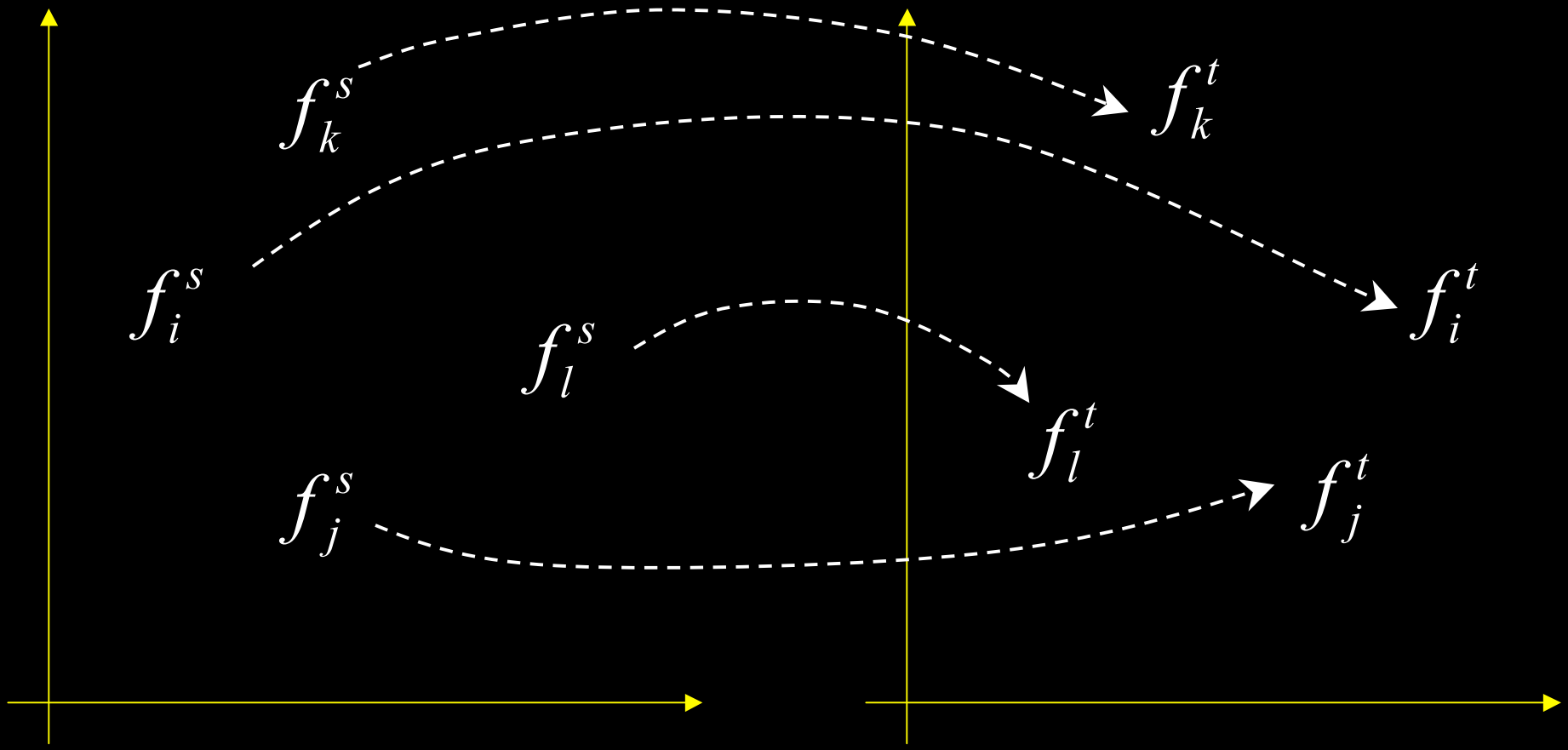


Source space

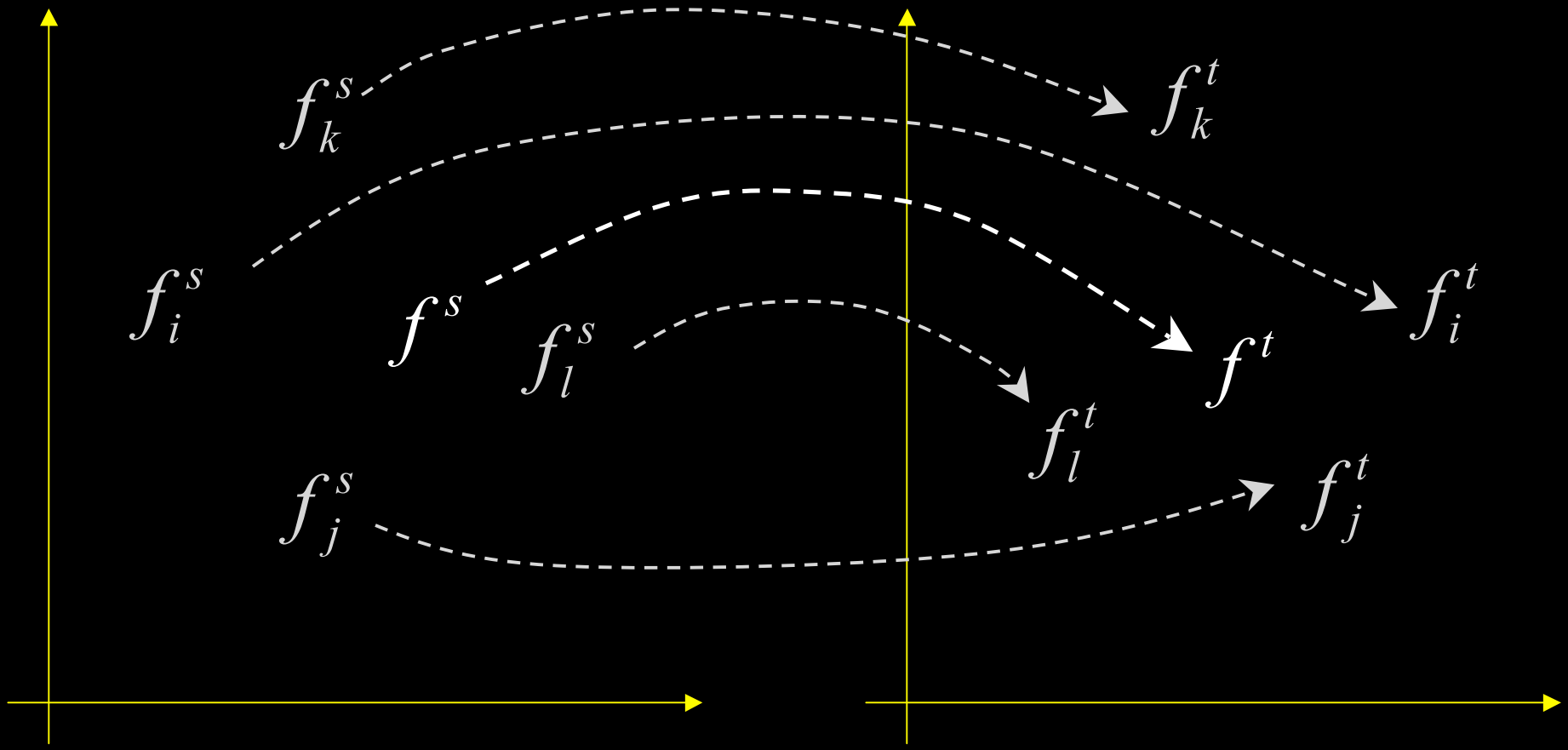
Target parameter space

Target space

Building a map from correspondences



Building a map from correspondences



$$f^t = M\left(f^s \mid (f_i^s, f_i^t), i \in \{1..N\}\right)$$

Main issues

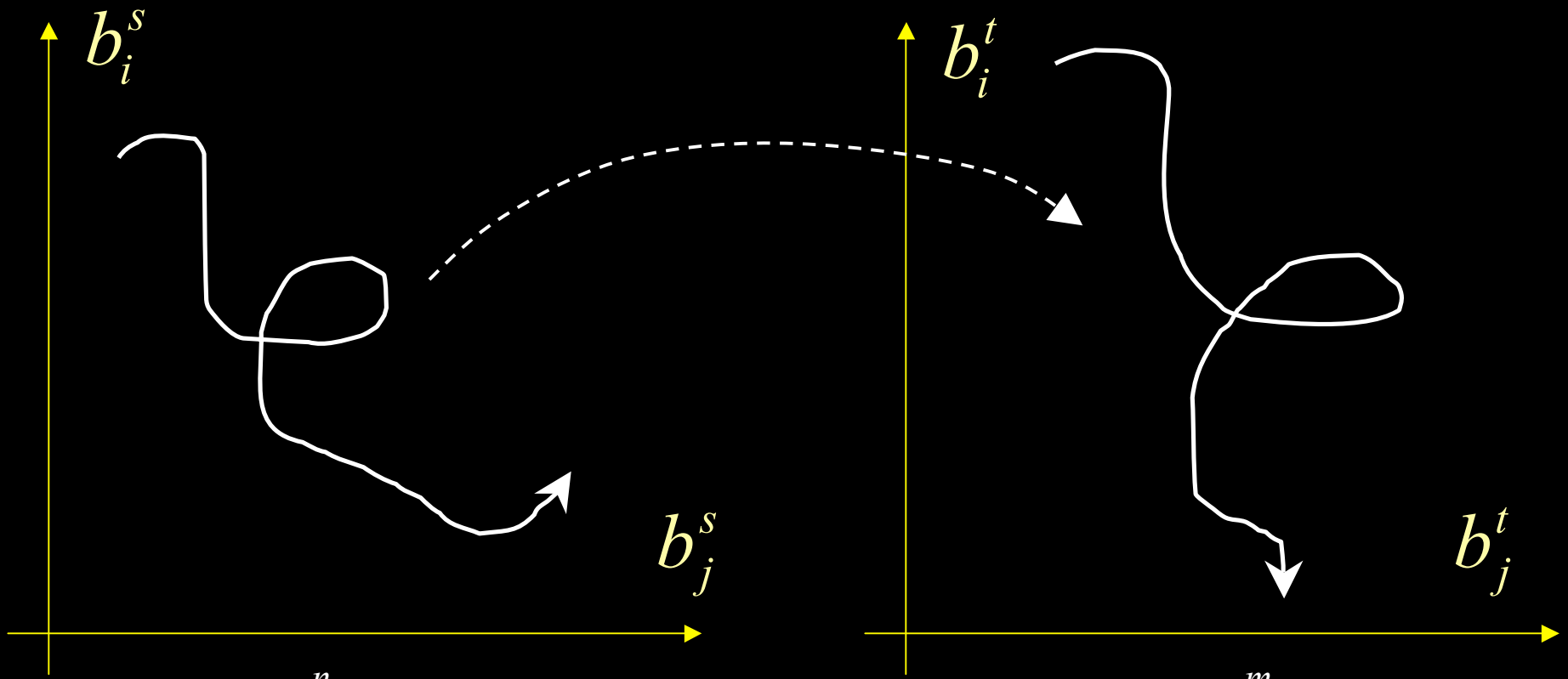
- How are the corresponding faces created?
- How to build mapping from correspondences?

Linear vs. non-linear mapping

- Linear
  - **Global blend-shape mapping**
    - B. Choe and H. Ko, **Analysis and Synthesis of Facial Expressions with Hand-Generated Muscle Actuation Basis**, *Computer Animation* 2001.
    - E. Chuang and C. Bregler, **Performance Driven Facial Animation using Blendshape Interpolation**, CS-TR-2002-02, Department of Computer Science, Stanford University
- Non-linear
  - **Piece-wise blend-shape mapping**
    - I. Buck, A. Finkelstein, C. Jacobs, A. Klein, D. H. Salesin, J. Seims, R. Szeliski, and K. Toyama, **Performance-driven hand-drawn animation**, *NPAR 2000.*
    - J.-Y. Noh and U. Neumann, **Expression Cloning**, *SIGGRAPH* 2001.
  - **Manifold learning**
    - Y. Wang, X. Huang, C.-S. Lee, S. Zhang, D. Samaras, D. Metaxas, A. Elgammal, and P. Huang, **High Resolution Acquisition, Learning, and Transfer of Dynamic 3-D Facial Expressions**, *Eurographics* 2004.
  - **Single-correspondence mapping**
    - J.-Y. Noh and U. Neumann, **Expression Cloning**, *SIGGRAPH* 2001.

Linear Mapping

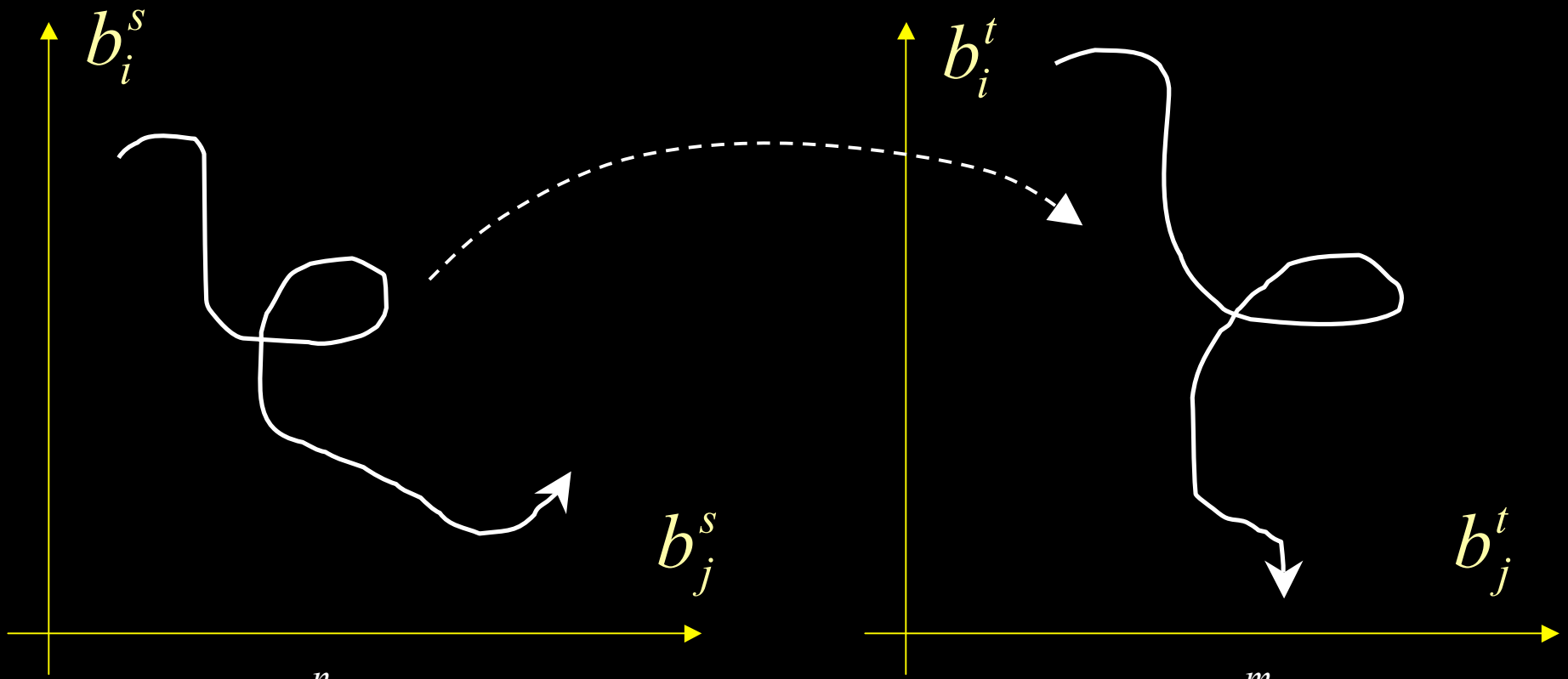
Blendshape mapping



$$f_k^s = \sum_{i=1}^{n} w_{k,i}^s b_i^s \longrightarrow f_k^t = \sum_{i=1}^{m} w_{k,i}^t b_i^t$$

(Global) Change of coordinate system

Blendshape mapping



$$f_k^s = \sum_{i=1}^{n} w_{k,i}^s b_i^s \longrightarrow f_k^t = \sum_{i=1}^{m} w_{k,i}^t b_i^t$$

(Global) Change of coordinate system

Blendshapes: definition

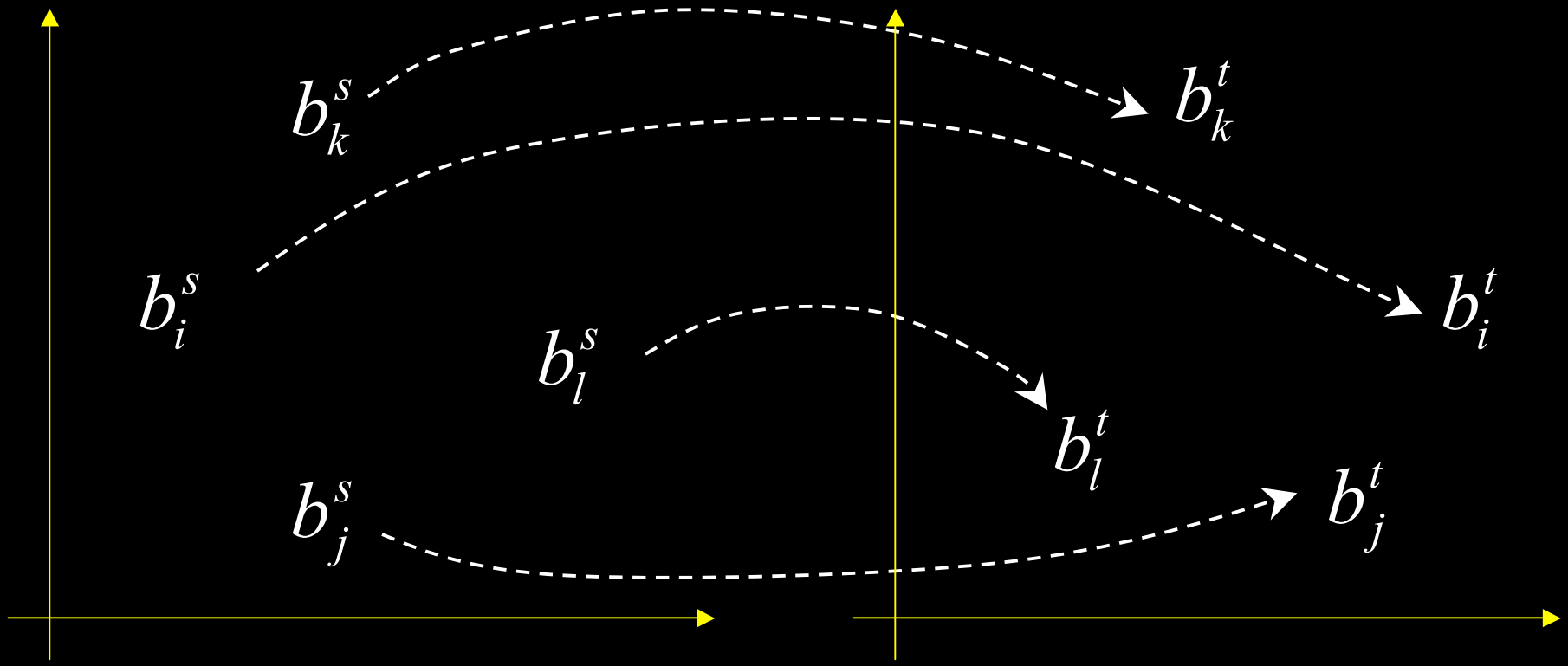
- per-vertex view: $\vec{\mathbf{f}}_k = \sum w_k \vec{\mathbf{b}}_k$

	Industry term	Math usage
w_k	Slider values	weights
b_k	Blendshape target	Basis vector

Blendshapes: definition

- Linear algebra of blendshapes:
- per-vertex view: $\vec{\mathbf{f}}_k = \sum w_k \vec{\mathbf{b}}_k$
- global view: $\mathbf{f} = \mathbf{B}\mathbf{w}$
- **f** : 3*n* x 1 vector containing all *n* vertices of the face, in some packing order e.g. *xyzxyzxyz....*
- **B**: 3*n* x *m* matrix; each column is one of the *m* blendshapes, using the same packing order.
- **w**: vector of *m* weights, animated over time

Parallel blendshapes



Use same blending weights: $w_{k,i}^t = w_{k,i}^s$

Parallel Model Construction

- Have similar blendshape controls in source, target models
- Advantage: conceptually simple
- Disadvantage: twice the work (or more!) -- unnecessary!
- Disadvantage: cannot use PCA

Solutions

- Adapt generic model to source (Choe et. al.)
- Derive source basis from data (Chuang and Bregler)
- Allow different source, target basis

Source model adaptation

- B. Choe and H. Ko, **Analysis and Synthesis of Facial Expressions with Hand-Generated Muscle Actuation Basis**, *Computer Animation* 2001
- Cross-mapping obtained simply by constructing two models with identical controls.
- Localized (delta) blendshape basis inspired by human muscles
- Face performance obtained from motion captured markers

Muscle actuation basis

- Model points corresponding to markers are identified
- Blendshape weights determined by least-squares fit of model points to markers
- Fit of model face to captured motion is improved with an alternating least squares procedure

Muscle actuation basis

- Fitting the model to the markers:
- alternate 1), 2)
  - 1) solve for weights given markers and corresponding target points
  - 2) solve for target points location
- warp the model geometry to fit the final model points using radial basis interpolation.

Muscle actuation basis

- Fitting the model to the markers:

$$f_k = Bw_k \quad \text{for all frames k}$$

$$F = BW \quad \text{stack all f,w in matrices}$$

- Alternate: solve for B, solve for W
- warp the model geometry to fit the final model points using radial basis interpolation.

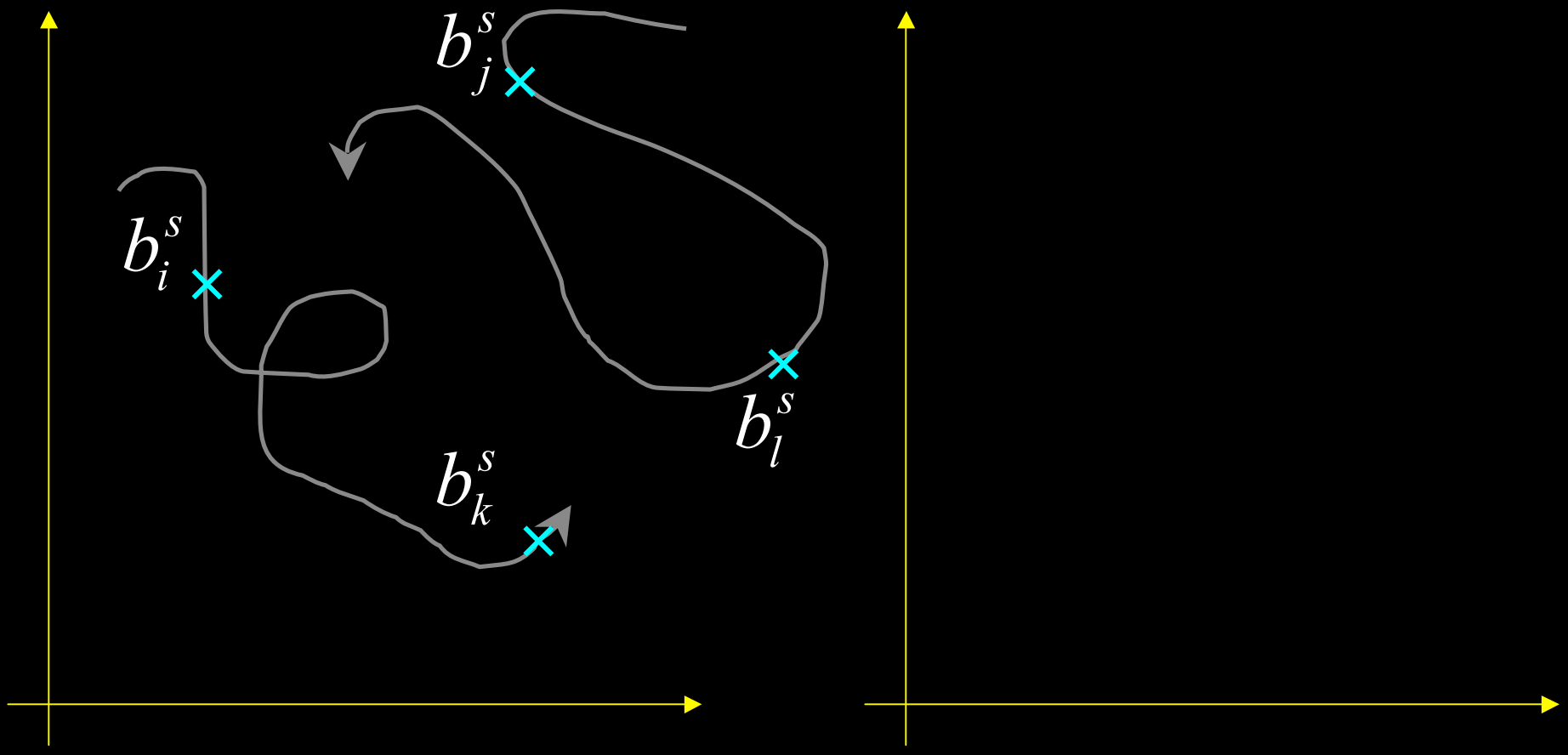
Choe & Ko

Derive source blendshapes from data

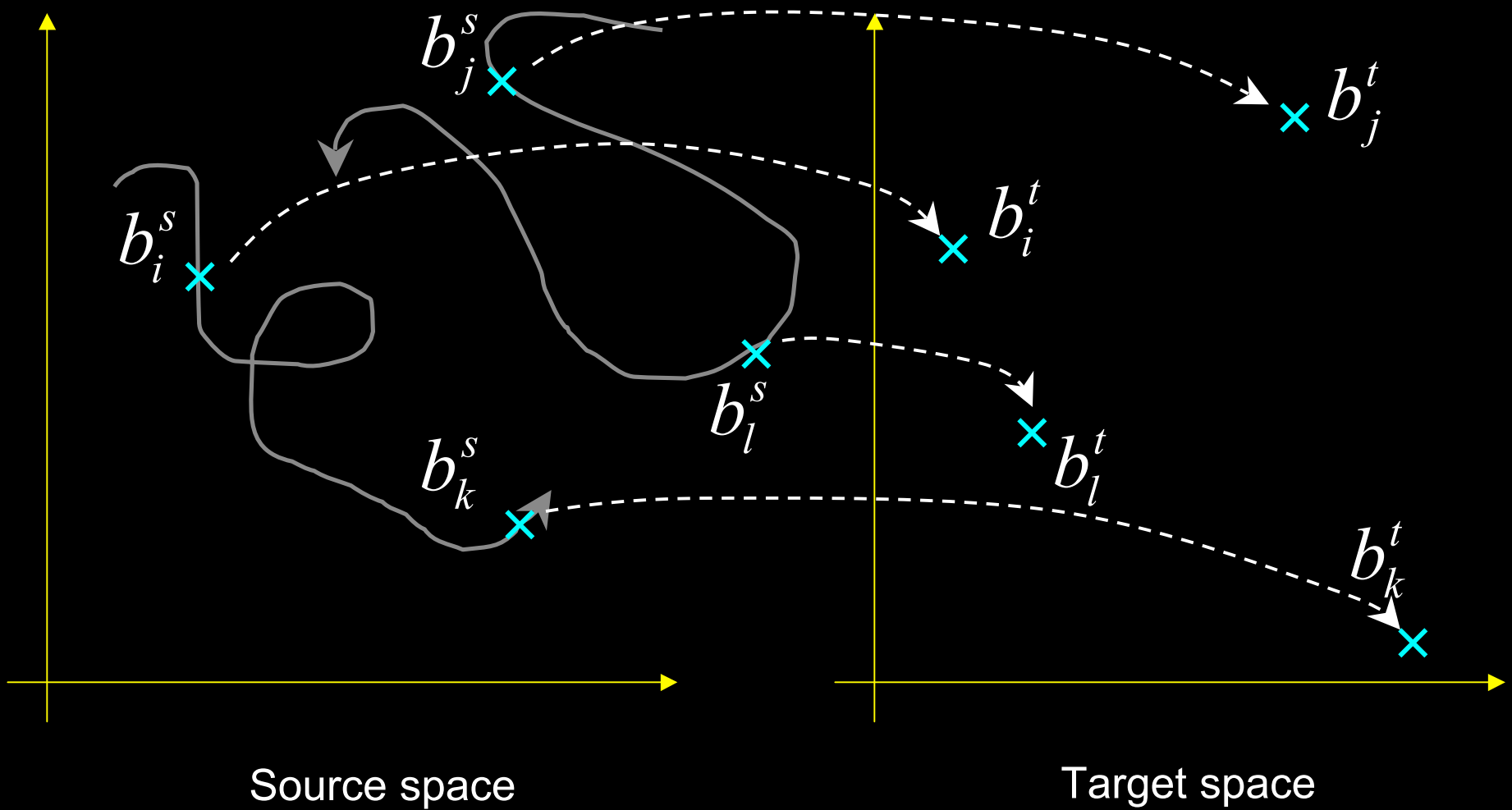
- Principal Component Analysis
- [Chuang and Bregler, 2002]

Recorded source motions

Source basis estimation



Target basis construction



Blendshapes by Principal Component Analysis (PCA)

- Automatic construction of blendshape model (given movement data)
- Advantage: automatic; the most accurate model for a given number of sliders (L2 sense), easy
- Disadvantage: the resulting model is not intuitive

Derive source basis from data

- E. Chuang and C. Bregler, **Performance Driven Facial Animation using Blendshape Interpolation**, CS-TR-2002-02, Department of Computer Science, Stanford University

Chuang and Bregler,

Derive source basis from data

- Parallel Model Construction approach:
  - Source model automatically derived,
  - Target manually sculpted
- Using PCA would be unpleasant

Chuang and Bregler,

Derive source basis from data

- Using PCA would be unpleasant
- Solution: use a subset of the motion capture frames as the blendshape model.
  - Subsets of the original motion capture start to “span the space” of that motion capture.
- Two new problems:
  1) Which motion capture frames to use?
  2) Source blendshape basis is not exact.

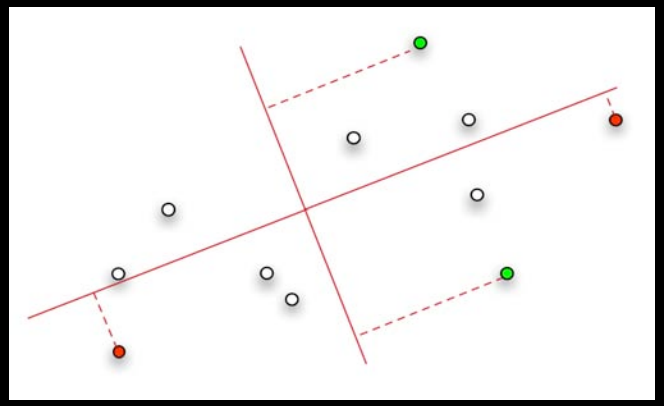
Chuang and Bregler,

Derive source blendshapes from data

Which motion capture frames to use?

Heuristic: for each of the leading PCA vectors,

Pick the mocap frame that have the largest (min,max) projections on that eigenvector.



Chuang and Bregler,

Derive source blendshapes from data

Two new problems:

1) Which motion capture frames to use?
2) Source blendshape basis is only approximate

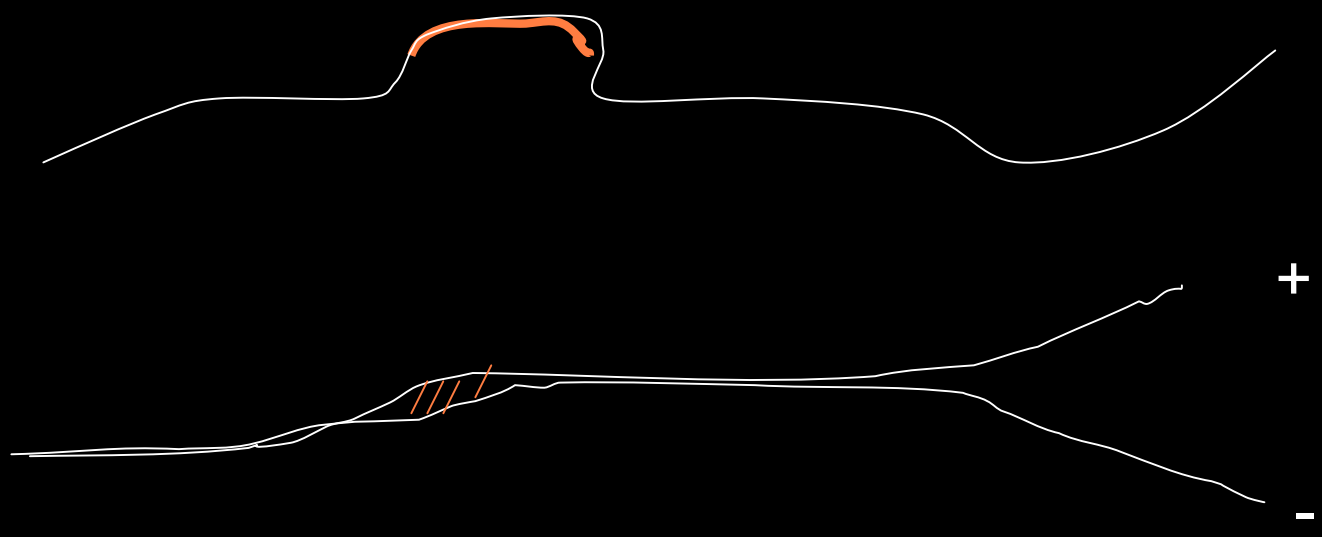
Observation:

- Directly reusing weights works poorly when the source model is not exact
  - Errors in representing the source can result in large cancelling basis combinations (nearly cancelling positive, negative weights)
  - Transferring these cancelling weights to target results in poor shapes

Chuang and Bregler,

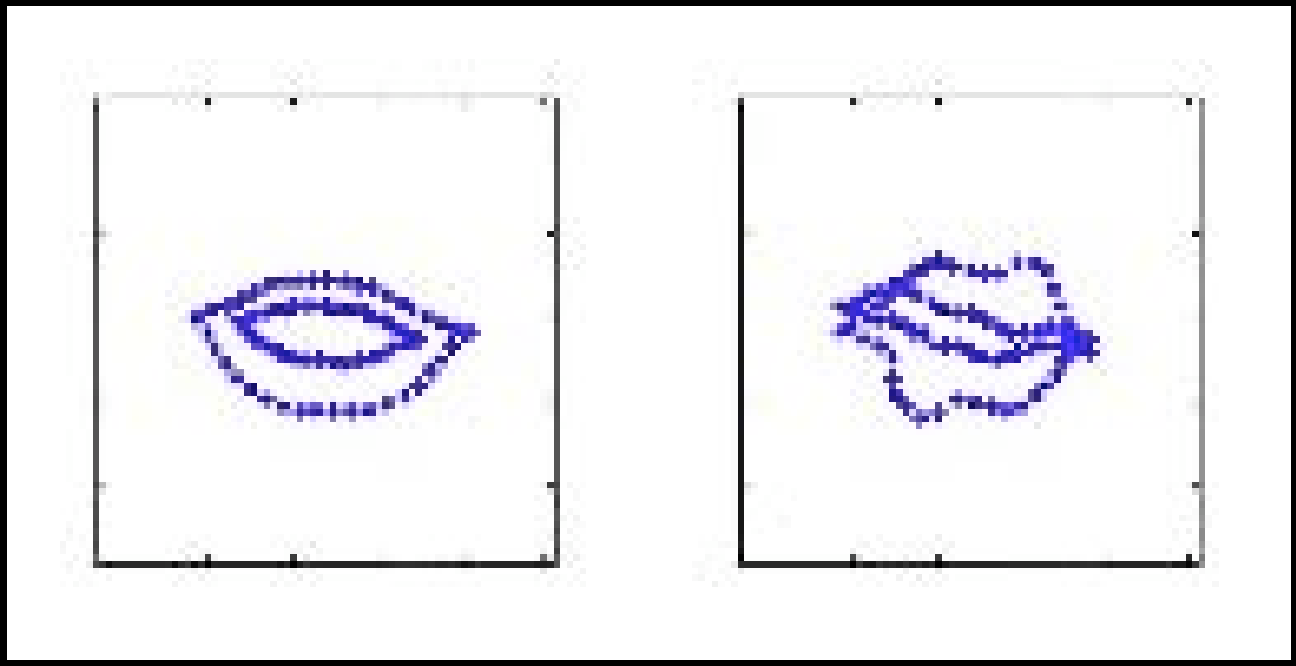
Derive source blendshapes from data

Errors in representing the source can result in large cancelling basis combinations (nearly cancelling positive, negative weights)



Chuang and Bregler,

Derive source blendshapes from data



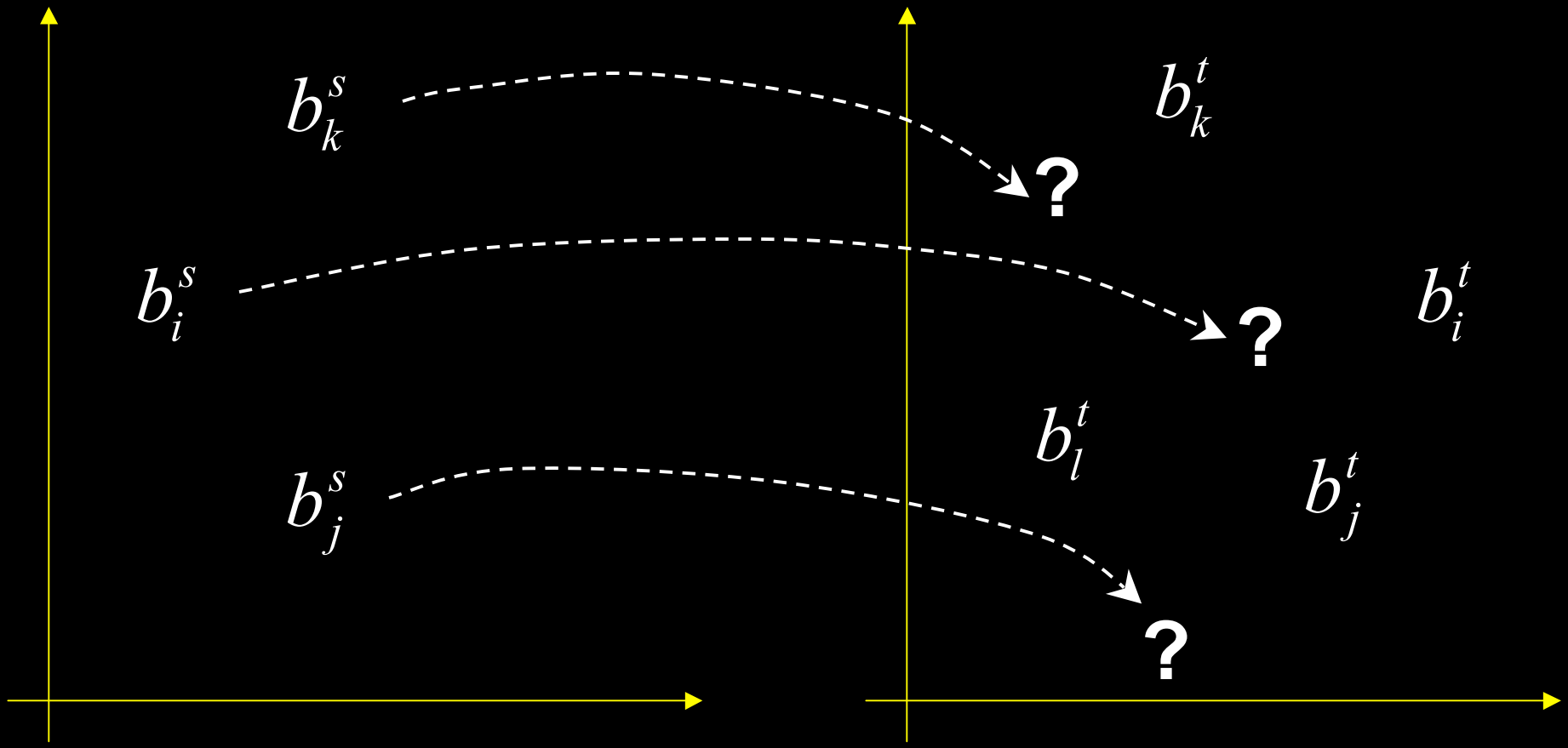
Chuang and Bregler,

Derive source basis from data

- Solution
  - Solve for the representation of the source with non-negative least squares. Prohibiting negative weights prevents the cancelling combinations.
  - Robust cross mapping.

Performance Driven Facial Animation using Blendshape Interpolation

Non-corresponding blendshapes



Is there a "best" blendshape basis?

- There are an infinite number of different blendshape models that have exactly the same range of movement. Proof:

$$\begin{aligned} f &= Bw \\ &= B(KK^{-1})w \\ &= (BK)(K^{-1}w) \\ &= Dx \text{ with } D \equiv BK, x \equiv K^{-1}w \end{aligned}$$

- And it's easy to interconvert between different blendshapes

  analogy: what is the best view of a 3D model? Why restrict yourself to only one view??

Global blendshape mapping

Motivating scenarios:

1) Use PCA for source!
2) Source or target model is pre-existing (e.g. from a library)

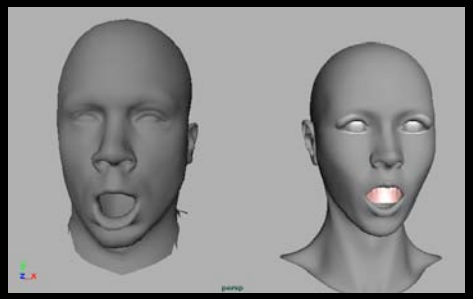
Global blendshape mapping

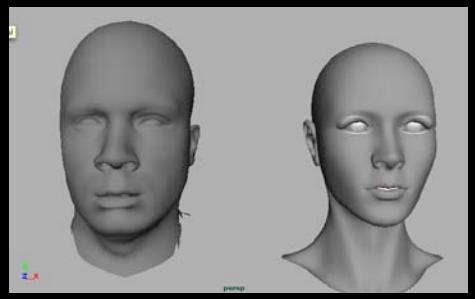
$\mathbf{s} = \mathbf{B}\mathbf{w}$ weights $\mathbf{w}$ pose the source

$\mathbf{t} = \mathbf{C}\mathbf{v}$ weights $\mathbf{v}$ pose the target

$\mathbf{B},\mathbf{C} \in \Re^{3n \times m}$ n vertices, m blendshape targets

Manually create $p \geq m$ corresponding poses of each model, with weights $\mathbf{v}_k$, $\mathbf{w}_k$





Global blendshape mapping

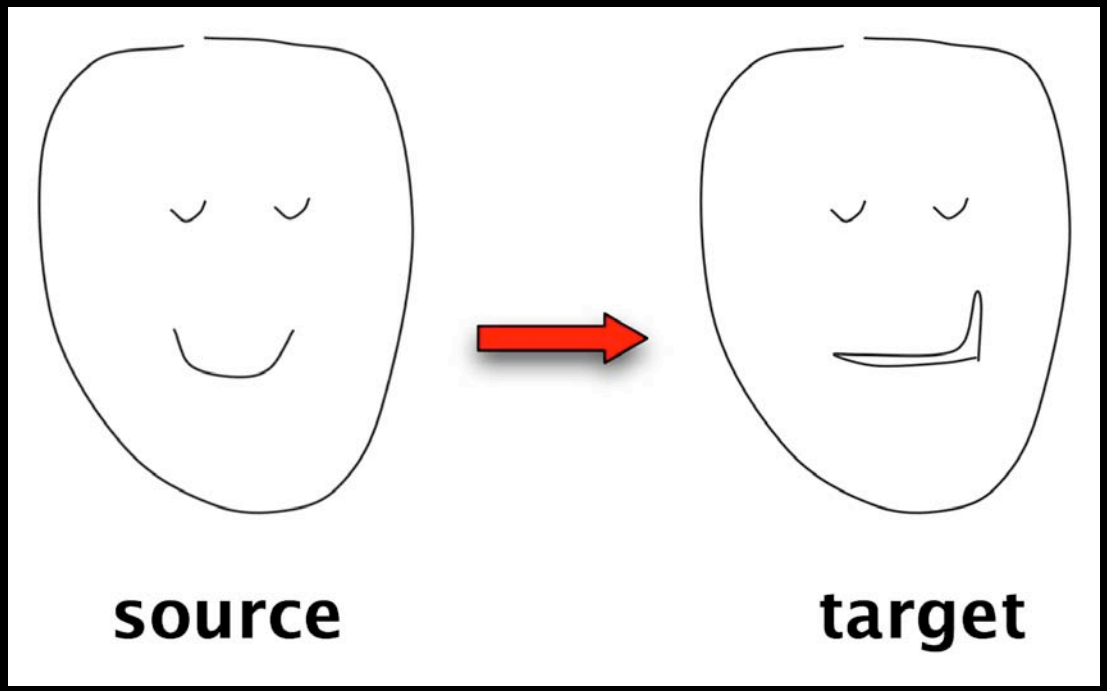
- Gather pose weight vectors $\mathbf{v}_k$, $\mathbf{w}_k$ in columns of $\mathbf{V}$,$\mathbf{W}$
- Solve for the "cloning matrix" $\mathbf{E}$:

$$V = EW \Rightarrow E = VW^T(WW^T)^{-1}$$

this matrix converts weights for one model to produce the equivalent expression in the other model.

Global blendshape mapping

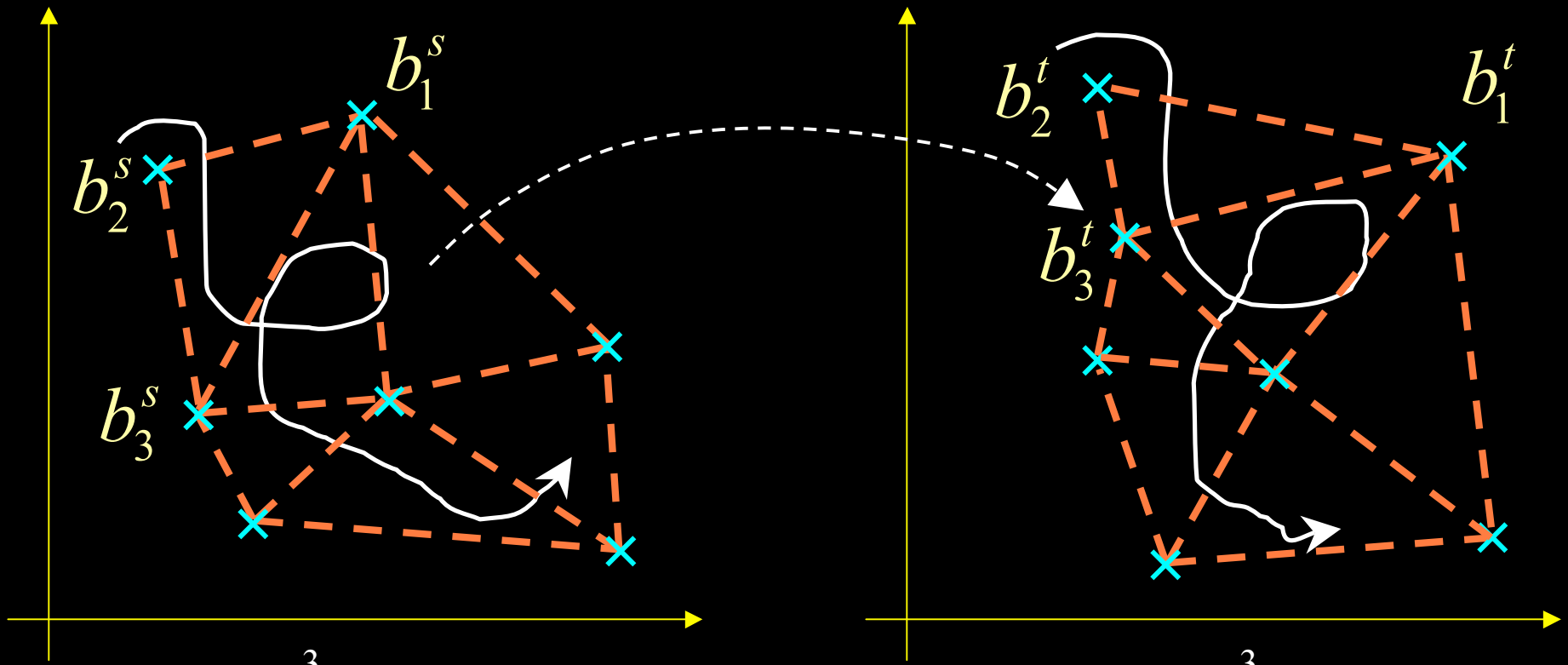
- *Intentional* source-target mismatch: style transfer (person on the right has asymmetric smile)



Non-linear Mapping

- Piecewise linear
- Manifold learning
- Single-correspondence

Piecewise linear mapping



$$f_k^s = \sum_{i=1}^{3} w_{k,i} b_{k,i}^s \longrightarrow f_k^t = \sum_{i=1}^{3} w_{k,i} b_{k,i}^t$$

(Local) Change of coordinate system

Piecewise linear mapping

- I. Buck, A. Finkelstein, C. Jacobs, A. Klein, D. H. Salesin, J. Seims, R. Szeliski, and K. Toyama, **Performance-driven hand-drawn animation**, *NPAR 2000*

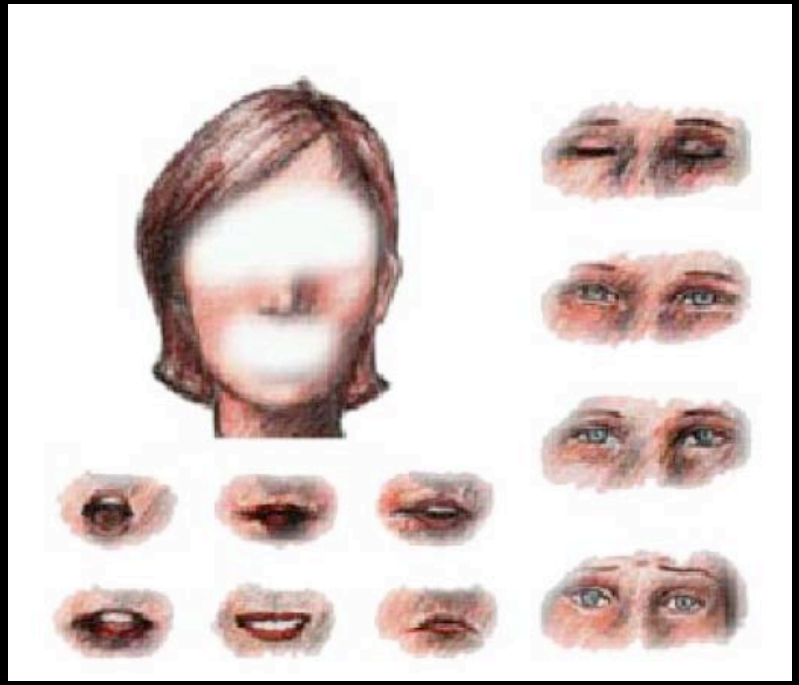
Buck et. al.

Piecewise linear mapping

- Project motion onto a 2D space (PCA)
- Construct Delaunay triangulation based on source blendshapes
- Within a triangle use barycentric coordinates as blending weights

Buck et. al.

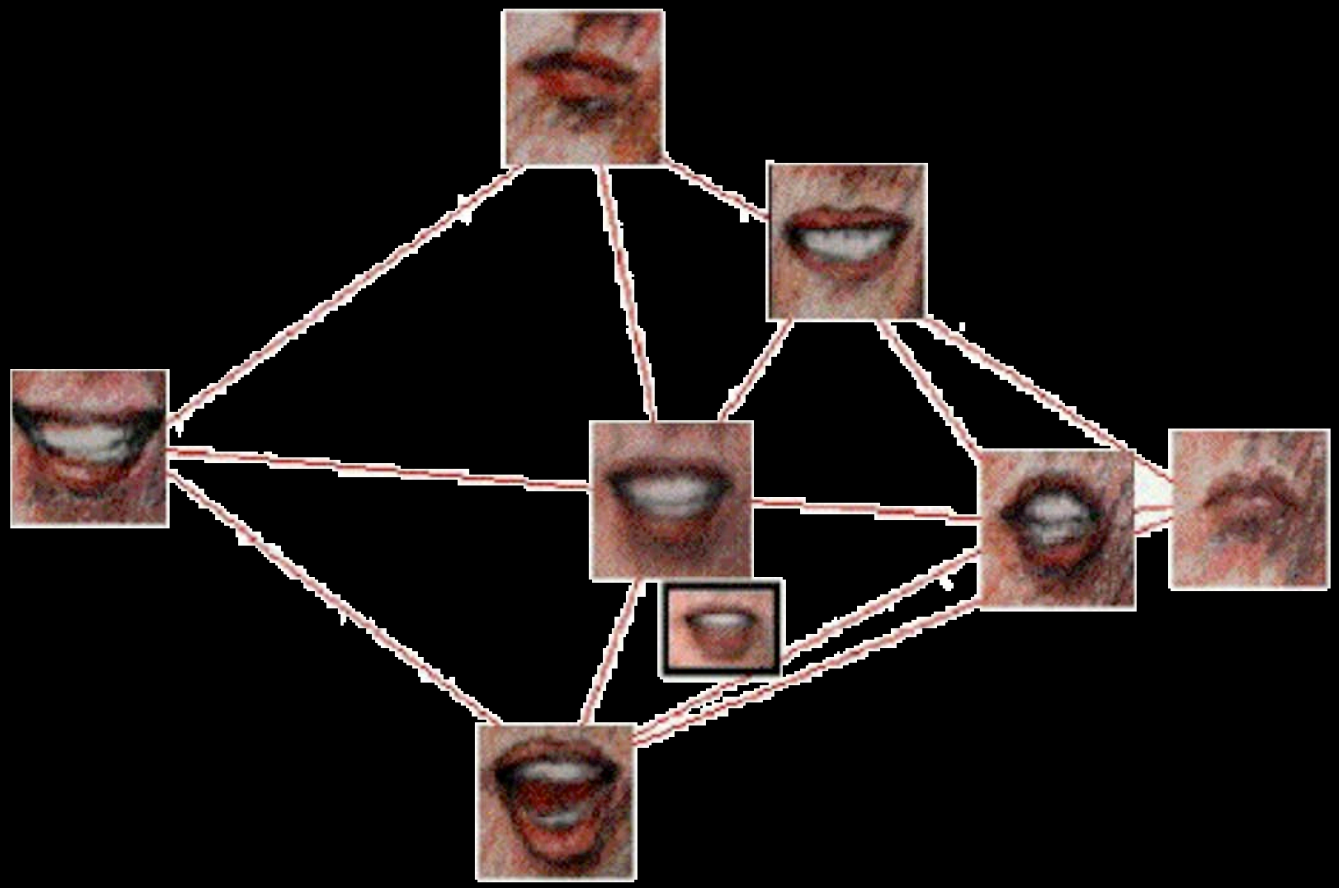
Piecewise linear mapping



- Split face into 3 regions (eyes, mouth, rest of the face)
- If frame outside of triangulated area, project onto convex hull

Buck et. al.

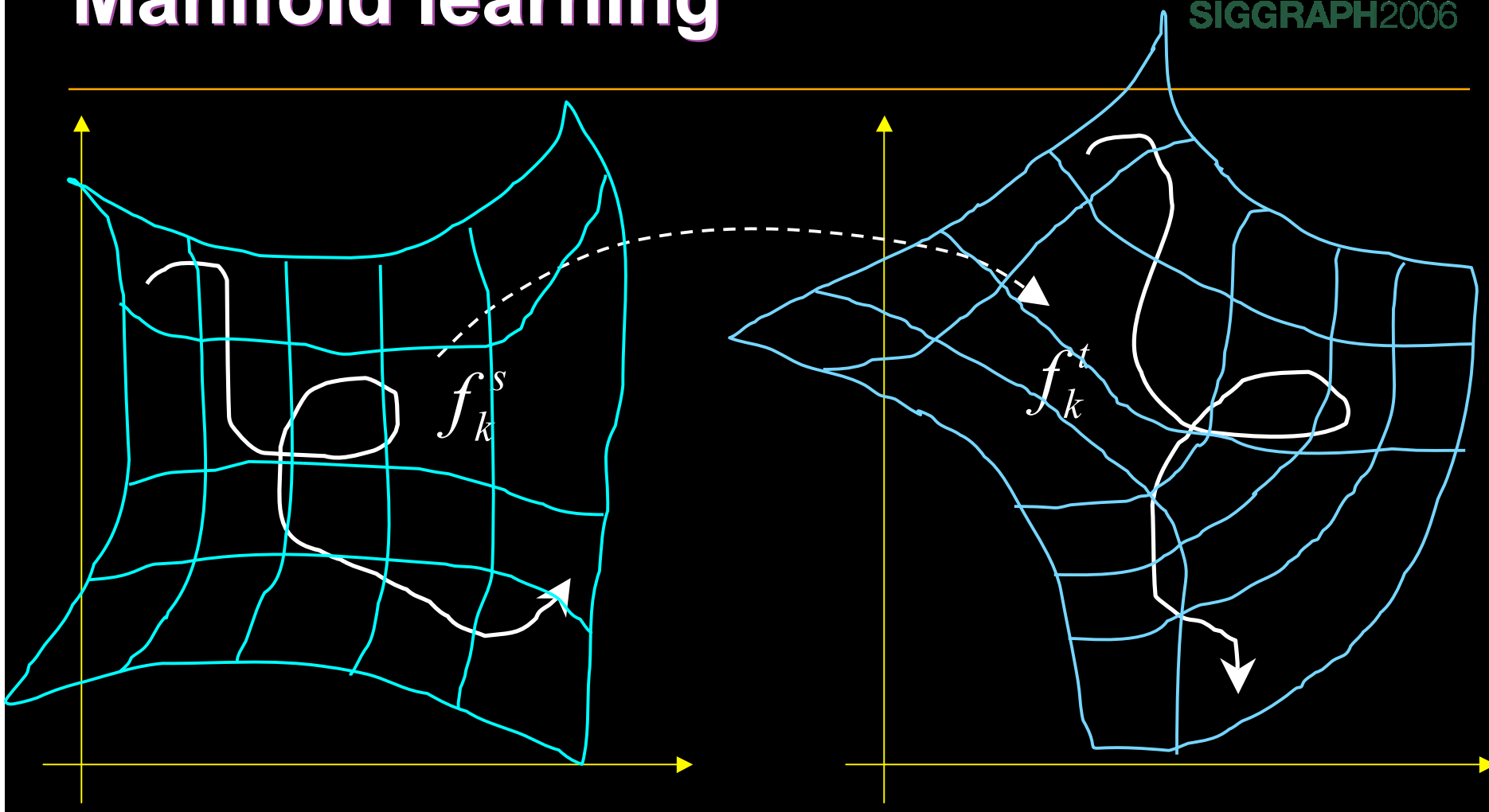
Triangulation



Video

- hand_drawn.avi

Manifold learning

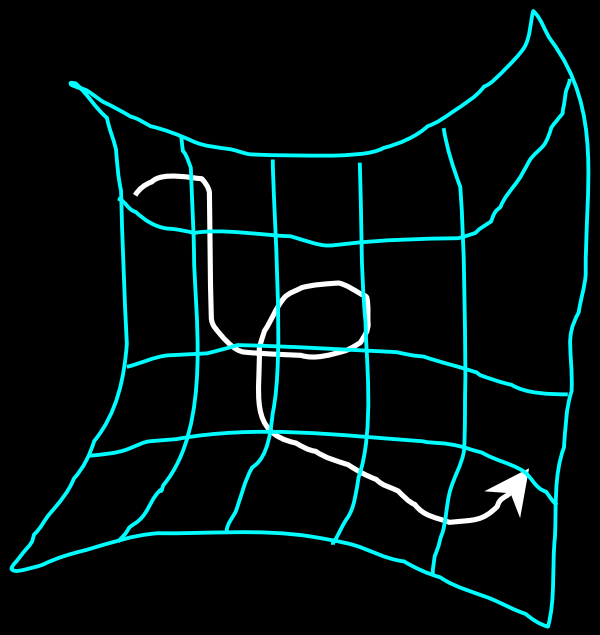


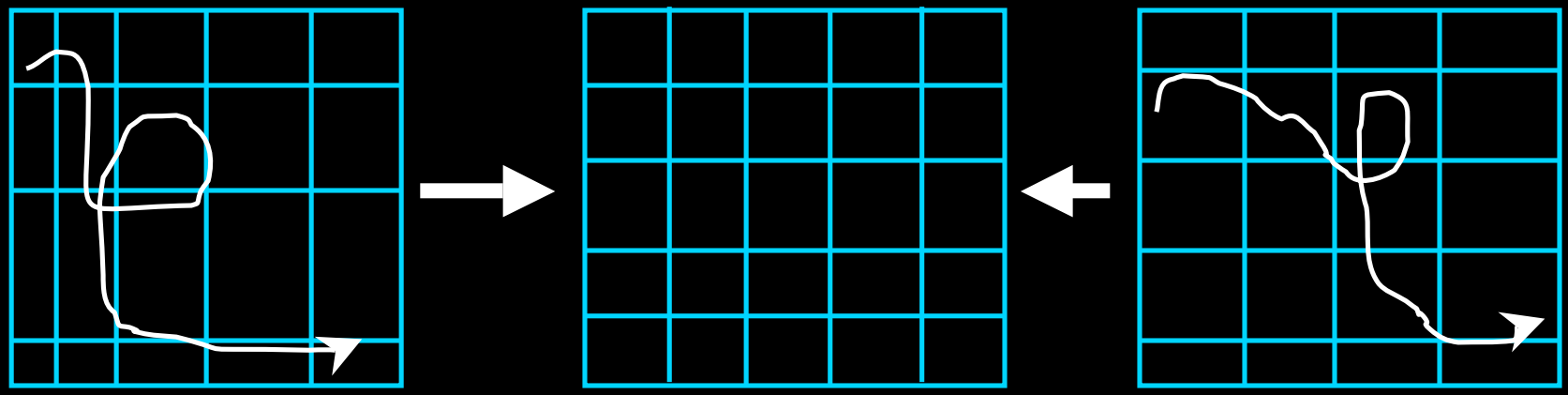
Manifold learning

- Y. Wang, X. Huang, C.-S. Lee, S. Zhang, D. Samaras, D. Metaxas, A. Elgammal, and P. Huang, **High Resolution Acquisition, Learning, and Transfer of Dynamic 3-D Facial Expressions**, *Eurographics* 2004.

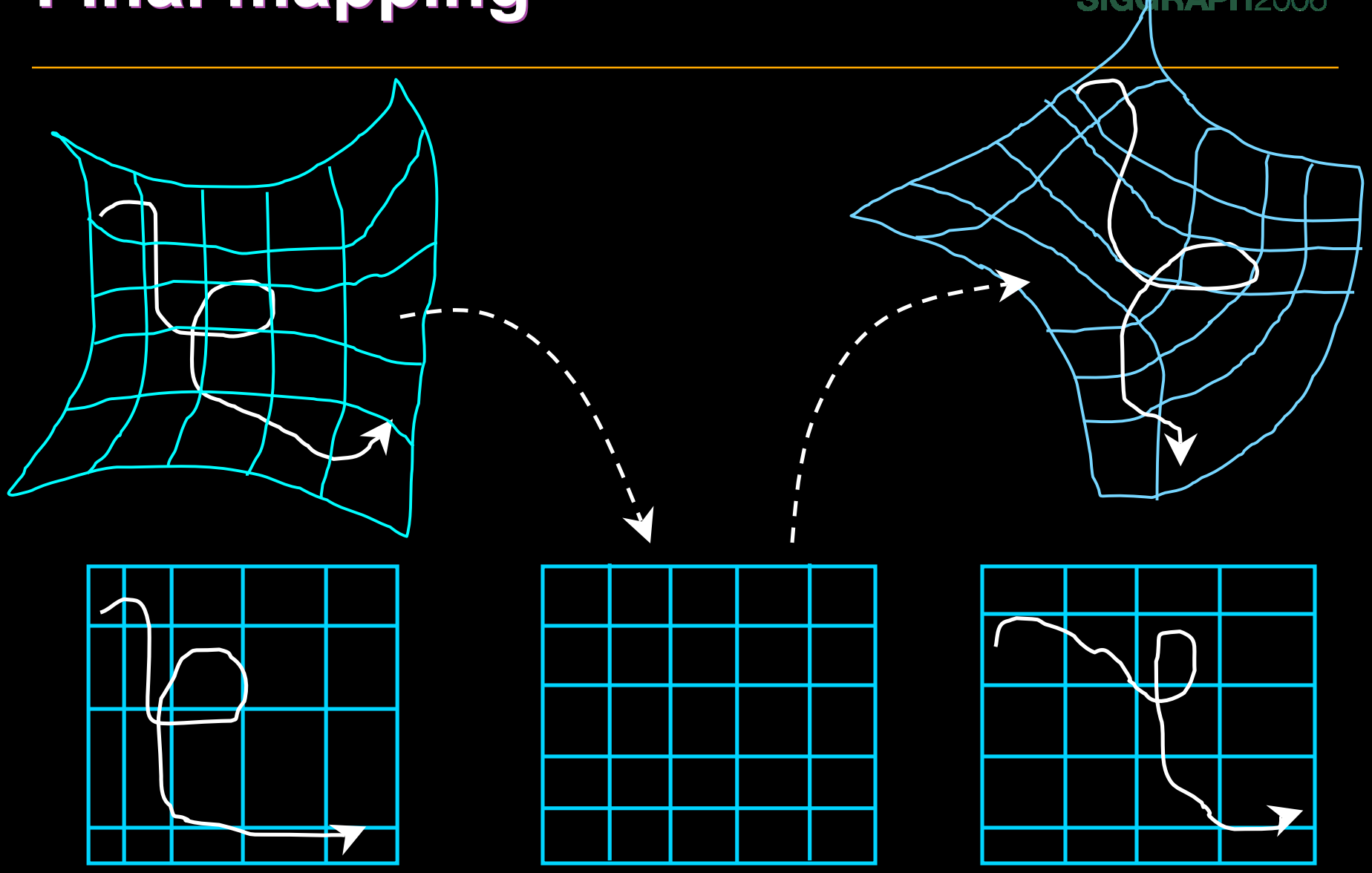
Source and target embeddings

Unified embedding

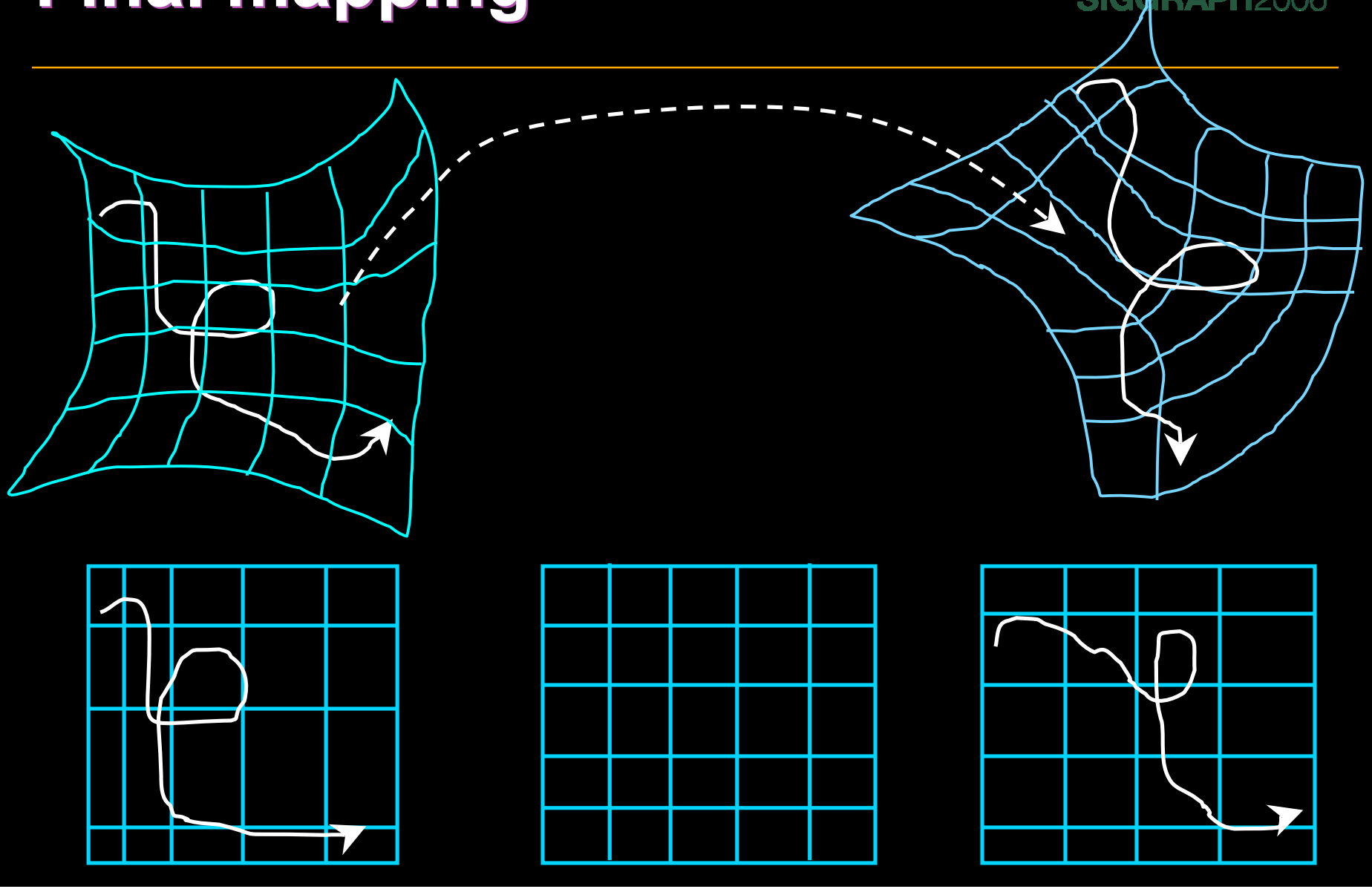




Final mapping

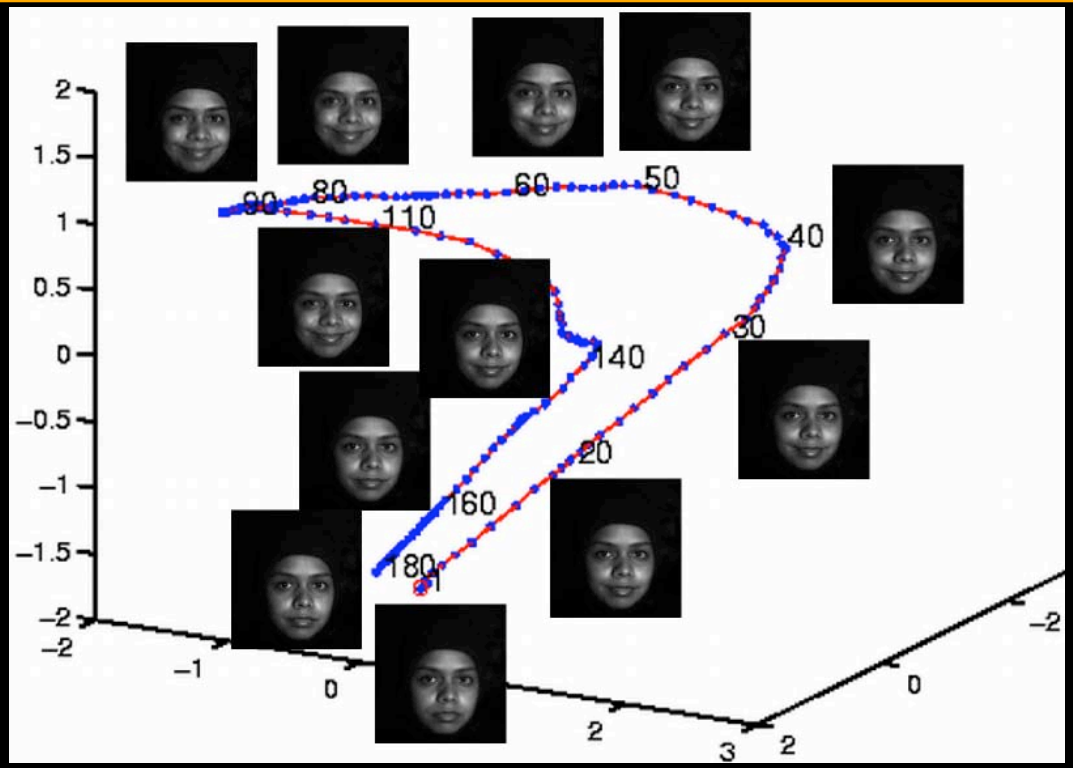


Final mapping



Wang et. al.

Manifold learning

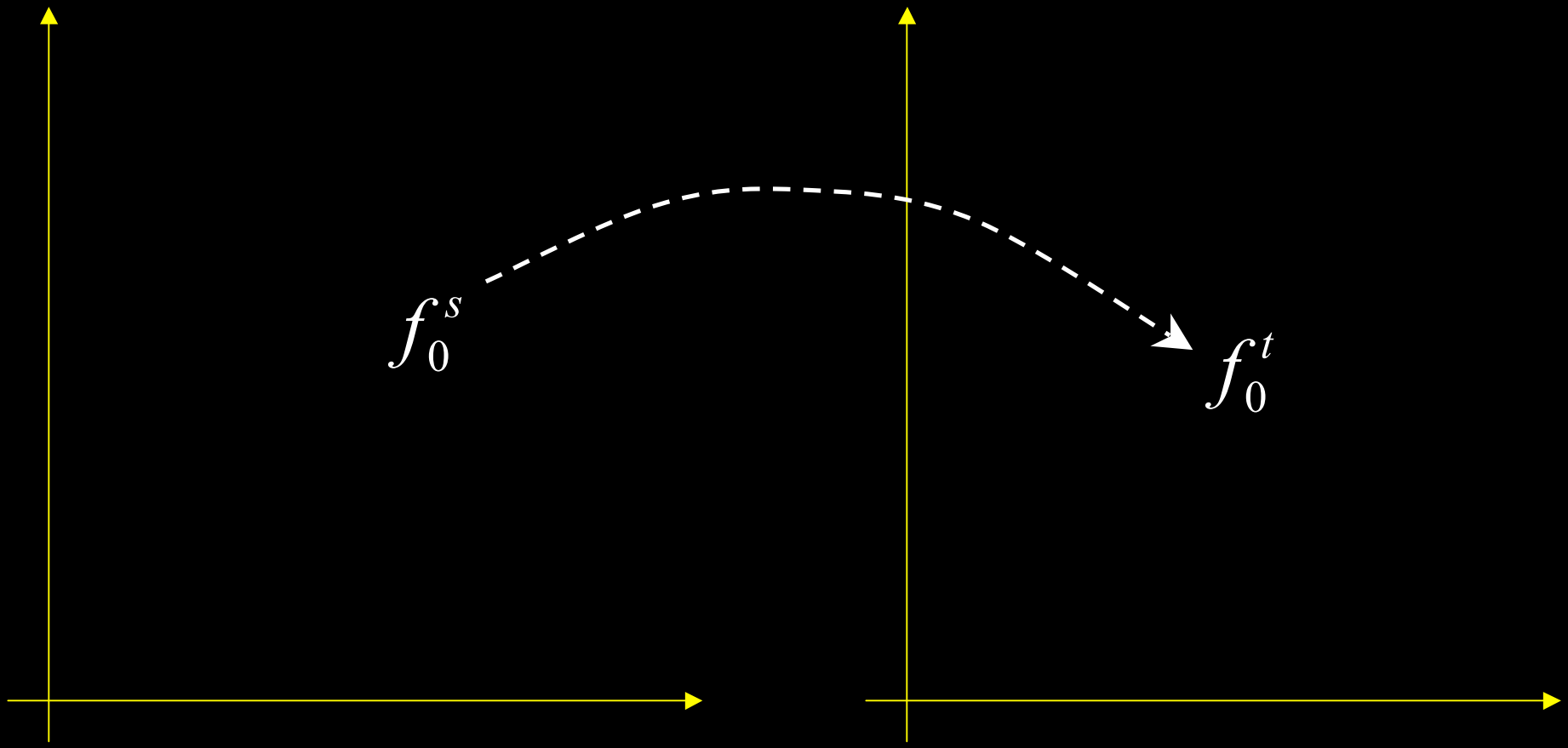


Manifold (curve) of smile motion obtained by Local Linear Embedding (from Wang et. al.)

Video

- final-video2-edit.mov

Mapping from a single correspondence



$$f^t = M\left(f^s \mid (f_0^s, f_0^t)\right)$$

Mapping from a single correspondence

- J.-Y. Noh and U. Neumann, **Expression Cloning**, *SIGGRAPH* 2001.

Noh et. al.

Two issues

- Find dense geometric correspondences between the two face models

- Map motion using local geometric deformations from source to target face

Noh et. al.

Estimating geometric correspondences

- Sparse correspondences through feature detection

- Dense correspondences by interpolating matching features (RBF)

Noh et. al.

Local geometric motion transformation

Source

Target

Noh et. al.

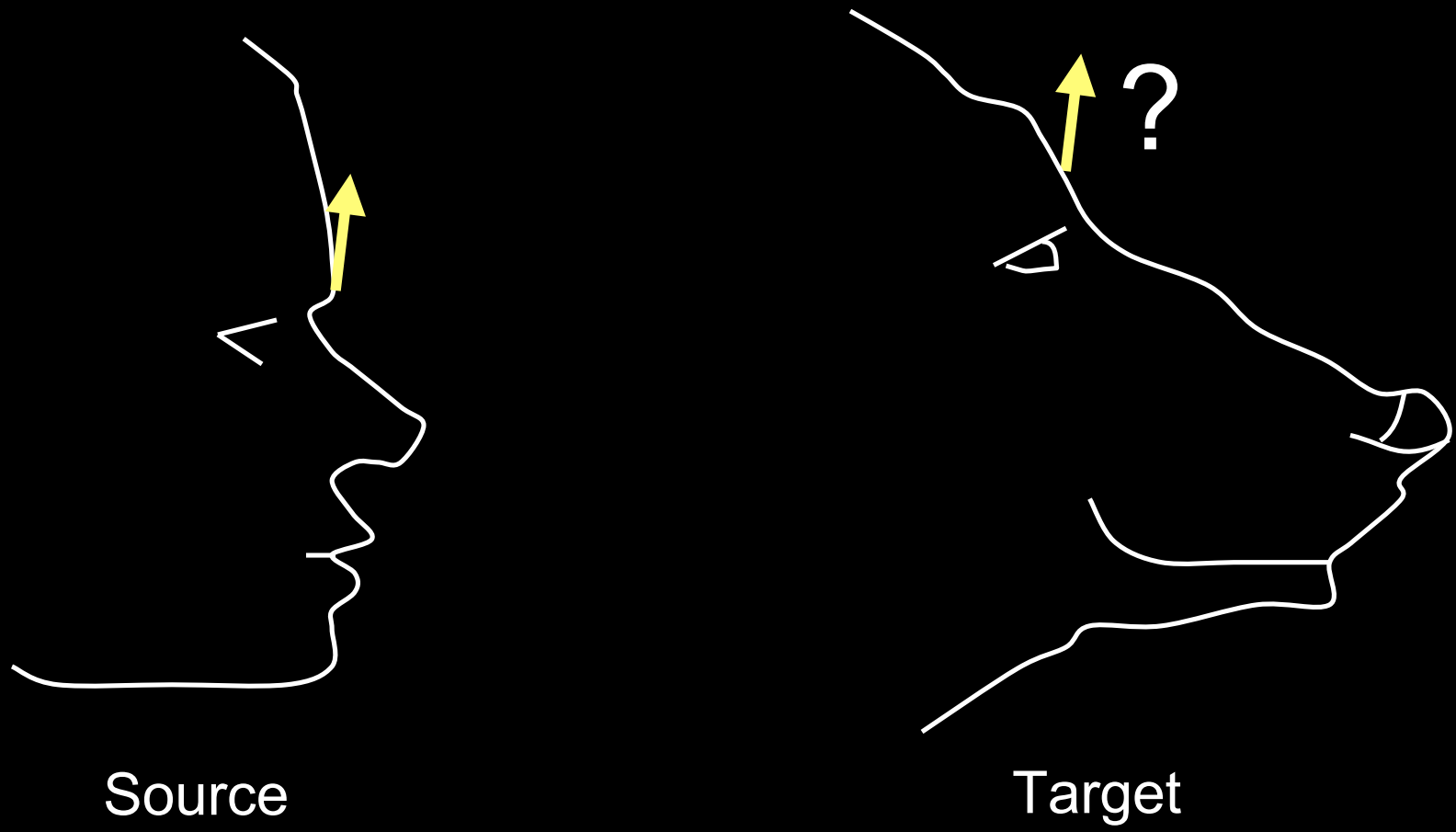
Animation as displacement from the neutral/rest face

Source

Target

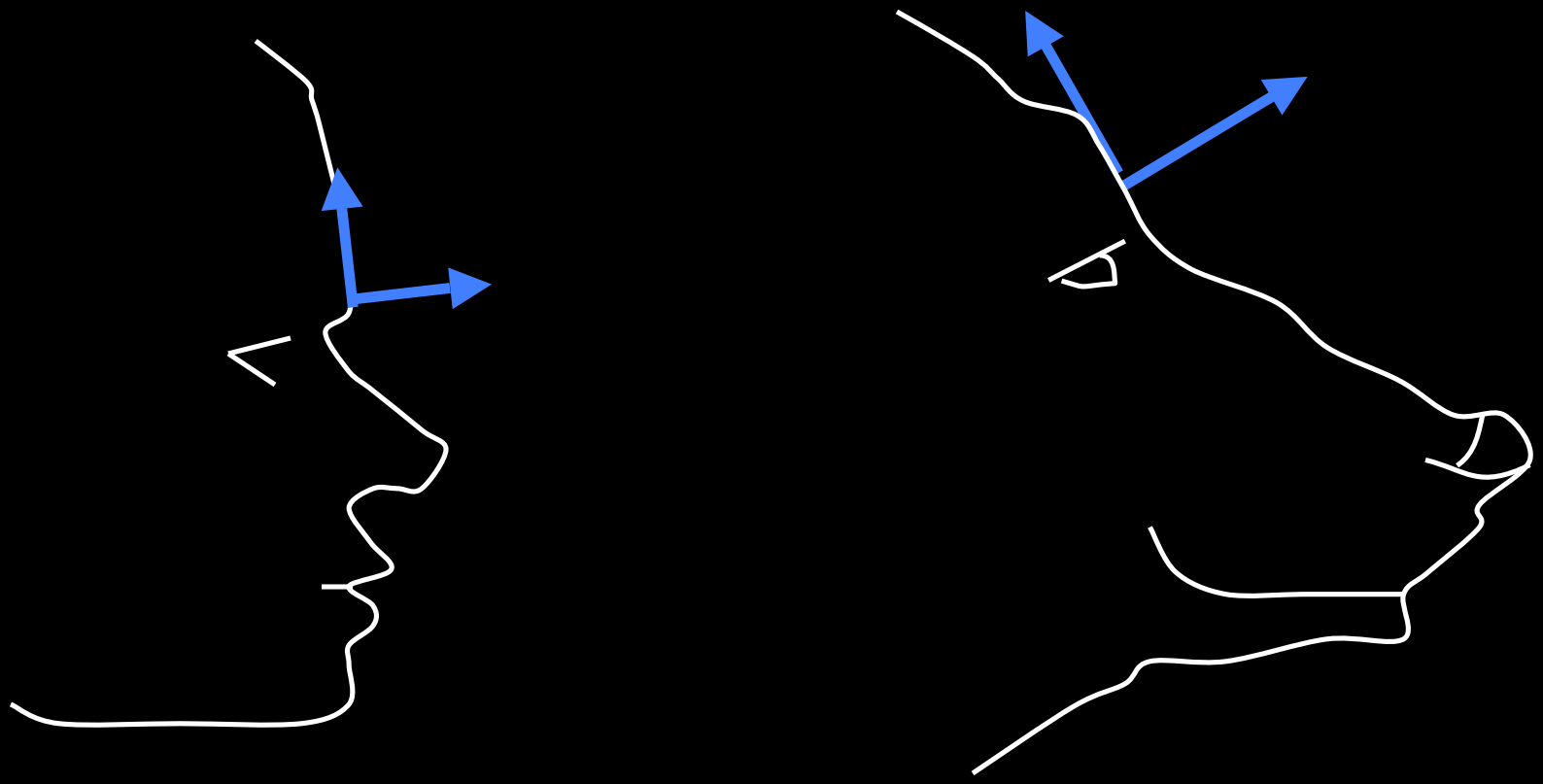
Noh et. al.

Local geometric motion transformation



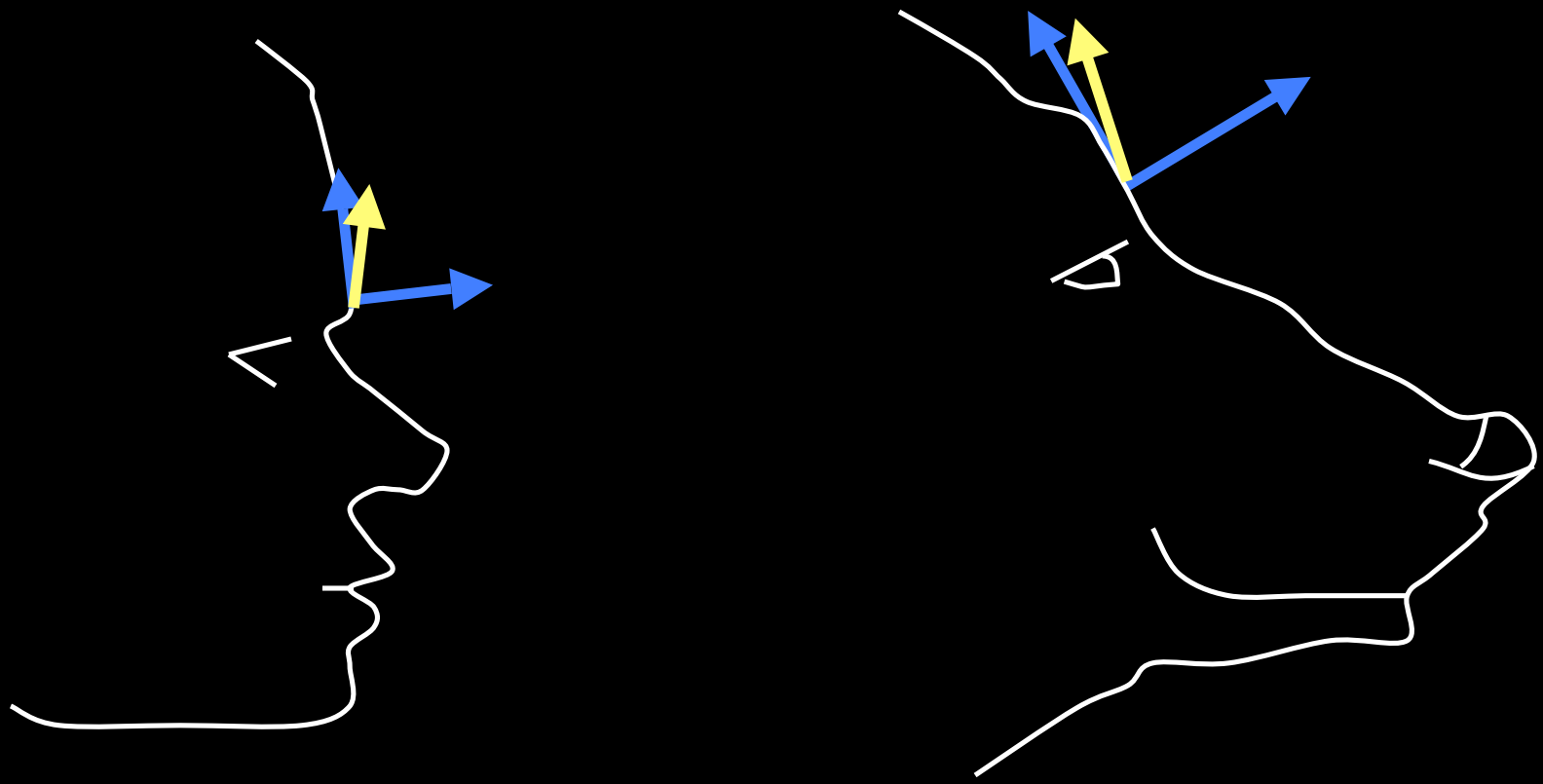
Noh et. al.

Local change of coordinates system



Noh et. al.

Local change of coordinates system



Noh et. al.

Local coordinates system

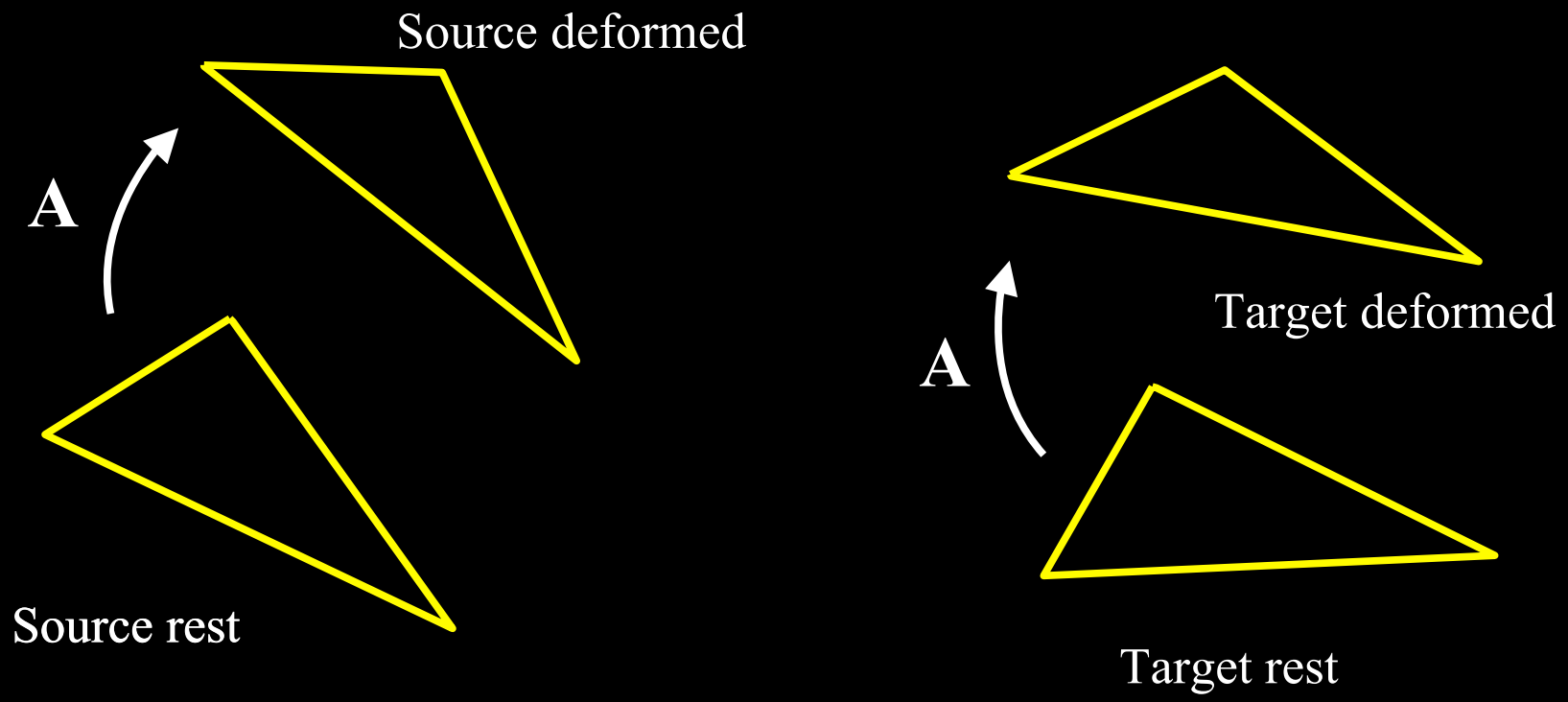
- Defined by
  - Tangent plane and surface normal
  - Scale factor: ratio of bounding boxes containing all triangles sharing vertex

Video

- monkeyExp.mov

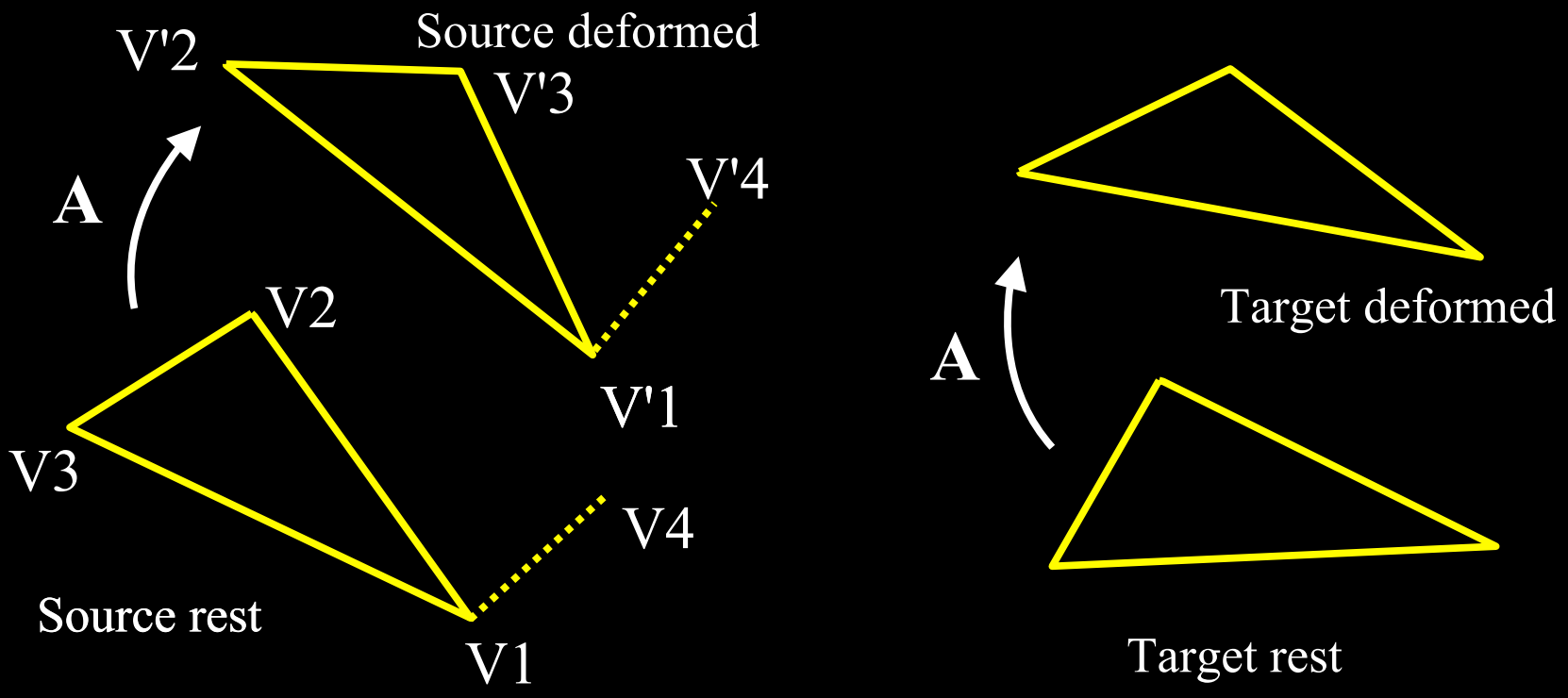
Summer and Popovic

Global geometric motion transformation



Summer and Popovic

Global geometric motion transformation



$$(V1, V2, V3, V4) \xrightarrow{A} (V'1, V'2, V'3, V'4)$$

with $V4 = V1 + (V2 - V1) \times (v3 - v1) / \sqrt{(\|(V2 - V1) \times (v3 - v1)\|)}$

Summer and Popovic

Global geometric motion transformation

- Constraint vertices to move consistently with respect to each triangle it belongs to

- Solve system of linear equations

References

- B. Choe and H. Ko, **Analysis and Synthesis of Facial Expressions with Hand-Generated Muscle Actuation Basis**, *Computer Animation* 2001.
- I. Buck, A. Finkelstein, C. Jacobs, A. Klein, D. H. Salesin, J. Seims, R. Szeliski, and K. Toyama, **Performance-driven hand-drawn animation**, *NPAR 2000.*
- E. Chuang and C. Bregler, **Performance Driven Facial Animation using Blendshape Interpolation**, CS-TR-2002-02, Department of Computer Science, Stanford University
- J.-Y. Noh and U. Neumann, **Expression Cloning**, *SIGGRAPH* 2001.
- R. W. Summer and J. Popovic, **Deformation Transfer for Triangle Meshes**,*SIGGRAPH* 2004
- Y. Wang, X. Huang, C.-S. Lee, S. Zhang, D. Samaras, D. Metaxas, A. Elgammal, and P. Huang, **High Resolution Acquisition, Learning, and Transfer of Dynamic 3-D Facial Expressions**, *Eurographics* 2004.

Future work

- Artists-driven retargeting
- Physically-based retargeting

SIGGRAPH Course 30: Performance-Driven Facial Animation

For Latest Version of Bregler's Slides and Notes please go to:

http://cs.nyu.edu/~bregler/sig-course-06-face/

SIGGRAPH Course 30: Performance-Driven Facial Animation

Section:

Markerless Face Capture and Automatic Model Construction

Part 1: Chris Bregler, NYU

Markerless Face Capture

Markerless Face Capture - Overview -

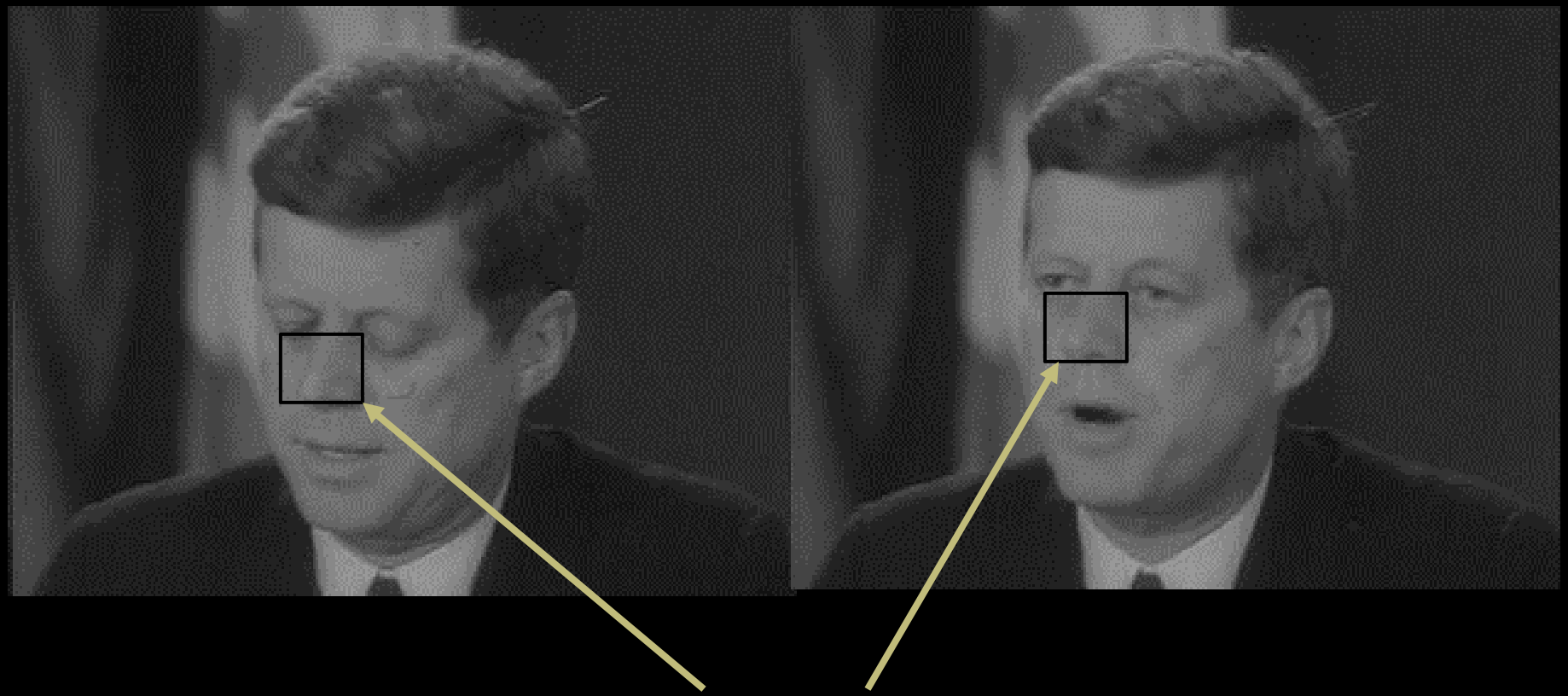
- Single / Multi Camera Input
- 2D / 3D Output
- Real-time / Off-line
- Interactive-Refinement / Face Dependent / Independent
- Make-up / Natural
- Flow / Contour / Texture / Local / Global Features
- Hand Crafted / Data Driven
- Linear / Nonlinear Models / Tracking

Markerless Face Capture – History –

- Single Camera Input
- 2D Output
- Off-line
- Interactive-Refinement
- Make-up
- Contour / Local Features
- Hand Crafted
- Linear Models / Tracking

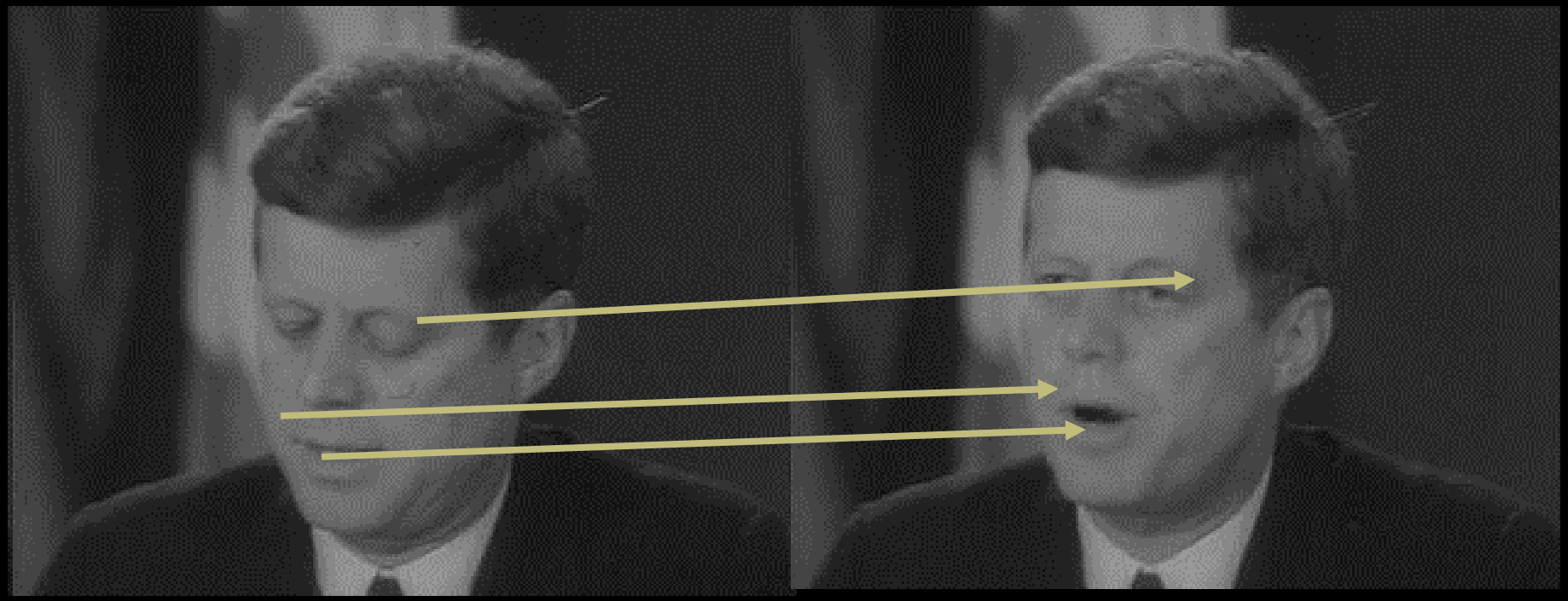
Kass, M., Witkin, A., & Terzopoulos, D. (1987) Snakes: Active contour models.

Tracking = Error Minimization



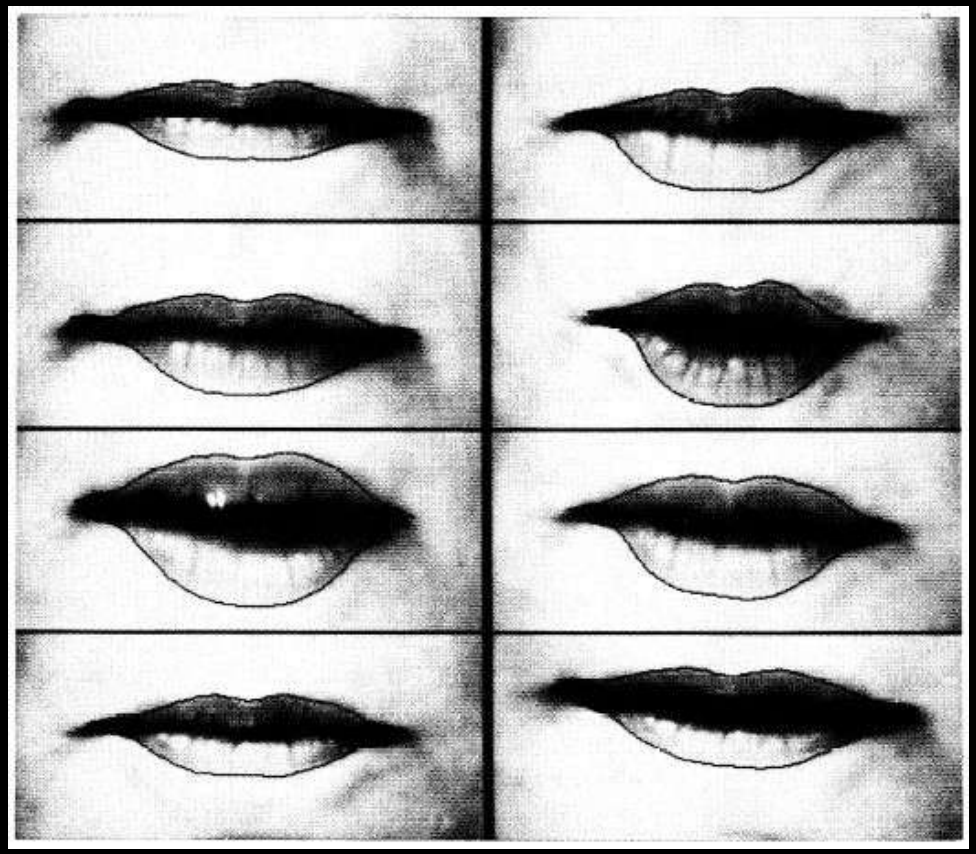
$$Err(u,v) = \sum || I(x,y) - J(x+u, y+v) ||$$

Tracking = Error Minimization



In general: ambiguous using local features

Tracking = Error Minimization



$$E^*_{\text{snake}} = \int_0^1 E_{\text{snake}}(\mathbf{v}(s))\, ds$$
$$= \int_0^1 E_{\text{int}}(\mathbf{v}(s)) + E_{\text{image}}(\mathbf{v}(s))$$
$$+ E_{\text{con}}(\mathbf{v}(s))\, ds$$

Kass, M., Witkin, A., & Terzopoulos, D. (1987) Snakes: Active contour models.

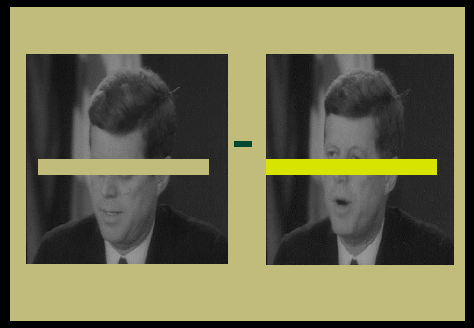
Tracking = Error Minimization

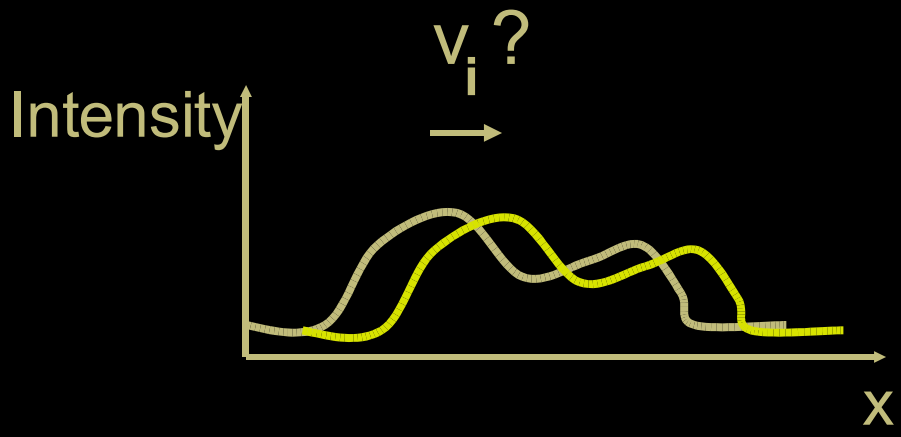
Error = Feature Error + Model Error

Tracking = Error Minimization

Error = **Optical Flow** + Model Error

Optical Flow (Lucas-Kanade)

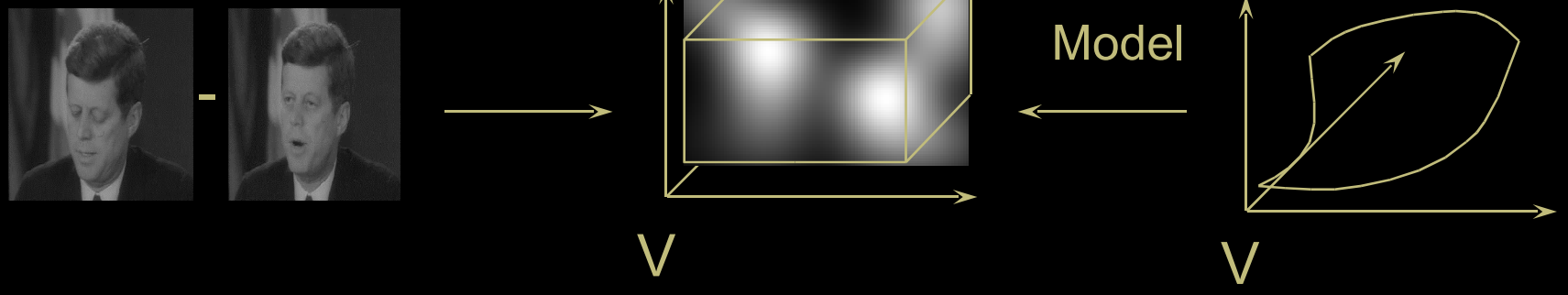




$$E(V) \qquad \left\| I(x_i) - J(x_i + v_i) \right\|^2$$

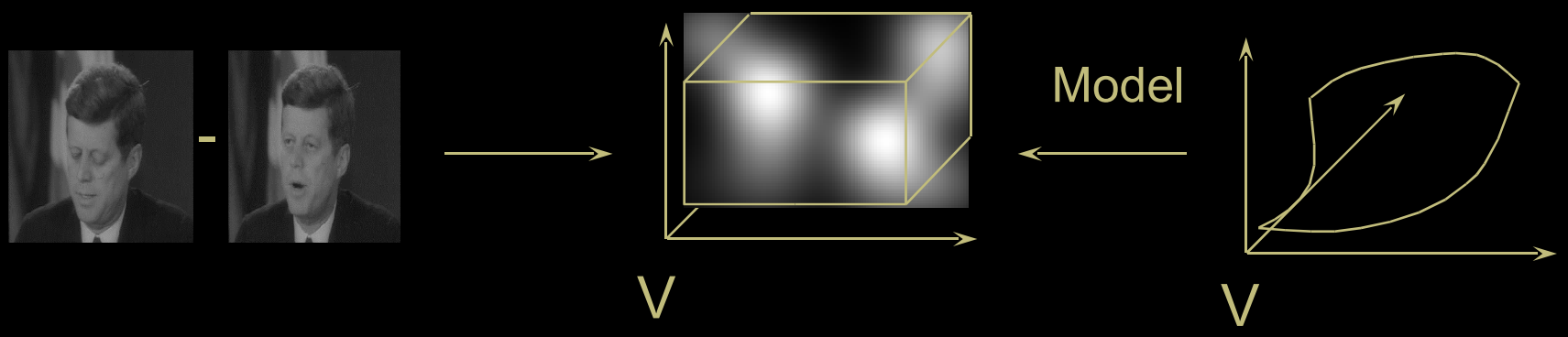
linearize $$\left\| I_t(x_i) - \nabla I(x_i) v_i \right\|^2$$

Optical Flow + Model



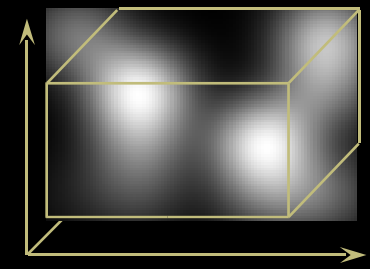
$$\left\| \begin{matrix} I_t(1) - \nabla I(1)\,\vec{v}_1 \\ I_t(2) - \nabla I(2)\,\vec{v}_2 \\ \dots \\ I_t(n) - \nabla I(n)\,\vec{v}_n \end{matrix} \right\|^2 = E(V)$$

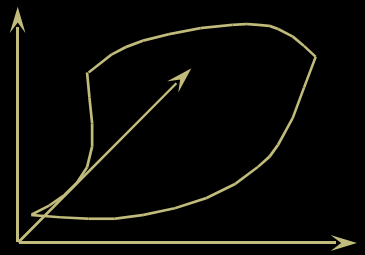
Optical Flow + Model



$$\left\| \begin{matrix} I_t(1) - \nabla I(1)\,\vec{v}_1 \\ I_t(2) - \nabla I(2)\,\vec{v}_2 \\ \dots \\ I_t(n) - \nabla I(n)\,\vec{v}_n \end{matrix} \right\|^2 = E(V) \qquad V = M(\theta)$$

Optical Flow + linearized Model





Model

V

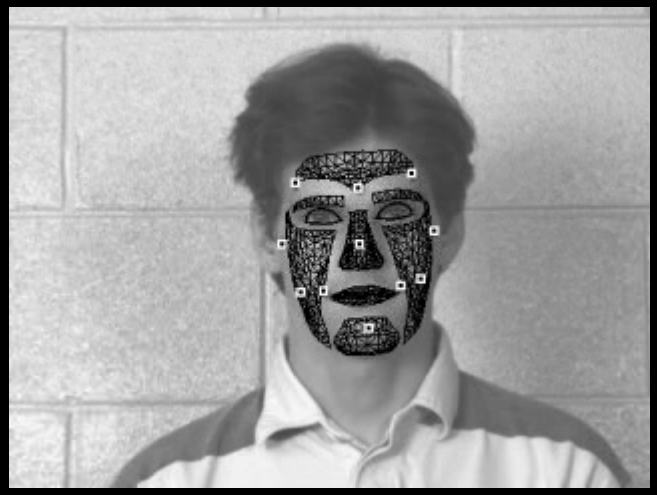
V

$$\left\| Z + H V \right\|^2$$

$$V = M \theta$$

$$\left\| Z + C \theta \right\|^2$$

Optical Flow + Hand-Crafted Model

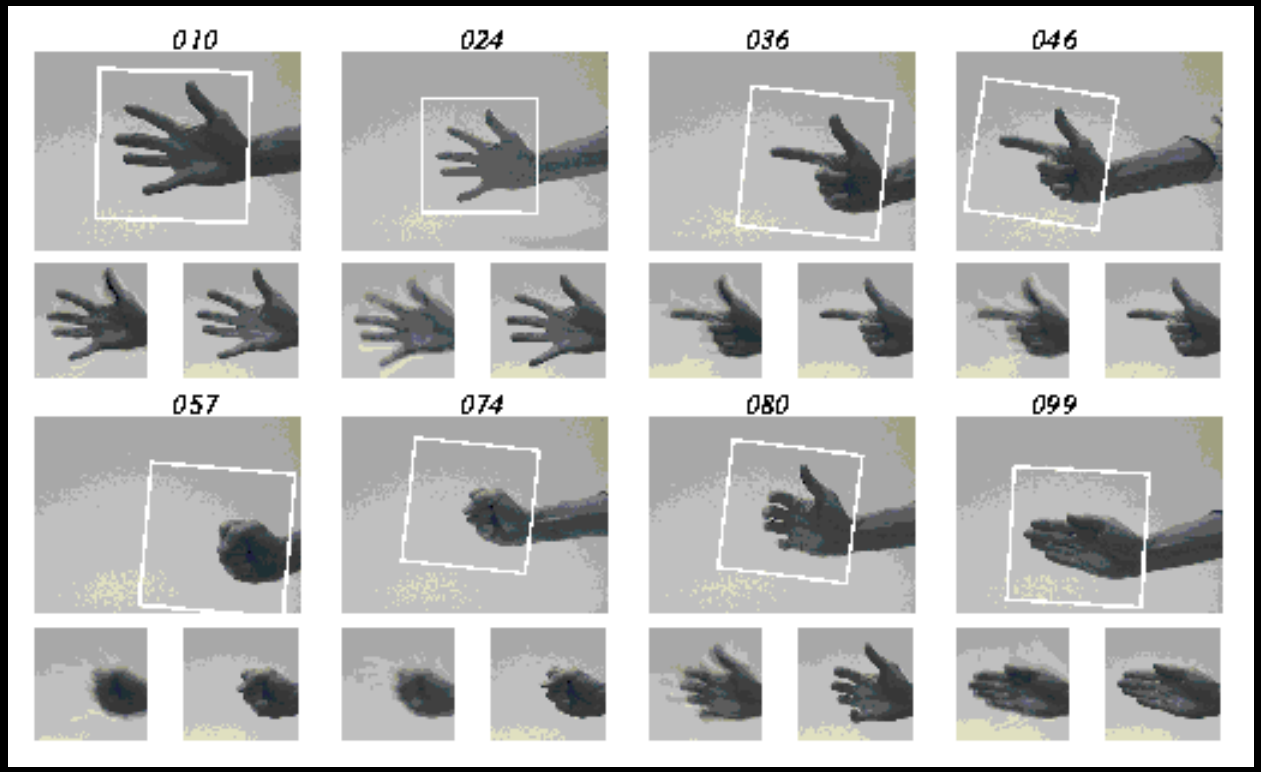


DeCarlo, Metaxas, 1999

Williams et a,l 2002

Optical Flow and PCA

Eigen Tracking
(Black and Jepson)



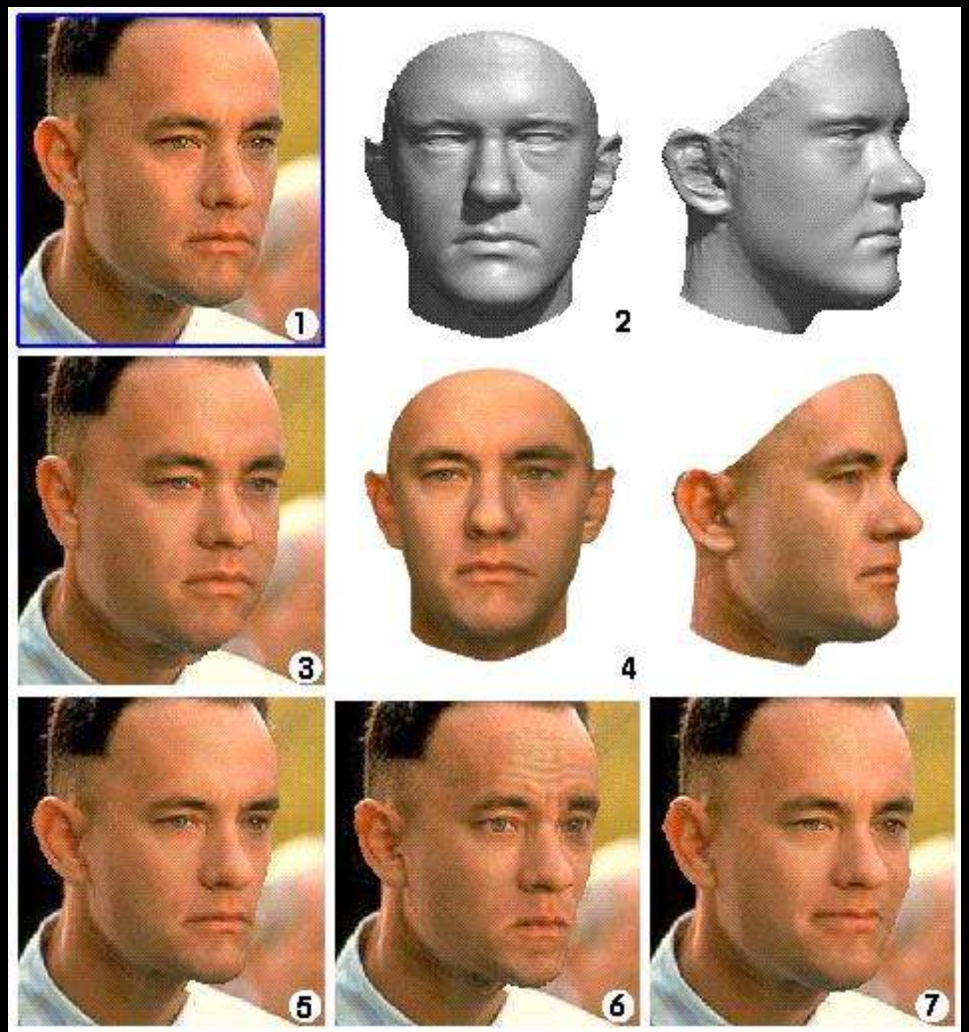
PCA over 2D texture and contours

Active Appearance Models (AAM): (Cootes et al)

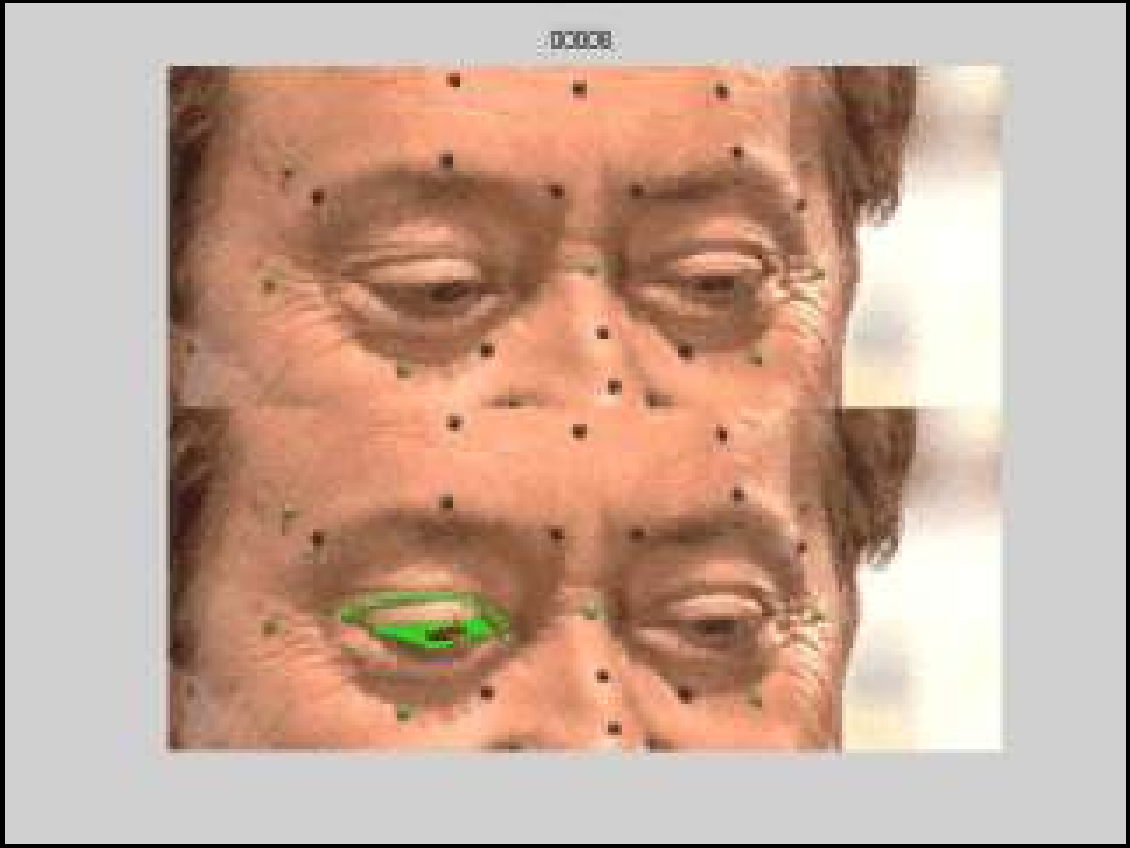
PCA over 2D texture and contours

PCA over texture and 3D shape

3D Morphable Models
(Blanz+Vetter 99)

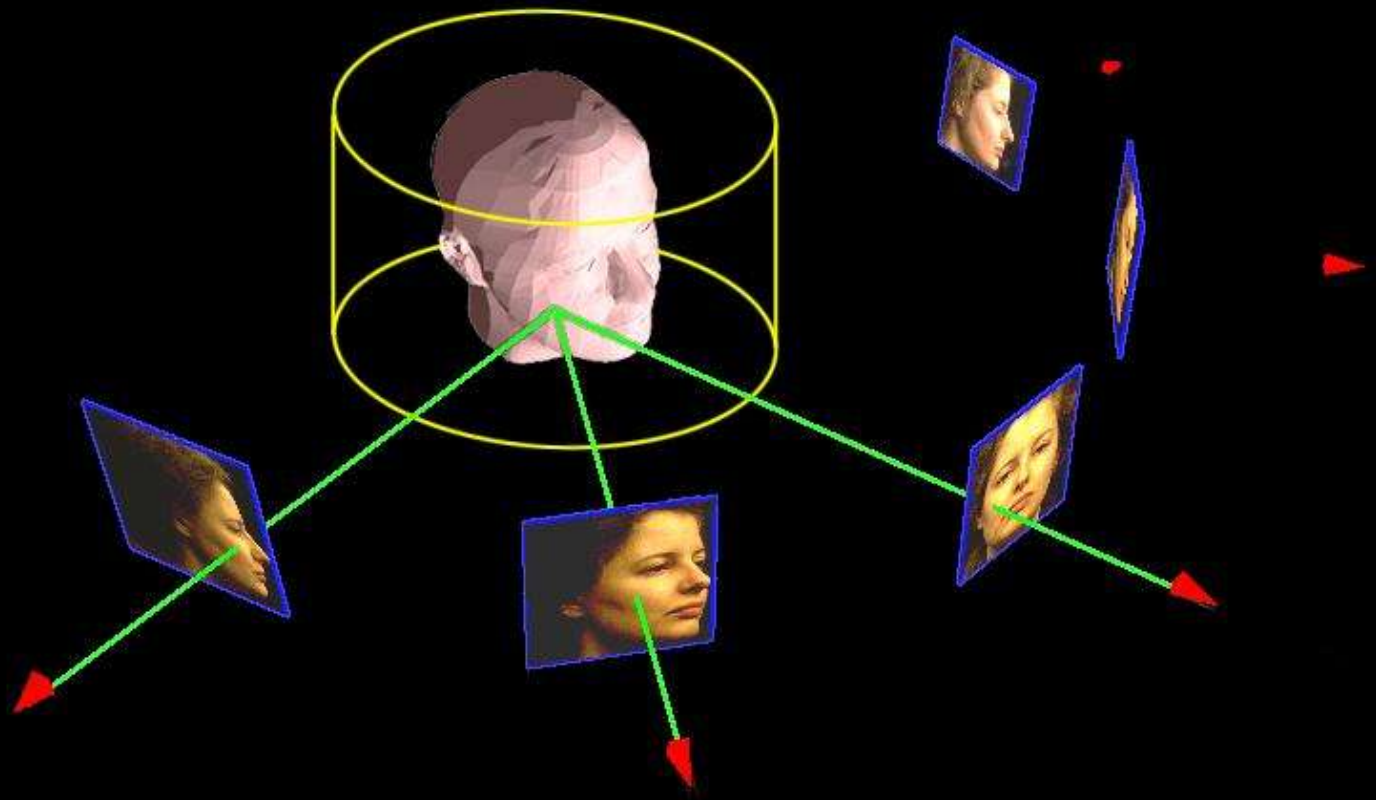


Affine Flow and PCA



3D Model Acquisition

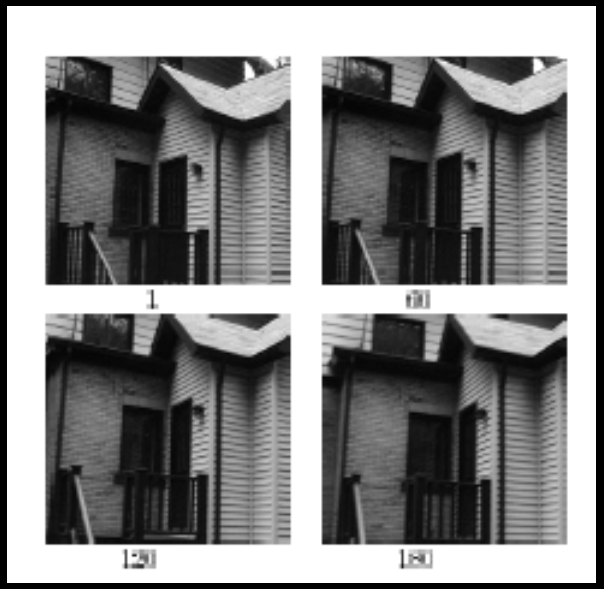
- Multi-view input: Pighin et al 98



Solution for Rigid 3D Acquisition

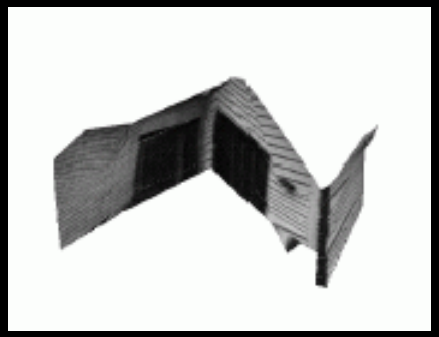
Structure from Motion:

- Tomasi-Kanade-92

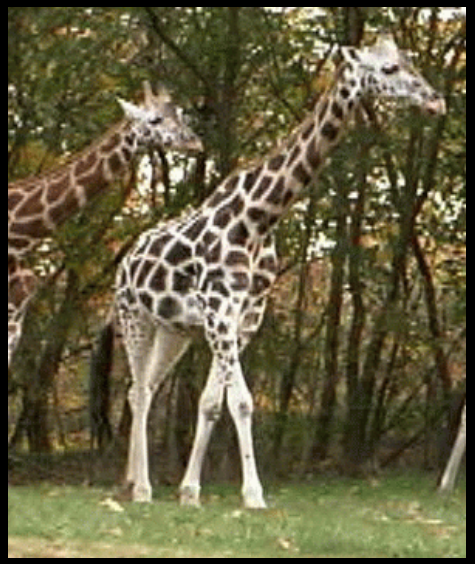


Factorization

3D Pose
3D rigid Object



Acquisition without prior model ?



- No Model available ?
- Model too generic/specific ?
- Stock-Footage only in 2D ?

Solution based on Factorization

- We want 3 things:
  - 3D non-rigid shape model
  - for each frame:
    - 3D Pose
    - non-rigid configuration (deformation)

-> Tomasi-Kanade-92:

$$W = P\,S$$

Solution based on Factorization

- We want 3 things:
  - 3D **non-rigid shape model**
  - for each frame:
    - 3D Pose
    - **non-rigid configuration (deformation)**

-> **PCA-based representations:**

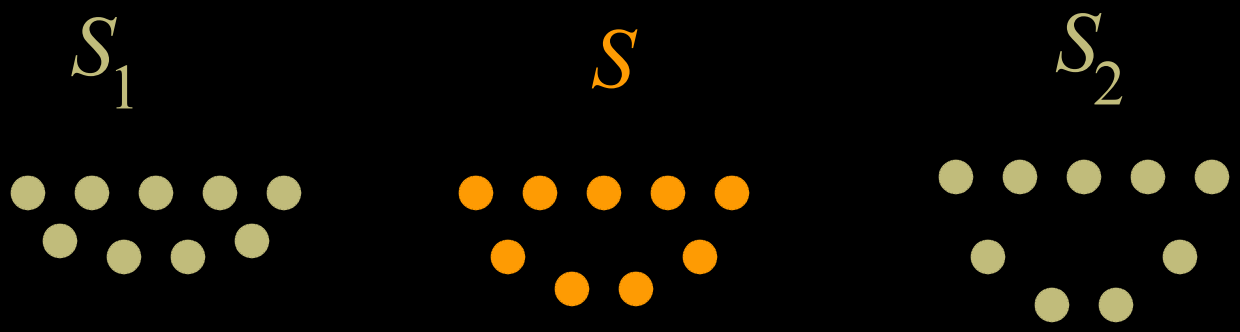
Rank K

$$W = P_{\textit{non-rigid}}\, S$$

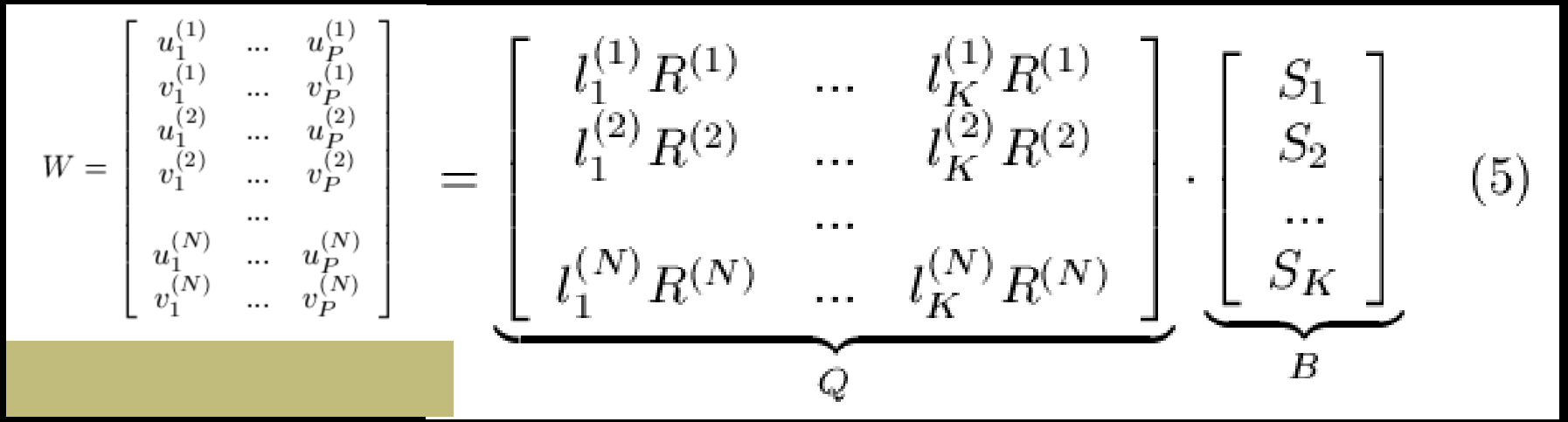
3D Shape Model

Linear Interpolation between 3D Key-Shapes:

$$S = \sum_{i=1}^{K} l_i \cdot S_i \qquad S, S_i \in \mathbb{R}^{3\times P}, l_i \in \mathbb{R} \qquad (1)$$



Basis Shape Factorization



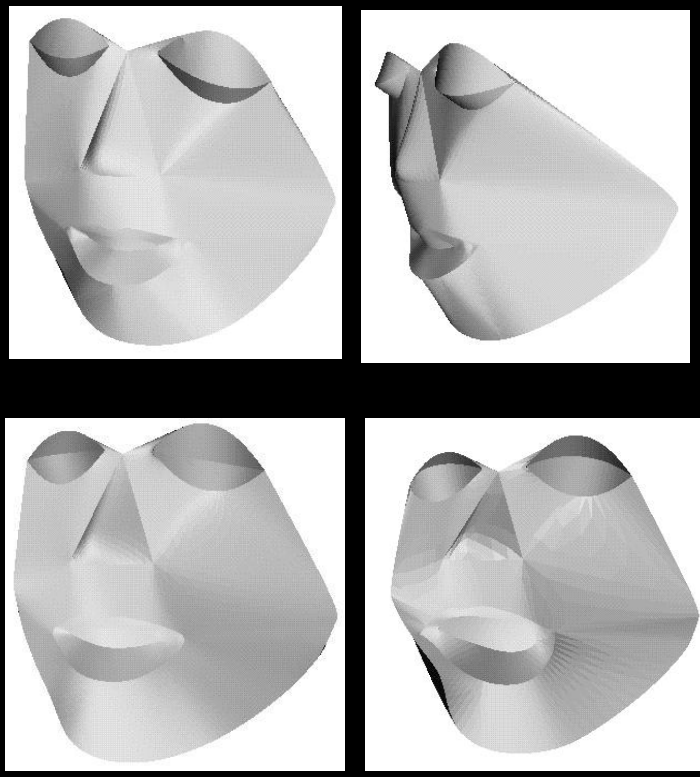
$$W = \begin{bmatrix} u_1^{(1)} & \dots & u_P^{(1)} \\ v_1^{(1)} & \dots & v_P^{(1)} \\ u_1^{(2)} & \dots & u_P^{(2)} \\ v_1^{(2)} & \dots & v_P^{(2)} \\ & \dots & \\ u_1^{(N)} & \dots & u_P^{(N)} \\ v_1^{(N)} & \dots & v_P^{(N)} \end{bmatrix} = \underbrace{\begin{bmatrix} l_1^{(1)} R^{(1)} & \dots & l_K^{(1)} R^{(1)} \\ l_1^{(2)} R^{(2)} & \dots & l_K^{(2)} R^{(2)} \\ & \dots & \\ l_1^{(N)} R^{(N)} & \dots & l_K^{(N)} R^{(N)} \end{bmatrix}}_{Q} \cdot \underbrace{\begin{bmatrix} S_1 \\ S_2 \\ \dots \\ S_K \end{bmatrix}}_{B} \quad (5)$$

Complete 2D Tracks or Flow

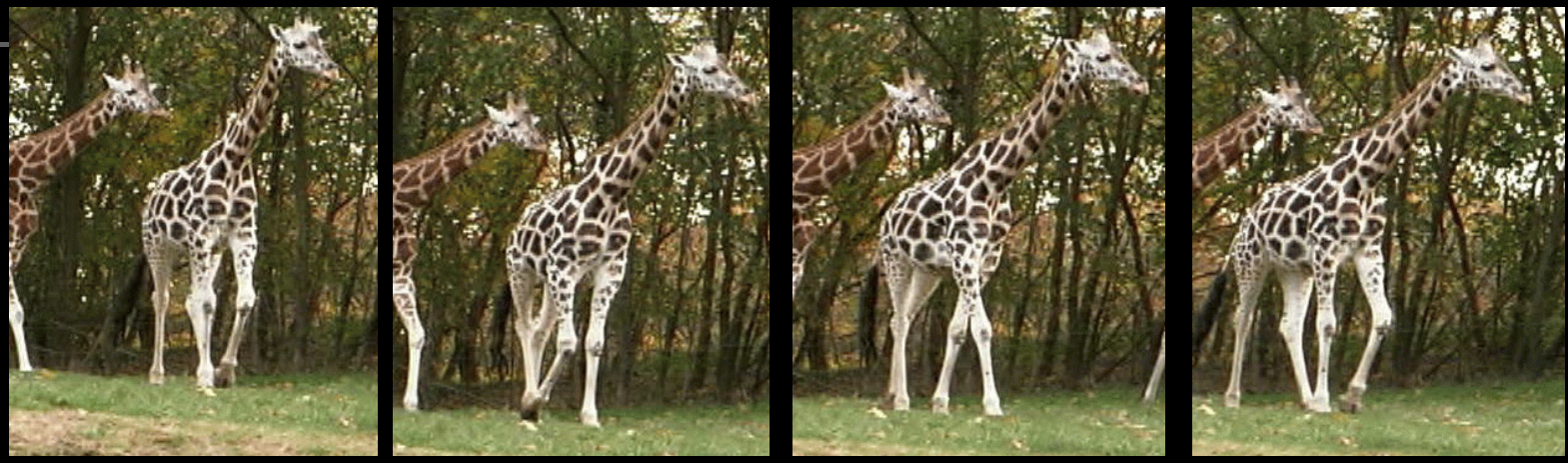
Matrix-Rank <= 3*K

Nonrigid 3D Kinematics from point tracks

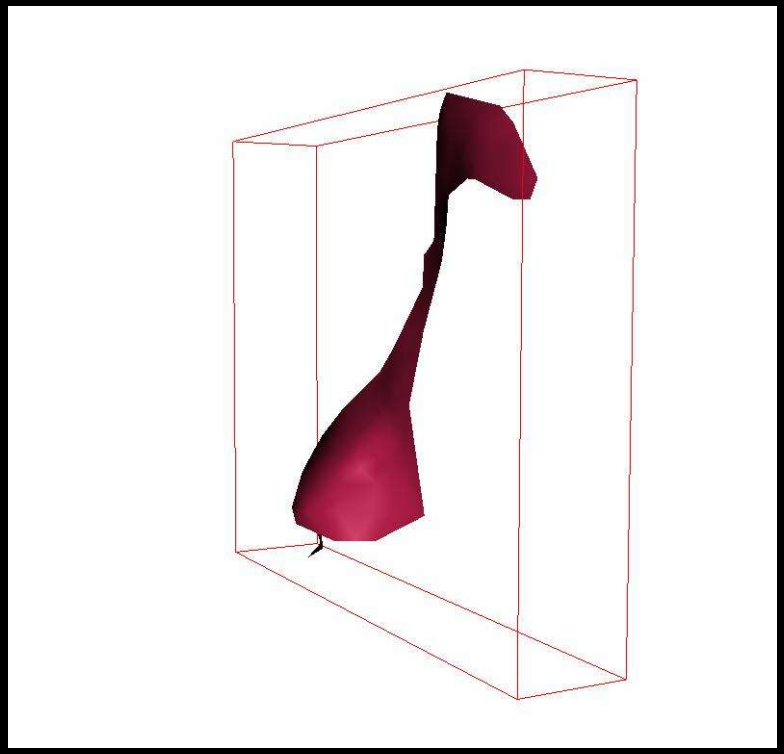
$$W = \underbrace{\begin{bmatrix} l_1^{(1)}R^{(1)} & \dots & l_K^{(1)}R^{(1)} \\ l_1^{(2)}R^{(2)} & \dots & l_K^{(2)}R^{(2)} \\ & \dots & \\ l_1^{(N)}R^{(N)} & \dots & l_K^{(N)}R^{(N)} \end{bmatrix}}_{Q} \cdot \underbrace{\begin{bmatrix} S_1 \\ S_2 \\ \dots \\ S_K \end{bmatrix}}_{B} \quad (5)$$

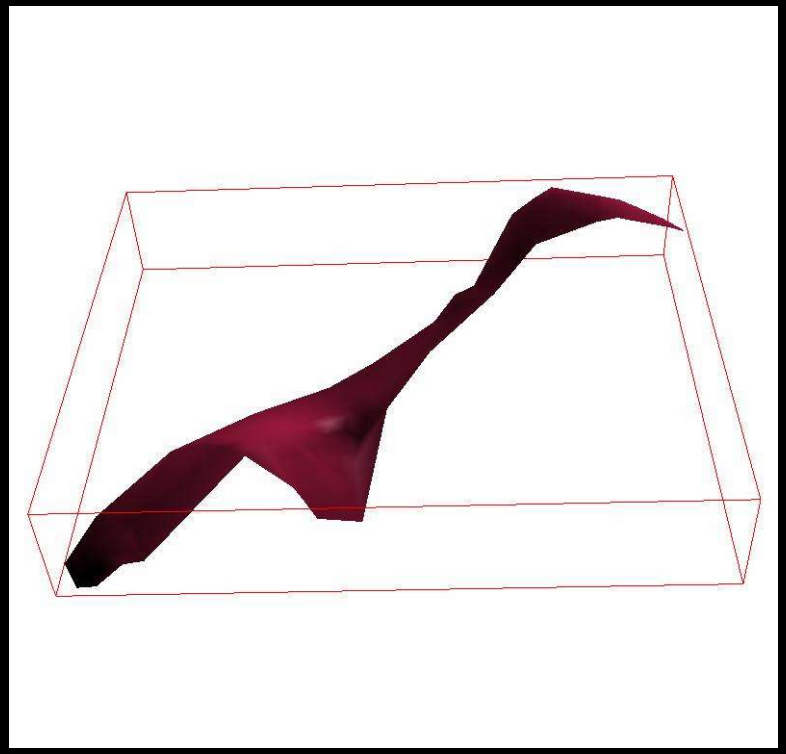


Nonrigid 3D Kinematics from dense flow

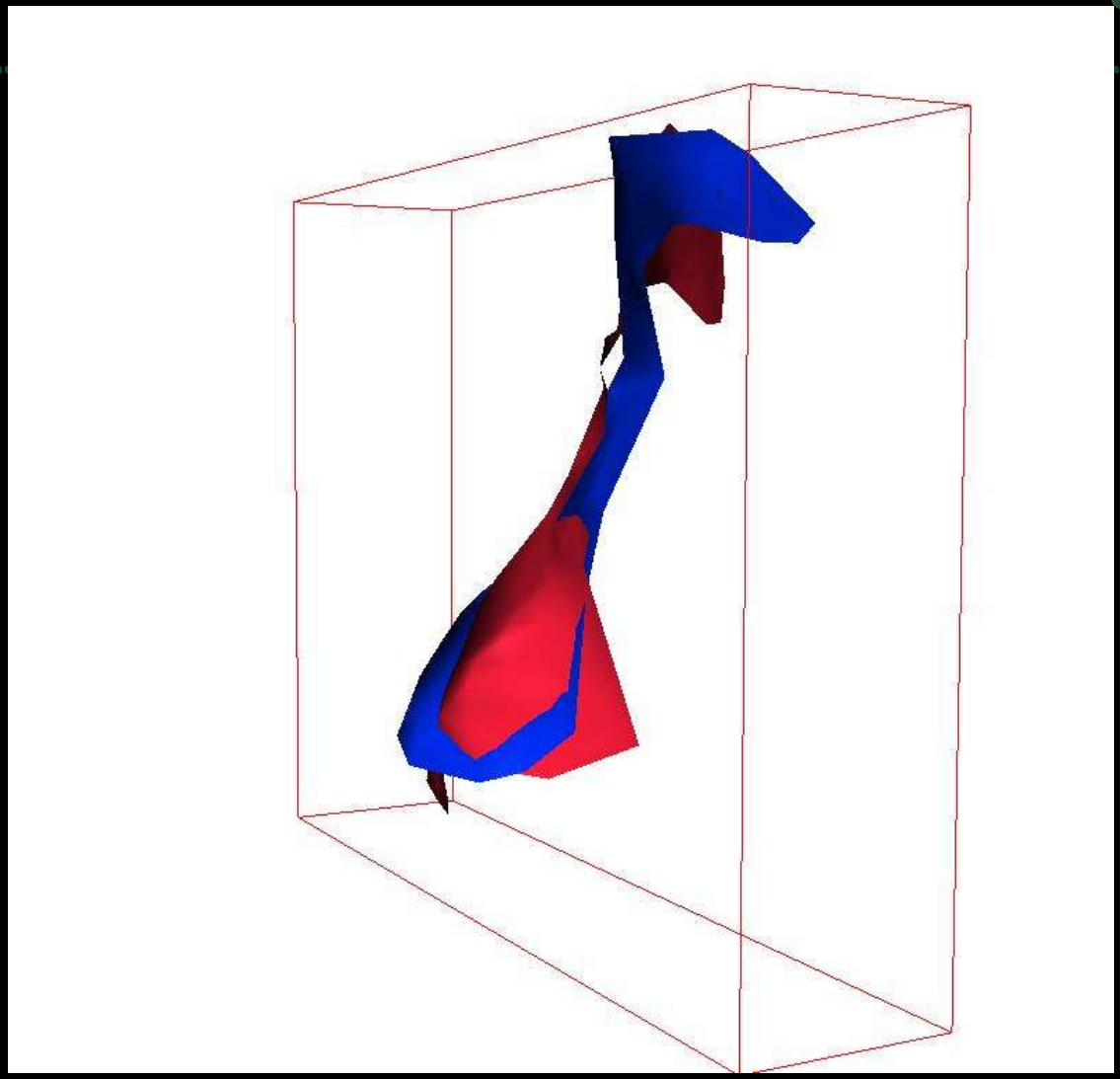


Nonrigid 3D Kinematics from dense flow





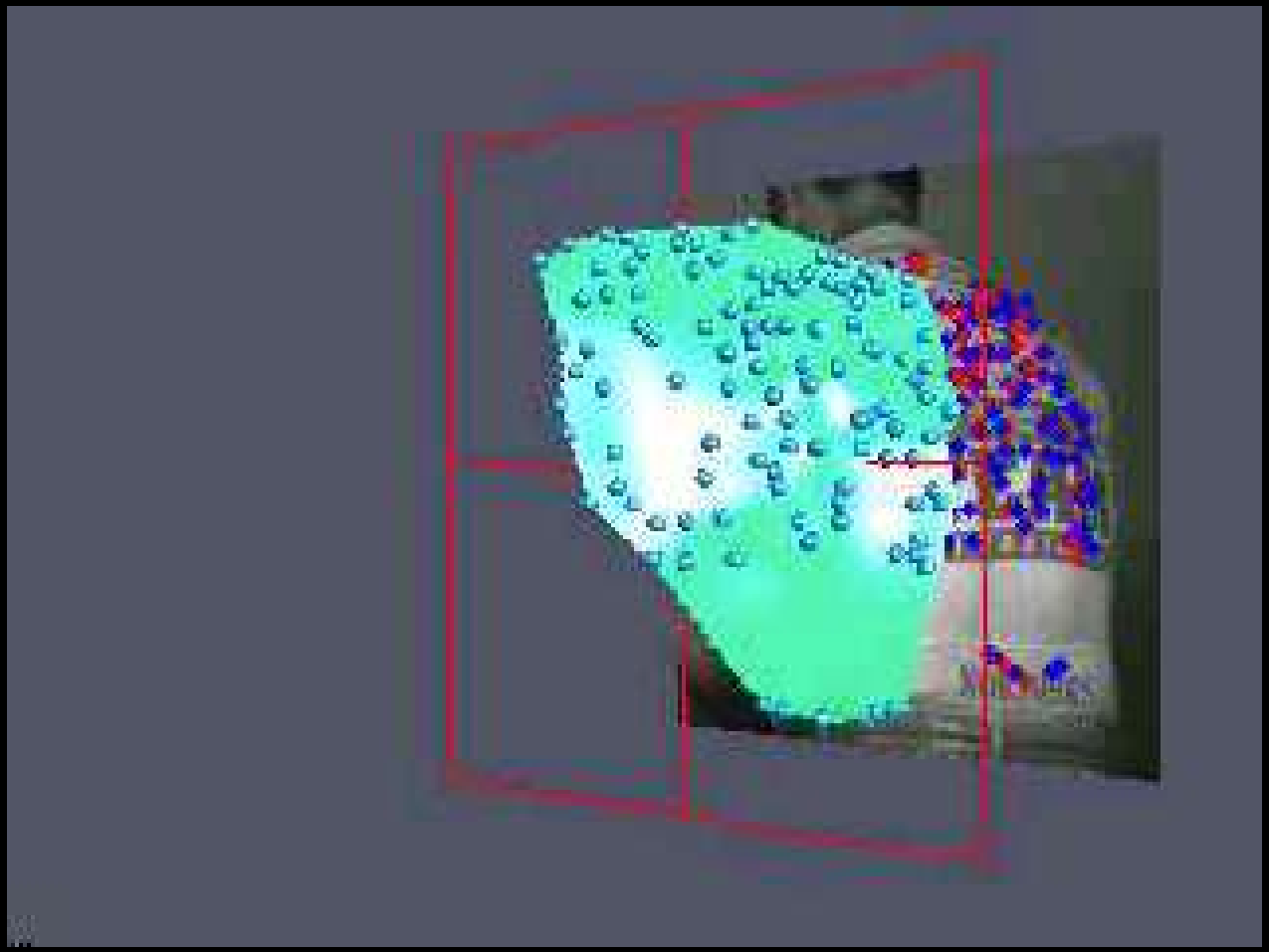
Nonrigid 3D Kinematics from dense flow



Nonrigid 3D Kinematics from dense flow

SIGGRAPH2006

- Synthesis
- Modeling
- Motion Capture



Markerless Face Capture - Summary -

- Single / Multi Camera Input
- 2D / 3D Output
- Real-time / Off-line
- Interactive-Refinement / Face Dependent / Independent
- Make-up / Natural
- Flow / Contour / Texture / Local / Global Features
- Hand Crafted / Data Driven
- Linear / Nonlinear Models / Tracking

SIGGRAPH Course 30: Performance-Driven Facial Animation

Section:

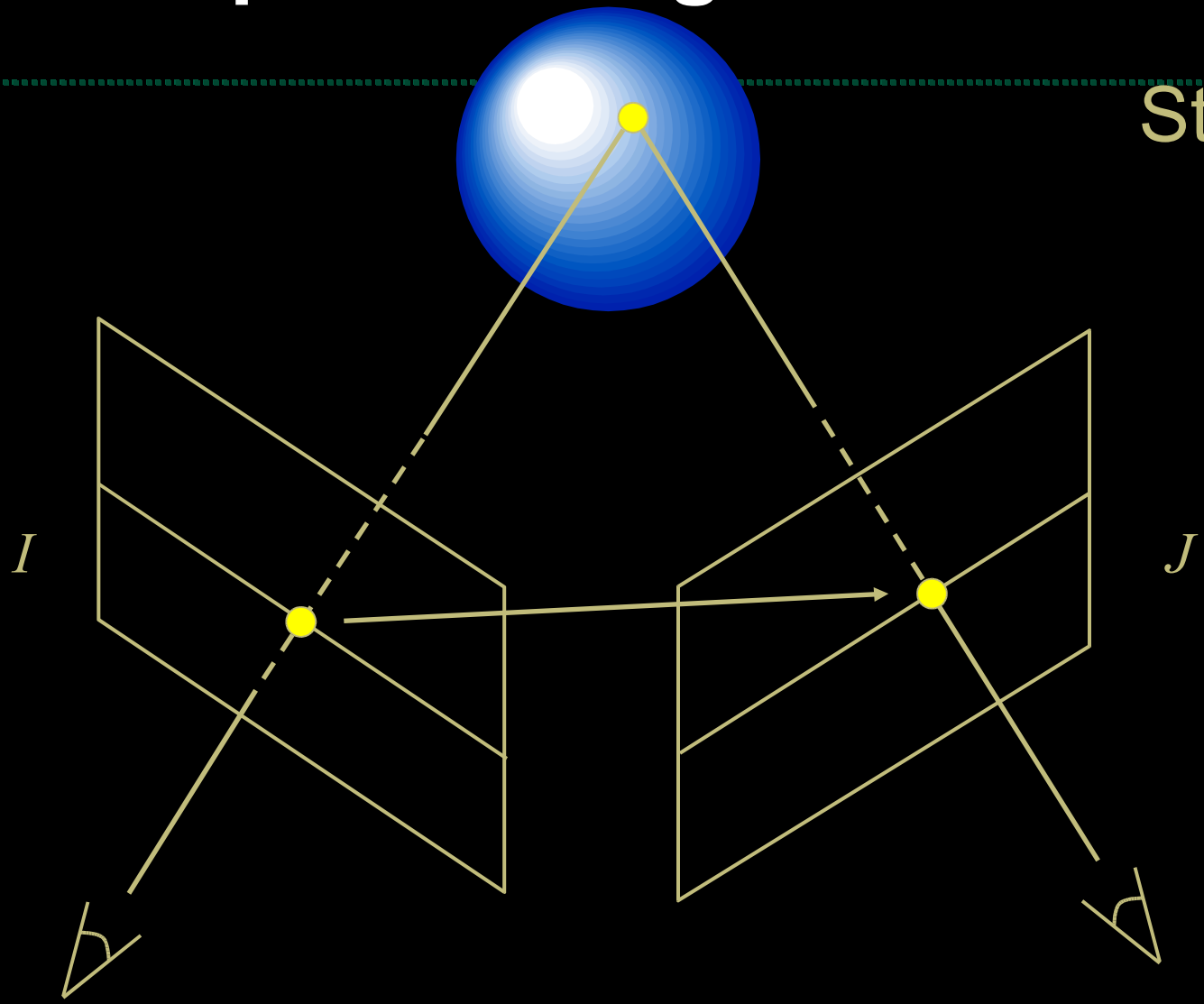
Markerless Face Capture and Automatic Model Construction

Part 2: Li Zhang, Columbia University

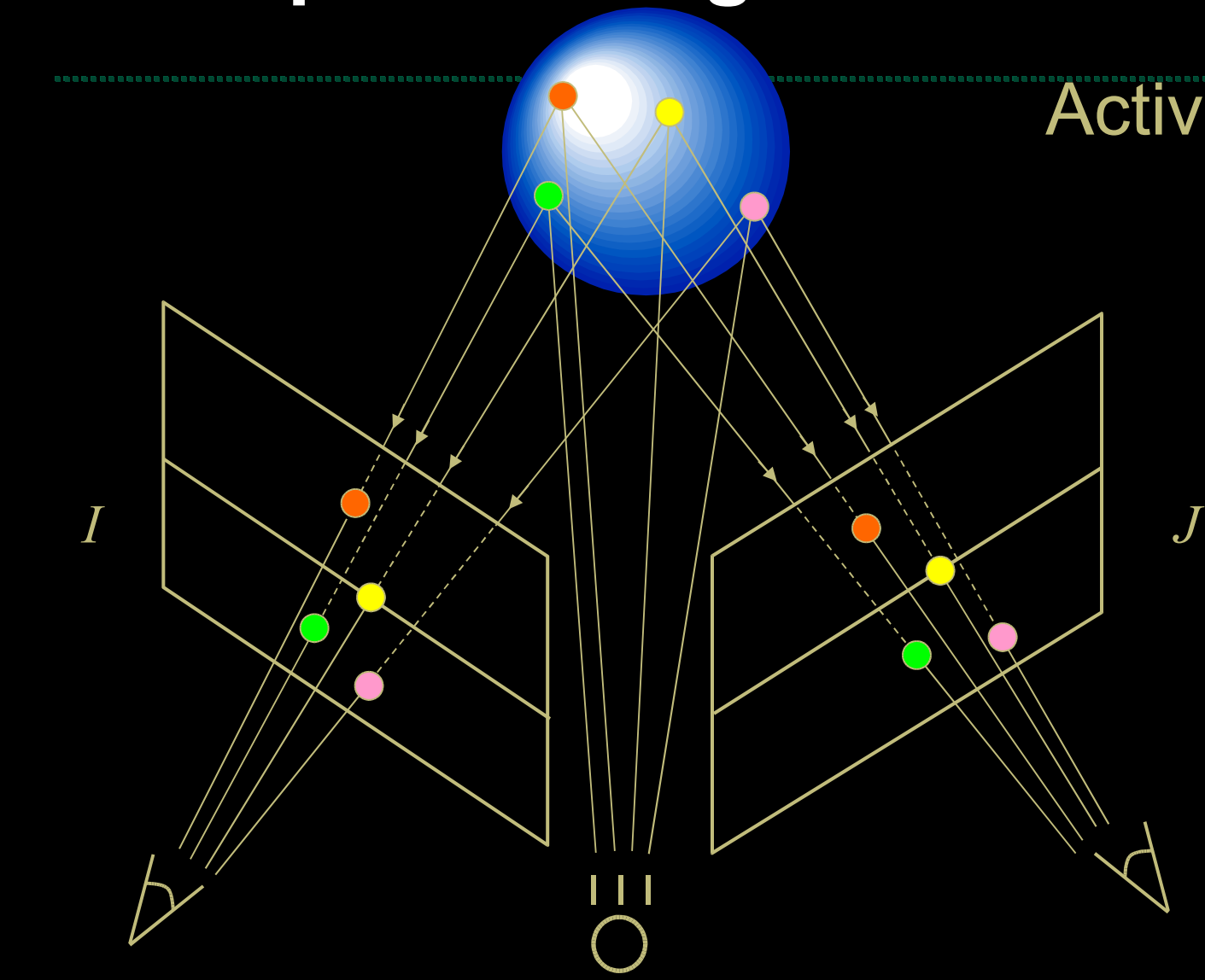
Outline

1. Scanning face models
   - Triangulation methods
   - Non triangulation methods
2. Dense facial motion capture
   - Marker based capture
   - Template fitting for face scans

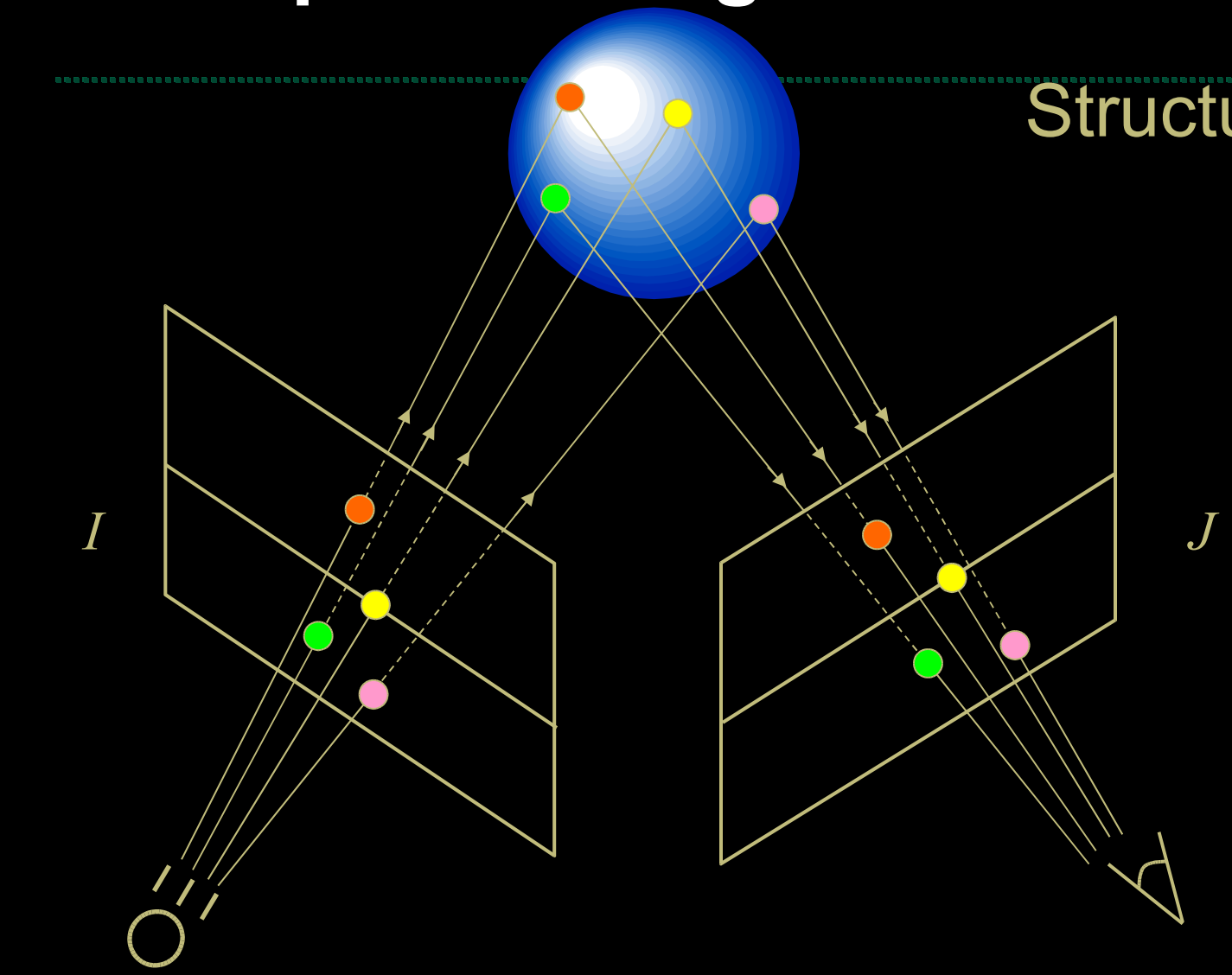
Principle 1: triangulation



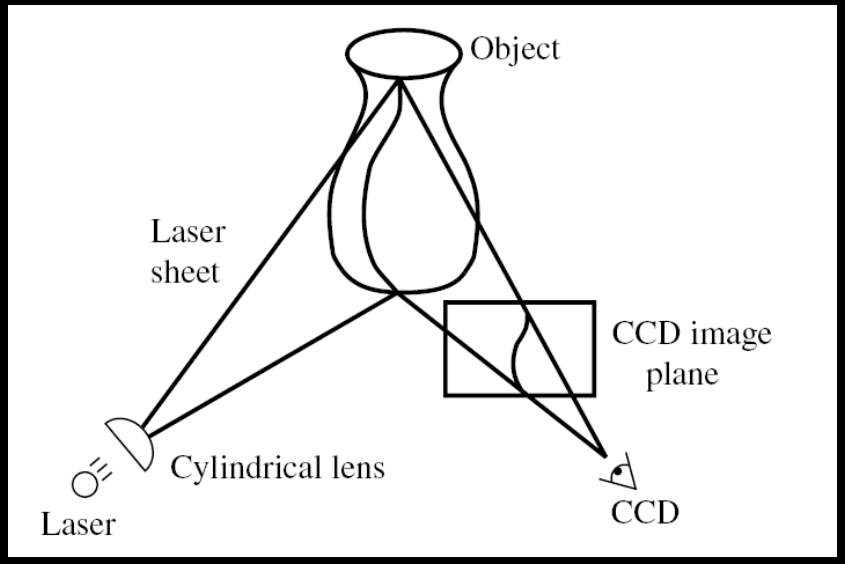
Principle 1: triangulation



Principle 1: triangulation



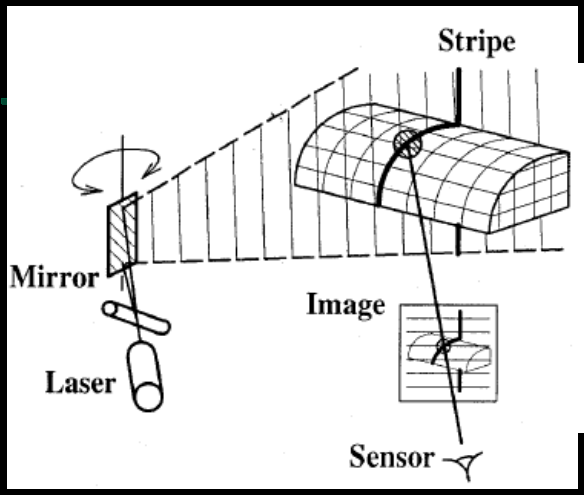
Laser scanner

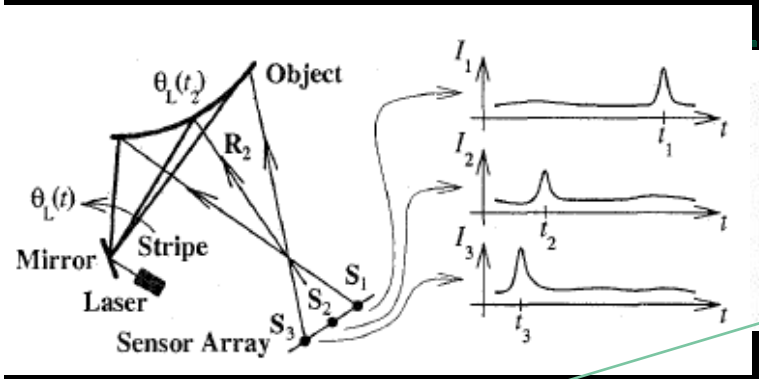


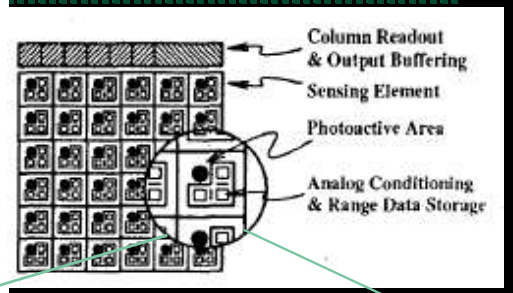
+ very accurate <0.01mm
− >10sec per scan

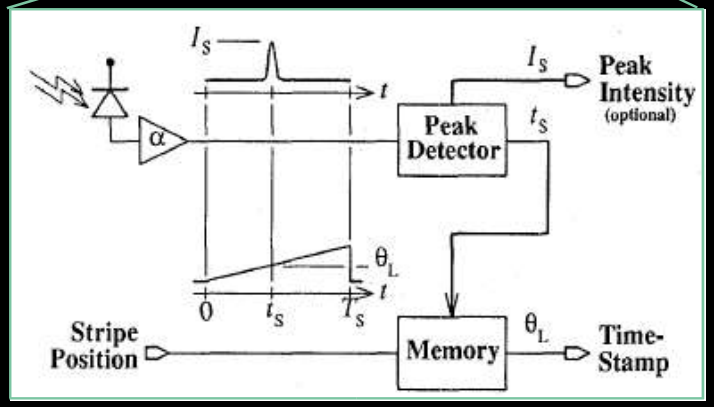
Cyberware® face and head scanner

Fast laser scanner (temporal)









\+ Fast – up to 1000Hz

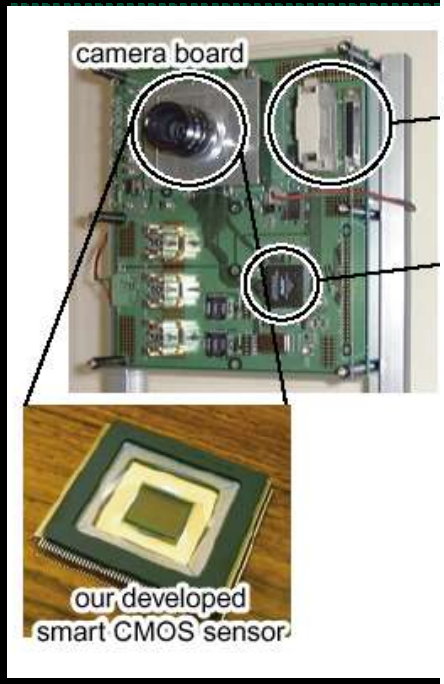
− Customized device

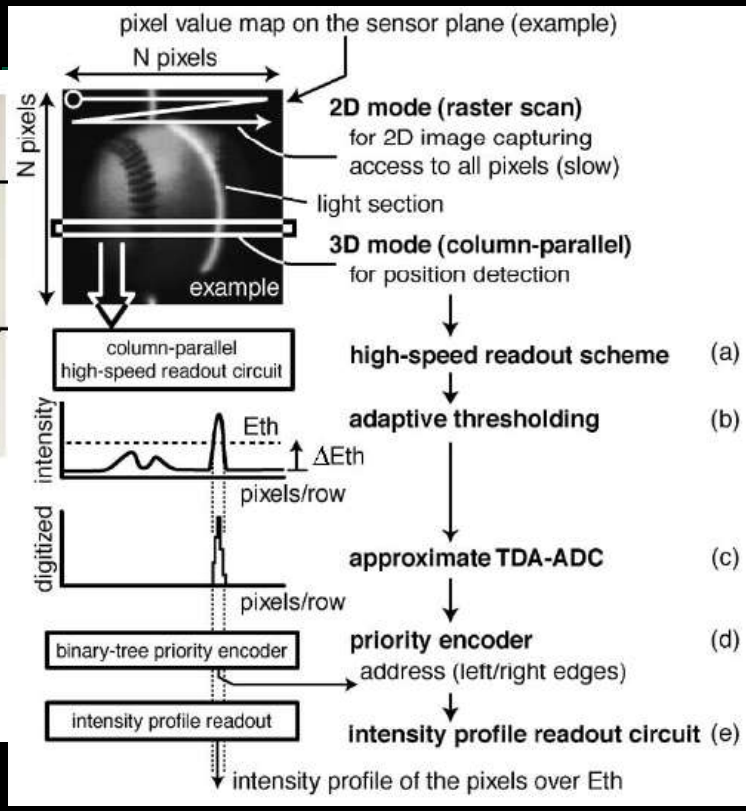
A. Gruss, S. Tada, and T. Kanade "A VLSI Smart Sensor for Fast Range Imaging," ICIRS 1992

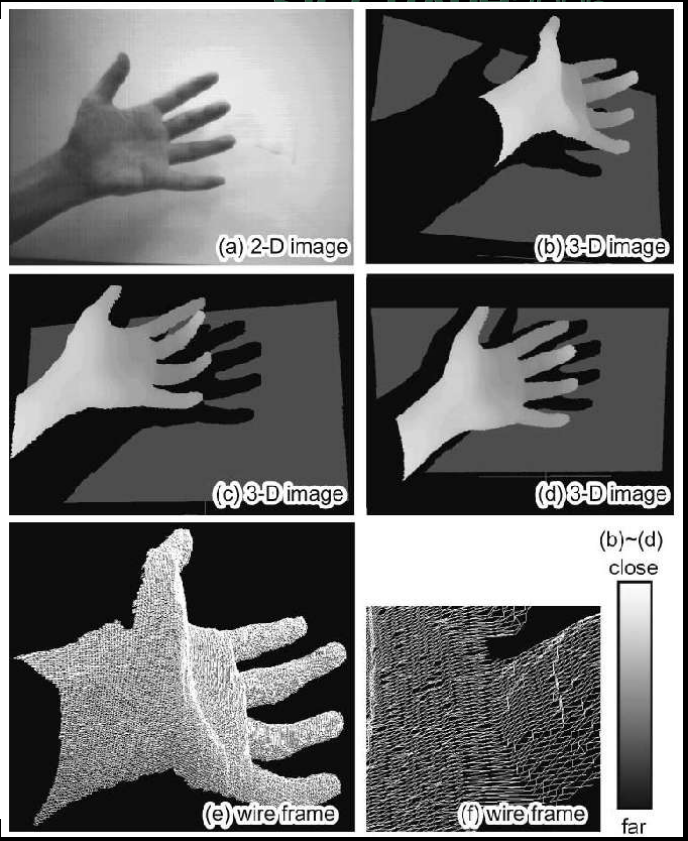
Working Volume: 350-500mm - Accuracy: 0.1%

Spatial Resolution: 28x32 - Speed: 1000Hz

Fast laser scanner (spatial)



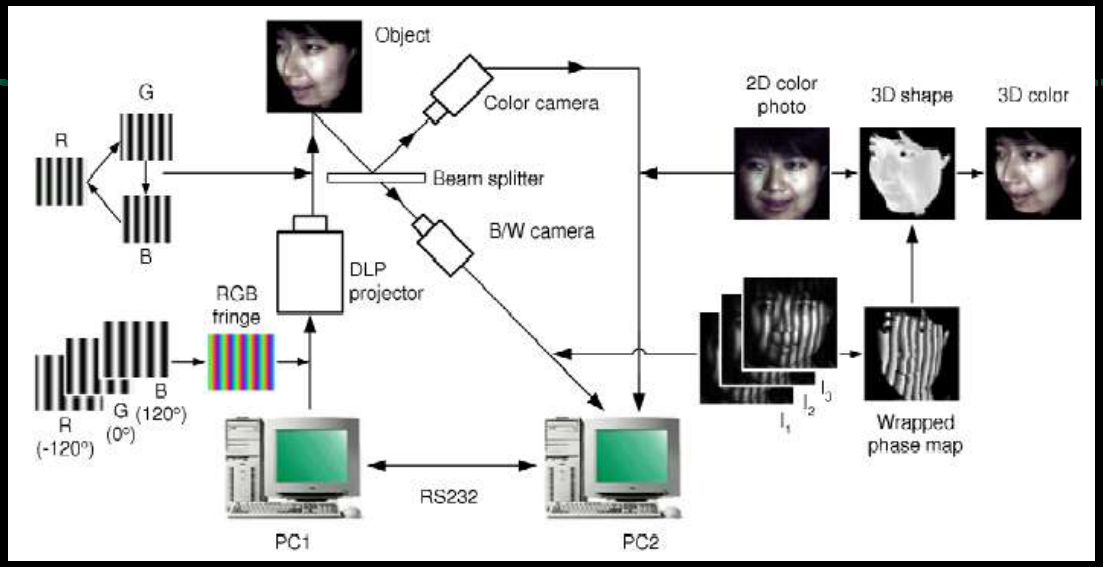


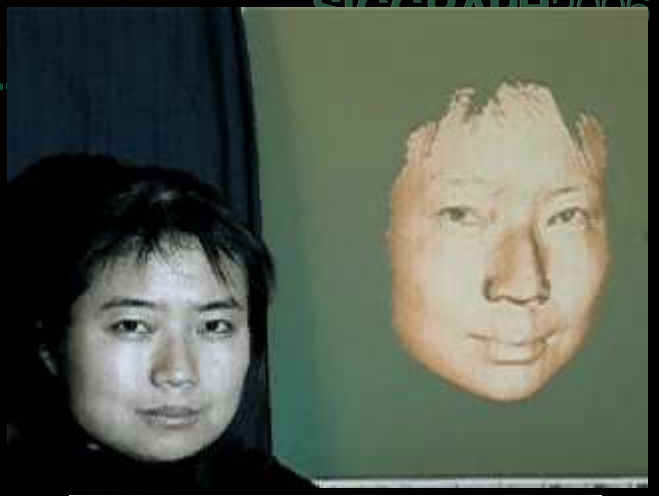


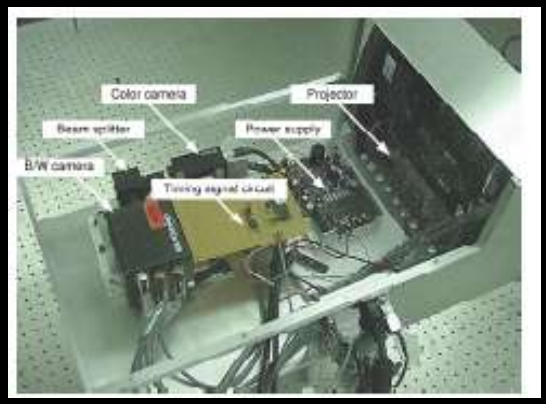
Possible issue: Stripes within a range map are not simultaneously measured.

Oike, Y. Ikeda, M. Asada, K., “Design and implementation of real-time 3-D image sensor with 640x480 pixel resolution”, IEEE Journal of Solid-State Circuits, 2004.
Working Volume: 1200mm - Accuracy: 0.07%
Spatial Resolution: 640x480 - Speed: 65Hz

Digital fringe range sensor



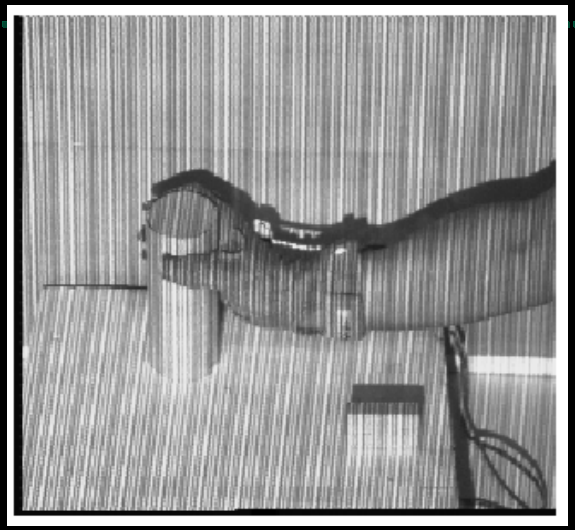


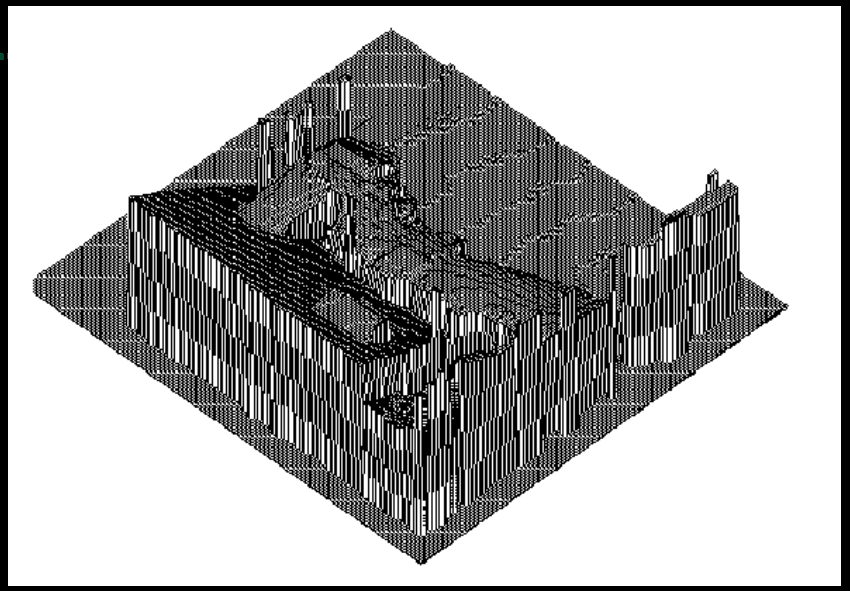


+ Real time performance
− Phase ambiguity near discontinuities
− Customized device
− Capture from one viewpoint at a time

P. Huang, C. Zhang, F. Chiang, “High-speed 3-D shape measurement based on digital fringe projection”, Journal of Optical Engineering, 2003
Working Volume: 10-2000mm - Accuracy: 0.025%
Spatial Resolution: 532x500 - Speed: 120Hz

Active multi-baseline stereo

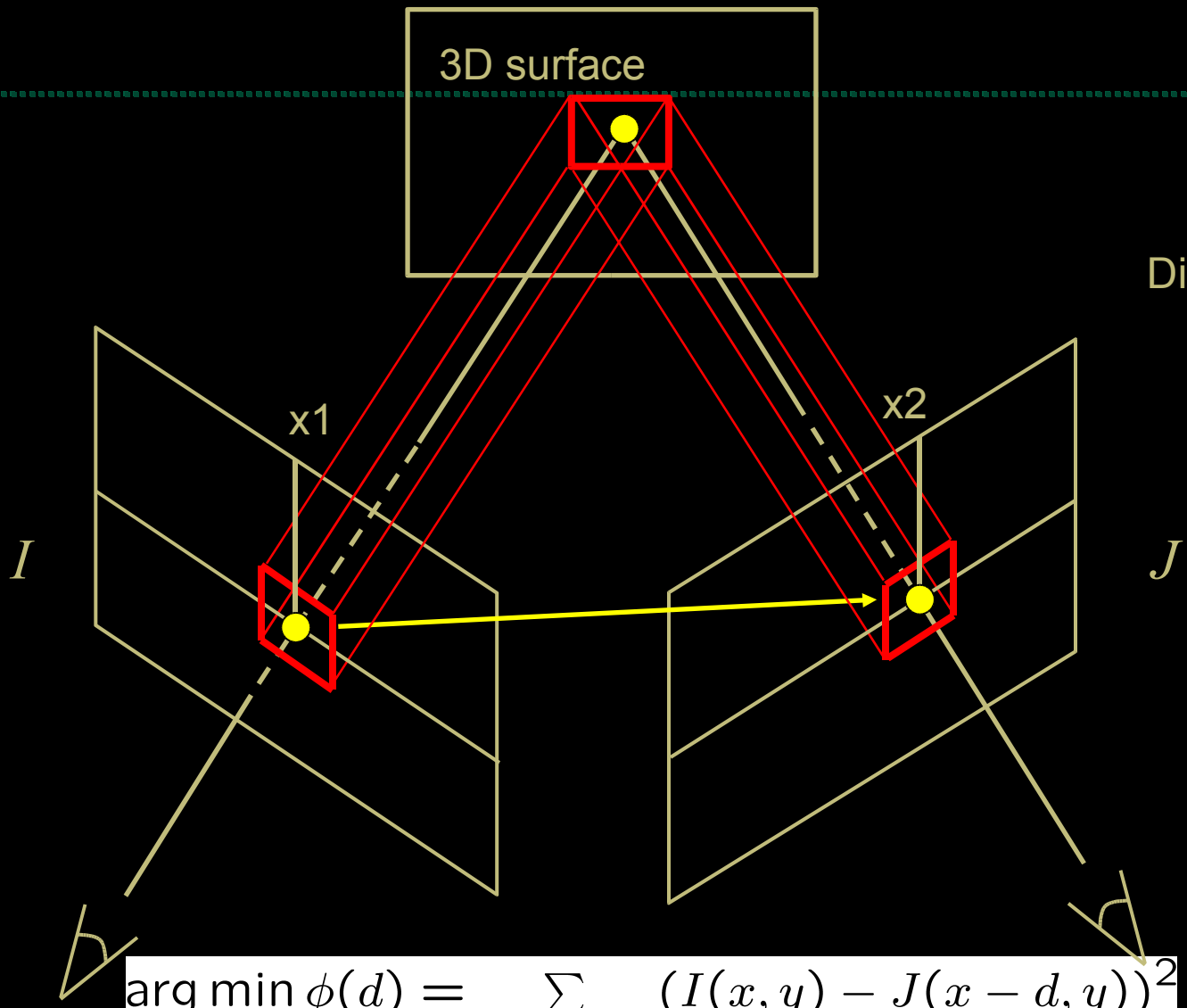




+ Only require one image per camera
+ Simultaneous multi-view capture
− Less accurate than laser scanners or fringe scanners

S. Kang, J.A. Webb, C. Zitnick, and T. Kanade, “A Multibaseline Stereo System with Active Illumination and Real-time Image Acquisition,” ICCV 1995.
Working Volume: 2000mm - Accuracy: 0.1%
Spatial Resolution: 100x100? - Speed: 30Hz

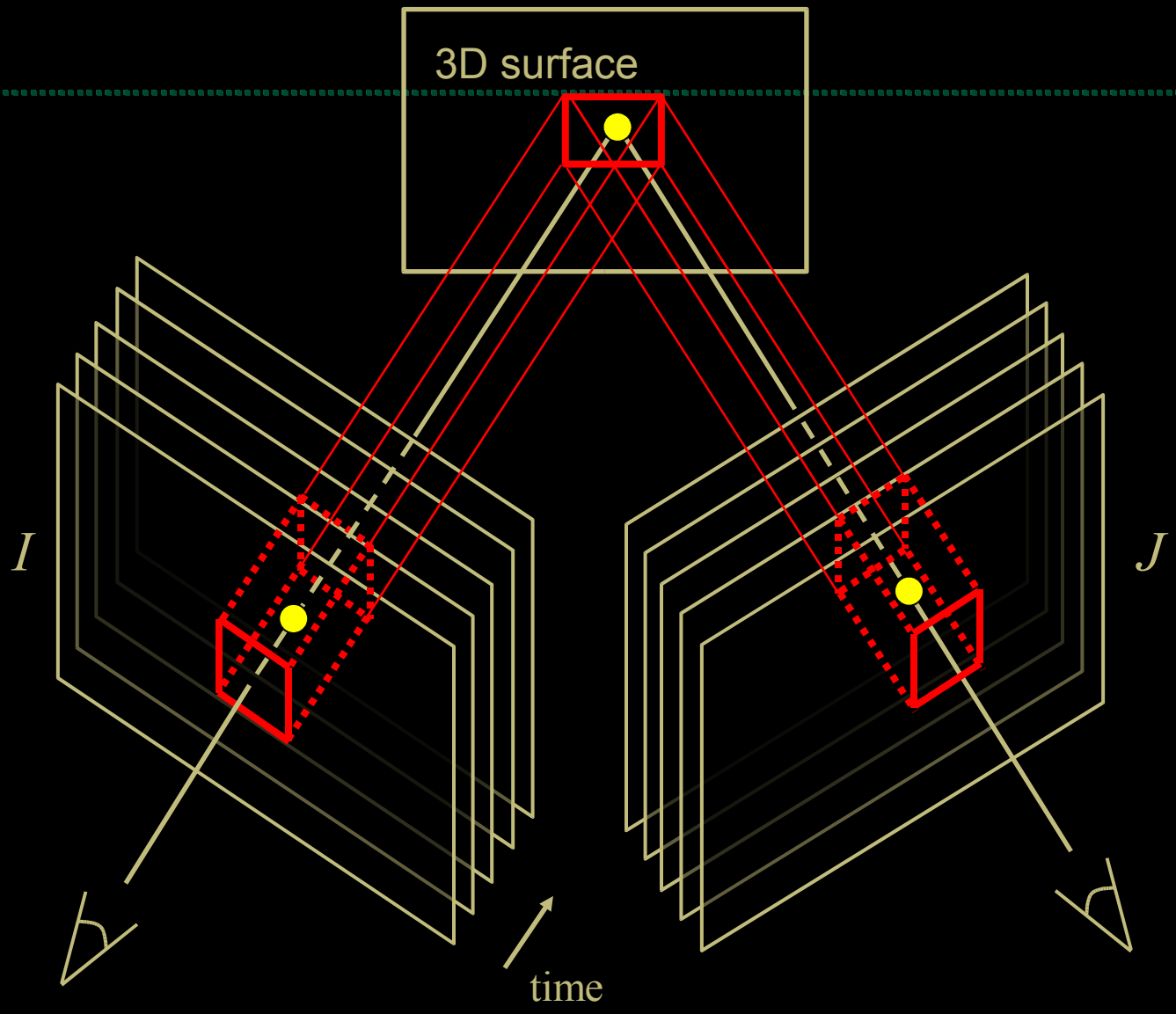
Spacetime stereo



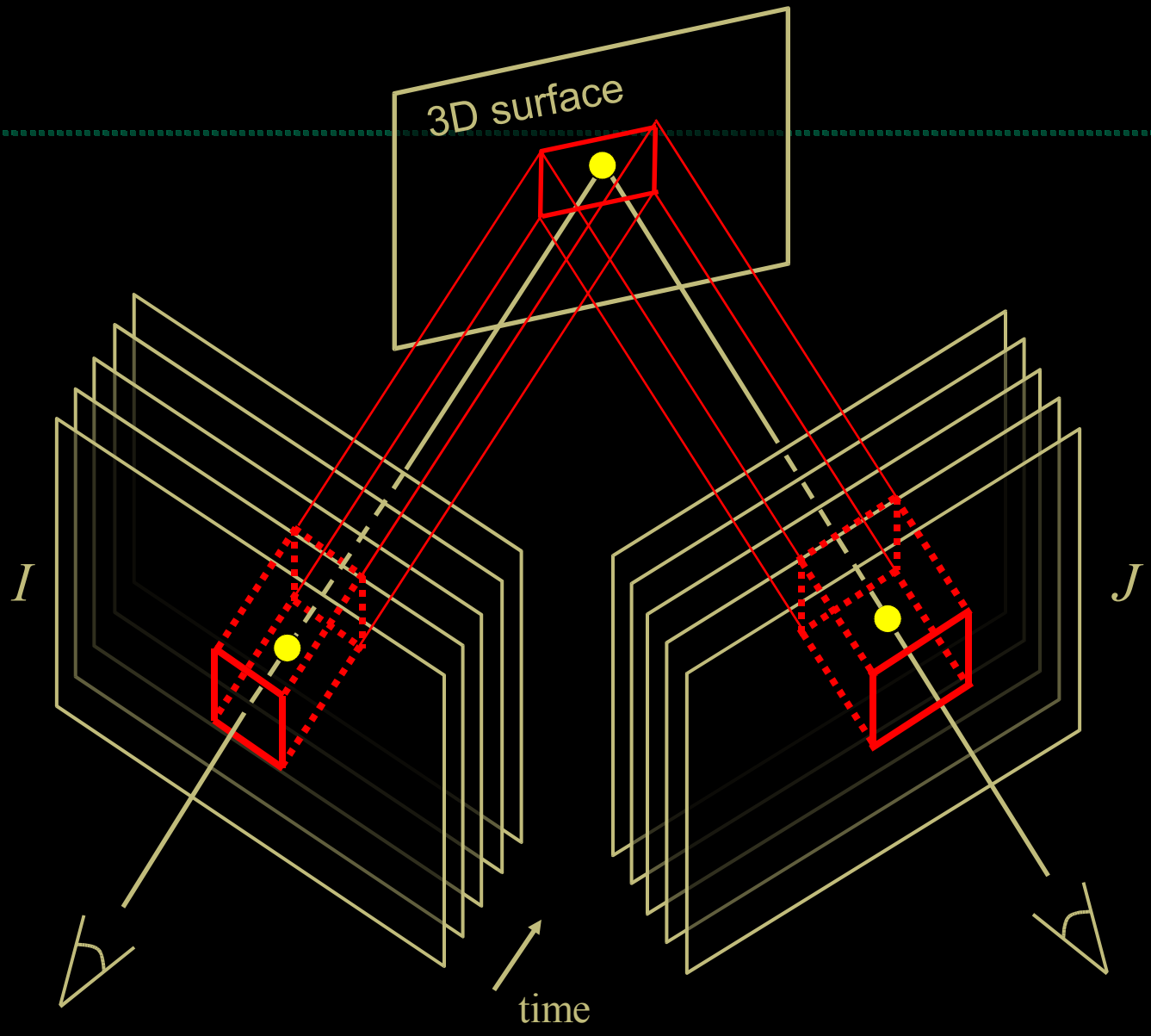
Disparity: d = x1 – x2

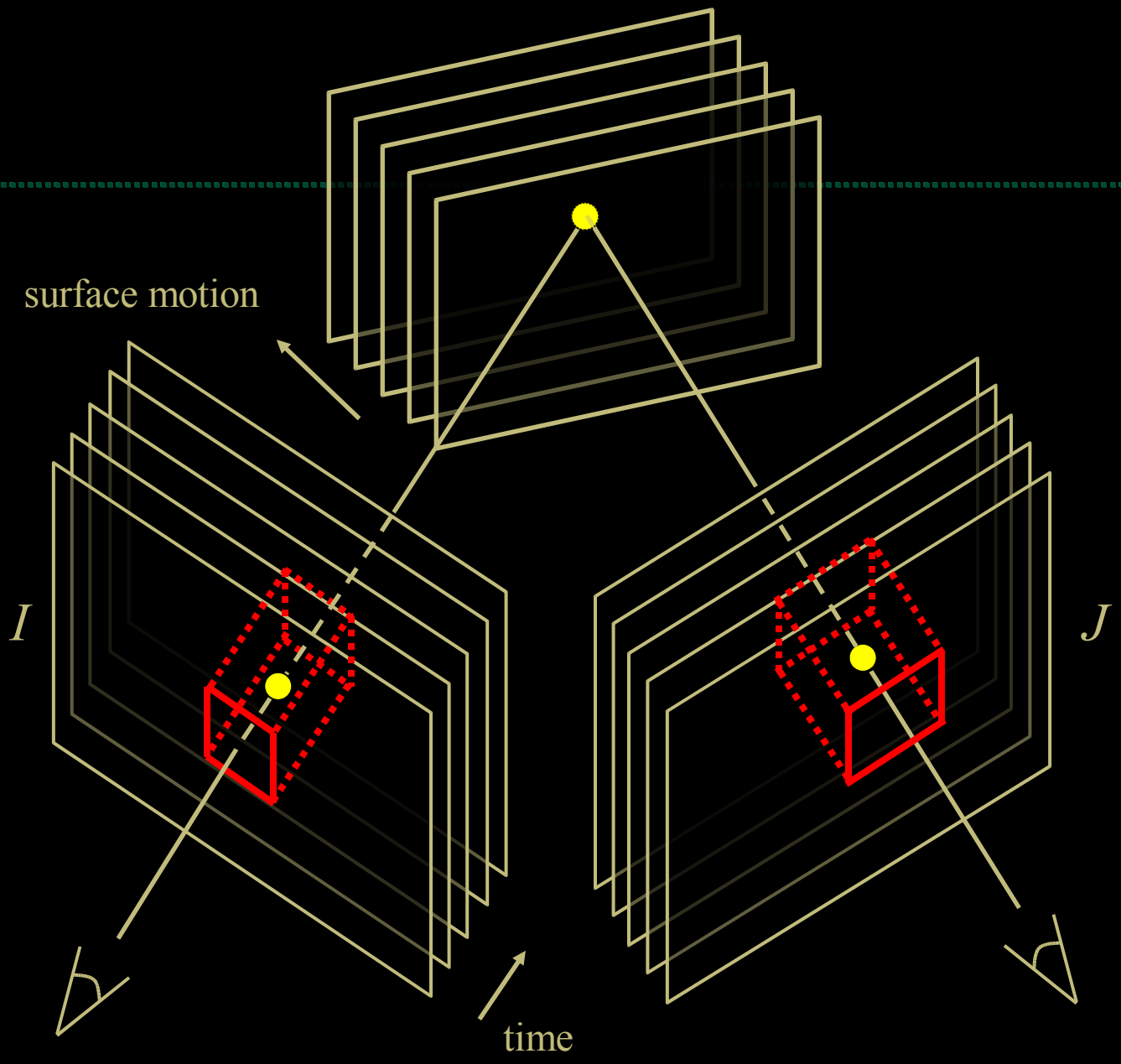
$$\arg\min_{d} \phi(d) = \sum_{(x,y)\in W} (I(x,y) - J(x-d,y))^2$$

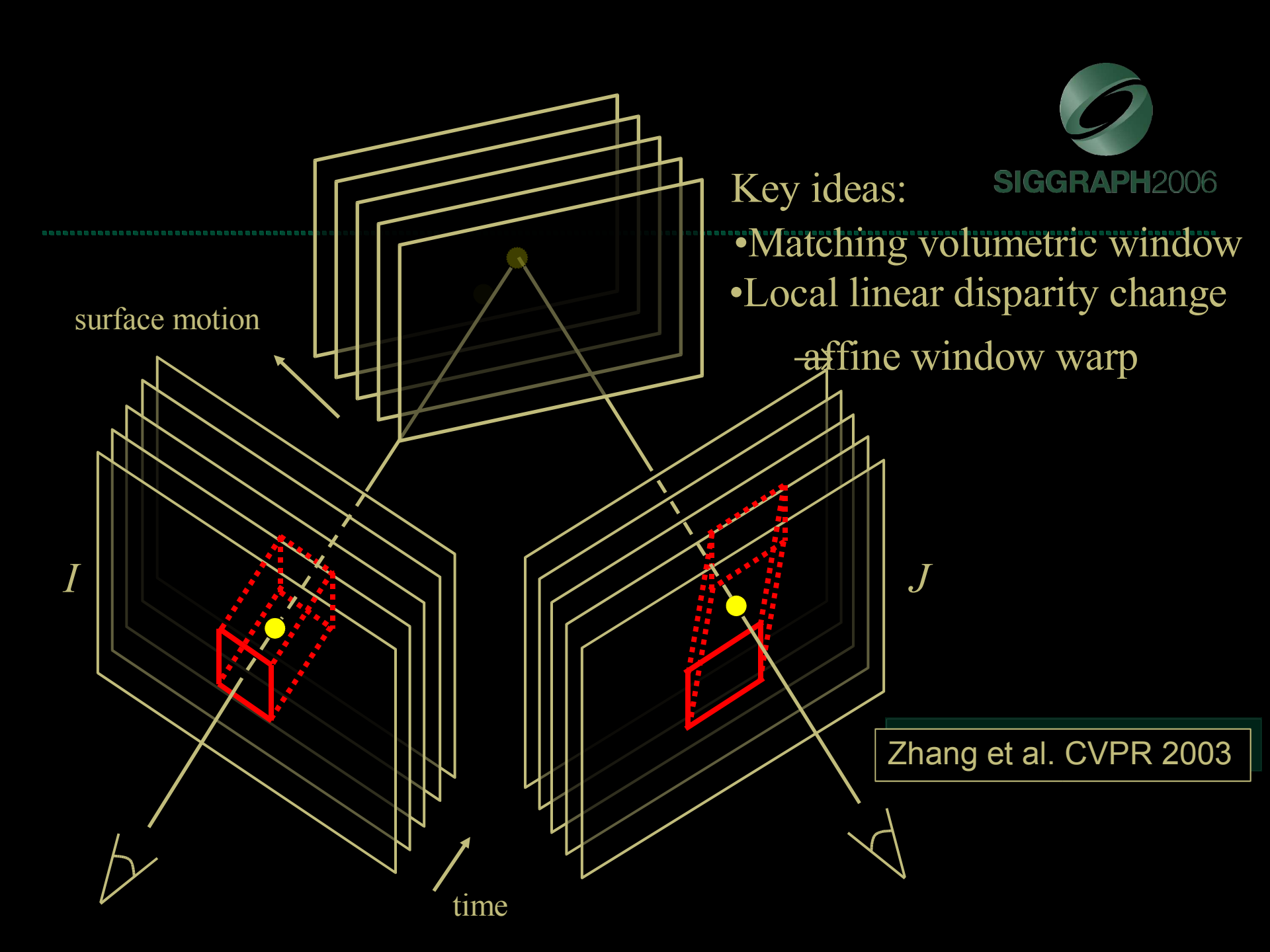
Spacetime stereo



Spacetime stereo

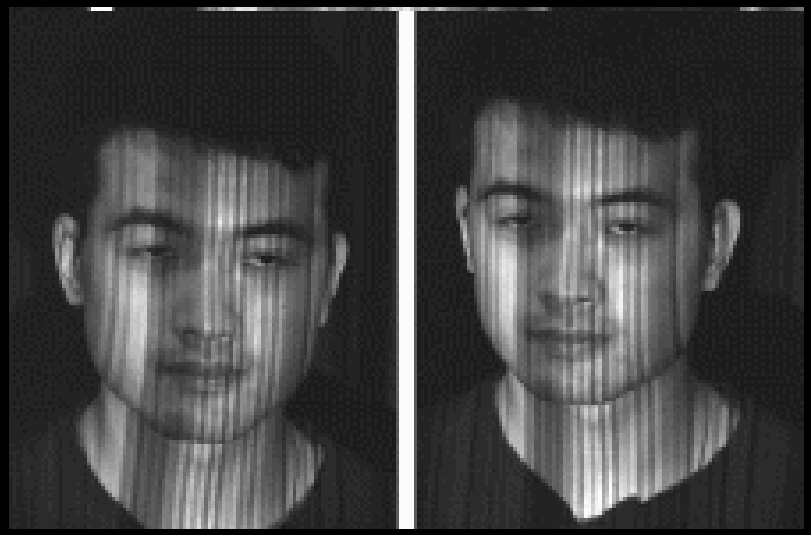



surface motion
I
J
time


SIGGRAPH2006
Key ideas:
•Matching volumetric window
•Local linear disparity change
affine window warp
surface motion
I
J
time
Zhang et al. CVPR 2003

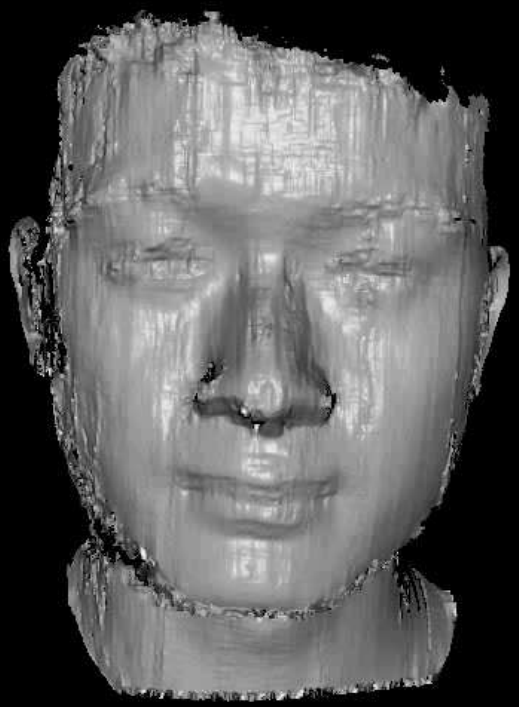
Spacetime stereo

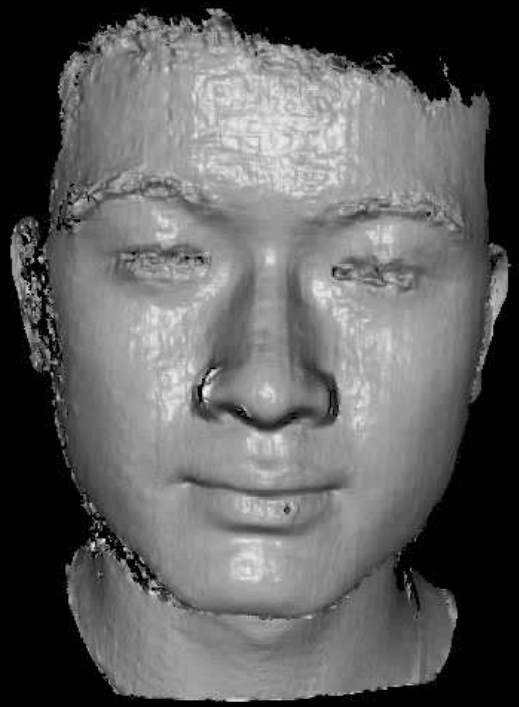
Input stereo video:



656x494x60fps videos captured by firewire cameras

Face Example: Result Comparison





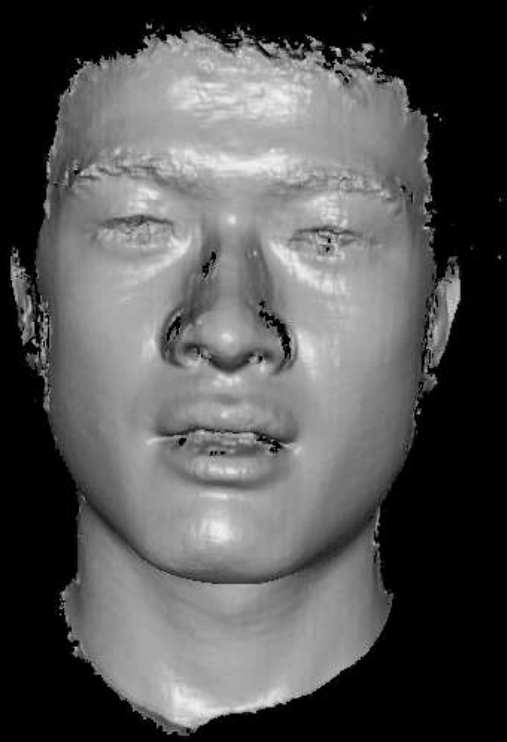
Frame-by-frame stereo

WxH=15x15 window

Spacetime stereo

WxHxT=9x5x5 window

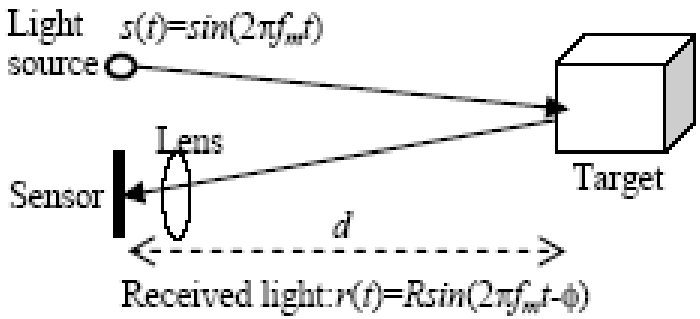
Face Example: Mouth motion

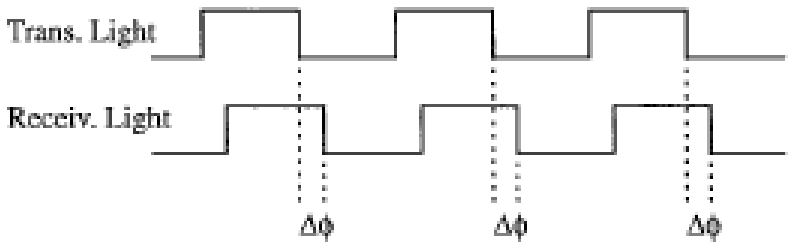


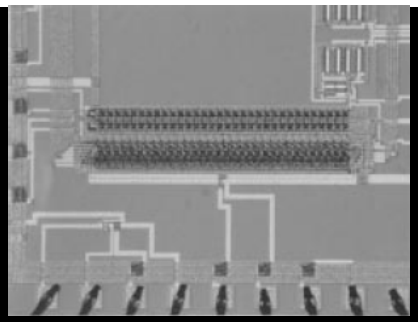
+ More accurate and stable than frame by frame stereo
+ Simultaneous multi-view capture
− Offline computation (3min per frame)

Zhang, L., Curless, B., Seitz, S., “Spacetime stereo”, CVPR 2003,
Working Volume: 300mm - Accuracy: 0.1%
Spatial Resolution: 640x480- Speed: 60Hz

Principle 2: Time-of-flight



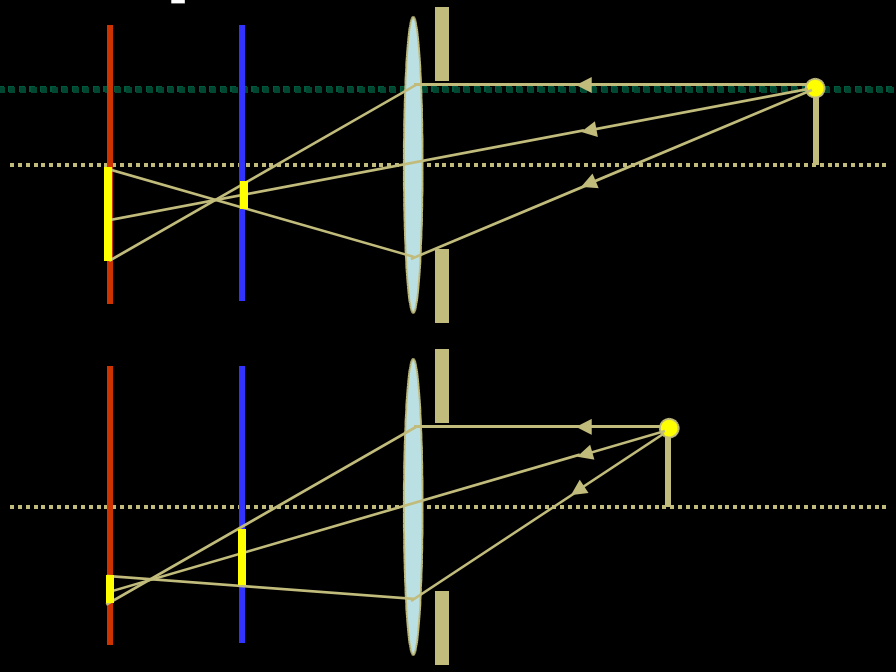




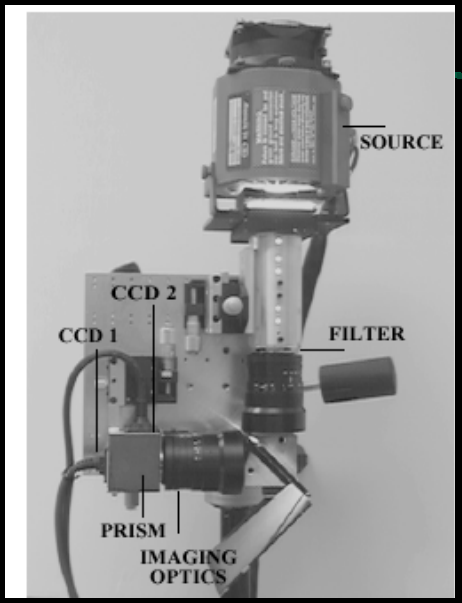
+ No baseline, no parallax shadows
+ Mechanical alignment is not as critical
− Low depth accuracy
− Single viewpoint capture

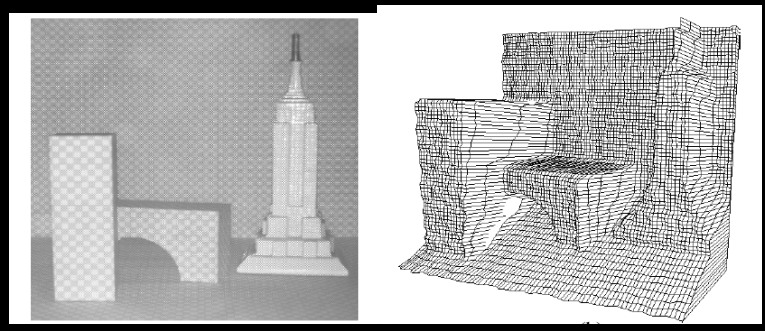
Miyagawa, R., Kanade, T., "CCD-Based Range Finding Sensor", IEEE Transactions on Electron Devices, 1997
Working Volume: 1500mm - Accuracy: 7%
Spatial Resolution: 1x32- Speed: ??

Principle 3: Defocus



Principle 3: Defocus





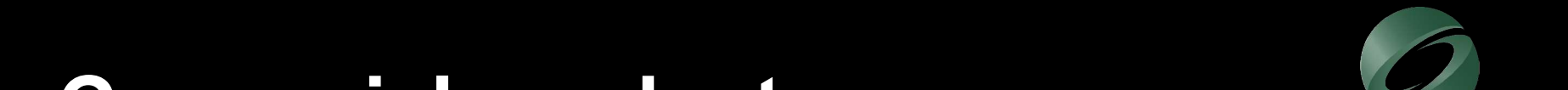
+ Hi resolution and accuracy, real-time
− Customized hardware
− Single view capture?

Nayar, S.K., Watanabe, M., Noguchi, M., “Real-Time Focus Range Sensor”, ICCV 1995
Working Volume: 300mm - Accuracy: 0.2%
Spatial Resolution: 512x480 - Speed: 30Hz

Commercial products

Company	Working principle	XY	Depth	Speed
Cyberware	Laser	resolution >500x500	accuracy 0.01mm	>10sec per
XYZRGB	Laser	Very high	0.01mm	scan >10sec per
Eyetronics	Structrued light	High	<2mm	scan <0.1sec
3Q	Active stereo	High	?	<0.1sec
3DV	Time of flight	720x486	1-2cm	30Hz
Canesta	Time of flight	64x64	1cm	30Hz

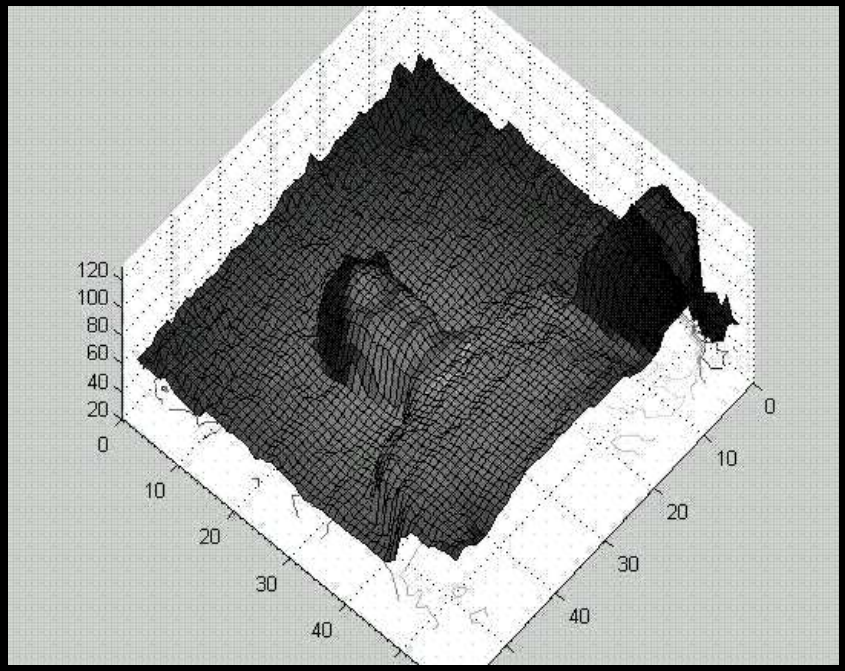
Comercial products



Canesta

64x64@30hz
Accuracy 1-2cm

Not accurate enough for face modeling, but good enough for layer extraction.

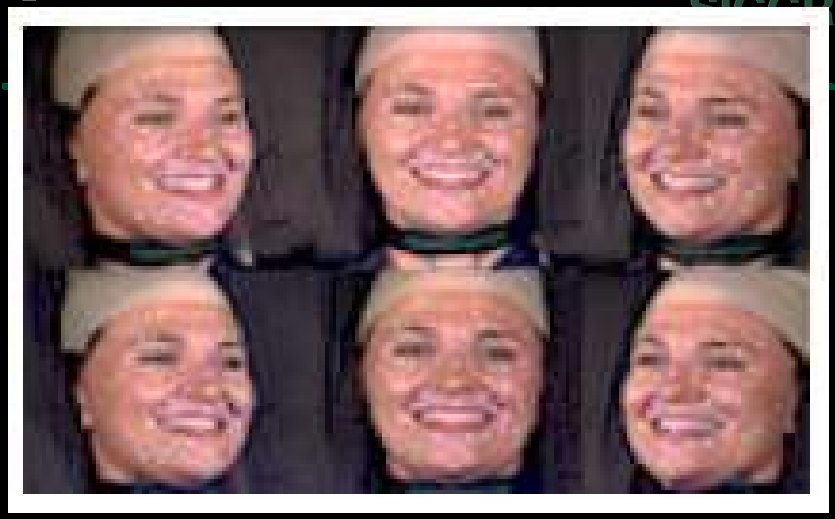


Outline

1. Scanning face models
   - Triangulation methods (created most accurate face models)
   - Non triangulation methods
2. Dense facial motion capture
   - Marker based capture
   - Template fitting for face scans

Marker based approach

182 colored dots on a face



6 cameras videotaping performance

3D dot motion

deforming face model

Dot removal for texturemap

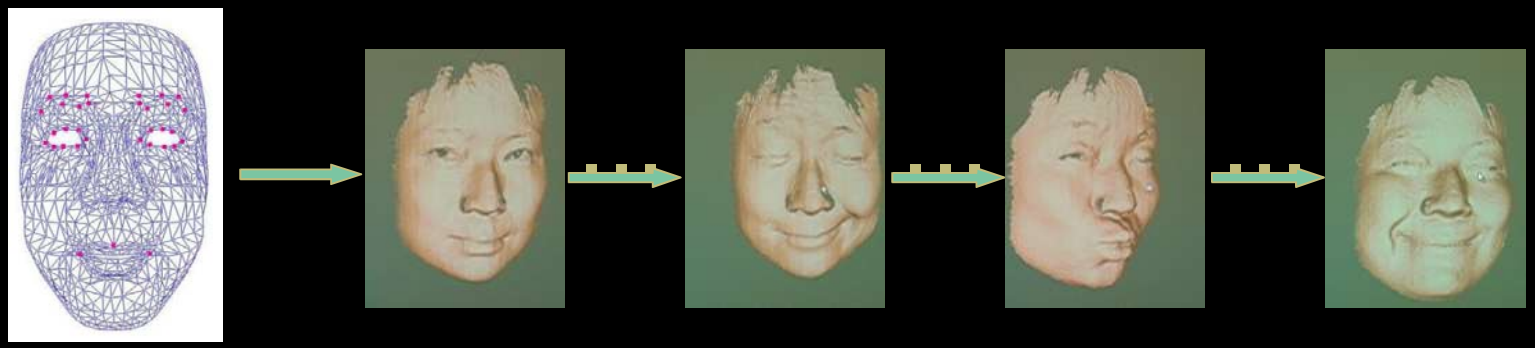
Guenter et al SIGGRAPH 1998

Making faces

+ Realistic appearance
− Limited geometry details
− The overhead of painting faces

Guenter et al SIGGRAPH 1998

High Resolution Acquisition of Dynamic 3-D expression



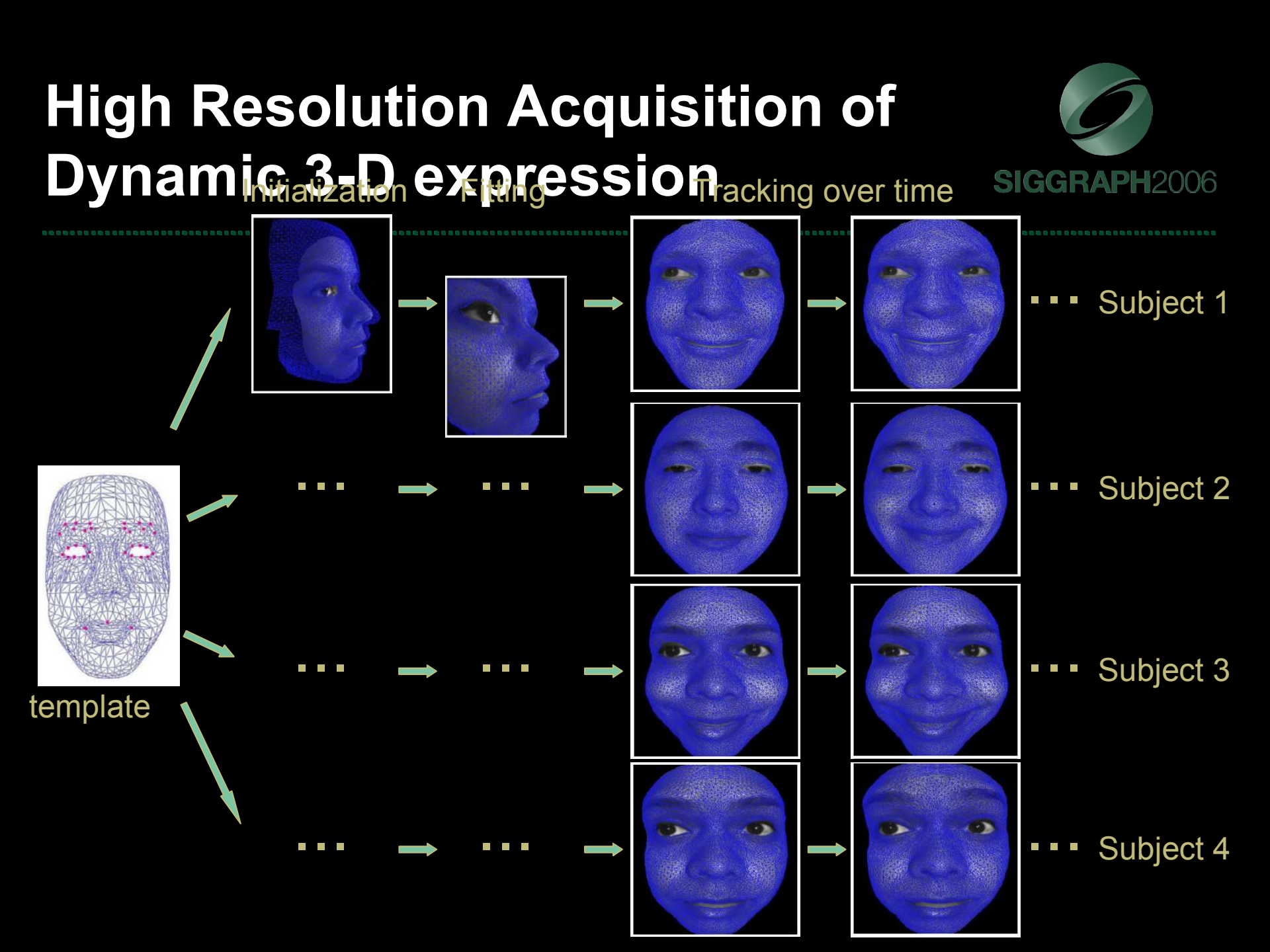
template

Problem: estimating 3D motion between shape measurement

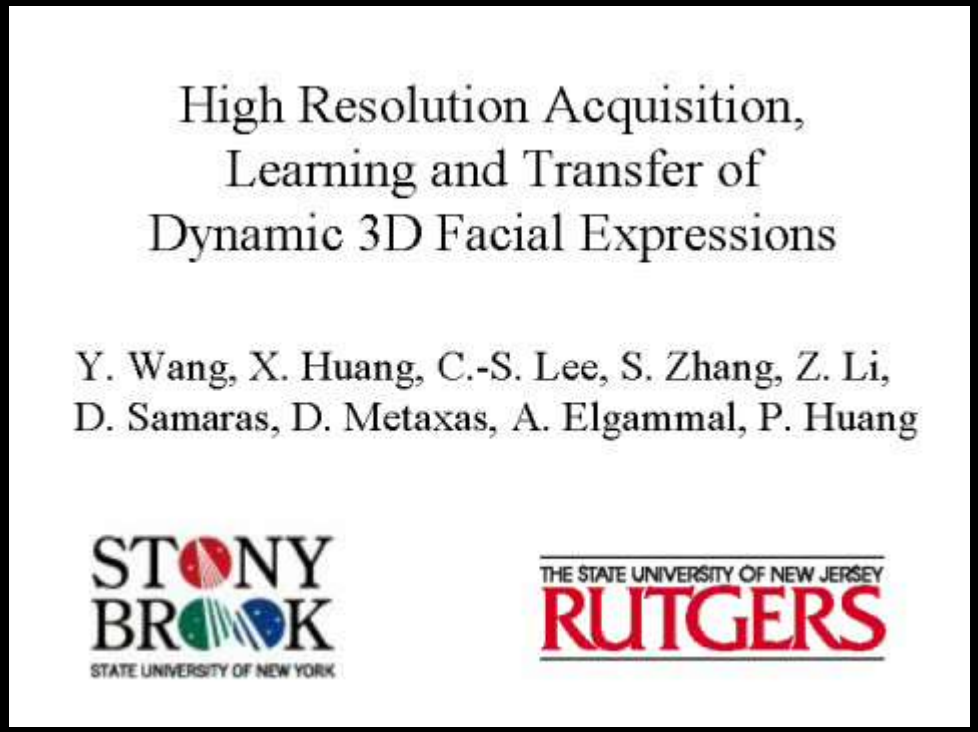
Approach: template fitting

Wang et al Eurographics 2004

High Resolution Acquisition of Dynamic 3-D expression



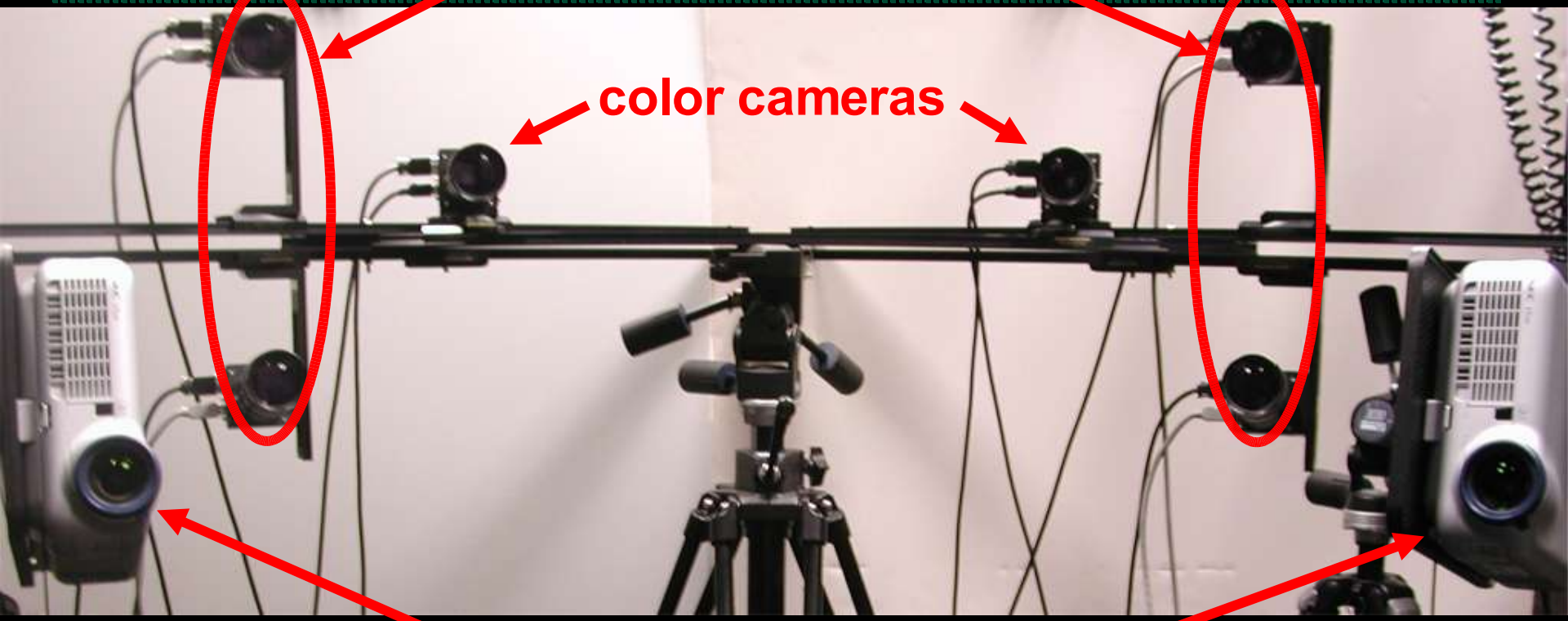
High Resolution Acquisition of Dynamic 3-D expression



+ High resolution motion
− less robust for larger inter-frame deformation

Wang et al Eurographics 2004

Spacetime faces



Face capture rig

Zhang et al SIGGRAPH 2004

Capture process

Input videos (640x480, 60fps)

Black & White
Top Left

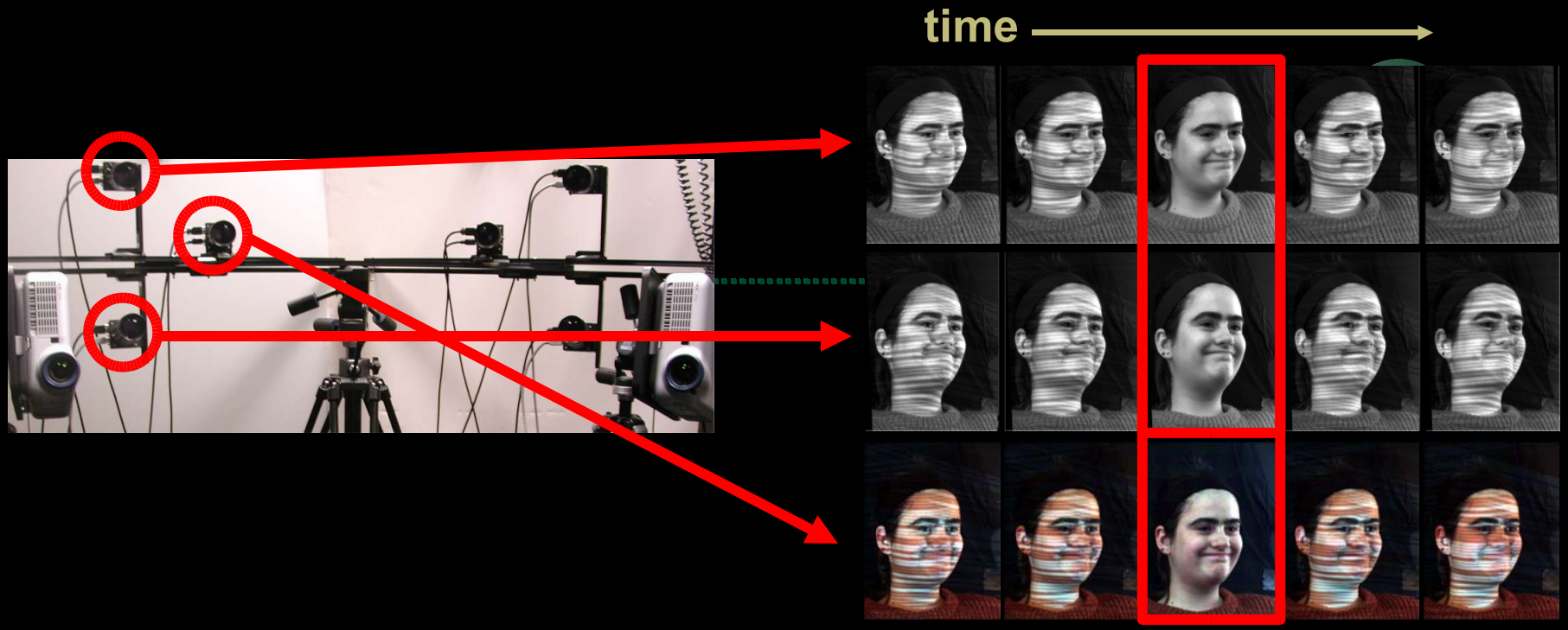
Color Left

Color Right

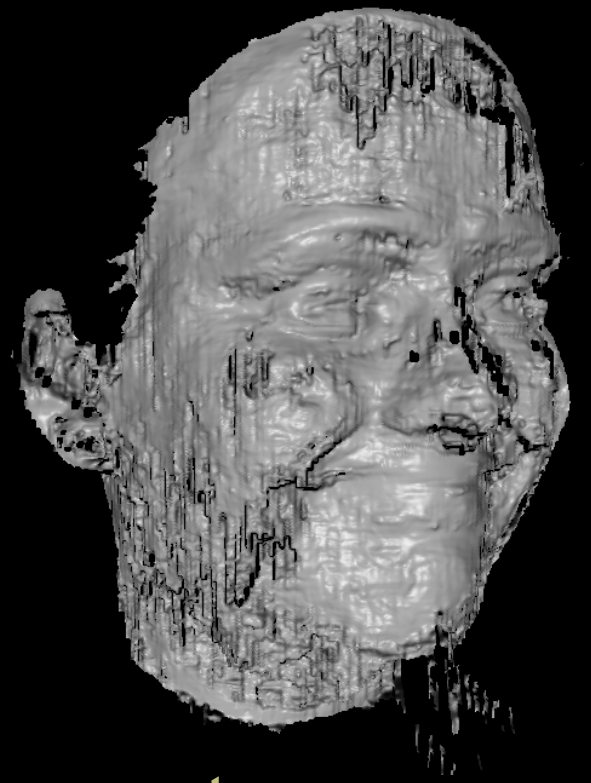
Black & White
Top Right

Black & White
Bottom Left

Black & White
Bottom Right

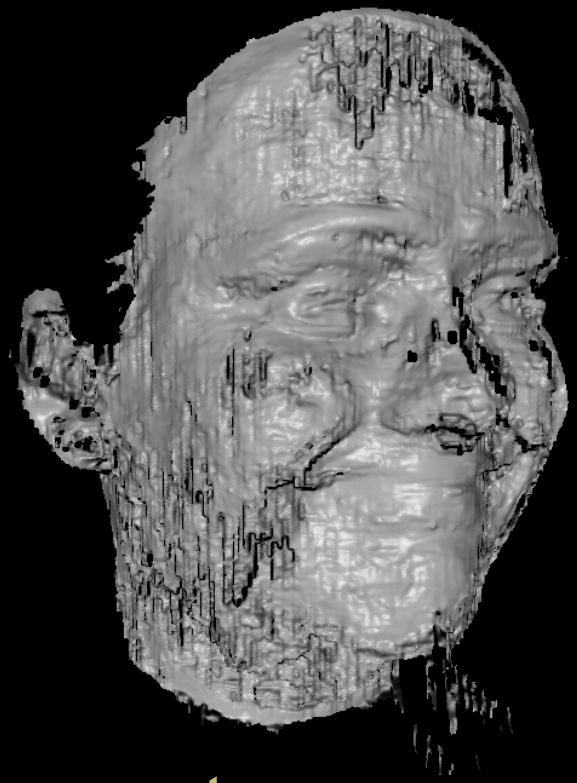

time

time

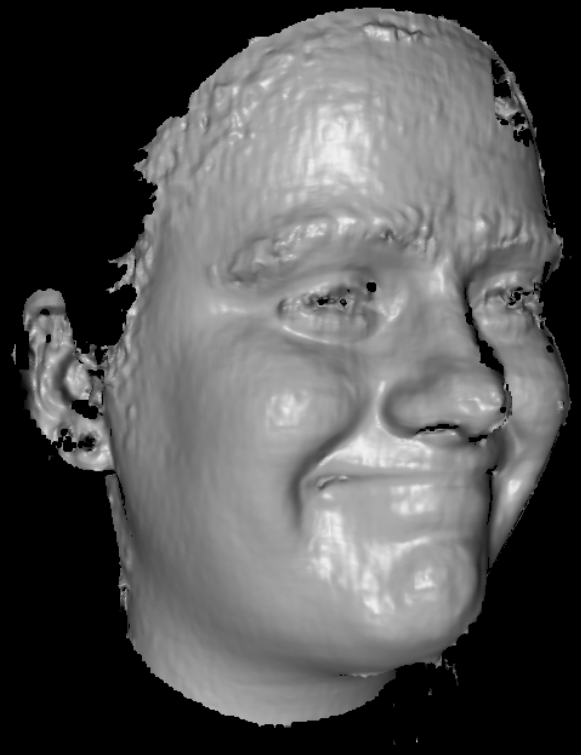


stereo

time

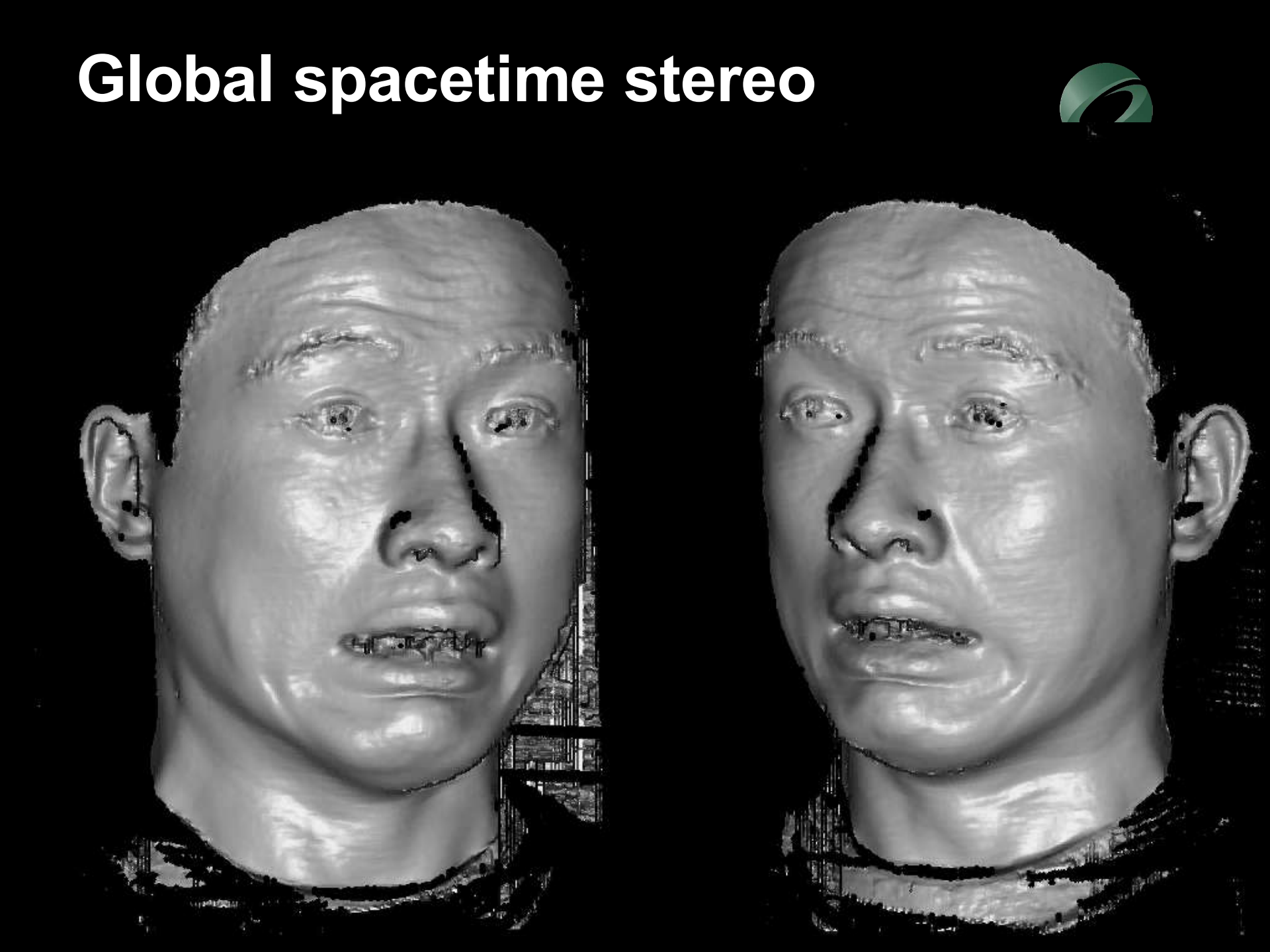


stereo



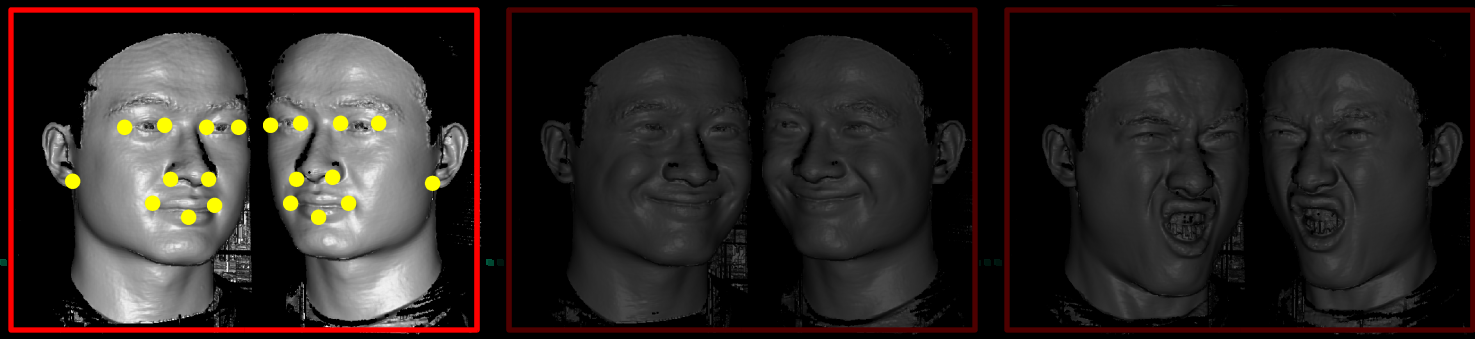
spacetime

Global spacetime stereo

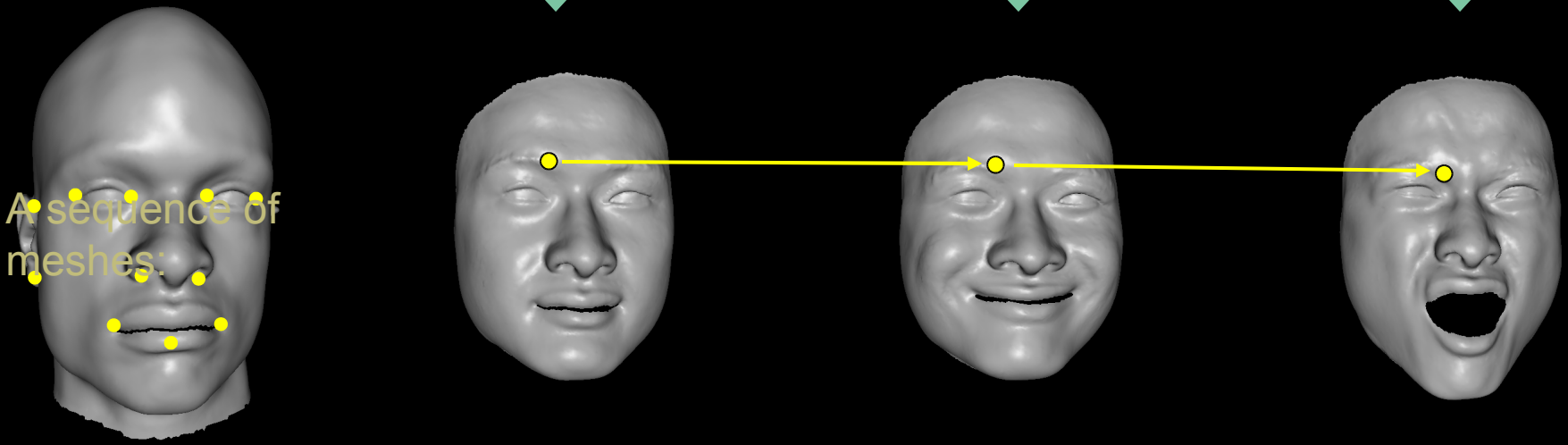


time

A sequence of depth map pairs:



A sequence of meshes:

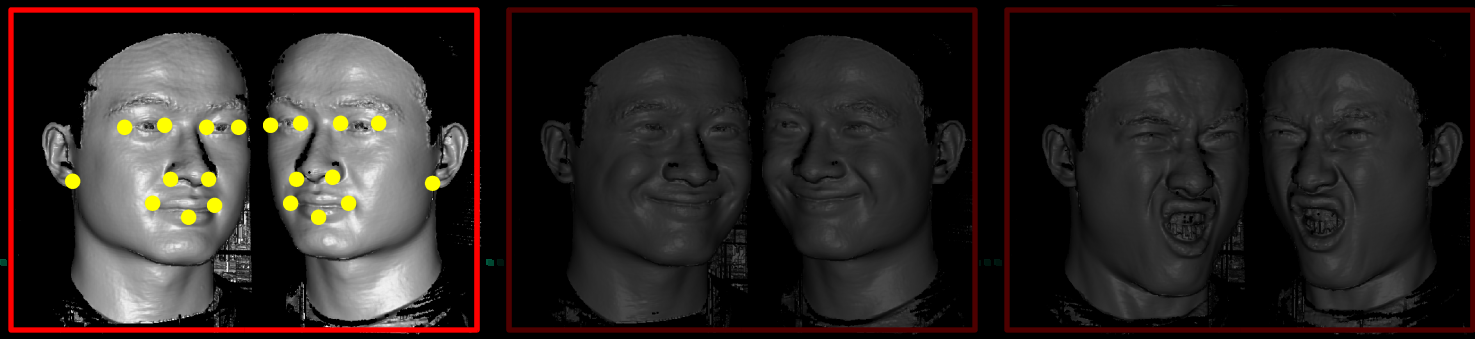


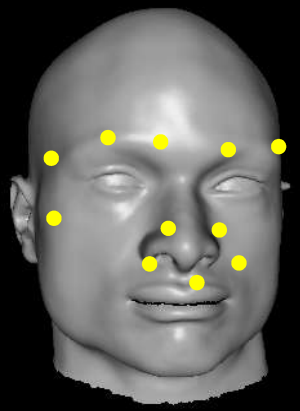
Template mesh

A sequence of color image pairs:

time

A sequence of depth map pairs:



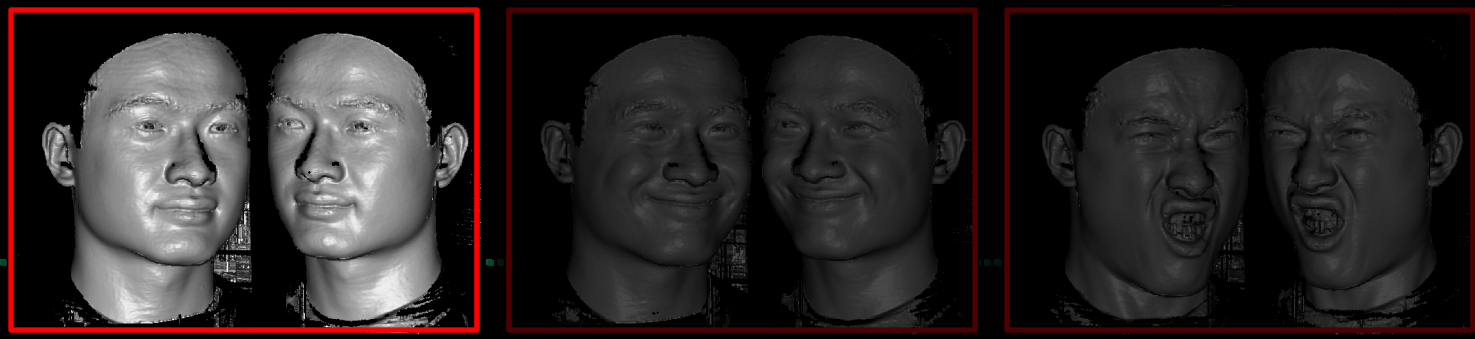


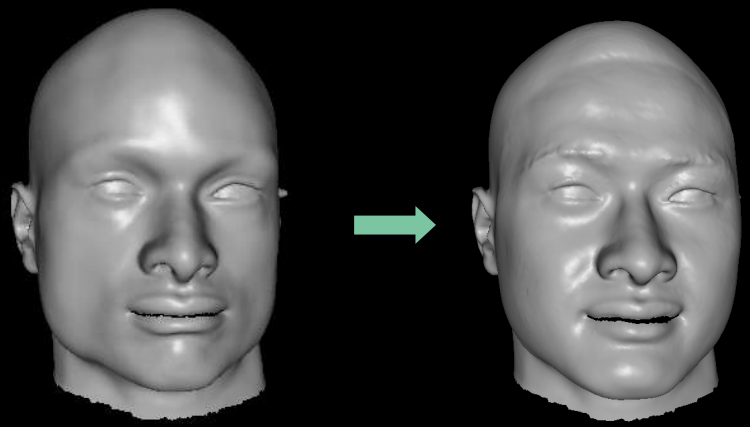
Warped template

A sequence of color image pairs:

time

A sequence of depth map pairs:





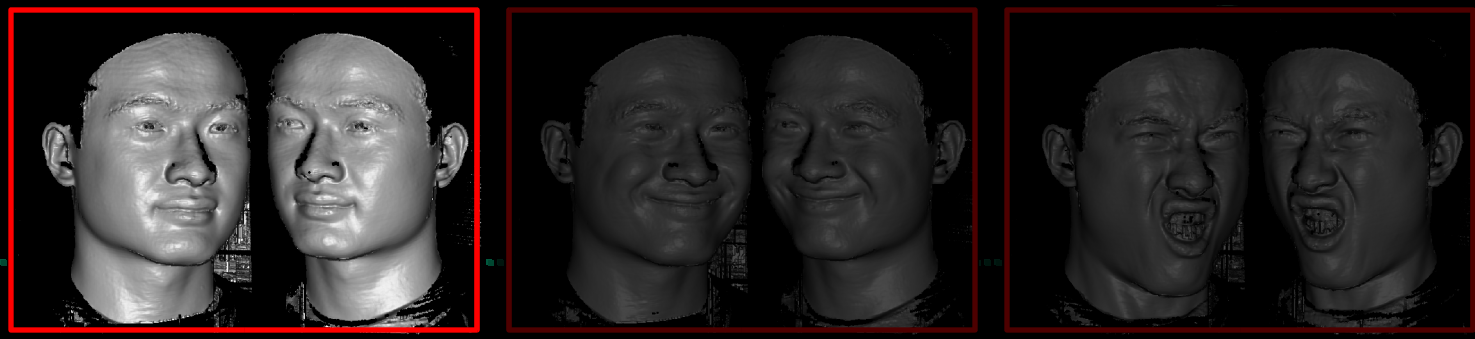
Warped template

Fitted template

A sequence of color image pairs:

time

A sequence of depth map pairs:



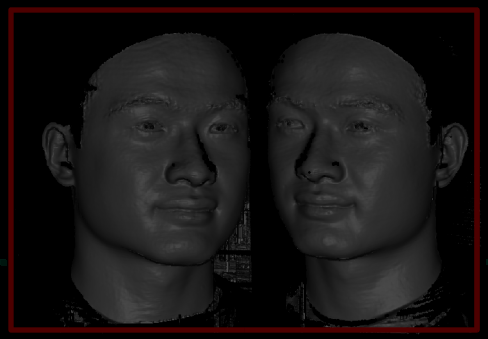
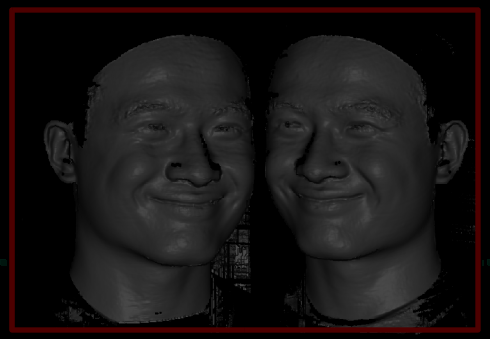
Warped template

Fitted template

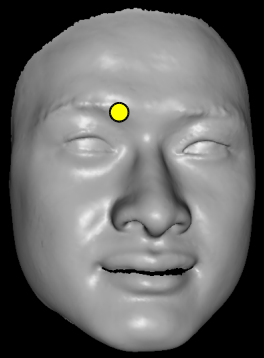
A sequence of color image pairs:

time

A sequence of depth map pairs:


Warped template



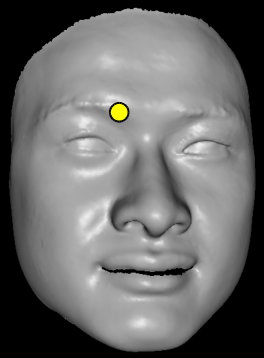
Fitted template

A sequence of color image pairs:



time

A sequence of depth map pairs:

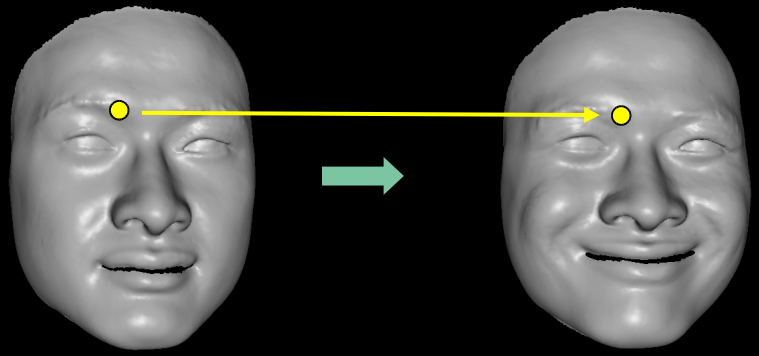


Fitted template

A sequence of color image pairs:

time

A sequence of depth map pairs:

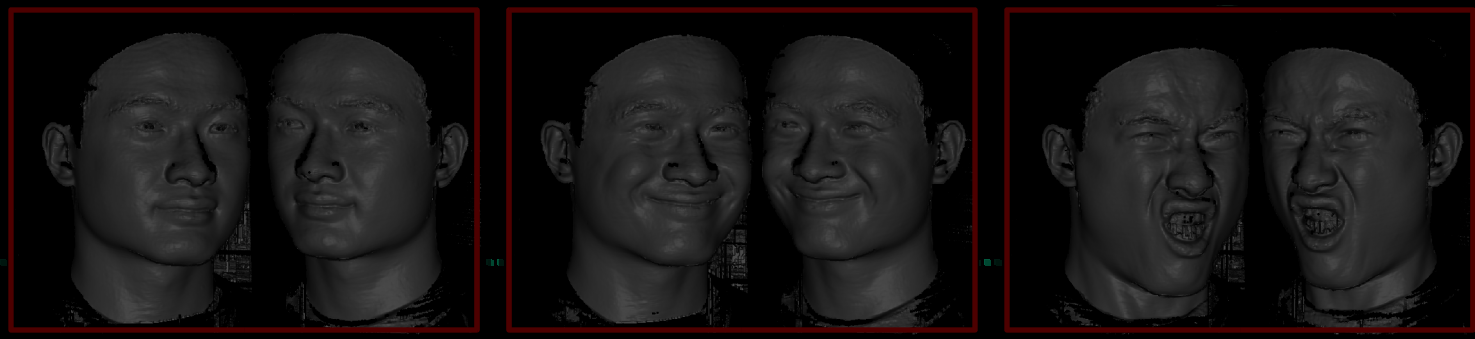


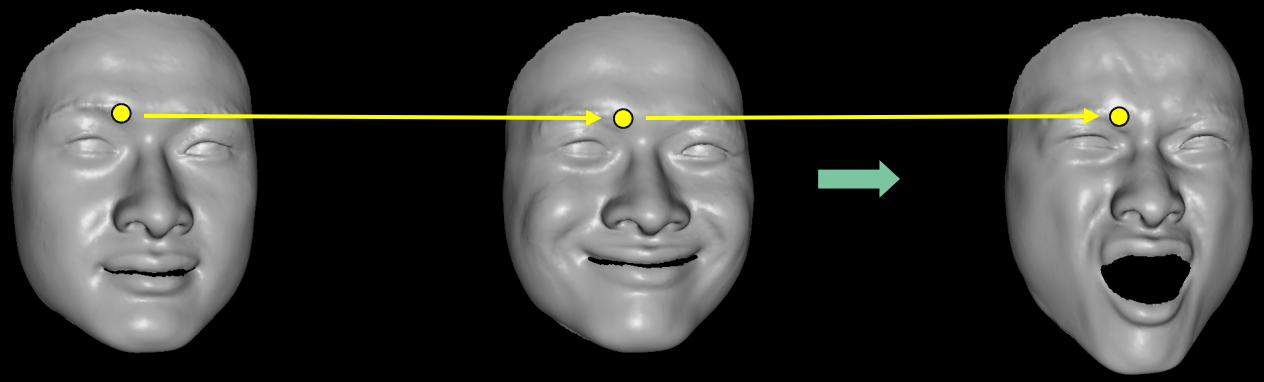
Fitted template

A sequence of color image pairs:

time

A sequence of depth map pairs:



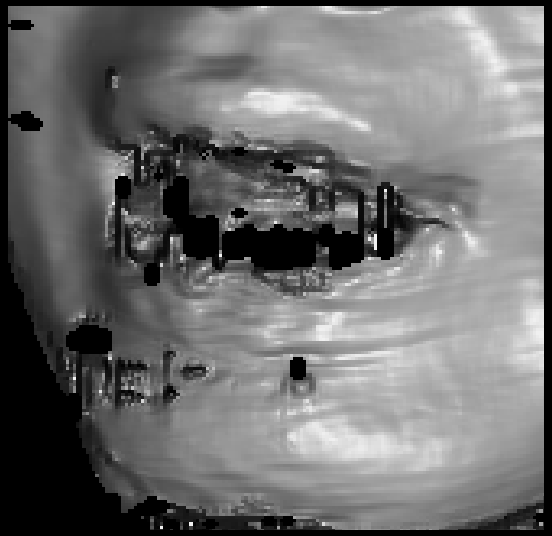


Fitted template

A sequence of color image pairs:

Spacetime faces

+ High resolution motion (~20K vertices)
− not robust for very fast motion



⇒ Fast cameras

⇒ Better skin models for template fitting

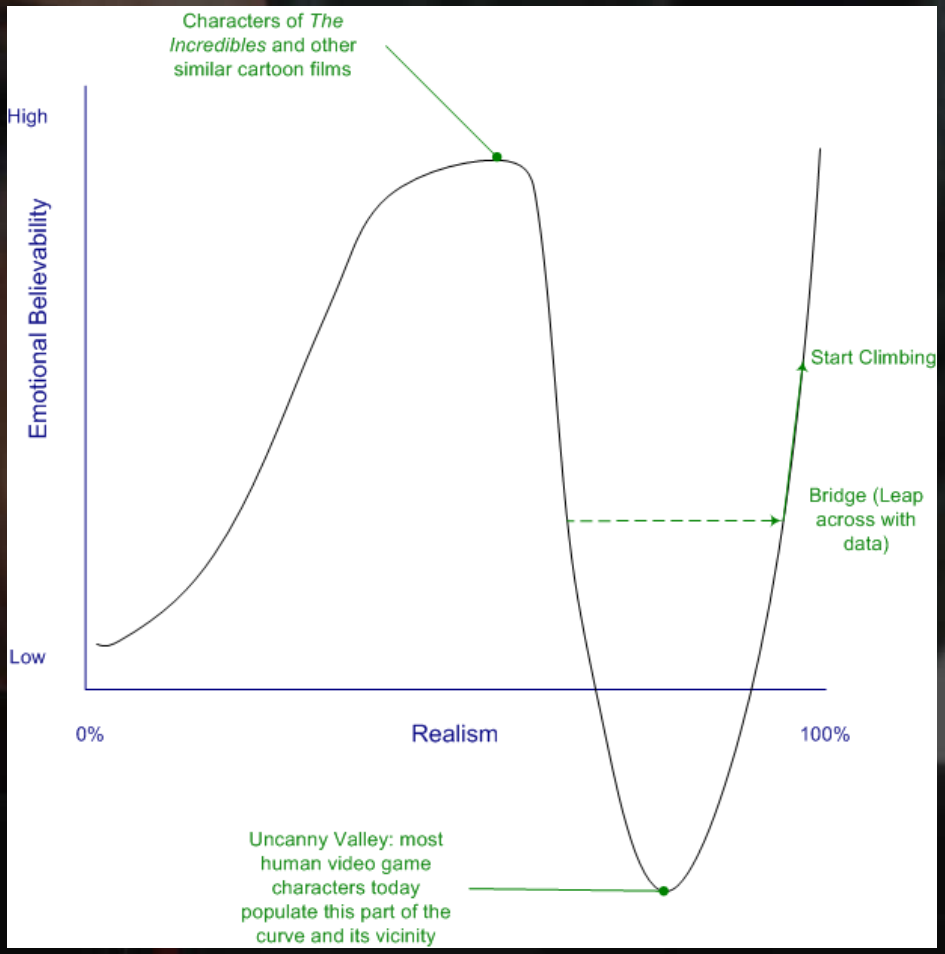
Zhang et al, SIGGRAPH 2004

Playable Universal Capture

George Borshukov Jefferson Montgomery
Witek Werner
James Lau Patrick Mooney Barry Ruff Dave Raposo
Electronic Arts, Inc.

Introduction

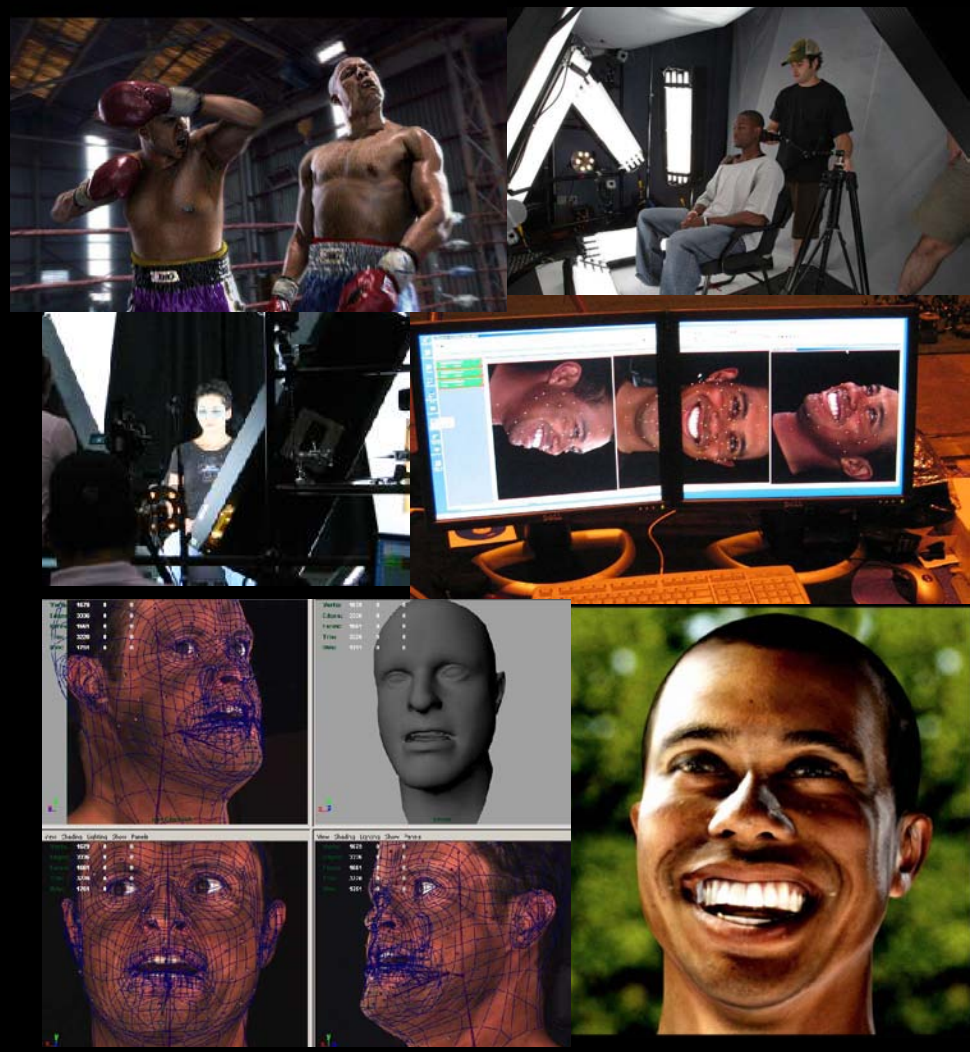
- UCap: High fidelity digital face cloning through accurate capture and reconstruction of both facial motion and texture
- What it gives you today
  - Emotionally Believable Characters??
  - Climb up higher on the right side of the Uncanny Valley?



Universal Capture (UCap)

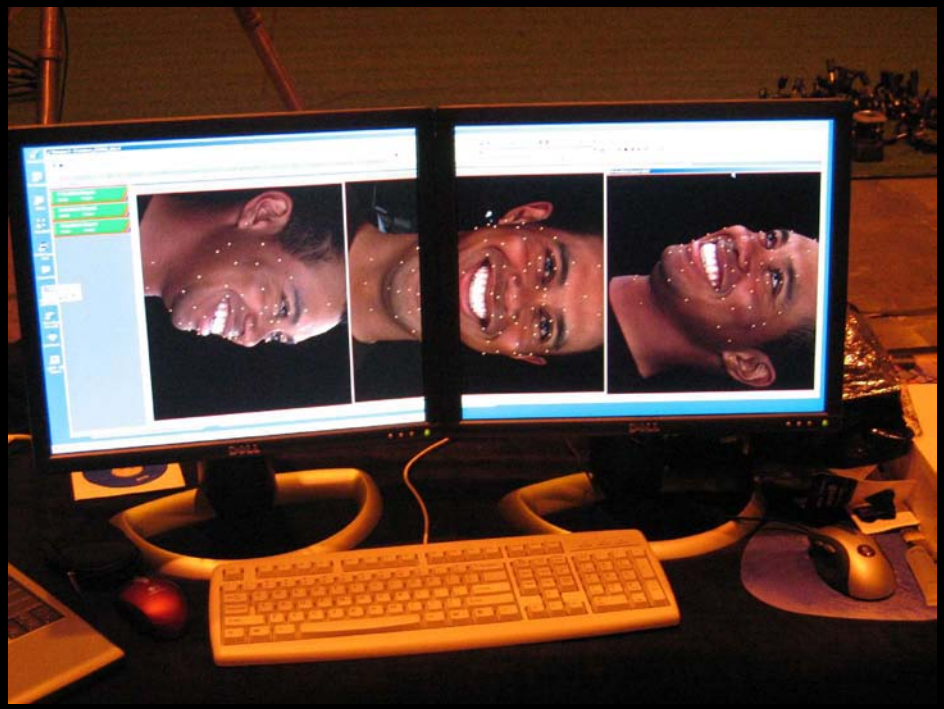
Collaborations

- Fight Night – Nov'04
- Tiger Woods – Jan/Dec'05
- EAJ Fighting Test – June'05
- NBA Live – Dwyane Wade July'05
- MOH:A – Tokyo Game Show – Aug'05
- C&C – Cane, principal character – Aug'05
- EA Mocap - ongoing tech transfer almost complete



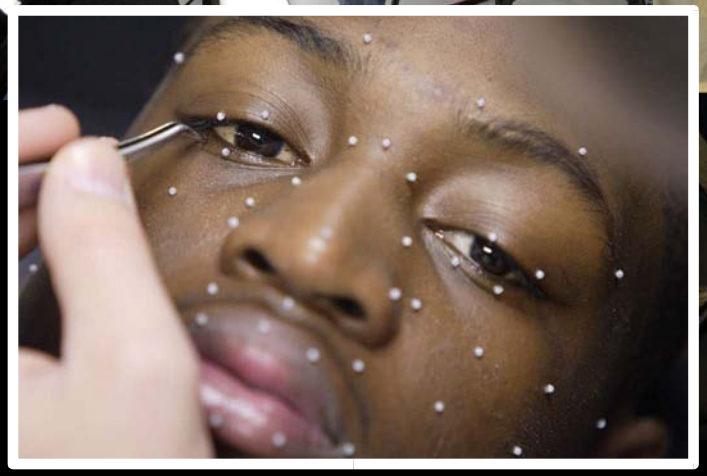
Tiger Woods – Jan/Dec 2005

- Session in Orlando
- Session in LA with 4 other pro golfers



Dwyane Wade – July 2005

- Session in Burnaby

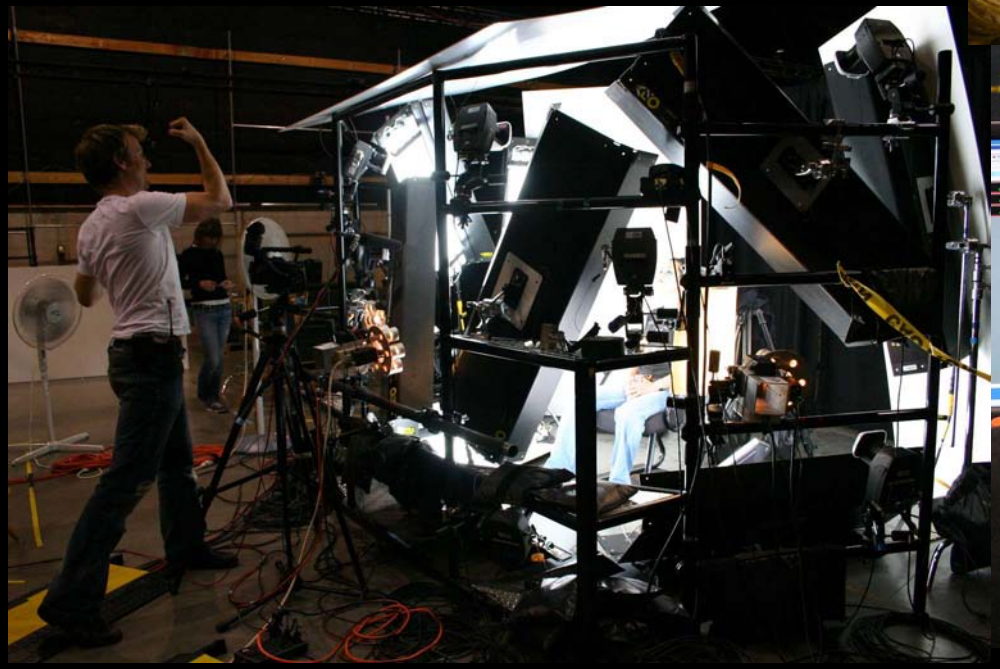


Evolution: UCap -> Playable UCap

- Universal Capture – linear
- Playable Universal Capture – non-linear
  - Capture an Emotion Tree (move tree/motion graph)
  - Apply "move tree" idea used almost universally for body animation in games to the face
  - Robust processing pipeline and tech transfer to EA Mocap
  - Variable basis PCA encoding of geometry & texture for memory efficient real-time playback
    - No facial rig for the runtime: use compressed PCA vertex streams for all facial deformations, which are decompressed at runtime on the SPU (PS3) or GPU (Xbox 360)
  - Identify create smooth transitions, loops
  - Interactive sequencing

EAJ Playable UCap Prototype

- Session in June
- Results presented in November



Real-time Demo Team

- George Borshukov
- Witek Werner
- Jefferson Montgomery
- James Lau
- Barry Ruff
- Dave Raposo
- Patrick Mooney

Real-time Demo Team

- George Borshukov
- Witek Werner
- Jefferson Montgomery
- James Lau
- Barry Ruff
- Dave Raposo
- Patrick Mooney

Can Do

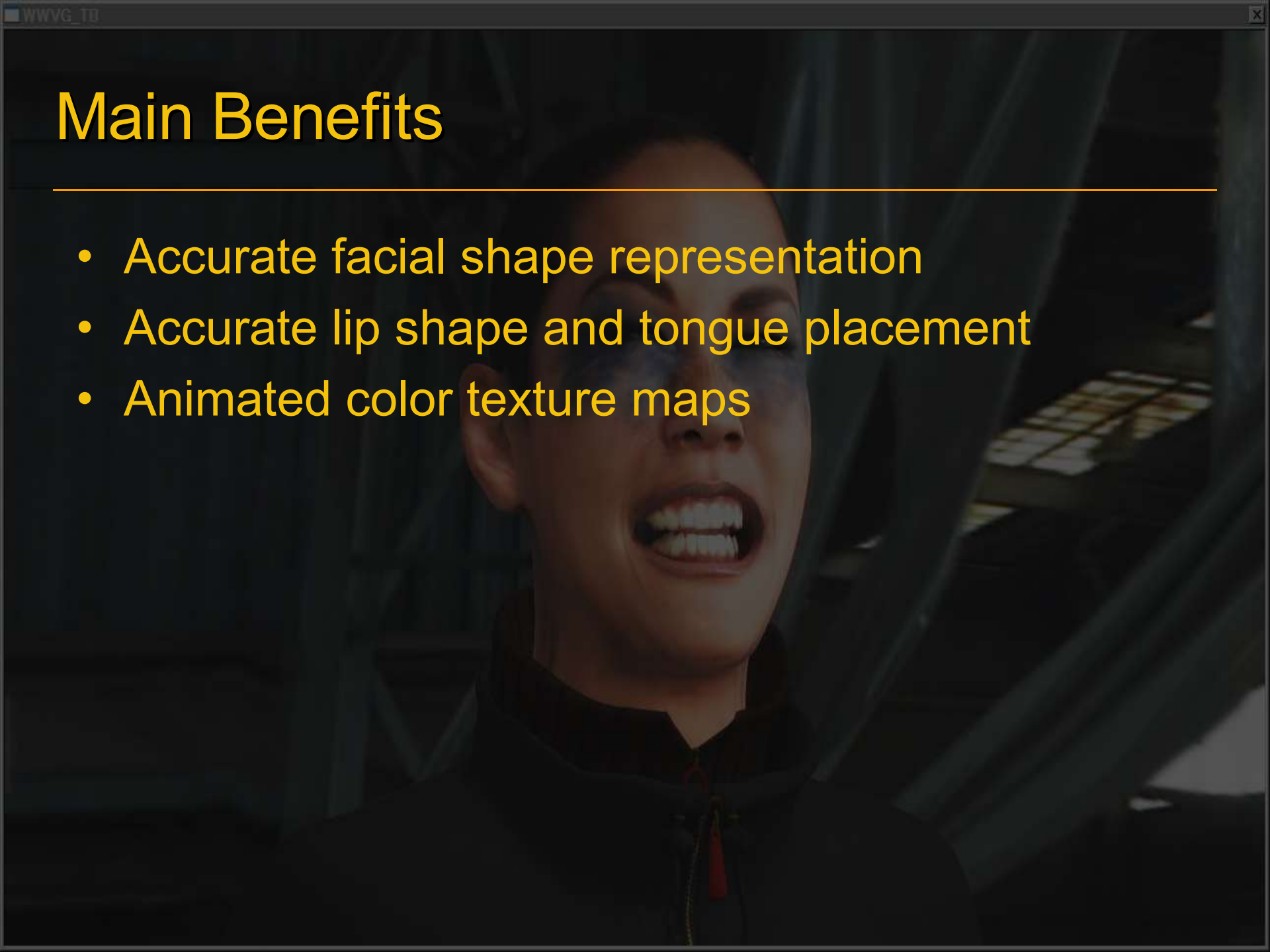
- Free camera
- Lighting - always best to mix at least 30% of the original which contains
  - subsurface scattering
  - ambient occlusion
  - microsopic wrinkling and self shadowing effects
- Stylize
  - Shoot actor in make up
  - Through the shaders

Can Do

- Apply deformations to mesh after data has been processed (example superpunch)
- Layer damage/sweat effects throught textures shader parameter adjustment
- Display results from a very complex facial rig
- Remove overall head movement and apply new overall head movement which can come from body mocap, hand animation, procedural techniques
- Interactively switch from clip to clip at pretty much any point triggering appropriate moves

Cannot Do

- Shoot one person and apply results to another
- Create a new performance or "meaningfully" edit that was not captured
- Change eye gaze direction
  - first problem we want to tackle in the next stage



Main Benefits

- Accurate facial shape representation
- Accurate lip shape and tongue placement
- Animated color texture maps

Playable UCap Assets

Head Geo at runtime:

- facial lifecast XYZ scanned at 250 microns (.25 mm)
- 3500 Quads - > 10,000 Quads after 1 level of subdivision
- 20,000 triangles

Facial Rig (used only during processing):

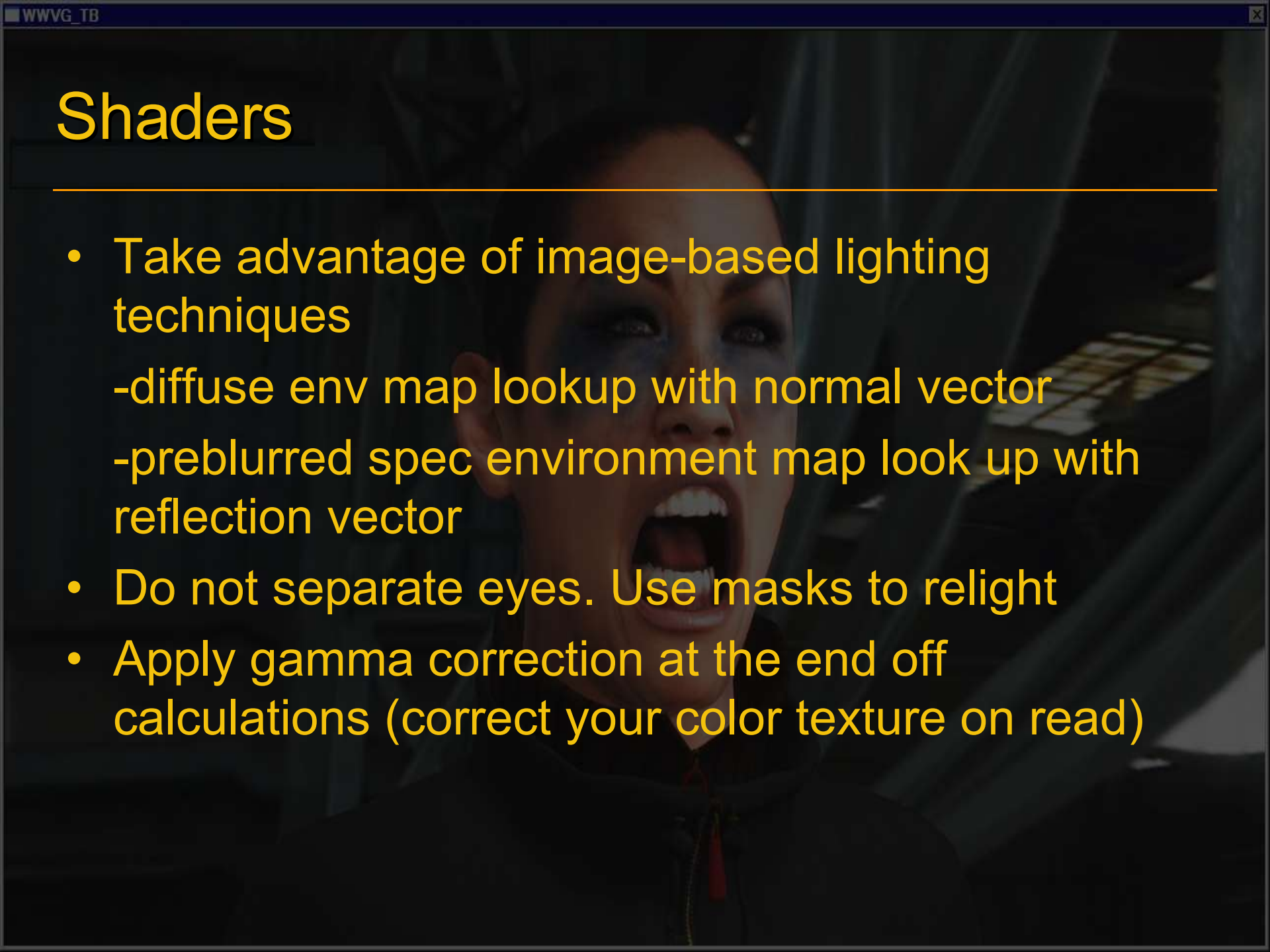
- translation bone-based (slightly modified Mocap facial rig)
- approx 70 bones, 1 per mocap marker
- 8 lip/mouth bones for hand-tracking the lips
- weighting is key

Textures:

- static texture for ears and back of head
- animating textures for face and neck – 1 texture per frame

Maps:

- normal map (static)
- specular map (static)
- eye/lip material mask to isolate for tweaking



Shaders

- Take advantage of image-based lighting techniques

  -diffuse env map lookup with normal vector

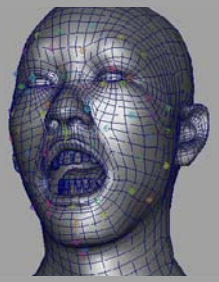
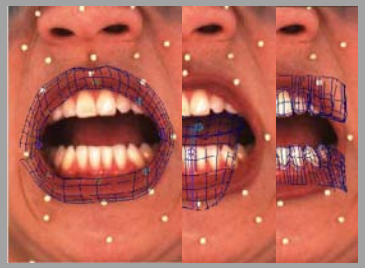
  -preblurred spec environment map look up with reflection vector
- Do not separate eyes. Use masks to relight
- Apply gamma correction at the end off calculations (correct your color texture on read)

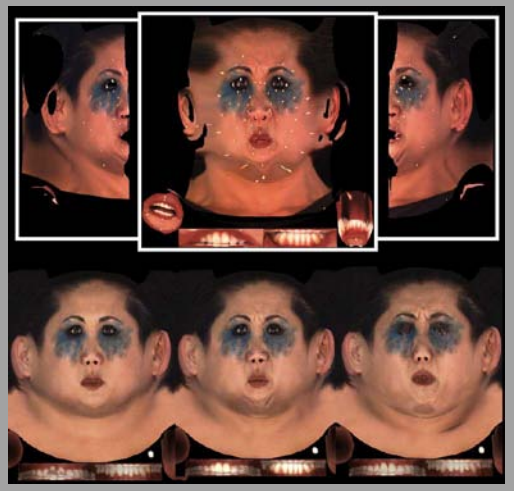
Advanced Shading Examples

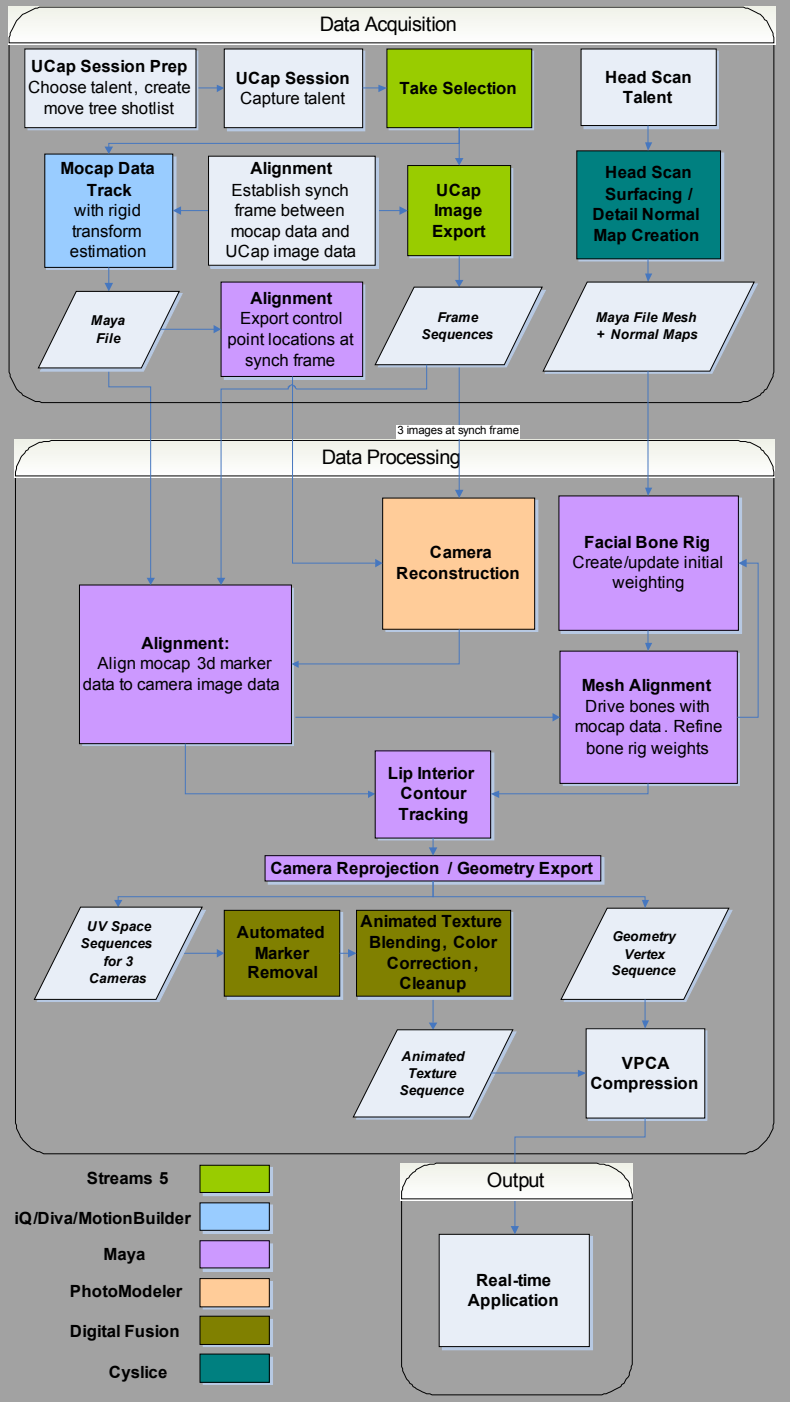
UCap Pipeline







Designing the Move Tree Shot List

Goal

- interactive ucap demo
- with full dynamic range of motions
- achieving believability and responsiveness

Tools

- DVD reference
- contrast (typical fighting facial expressions vs. openings & inserts)
- layered approach /intensities/
- classification
  - ATTACK
  - REACT
  - DEFENSE
  - INSERT
  - OPENING
  - BREATHING IDLES

A C T I O N S

NAME / INTENSITY LEVEL	SH#	FR#	FR#	DURATION	PRIORITY	NOTES	COMPLETION	COMPLETION
ATTACK_ANGRY_intenL1	0037_1	152	193	41	M		G	G
ATTACK_ANGRY_intenL2	0037_1	750	798	48	H		G	G
ATTACK_ANGRY_intenL3	0037_1	257	300	43	M		G	G
ATTACK_ANGRY_intenL4	0037_1	661	709	48	M		G	G
ATTACK_COCKY_intenL1	0038_1	203	257	54	M		W	W/G
ATTACK_COCKY_intenL2	0038_1	82	149	67	H		W	W/G
ATTACK_AGGRESIVE_intenL1	0039_1	78	110	32	L		W	W/G
ATTACK_AGGRESIVE_intenL2	0039_1	167	199	32	M		W	W/G
ATTACK_AGGRESIVE_intenL3	0039_1	527	560	33	M		W	W/G
ATTACK_AGGRESIVE_intenL4	0039_1	618	653	35	H		W	W/G
ATTACK_TIRED_intenL1	0041_1	722	744	22	L		W	W/G
ATTACK_TIRED_intenL2	0041_1	92	135	43	M		W	W/G
ATTACK_TIRED_intenL3	0041_1	197	236	39	M		W	W/G
ATTACK_TIRED_intenL4	0041_1	640	690	50	H		W	W/G
ATTACK_TIRED_intenL5	0041_1	816	868	52	M		W	W/G
ATTACK_STRENUOUS_intenL1	0042_1	66	131	65	H		P	W/G
ATTACK_STRENUOUS_intenL2	0042_1	171	244	73	M		P	W/G
ATTACK_STRENUOUS_intenL3	0042_1	522	580	58	L		P	W/G
ATTACK_STRENUOUS_intenL4	0042_1	616	718	102	M		P	W/G
ATTACK_KILLER_intenL1	0043_1	446	485	39	L		G	G
ATTACK_KILLER_intenL2	0043_1	188	234	46	H		G	G
ATTACK_KILLER_intenL3	0043_1	90	124	34	M		G	G
ATTACK_KILLER_intenL4	0043_1	344	430	86	L		G	G
REACT_FACE_R	0044_2	580	627	47	H		D	G
REACT_FACE_L	0044_2	657	704	47	H		D	G
REACT_TOES	0044_2	740	776	36	H		D	G
REACT_STOMACH_intenL1	0044_2	809	848	39	L		D	W/G
REACT_STOMACH_intenL2	0044_2	961	1025	64	M		D	G
REACT_STOMACH_intenL3	0044_2	897	942	45	H		D	G
REACT_AVOIDANCE_L	0048_3	37	64	27	H		P	G
REACT_AVOIDANCE_R	0048_3	98	127	29	H		P	G
REACT_AVOIDANCE_UP	0048_3	151	180	29	H		P	G
REACT_MORTAL	0049_1	91	216	125	H		G	G
DEFENSE_COCKY	0050_3	260	302	42	H		G	G
DEFENSE_DESPERATE_intenL1	0051_1	541	586	45	H		D	G
DEFENSE_DESPERATE_intenL2	0051_1	626	673	47	L		D	G
DEFENSE_HORRIFIED_intenL1	0052_2	64	97	33	L		P	W/G
DEFENSE_HORRIFIED_intenL2	0052_2	137	172	35	M		P	W/G
DEFENSE_HORRIFIED_intenL3	0052_2	217	251	34	H		P	W/G
INSERT_FRUSTRATION	0057_1	217	269	52	H		P	W/G
INSERT_PANIC	0058_2	100	174	74	H		W	W/G
INSERT_INTIMIDATION	0059_1	542	666	124	H		W	W/G
OPENING_PEACEFUL_SPIRITUAL	0060_1	295	484	189	H		P	W/G
OPENING_PHYSICAL_STRENGH	0061_2	355	448	93	H		P	W/G
OPENING_DISRESPECTFUL_AROGANT	0063_1	497	601	104	H		W	W/G
EYE_BLINK	0060_1	272	294	22	H		P	W/G

I D L E L O O P S

NAME / INTENSITY LEVEL	SH#	FR#	FR#	DURATION	PRIORITY	NOTES	COMPLITION

Captured Facial Move Tree

Post processing

Reference creation

- video analysis (roamer camera)
- comparison against wanted moves
- animation reference table creation (group moves)
- sub range identification (Adobe Premiere editing)
- final take list (clip ranges)
- request to process takes (motion, textures)

Leanne Processing Stats

- Length of capture session - 3 hours
- Total data captured - 197 GB
- Total # of moves captured - 33
- # of moves selected for use – 21

(ultimately split into 55 separate clips and grouped for state flow)

- Total minutes of footage captured - ~17 min
- Minutes of footage selected for use - ~ 2 min
- Average # of frames/move - 65 frames
- Median # of frames/move - 47 frames
- Shortest move - 27 frames, Longest move - 220 frames
- # of trackers – 3, approximate processing time - 1.5 months

Video (includes speech processing)

- UCAP_siggraph2006_852x480_H264_stereo_EA_Watermarked.mov

NFS In-Game Prototype

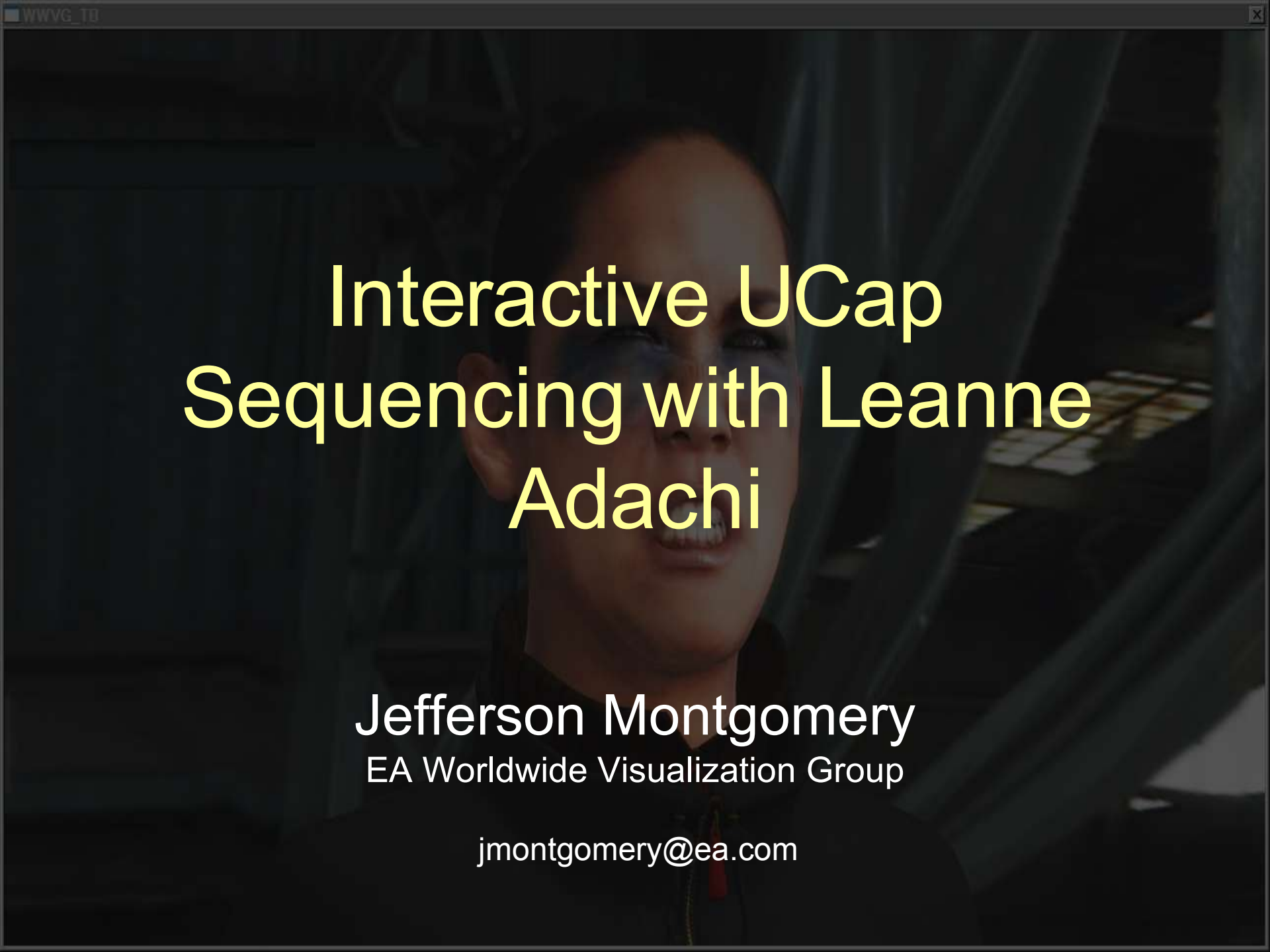
- Edward Douglas
- Collin O’Conner

Engineering Overview

- Jefferson Montgomery

Special Thanks

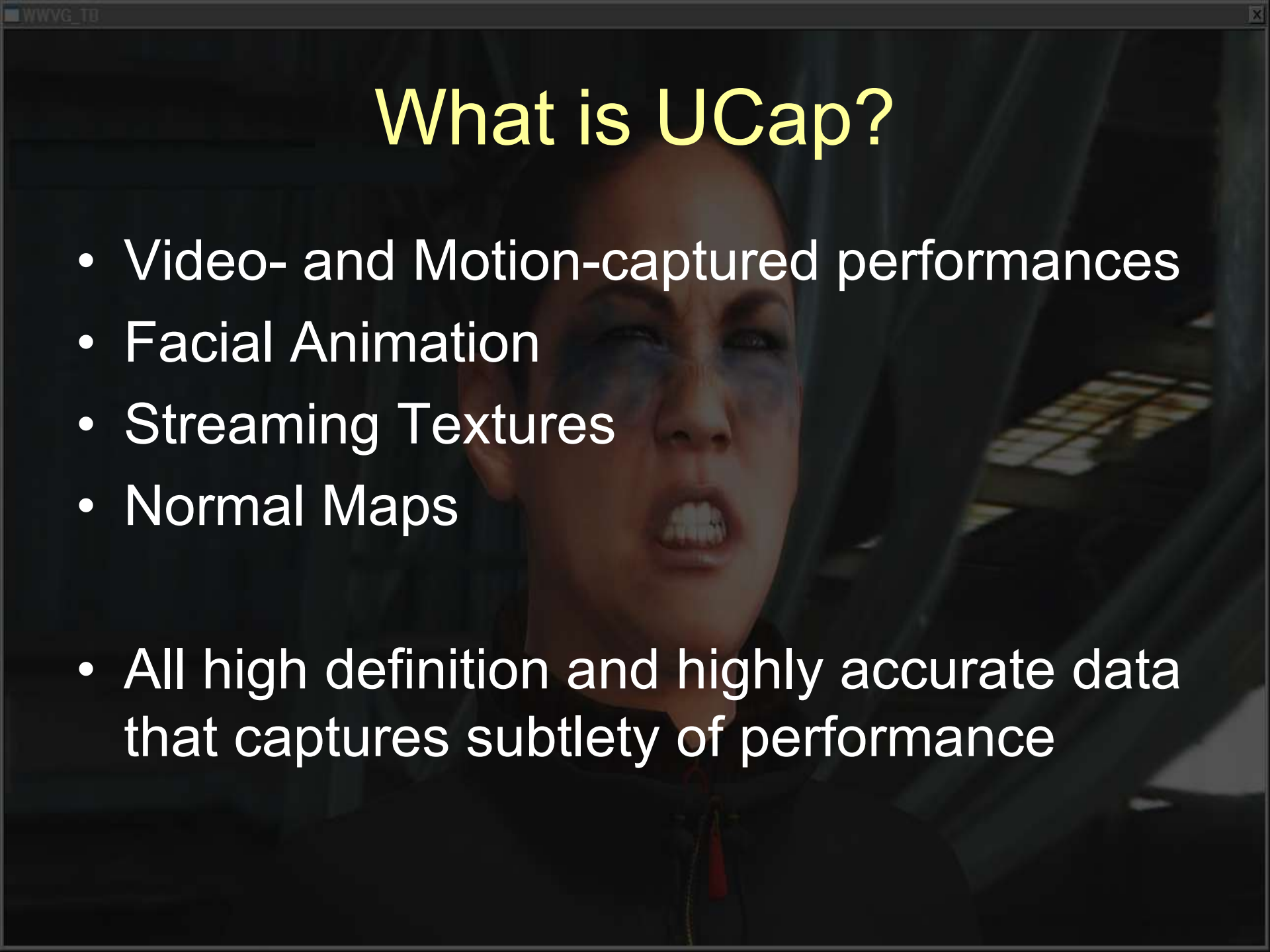
- John Hable, Hakan Kihlstrom, Jean-Luc Dupra, Paul Thuriot, Kevin Noone, James Grieve, Paul Lalonde
- Stefan Van Niekerk, Ben Guthrie, Doug Griffin and the rest of the EA Mocap crew
- Neil Eskuri, Sean Smillie, Collin O'Conner, Edward Douglas and the rest of the innovative NFS team
- Jeff O'Connell, Brian Wideen, Glenn Entis



Interactive UCap Sequencing with Leanne Adachi

Jefferson Montgomery
EA Worldwide Visualization Group

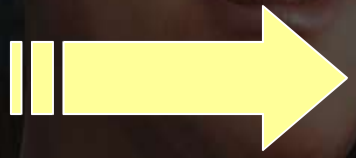
jmontgomery@ea.com


What is UCap?
• Video- and Motion-captured performances
• Facial Animation
• Streaming Textures
• Normal Maps
• All high definition and highly accurate data that captures subtlety of performance

Runtime Challenges

- Enormous amount of source data (GBytes per performance)

- Storage?
- GPU delivery?



Data Compression!

Data Compression

- VP6, MPG, etc.
  - Difficult random access
  - Artifacts can be abrasive and difficult to control
  - Complicated decode algorithms ★★★

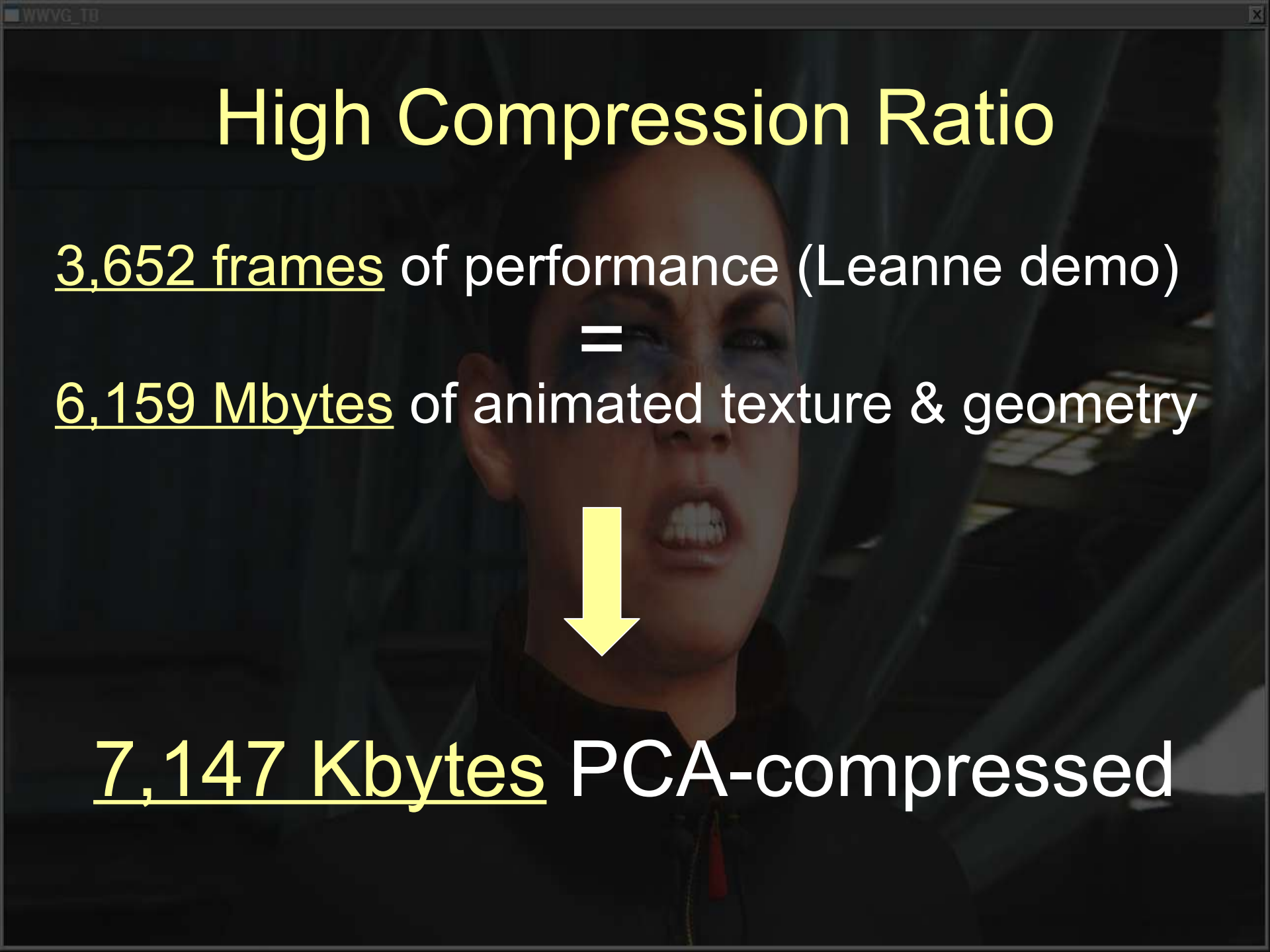
Principle Component Analysis

Advantages

- Forgiving defects (blurring)
- Potential for very high compression
- Automatic pipeline
- Simple reconstruction; well suited to vector processors

Forgiving Defects

Captured PCA


High Compression Ratio
3,652 frames of performance (Leanne demo)
=
6,159 Mbytes of animated texture & geometry
7,147 Kbytes PCA-compressed

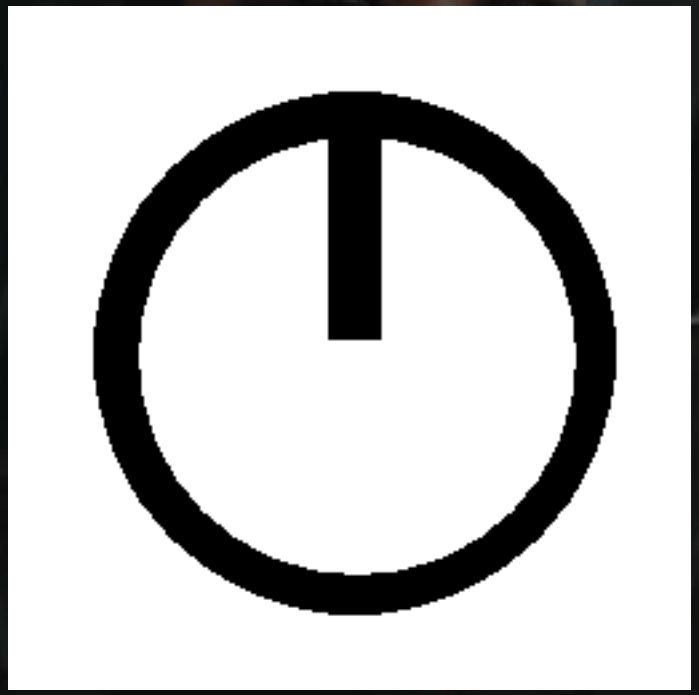
Simple Reconstruction

```
float PcaDecompress(
        float4 weights[4],
        float4 components[4])
{
    return      dot(weights[0], components[0]) +
                dot(weights[1], components[1]) +
                dot(weights[2], components[2]) +
                dot(weights[3], components[3]);
}
```

Per colour or vertex attribute

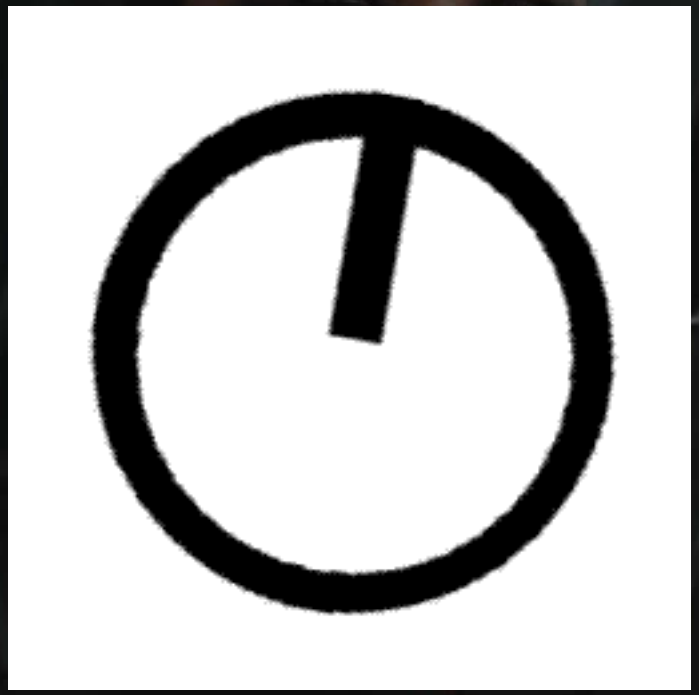
How it works…

- Toy example:



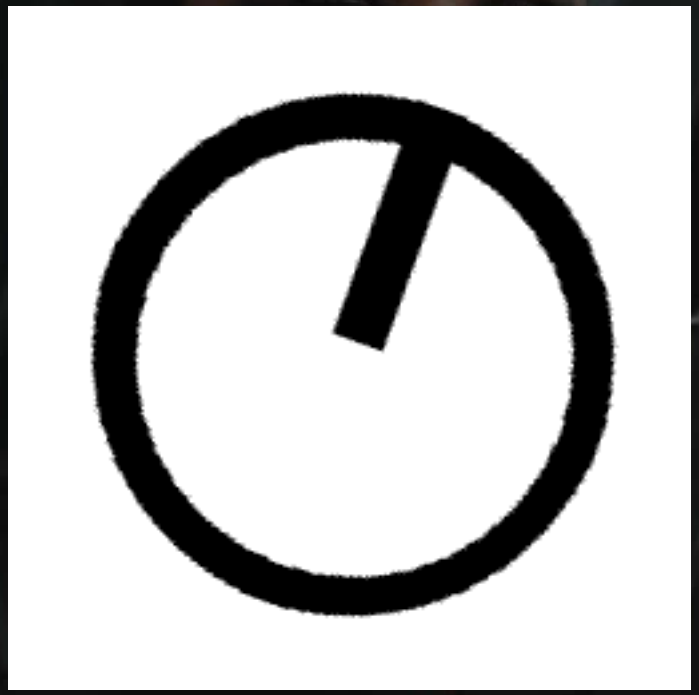
How it works…

- Toy example:



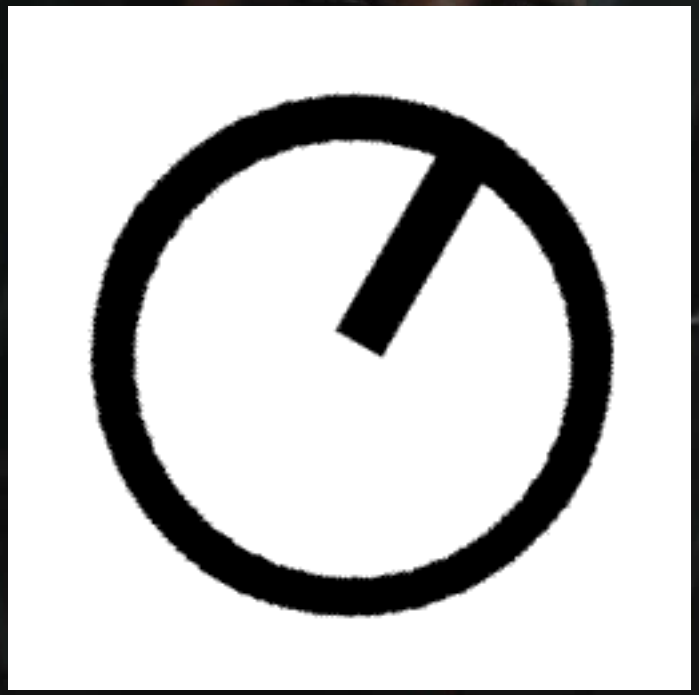
How it works…

- Toy example:



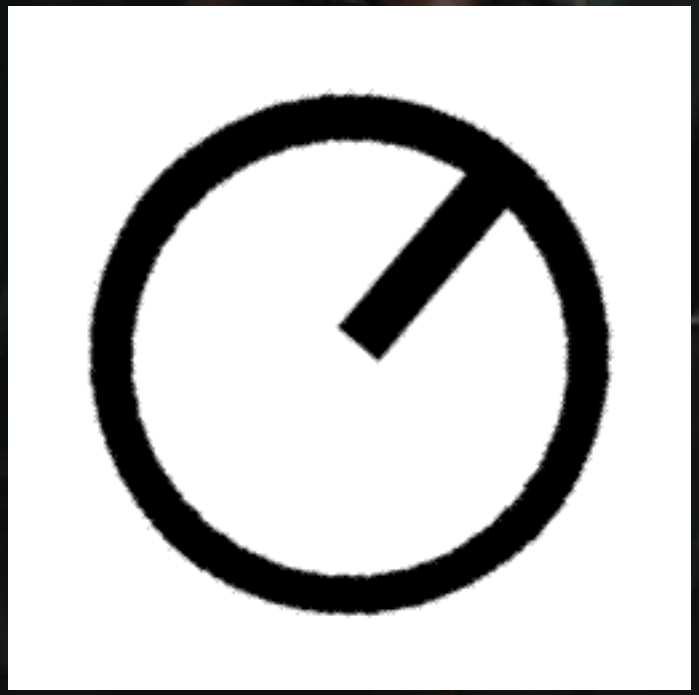
How it works…

- Toy example:



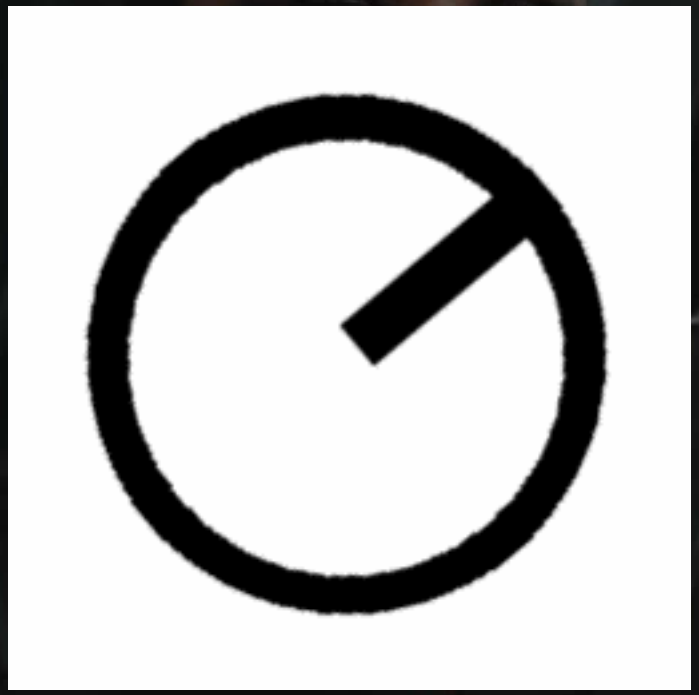
How it works…

- Toy example:



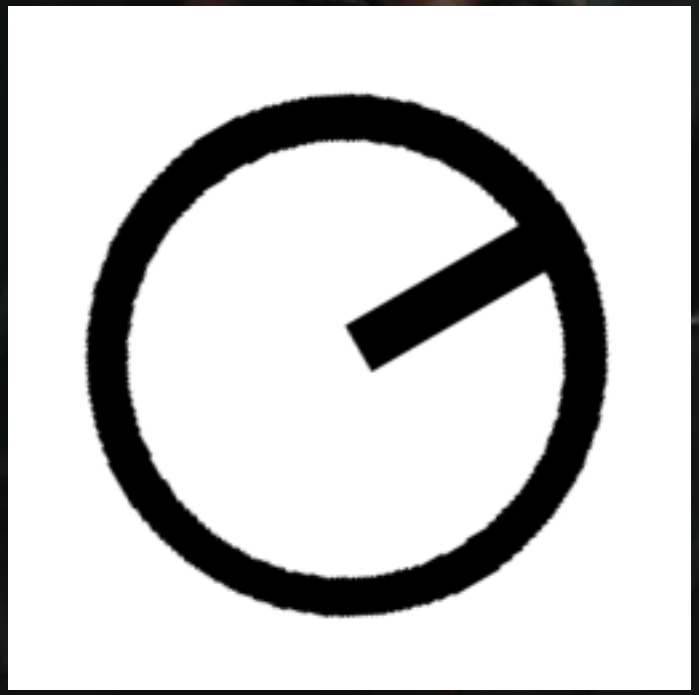
How it works…

- Toy example:



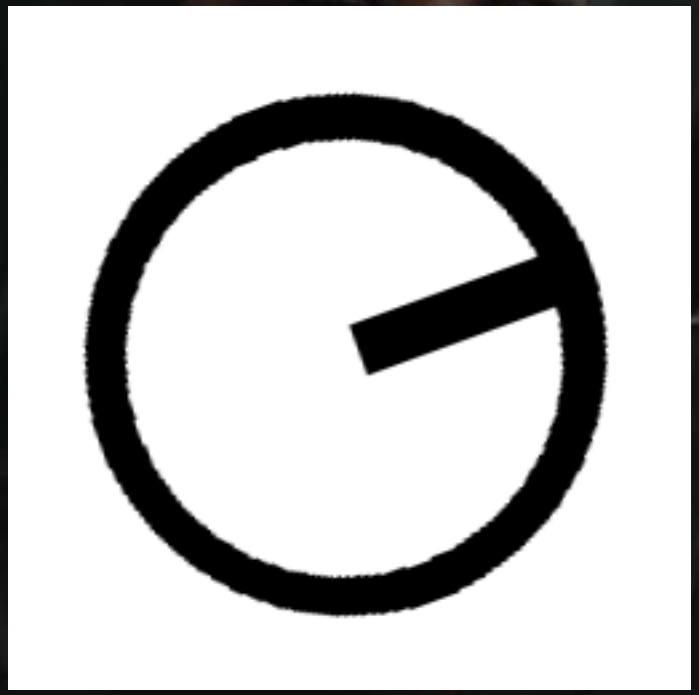
How it works…

- Toy example:



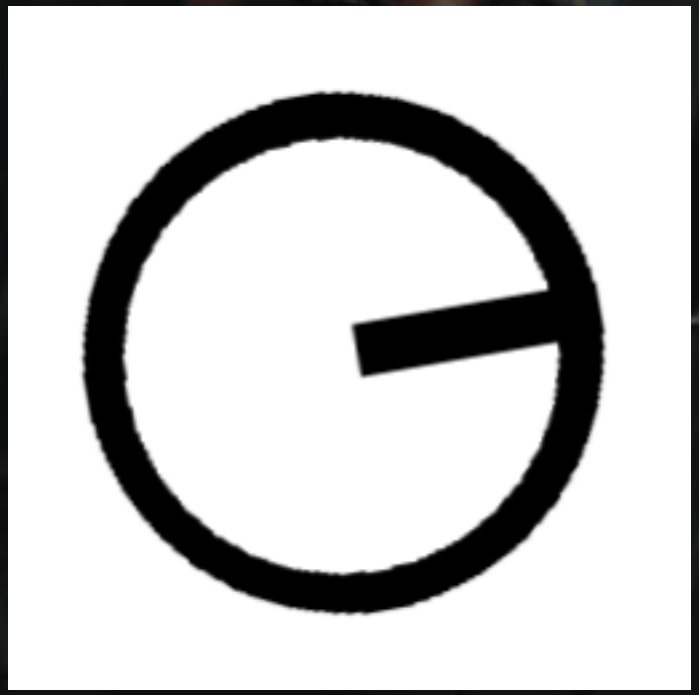
How it works…

- Toy example:



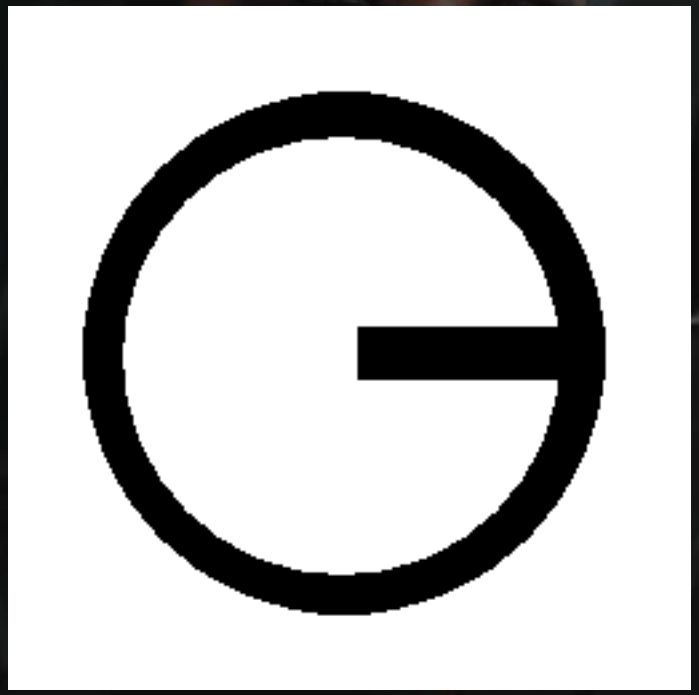
How it works…

- Toy example:



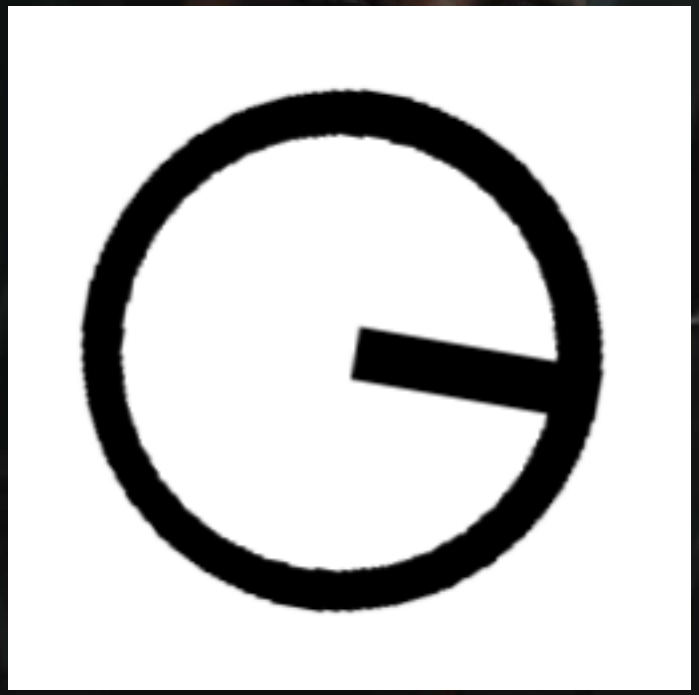
How it works…

- Toy example:



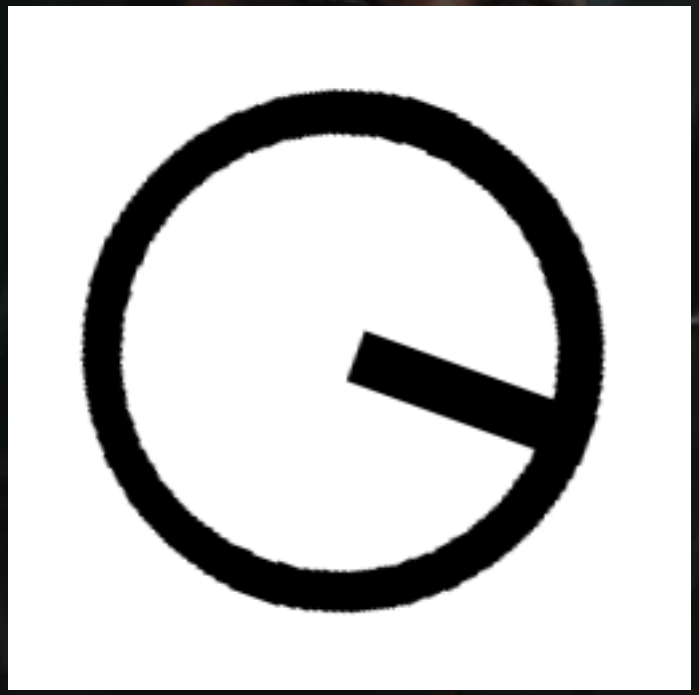
How it works…

- Toy example:



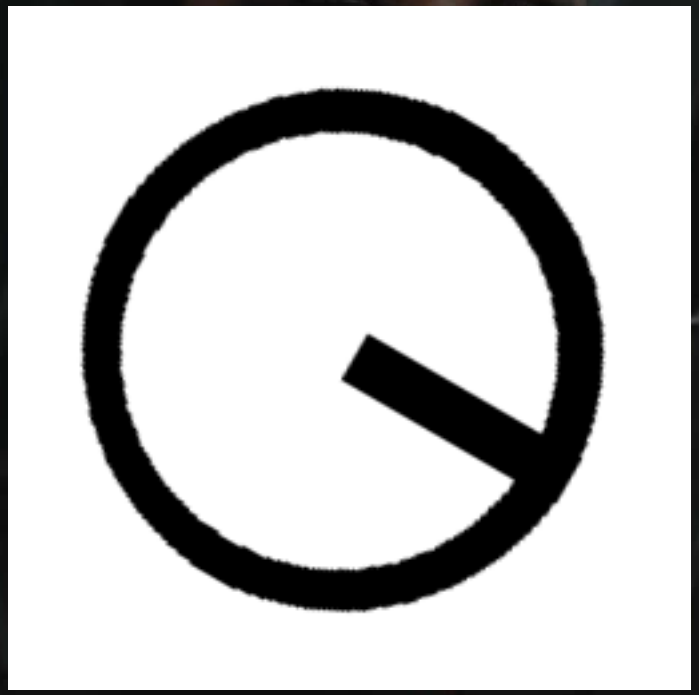
How it works…

- Toy example:



How it works…

- Toy example:



How it works…

- Toy example:

How it works…

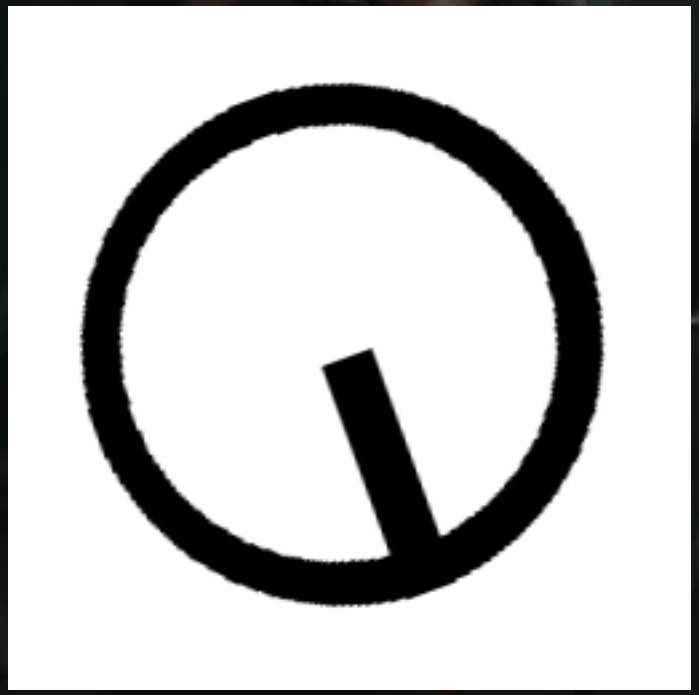
- Toy example:

How it works…

- Toy example:

How it works…

- Toy example:

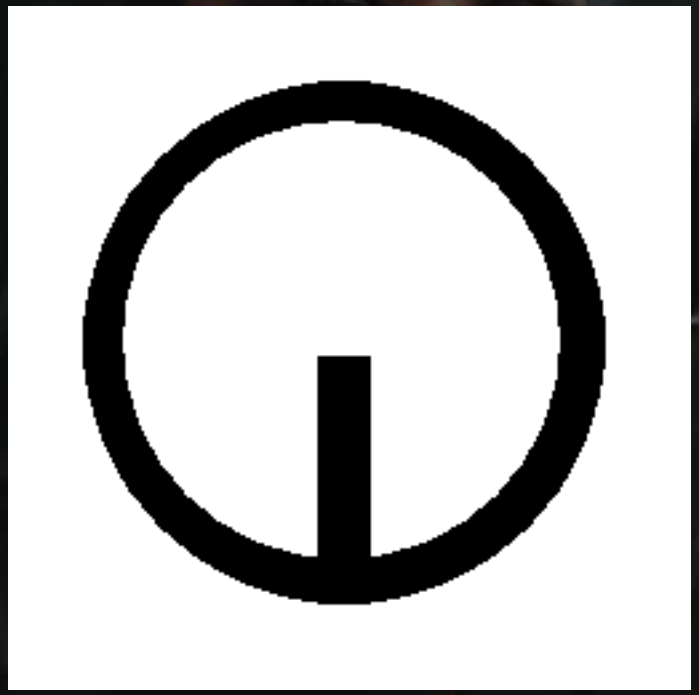


How it works…

- Toy example:

How it works…

- Toy example:

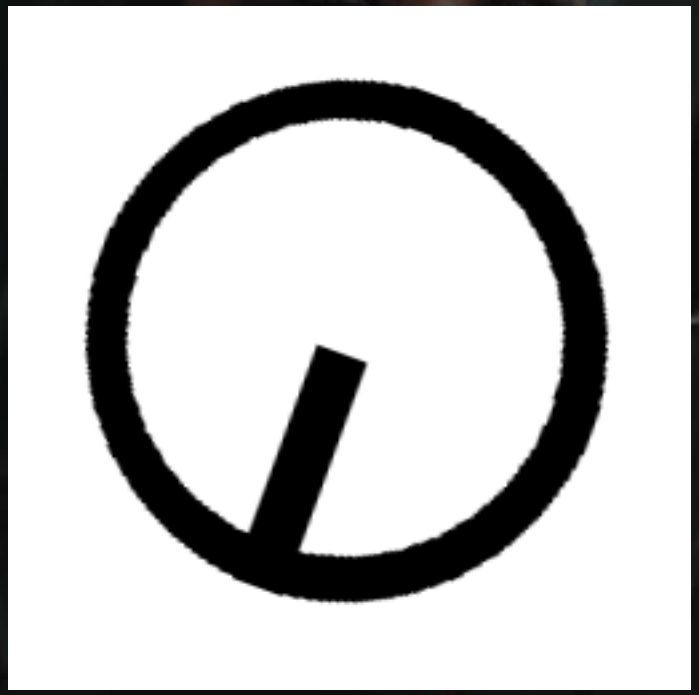


How it works…

- Toy example:

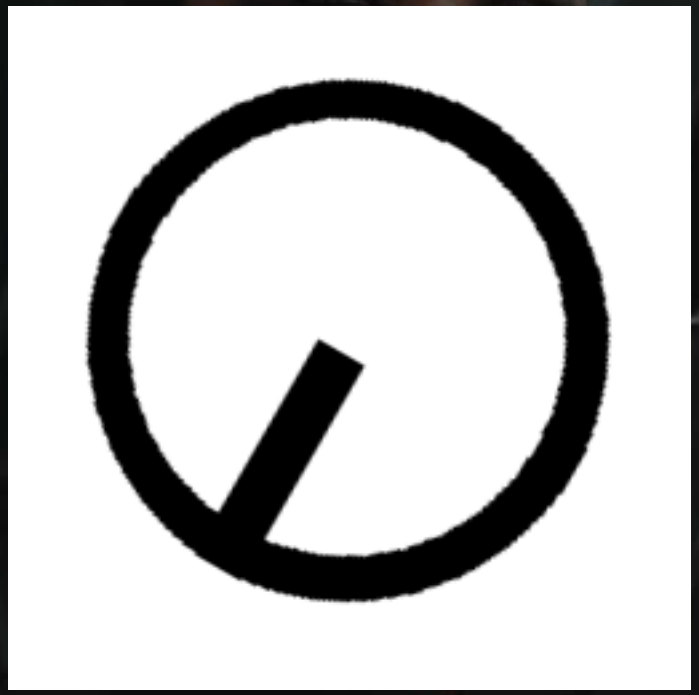
How it works…

- Toy example:



How it works…

- Toy example:



How it works…

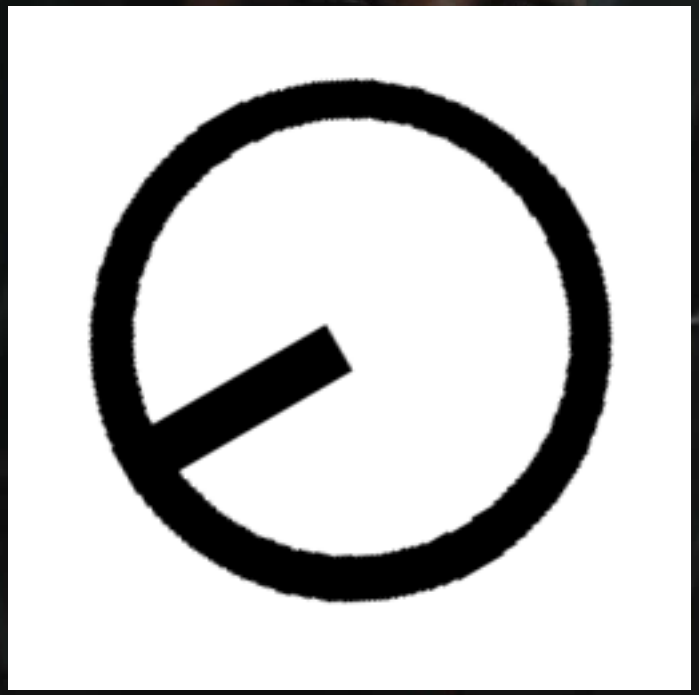
- Toy example:

How it works…

- Toy example:

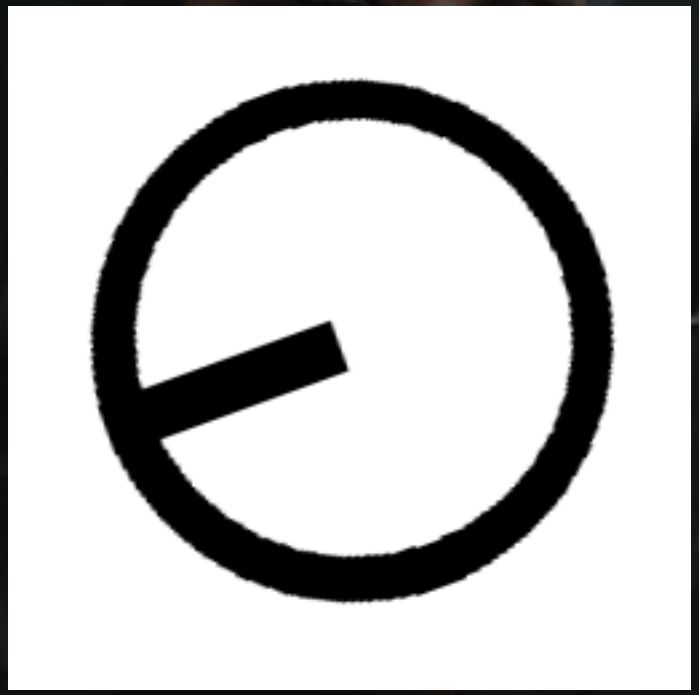
How it works…

- Toy example:



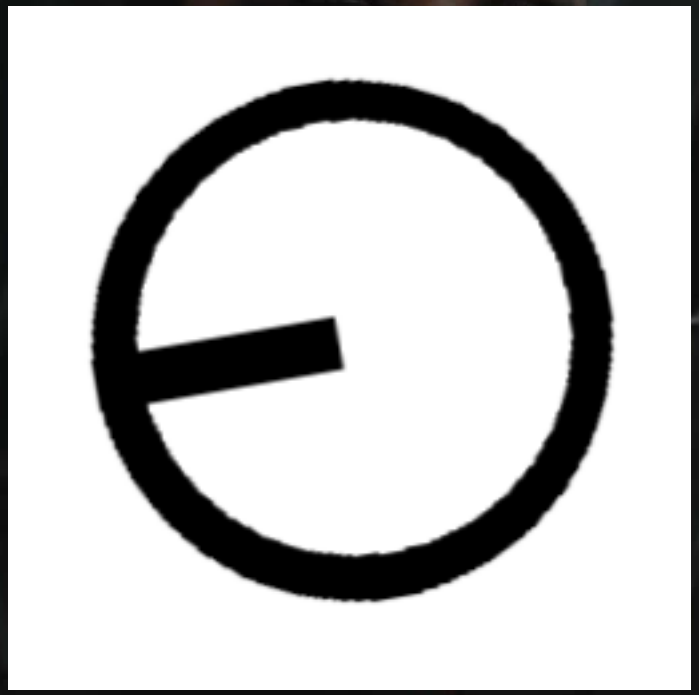
How it works…

- Toy example:



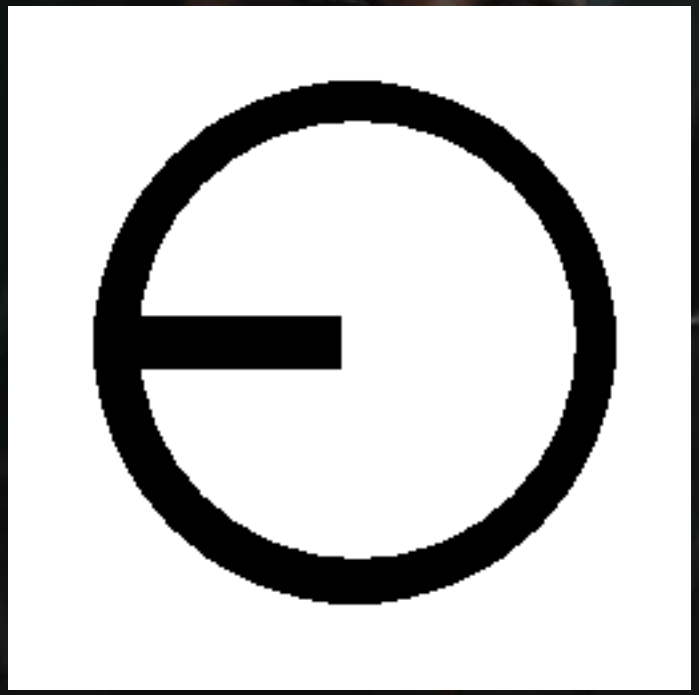
How it works…

- Toy example:



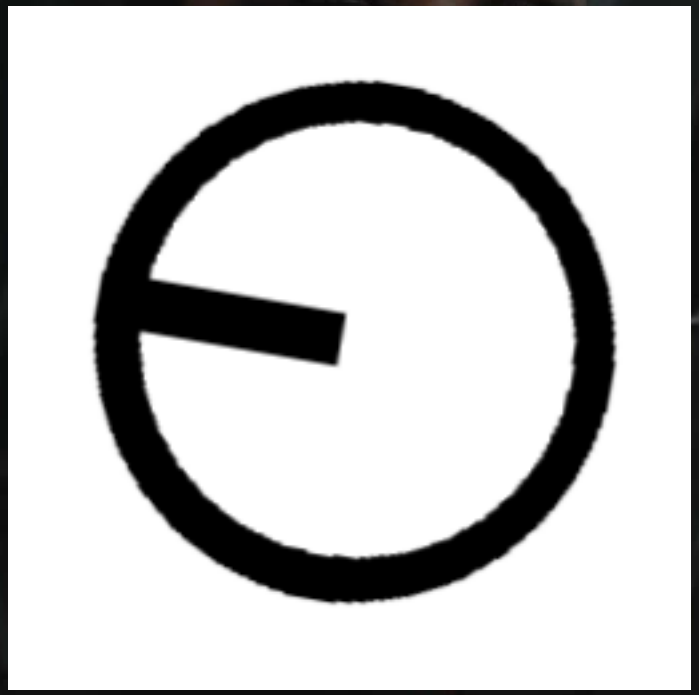
How it works…

- Toy example:



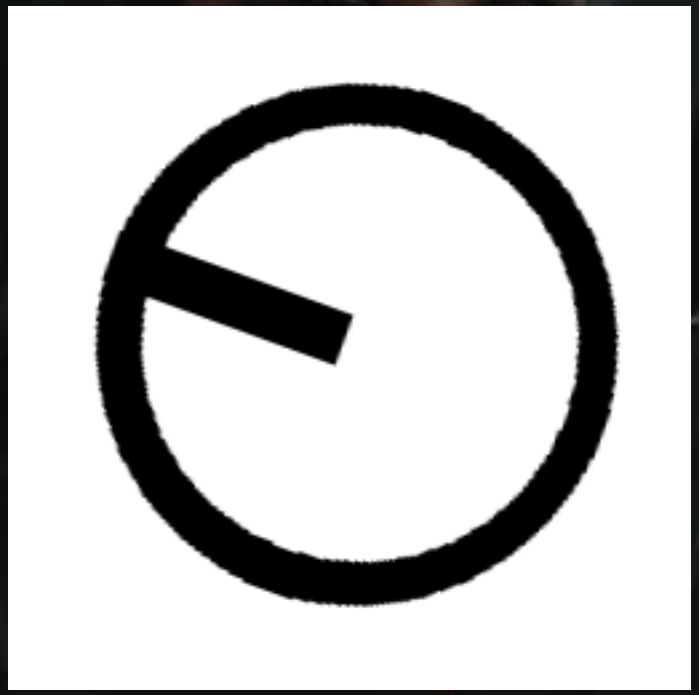
How it works…

- Toy example:



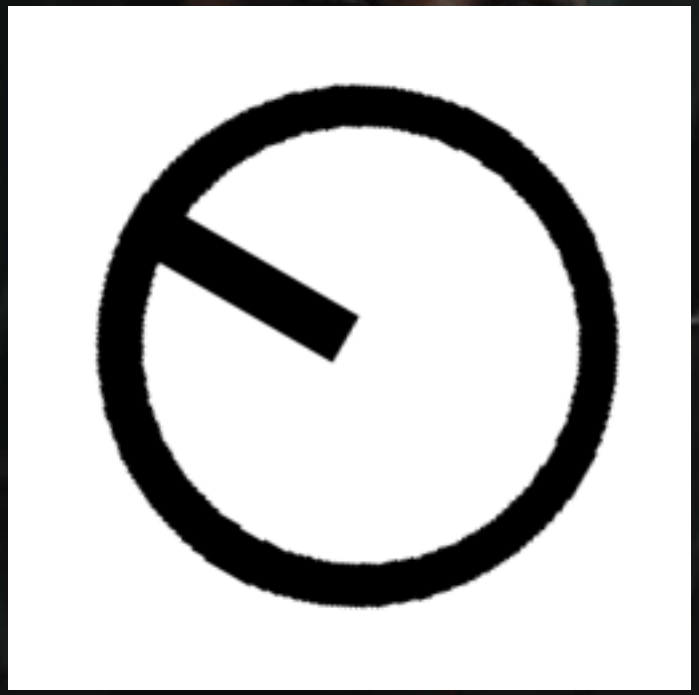
How it works…

- Toy example:



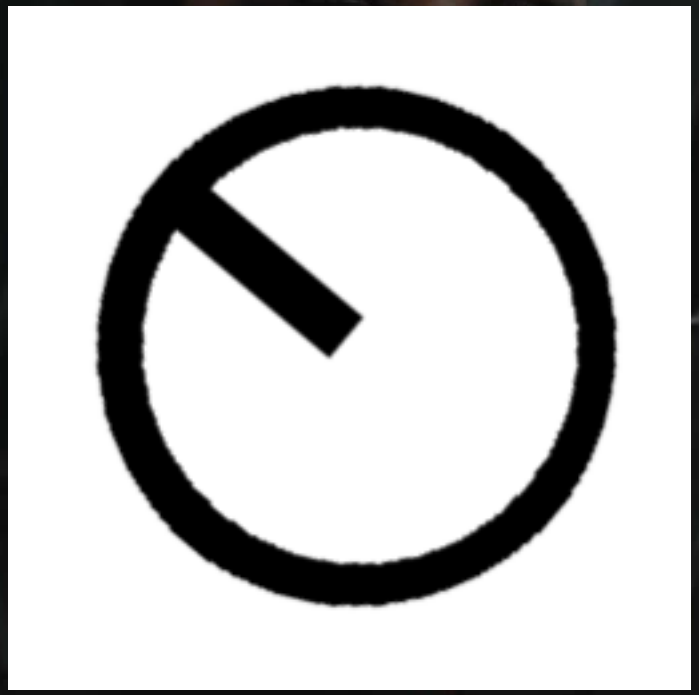
How it works…

- Toy example:



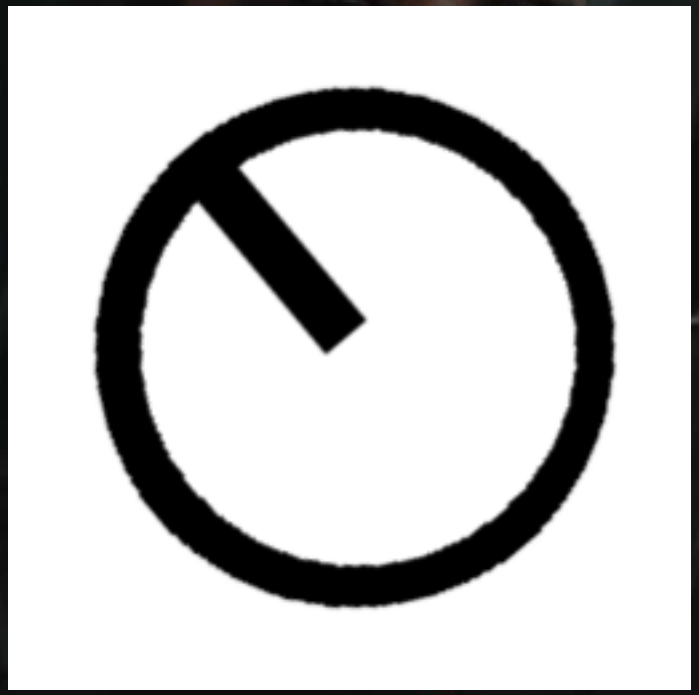
How it works…

- Toy example:



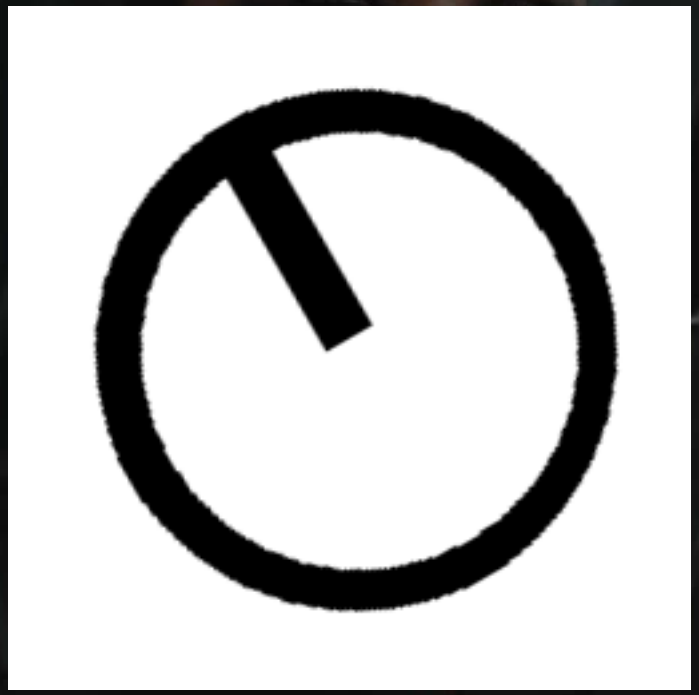
How it works…

- Toy example:



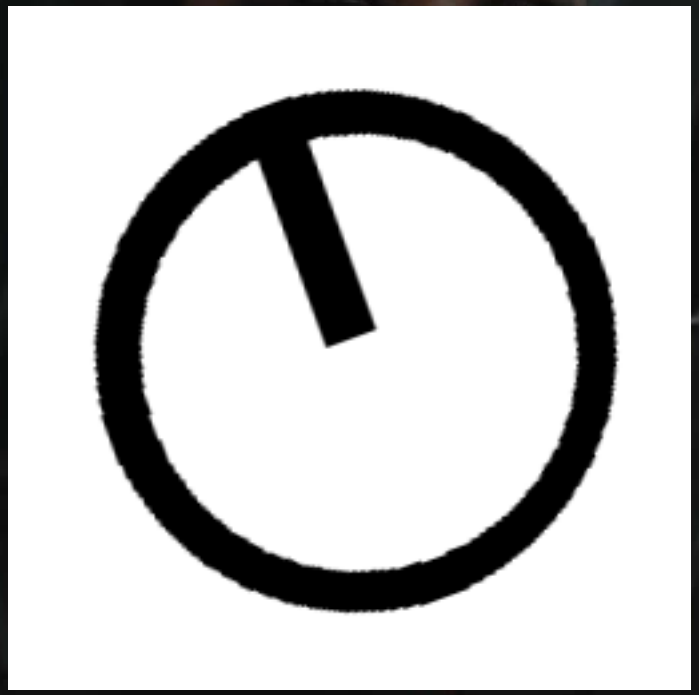
How it works…

- Toy example:



How it works…

- Toy example:

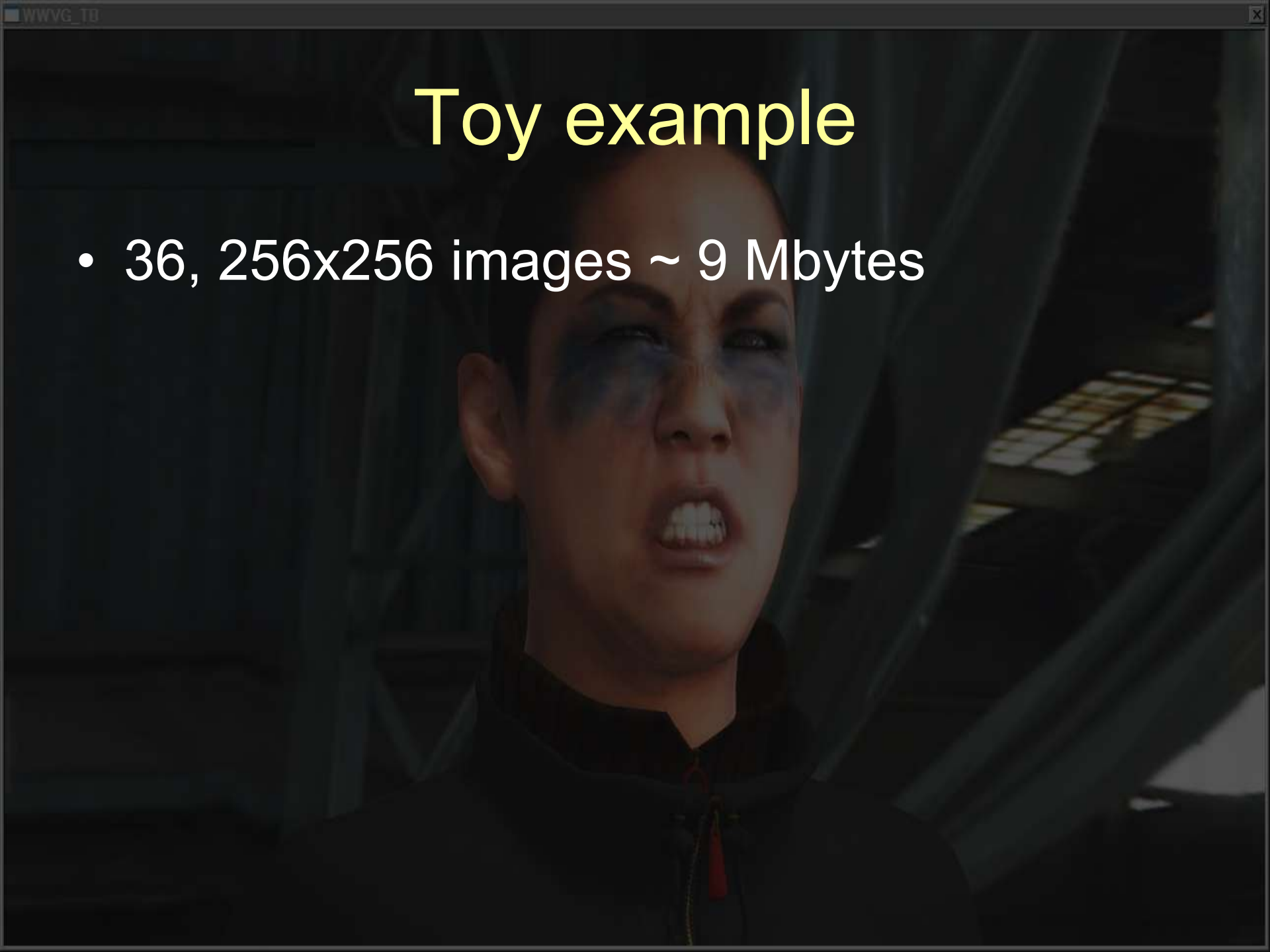


How it works…

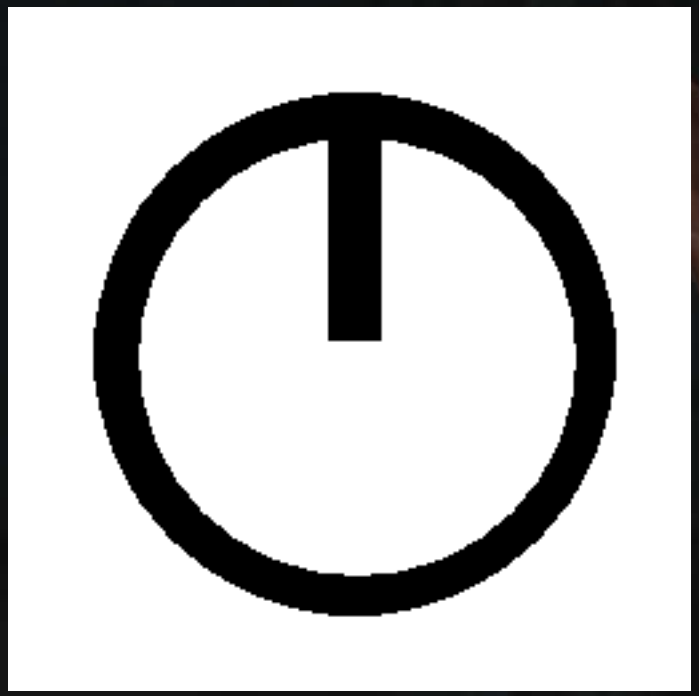
- Toy example:

Toy example

- 36, 256x256 images ~ 9 Mbytes



Alternate representation



&

Θ = { 0, 10, 20, 30, 40, 50,
60, 70, 80, 90, 100,
110, 120, 130, 140,
150, 160, 170, 180,
190, 200, 210, 220,
230, 240, 250, 260,
270, 280, 290, 300,
310, 320, 330, 340,
350 }

Alternate representation

1, 256x256 image + 36 angles

=

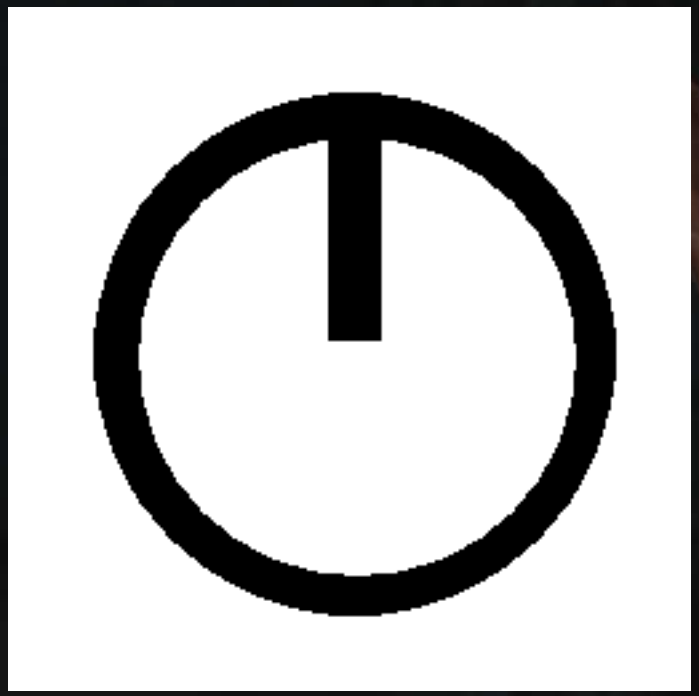
64 Kbytes

=

144:1 compression

This is PCA?

Component(s)

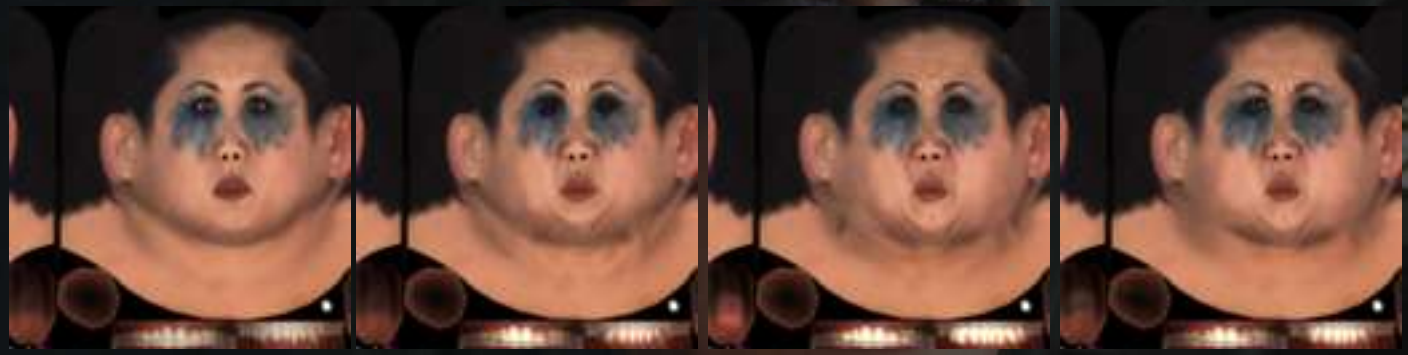


&

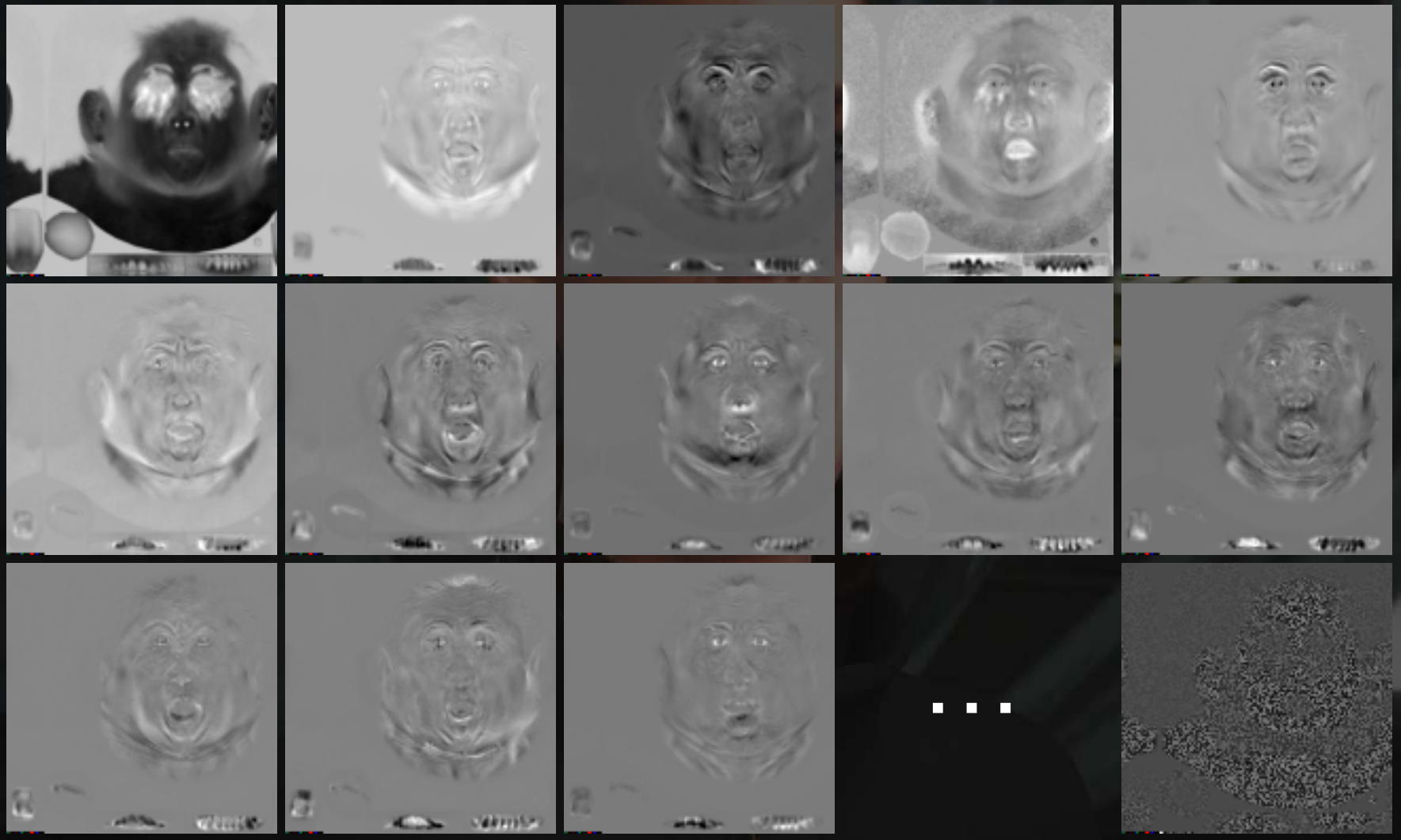
Weights

Θ = { 0, 10, 20, 30, 40, 50,
60, 70, 80, 90, 100,
110, 120, 130, 140,
150, 160, 170, 180,
190, 200, 210, 220,
230, 240, 250, 260,
270, 280, 290, 300,
310, 320, 330, 340,
350 }

Real data: not so nice

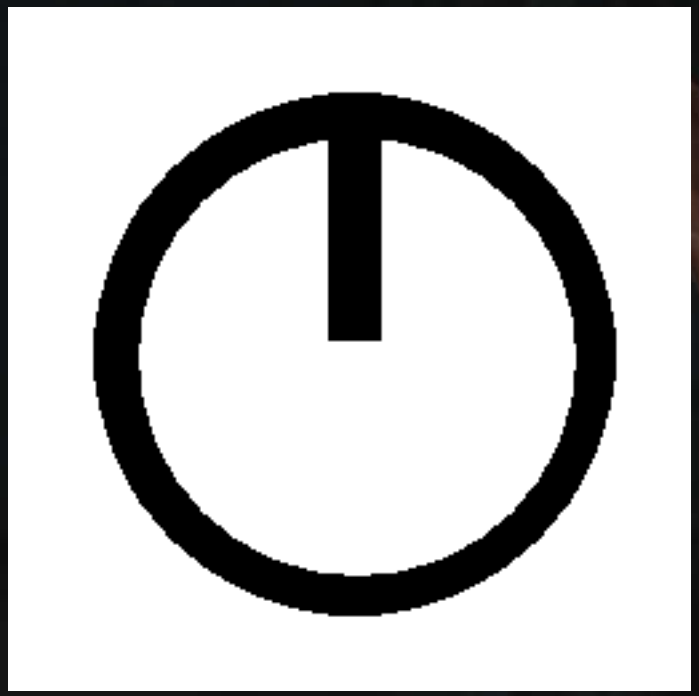


Data Components



Alternate representation

Component(s)



&

Weights

Θ = { 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 250, 260, 270, 280, 290, 300, 310, 320, 330, 340, 350 }

Alternate representation

Component(s)

C components

(For UCap, C ~ 16)

&

Weights

$\Theta = \{ w_0, w_1, w_2, w_3, \ldots \}$

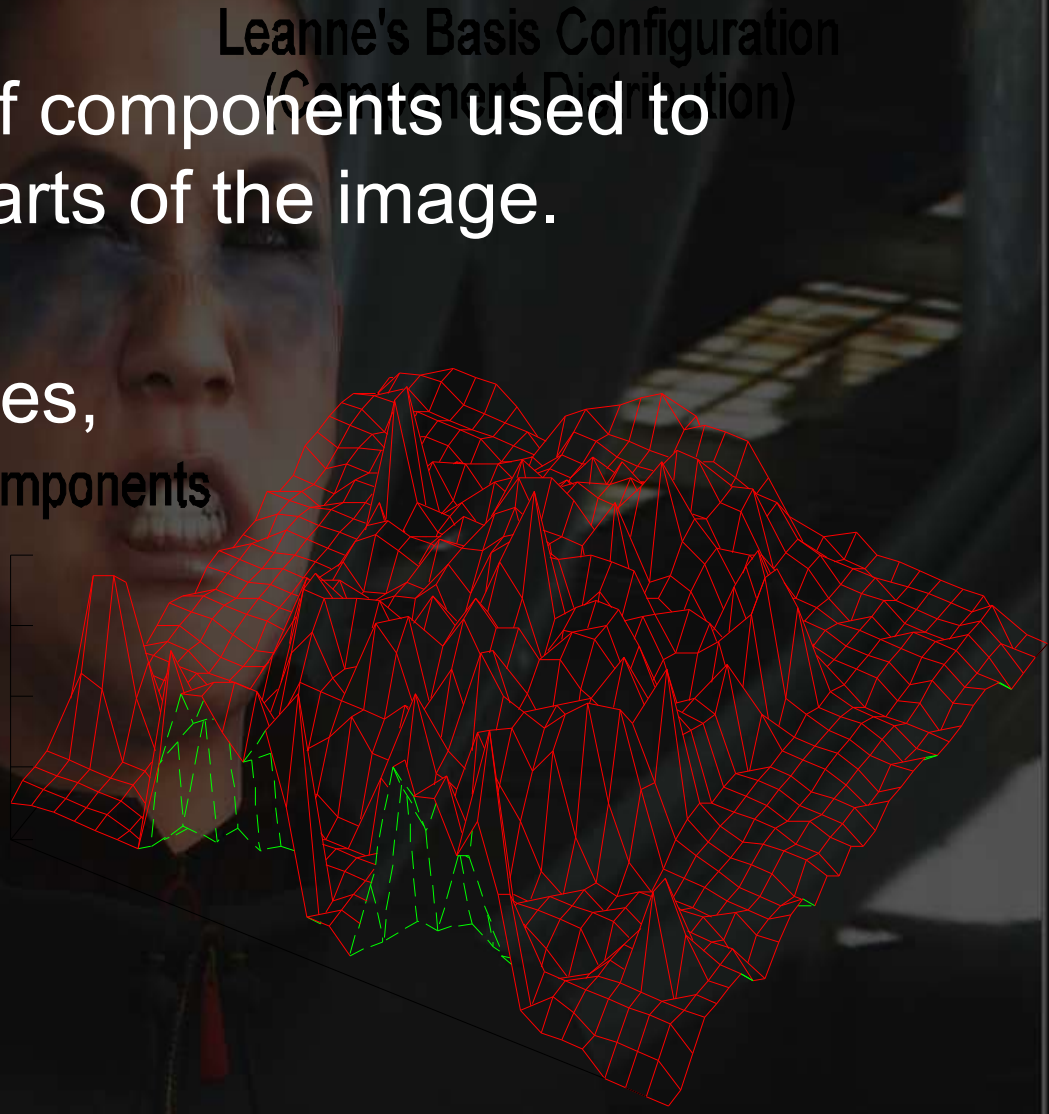
C-dimensional weight vectors

GPU Implementation

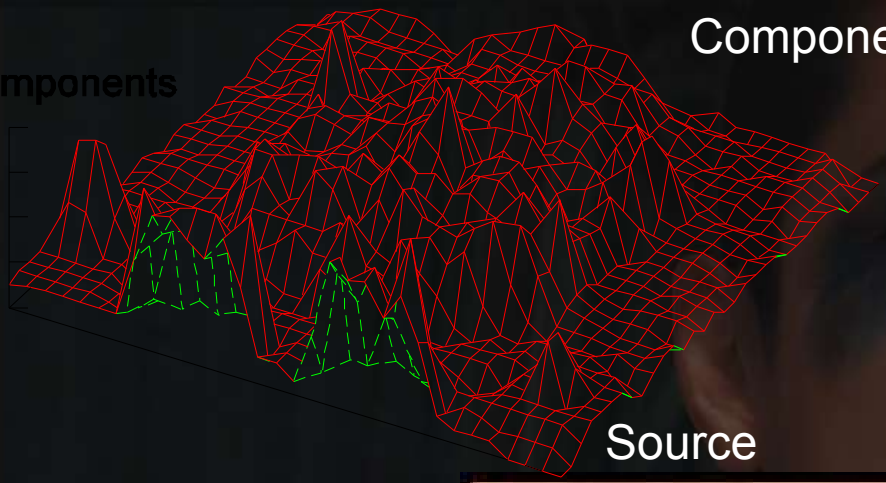
- Components are static
  - Live in GPU as textures, vertex attributes
- Weight vectors are small
  - Uploaded per frame
- Dot product is cheap

Variable Representation

- Modify the number of components used to represent different parts of the image.
- E.g., more representation for eyes, mouth, forehead
- Both automatic and artist-controlled optimization through weighting maps



Leanne Example



Component distribution

Source

PCA
(equivalent size)

Decompressed
result

Explosion Example

Component Usage

1) Original Data
2) Compressed to 1/3 size using PCA
3) Compressed to 1/12 using PCA
4) Compressed to 1/12 using VPCA

Compression Conclusion

- PCA/VPCA compression technique
  - High compression ratio (UCap performance in ~8Mbytes)
  - Low bandwidth requirements (16 float upload per frame)
  - Low decompression complexity (1 dot product of a 16D vector)

UCap Sequencing

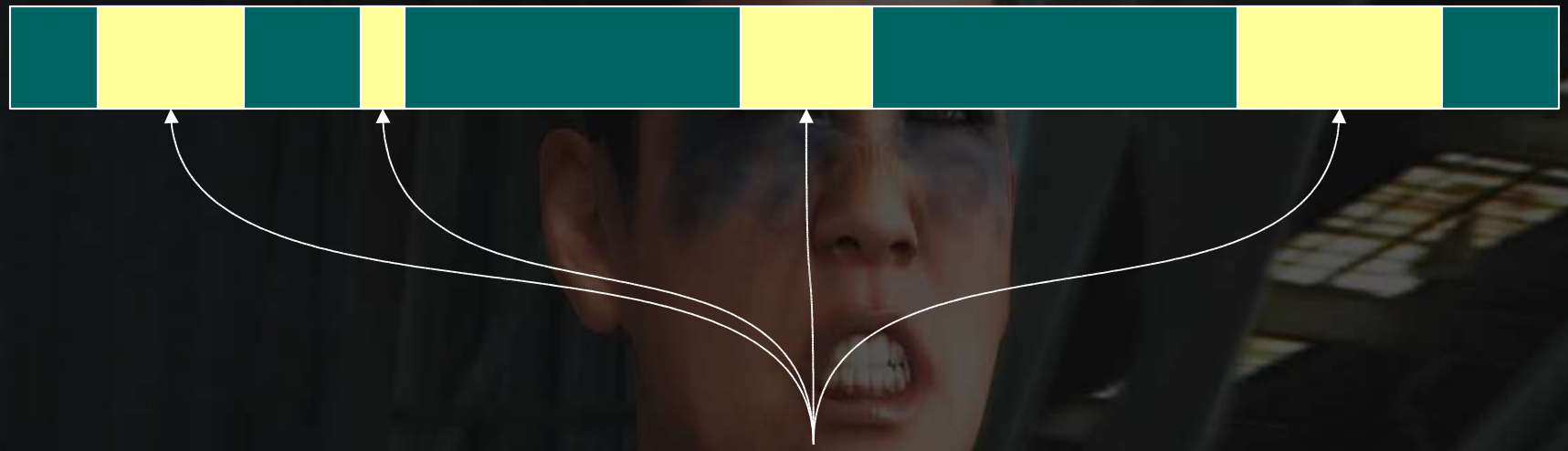
- Segment captured sequences and form triggerable state machine
- ANT authoring
- Geometry and texture blending over transitions
  - Pre-decompression blending (component weights)

Facial Motion Graph

Full performance

Facial Motion Graph

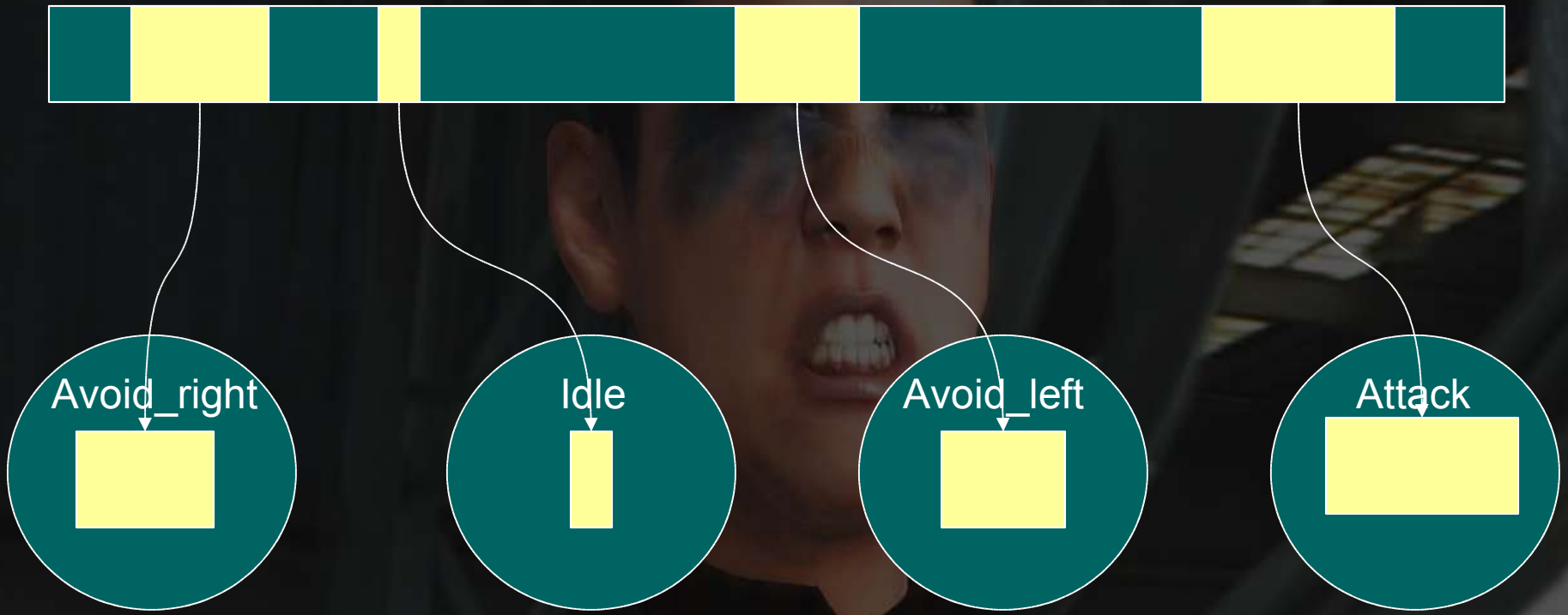
Full performance



Select desired facial expressions

Facial Motion Graph

Full performance



Expressions become states in state machine

Facial Motion Graph

State machine triggered (AI/game pad/etc.) to sequence facial expressions

Geometry & Texture blended across transitions

Performance-Driven Facial Animation

Lance Williams

Advanced Technology Group
Apple Computer, Inc.
20705 Valley Green Drive
Cupertino, CA 95014

ABSTRACT

As computer graphics technique rises to the challenge of rendering lifelike performers, more lifelike performance is required. The techniques used to animate robots, arthropods, and suits of armor, have been extended to flexible surfaces of fur and flesh. Physical models of muscle and skin have been devised. But more complex databases and sophisticated physical modeling do not directly address the performance problem. The gestures and expressions of a human actor are not the solution to a dynamic system. This paper describes a means of acquiring the expressions of real faces, and applying them to computer-generated faces. Such an "electronic mask" offers a means for the traditional talents of actors to be flexibly incorporated in digital animations. Efforts in a similar spirit have resulted in servo-controlled "animatrons," high-technology puppets, and CG puppetry [1]. The manner in which the skills of actors and puppetteers as well as animators are accommodated in such systems may point the way for a more general incorporation of human nuance into our emerging computer media.

The ensuing description is divided into two major subjects: the construction of a highly-resolved human head model with photographic texture mapping, and the concept demonstration of a system to animate this model by tracking and applying the expressions of a human performer.

Cr Categories and Subject Descriptors: I.3.7 [Computer Graphics]: Three Dimensional Graphics and Realism--Animation. I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling -- Curve, surface, solid, and object representations. J.5 [Computer Applications]: Arts and Humanities--Arts, fine and performing.

General Terms: Algorithms, Design.

Additional Keywords and Phrases: Animation, Facial Expression, Texture Mapping, Motion Tracking, 3D Digitization.

BACKGROUND

The seminal work of Parke [2] involved the photogrammetric digitization of human faces and expressions, and the creation of a parametric model that used local interpolations, geometric transformations, and mapping techniques to drive the features of a computer-animated face. Demonstrations of the face in action included lip-synchronized speech.

Platt and Badler [3],[4] applied physical modeling techniques to the parametric face problem. Once again, the goal was to drive the face at a higher level of control. Simulation of muscles in this paper established the use of deformation functions, rather than interpolation, to animate a face.

Brennan [5], working with hand-digitized 2D vector faces, implemented a kind of automatic caricature by exaggerating the differences of individual subjects from a prestored "norm."

Burson and Schneider [6] established a mapping correspondence between facial features in different photographs, a mapping defined by hand-digitizing key points of correspondence. Changes in shape and texture between one pair of photographs could then be used to map changes in shape and texture to a third. This process has been used to artificially "age" the photographs of missing children (that is, to estimate an image of the same child some years hence). Similar mappings have been used by Burson and Kramlich (Face Software, Inc., NY., NY.) to interpolate faces and create composites.

Concurrently, experiments at New York Institute of Technology involved mapping 2D animated features onto 3D characters (James Blinn and the author, 1976), and mapping of live-action video faces onto 3D "masks" (3D face surface models in which the eye and mouth regions have been smoothed over, so as not to interfere with the moving lips and blinking eyes of the mapped video). Live-action mapping was first essayed by Duane Palyka and the author in 1977, and was applied by Paul Heckbert in the NYIT videos, "Adventures in Success," and "3DV" (1984). At the same time, NYIT researcher Tom Brigham was doing extensive work with screen-space (2D) interpolation of texture and form. Brigham's "shape interpolation" was applied to faces, among other subjects, and must be considered an influence on this work. One conclusion of these experiments was that much of the detail and realism of the faces depicted was simple surface texture. A face model like Parke's would be much more powerful and convincing with photographic mapping, and perhaps more individual and personable, as well. The photographic mapping of [7] illustrates the power of texture alone, without surface shading of any kind.

An extended, expressive facial animation was essayed by Phillipe Bergeron, Pierre LaChapelle, and Daniel Langlois in 1985, in their short, "Tony de Peltrie"[8]. Photogrammetrics as in [2] were used to digitize both human expressions and the neutral features of a stylized model. Then, scaled differences of the various human expressions from the "neutral" human expression, were applied to the neutral stylized model. This basic cross-mapping scheme, as in [5], requires a norm to apply the mapping. Since the mapped differences are scaled, "caricature" (this time in 3D) is straightforward.

The most recent wrinkle in facial animation is Keith Waters' application of tailored local deformation functions analogous to musculature in animating a neutral face definition [9]. This technique was adopted by the Pixar animation team to control the patch-defined baby face of "Tin Toy" [10].

The formulations described in this paper are an attempt to extend the mapping of texture and expression to continuous motion mapping. Using current technologies, both human features and human performance can, in the opinion of the author, be acquired, edited, and abstracted with sufficient detail and precision to serve dramatic purposes.

Figure 1. Processed model with mosaic texture map

CONSTRUCTING THE MODEL

Dancer Annette White, a personal friend of the author, was a featured performer in Patrice Regnier's RUSH dance company which performed at SIGGRAPH in 1985. Her head was cast in plaster by Long Island sculptor Val Kupris, and the surface of the plaster head digitized at Cyberware, Inc. [11]]. A number of conventional color 35mm photos of Annette were taken at this time by Nancy Burson of Face Software, Inc., with the intent of creating a complete composite photograph (in cylindrical projection) of Annette's head to match the scanned relief data of the plaster cast. Face Software shared an interest in developing electronic mask technology, and cooperated in the initial test of the model. They were able to apply their mapping techniques to warp sections of the multiple photos to the range data acquired from scanning the plaster head. To facilitate this process, the 16-bit range data was compressed to eight bits, and a Sobel filter applied. This "edge enhancement" filtering brought out the important features (edges of the lips, corners of the eyes) from the otherwise ghostly range image. The mapping, based on a triangular mesh, is shown applied to the model in figure 1. Below the two projections of the model in the photo are the composite texture map, produced by Face Software, and an eight-bit compression of the range image after Sobel filtering, to the right. (The black regions in the filtered range image are the result of a color map artifact; they should be white.) A kerchief is worn about the head in this texture map. A shading discontinuity between photographic sections is visible in the composite map on the left.

Cyberware had been sent a plaster cast of Annette's head, and sent back a tape with the digitized surface data, scanned four times. Data were in the form of 16-bit integers defining a mesh surface in a cylindrical coordinate system, each entry a radius indexed by phi (azimuth) and Y. The mesh consisted of 512 columns of samples in phi, by 256 rows in Y. The mesh was in fine shape to serve as a map with which to register the photographic texture, but in the meantime, required some additional work to serve as a satisfactory 3D model.

The usual fashion in which Cyberware displayed digitized data, at the time this work was undertaken, was as a faceted model. Missing data -- data obscured from the sensor by occlusion of the projected laser line -- was mapped to the center of the Y axis, and was in most cases invisible. If the data were Gouraud shaded in a straightforward fashion, the missing samples would severely impact the visible ones, by clobbering the computed surface normals. In order to make smooth shading and texure mapping possible, it was necessary to refine the model: to "heal shut" the missing data, by interpolating the surface across neighboring samples. It was moreover desirable to apply a certain amount of data smoothing to suppress spurious texture, and to use larger local neighborhoods of samples to estimate surface normals. Such processing of the digitized surface, very similar to image processing, proved essential for subsequent mapping and shading. An outline of the surface processing steps follows.

Healing Shut the Missing Data

Missing data is marked, in the Cyberware format, by a reserved "Z" value. The first step in surface processing is to restore surface continuity, a prerequisite for many subsequent operations. The scanner data is copied into a floating point array with missing values set to 0.0, and a second floating point array, a matte, is set to 1.0 where valid data exists, and 0.0 elsewhere. In the vicinity of 0.0 matte values, a small blurring kernel is applied to both the matte and the surface. Where the matte value increases (due to the blur) above a threshold, the surface sample is replaced by the blurred surface divided by the blurred matte. Filtering is recursively applied until no matte values are below threshold. Thus, the missing data are replaced by smooth neighborhood estimates, that gradually grow together. An implementation of this basic general scheme was coded so that the smooth replacement regions could "grow outward" slightly, smoothing the correctly-acquired data at the boundaries. All iterations of the "healing" process were visible, and the matte threshold, outward creep, and blur kernel could be varied for different results. Practically speaking, the process worked well for the small regions of missing data on the digitized face. It could not be expected to perform well for missing regions much larger than the blur kernels used. For this reason, a more detailed discussion of the filtering algorithm would not be justified. For future work, the methods of [12] are preferred.

Hysteresis Filtering

In many types of data acquisition, noise at the scale of the sampling interval is particularly bothersome. Special

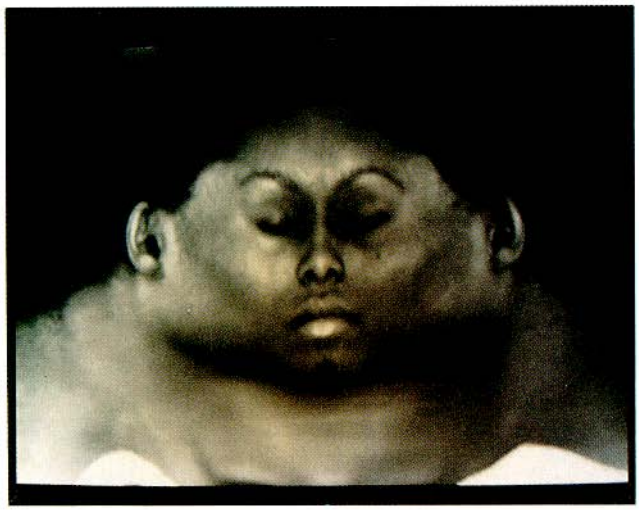


Figure 2. Processed peripheral photograph.

Figure 3. Sobel-filtered range image warped into a rage.

techniques for removing "salt and pepper" noise from imagery have been adopted for many purposes. A typical model will dampen high-amplitude sample-frequency noise in a nonlinear way. For the filtering used in the head model, a smooth estimate of the surface at a point was computed by a 3x3 blur kernel with unity gain and 0.0 as the center sample coefficient. If the estimate differed from the center sample by greater than a threshold amount, the center sample was replaced by the estimate. Such a filter is occasionally called a Tukey filter, or a "hysteresis" filter (because it modifies the data only when the threshold is exceeded).

Subjectively, this filtering process smoothed out a number of glitches and spurious details in the surface of the face. It performed equally well on small bubbles in the plaster cast and on data that, for whatever reason, seemed somewhat rougher than the plaster.

Low-Pass Filtering

By filtering and downsampling data, it is possible to trade precision for resolution directly. This tradeoff occurs in a continuous way as data are low-pass filtered. The reduced bandwidth of the surface is accompanied by an increase in precision for each of the samples. Where the data to be estimated are more bandlimited than sources of noise in the system (where the signal is pink, and the noise is white), low-pass filtering will increase the signal-to-noise ratio.

A weak low-pass filter was applied to the cylindrical R (range) coordinates of the data, before conversion of the surface from R(ø,Y) to X(u,v), Y(u,v), and Z(u,v). The skin seemed smoother after filtering, and faint striations visible in the surface normals (assumed to be scanner artifacts) diminished.

Filtering the Normals

In fact, surface normals are extremely sensitive indicators of perturbations in the digitized surface, including all noise and artifacts. The usual methods of computing surface normals from polygon meshes are too local for imperfect data so closely spaced. It is reasonable to smooth such local normal estimates, or to use normals of a surface somewhat smoother than the one actually displayed. In this way, satisfactory shading can be achieved without smoothing features from the surface.

When filtering is applied to the range data in the cylindrical mesh, artifacts occur at the poles, where sample aliasing is greatest (because the laser line may project very steeply). Because the spacing of the samples is far from uniform in 3D space, such filtering is very anisotropic. On the other hand, filtering the components of the derived normals is much more expensive, and requires special care. Some normals may vanish (magnitudes go to zero) after filtering; all normals require re-unitization.

Some polar artifacts were tolerated in the range-filtering stage, with the intent of completing the top of the head with a "scalp mesh." This makes sense because the texture, as well as the model, would otherwise be very poorly resolved on top of the head. Despite the difficulties imposed by the "singularity," the polar map is a very convenient representation for interacting with the head model. It provided the basis for interactive texture mapping, and interactive setup of expressions.

Registering Cylindrical Texture

Most of the model processing steps described could be performed automatically. The most tedious process described so far was that performed by Face Software: taking the photographs necessary to completely record the subject's head, and then painstakingly registering the mosaic of photos to the model range data. Ideally, this step, too, would be automatic. A scanner which simultaneously captured R, G, B, and range, would simplify "mask" acquisition greatly. Such a scanner has only recently become commercially available[11].

In order to generalize the texturing process, as well as to exercise the "virtual muscles" expected to drive the face, an alternate method was devised. The first step was to capture the head texture completely in a single photo. Traditional photographic methods exist, based on slit-scan cameras. A camera which exposes its film by moving it continuously past a vertical slit, while the camera and lens rotate horizontally, is termed a "panoramic" camera. If the camera is stationary while the subject rotates on a turntable, the device is termed a "peripheral" camera. Panoramic cameras are commercially available, and adapting one for peripheral photography should be straightforward. The peripheral camera used in this work was built by Prof. Andrew Davidhazy of the Rochester Institute of Technology. Prof. Davidhazy was kind enough to

photograph Annette's head while she stood on a rotating turntable; her dance experience proved very useful in maintaining a vertical and unwavering posture during the spin. The resulting photograph, like the Cyberware scan, is a cylindrical projection.

The peripheral photograph was scanned and digitized. An interactive texture-warping program was written for an SGI Iris workstation. Using the 12-bit double-buffered display mode, the user could ping-pong back and forth between a Sobel-filtered range image and the peripheral photo. Standard digital "painting" functions were provided by the program, but its key feature was a "coordinate airbrush." This is simply a deformation function in the style of [7] and [9], described in digital painting terms. In fact, the "coordinate airbrush" implements an inverse, rather than a forward, mapping. For many purposes this is equally convenient, and much more rapid to compute. The idea is that an X, Y coordinate offset is supplied for the center of the brush (so the center pixel is replaced by the pixel value in the image at the offset X, Y), and the airbrush kernel tapers the blend of the coordinate offsets away from the center of the brush. If the offset is larger than the span of the airbrush kernel, a singularity results, like a dimple in a specular surface. In fact, these

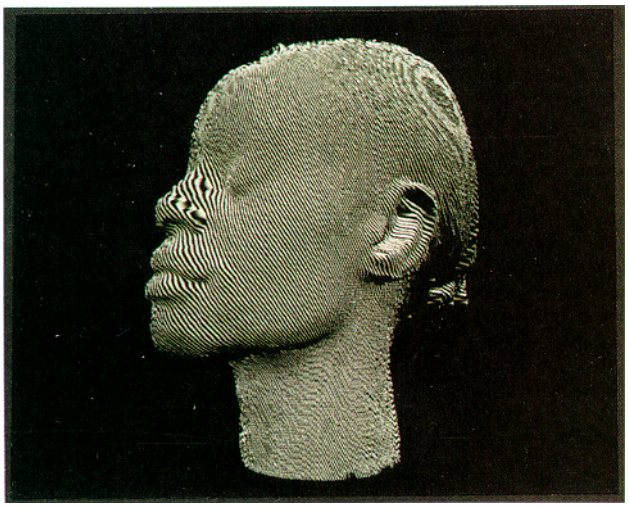


Figure 5. "Zebra" triangulation.

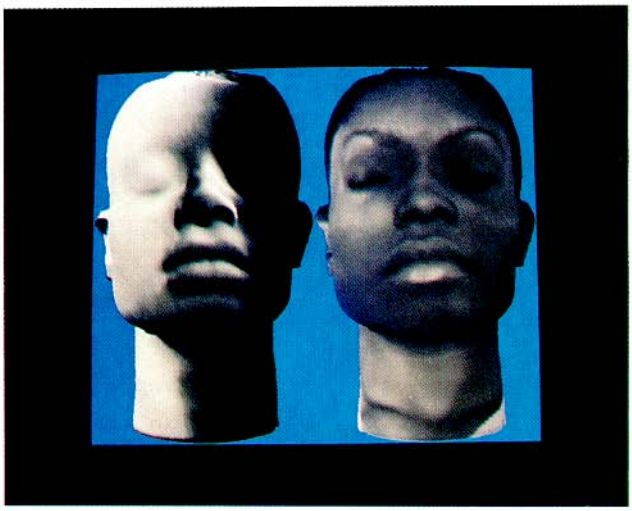


Figure 4. Final model, with and without texture.

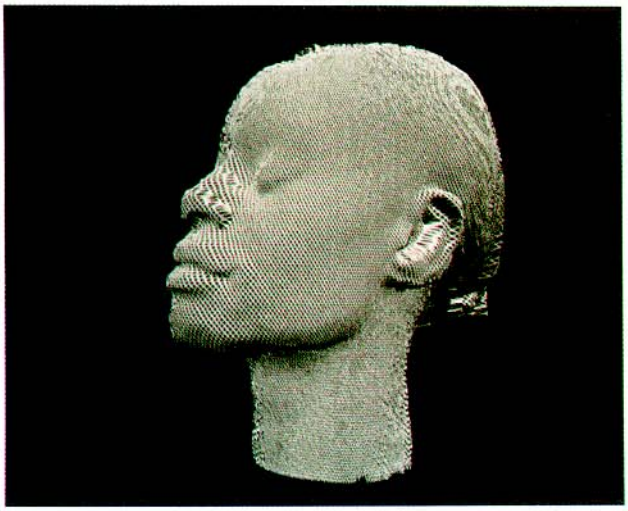
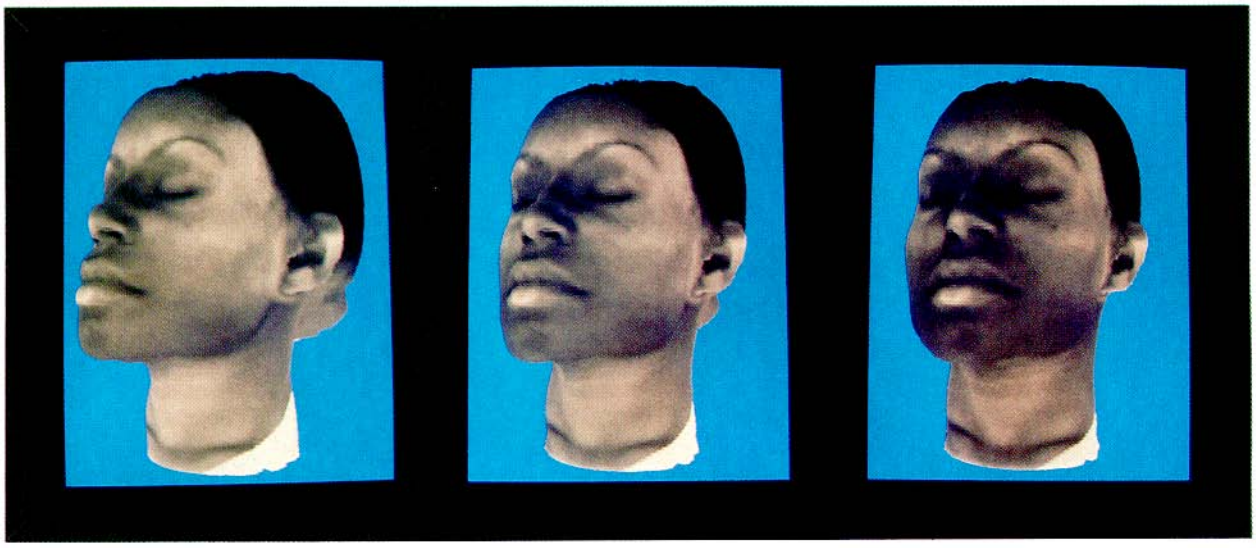


Figure 6. "Serpentine" triangulation.

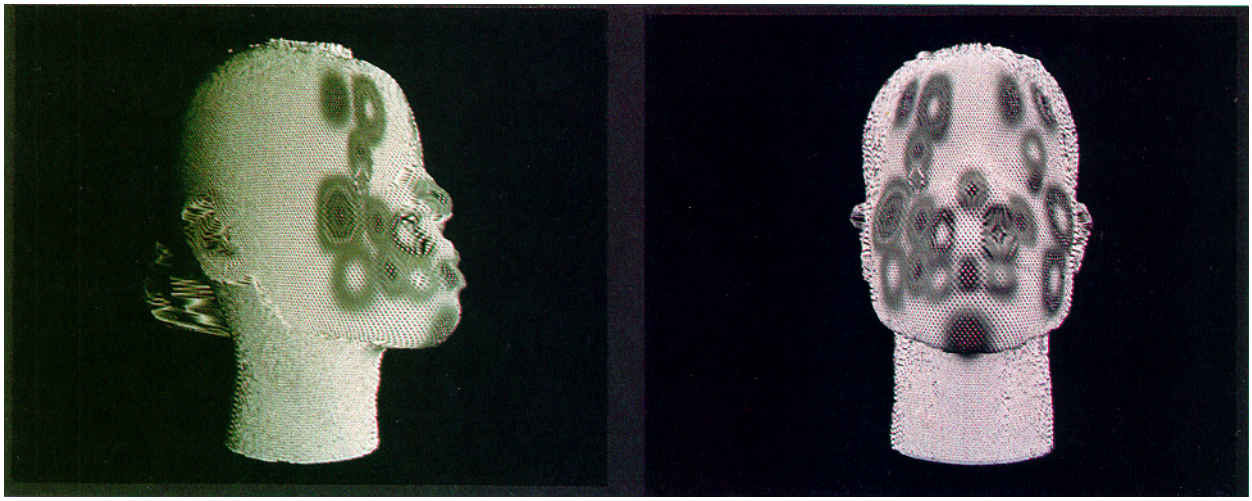
inverse mappings resemble nothing so much as reflections from curved mirrors. If the offset is smaller than the span of the brush kernel, then the mapping behaves much like the "forward" mapping, like a rubber sheet that can be stretched to fit points of correspondence together. Figure 2 shows the final texture after remapping. The general technique was to use large kernels first, and then use smaller and smaller warps to register the details. Registering the texture with the filtered range image took about three hours of interaction. Figure 3 shows a Sobel-filtered range image which has been interactively warped to change its neutral expression to one of profound annoyance (the contouring in the image is a result of the compression to 8 bits). This illustrates the primary motivation for development of this "warpware": to apply similar deformation functions to animation of the 3D model.

Figure 4 shows the model with and without texture. Rendering was performed using the SGI Iris graphics pipeline, and supporting rendering software by G. W. Hannaway & Associates. With a generation of graphics hardware that does not handle texture explicitly, many users effect "texture mapping" by rendering large meshes of polygons, with texture multiplying intensities at each mesh vertex. A mesh of vertices with the same resolution as the texture map essentially reproduces the map, warped to shape, with bilinear interpolation (Gouraud shading). To compress the map, filtering is necessary. In approaching texture this way, which is certainly a practical expedient on today's Z-buffer polygon engines, mesh tesselation is an issue. Figure 5 shows the head model textured with a pixel-scale checkerboard. The striping of the texture is due to triangulation. To avoid sending twisted quadrilaterals, all mesh elements are first divided into triangles. In figure 5, the triangulation diagonals cut across the mesh in parallel, and become stripes. In figure 6, the same texture is rendered with a "herringbone" or "serpentine" triangulation. Alternate mesh quads are divided on alternate, criss-crossing diagonals. Note that the texture still does not look like a checkerboard (we could hardly expect it to at this scale), but is much more isotropic.

After the mapping was corrected, an animation of the head rotating about the Y axis was taped, and the result examined



Figures 7-9. Rotations of the mapped 3D model.



Figures 10-11. Basis functions on the model face.

for registration (which looked good!). Figures 7-9 show the model rotating, exhibiting the mapped texture in 3D.

MODIFYING EXPRESSION

The next step is applying our warpware to animation of the model. First, a set of warping kernels is distributed about the face. The factors that govern this distribution include physiology (the placement of muscles in the face, and the location of the most motile areas) as well as various practical considerations relating to the planned use of spot-tracking on an actor's face to ultimately drive the model. The "basketweave" texture of figure 6 has been applied to the faces of figures 10 and 11, and the set of basis functions driving the face has been slightly offset. The resulting interpolations display the centers and relative sizes of the control kernels. The kernels are larger than they appear, but they are radially symmetric (like the basis functions of [7], rather than the more elaborate bases used by [9]). Each is a Hanning (cosine) window, scaled to 1.0 in the center, and diminishing smoothly to 0.0 at the edge, with no negative values.

Like the enraged expression applied to the range image of figure 3, the expressions of figures 12 and 13 were created by hand-warping (with the "coordinate airbrush") the polar representation of the model. The realistic model can be stretched in a completely unrealistic way, and actually resembles a latex mask in some respects.

Tracking Expressions of Live Performers

The final link in our proof-of-concept demonstration is to increase realism by deriving the basis-function control offsets from the expressions of a live actor. Video-based tracking was the method of choice, although various mechanical schemes were considered. Video offers the most leverage

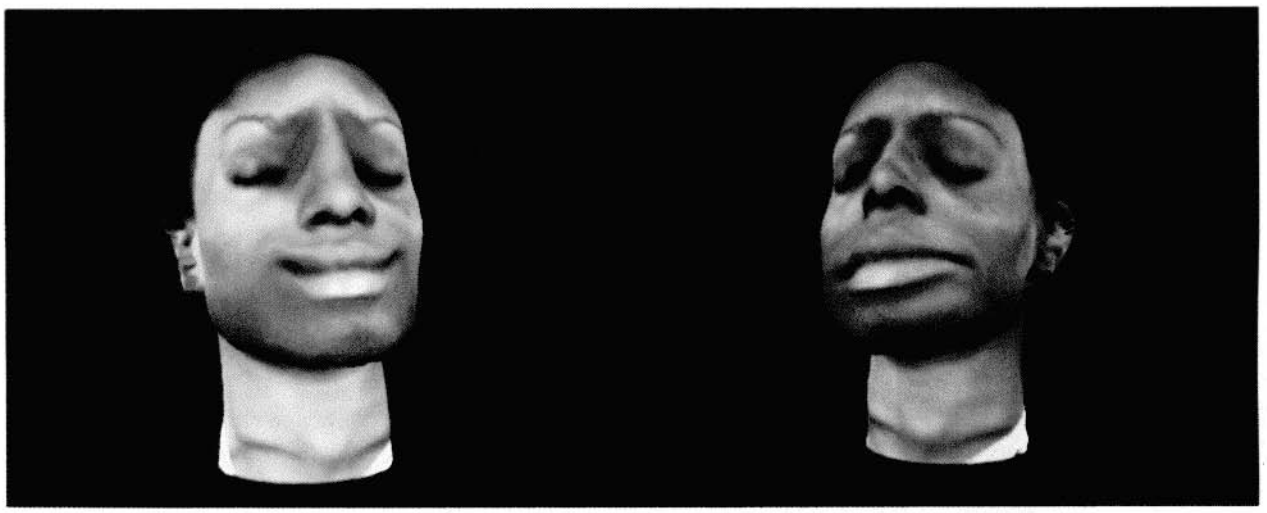


Figure 12. "Comedy"

Figure 13. "Tragedy"

Hand-animated exaggerated expressions.

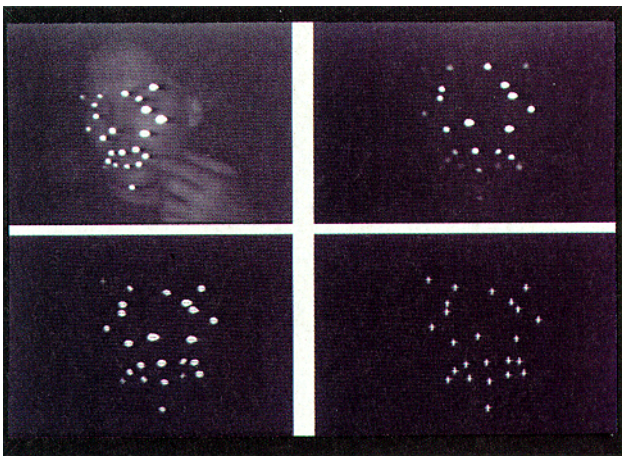
because it does not restrict the performer, it is a simple, widely-available technology, and promises ultimately to permit motion-tracking without special makeup or fiducial marks.

The "special makeup" in this case was Scotchlite®, a retroreflective material manufactured by 3M. Scotchlite uses a layer of tiny spheroids to behave much like a surface of tiny corner-reflectors; reflection efficiency is very high within a small angle from the incident light. A beam-splitter setup was employed to record the performer, which for budgetary reasons turned out to be the author. The beam-splitter is simply a sheet of window glass, between the camera and the performer, at 45° from the camera's optical axis. At right angles to the camera, aimed at the performer's side of the glass, was a slide projector, used as a light source. The setup is adjusted so that the light reflected from the glass illuminates the performer's face. The point of the apparatus is, that the light source and camera are coaxial. Since the light comes "from the camera's point of view," the efficiency of retroreflectors in the camera's field of view is very high. When properly set up, contrast and brightness can be adjusted so that the camera can see retroreflectors in its field of view, and little else.

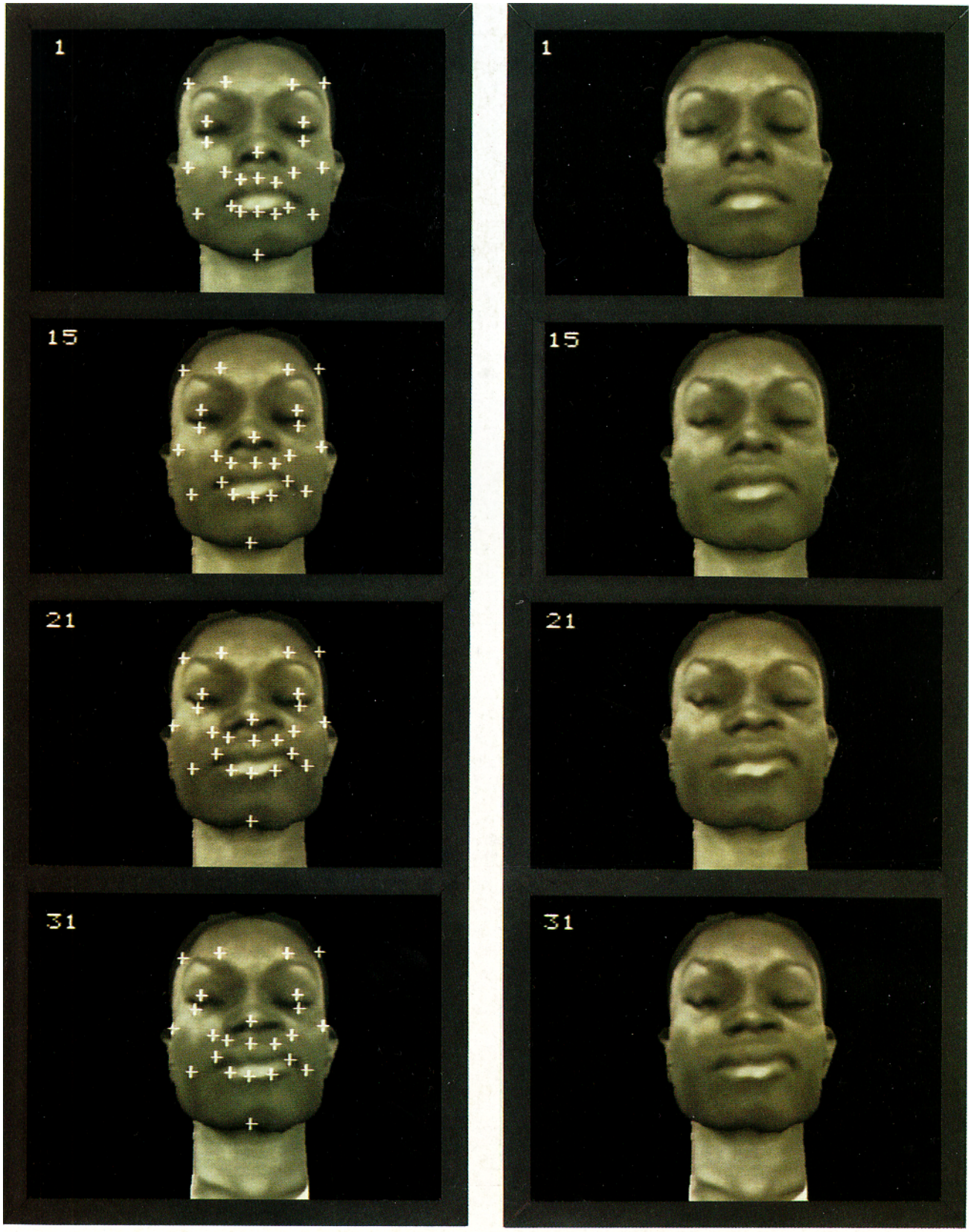
A paper punch was used to make little round adhesive spots from Scotchlite tape; figure 14 shows the author applying spots to his face (the brightness of the image has been scaled so that the face is visible; originally, only the spots could be seen). A typical frame (once again, brightened) is illustrated in figure 15.

The problem of digitizing the actor's performance now becomes one of tracking a set of bright spots in a dark field. The special case of tracking spots on a face which faces the camera is particularly favorable; the spots (if placed wisely) will never actually touch one another, and can never be obscured by another part of the face. For the test animation, the head was held relatively still, and only the facial expression changed. The algorithm used was to have a human operator indicate the position of each spot in the initial frame of a digitized sequence. This takes care of the correspondence between each spot and the basis function on the face that it controls. Spots are tracked automatically in subsequent frames. The spot tracking routine outputs the X, Y coordinates of the spot centers in each frame.

When the operator selects a spot, it is a matter of touching the spot, or the spot's near vicinity. A window about the point indicated is scanned, and the X, Y coordinates of each pixel are multiplied by the intensity of the pixel; running sums of the X and Y pixel intensity products are saved. A running sum is also kept of the pixel intensities. When the window is completely scanned, the summed X, Y's are each divided by the summed intensities. This supplies the window's "center of gravity" or first moment, a fairly robust estimate of the center of light intensity in the window, which has fractional pixel precision if the spot falls across a reasonable number of pixels. A new window is then scanned from the computed center, and the process iterated a few times. The window size should be slightly larger than the spots, so that the iterated result will be the first moment of the spot itself. In figure 16, tracking crosses have been superimposed on the spots. Note that crosses appear even where the spots they mark are very dim; the spot tracker proved robust, and the motion of the tracking crosses is quite convincing even for spots of marginal



Figures 14-17. Tracking spots on performer's face.



Figures 18-25. Tracked expressions applied to the model face.

visibility. Figure 17 shows the tracking crosses without the spots. Figures 18-25 show the model face being driven by the tracked spots, marked by the tracking crosses in one of each pair of images.

CONCLUSIONS AND FUTURE WORK

The appearance of the model head is quite realistic, and the test animation is very striking. Although short, the sequence exhibits some lifelike twitches and secondary motions which would be unlikely to arise in pure animation. The fundamental idea of mapping the motions of a live performer to a computer animated character promises to be a rich one. Previous efforts to animate faces by interpolating between various canonical expressions can now be supplemented by interpolation of canonical motion sequences. Driving a face or head of very different proportions or physiognomy should be attempted soon.

The animation of the face is very much a "proof of concept," not a completely realized system. At present, the face cannot open its eyes or mouth, and this portion of the model will demand a great deal of time. Some ripples on the eyes (visible in the last two figures) result from the fact that the performer could blink his eyes (in fact, found it hard not to) and the model could not. The test did establish that eyeblinks are quite trackable!

For reasons of efficiency, and to test the validity of the simplifications, the performer's face was tracked in 2D, and the result "projected" to the model, which was animated in the cylindrical coordinates of the range data before being converted into X, Y, Z meshes with normals. The changes in the performer's expressions were transferred to the model with appropriate scaling of the offsets, and a straight projection onto the face approximates projection onto a cylinder. A test setup with angled mirrors at the sides of the actor's head showed that 3D coordinates could be acquired with a single camera. A more fully-realized system would track expressions in 3D and apply them in Cartesian, rather than cylindrical, coordinates.

ACKNOWLEDGEMENTS

My profuse thanks go to Annette White, the model; to Nancy Burson and David Kramlich, my initial collaborators; to Ariel Shaw and Andrew Davidhazy, for extraordinary photographic assistance; to Wyndham Hannaway and Bill Bishop of G.W. Hannaway & Associates, who made available much of the computer time, utility software, and exotic paraphernalia this research required; to my manager, Mark Cutter, and to Ned Greene, Pete Litwinowicz, and Libby Patterson of Apple's Advanced Technology Animation Group, for constant support, advice, and inspiration.

REFERENCES

[1] Walters, Graham, The Story of Waldo C. Graphic. ACM SIGGRAPH '89 Course Notes, *3D Character Animation by Computer*, August 1989.

[2] Parke, Frederick I., A Parametric Model for Human Faces. Ph.D. dissertation, Department of Computer Science, University of Utah, 1974.

[3] Badler, Norman, and Platt, Stephen, Animating Facial Expressions. Proceedings of SIGGRAPH '81 (Dallas, Texas, August 3-7, 1981). In *Computer Graphics 15*, 3, (August 1981), 245-252.

[4] Platt, Stephen Michael, A Structural Model of the Human Face. Ph.D. Department of Computer and Information Science, School of Engineering and Applied Science, University of Pennsylvania, Philadelphia, PA., 1986.

[5] Brennan, Susan Elise, Caricature Generator. M.S. Visual Studies, Dept. of Architecture, Massachusetts Institute of Technology, Cambridge, MA. Sept. 1982.

[6] Burson, Nancy, and Schneider, Thomas, "Method and Apparatus for Producing an Image of a Person's Face at a Different Age," U.S. Patent #4276570, June 30, 1981.

[7] Oka, Masaaki, Tsutsui, Kyoya, Ohba, Akio, Kurauchi, Yoshitaka, Tago, Takashi, Real-Time Manipulation of Texture-Mapped Surfaces. Proceedings of SIGGRAPH '87 (Anaheim, California, July 27-31, 1987). In *Computer Graphics 21*, 4, (July 1987), 181-188.

[8] Lachapelle, Pierre, Bergeron, Philippe, Robidoux, P., and Langlois, Daniel, *Tony de Peltrie*. [film] 1985.

[9] Waters, Keith , A Muscle Model for Animating Three-Dimensional Facial Expression. Proceedings of SIGGRAPH '87 (Anaheim, California, July 27-31, 1987). In *Computer Graphics 21*, 4 (July 1987), 17-24.

[10] Lasseter, John, Ostby, Eben, Reeves, William, Good, Craig, Rydstrom, Gary. *Tin Toy*. [film] Pixar, 1988.

[11] Cyberware Laboratory, Inc.: 4020/PS 3D Scanner, 4020/RGB 3D Scanner with color digitizer. 8 Harris Court 3D, Monterey, California 93940.

[12] Burt, P.J., Ogden, J.M., Adelson, E.H., and Bergen, J.R., Pyramid-Based Computer Graphics. *RCA Engineer*, Vol. 30, 5, Sept.-Oct. 1985.

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

Learning Controls for Blend Shape Based Realistic Facial Animation

Pushkar Joshi[1†], Wen C. Tien[1], Mathieu Desbrun[1] and Frédéric Pighin[2]

1 University of Southern Califronia, Computer Science Department
2 Institute for Creative Technologies, University of Southern California

Abstract

Blend shape animation is the method of choice for keyframe facial animation: a set of blend shapes (key facial expressions) are used to define a linear space of facial expressions. However, in order to capture a significant range of complexity of human expressions, blend shapes need to be segmented into smaller regions where key idiosyncracies of the face being animated are present. Performing this segmentation by hand requires skill and a lot of time. In this paper, we propose an automatic, physically-motivated segmentation that learns the controls and parameters directly from the set of blend shapes. We show the usefulness and efficiency of this technique for both, motion-capture animation and keyframing. We also provide a rendering algorithm to enhance the visual realism of a blend shape model.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Animation

1. Introduction

The human face has always held a particular interest for the computer graphics community: its complexity is a constant challenge to our increasing ability to model, render, and animate lifelike synthetic objects. Facial animation requires a *deformable* model of the face to express the wide range of facial configurations related to speech or emotions. There are two traditional ways of creating deformable face models: using a physically-based model or a blend shape model. A physically-based model generally simulates various skin layers, muscles, fatty tissues, bones, and all the necessary components to approximate the real facial mechanics. A blend shape model, however, mostly disregards the mechanics; instead, it directly considers every facial expression as a linear combination of a few select facial expressions, the blend shapes. By varying the weights of the linear combination, a full range of facial expressions can be expressed with very little computation.

Nowadays, there are several options for creating blend shapes. A skilled digital artist can deform a base mesh into the different canonical shapes needed to cover the desired range of expressions. Alternatively, the blend shapes can be directly scanned by a range scanner from a real actor or a clay model. With this last technique, the scanned data needs to be registered in order to produce blend shapes that share a common topology [9] and can therefore be combined correctly.

To express a significant range of highly detailed expressions, digital animators often have to create large libraries of blend shapes. In this case a naive parameterization of the face model, one that would give a parameter for each blend shape, is not practical. In particular, the visual impact of changing the contribution of a blend shape might be diffcult to predict, leading to a tedious trial and error process for the user. Splitting the face geometry in several regions that can be specified individually somewhat alleviates this problem. By manipulating a smaller area the user is guaranteed that the modification will impact only a specific part of the face (e.g., the left eyebrow). However, segmenting the face manually is difficult and tedious. The segmentation should reflect the idiosyncracies of the face being modeled and provide editing and different level of details. In general, finding the right parameters and control knobs in a blend shape model is no simple task, and it often leverages an understanding of the mechanical structure of the face. In this paper we address

† e-mail: ppj@usc.edu

the problem of parameterization and control of blendshape models.

1.1. Related Work

Blend shape interpolation can be traced back to Parke's pioneering work in facial animation [12, 13]. This initial work has found many applications both, in computer graphics and in computer vision.

Parke's original idea was rapidly extended to a segmented face where the regions are blended individually [8], allowing a wider range of expressions. Traditionally these regions are defined *manually*. A prototypical example is the segmentation of a face into an upper region and a lower region: the upper region is used for expressing emotions, while the lower region expresses speech. Although this approximation is often used in practice, such an ad hoc separation does not reflect the subtle interdependencies appearing in reality.

Blend shape models have also found their way in the computer vision community where they help analyze face images and video. Blanz and Vetter [1] designed an algorithm that fits a blend shape model onto a single image. Their result is an estimate of the geometry and texture of the person's face. Pighin et al [15] extended this work by fitting their model to a whole sequence of images, allowing manipulation of the video by editing the fitted model throughout the video sequence.

There has been little research on interactive blend shape model manipulation with the exception of the work by Pighin et al [14]. They describe a keyframe animation system that uses a palette of facial expressions along with a painting interface to assign blending weights. The system gives the animator the freedom to assign the blending weights at the granularity of a vertex. This freedom is, in practice, a drawback: by not taking into account the physical limitations of the face, it is rather difficult to create realistic expressions. The system we propose is quite different: it does respect the mechanics of the face through an analysis of the physical properties of the data. In comparison, our system is more intuitive and helps generate plausible facial expressions. Choe et al [3] have done some interesting work on mapping motion onto a set of blend shapes, but where they build a segmentation of the face manually, we learn it from the data.

Segmentation is a very active topic in image processing and computer vision [7]. However, the problem we are addressing is very different from image or optical flow segmentation; our goal is to segment a 3D mesh that is a linear combination of sample meshes. Similarly, subdivision surfaces have become a popular representation for three-dimensional objects [16]. The goals of subdivision schemes as researched so far in the compter graphics community do not match that of this paper: the segmentation of a linear space of meshes.

1.2. Contribution and Overview

In this paper, we address the problems of *parameterization* and *control* of blend shape models. We design an *automatic technique* that extracts a set of parameters from a blend shape model. Instead of deriving our control mechanism from the biomechanics of the face, we learn it directly from the available data. This solution is thus specific to the processed blend shapes, and reflects the facial idiosyncracies present in data. We also demonstrate the usefulness of these parameters through two animation techniques: motion capture and keyframing. Finally, we propose a new rendering algorithm for blend shape models; one that addresses the problem of texture misregistration across blend shape textures.

We will describe our work by starting in section 2 with some definitions and notations. Section 3 and section 4 are then dedicated to the use of the model in motion capture animation and keyframing respectively. We also explain how some of the blur artifacts can be avoided while rendering the blend shape model in section 5. We finally conclude with a discussion of our results and ideas for future research.

2. Blend Shape Face Model

Setup We define a blend shape face model as being a convex linear combination of n basis vectors, each vector being one of the blend shapes. Each blend shape is a face model that includes geometry and texture. All the blend shape meshes for a given model share the same topology. The coordinates of a vertex $\boldsymbol{V}$ belonging to the blend shape model can then be written as follows:

$$\boldsymbol{V} = \sum_{i=1}^{n} \alpha_i \, \boldsymbol{V}_i$$

where the scalars α_i are the blending weights, $\boldsymbol{V}_i$ is the location of the vertex in the blend shape i, and n is the number of blend shapes. These weights must satisfy the convex constraint:

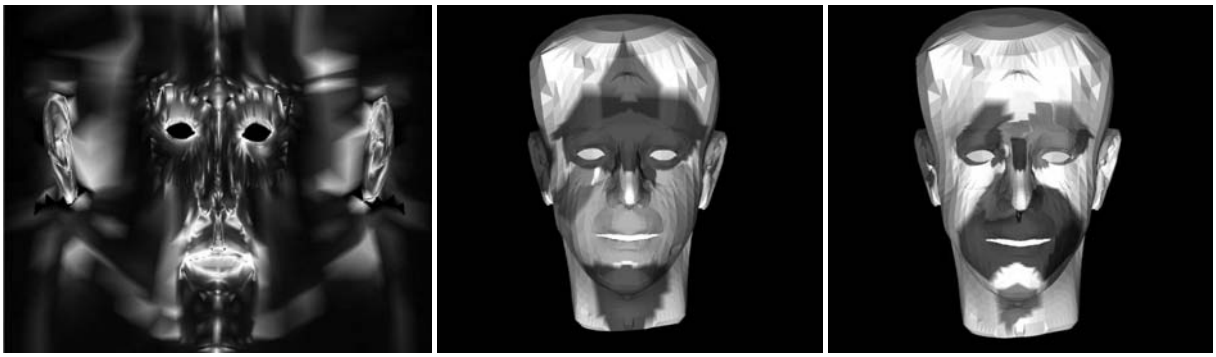
$$\alpha_i \geq 0, \text{ for all } i$$

and must sum to one for rotational and translational invariance:

$$\sum_{i=1}^{n} \alpha_i = 1$$

Similarly, the texture at a particular point of the blend shape model is a linear combination (i.e., alpha blending) of the blend shape textures with the same blending weights as those used for the geometry.

Learning Controls Spanning a complete range of facial expressions might require a large number of blend shapes. For instance, the facial animations of Gollum in the feature film



(a) (b) (c)

Figure 1: *Automatically generated regions: (a) Deformation map (the deformation in X, Y and Z directions is expressed as a respective RGB triplet) (b) Segmentation for a low threshold (c) and for a high threshold.*

The Two Towers required 675 blend shapes [6]. However, studies [5] have shown that it is possible to create complex and believable facial expressions using only a few blend shapes by combining smaller, local shapes. For instance, the face geometry can be split in three areas, one covering the mouth and the jaws, another covering the eyes, and the last one the eyebrows. If the regions can be manipulated independently, the number of possible combinations (and therefore the number of possible expressions) increases significantly. Although one can define the regions manually, it requires considerable skill and time, and needs to be performed each time a new character is animated. Instead, we propose a simple, automatic and fast (less than a minute for a typical blend shape model) segmentation process that leverages face deformation information directly from the input data to create meaningful blend regions.

Physical Model One of the simplest physical models for deformable objects is that of *linear elasticity*. The deformation of an object is measured by the displacement field $\boldsymbol{d}$ between each point's current position and its rest position. As explained, for instance, in Debunne et al [4], the governing equation of motion of a linear elastic model is the Lamé formulation:

$$\rho \boldsymbol{a} = \lambda \Delta \boldsymbol{d} + (\lambda + \mu) \nabla (\nabla \cdot \boldsymbol{d}) \quad (1)$$

In our current context, $\boldsymbol{d}$ is the displacement of the vertex from its position on the neutral face, ρ is the averaged face mass density, $\boldsymbol{a}$ is the vertex' acceleration, and λ and μ are the Lamé' coefficients that determine the material's behavior (related to Young's modulus and Poisson ratio). The interpretation of the previous equation is relatively simple: the laplacian vector $\Delta \boldsymbol{d}$ of the displacement field represents the propagation of deformation through the blend shape, while the second term represents the area-restoring force. These two second-order operators, null for any rigid deformation, are therefore *two complementary measures of deformation* of our face model. To further simplify our model, we will assume that the area distortion is negligible on a face (our tests confirm that this assumption does not change the results significantly); therefore, we only use the laplacian to segment the face into disjoint regions of similar amount of deformation, as explained next.

Segmentation Debunne et al [4] have introduced a simple discrete evaluation of the laplacian operator present in Eq. 1. We compute this discrete laplacian value at every vertex of every non-neutral (i.e., expressive) blend shape, and take the magnitude of the resulting vectors. This provides us with a deformation map for each expression. We gather these maps into a single deformation map M by computing for each vertex independently its maximum deformation value across all expressions. This resulting map (see Figure 3(a) - expressed as a vector map to show direction of deformation) measures the maximum amount of *local deformation* that our face model has for the blend shapes used. A fast segmentation can now be performed by simply splitting this map in the regions with low deformation, and those with high deformation. The threshold for this split can be chosen as:

$$threshold = \boldsymbol{D}[n\,t]$$

where $\boldsymbol{D}$ is the array of sorted deformation values, n is the size of this array and t is a scalar between 0 and 1.

That is, first sort all the deformation values, and then obtain the deformation at the position that is a function of the number of values. For instance, to generate the regions in Figure 3(b,c), we used $t = 0.25$ and $t = 0.75$ respectively. Depending on the threshold, disconnected regions are created all across the mesh. We automatically clean up the regions by absorbing isolated regions into larger regions and minimizing concavity of the regions. Finally, each region is extended by one vertex all around its boundary, in order to create an overlap with the neighboring regions. The result

is a large, least-deformed region (i.e. the background), and a number of overlapping regions where there is generally more significant deformation in the range of expressions. These latter regions (see Figure 3(b,c)) correspond to vertices that generally undergo similar deformation: locally, each region deforms in a quasi-rigid way. Thus, linear blending in each of these regions will reconstruct much more detail of the target face expression as demonstrated in the next two sections.

3. Animation with Motion Capture

We express the motion in the motion capture data using the blend shape model. That is, we assume that the motion (or the per-frame position) of a motion marker can be expressed as a linear combination of corresponding points in the blend shapes. Namely:

$$\boldsymbol{M}_j = \sum_{i=1}^{n} \alpha_i \, \boldsymbol{V}_{ij}$$

where $\boldsymbol{M}_j$ is a location on the face whose motion was recorded and $\boldsymbol{V}_{ij}$ is the corresponding location in blendshape i. m is the number of motion markers and n the number of blend shapes (as in Choe et. al. [3])

Given several such equations, we find the blending weights α_i. We recast this as a minimization problem, where we need to minimize the sum of the differences:

$$\sum_{j=1}^{m} [\boldsymbol{M}_j - (\sum_{i=1}^{n} \alpha_i \, \boldsymbol{V}_{ij})]^2 \qquad (2)$$

The whole system is a linear system of equations where the unknowns α_i are the weights in the blend shape combination. By using an iterative quadratic programming solver [11], we obtain the optimal values of the blending weights α_i in the least squares sense. Solving this system is equivalent to orthogonally projecting the motion onto the set of blend shapes. In general equation 2 does not have an exact solution, since the motions can be more expressive than what the set of blend shapes allows. To produce an animated mesh that follows the motion more precisely we complement the projection on the blend shape basis by translating the vertices in the mesh by the residual $(\boldsymbol{M}_j - \sum_{i=1}^{n} \alpha_i \cdot \boldsymbol{V}_{ij})$. The residual, which is only known for a small set of points, is interpolated to the rest of the facial mesh using radial basis functions [10]. The final coordinates, $\boldsymbol{V}_j$, of a vertex on the face are then constructed using:

$$\boldsymbol{V}_j = \boldsymbol{P}_j + RBF(\boldsymbol{P}_j)$$

where $\boldsymbol{P}_j$ is the projection on the set of blend shape:

$$\boldsymbol{P}_j = \sum_{i=1}^{n} \alpha_i \, \boldsymbol{V}_{ij}$$

and $RBF(\boldsymbol{P}_j)$ is the interpolated residual at vertex $\boldsymbol{P}_j$:

$$RBF(\boldsymbol{P}_j) = \sum_{i=1}^{m} \exp(-\|\boldsymbol{M}_i - \boldsymbol{P}_j\|) \, \boldsymbol{C}_i \qquad (3)$$

In equation 3 the vectors $\boldsymbol{C}_i$ are computed using the known values of the residual at $\boldsymbol{M}_i$. Since the system of equations is linear in the unknowns, using linear least-squares provides an estimate of the unknowns [14]. Note that only applying the residual would have a different effect; by first projecting on the set of blend shapes we obtain a face geometry that reflects the blend shapes, then we apply the residual which brings the geometry closer to the motion. Choe et al [3]'s approach to mapping motion onto a set of blend shapes is very similar. The main difference is how the residual is taken into account. In their approach the blend shapes are modified to adapt them to the motions. We, on the other hand, use radial basis functions to modify the geometry on a per-frame basis. Their method would probably be more effective for processing a large quantity of motions, whereas ours would perform better on a small dataset.

Instead of solving the above system for the entire model, we solve for each region created using our automatic segmentation process. Doing so gives us localized control over the face mesh and results in better satisfaction of the spatial constraints. This also allows us to express a wide range of motion using only a limited number of blend shapes (ten, in our case).

For every frame and for every region, we construct the above minimization problem and obtain blending weights. The same weights are then used to obtain, for all vertices of the region, new positions that match the motion. Thus, for every frame of motion, we can solve a minimization problem to obtain the blending weights and consequently the face mesh that follows the motion capture data.

4. Keyframe Editing

Using our blend shape model, we can interactively construct face meshes that can be used as keyframes in a keyframing-based facial animation tool.

Creating Keyframes Creating a keyframe is similar to producing a frame in a motion capture sequence in that we need to specify control points (markers), their respective mappings, and spatial constraints (i.e. the positions of the markers). In our interface, the user can interactively specify all the above by clicking and dragging with the mouse on the face model. As in the process used in the motion capture application (see Eq. 2), we construct a minimization problem using the interactively specified constraints and obtain blending weights.

Regions and Region Hierarchy We can segment the blend shape model into regions using our automatic segmentation

Figure 2: *Successive keyframe editing from coarse (left) to fine (right) level of details.*

technique. In order to allow keyframe editing at various levels of detail, we build a hierarchy of regions. This hierarchy is created by first running the segmentation algorithm described in section 2 with a high threshold value so as to generate small and localized regions. These regions constitute the lowermost region level. We can then merge regions iteratively so that contiguous regions are merged together as we generate higher region levels.

Motion Damping Some of the locations on the face do not move significantly throughout the set of blend shapes (e.g. tip of the nose). If we were to select such a location and try to deform it, using the interface describe so far, a small motion of the mouse would trigger a dramatic change in the facial expression. To reduce the sensitivity of the system we scale the displacement of the mouse according to a factor that is inversely proportional to the maximum displacement in the blend shape model at the selected point on the mesh.

Fig. 2 displays a sequence of manipulations performed on a keyframe. The successive keyframe editing is performed with increasing level of details to refine the facial expression in a localized manner.

5. Rendering Realistic Blend Shapes

Basic Process Rendering the blend shape model is pretty straightforward and can be done in two steps: first the consensus geometry is evaluated, and then it is rendered as many times as there are blend shapes in the model to blend the texture maps. This latter step is done by assigning to each vertex' alpha channel the corresponding weight for a given blend shape. To improve our renderings we decided not to blend the texture maps on the parts of the face whose texture should not vary as a function of the facial expression, in particular, in the hair, neck, and ears area. These areas are textured using the texture map of any blend shape (usually one corresponding to the neutral expression).

Realistic Textures Texture misregistration is a common problem with blend shape rendering for realistic facial animation. If the textures do not correspond at each point on the face geometry, combining them linearly will result in a blurred rendering. Thus, the frequency content of the rendered face images varies as a function of time. Figure 4 provides an illustration of this phenomenon. The leftmost image shows our model rendered with only one contributing blend shape. The middle image shows the rendered model with seven equally contributing blend shapes. In the middle rendering a lot of the details of the face texture have disappeared.

To alleviate this problem, we borrow an approach from the image processing community [2]: we base our blending on a band-pass decomposition of the textures. More specifically, we build a two level laplacian image pyramid out of each blend shape texture map. This results in the creation of two texture maps for each blend shape: the first is a low-pass version of the original texture, and the second is a signed detail texture. We then render the blend shape model as follow: we first render the lowpass texture maps and blend them together. Then we render the detail texture map of a single blend shape using the consensus geometry and add it to the previous rendering. The result is a rendering that both, better preserves the original spectral content of the blend shape textures and maintains the high frequency content constant throughout the animation. The rightmost rendering in figure 4 illustrates the improvement obtained by using this technique.

6. Results

We demonstrate the techniques described in this paper with a set of blend shapes modeled to capture the facial expression of an actor. We created ten blend shapes corresponding to extreme expressions. We used an image-based modeling technique similar to the one developed by Pighin et al [14]. Three photographs of the actor were processed to model each blend shape: front facing, 30 degree right, and 30 degree left. All the animations shown in the video were computed and rendered in real-time (30Hz) on a 1GhZ PC equipped with an NVidia GeForce 3 graphics card. We decided to animate the tongue, the lower teeth and the upper teeth in a simple proce-

dural manner; they are moved rigidly and follow the motion of separate sets of manually selected points on the mesh. The eyeballs are moved rigidly according to the rigid motion of the head.

Motion Capture As described in section 3, we can project recorded motion onto the blend shape model. The accompanying video includes a few animated sequences that demonstrate this technique. The deformations of the face are very natural and reflect the actor's personality. Fig. 5 shows some of the frames obtained. The example shown uses only 10 blend shapes. To animate speech motion usually a much larger set of shapes needs to be used. We are able to animate the lips by using radial basis functions as described in section 3.

Keyframe Editing Also included in the video is a demonstration of the interactive tool described in section 4. The tool allows us to sculpt the face in a very intuitive way. We start manipulating the face with a set of coarse regions and refine the expression by using increasingly finer segmentations.

7. Future Work

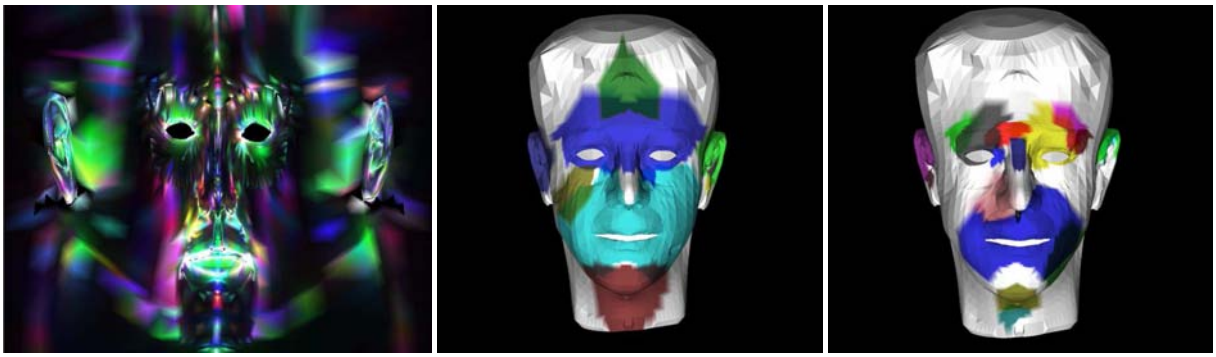
We would like to improve our results in different ways. In particular, we feel our rendering algorithm would benefit from a more principled frequency analysis of the blend shapes texture maps. Using a feature preserving filter to separate the high frequency data might lead to better results. It would be interesting to try this technique on a non-human character; one for which segmenting the face might be more challenging and non-intuitive. We would also like to test our technique on a larger dataset of blend shapes. Finally, our segmentation technique only takes into account geometric information. We would like to extend it to also take advantage of the texture information.

Acknowledgements

The authors would like to thank J.P. Lewis for discussions about blend shape animation and Andrew Gardner for its initial development. This project was supported in part by National Science Foundation (CCR-0133983, DMS- 0221666, DMS-0221669, EEC-9529152) and the U.S. Army Research Institute for the Behavioral and Social Sciences under ARO contract number DAAD 19-99-D-0046. Any opinions, findings, conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the Department of the Army.

References

1. T. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. In *SIGGRAPH 99 Conference Proceedings*. ACM SIGGRAPH, August 1999.
2. P.J. Burt and E.H. Adelson. A multiresolution spline with application to image mosaics. *ACM Transaction on Graphics*, 2(4), October 1983.
3. B. Choe, H. Lee, and H. Ko. Performance-driven muscle-based facial animation. In *Proceedings of Computer Animation*, volume 12, pages 67–79, May 2001.
4. G. Debunne, M. Desbrun, M. Cani, and A. Barr. Adaptive simulation of soft bodies in real-time. In *Proceedings of Computer Animation 2000*, pages 15–20, May 2000.
5. P. Ekman and W.V. Friesen. *Unmasking the face. A guide to recognizing emotions fron facial clues*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975.
6. J. Fordham. Middle earth strikes back. *Cinefex*, (92):71–142, 2003.
7. R.M. Haralick. Image segmentation survey. *Fundamentals in Computer Vision*, 1983.
8. J. Kleiser. A fast, efficient, accurate way to represent the human face. In *SIGGRAPH '89 Course Notes 22: State of the Art in Facial Animation*, 1989.
9. A. Lee, D. Dobkin, W. Sweldens, and P. Schröder. Multiresolution mesh morphing. In *Proceedings of SIGGRAPH 99*, pages 343–350, August 1999.
10. G.M. Nielson. Scattered data modeling. *IEEE Computer Graphics and Applications*, 13(1):60–70, January 1993.
11. J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer, New York, 1999.
12. F.I. Parke. Computer generated animation of faces. *Proceedings ACM annual conference.*, August 1972.
13. F.I. Parke. *A parametric model for human faces*. PhD thesis, University of Utah, Salt Lake City, Utah, December 1974. UTEC-CSc-75-047.
14. F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D.H. Salesin. Synthesizing realistic facial expressions from photographs. In *SIGGRAPH 98 Conference Proceedings*, pages 75–84. ACM SIGGRAPH, July 1998.
15. F. Pighin, R. Szeliski, and D.H. Salesin. Resynthesizing facial animation through 3d model-based tracking. In *Proceedings, International Conference on Computer Vision*, 1999.
16. D. Zorin, P. Schröder, A. DeRose, L. Kobbelt, A. Levin, and W. Sweldens. Subdivision for modeling and animation. In *SIGGRAPH 2000 Course Notes*. ACM SIGGRAPH, May 2000.



(a) (b) (c)

Figure 3: *Automatically generated regions: (a) Deformation map (the deformation in X, Y and Z directions is expressed as a respective RGB triplet) (b) Segmentation for a low threshold (c) and for a high threshold.*

(a) (b) (c)

Figure 4: *Blend shape renderings (a) a single contributing blend shape (b) seven equally contributing blend shapes without detail texture (c) seven equally contributing blend shapes with detail texture*

Figure 5: *Mapping motion capture data on a set of blend shapes*

Making Faces

Brian Guenter† Cindy Grimm† Daniel Wood‡
Henrique Malvar† Fredrick Pighin‡
†Microsoft Corporation ‡University of Washington

ABSTRACT

We have created a system for capturing both the three-dimensional geometry and color and shading information for human facial expressions. We use this data to reconstruct photorealistic, 3D animations of the captured expressions. The system uses a large set of sampling points on the face to accurately track the three dimensional deformations of the face. Simultaneously with the tracking of the geometric data, we capture multiple high resolution, registered video images of the face. These images are used to create a texture map sequence for a three dimensional polygonal face model which can then be rendered on standard 3D graphics hardware. The resulting facial animation is surprisingly life-like and looks very much like the original live performance. Separating the capture of the geometry from the texture images eliminates much of the variance in the image data due to motion, which increases compression ratios. Although the primary emphasis of our work is not compression we have investigated the use of a novel method to compress the geometric data based on principal components analysis. The texture sequence is compressed using an MPEG4 video codec. Animations reconstructed from 512x512 pixel textures look good at data rates as low as 240 Kbits per second.

CR Categories: I.3.7 [Computer Graphics]: Three Dimensional Graphics and Realism: Animation; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling

1 Introduction

One of the most elusive goals in computer animation has been the realistic animation of the human face. Possessed of many degrees of freedom and capable of deforming in many ways the face has been difficult to simulate accurately enough to convince the average person that a piece of computer animation is actually an image of a real person.

We have created a system for capturing human facial expression and replaying it as a highly realistic 3D "talking head" consisting of a deformable 3D polygonal face model with a changing texture map. The process begins with video of a live actor's face, recorded from multiple camera positions simultaneously. Fluorescent colored 1/8" circular paper fiducials are glued on the actor's face and their 3D position reconstructed over time as the actor talks and emotes. The 3D fiducial positions are used to distort a 3D polygonal face model in mimicry of the distortions of the real face. The fiducials are removed using image processing techniques and the video streams from the multiple cameras are merged into a single texture map. When the resulting fiducial-free texture map is applied to the 3D reconstructed face mesh the result is a remarkably life-like 3D animation of facial expression. Both the time varying texture created from the video streams and the accurate reproduction of the 3D face structure contribute to the believability of the resulting animation.

Our system differs from much previous work in facial animation, such as that of Lee [10], Waters [14], and Cassel [3], in that we are not synthesizing animations using a physical or procedural model of the face. Instead, we capture facial movements in three dimensions and then replay them. The systems of [10], [14] are designed to make it relatively easy to animate facial expression manually. The system of [3] is designed to automatically create a dialog rather than faithfully reconstruct a particular person's facial expression. The work of Williams [15] is most similar to ours except that he used a single static texture image of a real person's face and tracked points only in 2D. The work of Bregler et al [2] is somewhat less related. They use speech recognition to locate visemes[1] in a video of a person talking and then synthesize new video, based on the original video sequence, for the mouth and jaw region of the face to correspond with synthetic utterances. They do not create a three dimensional face model nor do they vary the expression on the remainder of the face. Since we are only concerned with capturing and reconstructing facial performances out work is unlike that of [5] which attempts to recognize expressions or that of [4] which can track only a limited set of facial expressions.

An obvious application of this new method is the creation of believable virtual characters for movies and television. Another application is the construction of a flexible type of video compression. Facial expression can be captured in a studio, delivered via CDROM or the internet to a user, and then reconstructed in real time on a user's computer in a virtual 3D environment. The user can select any arbitrary position for the face, any virtual camera viewpoint, and render the result at any size.

One might think the second application would be difficult to achieve because of the huge amount of video data required for the time varying texture map. However, since our system generates accurate 3D deformation information, the texture image data is precisely registered from frame to frame. This reduces most of the variation in image intensity due to geometric motion, leaving primarily shading and self shadowing effects. These effects tend to be of low spatial frequency and can be compressed very efficiently. The compressed animation looks good at data rates of 240 kbits per second for texture image sizes of 512x512 pixels, updating at 30 frames per second.

The main contributions of the paper are a method for robustly capturing both a 3D deformation model and a registered texture image sequence from video data. The resulting geometric and texture data can be compressed, with little loss of fidelity, so that storage

[1] Visemes are the visual analog of phonemes.

Figure 1: The six camera views of our actress' face.

requirements are reasonable for many applications.

Section 2 of the paper explains the data capture stage of the process. Section 3 describes the fiducial correspondence algorithm. In Section 4 we discuss capturing and moving the mesh. Sections 5 and 6 describe the process for making the texture maps. Section 7 of the paper describes the algorithm for compressing the geometric data.

2 Data Capture

We used six studio quality video cameras arranged in the pattern shown in Plate 1 to capture the video data. The cameras were synchronized and the data saved digitally. Each of the six cameras was individually calibrated to determine its intrinsic and extrinsic parameters and to correct for lens distortion. The details of the calibration process are not germane to this paper but the interested reader can find a good overview of the topic in [6] as well as an extensive bibliography.

We glued 182 dots of six different colors onto the actress' face. The dots were arranged so that dots of the same color were as far apart as possible from each other and followed the contours of the face. This made the task of determining frame to frame dot correspondence (described in Section 3.3) much easier. The dot pattern was chosen to follow the contours of the face (i.e., outlining the eyes, lips, and nasio-labial furrows), although the manual application of the dots made it difficult to follow the pattern exactly.

The actress' head was kept relatively immobile using a padded foam box; this reduced rigid body motions and ensured that the actress' face stayed centered in the video images. Note that rigid body motions can be captured later using a 3D motion tracker, if desired.

The actress was illuminated with a combination of visible and near UV light. Because the dots were painted with fluorescent pigments the UV illumination increased the brightness of the dots significantly and moved them further away in color space from the colors of the face than they would ordinarily be. This made them easier to track reliably. Before the video shoot the actress' face was digitized using a cyberware scanner. This scan was used to create the base 3D face mesh which was then distorted using the positions of the tracked dots.

3 Dot Labeling

The fiducials are used to generate a set of 3D points which act as control points to warp the cyberware scan mesh of the actress' head. They are also used to establish a stable mapping for the textures generated from each of the six camera views. This requires that each dot have a unique and consistent label over time so that it is associated with a consistent set of mesh vertices.

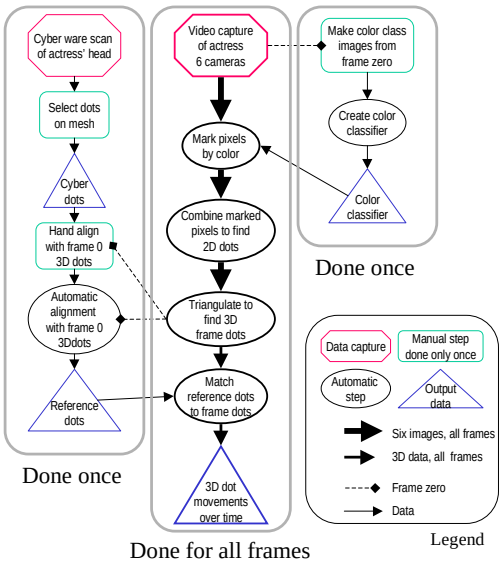


Figure 2: The sequence of operations needed to produce the labeled 3D dot movements over time.

The dot labeling begins by first locating (for each camera view) connected components of pixels which correspond to the fiducials. The 2D location for each dot is computed by finding the two dimensional centroid of each connected component. Correspondence between 2D dots in different camera views is established and potential 3D locations of dots reconstructed by triangulation. We construct a reference set of dots and pair up this reference set with the 3D locations in each frame. This gives a unique labeling for the dots that is maintained throughout the video sequence.

A flowchart of the dot labeling process is shown in Figure 2. The left side of the flowchart is described in Section 3.3.1, the middle in Sections 3.1, 3.2, and 3.3.2, and the right side in Section 3.1.1.

3.1 Two-dimensional dot location

For each camera view the 2D coordinates of the centroid of each colored fiducial must be computed. There are three steps to this process: color classification, connected color component generation, and centroid computation.

First, each pixel is classified as belonging to one of the six dot colors or to the background. Then depth first search is used to locate connected blobs of similarly colored pixels. Each connected colored blob is grown by one pixel to create a mask used to mark those pixels to be included in the centroid computation. This process is illustrated in Figure 4.

The classifier requires the manual marking of the fiducials for one frame for each of the six cameras. From this data a robust color classifier is created (exact details are discussed in Section 3.1.1). Although the training set was created using a single frame of a 3330 frame sequence, the fiducial colors are reliably labeled throughout the sequence. False positives are quite rare, with one major exception, and are almost always isolated pixels or two pixel clusters. The majority of exceptions arise because the highlights on the teeth and mouth match the color of the white fiducial training set. Fortunately, the incorrect white fiducial labelings occur at consistent 3D locations and are easily eliminated in the 3D dot processing stage.

The classifier generalizes well so that even fairly dramatic changes

in fiducial color over time do not result in incorrect classification. For example, Figure 5(b) shows the same green fiducial in two different frames. This fiducial is correctly classified as green in both frames.

The next step, finding connected color components, is complicated by the fact that the video is interlaced. There is significant field to field movement, especially around the lips and jaw, sometimes great enough so that there is no spatial overlap at all between the pixels of a fiducial in one field and the pixels of the same fiducial in the next field. If the two fields are treated as a single frame then a single fiducial can be fragmented, sometimes into many pieces.

One could just find connected color components in each field and use these to compute the 2D dot locations. Unfortunately, this does not work well because the fiducials often deform and are sometimes partially occluded. Therefore, the threshold for the number of pixels needed to classify a group of pixels as a fiducial has to be set very low. In our implementation any connected component which has more than three pixels is classified as a fiducial rather than noise. If just the connected pixels in a single field are counted then the threshold would have to be reduced to one pixel. This would cause many false fiducial classifications because there are typically a few 1 pixel false color classifications per frame and 2 or 3 pixel false clusters occur occasionally. Instead, we find connected components and generate lists of potential 2D dots in each field. Each potential 2D dot in field one is then paired with the closest 2D potential dot in field two. Because fiducials of the same color are spaced far apart, and because the field to field movement is not very large, the closest potential 2D dot is virtually guaranteed to be the correct match. If the sum of the pixels in the two potential 2D dots is greater than three pixels then the connected components of the two 2D potential dots are merged, and the resulting connected component is marked as a 2D dot.

The next step is to find the centroid of the connected components marked as 2D dots in the previous step. A two dimensional gradient magnitude image is computed by passing a one dimensional first derivative of Gaussian along the x and y directions and then taking the magnitude of these two values at each pixel. The centroid of the colored blob is computed by taking a weighted sum of positions of the pixel (x, y) coordinates which lie inside the gradient mask, where the weights are equal to the gradient magnitude.

3.1.1 Training the color classifier

We create one color classifier for each of the camera views, since the lighting can vary greatly between cameras. In the following discussion we build the classifier for a single camera.

The data for the color classifier is created by manually marking the pixels of frame zero that belong to a particular fiducial color. This is repeated for each of the six colors. The marked data is stored as 6 *color class images*, each of which is created from the original camera image by setting all of the pixels *not* marked as the given color to black (we use black as an out-of-class label because pure black never occurred in any of our images). A typical color class image for the yellow dots is shown in Figure 3. We generated the color class images using the "magic wand" tool available in many image editing programs.

A seventh color class image is automatically created for the background color (e.g., skin and hair) by labeling as out-of-class any pixel in the image which was previously marked as a fiducial in any of the fiducial color class images. This produces an image of the face with black holes where the fiducials were.

The color classifier is a discrete approximation to a nearest neighbor classifier [12]. In a nearest neighbor classifier the item

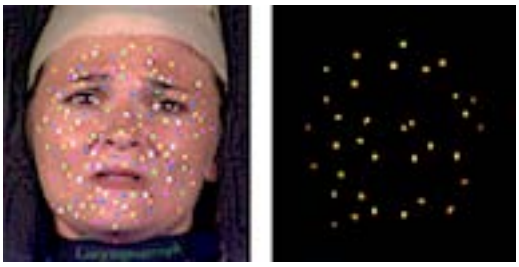


Figure 3: An image of the actress's face. A typical training set for the yellow dots, selected from the image on the left.

to be classified is given the label of the closest item in the training set, which in our case is the color data contained in the color class images. Because we have 3 dimensional data we can approximate the nearest neighbor classifier by subdividing the RGB cube uniformly into voxels, and assigning class labels to each RGB voxel. To classify a new color you quantize its RGB values and then index into the cube to extract the label.

To create the color classifier we use the color class images to assign color classes to each voxel. Assume that the color class image for color class C_i has n distinct colors, $c_1 ... c_n$. Each of the voxels corresponding to the color c_j is labeled with the color class C_i. Once the voxels for all of the known colors are labeled, the remaining unlabeled voxels are assigned labels by searching through all of the colors in each color class C_i and finding the color closest to p in RGB space. The color p is given the label of the color class containing the nearest color. Nearness in our case is the Euclidean distance between the two points in RGB space.

If colors from different color classes map to the same sub-cube, we label that sub-cube with the background label since it is more important to avoid incorrect dot labeling than it is to try to label every dot pixel. For the results shown in this paper we quantized the RGB color cube into a 32x32x32 lattice.

3.2 Camera to camera dot correspondence and 3D reconstruction

In order to capture good images of both the front and the sides of the face the cameras were spaced far apart. Because there are such extreme changes in perspective between the different camera views, the projected images of the colored fiducials are very different. Figure 5 shows some examples of the changes in fiducial shape and color between camera views. Establishing fiducial correspondence between camera views by using image matching techniques such as optical flow or template matching would be difficult and likely to generate incorrect matches. In addition, most of the camera views will only see a fraction of the fiducials so the correspondence has to be robust enough to cope with occlusion of fiducials in some of the camera views. With the large number of fiducials we have placed on the face false matches are also quite likely and these must be detected and removed. We used ray tracing in combination with a RANSAC [7] like algorithm to establish fiducial correspondence and to compute accurate 3D dot positions. This algorithm is robust to occlusion and to false matches as well.

First, all potential point correspondences between cameras are generated. If there are k cameras, and n 2D dots in each camera view then $\left(\begin{array}{c} k \\ 2 \end{array} \right) n^2$ point correspondences will be tested. Each correspondence gives rise to a 3D candidate point defined as the closest point of intersection of rays cast from the 2D dots in the

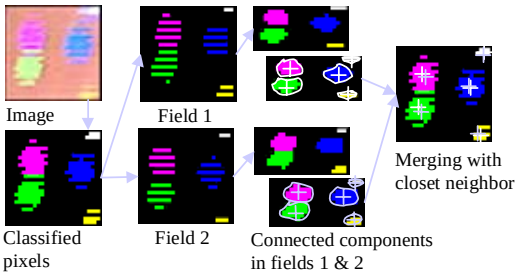


Figure 4: Finding the 2D dots in the images.

two camera views. The 3D candidate point is projected into each of the two camera views used to generate it. If the projection is further than a user-defined epsilon, in our case two pixels, from the centroid of either 2D point then the point is discarded as a potential 3D point candidate. All the 3D candidate points which remain are added to the 3D point list.

Each of the points in the 3D point list is projected into a reference camera view which is the camera with the best view of all the fiducials on the face. If the projected point lies within two pixels of the centroid of a 2D dot visible in the reference camera view then it is added to the list of potential 3D candidate positions for that 2D dot. This is the list of potential 3D matches for a given 2D dot.

For each 3D point in the potential 3D match list, $\binom{n}{3}$ possible combinations of three points in the 3D point list are computed and the combination with the smallest variance is chosen as the true 3D position. Then all 3D points which lie within a user defined distance, in our case the sphere subtended by a cone two pixels in radius at the distance of the 3D point, are averaged to generate the final 3D dot position. This 3D dot position is assigned to the corresponding 2D dot in the reference camera view.

This algorithm could clearly be made more efficient because many more 3D candidate points are generated then necessary. One could search for potential camera to camera correspondences only along the epipolar lines and use a variety of space subdivision techniques to find 3D candidate points to test for a given 2D point. However, because the number of fiducials in each color set is small (never more than 40) both steps of this simple and robust algorithm are reasonably fast, taking less than a second to generate the 2D dot correspondences and 3D dot positions for six camera views. The 2D dot correspondence calculation is dominated by the time taken to read in the images of the six camera views and to locate the 2D dots in each view. Consequently, the extra complexity of more efficient stereo matching algorithms does not appear to be justified.

3.3 Frame to frame dot correspondence and labeling

We now have a set of unlabeled 3D dot locations for each frame. We need to assign, across the entire sequence, consistent labels to the 3D dot locations. We do this by defining a reference set of dots D and matching this set to the 3D dot locations given for each frame. We can then describe how the reference dots move over time as follows: Let $d_j \in D$ be the neutral location for the reference dot j. We define the position of d_j at frame i by an offset, i.e.,

$$d_j^i = d_j + \vec{v}_j^i \tag{1}$$

Because there are thousands of frames and 182 dots in our data

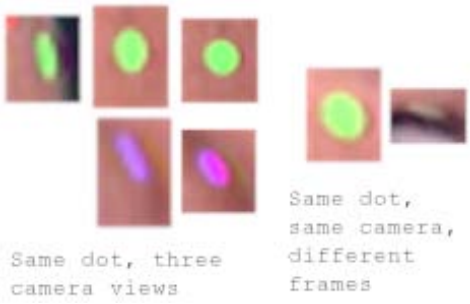


Figure 5: Dot variation. Left: Two dots seen from three different cameras (the purple dot is occluded in one camera's view). Right: A single dot seen from a single camera but in two different frames.

set we would like the correspondence computation to be automatic and quite efficient. To simplify the matching we used a fiducial pattern that separates fiducials of a given color as much as possible so that only a small subset of the unlabeled 3D dots need be checked for a best match. Unfortunately, simple nearest neighbor matching fails for several reasons: some fiducials occasionally disappear, some 3D dots may move more than the average distance between 3D dots of the same color, and occasionally extraneous 3D dots appear, caused by highlights in the eyes or teeth. Fortunately, neighboring fiducials move similarly and we can exploit this fact, modifying the nearest neighbor matching algorithm so that it is still efficient but also robust.

For each frame i we first move the reference dots to the locations found in the previous frame. Next, we find a (possibly incomplete) match between the reference dots and the 3D dot locations for frame i. We then move each matched reference dot to the location of its corresponding 3D dot. If a reference dot does not have a match we "guess" a new location for it by moving it in the same direction as its neighbors. We then perform a final matching step.

3.3.1 Acquiring the reference set of dots

The cyberware scan was taken with the dots glued onto the face. Since the dots are visible in both the geometric and color information of the scan, we can place the reference dots on the cyberware model by manually clicking on the model. We next need to align the reference dots and the model with the 3D dot locations found in frame zero. The coordinate system for the cyberware scan differs from the one used for the 3D dot locations, but only by a rigid body motion plus a uniform scale. We find this transform as follows: we first hand-align the 3D dots from frame zero with the reference dots acquired from the scan, then call the matching routine described in Section 3.3.2 below to find the correspondence between the 3D dot locations, f_i, and the reference dots, d_i. We use the method described in [9] to find the exact transform, T, between the two sets of dots. Finally, we replace the temporary locations of the reference dots with $d_i = f_i$.
and use T^{-1} to transform the cyberware model into the coordinate system of the video 3D dot locations.

3.3.2 The matching routine

The matching routine is run twice per frame. We first perform a conservative match, move the reference dots (as described below in Section 3.3.3), then perform a second, less conservative, match. By moving the reference dots between matches we reduce the problem of large 3D dot position displacements.

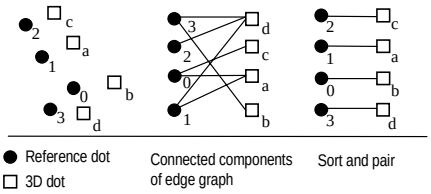


Figure 6: Matching dots.

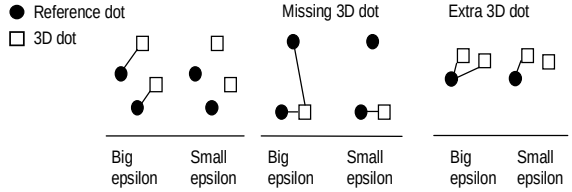


Figure 7: Examples of extra and missing dots and the effect of different values for ϵ.

The matching routine can be thought of as a graph problem where an edge between a reference dot and a frame dot indicates that the dots are potentially paired (see Figure 6). The matching routine proceeds in several steps; first, for each reference dot we add an edge for every 3D dot of the same color that is within a given distance ϵ. We then search for connected components in the graph that have an equal number of 3D and reference dots (most connected components will have exactly two dots, one of each type). We sort the dots in the vertical dimension of the plane of the face and use the resulting ordering to pair up the reference dots with the 3D dot locations (see Figure 6).

In the video sequences we captured, the difference in the 3D dot positions from frame to frame varied from zero to about 1.5 times the average distance separating closest dots. To adjust for this, we run the matching routine with several values of ϵ and pick the run that generates the most matches. Different choices of ϵ produce different results (see Figure 7): if ϵ is too small we may not find matches for 3D dots that have moved a lot. If ϵ is too large then the connected components in the graph will expand to include too many 3D dots. We try approximately five distances ranging from 0.5 to 1.5 of the average distance between closest reference dots.

If we are doing the second match for the frame we add an additional step to locate matches where a dot may be missing (or extra). We take those dots which have not been matched and run the matching routine on them with smaller and smaller ϵ values. This resolves situations such as the one shown on the right of Figure 7.

3.3.3 Moving the dots

We move all of the matched reference dots to their new locations then interpolate the locations for the remaining, unmatched reference dots by using their nearest, matched neighbors. For each reference dot we define a valid set of neighbors using the routine in Section 4.2.1, ignoring the blending values returned by the routine.

To move an unmatched dot d_k we use a combination of the offsets of all of its valid neighbors (refer to Equation 1). Let $n_k \subset D$ be the set of neighbor dots for dot d_k. Let $\hat{n}_k$ be the set of neighbors that have a match for the current frame i. Provided $\hat{n}_k \neq \emptyset$, the offset vector for dot d_k^i is calculated as follows: let $\vec{v}_j^i = d_j^i - d_j$ be the offset of dot j (recall that d_j is the initial position for the reference dot j).

$$\vec{v}_k^i = \frac{1}{||\hat{n}_k||} \sum_{d_j^i \in \hat{n}_k} \vec{v}_j^i$$

If the dot has no matched neighbors we repeat as necessary, treating the moved, unmatched reference dots as matched dots. Eventually, the movements will propagate through all of the reference dots.

4 Mesh construction and deformation

4.1 Constructing the mesh

To construct a mesh we begin with a cyberware scan of the head. Because we later need to align the scan with the 3D video dot data, we scanned the head with the fiducials glued on. The resulting scan suffers from four problems:

- The fluorescent fiducials caused "bumps" on the mesh.
- Several parts of the mesh were not adequately scanned, namely, the ears, one side of the nose, the eyes, and under the chin. These were manually corrected.
- The mesh does not have an opening for the mouth.
- The scan has too many polygons.

The bumps caused by the fluorescent fiducials were removed by selecting the vertices which were out of place (approximately 10-30 surrounding each dot) and automatically finding new locations for them by blending between four correct neighbors. Since the scan produces a rectangular grid of vertices we can pick the neighbors to blend between in (u, v) space, i.e., the nearest valid neighbors in the positive and negative u and v direction.

The polygons at the mouth were split and then filled with six rows of polygons located slightly behind the lips. We map the teeth and tongue onto these polygons when the mouth is open.

We reduced the number of polygons in the mesh from approximately $460,000$ to 4800 using Hoppe's simplification method [8].

4.2 Moving the mesh

The vertices are moved by a linear combination of the offsets of the nearest dots (refer to Equation 1). The linear combination for each vertex v_j is expressed as a set of blend coefficients, α_k^j, one for each dot, such that $\sum_{d_k \in D} \alpha_k^j = 1$ (most of the α_k^js will be zero). The new location p_j^i of the vertex v_j at frame i is then

$$p_j^i = p_j + \sum_k \alpha_k^j ||d_k^i - d_k||$$

where p_j is the initial location of the vertex v_j.

For most of the vertices the α_k^js are a weighted average of the closest dots. The vertices in the eyes, mouth, behind the mouth, and outside of the facial area are treated slightly differently since, for example, we do not want the dots on the lower lip influencing vertices on the upper part of the lip. Also, although we tried to keep the head as still as possible, there is still some residual rigid body motion. We need to compensate for this for those vertices that are not directly influenced by a dot (e.g., the back of the head).

We use a two-step process to assign the blend coefficients to the vertices. We first find blend coefficients for a grid of points evenly distributed across the face, then use this grid of points to

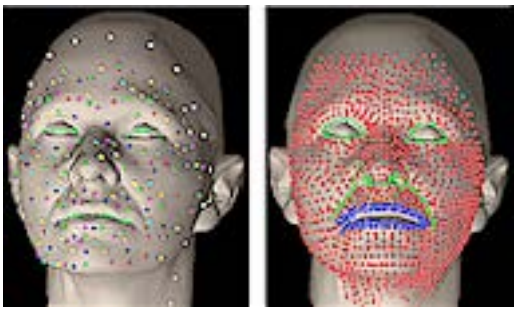


Figure 8: Left: The original dots plus the extra dots (in white). The labeling curves are shown in light green. Right: The grid of dots. Outline dots are green or blue.

Figure 9: Masks surrounding important facial features. The gradient of a blurred version of this mask is used to orient the low-pass filters used in the dot removal process.

assign blend coefficients to the vertices. This two-step process is helpful because both the fluorescent fiducials and the mesh vertices are unevenly distributed across the face, making it difficult to get smoothly changing blend coefficients.

The grid consists of roughly 1400 points, evenly distributed and placed by hand to follow the contours of the face (see Figure 8). The points along the nasolabial furrows, nostrils, eyes, and lips are treated slightly differently than the other points to avoid blending across features such as the lips.

Because we want the mesh movement to go to zero outside of the face, we add another set of unmoving dots to the reference set. These new dots form a ring around the face (see Figure 8) enclosing all of the reference dots. For each frame we determine the rigid body motion of the head (if any) using a subset of those reference dots which are relatively stable. This rigid body transformation is then applied to the new dots.

We label the dots, grid points, and vertices as being *above*, *below*, or *neither* with respect to each of the eyes and the mouth. Dots which are *above* a given feature can not be combined with dots which are *below* that same feature (or vice-versa). Labeling is accomplished using three curves, one for each of the eyes and one for the mouth. Dots directly above (or below) a curve are labeled as *above* (or *below*) that curve. Otherwise, they are labeled *neither*.

4.2.1 Assigning blends to the grid points

The algorithm for assigning blends to the grid points first finds the closest dots, assigns blends, then filters to more evenly distribute the blends.

Finding the ideal set of reference dots to influence a grid point is complicated because the reference dots are not evenly distributed across the face. The algorithm attempts to find two or more dots distributed in a rough circle around the given grid point. To do this we both compensate for the dot density, by setting the search distance using the two closest dots, and by checking for dots which will both "pull" in the same direction.

To find the closest dots to the grid point p we first find δ_1 and δ_2, the distance to the closest and second closest dot, respectively. Let $D_n \subset D$ be the set of dots within $1.8\frac{\delta_1+\delta_2}{2}$ distance of p whose labels do not conflict with p's label. Next, we check for pairs of dots that are more or less in the same direction from p and remove the furthest one. More precisely, let $\hat{v}_i$ be the normalized vector from p to the dot $d_i \in D_n$ and let $\hat{v}_j$ be the normalized vector from p to the dot $d_j \in D_n$. If $\hat{v}_1 \cdot \hat{v}_2 > 0.8$ then remove the furthest of d_i and d_j from the set D_n.

We assign blend values based on the distance of the dots from p. If the dot is not in D_n then its corresponding α value is 0. For the dots in D_n let $l_i = \frac{1.0}{||d_i - p||}$. Then the corresponding α's are

$$\alpha_i = \frac{l_i}{(\sum_{d_i \in D_n} l_i)}$$

We next filter the blend coefficients for the grid points. For each grid point we find the closest grid points – since the grid points are distributed in a rough grid there will usually be 4 neighboring points – using the above routine (replacing the dots with the grid points). We special case the outlining grid points; they are only blended with other outlining grid points. The new blend coefficients are found by taking 0.75 of the grid point's blend coefficients and 0.25 of the average of the neighboring grid point's coefficients. More formally, let $g_i = [\alpha_0, \ldots, \alpha_n]$ be the vector of blend coefficients for the grid point i. Then the new vector g_i' is found as follows, where N_i is the set of neighboring grid points for the grid point i:

$$g_i' = 0.75 g_i + \frac{0.25}{||N_i||} \sum_{j \in N_i} g_j$$

We apply this filter twice to simulate a wide low pass filter.

To find the blend coefficients for the vertices of the mesh we find the closest grid point with the same label as the vertex and copy the blend coefficients. The only exception to this is the vertices for the polygons inside of the mouth. For these vertices we take β of the closest grid point on the top lip and $1.0 - \beta$ of the closest grid point on the bottom lip. The β values are 0.8, 0.6, 0.4, 0.25, and 0.1 from top to bottom of the mouth polygons.

5 Dot removal

Before we create the textures, the dots and their associated illumination effects have to be removed from the camera images. Interreflection effects are surprisingly noticeable because some parts of the face fold dramatically, bringing the reflective surface of some dots into close proximity with the skin. This is a big problem along the naso-labial furrow where diffuse interreflection from the colored dots onto the face significantly alters the skin color.

First, the dot colors are removed from each of the six camera image sequences by substituting skin texture for pixels which are covered by colored dots. Next, diffuse interreflection effects and any remaining color casts from stray pixels that have not been properly substituted are removed.

The skin texture substitution begins by finding the pixels which correspond to colored dots. The nearest neighbor color classifier

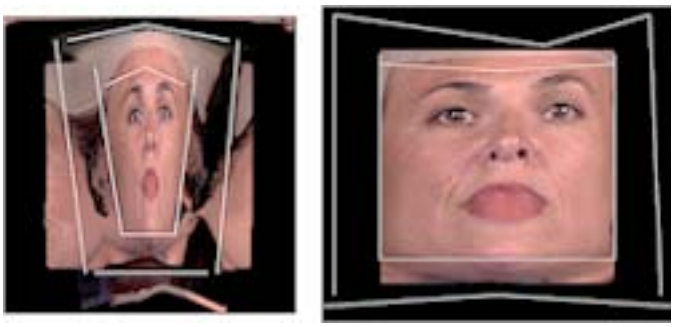


Figure 10: Standard cylindrical texture map. Warped texture map that focuses on the face, and particularly on the eyes and mouth. The warp is defined by the line pairs shown in white.

described in Section 3.1.1 is used to mark all pixels which have any of the dot colors. A special training set is used since in this case false positives are much less detrimental than they are for the dot tracking case. Also, there is no need to distinguish between dot colors, only between dot colors and the background colors. The training set is created to capture as much of the dot color and the boundary region between dots and the background colors as possible.

A dot mask is generated by applying the classifier to each pixel in the image. The mask is grown by a few pixels to account for any remaining pixels which might be contaminated by the dot color. The dot mask marks all pixels which must have skin texture substituted.

The skin texture is broken into low spatial frequency and high frequency components. The low frequency components of the skin texture are interpolated by using a directional low pass filter oriented parallel to features that might introduce intensity discontinuities. This prevents bleeding of colors across sharp intensity boundaries such as the boundary between the lips and the lighter colored regions around the mouth. The directionality of the filter is controlled by a two dimensional mask which is the projection into the image plane of a three dimensional polygon mask lying on the 3D face model. Because the polygon mask is fixed on the 3D mesh, the 2D projection of the polygon mask stays in registration with the texture map as the face deforms.

All of the important intensity gradients have their own polygon mask: the eyes, the eyebrows, the lips, and the naso-labial furrows (see 9). The 2D polygon masks are filled with white and the region of the image outside the masks is filled with black to create an image. This image is low-pass filtered. The intensity of the resulting image is used to control how directional the filter is. The filter is circularly symmetric where the image is black, i.e., far from intensity discontinuities, and it is very directional where the image is white. The directional filter is oriented so that its long axis is orthogonal to the gradient of this image.

The high frequency skin texture is created from a rectangular sample of skin texture taken from a part of the face that is free of dots. The skin sample is highpass filtered to eliminate low frequency components. At each dot mask pixel location the highpass filtered skin texture is first registered to the center of the 2D bounding box of the connected dot region and then added to the low frequency interpolated skin texture.

The remaining diffuse interreflection effects are removed by clamping the hue of the skin color to a narrow range determined from the actual skin colors. First the pixel values are converted from RGB to HSV space and then any hue outside the legal range is clamped to the extremes of the range. Pixels in the eyes and mouth, found using the eye and lip masks shown in Figure 9, are left unchanged.

Some temporal variation remains in the substituted skin texture due to imperfect registration of the high frequency texture from frame to frame. A low pass temporal filter is applied to the dot mask regions in the texture images, because in the texture map space the dots are relatively motionless. This temporal filter effectively eliminates the temporal texture substitution artifacts.

6 Creating the texture maps

Figure 11 is a flowchart of the texture creation process. We create texture maps for every frame of our animation in a four-step process. The first two steps are performed only once per mesh. First we define a parameterization of the mesh. Second, using this parameterization, we create a *geometry map* containing a location on the mesh for each texel. Third, for every frame, we create six preliminary texture maps, one from each camera image, along with weight maps. The weight maps indicate the relative quality of the data from the different cameras. Fourth, we take a weighted average of these texture maps to make our final texture map.

We create an initial set of texture coordinates for the head by tilting the mesh back 10 degrees to expose the nostrils and projecting the mesh vertices onto a cylinder. A texture map generated using this parametrization is shown on the left of Figure 10. We specify a set of line pairs and warp the texture coordinates using the technique described by Beier and Neely[1]. This parametrization results in the texture map shown on the right of Figure 10. Only the front of the head is textured with data from the six video streams.

Next we create the geometry map containing a mesh location for each texel. A mesh location is a triple (k, β_1, β_2) specifying a triangle k and barycentric coordinates in the triangle (β_1, β_2, $1 - \beta_1 - \beta_2$). To find the triangle identifier k for texel (u, v) we exhaustively search through the mesh's triangles to find the one that contains the texture coordinates (u, v). We then set the β_is to be the barycentric coordinates of the point (u, v) in the texture coordinates of the triangle k. When finding the mesh location for a pixel we already know in which triangles its neighbors above and to the left lie. Therefore, we speed our search by first searching through these triangles and their neighbors. However, the time required for this task is not critical as the geometry map need only be created once.

Next we create preliminary texture maps for frame f one for each camera. This is a modified version of the technique described in [11]. To create the texture map for camera c, we begin by deforming the mesh into its frame f position. Then, for each texel, we get its mesh location, (k, β_1, β_2), from the geometry map. With the 3D coordinates of triangle k's vertices and the barycentric coordinates β_i, we compute the texel's 3D location t. We transform t by camera c's projection matrix to obtain a location, (x, y), on camera c's image plane. We then color the texel with the color from camera c's image at (x, y). We set the texel's weight to the dot product of the mesh normal at t, $\hat{n}$, with the direction back to the camera, $\hat{d}$ (see Figure 12). Negative values are clamped to zero. Hence, weights are low where the camera's view is glancing. However, this weight map is not smooth at triangle boundaries, so we smooth it by convolving it with a Gaussian kernel.

Last, we merge the six preliminary texture maps. As they do not align perfectly, averaging them blurs the texture and loses detail. Therefore, we use only the texture map of our bottom, center camera for the center 46 % of the final texture map. We smoothly transition (over 23 pixels) to using a weighted average of each preliminary texture map at the sides.

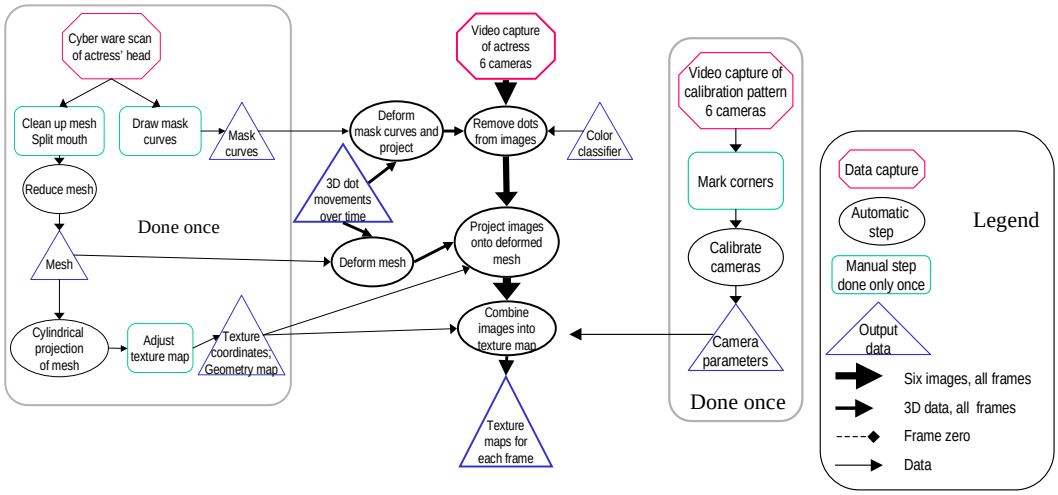


Figure 11: Creating the texture maps.

We texture the parts of the head not covered by the aforementioned texture maps with the captured reflectance data from our Cyberware scan, modified in two ways. First, because we replaced the mesh's ears with ears from a stock mesh (Section 4.1), we moved the ears in the texture to achieve better registration. Second, we set the alpha channel to zero (with a soft edge) in the region of the texture for the front of the head. Then we render in two passes to create an image of the head with both texture maps applied.

7 Compression

7.1 Principal Components Analysis

The geometric and texture map data have different statistical characteristics and are best compressed in different ways. There is significant long-term temporal correlation in the geometric data since similar facial expressions occur throughout the sequence. The short term correlation of the texture data is significantly increased over that of the raw video footage because in the texture image space the fiducials are essentially motionless. This eliminates most of the intensity changes associated with movement and leaves primarily shading changes. Shading changes tend to have low spatial frequencies and are highly compressible. Compression schemes such as MPEG, which can take advantage of short term temporal correlation, can exploit this increase in short term correlation.

For the geometric data, one way to exploit the long term correlation is to use principal component analysis. If we represent our data set as a matrix A, where frame i of the data maps column i of A, then the first principal component of A is

$$\max_u (A^T u)^T (A^T u) \tag{2}$$

The u which maximizes Equation 2 is the eigenvector associated with the largest eigenvalue of AA^T, which is also the value of the maximum. Succeeding principal components are defined similarly, except that they are required to be orthogonal to all preceding principal components, i.e., $u_i^T u_j = 0$ for $j \neq i$. The principal components form an orthonormal basis set represented by the matrix U where the columns of U are the principal components of A ordered by eigenvalue size with the most significant principal component in the first column of U.

The data in the A matrix can be projected onto the principal component basis as follows:

$$W = U^T A$$

Row i of W is the projection of column A_i onto the basis vector u_i. More precisely, the jth element in row i of W corresponds to the projection of frame j of the original data onto the ith basis vector. We will call the elements of the W matrix projection *coefficients*.

Similarly, A can be reconstructed exactly from W by multiplication by the basis set, i.e., $A = UW$.

The most important property of the principal components for our purposes is that they are the best linear basis set for reconstruction in the l_2 norm sense. For any given matrix U_k, where k is the number of columns of the matrix and $k < rank(A)$, the reconstruction error

$$e = ||A - U_k U_k^T A||_F^2 \tag{3}$$

where $||B||_F^2$ is the Frobenius norm defined to be

$$||B||_F^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} b_{ij}^2 \tag{4}$$

will be minimized if U_k is the matrix containing the k most significant principal components of A.

We can compress a data set A by quantizing the elements of its corresponding W and U matrices and entropy coding them. Since the compressed data cannot be reconstructed without the principal component basis vectors both the W and U matrices have to be compressed. The basis vectors add overhead that is not present with basis sets that can be computed independent of the original data set, such as the DCT basis.

For data sequences that have no particular structure the extra overhead of the basis vectors would probably out-weigh any gain in compression efficiency. However, for data sets with regular frame to frame structure the residual error for reconstruction with the principal component basis vectors can be much smaller than for other bases. This reduction in residual error can be great enough to compensate for the overhead bits of the basis vectors.

The principal components can be computed using the singular value decomposition (SVD) [13]. Efficient implementations of this algorithm are widely available. The SVD of a matrix A is

Figure 12: Creating the preliminary texture map.

$$A = U\Sigma V^T \tag{5}$$

where the columns of U are the eigenvectors of AA^T, the singular values, σ_i, along the diagonal matrix Σ are the square roots of the eigenvalues of AA^T, and the columns of V are the eigenvectors of A^TA. The ith column of U is the ith principal component of A. Computing the first k left singular vectors of A is equivalent to computing the first k principal components.

7.2 Geometric Data

The geometric data has the long term temporal coherence properties mentioned above since the motion of the face is highly structured. The overhead of the basis vectors for the geometric data is fixed because there are only 182 fiducials on the face. The maximum number of basis vectors is $182 * 3$ since there are three numbers, x, y, and z, associated with each fiducial. Consequently, the basis vector overhead steadily diminishes as the length of the animation sequence increases.

The geometric data is mapped to matrix form by taking the 3D offset data for the ith frame and mapping it the ith column of the data matrix A_g. The first k principal components, U_g, of A_g are computed and A_g is projected into the U_g basis to give the projection coefficients W_g.

There is significant correlation between the columns of projection coefficients because the motion of the dots is relatively smooth over time. We can reduce the entropy of the quantized projection coefficients by temporally predicting the projection coefficients in column i from column $i-1$, i.e., $c_i = c_{i-1} + \Delta_i$ where we encode Δ_i.

For our data set, only the projection coefficients associated with the first 45 principal components, corresponding to the first 45 rows of W_g, have significant temporal correlation so only the first 45 rows are temporally predicted. The remaining rows are entropy coded directly. After the temporal prediction the entropy is reduced by about 20 percent (Figure 13).

The basis vectors are compressed by choosing a peak error rate and then varying the number of quantization levels allocated to each vector based on the standard deviation of the projection coefficients for each vector.

We visually examined animation sequences with W_g and U_g compressed at a variety of peak error rates and chose a level which resulted in undetectable geometric jitter in reconstructed animation. The entropy of W_g for this error level is 26 Kbits per second and the entropy of U_g is 13 kbits per second for a total of 40 kbits per second for all the geometric data. These values were computed for our 3330 frame animation sequence.

8 Results

Figure 16 shows some typical frames from a reconstructed sequence of 3D facial expressions. These frames are taken from a 3330 frame animation in which the actress makes random expressions while reading from a script[2].

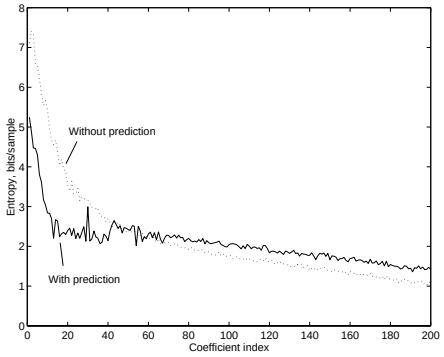


Figure 13: Reduction in entropy after temporal prediction.

The facial expressions look remarkably life-like. The animation sequence is similarly striking. Virtually all evidence of the colored fiducials and diffuse interreflection artifacts is gone, which is surprising considering that in some regions of the face, especially around the lips, there is very little of the actress' skin visible – most of the area is covered by colored fiducials.

Both the accurate 3D geometry and the accurate face texture contribute to the believability of the reconstructed expressions. Occlusion contours look correct and the subtle details of face geometry that are very difficult to capture as geometric data show up well in the texture images. Important examples of this occur at the nasolabial furrow which runs from just above the nares down to slightly below the lips, eyebrows, and eyes. Forehead furrows and wrinkles also are captured. To recreate these features using geometric data rather than texture data would require an extremely detailed 3D capture of the face geometry and a resulting high polygon count in the 3D model. In addition, shading these details properly if they were represented as geometry would be difficult since it would require computing shadows and possibly even diffuse interreflection effects in order to look correct. Subtle shading changes on the smooth parts of the skin, most prominent at the cheekbones, are also captured well in the texture images.

There are still visible artifacts in the animation, some of which are polygonization or shading artifacts, others of which arise because of limitations in our current implementation.

Some polygonization of the face surface is visible, especially along the chin contour, because the front surface of the head contains only 4500 polygons. This is not a limitation of the algorithm – we chose this number of polygons because we wanted to verify that believable facial animation could be done at polygon resolutions low enough to potentially be displayed in real time on inexpensive ($200) 3D graphics cards[3]. For film or television work, where real time rendering is not an issue, the polygon count can be made much higher and the polygonization artifacts will disappear. As graphics hardware becomes faster the differential in quality between offline and online rendered face images will diminish.

Several artifacts are simply the result of our current implementation. For example, occasionally the edge of the face, the tips of the nares, and the eyebrows appear to jitter. This usually occurs when dots are lost, either by falling below the minimum size threshold or by not being visible to three or more cameras. When a dot is lost the algorithm synthesizes dot position data which is

[2] The rubber cap on the actress' head was used to keep her hair out of her face.

[3] In this paper we have not addressed the issue of real time texture decompression and rendering of the face model, but we plan to do so in future work

usually incorrect enough that it is visible as jitter. More cameras, or better placement of the cameras, would eliminate this problem. However, overall the image is extremely stable.

In retrospect, a mesh constructed by hand with the correct geometry and then fit to the cyberware data [10] would be much simpler and possibly reduce some of the polygonization artifacts.

Another implementation artifact that becomes most visible when the head is viewed near profile is that the teeth and tongue appear slightly distorted. This is because we do not use correct 3D models to represent them. Instead, the texture map of the teeth and tongue is projected onto a sheet of polygons stretching between the lips. It is possible that the teeth and tongue could be tracked using more sophisticated computer vision techniques and then more correct geometric models could be used.

Shading artifacts represent an intrinsic limitation of the algorithm. The highlights on the eyes and skin remain in fixed positions regardless of point of view, and shadowing is fixed at the time the video is captured. However, for many applications this should not be a limitation because these artifacts are surprisingly subtle. Most people do not notice that the shading is incorrect until it is pointed out to them, and even then frequently do not find it particularly objectionable. The highlights on the eyes can probably be corrected by building a 3D eye model and creating synthetic highlights appropriate for the viewing situation. Correcting the skin shading and self shadowing artifacts is more difficult. The former will require very realistic and efficient skin reflectance models while the latter will require significant improvements in rendering performance, especially if the shadowing effect of area light sources is to be adequately modeled. When both these problems are solved then it will no longer be necessary to capture the live video sequence – only the 3D geometric data and skin reflectance properties will be needed.

The compression numbers are quite good. Figure 14 shows a single frame from the original sequence, the same frame compressed by the MPEG4 codec at 460 Kbps and at 260 KBps. All of the images look quite good. The animated sequences also look good, with the 260 KBps sequence just beginning to show noticeable compression artifacts. The 260 KBps video is well within the bandwidth of single speed CDROM drives. This data rate is probably low enough that decompression could be performed in real time in software on the fastest personal computers so there is the potential for real time display of the resulting animations. We intend to investigate this possibility in future work.

There is still room for significant improvement in our compression. A better mesh parameterization would significantly reduce the number of bits needed to encode the eyes, which distort significantly over time in the texture map space. Also the teeth, inner edges of the lips, and the tongue could potentially be tracked over time and at least partially stabilized, resulting in a significant reduction in bit rate for the mouth region. Since these two regions account for the majority of the bit budget, the potential for further reduction in bit rate is large.

9 Conclusion

The system produces remarkably lifelike reconstructions of facial expressions recorded from live actors' performances. The accurate 3D tracking of a large number of points on the face results in an accurate 3D model of facial expression. The texture map sequence captured simultaneously with the 3D deformation data captures details of expression that would be difficult to capture any other way. By using the 3D deformation information to register the texture maps from frame to frame the variance of the texture map sequence is significantly reduced which increases its compressibility. Image quality of 30 frame per second animations, reconstructed at approximately 300 by 400 pixels, is still good at data rates as low as 240 Kbits per second, and there is significant potential for lowering this bit rate even further. Because the bit overhead for the geometric data is low in comparison to the texture data one can get a 3D talking head, with all the attendant flexibility, for little more than the cost of a conventional video sequence. With the true 3D model of facial expression, the animation can be viewed from any angle and placed in a 3D virtual environment, making it much more flexible than conventional video.

References

[1] BEIER, T., AND NEELY, S. Feature-based image metamorphosis. In *Computer Graphics (SIGGRAPH '92 Proceedings)* (July 1992), E. E. Catmull, Ed., vol. 26, pp. 35–42.

[2] BREGLER, C., COVELL, M., AND SLANEY, M. Video rewrite: Driving visual speech with audio. *Computer Graphics 31*, 2 (Aug. 1997), 353–361.

[3] CASSELL, J., PELACHAUD, C., BADLER, N., STEEDMAN, M., ACHORN, B., BECKET, T., DOUVILLE, B., PREVOST, S., AND STONE, M. Animated conversation: Rule-based generation of facial expression, gesture and spoken intonation for multiple conversational agents. *Computer Graphics 28*, 2 (Aug. 1994), 413–420.

[4] DECARLO, D., AND METAXAS, D. The integration of optical flow and deformable models with applications to human face shape and motion estimation. *Proceedings CVPR* (1996), 231–238.

[5] ESSA, I., AND PENTLAND, A. Coding, analysis, interpretation and recognition of facial expressions. *IEEE Transactions on Pattern Analysis and Machine Intelligence 19*, 7 (1997), 757–763.

[6] FAUGERAS, O. *Three-dimensional computer vision*. MIT Press, Cambridge, MA, 1993.

[7] FISCHLER, M. A., AND BOOLES, R. C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM 24*, 6 (Aug. 1981), 381–395.

[8] HOPPE, H. Progressive meshes. In *SIGGRAPH 96 Conference Proceedings* (Aug. 1996), H. Rushmeier, Ed., Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 99–108. held in New Orleans, Louisiana, 04-09 August 1996.

[9] HORN, B. K. P. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America 4*, 4 (Apr. 1987).

[10] LEE, Y., TERZOPOULOS, D., AND WATERS, K. Realistic modeling for facial animation. *Computer Graphics 29*, 2 (July 1995), 55–62.

[11] PIGHIN, F., AUSLANDER, J., LISHINSKI, D., SZELISKI, R., AND SALESIN, D. Realistic facial animation using image based 3d morphing. Tech. Report TR-97-01-03, Department of Computer Science and Engineering, University of Washington, Seattle, Wa, 1997.

[12] SCHÜRMANN, J. *Pattern Classification: A Unified View of Statistical and Neural Approaches*. John Wiley and Sons, Inc., New York, 1996.

Figure 14: Left to Right: Mesh with uncompressed textures, compressed to 400 kbits/sec, and compressed to 200 kbits/sec

[13] STRANG. *Linear Algebra and its Application*. HBJ, 1988.

[14] WATERS, K. A muscle model for animating three-dimensional facial expression. In *Computer Graphics (SIGGRAPH '87 Proceedings)* (July 1987), M. C. Stone, Ed., vol. 21, pp. 17–24.

[15] WILLIAMS, L. Performance-driven facial animation. *Computer Graphics 24*, 2 (Aug. 1990), 235–242.

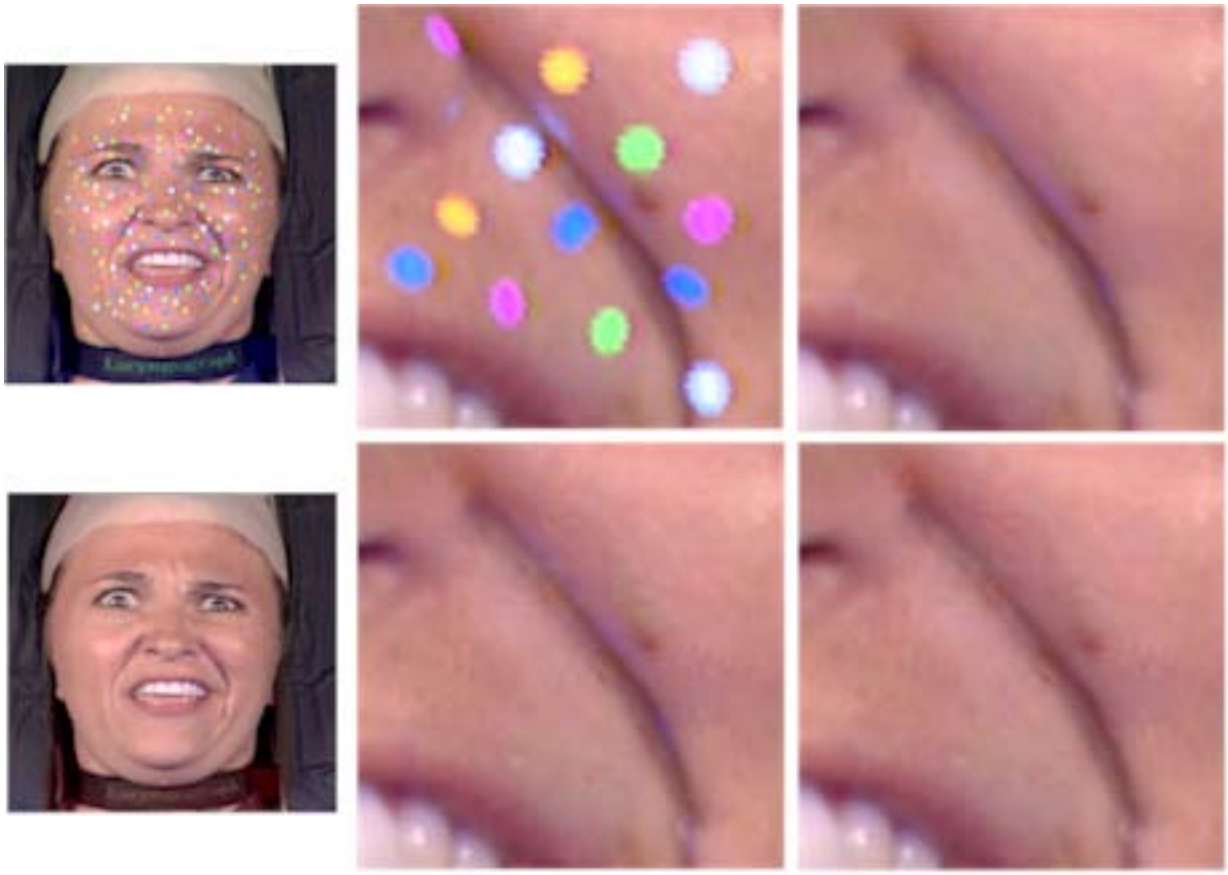


Figure 15: Face before and after dot removal, with details showing the steps in the dot removal process. From left to right, top to bottom: Face with dots, dots replaced with low frequency skin texture, high frequency skin texture added, hue clamped.

Figure 16: Sequence of rendered images of textured mesh.

Synthesizing Realistic Facial Expressions from Photographs

Frédéric Pighin *Jamie Hecker* *Dani Lischinski*[†] *Richard Szeliski*[‡] *David H. Salesin*

University of Washington [†]The Hebrew University [‡]Microsoft Research

Abstract

We present new techniques for creating photorealistic textured 3D facial models from photographs of a human subject, and for creating smooth transitions between different facial expressions by morphing between these different models. Starting from several uncalibrated views of a human subject, we employ a user-assisted technique to recover the camera poses corresponding to the views as well as the 3D coordinates of a sparse set of chosen locations on the subject's face. A scattered data interpolation technique is then used to deform a generic face mesh to fit the particular geometry of the subject's face. Having recovered the camera poses and the facial geometry, we extract from the input images one or more texture maps for the model. This process is repeated for several facial expressions of a particular subject. To generate transitions between these facial expressions we use 3D shape morphing between the corresponding face models, while at the same time blending the corresponding textures. Using our technique, we have been able to generate highly realistic face models and natural looking animations.

CR Categories: I.2.10 [Artificial Intelligence]: Vision and Scene Understanding — Modeling and recovery of physical attributes; I.3.7 [Computer Graphics]: Three-Dimensional Graphics — Animation; I.3.7 [Computer Graphics]: Three-Dimensional Graphics — Color, shading, shadowing and texture.

Additional Keywords: facial modeling, facial expression generation, facial animation, photogrammetry, morphing, view-dependent texture-mapping

1 Introduction

> *There is no landscape that we know as well as the human face. The twenty-five-odd square inches containing the features is the most intimately scrutinized piece of territory in existence, examined constantly, and carefully, with far more than an intellectual interest. Every detail of the nose, eyes, and mouth, every regularity in proportion, every variation from one individual to the next, are matters about which we are all authorities.*
>
> — Gary Faigin [14],
> from *The Artist's Complete Guide to Facial Expression*

Realistic facial synthesis is one of the most fundamental problems in computer graphics — and one of the most difficult. Indeed, attempts to model and animate realistic human faces date back to the early 70's [34], with many dozens of research papers published since. The applications of facial animation include such diverse fields as character animation for films and advertising, computer games [19], video teleconferencing [7], user-interface agents and avatars [44], and facial surgery planning [23, 45]. Yet no perfectly realistic facial animation has ever been generated by computer: no "facial animation Turing test" has ever been passed.

There are several factors that make realistic facial animation so elusive. First, the human face is an extremely complex geometric form. For example, the human face models used in Pixar's *Toy Story* had several thousand control points each [10]. Moreover, the face exhibits countless tiny creases and wrinkles, as well as subtle variations in color and texture — all of which are crucial for our comprehension and appreciation of facial expressions. As difficult as the face is to model, it is even more problematic to animate, since facial movement is a product of the underlying skeletal and muscular forms, as well as the mechanical properties of the skin and subcutaneous layers (which vary in thickness and composition in different parts of the face). All of these problems are enormously magnified by the fact that we as humans have an uncanny ability to read expressions — an ability that is not merely a learned skill, but part of our deep-rooted instincts. For facial expressions, the slightest deviation from truth is something any person will immediately detect.

A number of approaches have been developed to model and animate realistic facial expressions in three dimensions. (The reader is referred to the recent book by Parke and Waters [36] for an excellent survey of this entire field.) Parke's pioneering work introduced simple geometric interpolation between face models that were digitized by hand [34]. A radically different approach is performance-based animation, in which measurements from real actors are used to drive synthetic characters [4, 13, 47]. Today, face models can also be obtained using laser-based cylindrical scanners, such as those produced by Cyberware [8]. The resulting range and color data can be fitted with a structured face mesh, augmented with a physically-based model of skin and muscles [29, 30, 43, 46]. The animations produced using these face models represent the state-of-the-art in automatic physically-based facial animation.

For sheer photorealism, one of the most effective approaches to date has been the use of 2D morphing between photographic images [3]. Indeed, some remarkable results have been achieved in this way — most notably, perhaps, the Michael Jackson video produced by PDI, in which very different-looking actors are seemingly transformed into one another as they dance. The production of this video, however, required animators to painstakingly specify a few dozen carefully chosen correspondences between physical features of the actors in almost every frame. Another problem with 2D image morphing is that it does not correctly account for changes in viewpoint or object pose. Although this shortcoming has been recently addressed by a technique called "view morphing" [39], 2D morphing still lacks some of the advantages of a 3D model, such as the complete freedom of viewpoint and the ability to composite the image with other 3D graphics. Morphing has also been applied in 3D: Chen *et al.* [6] applied Beier and Neely's 2D morphing technique [3] to morph between cylindrical laser scans of human heads. Still, even in this case the animator must specify correspondences for every pair of expressions in order to produce a transition between them. More recently,

Bregler *et al.* [5] used morphing of mouth regions to lip-synch existing video to a novel sound-track.

In this paper, we show how 2D morphing techniques can be combined with 3D transformations of a geometric model to automatically produce 3D facial expressions with a high degree of realism. Our process consists of several basic steps. First, we capture multiple views of a human subject (with a given facial expression) using cameras at arbitrary locations. Next, we digitize these photographs and manually mark a small set of initial corresponding points on the face in the different views (typically, corners of the eyes and mouth, tip of the nose, etc.). These points are then used to automatically recover the camera parameters (position, focal length, etc.) corresponding to each photograph, as well as the 3D positions of the marked points in space. The 3D positions are then used to deform a generic 3D face mesh to fit the face of the particular human subject. At this stage, additional corresponding points may be marked to refine the fit. Finally, we extract one or more texture maps for the 3D model from the photos. Either a single view-independent texture map can be extracted, or the original images can be used to perform view-dependent texture mapping. This whole process is repeated for the same human subject, with several different facial expressions. To produce facial animations, we interpolate between two or more different 3D models constructed in this way, while at the same time blending the textures. Since all the 3D models are constructed from the same generic mesh, there is a natural correspondence between all geometric points for performing the morph. Thus, transitions between expressions can be produced entirely automatically once the different face models have been constructed, without having to specify pairwise correspondences between any of the expressions.

Our modeling approach is based on photogrammetric techniques in which images are used to create precise geometry [31, 40]. The earliest such techniques applied to facial modeling and animation employed grids that were drawn directly on the human subject's face [34, 35]. One consequence of these grids, however, is that the images used to construct geometry can no longer be used as valid texture maps for the subject. More recently, several methods have been proposed for modeling the face photogrammetrically without the use of grids [20, 24]. These modeling methods are similar in concept to the modeling technique described in this paper. However, these previous techniques use a small predetermined set of features to deform the generic face mesh to the particular face being modeled, and offer no mechanism to further improve the fit. Such an approach may perform poorly on faces with unusual features or other significant deviations from the normal. Our system, by contrast, gives the user complete freedom in specifying the correspondences, and enables the user to refine the initial fit as needed. Another advantage of our technique is its ability to handle fairly arbitrary camera positions and lenses, rather than using a fixed pair that are precisely oriented. Our method is similar, in concept, to the work done in architectural modeling by Debevec *et al.* [9], where a set of annotated photographs are used to model buildings starting from a rough description of their shape. Compared to facial modeling methods that utilize a laser scanner, our technique uses simpler acquisition equipment (regular cameras), and it is capable of extracting texture maps of higher resolution. (Cyberware scans typically produce a cylindrical grid of 512 by 256 samples). The price we pay for these advantages is the need for user intervention in the modeling process.

We employ our system not only for creating realistic face models, but also for performing realistic transitions between different expressions. One advantage of our technique, compared to more traditional animatable models with a single texture map, is that we can capture the subtle changes in illumination and appearance (e.g., facial creases) that occur as the face is deformed. This degree of realism is difficult to achieve even with physically-based models, because of the complexity of skin folding and the difficulty of simulating interreflections and self-shadowing [18, 21, 32].

This paper also presents several new expression synthesis techniques based on extensions to the idea of morphing. We develop a morphing technique that allows for different regions of the face to have different "percentages" or "mixing proportions" of facial expressions. We also introduce a painting interface, which allows users to locally add in a little bit of an expression to an existing composite expression. We believe that these novel methods for expression generation and animation may be more natural for the average user than more traditional animation systems, which rely on the manual adjustments of dozens or hundreds of control parameters.

The rest of this paper is organized as follows. Section 2 describes our method for fitting a generic face mesh to a collection of simultaneous photographs of an individual's head. Section 3 describes our technique for extracting both view-dependent and view-independent texture maps for photorealistic rendering of the face. Section 4 presents the face morphing algorithm that is used to animate the face model. Section 5 describes the key aspects of our system's user interface. Section 6 presents the results of our experiments with the proposed techniques, and Section 7 offers directions for future research.

2 Model fitting

The task of the model-fitting process is to adapt a generic face model to fit an individual's face and facial expression. As input to this process, we take several images of the face from different viewpoints (Figure 1a) and a generic face model (we use the generic face model created with Alias|Wavefront [2] shown in Figure 1c). A few features points are chosen (13 in this case, shown in the frames of Figure 1a) to recover the camera pose. These same points are also used to refine the generic face model (Figure 1d). The model can be further refined by drawing corresponding curves in the different views (Figure 1b). The output of the process is a face model that has been adapted to fit the face in the input images (Figure 1e), along with a precise estimate of the camera pose corresponding to each input image.

The model-fitting process consists of three stages. In the *pose recovery* stage, we apply computer vision techniques to estimate the viewing parameters (position, orientation, and focal length) for each of the input cameras. We simultaneously recover the 3D coordinates of a set of *feature points* on the face. These feature points are selected interactively from among the face mesh vertices, and their positions in each image (where visible) are specified by hand. The *scattered data interpolation* stage uses the estimated 3D coordinates of the feature points to compute the positions of the remaining face mesh vertices. In the *shape refinement* stage, we specify additional correspondences between facial vertices and image coordinates to improve the estimated shape of the face (while keeping the camera pose fixed).

2.1 Pose recovery

Starting with a rough knowledge of the camera positions (e.g., frontal view, side view, etc.) and of the 3D shape (given by the generic head model), we iteratively improve the pose and the 3D shape estimates in order to minimize the difference between the predicted and observed feature point positions. Our formulation is based on the non-linear least squares structure-from-motion algorithm introduced by Szeliski and Kang [41]. However, unlike the method they describe, which uses the Levenberg-Marquardt algorithm to perform a complete iterative minimization over all of the unknowns simultaneously, we break the problem down into a series of linear least squares problems that can be solved using very simple

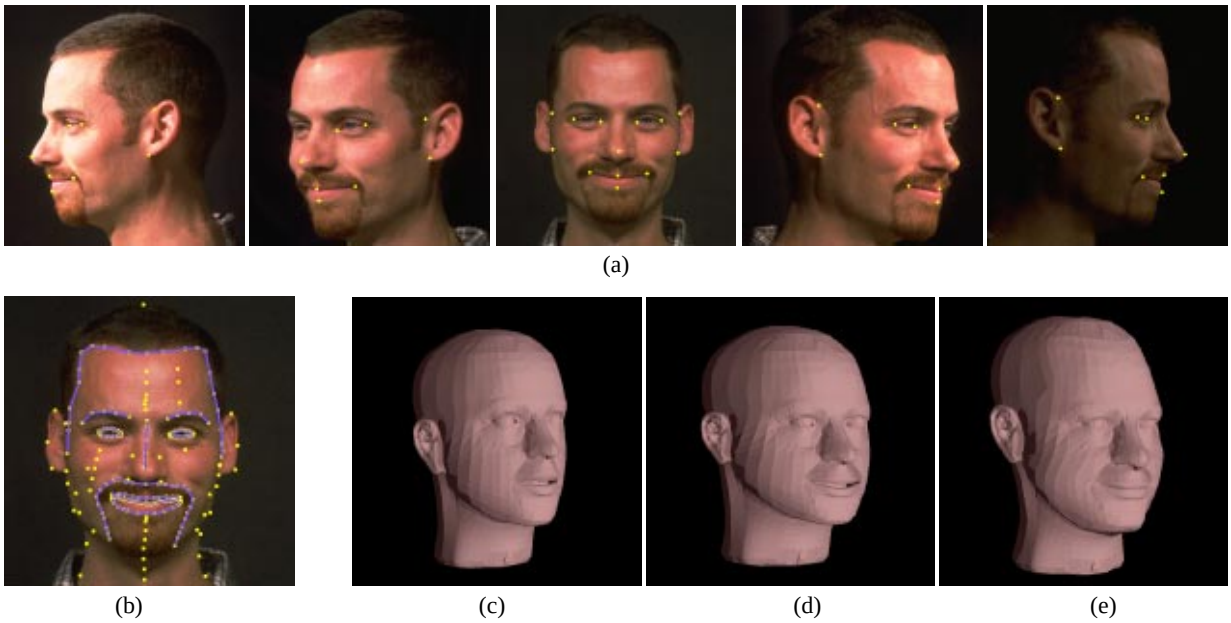


Figure 1 Model-fitting process: (a) a set of input images with marked feature points, (b) facial features annotated using a set of curves, (c) generic face geometry (shaded surface rendering), (d) face adapted to initial 13 feature points (after pose estimation) (e) face after 99 additional correspondences have been given.

and numerically stable techniques [16, 37].

To formulate the pose recovery problem, we associate a rotation matrix $\boldsymbol{R}^k$ and a translation vector $\boldsymbol{t}^k$ with each camera pose k. (The three rows of $\boldsymbol{R}^k$ are $\boldsymbol{r}_x^k$, $\boldsymbol{r}_y^k$, and $\boldsymbol{r}_z^k$, and the three entries in $\boldsymbol{t}^k$ are t_x^k, t_y^k, t_z^k.) We write each 3D feature point as $\boldsymbol{p}_i$, and its 2D screen coordinates in the k-th image as (x_i^k, y_i^k).

Assuming that the origin of the (x, y) image coordinate system lies at the optical center of each image (i.e., where the optical axis intersects the image plane), the traditional 3D projection equation for a camera with a focal length f^k (expressed in pixels) can be written as

$$x_i^k = f^k \frac{\boldsymbol{r}_x^k \cdot \boldsymbol{p}_i + t_x^k}{\boldsymbol{r}_z^k \cdot \boldsymbol{p}_i + t_z^k} \qquad y_i^k = f^k \frac{\boldsymbol{r}_y^k \cdot \boldsymbol{p}_i + t_y^k}{\boldsymbol{r}_z^k \cdot \boldsymbol{p}_i + t_z^k} \tag{1}$$

(This is just an explicit rewriting of the traditional projection equation $\boldsymbol{x}_i^k \propto \boldsymbol{R}^k \boldsymbol{p}_i + \boldsymbol{t}^k$ where $\boldsymbol{x}_i^k = (x_i^k, y_i^k, f^k)$.)

Instead of using (1) directly, we reformulate the problem to estimate inverse distances to the object [41]. Let $\eta^k = 1/t_z^k$ be this inverse distance and $s^k = f^k \eta^k$ be a world-to-image scale factor. The advantage of this formulation is that the scale factor s^k can be reliably estimated even when the focal length is long, whereas the original formulation has a strong coupling between the f^k and t_z^k parameters.

Performing these substitution, we obtain

$$x_i^k = s^k \frac{\boldsymbol{r}_x^k \cdot \boldsymbol{p}_i + t_x^k}{1 + \eta^k \boldsymbol{r}_z^k \cdot \boldsymbol{p}_i}$$
$$y_i^k = s^k \frac{\boldsymbol{r}_y^k \cdot \boldsymbol{p}_i + t_y^k}{1 + \eta^k \boldsymbol{r}_z^k \cdot \boldsymbol{p}_i}.$$

If we let $w_i^k = (1 + \eta^k (\boldsymbol{r}_z^k \cdot \boldsymbol{p}_i))^{-1}$ be the inverse denominator, and collect terms on the left-hand side, we get

$$w_i^k \left(x_i^k + x_i^k \eta^k (\boldsymbol{r}_z^k \cdot \boldsymbol{p}_i) - s^k (\boldsymbol{r}_x^k \cdot \boldsymbol{p}_i + t_x^k) \right) = 0 \tag{2}$$
$$w_i^k \left(y_i^k + y_i^k \eta^k (\boldsymbol{r}_z^k \cdot \boldsymbol{p}_i) - s^k (\boldsymbol{r}_y^k \cdot \boldsymbol{p}_i + t_y^k) \right) = 0$$

Note that these equations are linear in each of the unknowns that we wish to recover, i.e., $\boldsymbol{p}_i$, t_x^k, t_y^k, η^k, s^k, and $\boldsymbol{R}^k$, if we ignore the variation of w_i^k with respect to these parameters. (The reason we keep the w_i^k term, rather than just dropping it from these equations, is so that the linear equations being solved in the least squares step have the same magnitude as the original measurements (x_i^k, y_i^k). Hence, least-squares will produce a *maximum likelihood* estimate for the unknown parameters [26].)

Given estimates for initial values, we can solve for different subsets of the unknowns. In our current algorithm, we solve for the unknowns in five steps: first s^k, then $\boldsymbol{p}_i$, $\boldsymbol{R}^k$, t_x^k and t_y^k, and finally η^k. This order is chosen to provide maximum numerical stability given the crude initial pose and shape estimates. For each parameter or set of parameters chosen, we solve for the unknowns using linear least squares (Appendix A). The simplicity of this approach is a result of solving for the unknowns in five separate stages, so that the parameters for a given camera or 3D point can be recovered independently of the other parameters.

2.2 Scattered data interpolation

Once we have computed an initial set of coordinates for the feature points $\boldsymbol{p}_i$, we use these values to deform the remaining vertices on the face mesh. We construct a smooth interpolation function that gives the 3D displacements between the original point positions and the new adapted positions for every vertex in the original generic face mesh. Constructing such an interpolation function is a standard problem in scattered data interpolation. Given a set of known displacements $\boldsymbol{u}_i = \boldsymbol{p}_i - \boldsymbol{p}_i^{(0)}$ away from the original positions $\boldsymbol{p}_i^{(0)}$ at every constrained vertex i, construct a function that gives the displacement $\boldsymbol{u}_j$ for every unconstrained vertex j.

There are several considerations in choosing the particular data interpolant [33]. The first consideration is the embedding space, that is, the domain of the function being computed. In our case, we use the original 3D coordinates of the points as the domain. (An alternative would be to use some 2D parameterization of the surface mesh, for instance, the cylindrical coordinates described in Section 3.) We therefore attempt to find a smooth vector-valued function $\boldsymbol{f}(\boldsymbol{p})$ fitted

to the known data $\boldsymbol{u}_i = \boldsymbol{f}(\boldsymbol{p}_i)$, from which we can compute $\boldsymbol{u}_j = \boldsymbol{f}(\boldsymbol{p}_j)$.

There are also several choices for how to construct the interpolating function [33]. We use a method based on *radial basis functions*, that is, functions of the form

$$\boldsymbol{f}(\boldsymbol{p}) = \sum_i \boldsymbol{c}_i \phi(\|\boldsymbol{p} - \boldsymbol{p}_i\|),$$

where $\phi(r)$ are radially symmetric basis functions. A more general form of this interpolant also adds some low-order polynomial terms to model global, e.g., affine, deformations [27, 28, 33]. In our system, we use an affine basis as part of our interpolation algorithm, so that our interpolant has the form:

$$\boldsymbol{f}(\boldsymbol{p}) = \sum_i \boldsymbol{c}_i \phi(\|\boldsymbol{p} - \boldsymbol{p}_i\|) + \boldsymbol{M}\boldsymbol{p} + \boldsymbol{t}, \quad (3)$$

To determine the coefficients $\boldsymbol{c}_i$ and the affine components $\boldsymbol{M}$ and $\boldsymbol{t}$, we solve a set of linear equations that includes the interpolation constraints $\boldsymbol{u}_i = \boldsymbol{f}(\boldsymbol{p}_i)$, as well as the constraints $\sum_i \boldsymbol{c}_i = 0$ and $\sum_i \boldsymbol{c}_i \boldsymbol{p}_i^{\mathrm{T}} = 0$, which remove affine contributions from the radial basis functions.

Many different functions for $\phi(r)$ have been proposed [33]. After experimenting with a number of functions, we have chosen to use $\phi(r) = e^{-r/64}$, with units measured in inches.

Figure 1d shows the shape of the face model after having interpolated the set of computed 3D displacements at 13 feature points shown in Figure 1 and applied them to the entire face.

2.3 Correspondence-based shape refinement

After warping the generic face model into its new shape, we can further improve the shape by specifying additional correspondences. Since these correspondences may not be as easy to locate correctly, we do not use them to update the camera pose estimates. Instead, we simply solve for the values of the new feature points $\boldsymbol{p}_i$ using a simple least-squares fit, which corresponds to finding the point nearest the intersection of the viewing rays in 3D. We can then re-run the scattered data interpolation algorithm to update the vertices for which no correspondences are given. This process can be repeated until we are satisfied with the shape.

Figure 1e shows the shape of the face model after 99 additional correspondences have been specified. To facilitate the annotation process, we grouped vertices into polylines. Each polyline corresponds to an easily identifiable facial feature such as the eyebrow, eyelid, lips, chin, or hairline. The features can be annotated by outlining them with hand-drawn curves on each photograph where they are visible. The curves are automatically converted into a set of feature points by stepping along them using an arc-length parametrization. Figure 1b shows annotated facial features using a set of curves on the front view.

3 Texture extraction

In this section we describe the process of extracting the texture maps necessary for rendering photorealistic images of a reconstructed face model from various viewpoints.

The texture extraction problem can be defined as follows. Given a collection of photographs, the recovered viewing parameters, and the fitted face model, compute for each point $\boldsymbol{p}$ on the face model its texture color $T(\boldsymbol{p})$.

Each point $\boldsymbol{p}$ may be visible in one or more photographs; therefore, we must identify the corresponding point in each photograph and decide how these potentially different values should be combined

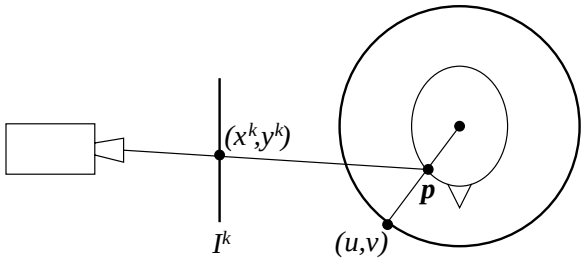


Figure 2 Geometry for texture extraction

(blended) together. There are two principal ways to blend values from different photographs: *view-independent blending*, resulting in a texture map that can be used to render the face from any viewpoint; and *view-dependent blending*, which adjusts the blending weights at each point based on the direction of the current viewpoint [9, 38]. Rendering takes longer with view-dependent blending, but the resulting image is of slightly higher quality (see Figure 3).

3.1 Weight maps

As outlined above, the texture value $T(\boldsymbol{p})$ at each point on the face model can be expressed as a convex combination of the corresponding colors in the photographs:

$$T(\boldsymbol{p}) = \frac{\sum_k m^k(\boldsymbol{p})\, I^k(x^k, y^k)}{\sum_k m^k(\boldsymbol{p})}.$$

Here, I^k is the image function (color at each pixel of the k-th photograph,) and (x^k, y^k) are the image coordinates of the projection of $\boldsymbol{p}$ onto the k-th image plane. The *weight map* $m^k(\boldsymbol{p})$ is a function that specifies the contribution of the k-th photograph to the texture at each facial surface point.

The construction of these weight maps is probably the trickiest and the most interesting component of our texture extraction technique. There are several important considerations that must be taken into account when defining a weight map:

1. *Self-occlusion:* $m^k(\boldsymbol{p})$ should be zero unless $\boldsymbol{p}$ is front-facing with respect to the k-th image and visible in it.
2. *Smoothness:* the weight map should vary smoothly, in order to ensure a seamless blend between different input images.
3. *Positional certainty:* $m^k(\boldsymbol{p})$ should depend on the "positional certainty" [24] of $\boldsymbol{p}$ with respect to the k-th image. The positional certainty is defined as the dot product between the surface normal at $\boldsymbol{p}$ and the k-th direction of projection.
4. *View similarity:* for view-dependent texture mapping, the weight $m^k(\boldsymbol{p})$ should also depend on the angle between the direction of projection of $\boldsymbol{p}$ onto the j-th image and its direction of projection in the new view.

Previous authors have taken only a subset of these considerations into account when designing their weighting functions. For example, Kurihara and Arai [24] use positional certainty as their weighting function, but they do not account for self-occlusion. Akimoto *et al.* [1] and Ip and Yin [20] blend the images smoothly, but address neither self-occlusion nor positional certainty. Debevec *et al.* [9], who describe a view-dependent texture mapping technique for modeling and rendering buildings from photographs, do address occlusion but do not account for positional certainty. (It should be noted, however, that positional certainty is less critical in photographs of buildings, since most buildings do not tend to curve away from the camera.)

To facilitate fast visibility testing of points on the surface of the face from a particular camera pose, we first render the face model using the recovered viewing parameters and save the resulting depth map from the Z-buffer. Then, with the aid of this depth map, we can quickly classify the visibility of each facial point by applying the viewing transformation and comparing the resulting depth to the corresponding value in the depth map.

3.2 View-independent texture mapping

In order to support rapid display of the textured face model from any viewpoint, it is desirable to blend the individual photographs together into a single texture map. This texture map is constructed on a virtual cylinder enclosing the face model. The mapping between the 3D coordinates on the face mesh and the 2D texture space is defined using a cylindrical projection, as in several previous papers [6, 24, 29].

For view-independent texture mapping, we will index the weight map m^k by the (u, v) coordinates of the texture being created. Each weight $m^k(u, v)$ is determined by the following steps:

1. Construct a feathered visibility map F^k for each image k. These maps are defined in the same cylindrical coordinates as the texture map. We initially set $F^k(u, v)$ to 1 if the corresponding facial point $\boldsymbol{p}$ is visible in the k-th image, and to 0 otherwise. The result is a binary visibility map, which is then smoothly ramped (feathered) from 1 to 0 in the vicinity of the boundaries [42]. A cubic polynomial is used as the ramping function.
2. Compute the 3D point $\boldsymbol{p}$ on the surface of the face mesh whose cylindrical projection is (u, v) (see Figure 2). This computation is performed by casting a ray from (u, v) on the cylinder towards the cylinder's axis. The first intersection between this ray and the face mesh is the point $\boldsymbol{p}$. (Note that there can be more than one intersection for certain regions of the face, most notably the ears. These special cases are discussed in Section 3.4.) Let $P^k(\boldsymbol{p})$ be the positional certainty of $\boldsymbol{p}$ with respect to the k-th image.
3. Set weight $m^k(u, v)$ to the product $F^k(u, v)\, P^k(\boldsymbol{p})$.

For view-independent texture mapping, we will compute each pixel of the resulting texture $T(u, v)$ as a weighted sum of the original image functions, indexed by (u, v).

3.3 View-dependent texture mapping

The main disadvantage of the view-independent cylindrical texture map described above is that its construction involves blending together resampled versions of the original images of the face. Because of this resampling, and also because of slight registration errors, the resulting texture is slightly blurry. This problem can be alleviated to a large degree by using a view-dependent texture map [9] in which the blending weights are adjusted dynamically, according to the current view.

For view-dependent texture mapping, we render the model several times, each time using a different input photograph as a texture map, and blend the results. More specifically, for each input photograph, we associate texture coordinates and a blending weight with each vertex in the face mesh. (The rendering hardware performs perspective-correct texture mapping along with linear interpolation of the blending weights.)

Given a viewing direction $\boldsymbol{d}$, we first select the subset of photographs used for the rendering and then assign blending weights to each of these photographs. Pulli *et al.* [38] select three photographs based on a Delaunay triangulation of a sphere surrounding the object. Since our cameras were positioned roughly in the same plane, we select just the two photographs whose view directions $\boldsymbol{d}^\ell$ and $\boldsymbol{d}^{\ell+1}$ are the closest to $\boldsymbol{d}$ and blend between the two.

Figure 3 Comparison between view-independent (left) and view-dependent (right) texture mapping. Higher frequency details are visible in the view-dependent rendering.

In choosing the view-dependent term $V^k(\boldsymbol{d})$ of the blending weights, we wish to use just a single photo if that photo's view direction matches the current view direction precisely, and to blend smoothly between the nearest two photos otherwise. We used the simplest possible blending function having this effect:

$$V^k(\boldsymbol{d}) = \begin{cases} \boldsymbol{d} \cdot \boldsymbol{d}^k - \boldsymbol{d}^\ell \cdot \boldsymbol{d}^{\ell+1} & \text{if } \ell \le k \le \ell + 1 \\ 0 & \text{otherwise} \end{cases}$$

For the final blending weights $m^k(\boldsymbol{p}, \boldsymbol{d})$, we then use the product of all three terms $F^k(x^k, y^k)\, P^k(\boldsymbol{p})\, V^k(\boldsymbol{d})$.

View-dependent texture maps have several advantages over cylindrical texture maps. First, they can make up for some lack of detail in the model. Second, whenever the model projects onto a cylinder with overlap, a cylindrical texture map will not contain data for some parts of the model. This problem does not arise with view-dependent texture maps if the geometry of the mesh matches the photograph properly. One disadvantage of the view-dependent approach is its higher memory requirements and slower speed due to the multi-pass rendering. Another drawback is that the resulting images are much more sensitive to any variations in exposure or lighting conditions in the original photographs.

3.4 Eyes, teeth, ears, and hair

The parts of the mesh that correspond to the eyes, teeth, ears, and hair are textured in a separate process. The eyes and teeth are usually partially occluded by the face; hence it is difficult to extract a texture map for these parts in every facial expression. The ears have an intricate geometry with many folds and usually fail to project without overlap on a cylinder. The hair has fine-detailed texture that is difficult to register properly across facial expressions. For these reasons, each of these facial elements is assigned an individual texture map. The texture maps for the eyes, teeth, and ears are computed by projecting the corresponding mesh part onto a selected input image where that part is clearly visible (the front view for eyes and teeth, side views for ears).

The eyes and the teeth are usually partially shadowed by the eyelids and the mouth respectively. We approximate this shadowing by modulating the brightness of the eye and teeth texture maps according to the size of the eyelid and mouth openings.

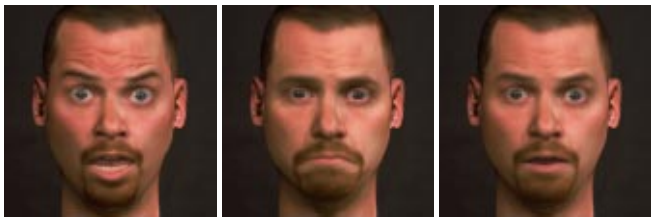


Figure 4 A global blend between "surprised" (left) and "sad" (center) produces a "worried" expression (right).

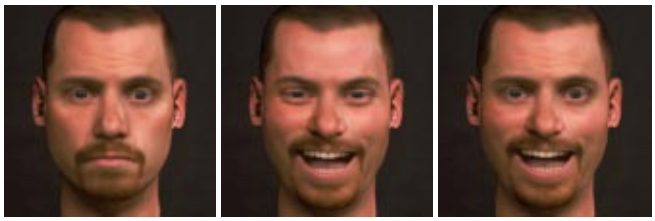


Figure 5 Combining the upper part of a "neutral" expression (left) with the lower part of a "happy" expression (center) produces a "fake smile" (right).

4 Expression morphing

A major goal of this work is the generation of continuous and realistic transitions between different facial expressions. We achieve these effects by morphing between corresponding face models.

In general the problem of morphing between arbitrary polygonal meshes is a difficult one [22], since it requires a set of correspondences between meshes with potentially different topology that can produce a reasonable set of intermediate shapes. In our case, however, the topology of all the face meshes is identical. Thus, there is already a natural correspondence between vertices. Furthermore, in creating the models we attempt to mark facial features consistently across different facial expressions, so that the major facial features correspond to the same vertices in all expressions. In this case, a satisfactory 3D morphing sequence can be obtained using simple linear interpolation between the geometric coordinates of corresponding vertices in each of the two face meshes.

Together with the geometric interpolation, we need to blend the associated textures. Again, in general, morphing between two images requires pairwise correspondences between images features [3]. In our case, however, correspondences between the two textures are implicit in the texture coordinates of the two associated face meshes. Rather than warping the two textures to form an intermediate one, the intermediate face model (obtained by geometric interpolation) is rendered once with the first texture, and again with the second. The two resulting images are then blended together. This approach is faster than warping the textures (which typically have high resolution), and it avoids the resampling that is typically performed during warping.

4.1 Multiway blend and localized blend

Given a set of facial expression meshes, we have explored ways to enlarge this set by combining expressions. The simplest approach is to use the morphing technique described above to create new facial expressions, which can be added to the set. This idea can be generalized to an arbitrary number of starting expressions by taking convex combinations of them all, using weights that apply both to the coordinates of the mesh vertices and to the values in the texture map. (Extrapolation of expressions should also be possible by allowing weights to have values outside of the interval $[0, 1]$; note, however, that such weights might result in colors outside of the allowable gamut.)

We can generate an even wider range of expressions using a localized blend of the facial expressions. Such a blend is specified by a set of blend functions, one for each expression, defined over the vertices of the mesh. These blend functions describe the contribution of a given expression at a particular vertex.

Although it would be possible to compute a texture map for each new expression, doing so would result in a loss of texture quality. Instead, the weights for each new blended expression are always factored into weights over the vertices of the original set of expressions. Thus, each blended expression is rendered using the texture map of an original expression, along with weights at each vertex, which control the opacity of that texture. The opacities are linearly interpolated over the face mesh using Gouraud shading.

4.2 Blend specification

In order to design new facial expressions easily, the user must be provided with useful tools for specifying the blending functions. These tools should satisfy several requirements. First, it should be possible to edit the blend at different resolutions. Moreover, we would like the specification process to be continuous so that small changes in the blend parameters do not trigger radical changes in the resulting expression. Finally, the tools should be intuitive to the user; it should be easy to produce a particular target facial expression from an existing set.

We explored several different ways of specifying the blending weights:

- *Global blend.* The blending weights are constant over all vertices. A set of sliders controls the mixing proportions of the contributing expressions. Figure 4 shows two facial expressions blended in equal proportions to produce a halfway blend.
- *Regional blend.* According to studies in psychology, the face can be split into several regions that behave as coherent units [11]. Usually, three regions are considered: one for the forehead (including the eyebrows), another for the eyes, and another for the lower part of the face. Further splitting the face vertically down the center results in six regions and allows for asymmetric expressions. We similarly partition the face mesh into several (softly feathered) regions and assign weights so that vertices belonging to the same region have the same weights. The mixing proportions describing a selected region can be adjusted by manipulating a set of sliders. Figure 5 illustrates the blend of two facial expressions with two regions: the upper part of the face (including eyes and forehead) and the lower part (including nose, mouth, and chin.)
- *Painterly interface.* The blending weights can be assigned to the vertices using a 3D painting tool. This tool uses a palette in which the "colors" are facial expressions (both geometry and color), and the "opacity" of the brush controls how much the expression contributes to the result. Once an expression is selected, a 3D brush can be used to modify the blending weights in selected areas of the mesh. The fraction painted has a gradual drop-off and is controlled by the opacity of the brush. The strokes are applied directly on the rendering of the current facial blend, which is updated in real-time. To improve the rendering speed, only the portion of the mesh that is being painted is re-rendered. Figure 7 illustrates the design of a debauched smile: starting with a neutral expression, the face is locally modified using three other expressions. Note that in the last step, the use of a partially transparent brush with the "sleepy" expression results in the actual geometry of the eyelids becoming partially lowered.

Figure 6 Animation interface. On the left is the "expression gallery"; on the right an expression is being designed. At the bottom expressions and poses are scheduled on the timeline.

Combining different original expressions enlarges the repertoire of expressions obtained from a set of photographs. The expressions in this repertoire can themselves be blended to create even more expressions, with the resulting expression still being representable as a (locally varying) linear combination of the original expressions.

5 User interface

We designed an interactive tool to fit a 3D face mesh to a set of images. This tool allows a user to select vertices on the mesh and mark where these curves or vertices should project on the images. After a first expression has been modeled, the set of annotations can be used as an initial guess for subsequent expressions. These guesses are automatically refined using standard correlation-based search. Any resulting errors can be fixed up by hand. The extraction of the texture map does not require user intervention, but is included in the interface to provide feedback during the modeling phase.

We also designed a keyframe animation system to generate facial animations. Our animation system permits a user to blend facial expressions and to control the transitions between these different expressions (Figure 6). The expression gallery is a key component of our system; it is used to select and display (as thumbnails) the set of facial expressions currently available. The thumbnails can be dragged and dropped onto the timeline (to set keyframes) or onto the facial design interface (to select or add facial expressions). The timeline is used to schedule the different expression blends and the changes in viewing parameters (pose) during the animation. The blends and poses have two distinct types of keyframes. Both types of keyframes are linearly interpolated with user-controlled cubic Bézier curves. The timeline can also be used to display intermediate frames at low resolution to provide a quick feedback to the animator. A second timeline can be displayed next to the composition timeline. This feature is helpful for correctly synchronizing an animation with live video or a soundtrack. The eyes are animated separately from the rest of the face, with the gaze direction parameterized by two Euler angles.

6 Results

In order to test our technique, we photographed both a man (J. R.) and a woman (Karla) in a variety of facial expressions. The photog-

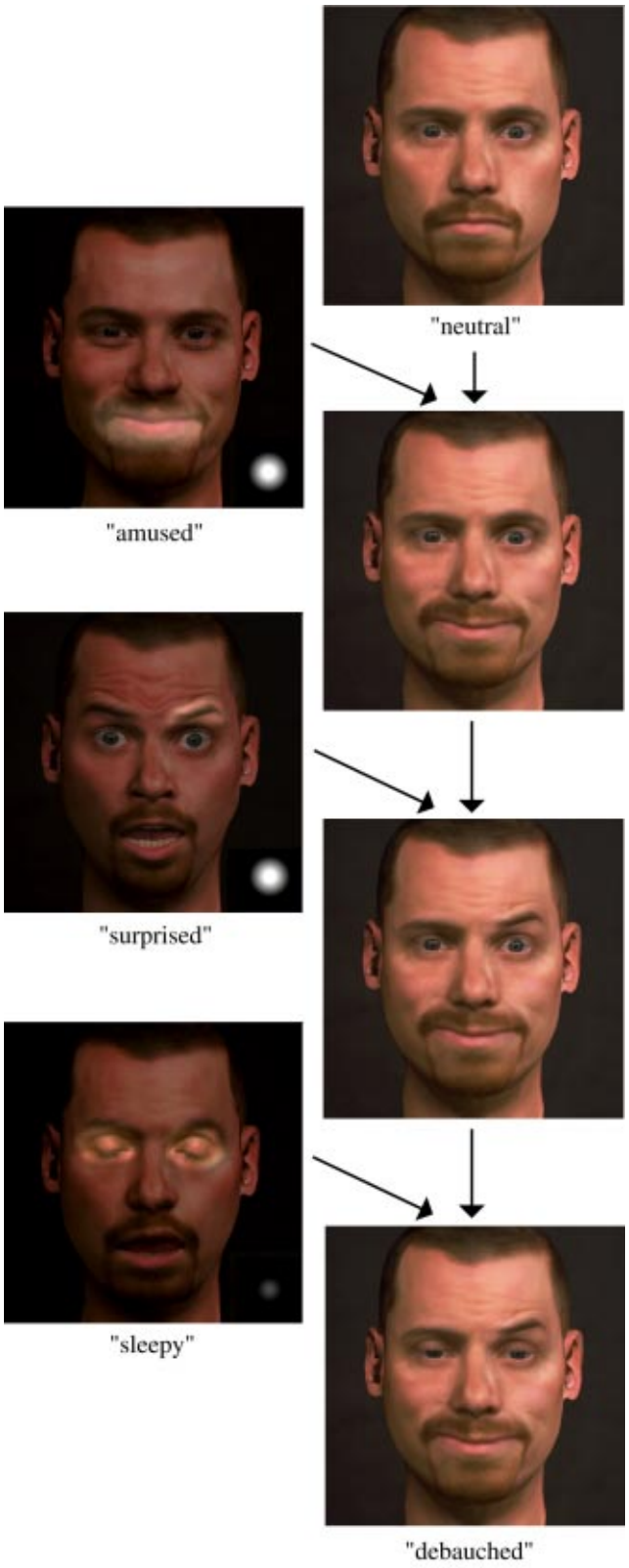


Figure 7 Painterly interface: design of a debauched smile. The right column shows the different stages of the design; the left column shows the portions of the original expressions used in creating the final expression. The "soft brush" used is shown at the bottom-right corner of each contributing expression.

raphy was performed using five cameras simultaneously. The cameras were not calibrated in any particular way, and the lenses had different focal lengths. Since no special attempt was made to illuminate the subject uniformly, the resulting photographs exhibited considerable variation in both hue and brightness. The photographs were digitized using the Kodak PhotoCD process. Five typical images (cropped to the size of the subject's head) are shown in Figure 1a.

We used the interactive modeling system described in Sections 2 and 3 to create the same set of eight face models for each subject: "happy," "amused," "angry," "surprised," "sad," "sleepy," "pained," and "neutral."

Following the modeling stage, we generated a facial animation for each of the individuals starting from the eight original expressions. We first created an animation for J. R. We then applied the very same morphs specified by this animation to the models created for Karla. For most frames of the animation, the resulting expressions were quite realistic. Figure 8 shows five frames from the animation sequence for J. R. and the purely automatically generated frames in the corresponding animation for Karla. With just a small amount of additional retouching (using the blending tools described in Section 4.2), this derivative animation can be made to look as good as the original animation for J. R.

7 Future work

The work described in this paper is just the first step towards building a complete image-based facial modeling and animation system. There are many ways to further enhance and extend the techniques that we have described:

Color correction. For better color consistency in facial textures extracted from photographs, color correction should be applied to simultaneous photographs of each expression.

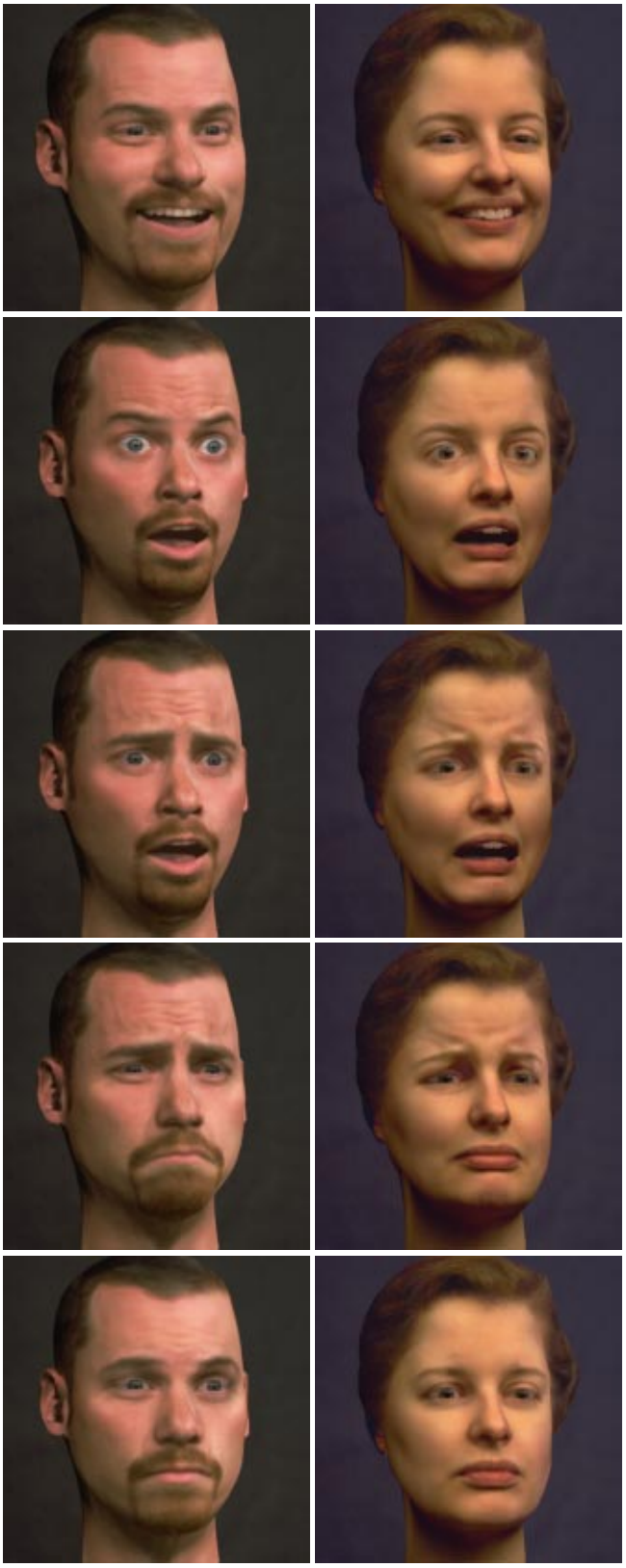
Improved registration. Some residual ghosting or blurring artifacts may occasionally be visible in the cylindrical texture map due to small misregistrations between the images, which can occur if geometry is imperfectly modeled or not detailed enough. To improve the quality of the composite textures, we could locally warp each component texture (and weight) map before blending [42].

Texture relighting. Currently, extracted textures reflect the lighting conditions under which the photographs were taken. Relighting techniques should be developed for seamless integration of our face models with other elements.

Automatic modeling. Our ultimate goal, as far as the facial modeling part is concerned, is to construct a fully automated modeling system, which would automatically find features and correspondences with minimal user intervention. This is a challenging problem indeed, but recent results on 2D face modeling in computer vision [25] give us cause for hope.

Modeling from video. We would like to be able to create face models from video or old movie footage. For this purpose, we would have to improve the robustness of our techniques in order to synthesize face meshes and texture maps from images that do not correspond to different views of the same expression. Adding anthropomorphic constraints to our face model might make up for the lack of coherence in the data [48].

Complex animations. In order to create complex animations, we must extend our vocabulary for describing facial movements beyond blending between different expressions. There are several potential ways to attack this problem. One would be to adopt an action-unit-based system such as the Facial Action Coding System



(a) (b)

Figure 8 On the left are frames from an original animation, which we created for J. R. The morphs specified in these frames were then directly used to create a derivative animation for Karla, shown on the right.

(FACS) [12]. Another possibility would be to apply modal analysis (principal component analysis) techniques to describe facial expression changes using a small number of motions [25]. Finding natural control parameters to facilitate animation and developing realistic-looking temporal profiles for such movements are also challenging research problems.

Lip-synching. Generating speech animation with our keyframe animation system would require a large number of keyframes. However, we could use a technique similar to that of Bregler *et al*. [5] to automatically lip-synch an animation to a sound-track. This would require the synthesis of face models for a wide range of visemes. For example, such database of models could be constructed using video footage to reconstruct face models automatically [17].

Performance-driven animation. Ultimately, we would also like to support performance-driven animation, i.e., the ability to automatically track facial movements in a video sequence, and to automatically translate these into animation control parameters. Our current techniques for registering images and converting them into 3D movements should provide a good start, although they will probably need to be enhanced with feature-tracking techniques and some rudimentary expression-recognition capabilities. Such a system would enable not only very realistic facial animation, but also a new level of video coding and compression techniques (since only the expression parameters would need to be encoded), as well as real-time control of avatars in 3D chat systems.

8 Acknowledgments

We would like to thank Katrin Petersen and Andrew Petty for modeling the generic face model, Cassidy Curtis for his invaluable advice on animating faces, and Joel Auslander and Jason Griffith for early contributions to this project. This work was supported by an NSF Presidential Faculty Fellow award (CCR-9553199), an ONR Young Investigator award (N00014-95-1-0728), and industrial gifts from Microsoft and Pixar.

References

[1] Takaaki Akimoto, Yasuhito Suenaga, and Richard S. Wallace. Automatic Creation of 3D Facial Models. *IEEE Computer Graphics and Applications*, 13(5):16–22, September 1993.

[2] Alias | Wavefront, Toronto, Ontario. *Alias V7.0*, 1995.

[3] Thaddeus Beier and Shawn Neely. Feature-based Image Metamorphosis. In *SIGGRAPH 92 Conference Proceedings*, pages 35–42. ACM SIGGRAPH, July 1992.

[4] Philippe Bergeron and Pierre Lachapelle. Controlling Facial Expressions and Body Movements in the Computer-Generated Animated Short "Tony De Peltrie". In *SIGGRAPH 85 Advanced Computer Animation seminar notes*. July 1985.

[5] Christoph Bregler, Michele Covell, and Malcolm Slaney. Video Rewrite: Driving Visual Speech with Audio. In *SIGGRAPH 97 Conference Proceedings*, pages 353–360. ACM SIGGRAPH, August 1997.

[6] David T. Chen, Andrei State, and David Banks. Interactive Shape Metamorphosis. In *1995 Symposium on Interactive 3D Graphics*, pages 43–44. ACM SIGGRAPH, April 1995.

[7] Chang S. Choi, Kiyoharu, Hiroshi Harashima, and Tsuyoshi Takebe. Analysis and Synthesis of Facial Image Sequences in Model-Based Image Coding. In *IEEE Transactions on Circuits and Systems for Video Technology*, volume 4, pages 257 – 275. June 1994.

[8] Cyberware Laboratory, Inc, Monterey, California. *4020/RGB 3D Scanner with Color Digitizer*, 1990.

[9] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-Based Approach. In *SIGGRAPH 96 Conference Proceedings*, pages 11–20. ACM SIGGRAPH, August 1996.

[10] Eben Ostby, Pixar Animation Studios. Personal communication, January 1997.

[11] Paul Ekman and Wallace V. Friesen. *Unmasking the Face. A guide to recognizing emotions fron facial clues*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975.

[12] Paul Ekman and Wallace V. Friesen. *Manual for the Facial Action Coding System*. Consulting Psychologists Press, Inc., Palo Alto, California, 1978.

[13] Irfan Essa, Sumit Basu, Trevor Darrell, and Alex Pentland. Modeling, Tracking and Interactive Animation of Faces and Heads Using Input from Video. In *Computer Animation Conference*, pages 68–79. June 1996.

[14] Gary Faigin. *The Artist's Complete Guide to Facial Expression*. Watson-Guptill Publications, New York, 1990.

[15] Olivier Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, Cambridge, Massachusetts, 1993.

[16] G. Golub and C. F. Van Loan. *Matrix Computation, third edition*. The John Hopkins University Press, Baltimore and London, 1996.

[17] Brian Guenter, Cindy Grimm, Daniel Wood, Henrique Malvar, and Frédéric Pighin. Making Faces. In *SIGGRAPH 98 Conference Proceedings*. ACM SIGGRAPH, July 1998.

[18] Pat Hanrahan and Wolfgang Krueger. Reflection from Layered Surfaces Due to Subsurface Scattering. In *SIGGRAPH 93 Conference Proceedings*, volume 27, pages 165–174. ACM SIGGRAPH, August 1993.

[19] Bright Star Technologies Inc. *Beginning Reading Software*. Sierra On-Line, Inc., 1993.

[20] Horace H. S. Ip and Lijun Yin. Constructing a 3D Individualized Head Model from Two Orthogonal Views. *The Visual Computer*, 12:254–266, 1996.

[21] Gregory Ward J., Francis M. Rubinstein, and Robert D. Clear. A Ray Tracing Solution for Diffuse Interreflection. In *SIGGRAPH 88 Conference Proceedings*, volume 22, pages 85–92. August 1988.

[22] James R. Kent, Wayne E. Carlson, and Richard E. Parent. Shape Transformation for Polyhedral Objects. In *SIGGRAPH 92 Proceedings Conference*, volume 26, pages 47–54. ACM SIGGRAPH, July 1992.

[23] Rolf M. Koch, Markus H. Gross, Friedrich R. Carls, Daniel F. von Büren, George Fankhauser, and Yoav I. H. Parish. Simulating Facial Surgery Using Finite Element Methods. In *SIGGRAPH 96 Conference Proceedings*, pages 421–428. ACM SIGGRAPH, August 1996.

[24] Tsuneya Kurihara and Kiyoshi Arai. A Transformation Method for Modeling and Animation of the Human Face from Photographs. In Nadia Magnenat Thalmann and Daniel Thalmann, editors, *Computer Animation 91*, pages 45–58. Springer-Verlag, Tokyo, 1991.

[25] A. Lanitis, C. J. Taylor, and T. F. Cootes. A Unified Approach for Coding and Interpreting Face Images. In *Fifth International Conference on Computer Vision (ICCV 95)*, pages 368–373. Cambridge, Massachusetts, June 1995.

[26] C. L. Lawson and R. J. Hansen. *Solving Least Squares Problems*. Prentice-Hall, Englewood Cliffs, 1974.

[27] Seung-Yong Lee, Kyung-Yong Chwa, Sung Yong Shin, and George Wolberg. Image Metamorphosis Using Snakes and Free-Form Deformations. In *SIGGRAPH 95 Conference Proceedings*, pages 439–448. ACM SIGGRAPH, August 1995.

[28] Seung-Yong Lee, George Wolberg, Kyung-Yong Chwa, and Sung Yong Shin. Image Metamorphosis with Scattered Feature Constraints. *IEEE Transactions on Visualization and Computer Graphics*, 2(4), December 1996.

[29] Yuencheng Lee, Demetri Terzopoulos, and Keith Waters. Realistic Modeling for Facial Animation. In *SIGGRAPH 95 Conference Proceedings*, pages 55–62. ACM SIGGRAPH, August 1995.

[30] Yuencheng C. Lee, Demetri Terzopoulos, and Keith Waters. Constructing Physics-Based Facial Models of Individuals. In *Proceedings of Graphics Interface 93*, pages 1–8. May 1993.

[31] Francis H. Moffitt and Edward M. Mikhail. *Photogrammetry*. Harper & Row, New York, 3 edition, 1980.

[32] Shree K. Nayar, Katsushi Ikeuchi, and Takeo Kanade. Shape from Interreflections. *International Journal of Computer Vision*, 6:173–195, 1991.

[33] Gregory M. Nielson. Scattered Data Modeling. *IEEE Computer Graphics and Applications*, 13(1):60–70, January 1993.

[34] Frederic I. Parke. Computer Generated Animation of Faces. *Proceedings ACM annual conference.*, August 1972.

[35] Frederic I. Parke. A Parametric Model for Human Faces. PhD thesis, University of Utah, Salt Lake City, Utah, December 1974. UTEC-CSc-75-047.

[36] Frederic I. Parke and Keith Waters. *Computer Facial Animation*. A K Peters, Wellesley, Massachusetts, 1996.

[37] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, England, second edition, 1992.

[38] Kari Pulli, Michael Cohen, Tom Duchamp, Hugues Hoppe, Linda Shapiro, and Werner Stuetzle. View-based rendering: Visualizing real objects from scanned range and color data. In *Proc. 8th Eurographics Workshop on Rendering*. June 1997.

[39] Steven M. Seitz and Charles R. Dyer. View Morphing. In *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 21–30. ACM SIGGRAPH, August 1996.

[40] Chester C. Slama, editor. *Manual of Photogrammetry*. American Society of Photogrammetry, Falls Church, Virginia, fourth edition, 1980.

[41] Richard Szeliski and Sing Bing Kang. Recovering 3D Shape and Motion from Image Streams using Nonlinear Least Squares. *Journal of Visual Communication and Image Representation*, 5(1):10–28, March 1994.

[42] Richard Szeliski and Heung-Yeung Shum. Creating Full View Panoramic Image Mosaics and Texture-Mapped Models. In *SIGGRAPH 97 Conference Proceedings*, pages 251–258. ACM SIGGRAPH, August 1997.

[43] Demetri Terzopoulos and Keith Waters. Physically-based Facial Modeling, Analysis, and Animation. *Journal of Visualization and Computer Animation*, 1(4):73–80, March 1990.

[44] Kristinn R. Thórisson. Gandalf: An Embodied Humanoid Capable of Real-Time Multimodal Dialogue with People. In *First ACM International Conference on Autonomous Agents*. 1997.

[45] Michael W. Vannier, Jeffrey F. Marsh, and James O. Warren. Three-dimentional Computer Graphics for Craniofacial Surgical Planning and Evaluation. In *SIGGRAPH 83 Conference Proceedings*, volume 17, pages 263–273. ACM SIGGRAPH, August 1983.

[46] Keith Waters. A Muscle Model for Animating Three-Dimensional Facial Expression. In *SIGGRAPH 87 Conference Proceedings)*, volume 21, pages 17–24. ACM SIGGRAPH, July 1987.

[47] Lance Williams. Performance-Driven Facial Animation. In *SIGGRAPH 90 Conference Proceedings*, volume 24, pages 235–242. August 1990.

[48] Z. Zhang, K. Isono, and S. Akamatsu. Euclidean Structure from Uncalibrated Images Using Fuzzy Domain Knowledge: Application to Facial Images Synthesis. In *Proc. International Conference on Computer Vision (ICCV'98)*. January 1998.

A Least squares for pose recovery

To solve for a subset of the parameters given in Equation (2), we use linear least squares. In general, given a set of linear equations of the form

$$\boldsymbol{a}_j \cdot \boldsymbol{x} - b_j = 0, \tag{4}$$

we solve for the vector $\boldsymbol{x}$ by minimizing

$$\sum_j (\boldsymbol{a}_j \cdot \boldsymbol{x} - b_j)^2. \tag{5}$$

Setting the partial derivative of this sum with respect to $\boldsymbol{x}$ to zero, we obtain

$$\sum_j (\boldsymbol{a}_j \boldsymbol{a}_j^T)\boldsymbol{x} - b_j \boldsymbol{a}_j = 0, \tag{6}$$

i.e., we solve the set of *normal equations* [16]

$$\left(\sum_j \boldsymbol{a}_j \boldsymbol{a}_j^T \right) \boldsymbol{x} = \sum_j b_j \boldsymbol{a}_j. \tag{7}$$

More numerically stable methods such as *QR* decomposition or Singular Value Decomposition [16] can also be used to solve the least squares problem, but we have not found them to be necessary for our application.

To update one of the parameters, we simply pull out the relevant linear coefficient $\boldsymbol{a}_j$ and scalar value b_j from Equation (2). For example, to solve for $\boldsymbol{p}_i$, we set

$$\begin{aligned} \boldsymbol{a}_{2k+0} &= w_i^k (x_i^k \eta^k \boldsymbol{r}_z^k - s^k \boldsymbol{r}_x^k), & b_{2k+0} &= w_i^k (s^k t_x^k - x_i^k) \\ \boldsymbol{a}_{2k+1} &= w_i^k (y_i^k \eta^k \boldsymbol{r}_z^k - s^k \boldsymbol{r}_y^k), & b_{2k+1} &= w_i^k (s^k t_y^k - y_i^k). \end{aligned}$$

For a scalar variable like s^k, we obtain scalar equations

$$\begin{aligned} \boldsymbol{a}_{2k+0} &= w_i^k (\boldsymbol{r}_x^k \cdot \boldsymbol{p}_i + t_x^k), & b_{2k+0} &= w_i^k \left(x_i^k + x_i^k \eta^k (\boldsymbol{r}_z^k \cdot \boldsymbol{p}_i) \right) \\ \boldsymbol{a}_{2k+1} &= w_i^k (\boldsymbol{r}_y^k \cdot \boldsymbol{p}_i + t_y^k), & b_{2k+1} &= w_i^k \left(y_i^k + y_i^k \eta^k (\boldsymbol{r}_z^k \cdot \boldsymbol{p}_i) \right). \end{aligned}$$

Similar equations for $\boldsymbol{a}_j$ and b_j can be derived for the other parameters t_x^k, t_y^k, and η^k. Note that the parameters for a given camera k or 3D point i can be recovered independently of the other parameters.

Solving for rotation is a little trickier than for the other parameters, since $\boldsymbol{R}$ must be a valid rotation matrix. Instead of updating the elements in $\boldsymbol{R}_k$ directly, we replace the rotation matrix $\boldsymbol{R}^k$ with $\tilde{\boldsymbol{R}}\boldsymbol{R}^k$ [42], where $\tilde{\boldsymbol{R}}$ is given by Rodriguez's formula [15]:

$$\tilde{\boldsymbol{R}}(\hat{\boldsymbol{n}}, \theta) = \boldsymbol{I} + \sin\theta \boldsymbol{X}(\hat{\boldsymbol{n}}) + (1 - \cos\theta)\boldsymbol{X}^2(\hat{\boldsymbol{n}}), \tag{8}$$

where θ is an incremental rotation angle, $\hat{\boldsymbol{n}}$ is a rotation axis, and $\boldsymbol{X}(\boldsymbol{v})$ is the cross product operator

$$\boldsymbol{X}(\boldsymbol{v}) = \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix}. \tag{9}$$

A first order expansion of $\tilde{\boldsymbol{R}}$ in terms of the entries in $\boldsymbol{v} = \theta\hat{\boldsymbol{n}} = (v_x, v_y, v_z)$ is given by $\boldsymbol{I} + \boldsymbol{X}(\boldsymbol{v})$.

Substituting into Equation (2) and letting $\boldsymbol{q}_i = \boldsymbol{R}^k \boldsymbol{p}_i$, we obtain

$$\begin{aligned} w_i^k \left(x_i^k + x_i^k \eta^k (\tilde{\boldsymbol{r}}_z^k \cdot \boldsymbol{q}_i) - s^k (\tilde{\boldsymbol{r}}_x^k \cdot \boldsymbol{q}_i + t_x^k) \right) &= 0 \\ w_i^k \left(y_i^k + y_i^k \eta^k (\tilde{\boldsymbol{r}}_z^k \cdot \boldsymbol{q}_i) - s^k (\tilde{\boldsymbol{r}}_y^k \cdot \boldsymbol{q}_i + t_y^k) \right) &= 0, \end{aligned} \tag{10}$$

where $\tilde{\boldsymbol{r}}_x^k = (1, -v_z, v_y)$, $\tilde{\boldsymbol{r}}_y^k = (v_z, 1, -v_x)$, $\tilde{\boldsymbol{r}}_z^k = (-v_y, v_x, 1)$, are the rows of $[\boldsymbol{I} + \boldsymbol{X}(\boldsymbol{v})]$. This expression is linear in (v_x, v_y, v_z), and hence leads to a 3×3 set of normal equations in (v_x, v_y, v_z). Once the elements of $\boldsymbol{v}$ have been estimated, we can compute θ and $\hat{\boldsymbol{n}}$, and update the rotation matrix using

$$\boldsymbol{R}^k \leftarrow \tilde{\boldsymbol{R}}(\hat{\boldsymbol{n}}^k, \theta^k)\boldsymbol{R}^k.$$

Universal Capture – Image-based Facial Animation for “The Matrix Reloaded”

George Borshukov, Dan Piponi, Oystein Larsen, J.P.Lewis, Christina Tempelaar-Lietz
ESC Entertainment

Introduction

The VFX R&D stage for *The Matrix Reloaded* was kicked off in January 2000 with the challenge to create realistic human faces. We believed that traditional facial animation approaches like muscle deformers or blend shapes would simply never work, both because of the richness of facial movement and because of the human viewer’s extreme sensitivity to facial nuances. Our task was further complicated as we had to recreate familiar actors such as Keanu Reeves and Lawrence Fishburne. Our team had been very successful at applying image-based techniques for photorealistic film set/location rendering, so we decided to approach the problem from the image-based side again. We wanted to produce a 3-d recording of the real actor's performance and be able to play it back from different angles and under different lighting conditions. Just as we can extract geometry, texture, or light from images, we are now able to extract movement. Universal Capture combines two powerful computer vision techniques: optical flow and photogrammetry.

HiDef Capture Setup

We used a carefully placed array of five synchronized cameras that captured the actor's performance in ambient lighting. For the best image quality we deployed a sophisticated arrangement of Sony/Panavision HDW-F900 cameras and computer workstations that captured the images in uncompressed digital format straight to hard disks at data rates close to 1G/sec.

Optical Flow + Photogrammetry

We use optical flow to track each pixel's motion over time in each camera view. The result of this process is then combined with a cyberscan model of a neutral expression of the actor and with photogrammetric reconstruction of the camera positions. The algorithm works by projecting a vertex of the model into each of the cameras and then tracking the motion of that vertex in 2-d using the optical flow where at each frame the 3-d position is estimated using triangulation. The result is an accurate reconstruction of the path of each vertex though 3-d space over time.

Keyshaping, Adapt, Removing Global Motion

Optical flow errors can accumulate over time, causing an undesirable drift in the 3-d reconstruction. To minimize the drift we make use of reverse optical flow. On this production the problem was eliminated by introducing a manual keyshaping step: when the flow error becomes unacceptably large the geometry is manually corrected and the correction is then algorithmically propagated to previous frames.

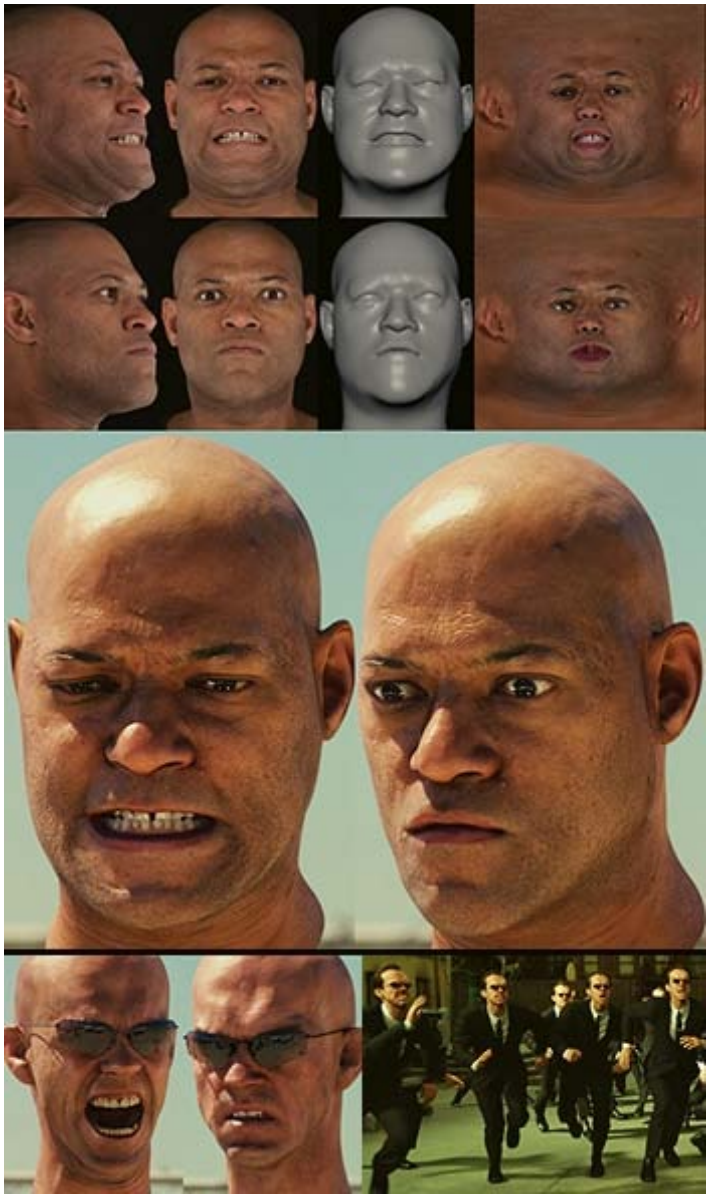
The reconstructed motion contains the global "rigid" head movement. In order to attach facial performances to CG bodies or blend between different performances this movement must be removed. We estimate the rigid transformation using a least squares fit of a neutral face and then subtract this motion to obtain the non-rigid deformation.

Texture Map Extraction

No believable facial rendering can be done without varying the face texture over time. The fact that we did not use any markers on the face to assist feature tracking gave us the important advantage that we could combine the images from the multiple camera views over time to produce animated seamless UV color maps capturing important textural variation across the face, such as the forming of fine wrinkles or changes in color due to strain, in high-res detail on each side of the face.

Rendering

Although the extracted facial animation had most of the motion nuances it lacked the small-scale surface detail like pores and wrinkles. We obtained that by using a highly detailed 100-micron scan of the actor’s face. The detail is then extracted in a bump (displacement) map. Dynamic wrinkles were identified by image processing on the texture maps; these are then isolated and layered over the static bump map. We then combine these with image-based skin BRDF estimation, subsurface scattering approximation, and real-world lighting reconstruction for the highly photorealistic human face renderings below.



Acknowledgments: Steve Avoujageli, Ken Faiman, Steve Rembuskos, Mike Dalzell, John Llewellyn, Ryan Schnizlein, Paul Ryan, John Jack, Kim Libreri, and John Gaeta

Analysis and Synthesis of Facial Expressions with Hand-Generated Muscle Actuation Basis

Byoungwon Choe* Hyeong-Seok Ko

School of Electrical Engineering and Computer Science
Seoul National University, Seoul, KOREA

Abstract

We present a performance-driven facial animation system for analyzing captured expressions to find muscle actuation and synthesizing expressions with the actuation values. Significantly different approach of our work is that we let artists sculpt the initial draft of the actuation basis—the basic facial shapes corresponding to the isolated actuation of individual muscles, instead of calculating skin surface deformation entirely relying on the mathematical models such as finite element methods. We synthesize expressions by linear combinations of the basis elements, and analyze expressions by finding the weights for the combinations. Even though the hand-generated actuation basis represents the essence of the subject's characteristic expressions, it is not accurate enough to be used in the subsequent computational procedures. We also describe an iterative algorithm to increase the accuracy of the actuation basis. The experimental results suggest that our artist-in-the-loop method produces more predictable and controllable outcome than pure mathematical models, thus can be a quite useful tool in animation productions.

1. Introduction

Since Williams' pioneering work on performance-driven facial animation [23], applying facial expressions from human faces to computer-generated characters has been widely studied [9, 3, 24, 12, 19]. To control facial movement, facial expressions were analyzed into the position of feature points [9, 3, 24] or the weights for blending pre-modeled expressions [12, 19]. Another fascinating approach, which we took in this work, is finding muscle actuation parameters from facial expressions [21, 6, 1]. Expressions can be easily modified by editing muscle actuation curves [1], and the actuation values can be converted to other control parameters such as the values of Actuation Units in Facial Action Coding System [4] without much effort.

Terzopoulos *et al.* [21] and Essa *et al.* [6] analyzed the expressions recorded in video footage into muscle actuation values for *facial expression recognition*. They also synthesized facial expressions with the actuation values. The synthetic expressions, however, showed only conspicuous ones such as 'opening mouth' or 'raising eyebrows', which were not yet to be used in high quality animation production. Recently, Choe *et al.* [1] proposed an algorithm to find muscle actuation values from the trajectory of feature points generated by an optical capture system. They could reproduce delicate facial movements by extracting complicated set of muscle actuation values with a linear finite element model, and showed the possibility of practical use in character animation. Still, the heuristic muscle model could misinterpret the original expressions, and simplified finite element model occasionally produced unnatural artifacts in skin deformation.

In this work, instead of relying entirely on the mathematical models to compute the 3D facial shape, we include the artists' modeling capability as an integral part of the method. According to our previous tests, the result of pure mathematical modeling was usually distant from what was *expected*.[1] Such expectation cannot be quantitatively stated; we thought that an artist may be able to form the expected (or desired) facial shape. Thus we made the artists sculpt manually a set of expressions called the *muscle actuation basis*, and let the computer program synthesize expressions based on the basis elements. Each element of the actuation basis corresponds to the facial shape when a single expression muscle is fully actuated and the rest are left relaxed.

We can synthesize a facial expression by the linear combination of the basis elements on the same principle as the linear muscle model [1]. Then our algorithm is basically re-

* 133-316, Seoul National University, Shillim-dong, Gwanak-gu, Seoul, 151-742, KOREA. drei@graphics.snu.ac.kr

[1] Some of the reasons might be that we could not calibrate the muscle size and layout of the computer model with those of the subject being captured, and we made too many simplifying assumptions to use mathematical models.

duced to the methods that synthesize expressions by blending pre-modeled expressions, which was experimented by Kouadio *et al.* [12] and Pighin *et al.* [18, 19]. Our method is different from theirs in the pre-modeled expression set: we use an artificial but functional set of expressions instead of using real human expression samples such as 'happiness' or 'sadness'. Using the actuation basis rather than real human expression samples has an important consequence. The elements in the actuation basis are *orthogonal* to each other, and form a meaningful basis for the facial expression space—the actuation basis can produce (or in mathematical terms, *span*) the complete set of human expressions. When real human expressions are used, on the other hand, the linear combinations of them cannot generally guarantee to produce the complete set of expressions.[2]

We can summarize our facial animation process into two major steps: **modeling** to set up the neutral face and the actuation basis of a subject and **analysis** to find muscle contractions from the subject's performance using the actuation basis. We can synthesize expressions by applying the analyzed muscle contractions to any computer model with an equivalent muscle structure.

In order to model the actuation basis, we first obtain the neutral face of the subject using a 3D scanning device. Starting from the neutral face, we let an artist sculpt the basis elements considering the human facial anatomy [2]. The work of Faigin [7], which illustrates the facial shape corresponding to the actuation of each individual muscle, serves a good guide for the job. It could be expected that the first hand-generated draft would not give a satisfactory result. Moreover, considering that the accuracy of the actuation basis greatly affects the result of the **analysis**, we need to develop a procedure for improving the basis. The improvement procedure (described in Section 4.2), in turn, refers to the result of the **analysis** on some trial data; the procedure takes the form of fixed point iteration between **modeling** and **analysis**.

Once the actuation basis of a subject is ready, we can start analyzing the expressions captured from the subject. We approximate each frame of the facial performance by a linear combination of the basis elements. Finding the best approximation can be formulated as a *constrained quadratic programming*, and the coefficients in the resulting solution are interpreted as the muscle contraction values.

The rest of this paper is organized as follows. Section 2 reviews related work in facial animation. Section 3 and Section 4 present the **modeling** and **analysis** procedures respectively. Section 5 shows the experimental results of our method, and Section 6 concludes the paper.

[2]There have been efforts to resolve the correlation among human expression samples and map the expressions into an orthogonal domain [14]. A popular domain studied first in psychology was a two-dimensional space represented by *pleasure* and *arousal* axes [8]. However, the quantitative use of the parameters (e.g., for expression synthesis) does not seem suitable since the dimension is quite limited and assigning the coordinate values is done in a subjective manner.

2. Background

This section reviews the state-of-the-art techniques on performance-driven facial animation and muscle-based facial modeling. More topics on facial modeling and animation can be found in [17].

Williams [23] introduced a performance-driven facial animation system which synthesized expressions by changing texture coordinates calculated from the position of feature points on the face. Guenter *et al.* [9] captured both the 3D geometry and shading information of a human face, and reproduced photorealistic expressions. Eisert and Girod [3] modeled a face with a triangular B-spline surface, and analyzed facial expressions by estimating the facial animation parameters of MPEG-4 standard. Pighin *et al.* [18] reconstructed the geometry and texture of an individual face from five photo images of the subject. With this method, they modeled basic expressions such as 'joy' or 'surprise', and synthesized novel expressions by blending them. The result was photo-realistic, showing detailed wrinkles and creases. Later, they proposed an algorithm to find the blending weights from the video recording of a performance [19]. Kouadio *et al.* [12] animated a synthetic character by the linear combination of previously modeled 3D facial expressions by extracting the interpolation weights from the feature points traced by an optical capture device.

Waters [22] introduced an anatomically based muscle model which was kinematically formulated. Terzopoulos *et al.* [20, 13] represented the mesh of the skin surface by a mass-spring model, and calculated skin deformation due to muscle actuation. Koch *et al.* predicted the geometry of skin surface due to the skull shape change using a finite-element model [11], and synthesized expressions by embedding expression muscles [10].

Terzopoulos and Waters [21] developed a method to extract muscle contractions from the expressions recorded in video footage based on a dynamic muscle model. Essa *et al.* [5, 6] developed a system to estimate muscle actuation corresponding to a given expression using feedback control theory. Choe *et al.* [1] calculated muscle actuation values based on the finite element skin model and linear muscle model.

3. Modeling Muscle Actuation Basis

Muscle actuation basis is a set of expressions $\{\mathbf{E}_1, \mathbf{E}_2, \ldots, \mathbf{E}_m\}$, each of which represents the 3D facial shape when a single expression muscle is fully actuated and the others are relaxed. Figure 1 shows an example of the actuation basis.

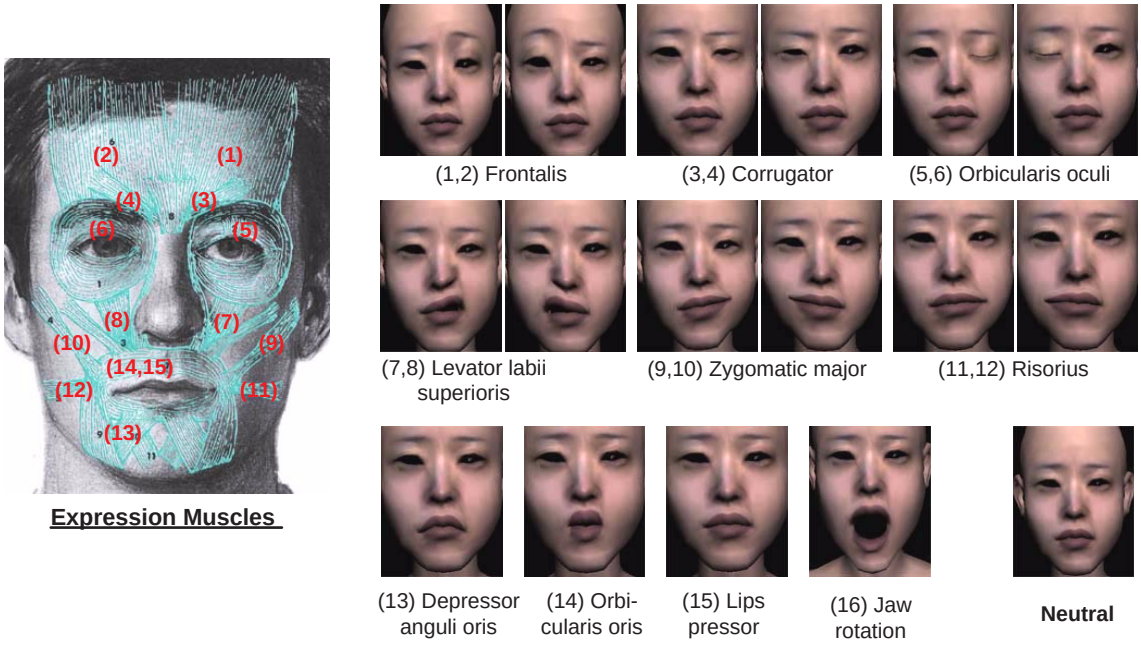


Figure 1. Expression muscles and the corresponding basis elements in the actuation basis.

Once we have the actuation basis, we can synthesize facial expressions by linear combinations of the basis elements if we assume the linear muscle model [1]. Let $\mathbf{E}_0$ denote the neutral expression—the position of about 1,500 vertices that constitute the facial surface. Let $\mathbf{e}_i = \mathbf{E}_i - \mathbf{E}_0$ ($i = 1, 2, \ldots, m$) be the difference between the basis element and the neutral expression, where m is the number of the basis elements. When the muscle contractions $x_1, x_2, \ldots, x_m$ are given, we synthesize an expression $\mathbf{E}$ by

$$\mathbf{E} = \mathbf{E}_0 + \sum_{i=1}^{m} x_i \mathbf{e}_i. \qquad (1)$$

We normally expect the muscle contractions have the value in $[0, 1]$ since each basis element embodies the full actuation of an expression muscle.

In this work, we used an actuation basis of 16 elements as shown in Figure 1: *six* for the muscles in the upper region around the eyebrows (Figure 1 (1)∼(6)), *ten* for the muscles in the lower region around the mouth (Figure 1 (7)∼(16)). We get the reduced set of 16 basis elements to represent the operation of not less than 26 expression muscles in the human face. The operation of several muscles can be merged into a single basis element if they are dependent on each other, and the operation of a single muscle should be represented by multiple basis elements if the actuation of the muscle can produce multiple distinct shapes:

- **Merging:** We merge the operation of the muscles into a single basis element if they usually actuate simultaneously. For example, we merge the three muscles *Levator labii superioris alaeque nasi*, *Levator labii superioris*, and *Zygomatic minor* which are known as the sneering muscles (see Figure 1 (7, 8)) into the single basis element *Levator labii superioris*. The basis elements *Corrugator* (Figure 1 (3, 4)), *Risorius* (Figure 1 (11, 12)), and *Depressor anguli oris* (Figure 1 (13)) are also the results of merging the operation of two or three muscles.

- **Mouth:** The operation of *Orbicularis oris* around the mouth is very complicated, and the full actuation of the muscle can generate many different shapes. In this work, we created two basis elements to represent the operation of the muscle: normal *Orbicularis oris* which corresponds to the mouth shape when pronouncing /u/ sound (Figure 1 (14)), and the *Lips pressor* which corresponds to the protruded (upset) mouth (Figure 1 (15)).[3] Gentle closing of the mouth is covered by the neutral expression $\mathbf{E}_0$.

- **Eyes:** *Orbicularis oculi*, the sphincter muscle at the eyes, consists of the palpebral and orbital parts. In this

[3] *Orbicularis oris* was an obvious choice for the basis, but the inclusion of *Lips pressor* was based upon our experiences: without the *Lips pressor*, we observed the elements *Risorius* and *Orbicularis oris* had to combine frequently to produce the shape of *Lips pressor*, which was quite unnatural in the operation of human expression muscles.

work, we implemented only the operation of the palpebral part (gentle closing of the eyes) as a basis element (Figure 1 (5, 6)). Therefore emphatic closing of the eyes cannot be generated.

We let artists model the basis elements considering the size and location of expression muscles [2]. Faigin [7] illustrated the facial expressions resulting from the actuation of a single or multiple expression muscles, which served an excellent guide to the modeling job. The actuation basis only used for expression *synthesis* does not need to come from a human subject. However, the actuation basis for expression *analysis* should accurately reflect the operation of expression muscles of the subject because it affects drastically the result of expression analysis (Section 4.1). Therefore artists were asked to watch carefully the video recording of the subject (or the *training data* in Section 4.2) where the subject was asked to make all kinds of expressions including the extreme actuation of each muscle.

It would be impractical to assume that the hand-generated actuation basis is accurate enough. Fortunately, there is a way to evaluate the given basis: we simply run the expression analysis procedure on the training data, then we can infer that the basis is not accurate when the resulting muscle contractions go far beyond the expected range $[0, 1]$. In such a case, we ask the artists to re-model the basis elements. We repeat the step until a reasonable basis is obtained. However, it would be still impractical to assume that the resulting basis is accurate enough to start our computational steps of expression analysis. We present an algorithm that improves further the actuation basis (at this time without the help of artists) by taking iterations between the expression analysis and basis modification procedures. The algorithm cannot be fully described until the expression analysis procedure is understood, so the description is deferred to the end of the next section.

4. Analysis of Facial Expressions

This section presents the procedures to extract muscle contractions from facial performances, and shows how the procedure can be used for improving the hand-generated actuation basis.

4.1. Extracting Muscle Contractions

We analyze the facial expressions by finding muscle contractions to reproduce optimally the marker trajectories generated by optical capture systems. We improved the algorithm proposed by Choe *et al.* [1].

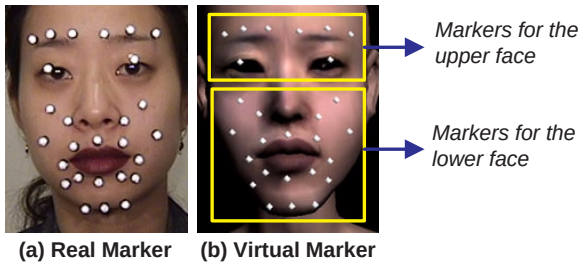


Figure 2. Real markers and corresponding virtual markers on the synthetic model.

4.1.1 Coordinate Alignment

While the 3D geometry of the synthetic face is resolved in its own local coordinate system $\{M\}$ (*model coordinate system*), the marker points in the performance data are resolved in another coordinate system $\{P\}$ (*performance coordinate system*). Before calculating the muscle contractions, we first have to transform the marker points from the performance coordinate system to the model coordinate system. We assume the transform from $\{P\}$ to $\{M\}$ is an affine (similarity) transform with scale s, rotation $\mathbf{R}$, and translation $\mathbf{t}$. We calculate the transform only once at the first frame of the performance data, and apply the same transform to all the other frames. In the following, we use the notation ${}^{Z}\mathbf{p}$ to denote the 3D coordinate of a point $\mathbf{p}$ resolved in the coordinate system $\{Z\}$

Let the position of the j-th marker be ${}^{P}\mathbf{p}_j$. We define the *virtual marker* ${}^{M}\mathbf{p}_j$ to be the corresponding point in $\{M\}$. Figure 2 shows the real and virtual markers. We want to find s, $\mathbf{R}$, and $\mathbf{t}$ that satisfy the following equations,

$$ {}^{M}\mathbf{p}_j = s\mathbf{R}\,{}^{P}\mathbf{p}_j + \mathbf{t}, \quad (j = 1, 2, \ldots, n) $$

where n is the number of markers. To initiate the computation, we first manually mark the virtual marker positions looking at both the first frame of the performance and the 3D model of the face. Then we solve the linear least square problems to find s, $\mathbf{t}$, and $\mathbf{R}$ sequentially, and repeat the procedure until the least square error is saturated.

The accuracy of s, $\mathbf{R}$, and $\mathbf{t}$ solved from the above procedure is at best limited to the accuracy of hand-marked position of the virtual markers. We note that, once we have s, $\mathbf{R}$, and $\mathbf{t}$, then the real markers can now be transformed to the model coordinate system. But the resulting points may not lie exactly on the surface of the 3D face model. By slightly adjusting the points along the normal directions, we can make the points lie on the surface, and get the next estimation of the (virtual) markers. We can further improve the accuracy of s, $\mathbf{R}$, and $\mathbf{t}$ by repeating the least square procedure with the new position of the markers.

The above assumes that the facial shape at the first frame of the performance is the same with the pre-modeled neutral face. Therefore we asked the actors to make a consistent neutral expression at the beginning of each performance capture.

4.1.2 Calculating Muscle Contractions

The final position of the virtual marker j in the above procedure will be a point within one of the triangles that constitute the facial geometry. Thus the marker point $\mathbf{p}_j$ can be encoded by the triangle index and the relative position within the triangle (barycentric coordinates), which do not depend on the subsequent deformation of the face. But the marker point will have different 3D position depending on the current shape of the face.

Let $\mathbf{d}_{ij}$ be the displacement of $\mathbf{p}_j$ at basis element $\mathbf{E}_i$ from $\mathbf{p}_j$ at neutral expression $\mathbf{E}_0$. From the synthesis equation (1), if muscle contractions x_i are given, the total displacement $\mathbf{d}_j$ of $\mathbf{p}_j$ is given by

$$\mathbf{d}_j = \sum_{i=1}^{m} x_i \mathbf{d}_{ij} \ .$$

We find the muscle contractions so that $\mathbf{d}_j$ is closest to the observed displacement $\hat{\mathbf{d}}_j$ from the performance by minimizing

$$\sum_{j=1}^{n} |\hat{\mathbf{d}}_j - \mathbf{d}_j|^2 = \sum_{j=1}^{n} |\hat{\mathbf{d}}_j - \sum_{i=1}^{m} x_i \mathbf{d}_{ij}|^2.$$

Because the muscle contractions should be lie in $[0, 1]$, we can find the contractions by solving the following optimization problem:

$$\begin{array}{ll} \textbf{minimize} & \sum_{j=1}^{n} |\hat{\mathbf{d}}_j - \sum_{i=1}^{m} x_i \mathbf{d}_{ij}|^2 \\ \textbf{subject to} & 0 \le x_i \le 1 \ (i = 1, 2, \ldots, m) \end{array} \ . \qquad (2)$$

The muscle contraction vector $\mathbf{x} = [x_1, x_2, \ldots, x_m]^T$ can be obtained by solving the constrained quadratic programming

$$\begin{array}{ll} \textbf{minimize} & \frac{1}{2}\mathbf{x}^T \mathbf{Q} \mathbf{x} - \mathbf{x}^T \mathbf{c} \\ \textbf{subject to} & 0 \le x_i \le 1 \ (i = 1, 2, \ldots, m) \end{array} \ , \qquad (3)$$

where

$$\mathbf{Q} = 2 \begin{pmatrix} \sum_{j=1}^{n} |\mathbf{d}_{1j}|^2 & \sum_{j=1}^{n} \mathbf{d}_{1j} \cdot \mathbf{d}_{2j} & \cdots & \sum_{j=1}^{n} \mathbf{d}_{1j} \cdot \mathbf{d}_{mj} \\ \sum_{j=1}^{n} \mathbf{d}_{2j} \cdot \mathbf{d}_{1j} & \sum_{j=1}^{n} |\mathbf{d}_{2j}|^2 & \cdots & \sum_{j=1}^{n} \mathbf{d}_{2j} \cdot \mathbf{d}_{mj} \\ \vdots & \vdots & \vdots & \vdots \\ \sum_{j=1}^{n} \mathbf{d}_{mj} \cdot \mathbf{d}_{1j} & \sum_{j=1}^{n} \mathbf{d}_{mj} \cdot \mathbf{d}_{2j} & \cdots & \sum_{j=1}^{n} |\mathbf{d}_{mj}|^2 \end{pmatrix},$$

$$\mathbf{c} = 2 \begin{pmatrix} \sum_{j=1}^{n} \hat{\mathbf{d}}_j \cdot \mathbf{d}_{1j} \\ \sum_{j=1}^{n} \hat{\mathbf{d}}_j \cdot \mathbf{d}_{2j} \\ \vdots \\ \sum_{j=1}^{n} \hat{\mathbf{d}}_j \cdot \mathbf{d}_{mj} \end{pmatrix} .$$

We solve this problem using the active set method, and apply Lagrange method for the sub-problems derived from the active sets [15]. To make the optimization procedure more robust, we divided the face into the upper and lower regions. The contractions of *Frontalis*, *Corrugator*, and *Orbicularis oculi* were calculated using only the markers in the upper region, and contractions of the other muscles were calculated using only the markers in the lower region (Figure 2). A muscle contraction value larger than one can be thought of as an exaggerated expression. So, we set only $x_i \ge 0$ as the constraints if we need to allow the exaggeration.

4.2. Improving the Actuation Basis

The actuation basis only used for expression synthesis can be entirely depend on the craftsmanship of the artist. However, the actuation basis for the subject being captured needs to have a certain level of accuracy to get reliable expression analysis results. It is not likely that hand-generated basis has such an accuracy. Therefore we develop an iterative algorithm that increases the accuracy of an actuation basis.

The algorithm takes the form of a fixed point iteration between **modeling** and **analysis**—the result of **modeling** is used for the **analysis**, and the result of **analysis** is in turn used for improving the actuation basis. For the iteration, we collect a performance data called *training data* in which the actor is asked to make all sorts of expressions. We let the actor fully contract each individual muscles. Even though ordinary people cannot make isolated muscle actuation, the facial expressions generated in the process of trying to use only a single muscle contain important information about the operation of the muscles, and helps to find more optimal basis elements. The training data also includes a significant amount of ordinary expressions that involve compound actuation of multiple muscles.

We first calculate muscle contractions at all frames of the training data by solving (3). Then the following equations should be satisfied in ideal cases for the marker point $\mathbf{p}_j$:

$$\begin{array}{c} x_1^{(1)} \mathbf{d}_{1j} + x_2^{(1)} \mathbf{d}_{2j} + x_3^{(1)} \mathbf{d}_{3j} + \cdots + x_m^{(1)} \mathbf{d}_{mj} = \hat{\mathbf{d}}_j^{(1)} \\ x_1^{(2)} \mathbf{d}_{1j} + x_2^{(2)} \mathbf{d}_{2j} + x_3^{(2)} \mathbf{d}_{3j} + \cdots + x_m^{(2)} \mathbf{d}_{mj} = \hat{\mathbf{d}}_j^{(2)} \\ \vdots \\ x_1^{(N)} \mathbf{d}_{1j} + x_2^{(N)} \mathbf{d}_{2j} + x_3^{(N)} \mathbf{d}_{3j} + \cdots + x_m^{(N)} \mathbf{d}_{mj} = \hat{\mathbf{d}}_j^{(N)}, \end{array}$$

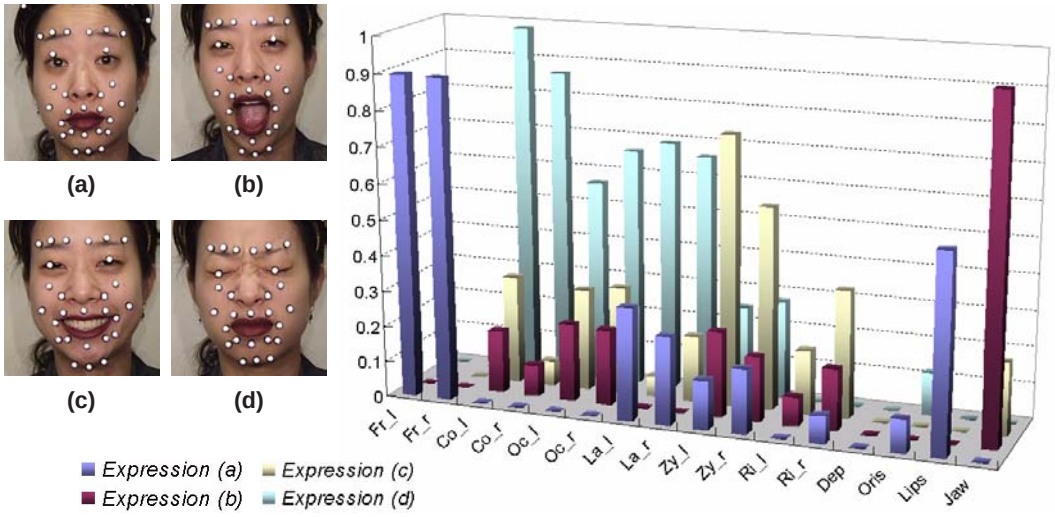


Figure 3. The four snapshots and plotting of the corresponding muscle contraction vectors

where $x_1^{(t)}, \ldots, x_m^{(t)}$ are analyzed muscle contractions at frame t, $\hat{\mathbf{d}}_j^{(t)}$ is the observed displacements at frame t, and N is the total number of frames in the training data. In reality, however, the equalities do not hold. But, if we solve the equations for $(\mathbf{d}_{1j}, \ldots, \mathbf{d}_{mj})$, the least square solution can provide us the improved position of the marker point $\mathbf{p}_j$ in each of the basis elements $\mathbf{E}_1, \ldots, \mathbf{E}_m$. If we perform the above steps for all the marker points $\mathbf{p}_j$ $(j = 1, \ldots, n)$, we can get a new (improved) actuation basis.

Thus we get an improved actuation basis from the initial draft: (1) calculating muscle contractions from the initial draft, (2) finding new $\mathbf{d}_{ij}$ $(i = 1, 2, \ldots, m, j = 1, 2, \ldots, n)$ with the muscle contractions. We repeat the cycle until the *total analysis error* $\sum_t \sum_j |\hat{\mathbf{d}}_j^{(t)} - \mathbf{d}_j^{(t)}|^2$ is saturated, and finally get the optimized displacements $\bar{\mathbf{d}}_{ij}$ $(i = 1, 2, \ldots, m, j = 1, 2, \ldots, n)$. Finally, using the scattered data interpolation method with a radial basis function [16], we can form an optimized actuation basis from $\bar{\mathbf{d}}_{ij}$ and the initial draft of actuation basis.

5. Experiments

We implemented our algorithms on a PC platform. For the experiment, we also developed an optical capture system with five video cameras [1], which generated 3D trajectory of the markers and gross motion of the head at 30 frames per second. The experimental results (demo clips) described in this Section is available at `http://graphics.snu.ac.kr/research/basis/`.

First, we 3D-scanned the face of an actor, let an artist model the actuation basis of it, ran the improvement algorithm described in Section 4.2, and got the final actuation basis. Then we captured performances and analyzed them into muscle actuation values. Figure 3 shows four expression snapshots during a performance: (a) raising eyebrows, (b) jaw rotation, (c) smiling, and (d) frowning. The graph in the figure plots the muscle contractions at the snapshots which were analyzed by the algorithm described in Section 4.1. The result agrees well with our anticipation:

- **Expression (a):** The contractions (about 0.9) of the left and right *Frontalis* are dominant in raising eyebrows.
- **Expression (b):** We can see the jaw rotation is conspicuous.
- **Expression (c):** The two dominant contractions in the middle correspond to the left and right *Zygomatic majors*, which matches well with the muscle actuation in real smiling.
- **Expression (d):** We can see the six muscles are dominant in the last row: the pairs of *Corrugator*, *Orbicularis oculi*, and *Levator labii superioris*. The contraction of *Corrugator* and *Levator labii superioris* matches well with the muscle actuation in real frowning. *Orbicularis oculi* resulted from the close of the eyes are not directly related to this expression.

Figure 4 shows the result of applying the contractions of the expressions to the computer model of the actor and other two cartoon-like characters.

Figure 5 plots the contractions of left *Frontalis* and *Jaw rotation* during the performance of "Tulip Season" contained in the demo clip. The x-axis represents the frame

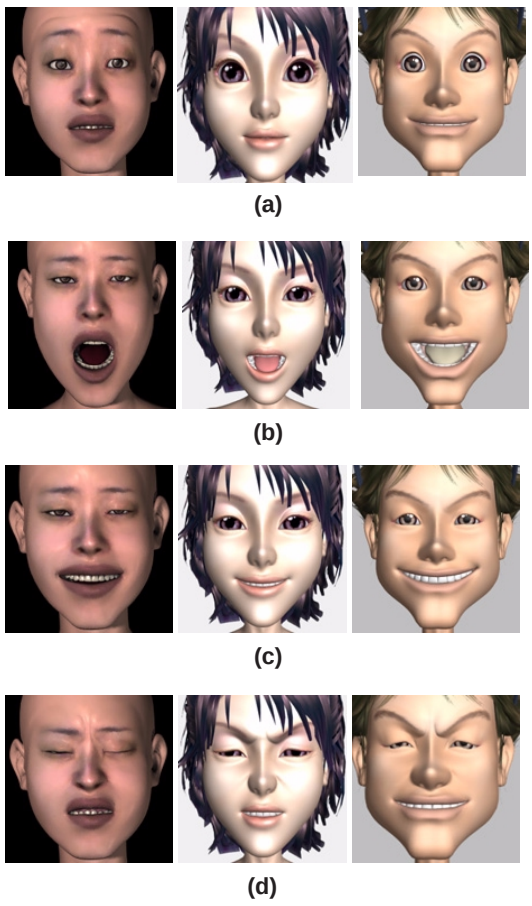


Figure 4. Results of applying the muscle contractions in Figure 3 to different 3D models.

number and the y-axis represents the muscle contractions in $[0, 1]$. The figure also shows *per-frame marker error* $(\sum_{j=1}^{n} |\hat{\mathbf{d}}_j - \mathbf{d}_j|)/n$, which is measured in centimeters. The error was computed separately for the upper and lower regions of the face. The error is bigger in the lower region due to the nonlinear and irregular movement around the mouth, which is mainly caused by *Orbicularis oris* muscle.

6. Conclusion

In this paper, we presented a new muscle-based facial animation technique that uses the actuation basis, a set of 3D facial shapes corresponding to the full actuation of individual muscles. Instead of completely relying on a mathematical method, we let artists manually sculpt (the initial draft of) the basis elements so that we could get more predictable deformation of the face. To increase the accuracy of the actuation basis, we developed an iterative algorithm that refined the actuation basis. Once an actuation basis was ready, a performance could be analyzed quite accurately, and the result could be applied to any 3D models with equivalent muscle structures.

An interesting contribution of this paper is that it proposed a technique that includes the artists' modeling capability as an integral part of the algorithm. The manual shaping of the basis elements complemented the pure mathematical approach which produced unexpected results occasionally. The proposed method is robust, and we believe that this artist-in-the-loop method can be quite useful in animation productions until the mathematical models can accurately simulate the operation of the muscles and concomitant movements in the facial tissue and skin.

Acknowledgment

This work was supported by the Brain Korea 21 Project. This work was also supported by ASRI (The Automation and Systems Research Institute) at Seoul National University. We thank Eunjeong Kang very much for modeling all the computer models and basis elements of them used in the experiments.

References

[1] B. Choe, H. Lee, and H.-S. Ko. Performance-driven muscle-based facial animation. *The Journal of Visualization and Computer Animation*, 12(2):67–79, May 2001.

[2] C. D. Clemente. *Anatomy: A Regional Atlas of the Human Body, 2nd edition*. Urban and Schwarzenberg, 1981.

[3] P. Eisert and B. Girod. Analyzing facial expression for virtual conferencing. *IEEE Computer Graphics & Applications*, 18(5):70–78, September - October 1998. ISSN 0272-1716.

[4] P. Ekman and W. V. Friesen. *Facial Action Coding System*. Consulting Psychologists Press, Inc., 1978.

[5] 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 '96 Conference*, June 1996. Geneva, Switzerland.

[6] I. A. Essa and A. P. Pentland. Coding, analysis, interpretation and recognition of facial expressions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):757–763, July 1997.

[7] G. Faigin. *The Artist's Complete Guide to Facial Expression*. Watson-Guptill Publications, 1990.

[8] M. K. Greenwald, E. W. C. III, and P. J. Lang. Affective judgment and psychophysiological response: dimensional covariation in the evaluation of pictorial stimuli. *Journal of Pyschophysiology*, 3:51–64, 1989.

[9] B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin. Making faces. In *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 55–66. ACM SIGGRAPH, Addison Wesley, July 1998.

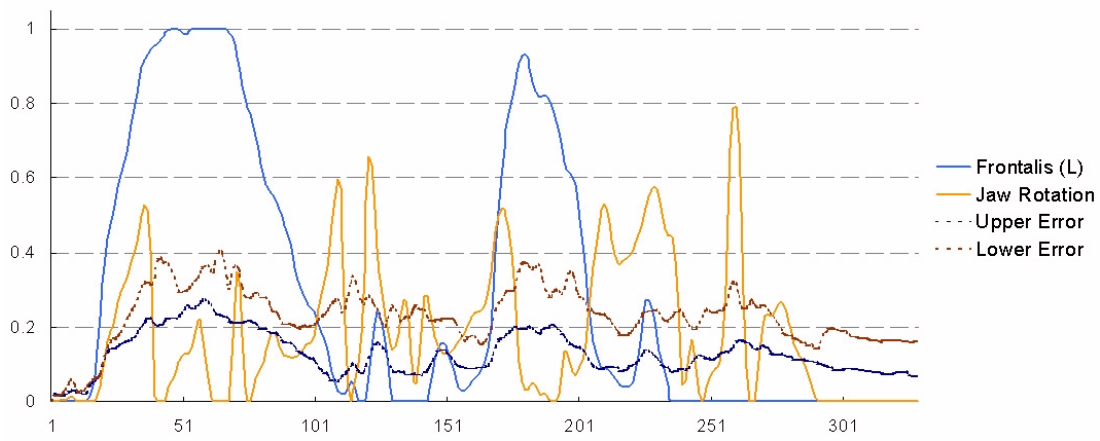


Figure 5. Contractions of 'left frontalis' / 'jaw rotate' and marker errors.

[10] R. M. Koch, M. H. Gross, and A. Bosshard. Emotion editing using finite elements. *Computer Graphics Forum*, 17(3):295–302, 1998. ISSN 1067-7055.

[11] R. M. Koch, M. H. Gross, F. R. Carls, D. F. von Büren, G. Fankhauser, and Y. Parish. Simulating facial surgery using finite element methods. In *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 421–428. ACM SIGGRAPH, Addison Wesley, Aug. 1996.

[12] C. Kouadio, P. Poulin, and P. Lachapelle. Real-time facial animation based upon a bank of 3D facial expressions. In *Proceedings of Computer Animation '98 Conference*. IEEE Computer Society Press, 1998.

[13] Y. Lee, D. Terzopoulos, and K. Waters. Realistic face modeling for animation. In *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 55–62. ACM SIGGRAPH, Addison Wesley, Aug. 1995.

[14] J. P. Lewis, M. Cordner, and N. Fong. Pose space deformations: A unified approach to shape interpolation a nd skeleton-driven deformation. *Proceedings of SIGGRAPH 2000*, pages 165–172, July 2000. ISBN 1-58113-208-5.

[15] D. G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley, 2nd edition, 1984.

[16] G. M. Nielson. Scattered data modeling. *IEEE Computer Graphics and Applications*, 13(1):60–70, Jan. 1993.

[17] F. I. Parke and K. Waters. *Computer Facial Animation*. A K Peters, 1996. ISBN 1-56881-014-8.

[18] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D. H. Salesin. Synthesizing realistic facial expressions from photographs. In *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 75–84. ACM SIGGRAPH, Addison Wesley, July 1998.

[19] F. Pighin, R. Szeliski, and D. H. Salesin. Resynthesizing facial animation through 3D model-based tracking. In *Seventh International Conference on Computer Vision (ICCV '99) Conference Proceedings*, pages 143–150, September 1999. Corfu, Greece.

[20] D. Terzopoulos and K. Waters. Physically-based facial modelling, analysis, and animation. *The Journal of Visualization and Computer Animation*, 1:73–80, 1990.

[21] D. Terzopoulos and K. Waters. Analysis and synthesis of facial image sequences using physical and anatomical models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):569–579, June 1993.

[22] K. Waters. A muscle model for animating three-dimensional facial expression. *Computer Graphics (Proceedings of SIGGRAPH 87)*, 21(4):17–24, July 1987. Held in Anaheim, California.

[23] L. Williams. Performance-driven facial animation. *Computer Graphics (Proceedings of SIGGRAPH 90)*, 24(4):235–242, August 1990. ISBN 0-201-50933-4. Held in Dallas, Texas.

[24] J. yong Noh and U. Neumann. Expression cloning. *Proceedings of SIGGRAPH 2001*, pages 277–288, August 2001. ISBN 1-58113-292-1.

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.

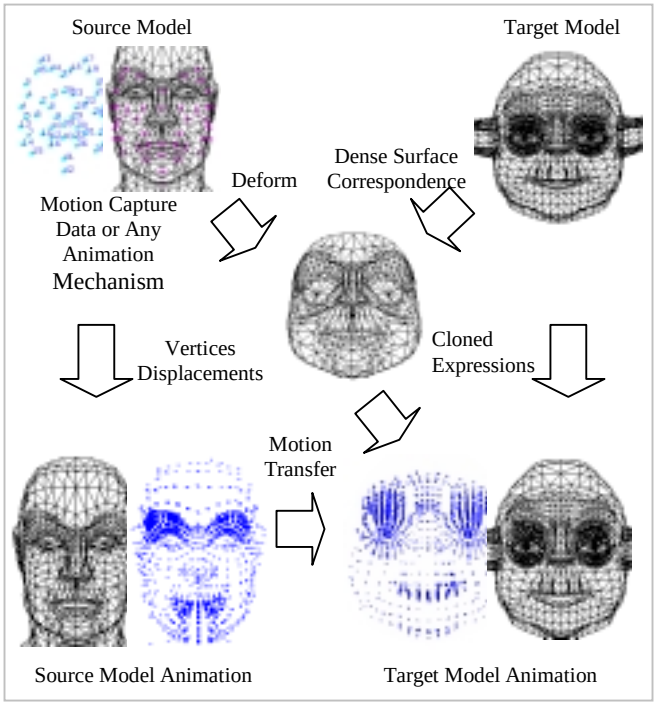


Figure 1 The expression cloning system

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.

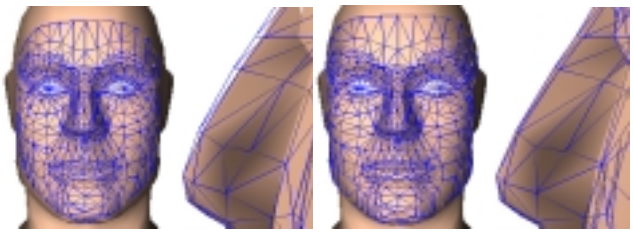
2 Expression Cloning

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.

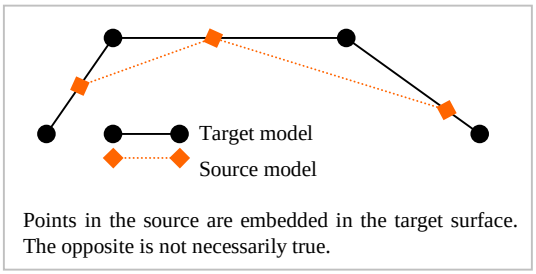


Figure 3 Side view of the two models after the projection

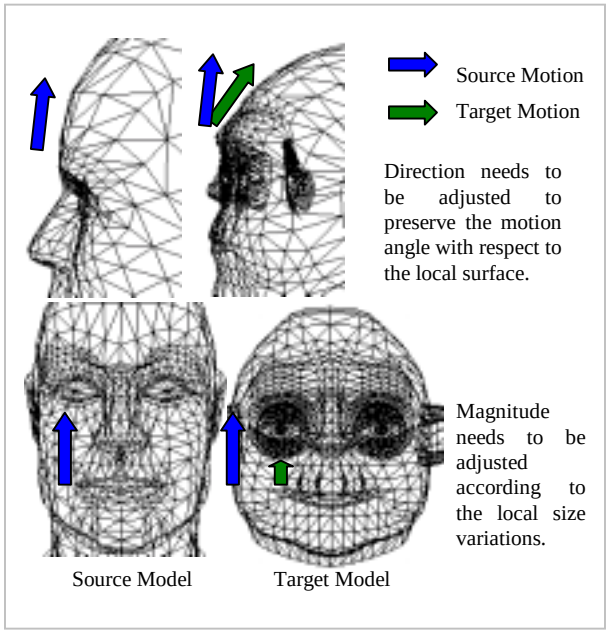


Figure 4 The direction and the magnitude adjustment of motion vectors

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

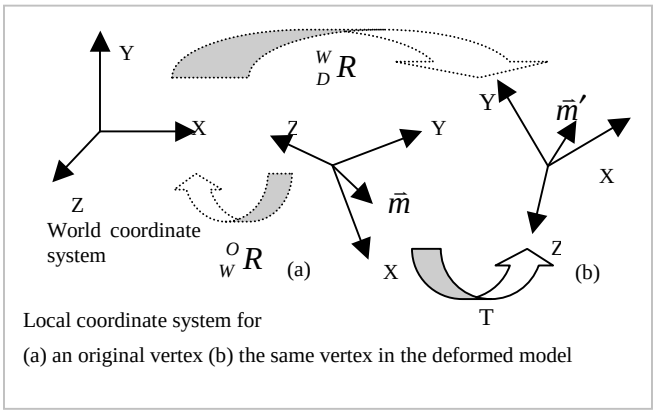


Figure 5 Transformation matrix as a means to adjust a motion vector direction

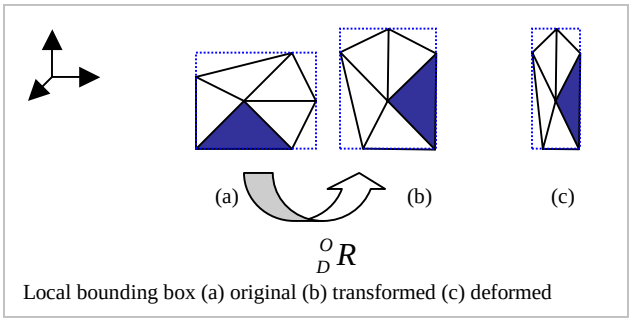


Figure 6 Local bounding box produces scaling factors.

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

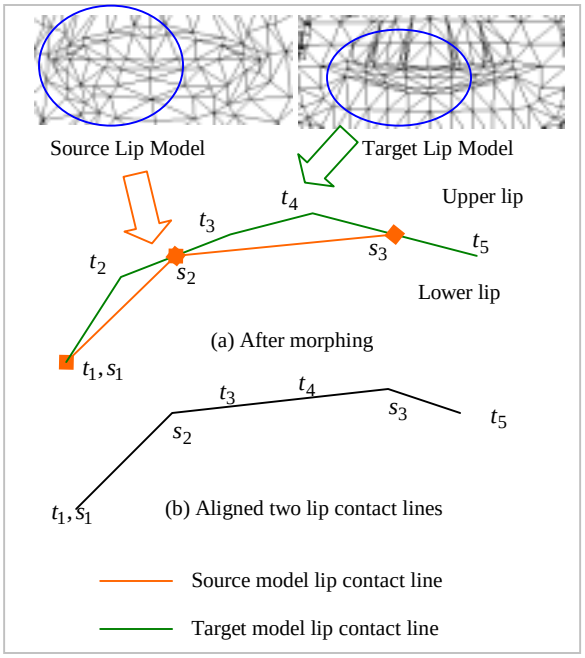


Figure 7 Lip contact line alignment

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).

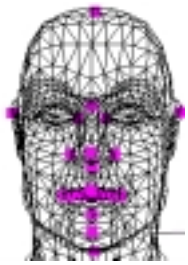
Model	Polygons	Vertices
Source Man	1954	988
Woman	5416	2859
Man	4314	2227
Rick	927	476
Yoda	3740	1945
Cat	5405	2801
Monkey	2334	1227
Dog	927	476
Baby	1253	2300

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.

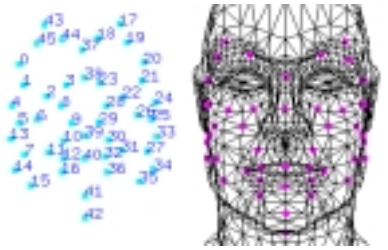
$$\%Error = 100\frac{size(PositionError)}{size(MotionVector)} \tag{20}$$

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

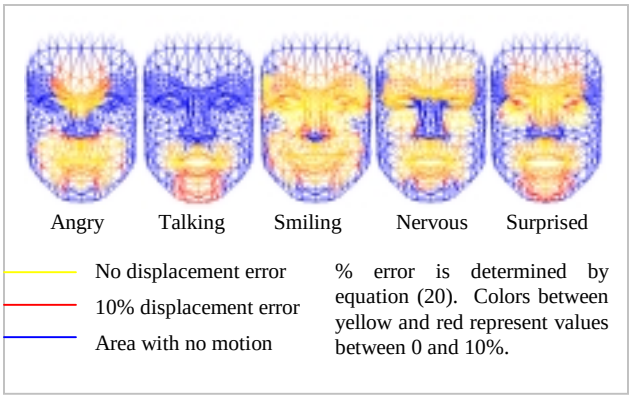


Figure 10 Visually depicted displacement errors

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

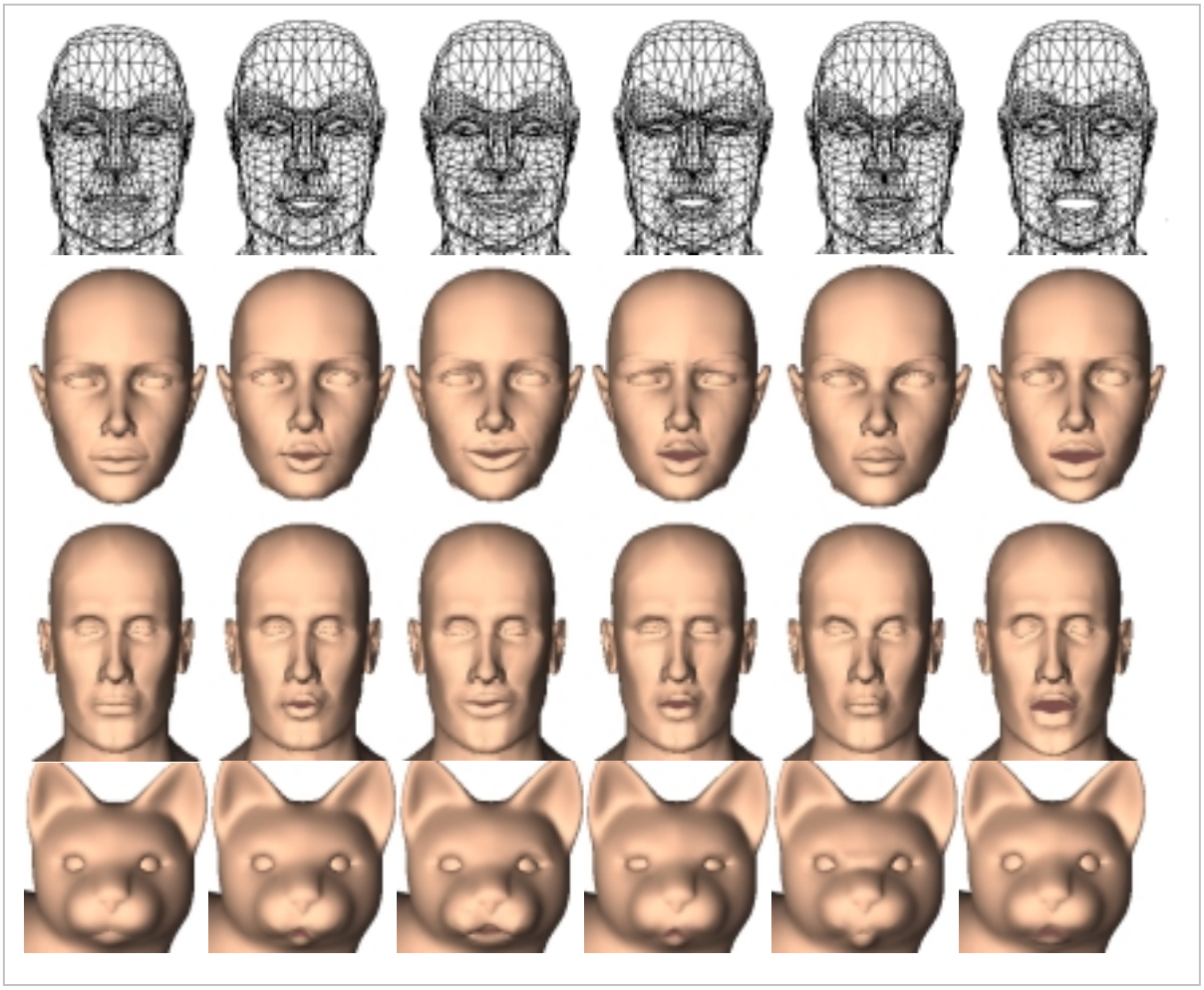


Figure 11 Cloned expressions on various models

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.

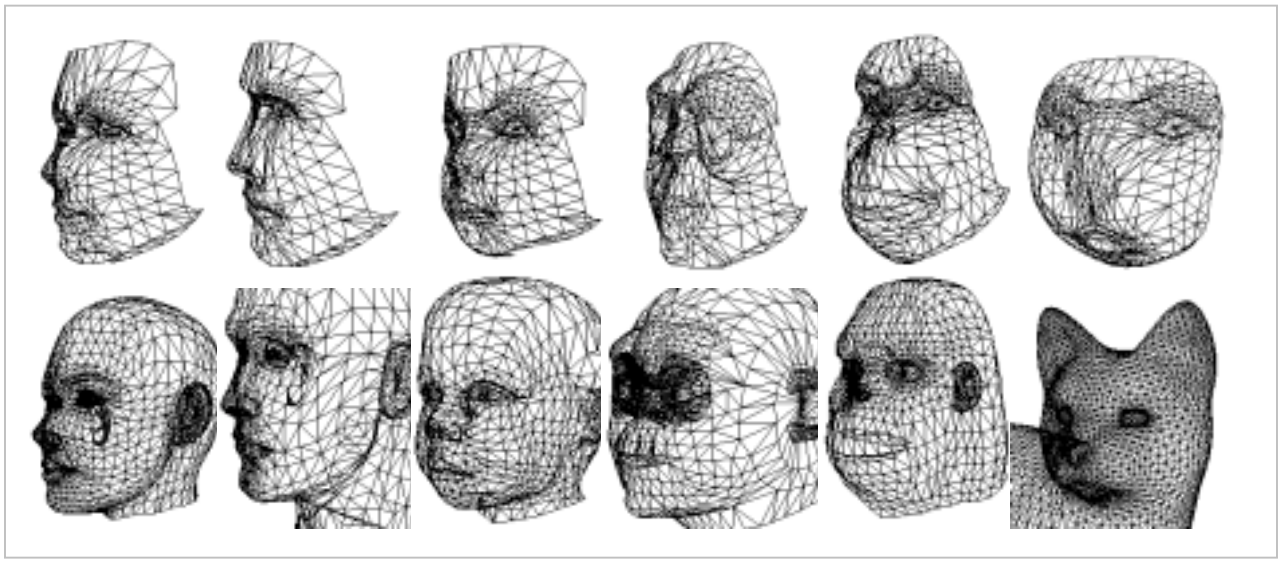


Figure 12 Deformed models produce dense surface correspondences.

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.

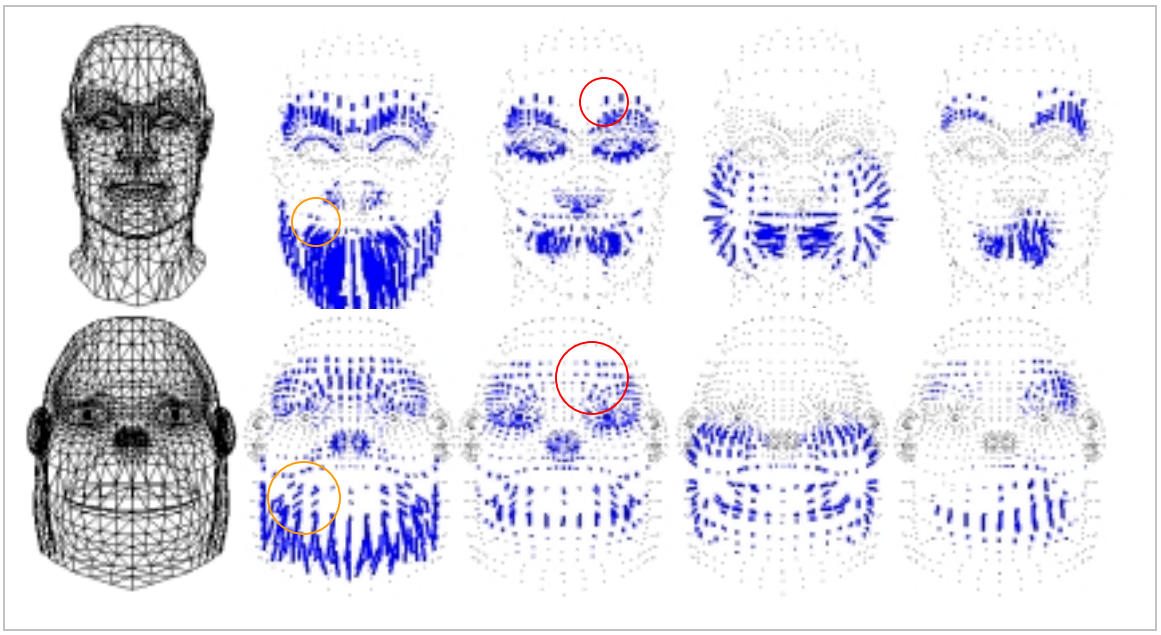


Figure 13 The direction and magnitude adjustments for the motion vector transfer

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).

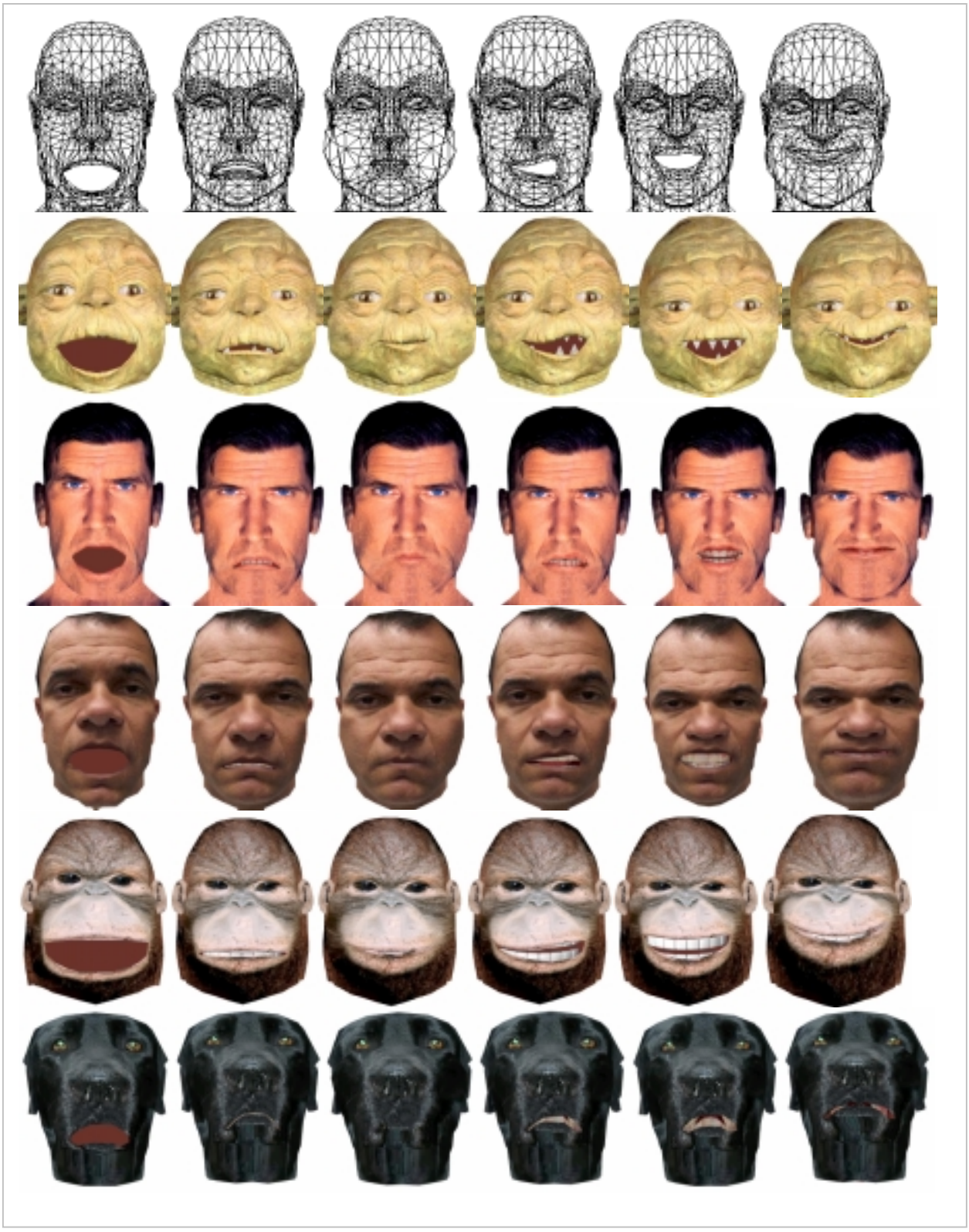


Figure 14 Exaggerated expressions cloned on a wide variety of texture-mapped target models

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

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

An Example-Based Approach for Facial Expression Cloning

Hyewon Pyun[1], Yejin Kim[2], Wonseok Chae[1], Hyung Woo Kang[1], and Sung Yong Shin[1]

[1]Korea Advanced Institute of Science and Technology
[2]Electronics and Telecommunications Research Institute
e-mail: [1]{hyewon, wschae, henry, syshin}@jupiter.kaist.ac.kr, [2]yejink@etri.re.kr

Abstract

In this paper, we present a novel example-based approach for cloning facial expressions of a source model to a target model while reflecting the characteristic features of the target model in the resulting animation. Our approach comprises three major parts: key-model construction, parameterization, and expression blending. We first present an effective scheme for constructing key-models. Given a set of source example key-models and their corresponding target key-models created by animators, we parameterize the target key-models using the source key-models and predefine the weight functions for the parameterized target key-models based on radial basis functions. In runtime, given an input model with some facial expression, we compute the parameter vector of the corresponding output model, to evaluate the weight values for the target key-models and obtain the output model by blending the target key-models with those weights. The resulting animation preserves the facial expressions of the input model as well as the characteristic features of the target model specified by animators. Our method is not only simple and accurate but also fast enough for various real-time applications such as video games or internet broadcasting.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Three-Dimensional Graphics and Realism]: Animation; I.3.5 [Computational Geometry and Object Modeling]: Geometric Algorithms

Keywords: Facial Animation, Facial Expression Cloning, Example-based Synthesis, Scattered Data Interpolation, Motion Retargetting

1. Introduction

1.1. Motivation

Synthesis by reusing existing data has recently been popular in a wide spectrum of computer graphics, including shape modelling [29, 1], image or texture synthesis [16, 32], and motion generation [8, 19]. Inspired by motion retargetting [8, 14, 26], Noh and Neumann [18] posed the problem of cloning facial expressions of an existing 3D face model to a new model. Based on 3D geometry morphing, their solution to this problem is first to compute the motion vectors of the source model and then deform these to add to the target model. This approach works well for face models with similar shapes. In general, it is hard to simulate the imagination (or intention) of an animator by this rather mechanical manipulation of motion vectors.

There is a stream of research on parameter-driven facial animation such as facial action coding system or model-based persona transmission [20]. In particular, 3D facial animation is generated from 2D videos in performance-driven facial animation, which can be thought of as a process of transferring parameters from the source space (2D videos) to the target space (3D face models) [12, 24, 6]. In general, the target animation is obtained by deforming the target model or blending 3D face models with different expressions, to match the parameters transferred from the source space. The parameter-driven facial animation usually experiences time-consuming optimization in deformation or blending.

Based on the notion of parameter transfer, Bregler et al. [2] proposed an elegant scheme for cartoon motion capture and retargetting. They chose source example key-shapes from a given input cartoon animation and modelled their corresponding target key-shapes. Given the shape of an input cartoon character, they interpret it as a shape interpolated from

source key-shapes, which is deformed by an affine transformation. To capture a snapshot (posture) of cartoon motion, they extracted the affine transformation parameters together with the interpolation weight values of source example key-shapes at every time step by least squares approximation. They then applied the extracted parameters and weights to the target example key-shapes for motion retargetting. By allowing animators to model the target key-shapes explicitly, their imagination can be realized in the resulting cartoon animation.

Based on numerical optimization, their scheme is a little too time-consuming to be applied directly to the cloning problem, in particular, for real-time applications such as computer games and internet broadcasting. Furthermore, facial expressions result from local deformations of various parts of a face model, which may not be reflected properly by an affine transformation. To address these issues, we propose a novel scheme for cloning facial expressions of a source model while preserving the characteristic features of target expressions specified by animators. Based on scattered data interpolation, the proposed scheme is not only simple and efficient but also reflects animators' intention accurately.

1.2. Related Work

There have been extensive efforts on the development of 3D facial animation techniques since Parke's pioneering work [21]. An excellent survey of these efforts can be found in [20]. We begin with traditional approaches for generating facial animation from scratch and then move on to more recent work directly related to our scheme.

Facial Animation From Scratch: Physically-based approaches have been used to generate facial expressions by simulating the physical properties of facial skin and muscles [15, 25, 30, 31]. Parke proposed a parametric approach to represent the motion of a group of vertices with a parameter vector and used this approach to generate a wide range of facial expressions [22]. In performance-driven approaches, facial animations were synthesized based on facial motion data captured from live actors' performances [33, 9, 6]. Kalra et al. [11] used free-form deformations to manipulate facial expressions. Pighin et al. [23] presented an image-based approach to generate photorealistic 3D facial expressions from a set of 2D photographs. All of those approaches are common in that the same process needs to be repeated for animating a new face model, even when a similar expression sequence has already been available for a different model.

Retargetting and Cloning: Recently, there have been rich research results on reusing existing animation data. Gleicher [8] described a method for retargetting motions to new characters with different segment proportions. Lee and Shin [14] enhanced this idea by using a hierarchical displacement mapping technique based on multilevel B-spline approximation. Popovic and Witkin [26] presented a physically-based motion transformation technique that preserves the essential physical properties of a motion by using the spacetime constraints formulation. The approaches for motion retargetting focused on skeleton-based articulated body motions. Noh and Neumann [18] adopted the underlying idea of motion retargetting for reusing facial animation data. Based on 3D geometry morphing between the source and target face models, their approach transfers the facial motion vectors from a source model to a target model in order to generate cloned expressions on the target model. This method is suitable for mutually morphable face models with a strong shape resemblance to reproduce facial expressions as intended by animators.

Example-based Motion Synthesis: Example-based motion synthesis is another stream of research directly related to our approach. Rose et al. [27] and Sloan et al. [29] proposed example-based motion blending frameworks, employing scattered data interpolation with radial basis functions. Park et al. [19] proposed an on-line motion blending scheme for locomotion generation adopting this idea. Lewis et al. [17] introduced an example-based pose space deformation technique, and Allen et al. [1] applied a similar technique to range-scan data for creating a new pose model. While most of the previous techniques focused on the pure synthesis aspect, Bregler et al. [2] proposed an example-based approach for cartoon motion capture and retargetting. Based on affine transformations, their approach first extracts the transformation parameters and the interpolation weights of the source key-shapes at each frame of the input cartoon animation and then generates an output shape by applying both the parameters and weights to the corresponding target key-shapes. This approach is non-trivially adapted for cloning facial expressions in our work. We also note that there have been some similar example-based approaches in the domain of peformance-driven facial animation, for retargetting facial expressions from 2D videos to 2D drawings [3] or to 3D models [4].

1.3. Overview

Inspired by cartoon motion capture and retargetting [2], we adopt an example-based approach for retargetting facial expressions from one model to another. Figure 1 illustrates our example-based approach for expression cloning. Given an input 3D facial animation for a source face model, we generate a similar animation for a target model by blending the predefined target models corresponding to the example source face models with extreme expressions called the *key-models*. Our approach comprises three major parts: key-model construction, parameterization, and expression blending. The first two parts are done once at the beginning, and the last part is repeatedly executed for each input expression in runtime.

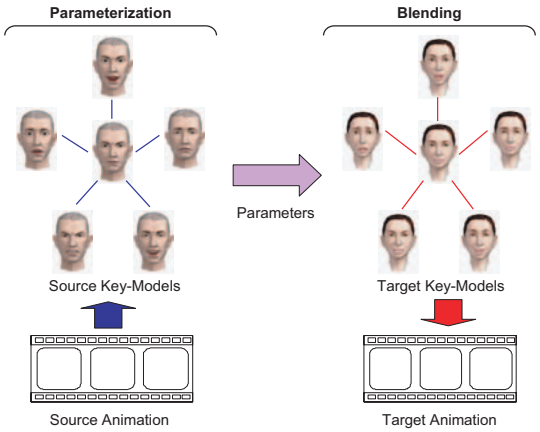


Figure 1: *Overview of our example-based cloning*

First, we identify a set of example extreme expressions for a given source model. These expressions should be generic enough to handle the facial animations of the source model effectively. Provided with the example extreme expressions, animators construct the corresponding key-models for both source and target face models. The source key-models should reflect the actual extreme expressions of the source model accurately. However, the animators can breathe their creativity and imagination into the target key-models while constructing them. To minimize the time and efforts of animators, we provide a novel scheme for compositing key-models. Next, the target key-models are parameterized to apply scattered data interpolation [27, 29]. We provide a simple, elegant parameterization scheme for effective motion blending. Finally, given an input model with some facial expression, the parameter vector for the corresponding output model is analytically computed to evaluate the predefined weight functions of target key-models. The output model with the cloned expression is obtained by blending the target key-models with respect to those weight values.

The remainder of this paper is organized as follows: In Sections 2, 3, and 4, we describe key-model construction, parameterization, and expression blending, respectively. We show experimental results and compare our approach with the previous one [18] in both quality and efficiency in Section 5. Finally, in Section 6 we conclude this paper and discuss future research issues.

2. Key-Model Construction

The facial expression cloning problem has a quite different nature than the cartoon motion capture and retargetting problem. In the latter problem, cartoon characters have the capability of changing their shapes dynamically while still preserving their identities. Bregler et al. tried to capture those dynamic shape changes with 2D affine transformations together with key-shape interpolation. In cartoon animations, it is hard to identify all generic extreme key-shapes due to their dynamic nature. Thus, the authors selected extreme key-shapes from a given source animation.

On the other hand, facial expressions are determined by a combination of subtle local deformations on a source face model rather than global shape change. Unlike cartoon motions, face motions (expressions) have been well characterized. In particular, we adopt two categories of key-expressions: emotional key-expressions and verbal key-expressions. The former category of key-expressions reflect emotional states, and the latter expressions mainly result from lip movements for verbal communications. We combine them to define generic key-expressions for a source model, and then create the corresponding key-models for both source and target face models, by deforming their respective base models with neutral expressions. Through crafting the target key-models, animators can realize their imaginations.

Referring to the emotion space diagram [28], we choose six purely emotional key-expressions including neutral, happy, sad, surprised, afraid, and angry expressions as shown in Figure 2. The neutral expression is chosen as the base expression. Based on the notion of *visemes*†, that is, the distinct visual mouth expressions observed in speech [7], we take thirteen visemes as the purely verbal key-expressions as depicted in Figure 3. Combining these two categories of key-expressions, we have 78 (6×13) generic key-expressions together with six purely emotional expressions. Notice that the purely verbal expressions are regarded as the verbal expressions combined with the neutral emotional expression.

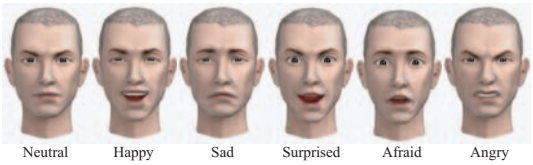


Figure 2: *Six emotional key-expressions.*

To facilitate example-based expression cloning, we need to have the corresponding 84 key-models for each of the source and target face models. It is time-consuming to craft all of them by hand. Instead, we take a semi-automatic approach: preparing the purely emotional and purely verbal key-models by hand and then combining them automatically. Now, our problem of automatic key-model creation is reduced to a geometry compositing problem: Given an emotional key-expression and a verbal key-expression, how can we obtain their combined key-expressions?

† The term 'viseme' is the abbreviation of 'visual phoneme'.

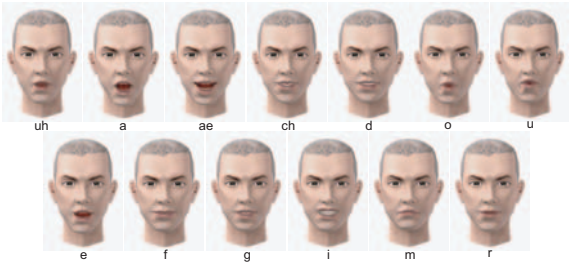


Figure 3: *Thirteen verbal key-expressions.*

Without loss of generality, suppose that the face models are represented as polygonal meshes (polyhedra). Then, expressions cause the movements of vertices on a face model. Analysing the purely emotional and verbal key-models with respect to the base model, we characterize the vertices in terms of their contributions to facial expressions. For example, vertices near eyes contribute mainly to making emotional expressions. Vertices near the mouth contribute to both emotional and verbal expressions. However, the movements are mainly constrained by verbal expressions to produce accurate pronunciations when there are any conflicts between two types of expressions.

Figure 4: *A distribution of importance values.*

Based on this observation, we introduce the notion of importance, which measures the relative contribution of each vertex to the verbal expressions with respect to the emotional expressions. The importance value α_i of every vertex v_i is estimated empirically from the purely emotional and verbal key-models such that $0 \le \alpha_i \le 1$ for all i. If $\alpha_i \ge 0.5$, then the movement of v_i is constrained by the verbal expressions; otherwise, it is constrained by the emotional expressions. Figure 4 shows the distribution of importance values over a face model estimated by our scheme as given in the Appendix, to which we refer readers for details. Brighter regions contain vertices of higher importance values.

Now, we are ready to explain how to composite an emotional key-model E and a verbal key-model P derived from the base model B. Let

$$\begin{aligned} B &= \{v_1,\ v_2,\ \ldots,\ v_n\}, \\ E &= \{v_1^E,\ v_2^E,\ \ldots,\ v_n^E\},\ \text{and} \\ P &= \{v_1^P,\ v_2^P,\ \ldots,\ v_n^P\}. \end{aligned}$$

v_i^E and $v_i^P, 1 \le i \le n$ are obtained by displacing v_i, if needed, and thus the natural correspondence is established for the vertices with the same subscript. For every vertex v_i, we define displacements Δv_i^E and Δv_i^P as follows:

$$\begin{aligned} \Delta v_i^E &= v_i^E - v_i,\ \text{and} \\ \Delta v_i^P &= v_i^P - v_i. \end{aligned}$$

Let $C = \{v_1^C,\ v_2^C,\ \ldots,\ v_n^C\}$ and $\Delta v_i^C = v_i^C - v_i$ be the combined key-model and the displacement of a vertex $v_i^C \in C$, respectively. Since the problem of computing v_i^C can be reduced to the compositing of two vectors Δv_i^E and Δv_i^P, we assume that v_i^C lies on the plane spanned by Δv_i^E and Δv_i^P and containing v_i as shown in Figure 5.

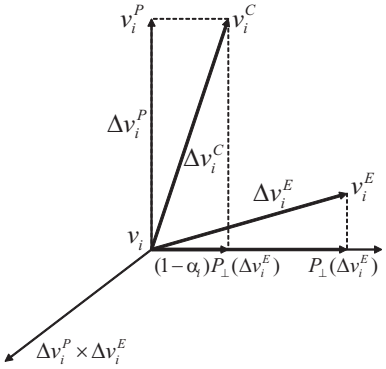


Figure 5: *Composition of the two displacements*

Consider a vertex $v_i^C, 1 \le i \le n$ of the combined key-model in C. If $\alpha_i \ge 0.5$, then the verbal component Δv_i^P should be preserved in Δv_i^C for accurate pronunciation. Therefore, letting $P_\perp(\Delta v_i^E)$ be the component of Δv_i^E perpendicular to Δv_i^P, only this component $P_\perp(\Delta v_i^E)$ of Δv_i^E can make a contribution to Δv_i^C on top of Δv_i^P. That is, we ignore the other component $P_\parallel(\Delta v_i^E)$ of Δv_i^E which is parallel to Δv_i^P. Otherwise, the constraints on accurate pronunciation would not be satisfied when there are conflicts between the two types of expressions. (see Figure 5). If $\alpha_i < 0.5$, the roles of Δv_i^P and Δv_i^E are switched. Thus, we have

$$v_i^C = \begin{cases} v_i + (\Delta v_i^P + (1-\alpha_i)P_\perp(\Delta v_i^E)) & \text{if } \alpha_i \ge 0.5 \\ v_i + (\Delta v_i^E + \alpha_i E_\perp(\Delta v_i^P)) & \text{otherwise,} \end{cases}$$

where

$$\begin{aligned} P_\perp(\Delta v_i^E) &= \Delta v_i^E - \frac{\Delta v_i^E \cdot \Delta v_i^P}{|\Delta v_i^P|^2} \cdot \Delta v_i^P,\ \text{and} \\ E_\perp(\Delta v_i^P) &= \Delta v_i^P - \frac{\Delta v_i^P \cdot \Delta v_i^E}{|\Delta v_i^E|^2} \cdot \Delta v_i^E. \end{aligned}$$

Figure 6 shows a combined key-model (Figure 6(c)) constructed from a key-model with a verbal expression (Figure 6(a)) and a key-model with an emotional expression

(Figure 6(b)). Note that the verbal expression is preserved around the mouth and the emotional expression is preserved in other parts.

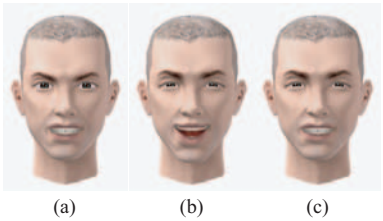


Figure 6: *Compositing key-models: (a) verbal key-model (vowel 'i') (b) emotional key-model ('happy') (c) combined key-model.*

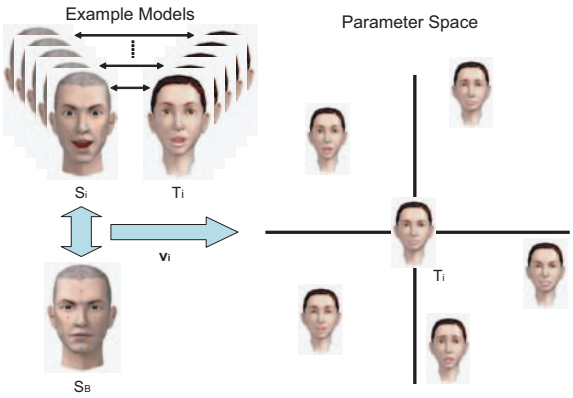


Figure 8: *The displacement vector of each source key-model* $\mathbf{S}_i$ *is used for parameterizing the corresponding target key-model* $\mathbf{T}_i$.

3. Parameterization

We parameterize the target key-models based on the correspondences between the source base model and the source key-models. We interactively select a number of feature points on the source base model and then extract their displacements to the corresponding points on each of the source key-models. Concatenating these displacements, the displacement vector of each source key-model is formed to parameterize the corresponding target key-model. Most individual parameter components tend to be correlated to each other. Thus, based on PCA (principal component analysis) [10], the dimensionality of the parameter space can be reduced by removing less significant basis vectors of the resulting eigenspace.

As illustrated in Figure 7, we select about 20 feature points from the source base model. While the number of feature points depends on the shape and complexity of the base model, we believe empirically that two to four feature points around the facial parts such as the mouth, eyes, eyebrows, the forehead, the chin, and cheeks are sufficient to represent distinct facial expressions. Note that we only mark the feature points on the base model. Then, those on the other key-models are automatically determined from their vertex correspondences to the base model.

Figure 7: *The source base key-model with 20 manually selected feature points.*

The displacement vector $\mathbf{v}_i$ of a source key-model $\mathbf{S}_i$ from the source base key-model $\mathbf{S}_B$ is defined as follows:

$$\mathbf{v}_i = \mathbf{s}_i - \mathbf{s}_B,\ 1 \le i \le M, \tag{1}$$

where $\mathbf{s}_B$ and $\mathbf{s}_i$ are vectors obtained by concatenating, in a fixed order, the 3D coordinates of feature points on $\mathbf{S}_B$ and those on $\mathbf{S}_i$, respectively, and M is the number of source key-models. As shown in Figure 8, $\mathbf{v}_i$ places each target key-model $\mathbf{T}_i$ in the N-dimensional parameter space, where N is the number of components, that is, three times the number of feature points.

Since the dimensionality N of the parameter space is rather high compared to the number M of key-models, we employ PCA to reduce it. Given M displacement vectors of dimension N, we first generate their component covariance matrix, which is an $N \times N$ square matrix, to compute the eigenvectors of the matrix and the corresponding eigenvalues. These eigenvectors are called the *principal components* representing the principal axes that characterize the distribution of displacement vectors. The dimensionality of the parameter space can be reduced by removing less significant eigenvectors, which have small eigenvalues. In our experiments, we use an empirical threshold value of 0.00001 to remove those eigenvectors. The removal of such eigenvectors may cause some characteristics of the key-models not to be parameterized. With our choice of the threshold, we have observed that the effect is negligible. In experiments, the dimensionality of the parameter space can be reduced from 60 to 18 without any difficulty.

Let $\mathbf{e}_i, 1 \le i \le N$ be the eigenvector corresponding to the ith largest eigenvalue. Suppose that we choose $\bar{N}$ eigenvectors as the coordinate axes of the parameter space, where $\bar{N} < N$. To transform an original N-dimensional displacement vector into an $\bar{N}$-dimensional parameter vector, an

$\bar{N} \times N$ matrix $\mathbf{F}$ called the *feature matrix* is constructed:

$$\mathbf{F} = [\mathbf{e}_1\ \mathbf{e}_2\ \mathbf{e}_3 \ldots \mathbf{e}_{\bar{N}}]^\top, \tag{2}$$

Using the feature matrix $\mathbf{F}$, the parameter vector $\mathbf{p}_i$ corresponding to the displacement vector v_i of a target key-model $\mathbf{T}_i$ is derived as follows:

$$\mathbf{p}_i = \mathbf{F}\mathbf{v}_i,\ 1 \le i \le M, \tag{3}$$

which reduces the dimensionality of the parameter space from N to $\bar{N}$. This is equivalent to projecting each displacement vector $\mathbf{v}_i$ onto the eigenspace spanned by the $\bar{N}$ selected eigenvectors. We later use this feature matrix $\mathbf{F}$ to compute the parameter vector from a given displacement vector.

4. Expression Blending

With the target key-models thus parameterized, our cloning problem is transformed to a scattered data interpolation problem. To solve this problem, our expression blending scheme predefines the weight functions for each target key-model based on cardinal basis functions [29], which consist of linear and radial basis functions. The global shape of a weight function is first approximated by linear basis functions, and then adjusted locally by radial basis functions to exactly interpolate the corresponding key-model. Given the input face model with a facial expression, a novel output model with the cloned expression is obtained in runtime by blending the target key-models as illustrated in Figure 9. Our scheme first computes the displacement vector of the input face model and then derives the parameter vector of the output model from the displacement vector. Finally, the predefined weight functions are evaluated at this parameter vector to produce the weight values, and the output model with the cloned facial expression is generated by blending the target key-models with respect to those weight values.

The weight function $w_i(\cdot)$ of each target example model $\mathbf{T}_i, 1 \le i \le M$ at a parameter vector $\mathbf{p}$ is defined as follows:

$$w_i(\mathbf{p}) = \sum_{l=0}^{\bar{N}} a_{il} A_l(\mathbf{p}) + \sum_{j=1}^{M} r_{ji} R_j(\mathbf{p}). \tag{4}$$

where $A_l(\mathbf{p})$ and a_{il} are the linear basis functions and their linear coefficients, respectively. $R_j(\mathbf{p})$ and r_{ij} are the radial basis functions and their radial coefficients. Let $\mathbf{p}_i, 1 \le i \le M$ be the parameter vector of a target key-model $\mathbf{T}_i$. To interpolate the target key-models exactly, the weight of a target key-model $\mathbf{T}_i$ should be one at $\mathbf{p}_i$ and zero at $\mathbf{p}_j, i \ne j$, that is, $w_i(\mathbf{p}_i) = 1$ for $i = j$ and $w_i(\mathbf{p}_j) = 0$ for $i \ne j$.

Ignoring the second term of Equation (4), we solve for the linear coefficients a_{il} to fix the first term:

$$w_i(\mathbf{p}) = \sum_{l=0}^{\bar{N}} a_{il} A_l(\mathbf{p}). \tag{5}$$

The linear bases are simply $A_l(\mathbf{p}) = \mathbf{p}^l, 1 \le l \le \bar{N}$, where

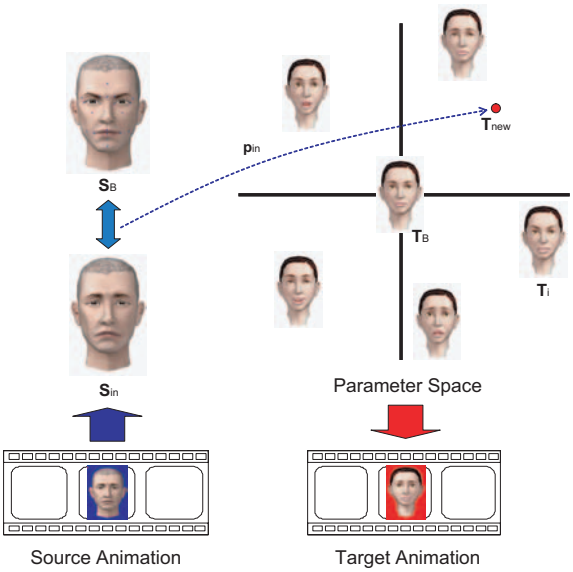


Figure 9: *Generating a new face model by blending target key-models*

$\mathbf{p}^l$ is the lth component of $\mathbf{p}$, and $A_0(\mathbf{p}) = 1$. Using the parameter vector $\mathbf{p}_i$ of each target key-model and its weight value $w_i(\mathbf{p}_i)$, we employ a least squares method to evaluate the unknown linear coefficients a_{il} of the linear bases.

To fix the second term, we compute the residuals for the target key-models:

$$w_i'(\mathbf{p}) = w_i(\mathbf{p}) - \sum_{l=0}^{\bar{N}} a_{il} A_l(\mathbf{p}) \text{ for all } i. \tag{6}$$

The radial basis function $R_j(\mathbf{p})$ is a function of the Euclidean distance between $\mathbf{p}$ and $\mathbf{p}_j$ in the parameter space:

$$R_j(\mathbf{p}) = B\left(\frac{\| \mathbf{p} - \mathbf{p}_j \|}{\alpha}\right) \text{ for } 1 \le j \le M, \tag{7}$$

where $B(\cdot)$ is the cubic B-spline function, and α is the dilation factor, which is the separation to the nearest other example in the parameter space. The radial coefficients r_{ij} are obtained by solving the matrix equation,

$$\mathbf{r}\mathbf{R} = \mathbf{w}', \tag{8}$$

where $\mathbf{r}$ is an $M \times M$ matrix of the unknown radial coefficients r_{ij}, and $\mathbf{R}$ and $\mathbf{w}'$ are the matrices of the same size defined by the radial bases and the residuals, respectively, such that $\mathbf{R}_{ij} = R_i(\mathbf{p}_j)$ and $\mathbf{w}'_{ij} = w_i'(\mathbf{p}_j)$.

With the weight functions predefined, we are now ready to explain how to blend the target key-models in runtime. For the input face model $\mathbf{S}_{in}$ at each frame of an input animation, the displacement vector $\mathbf{d}_{in}$ is computed with respect to the source base model $\mathbf{S}_B$:

$$\mathbf{d}_{in} = \mathbf{s}_{in} - \mathbf{s}_B \tag{9}$$

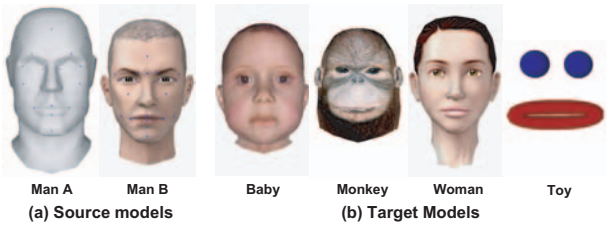


Figure 10: *Models used for the experiments*

	Man A	Man B	Baby	Monkey	Woman	Toy
V	988	1192	1253	1227	1220	931
P	1954	2194	2300	2344	2246	957

Table 1: *Model specification*

(V:Vertices, P:Polygons)

where $\mathbf{s}_{in}$ and $\mathbf{s}_B$ are respectively vectors obtained by concatenating the 3D coordinates of feature points on $\mathbf{S}_{in}$ and $\mathbf{S}_B$ as explained previously. Given this N-dimensional displacement vector $\mathbf{d}_{in}$, we then obtain the corresponding N-dimensional parameter vector $\mathbf{p}_{in}$ as follows:

$$\mathbf{p}_{in} = \mathbf{F}\mathbf{d}_{in}, \tag{10}$$

where $\mathbf{F}$ is the feature matrix defined in Equation (2).

Using the predefined weight functions for the target key-models $\mathbf{T}_i$ as given in Equation (4), we estimate the weight values $w_i(\mathbf{p}_{in})$ of all target key-models $\mathbf{T}_i, 1 \leq i \leq M$ at the parameter $\mathbf{p}_{in}$ to generate the output face model $\mathbf{T}_{new}(\mathbf{p}_{in})$:

$$\mathbf{T}_{new}(\mathbf{p}_{in}) = \mathbf{T}_B + \sum_{i=1}^{M} w_i(\mathbf{p}_{in})(\mathbf{T}_i - \mathbf{T}_B), \tag{11}$$

where $\mathbf{T}_B$ is the target base model corresponding to the source base key-model $\mathbf{S}_B$ with the neutral expression.

5. Experimental Results

As shown in Figures 10(a) and 10(b), we used two source models and four target models in our experiments. Table 1 gives the number of vertices and that of polygons in each model. We manually selected 20 feature points on each source model as described in Section 3. As input animations, we prepared two different facial animations: a facial animation of Man A with various exaggerated expressions, and a facial animation of Man B with verbal expressions combined with emotional expressions.

Our first two experiments were intended to qualitatively show the effectiveness of our approach. In the first experiment, we used Man A as the source face model and the baby and monkey models as the target face models. To clone the facial expression of Man A, we used six key-models for the source face model as well as the target face model. The first row of Figure 11 shows the input expressions of Man A sampled from the input animation. The baby and monkey models with cloned facial expressions are shown in the second and third rows of Figure 11, respectively. We can observe that the expressions of the source model were nicely cloned to the target models while reflecting the characteristic features of the target models.

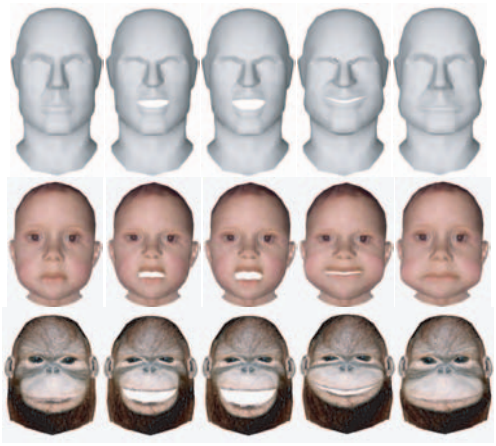


Figure 11: *Cloning expressions from Man A to the target models*

Figure 12: *Cloning expressions from Man B to the target model*

In the second experiment, we used Man B as the source model and the woman model as the target model to clone combined expressions. We prepared a total of 84 key-models: six purely emotional key-expressions (neutral, happy, angry, sad, surprised, and afraid expressions) and thirteen visemes (seven for vowels and six for consonants) for each of the six emotional key-expressions. After manually creating thirteen purely verbal expressions together with six purely emotional ones, the rest of them are obtained automatically by employing our expression compositing scheme.

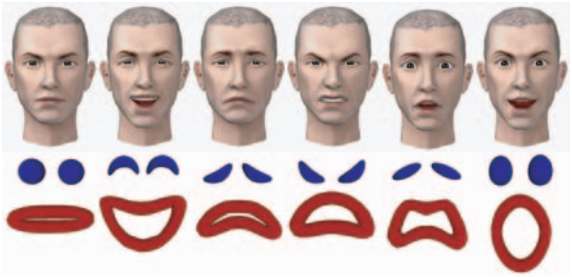


Figure 13: *Cloning expressions from Man B to the topologically different model*

As shown in Figure 12, the combined expressions of the source model were convincingly reproduced in the target model. We also cloned emotional expressions of Man B to the topologically different model as shown in the Figure 13.

	Man A ⇒ Man A		Man B ⇒ Man B	
	Ours	Noh et al.'s	Ours	Noh et al.'s
x	0.110%	0.234%	0.051%	0.176%
y	0.111%	0.196%	0.057%	0.133%
z	0.050%	0.077%	0.100%	0.213%

Table 2: *Average errors of cloned animations (source ⇒ source)*

	Man A ⇒ Baby Baby ⇒ Man A		Man B ⇒ Woman Woman ⇒ Man B	
	Ours	Noh et al.'s	Ours	Noh et al.'s
x	0.112%	2.120%	0.118%	3.076%
y	0.113%	1.936%	0.214%	3.893%
z	0.051%	1.004%	0.268%	4.183%

Table 3: *Average errors of cloned animations (source ⇒ target ⇒ source)*

The next two experiments were intended to quantitatively measure the effectiveness of our approach. In the third experiment, we used the same face model for both source and target models, that is, cloning expressions from Man A to itself and also from Man B to itself. We measured the difference of the resulting animation from the input animation for each of the models. The error at each individual frame is defined as follows:

$$\frac{\sum_{j=1}^{N} ||v_j - v'_j||}{\sum_{j=1}^{N} ||v_j||} \times 100, \qquad (12)$$

where v_j and v'_j are a vertex of the input model and the corresponding vertex of the cloned output model, respectively, and N is the number of vertices. The average error over all constituent frames is also computed for making comparisons with the previous work [18]. In the last experiment, we cloned expressions of Man A to the baby model and then the intermediate results back to Man A. For Man B, we repeated the same procedure with the woman model as the intermediate model. The average error is measured between the original and final animations.

Ideally, the vertex positions of cloned models in the resulting animations should be identical to those of the corresponding models in the input animation. Table 2 and Table 3 show the average errors of cloned animations for the x, y, and z coordinates in the last two experiments, respectively. In both experiments, our example-based approach made much smaller average errors than the previous approach [18]. This was mainly ascribed to the inherent accuracy of radial basis functions in scattered data interpolation, together with the fact that the previous method is based on 3D geometric morphing which may compromise the accuracy in trying to find a full surface correspondence between two models with just a small number of feature points.

The performance of our approach was summarized and compared with the same previous approach (see Table 4). Both schemes were implemented with C++ and OpenGL on an Intel Pentium® PC (P-4 2.4GHz processor, 512MB RAM, and GeForce 4®). As shown in the table, our scheme spent less than 1 millisecond in generating one frame for all experiments. Thus, the frame rate was over 1000Hz to guarantee a real-time performance. Compared to the previous approach, our approach required significantly less time in both preprocessing and retargetting steps. The efficiency of our approach was due to the supreme performance of the scattered data interpolation [29] that we have adopted for expression blending.

	Man A ⇒ Baby (1201 Frames)		Man B ⇒ Woman (880 Frames)	
	Ours	Noh et al.'s	Ours	Noh et al.'s
P	0.032 s	197.1 s	0.062 s	217.3 s
R	0.326 s	23.5 s	0.548 s	18.5 s
A	0.27 ms	19.6 ms	0.69 ms	21.0 ms

Table 4: *Computation time*

(P:Preprocessing, R:Retargetting, A:Average time / frame)

6. Conclusions

We have presented a novel example-based approach for cloning facial expressions from a source model to a target

model while preserving the characteristic features of the target model. Our approach consists of three parts: key-model construction, parameterization, and expression blending. For key-model construction, we present a novel scheme for compositing a pair of verbal and emotional key-models. Based on a simple but effective parameterization scheme, we are able to place the target key-models in the parameter space. To predefine the weight functions for the parameterized target key-models, we adopt multi-dimensional scattered data interpolation with radial basis functions. In runtime, a cloned target model is generated by blending the target key-models using the predefined weight functions. As shown in the experimental results, our approach has accurately performed expression cloning with great efficiency.

One limitation of our method might be that it requires animators to prepare a set of key-models for source and target models as a preprocess, but at the same time it can be thought of as an advantage in that it allows for human control of the cloning results so that the characteristics of the target model are fully reflected. Another limitation is that our method cannot correctly clone an expression when it falls too far outside from the basis of the constructed source key-models. In this case, it would be better to select the source key-models from the source animation frames as in [2], rather than constructing them based on generic human facial key-expressions.

In future, we are planning to extend our approach to region-based expression cloning. According to results in psychology, a face can be split into several regions that behave as coherent units [5]. For example, the parts such as eyes, eyebrows, and forehead are used for emotional expressions, and those such as mouth, cheeks, and chin are used for verbal expressions. If we prepare example data for each of the parts separately, we could generate more diverse expressions with less example data. To achieve this, we need an effective way to combine separately-generated facial parts seamlessly.

References

1. B. Allen, B. Curless and Z. Popovic. "Articulated body deformation from range scan data", In *Proceedings of SIGGRAPH 02*, pp. 612-619, 2002.
2. C. Bregler, L. Loeb and E. Chuang and H. Deshpande. "Turning to the masters: Motion capturing cartoons", In *Proceedings of SIGGRAPH 02*, pp. 399-407, 2002.
3. I. Buck, A. Finkelstein, C. Jacobs, A. Klein, D. Salesin, J. Seims, R. Szeliski, and K. Toyama. "Performance-Driven Hand-Drawn Animation", In *Proceedings of Symposium on Non-Photorealistic Animation and Rendering*, 2000.
4. E. Chuang and C. Bregler. "Performance Driven Facial Animation using Blendshape Interpolation", *Standford University Computer Science Technical Report*, CS-TR-2002-02, April 2002.
5. P. Ekman and W. V. Friesen. "Unmasking the face: A guide to recognizing emotions from facial clues", Prentice-Hall Inc., 1975.
6. D. Fidaleo, J-Y. Noh, T. Kim, R. Enciso and U. Neumann. "Classification and Volume Morphing for Performance-Driven Facial Animation", In *Proceedings of Internatinoal Workshop on Digital and Computational Video*, 2000.
7. C. G. Fisher. "Confusions among visually perceived consonants.", *Jour. Speech and Hearing Research*, **11**, pp. 796-804, 1968.
8. M. Gleicher. "Retargetting motion to new Characters", *ACM SIGGRAPH 98 Conference Proceedings*, pp. 33-42, 1998.
9. B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin. "Making Faces", *ACM SIGGRAPH 98 Conference Proceedings*, pp. 55-66, July 1998.
10. I. T. Jollife. "Principal components analysis", New York: Spinger, 1986.
11. P. Kalra and A. Mangili and N. M. Thalmann and D. Thalmann. "Simulation of facial muscle actions based on rational free from deformations", In *Proceedings of Eurographics 92*, pp. 59-69, 1992.
12. C. Kouadio and P. Poulin and P. Lachapelle. "Real-Time Facial Animation Based Upon a Bank of 3D Facial Expressions", *Proc. Computer Animation*, pp. 128-136, 1998.
13. G. A. Kalberer and L. V. Gool. "Face Animation Based on Observed 3D Speech Dynamics", *Computer Animation 2001*, pp. 20–27, November 2001.
14. J. Lee and S. Y. Shin. "A hierarchical approach to interactive motion editing for human-like figures", *Proceedings of SIGGRAPH 99*, pp. 39–48, 1999.
15. Y. C. Lee, D. Terzopoulos and K. Waters. "Realistic modeling for facial animation", In *Proceedings of SIGGRAPH 95*, pp. 55-62, 1995.
16. Marc Levoy and Pat Hanrahan. "Light Field Rendering", *Proceedings of SIGGRAPH 96*, pp. 31-42, 1996.
17. J. P. Lewis, M. Cordner, and N. Fong. "Pose Space Deformation: A Unified Approach to Shape Interpolation and Skeleton-Drive Deformation", *ACM SIGGRAPH 2000 Conference Proceedings*, pp. 165-172, July 2000.
18. J. Y. Noh and U. Neumann. "Expression cloning", In *Proceedings of SIGGRAPH 01*, pp. 277-288, 2001.
19. S. I. Park, H. J. Shin and S. Y. Shin. "On-line locomotion generation based on motion blending", In *ACM SIGGRAPH Symposium on Computer Animation*, pp. 105-111, 2002.

20. F. I. Parke and K. Waters. *Computer Facial Animation.* A K Peters, 289 Linden Street, Wellesley, MA 02181, 1996.

21. F. I. Parke. "Computer generated animation of faces", Master's thesis, University of Utah, 1972.

22. F. I. Parke. "Parameterized models for facial animation", In *IEEE Computer Graphics and Applications*, Vol. 2, No. 9, pp. 61-68, 1982.

23. F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D.H. Salesin. "Synthesizing Realistic Facial Expressions from Photographs", In *ACM SIGGRAPH 98 Conference Proceedings*, pp. 75-84, July 1998.

24. F. Pighin, R. Szeliski and D. H. Salesin. "Resynthesizing facial animation through 3D model-based tracking", *Proceedings of International Conference on Computer Vision 99*, pp. 143-150, July 1999.

25. S. M. Platt and N. I. Badler. "Animating facial expressions", In *Computer Graphics*, Vol. 15(3), pp. 245-252, 1981.

26. Z. Popovic and A. Witkin. "Physically based motion transformation", In *Proceedings of SIGGRAPH 99*, pp. 11-20, 1999.

27. C. Rose, M. F. Cohen and B. Bodenheimer. "Verbs and adverbs: Multidimensional motion interpolation", In *IEEE Computer Graphics and Applications*, Vol. 18(5), pp. 32-40, 1998.

28. J. A. Russel. "A Circomplex Model of Affect", In *J. Personality and Social Psychology*, Vol. 39, pp. 1161-1178, 1980.

29. P. -P. Sloan and C. F. Rose and Michael F. Cohen. "Shape by example", In *Proceedings of 2001 Symposium on Interactive 3D Graphics*, pp. 135-144, 2001.

30. D. Terzopoulos and K. Waters. "Physically-based facial modeling, analysis, and animation", In *Journal of Visualization and Computer Animation*, Vol. 1, No. 4, pp. 73-80, 1990.

31. K. Waters. "A muscle model for animating three-dimensional facial expressions", In *Proceedings of SIGGRAPH 87*, pp. 17-24, 1987.

32. Li-Yi Wei and Marc Levoy. "Fast Texture Synthesis Using Tree-Structured Vector Quantization", In Proceedings of SIGGRAPH 00, pp. 479-488, 2000.

33. L. Williams. "Performance driven facial animation", In *Proceedings of SIGGRAPH 90*, pp. 235-242, 1990.

Appendix

To compute the importance value of every vertex, we have empirically derived the following three rules: First, the importance of a vertex is proportional to the norm of displacement vector. Second, even a vertex with a small displacement is considered to be important if it has a neighboring vertex with a large displacement. Finally, a vertex of high importance is constrained by verbal expressions, and a vertex of low importance drives emotional expressions.

According to those rules, we compute the importance of each vertex in three steps. In the first two steps, two independent importance values are computed from the verbal key-models and the emotional key-models, respectively. Then, they are combined to give the importance in the final step. Let $p_1(v_i)$ and $e_1(v_i)$, $1 \le i \le n$ be the verbal and emotional importances, respectively. In the first step, these importances are computed from the maximum norms of the displacement vectors of each vertex v_i over their respective key-models:

$$p_1(v_i) = \max_j(|v_i^{P_j} - v_i|) / \max_{j,k}(|v_k^{P_j} - v_k|), \text{ and}$$
$$e_1(v_i) = \max_j(|v_i^{E_j} - v_i|) / \max_{j,k}(|v_k^{E_j} - v_k|),$$

where $v_i^{P_j}$ and $v_i^{E_j}$ respectively denote the vertices in a verbal key-model P_j and an emotional key-model E_j corresponding to a vertex v_i in the base model. Note that we normalize the importances so that their values range from 0 to 1. We then propagate the importance value of each vertex to the neighboring vertices if it is big enough. Thus, in the second step, the importances $p_2(v_i)$ and $e_2(v_i)$ are obtained as follows:

$$p_2(v_i) = \max(\{p_1(v_i)\} \cup \{p_1(v_j) \mid |v_i - v_j| < L_p,\ p_1(v_j) > S_1\}),$$
$$e_2(v_i) = \max(\{e_1(v_i)\} \cup \{e_1(v_j) \mid |v_i - v_j| < L_e,\ e_1(v_j) > S_1\}),$$

where S_1, L_p, and L_e are control parameters. In the final step, the importance α_i of the vertex v_i is obtained as follows:

$$\alpha_i = \begin{cases} p_2(v_i)(1 - e_2(v_i)), & \text{if } p_2(v_i) < S_2 \\ 1 - (1 - p_2(v_i))e_2(v_i), & \text{otherwise.} \end{cases} \quad (13)$$

Equation (13) adjusts importance values so that they are clustered near both extremes, that is, zero and one. Figure 14 shows the importance distributions for verbal and emotional expressions after the first, second, and third steps. Brighter regions indicate higher importance values.

(a) $p_1(\mathbf{v})$ (b) $e_1(\mathbf{v})$ (c) $p_2(\mathbf{v})$ (d) $e_2(\mathbf{v})$ (e) α

Figure 14: *Importance distributions*

To appear in SIGGRAPH 2005.

Face Transfer with Multilinear Models

Daniel Vlasic* † Matthew Brand † Hanspeter Pfister Jovan Popović

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology

† Mitsubishi Electric Research Laboratories

Figure 1: Face Transfer with multilinear models gives animators decoupled control over facial attributes such as identity, expression, and viseme. In this example, we combine pose and identity from the first frame, surprised expression from the second, and a viseme (mouth articulation for a sound midway between "oo" and "ee") from the third. The resulting composite is blended back into the original frame.

Abstract

Face Transfer is a method for mapping videorecorded performances of one individual to facial animations of another. It extracts visemes (speech-related mouth articulations), expressions, and three-dimensional (3D) pose from monocular video or film footage. These parameters are then used to generate and drive a detailed 3D textured face mesh for a target identity, which can be seamlessly rendered back into target footage. The underlying face model automatically adjusts for how the target performs facial expressions and visemes. The performance data can be easily edited to change the visemes, expressions, pose, or even the identity of the target—the attributes are separably controllable. This supports a wide variety of video rewrite and puppetry applications.

Face Transfer is based on a multilinear model of 3D face meshes that separably parameterizes the space of geometric variations due to different attributes (e.g., identity, expression, and viseme). Separability means that each of these attributes can be independently varied. A multilinear model can be estimated from a Cartesian product of examples (identities $\times$ expressions $\times$ visemes) with techniques from statistical analysis, but only after careful preprocessing of the geometric data set to secure one-to-one correspondence, to minimize cross-coupling artifacts, and to fill in any missing examples. Face Transfer offers new solutions to these problems and links the estimated model with a face-tracking algorithm to extract pose, expression, and viseme parameters.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.4.9 [Image Processing and Computer Vision]: Applications;

Keywords: Facial Animation, Computer Vision—Tracking

*MIT CSAIL, The Stata Center, 32 Vassar Street, Cambridge, MA 02139, USA

1 Introduction

Performance-driven animation has a growing role in film production because it allows actors to express content and mood naturally, and because the resulting animations have a degree of realism that is hard to obtain from synthesis methods [Robertson 2004]. The search for the highest quality motions has led to complex, expensive, and hard-to-use systems. This paper introduces new techniques for producing compelling facial animations that are inexpensive, practical, versatile, and well suited for editing performances and retargeting to new characters.

Face Transfer extracts performances from ordinary video footage, allowing the transfer of facial action of actors who are unavailable for detailed measurement, instrumentation, or for rerecording with specialized scanning equipment. Expressions, visemes (speech-related mouth articulations), and head motions are extracted automatically, along with a performance-driven texture function. With this information in hand, our system can either rewrite the original footage with adjusted expressions and visemes or transfer the performance to a different face in a different footage.

Multilinear models are ideally suited for this application because they can describe face variations with separable attributes that can be estimated from video automatically. In this paper, we estimate such a model from a data set of three-dimensional (3D) face scans that vary according to expression, viseme, and identity. The multilinear model decouples the three attributes (i.e., identity or viseme can be varied while expression remains constant) and encodes them consistently. Thus the attribute vector that encodes a smile for one person encodes a smile for every face spanned by the model, regardless of identity or viseme. Yet the model captures the fact that every person smiles in a slightly different way. Separability and consistency are the key properties that enable the transfer of a performance from one face to another without a change in content.

Contributions. This paper describes a general, controllable, and practical system for facial animation. It estimates a multilinear model of human faces by examining geometric variations between 3D face scans. In principle, given a large and varied data set, the model can generate any face, any expression, any viseme. As proof of concept, we estimate the model from a couple of geometric data sets: one with 15 identities and 10 expressions, and another with

16 identities, 5 expressions, and 5 visemes. Existing estimation algorithms require perfect one-to-one correspondence between all meshes, and a mesh for every possible combination of expression, viseme, and identity. Because acquiring the full Cartesian product of meshes and putting them into dense correspondence is extremely difficult, this paper introduces methods for populating the Cartesian product from a sparse sampling of faces, and for placing unstructured face scans into correspondence with minimal cross-coupling artifacts.

By linking the multilinear model to optical flow, we obtain a single-camera tracker that estimates performance parameters and detailed 3D geometry from video recordings. The model defines a mapping from performance parameters back to 3D shape, thus we can arbitrarily mix pose, identity, expressions, and visemes from two or more videos and render the result back into a target video. As a result, the system provides an intuitive interface for both animators (via separably controllable attributes) and performers (via acting). And because it does not require performers to wear visible facial markers or to be recorded by special face-scanning equipment, it is an inexpensive and easy-to-use facial animation system.

2 Related Work

Realistic facial animation remains a fundamental challenge in computer graphics. Beginning with Parke's pioneering work [1974], desire for improved realism has driven researchers to extend geometric models [Parke 1982] with physical models of facial anatomy [Waters 1987; Lee et al. 1995] and to combine them with non-linear finite element methods [Koch et al. 1996] in systems that could be used for planning facial surgeries. In parallel, Williams presented a compelling argument [1990] in favor of performance-driven facial animation, which anticipated techniques for tracking head motions and facial expressions in video [Li et al. 1993; Essa et al. 1996; DeCarlo and Metaxas 1996; Pighin et al. 1999]. A more expensive alternative could use a 3D scanning technique [Zhang et al. 2004], if the performance can be re-recorded with such a system.

Much of the ensuing work on face estimation and tracking relied on the observation that variation in faces is well approximated by a linear subspace of low dimension [Sirovich and Kirby 1987]. These techniques estimate either linear coefficients for known basis shapes [Bascle and Blake 1998; Brand and Bhotika 2001] or both the basis shapes and the coefficients, simultaneously [Bregler et al. 2000; Torresani et al. 2001]. In computer graphics, the combination of accurate 3D geometry with linear texture models [Pighin et al. 1998; Blanz and Vetter 1999] produced striking results. In addition, Blanz and Vetter [1999] presented a process for estimating the shape of a face in a single photograph, and a set of controls for intuitive manipulation of appearance attributes (thin/fat, feminine/masculine).

These and other estimation techniques share a common challenge of decoupling the attributes responsible for observed variations. As an early example, Pentland and Sclaroff estimate geometry of deformable objects by decoupling linear elastic equations into orthogonal vibration modes [1991]. In this case, modal analysis uses eigen decomposition to compute the independent vibration modes. Similar factorizations are also relied upon to separate variations due to pose and lighting, pose and expression, identity and lighting, or style and content in general [Freeman and Tenenbaum 1997; Bregler et al. 2000; DeCarlo and Metaxas 2000; Georghiades et al. 2001; Cao et al. 2003].

A technical limitation of these formulations is that each pair of factors must be considered in isolation; they cannot easily decouple variations due to a combination of more than two factors. The extension of such two-mode analysis to more modes of variation was first introduced by Tucker [1966] and later formalized and improved on by Kroonenberg and de Leeuw [1980]. These techniques were successfully applied to multilinear analysis of images [Vasilescu and Terzopoulos 2002; Vasilescu and Terzopoulos 2004].

This paper describes multilinear analysis of *three-dimensional (3D)* data sets and generalizes face-tracking techniques to create a unique performance-driven system for animation of any face, any expression, and any viseme. In consideration of similar needs, Bregler and colleagues introduced a two-dimensional method for transferring mouth shapes from one performance to another [1997]. The method is ideal for film dubbing—a problem that could also be solved without performance by first learning the mouth shapes on a canonical data set and then generating new shapes for different texts [Ezzat and Poggio 2000]. These methods are difficult to use for general performance-driven animation because they cannot change emotions of a face. Although the problem can be resolved by decoupling emotion and content via two-mode analysis [Chuang et al. 2002], all three techniques are view specific, which presents difficulties when view, illumination, or both have to change.

Our Face Transfer learns a model of 3D facial geometry variations in order to infer a particular face shape from 2D images. Previous work combines identity and expression spaces by copying deformations from one subject onto the geometry of other faces [DeCarlo and Metaxas 2000; Blanz et al. 2003; Chai et al. 2003]. Expression cloning [Noh and Neumann 2001; Sumner and Popović 2004] improves on this process but does not account for actor-specific idiosyncrasies that can be revealed by statistical analysis of the entire data set (i.e., the mesh vertex displacements that produce a smile should depend on who is smiling and on what they are saying at the same time). Other powerful models of human faces have been explored [Wang et al. 2004] at the cost of making the estimation and transfer of model parameters more difficult. This paper describes a method that incorporates all such information through multilinear analysis, which naturally accommodates variations along multiple attributes.

3 Multilinear Algebra

Multilinear algebra is a higher order generalization of linear algebra. In this section we provide insight behind the basic concepts needed for understanding of our Face Transfer system. De Lathauwer's dissertation [1997] provides a comprehensive treatment of this topic. Concise overviews have also been published in the graphics and vision literature [Vasilescu and Terzopoulos 2002; Vasilescu and Terzopoulos 2004].

Tensors. The basic mathematical object of multilinear algebra is the tensor, a natural generalization of vectors (1^{st} order tensors) and matrices (2^{nd} order tensors) to multiple indices. An N^{th}-order tensor can be thought of as a block of data indexed by N indices: $\mathcal{T} = (t_{i_1 i_2 \ldots i_N})$. Figure 2 shows a 3^{rd}-order (or 3-mode) tensor with a total of $d_1 \times d_2 \times d_3$ elements. Different modes usually correspond to particular attributes of the data (e.g, expression, identity, etc.).

Mode Spaces. A matrix has two characteristic spaces, row and column space; a tensor has one for each mode, hence we call them *mode spaces*. The $d_1 \times d_2 \times d_3$ 3-tensor in Figure 2 has three mode spaces. Viewing the data as a set of d_1-dimensional vectors stored parallel to the first axis (Figure 2b), we can define the mode-1 space as the span of those vectors. Similarly, mode-2 space is defined as the span of the vectors stored parallel to the second axis, each of size d_2 (Figure 2c). Finally, mode-3 space is spanned by vectors in the third mode, of dimensionality d_3 (Figure 2d). Multilinear algebra revolves around the analysis and manipulation of these spaces.

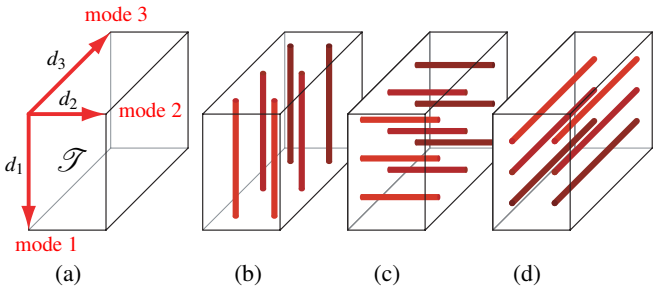


Figure 2: In (a) we show a 3^{rd}-order (3-mode) tensor $\mathscr{T}$ whose modes have d_1, d_2, and d_3 elements respectively. Depending on how we look at the data within the tensor, we can identify three mode spaces. By viewing the data as vectors parallel to the first mode (b), we define mode-1 space as the span of those vectors. Similarly, mode-2 space is spanned by vectors parallel to the second mode (c), and mode-3 space by vectors in the third mode (d).

Mode-n Product. The most obvious way of manipulating mode spaces is via linear transformation, officially referred to as the *mode-n product*. It is defined between a tensor $\mathscr{T}$ and a matrix $\mathbf{M}$ for a specific mode n, and is written as a multiplication with a subscript: $\mathscr{T} \times_n \mathbf{M}$. This notation indicates a linear transformation of vectors in $\mathscr{T}$'s mode-n space by the matrix $\mathbf{M}$. Concretely, $\mathscr{T} \times_2 \mathbf{M}$ would replace each mode-2 vector $\mathbf{v}$ (Figure 2c) with a transformed vector $\mathbf{Mv}$.

Tensor Decomposition. One particularly useful linear transformation of mode data is the N-mode singular value decomposition (N-mode SVD). It rotates the mode spaces of a *data tensor* $\mathscr{T}$ producing a *core tensor* $\mathscr{C}$, whose variance monotonically decreases from first to last element in each mode (analogous to matrix SVD). This enables us to truncate the insignificant components and get a reduced model of our data.

Mathematically, N-mode SVD can be expressed with mode products

$$\mathscr{T} \times_1 \mathbf{U}_1^\top \times_2 \mathbf{U}_2^\top \times_3 \mathbf{U}_3^\top \cdots \times_N \mathbf{U}_N^\top = \mathscr{C} \qquad (1)$$

$$\Longrightarrow \quad \mathscr{T} = \mathscr{C} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \mathbf{U}_3 \cdots \times_N \mathbf{U}_N, \qquad (2)$$

where $\mathscr{T}$ is the data tensor, $\mathscr{C}$ is the core tensor, and $\mathbf{U}_i$'s (or more precisely their transposes) rotate the mode spaces. Each $\mathbf{U}_i$ is an orthonormal matrix whose columns contain left singular vectors of the ith mode space, and can be computed via regular SVD of those spaces [De Lathauwer 1997]. Since variance is concentrated in one corner of the core tensor, data can be approximated by

$$\mathscr{T} \simeq \mathscr{C}_{reduced} \times_1 \check{\mathbf{U}}_1 \times_2 \check{\mathbf{U}}_2 \times_3 \check{\mathbf{U}}_3 \cdots \times_N \check{\mathbf{U}}_N\,, \qquad (3)$$

where $\check{\mathbf{U}}_i$'s are truncated versions of $\mathbf{U}_i$'s with last few columns removed. This truncation generally yields high quality approximations but it is not optimal—one of several matrix-SVD properties that do not generalize in multilinear algebra. One can obtain a better approximation with further refinement of $\check{\mathbf{U}}_i$'s and $\mathscr{C}_{reduced}$ via alternating least squares [De Lathauwer 1997].

4 Multilinear Face Model

To construct the multilinear face model, we first acquire a range of 3D face scans, put them in full correspondence, appropriately arrange them into a data tensor (Figure 3), and use the N-mode SVD to compute a model that captures the face geometry and its variation due to attributes such as identity and expression.

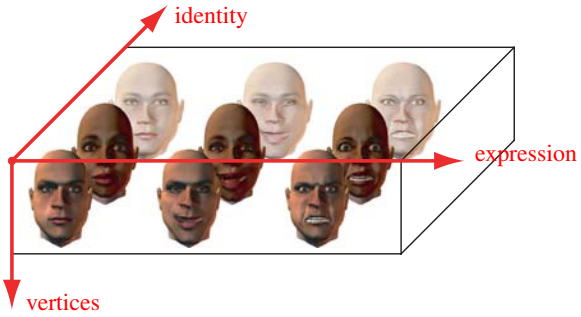


Figure 3: Data tensor for a bilinear model that varies with identity and expression; the first mode contains vertices, while the second and third modes correspond to expression and identity respectively. The data is arranged so that each slice along the second mode contains the same expression (in different identities) and each slice along the third mode contains the same identity (in different expressions). In our trilinear experiments we have added a fourth mode, where scans in each slice share the same viseme.

4.1 Face Data

We demonstrate our proof-of-concept system on two separate face models: a bilinear model, and a trilinear model. Both were estimated from detailed 3D scans ($\sim$ 30K vertices) acquired with 3dMD/3Q's structured light scanner (`http://www.3dmd.com/`) in a process similar to regular flash photography, although our methods would apply equally to other geometric data sets such as motion capture. As a preprocess, the scans were smoothed using the bilateral filter [Jones et al. 2003] to eliminate some of the capture noise. The subject pool included men, women, Caucasians, and Asians, from the mid-20s to mid-50s.

Bilinear model. 15 subjects were scanned performing the same 10 facial expressions. The expressions were picked for their familiarity as well as distinctiveness, and include neutral, smile, frown, surprise, anger, and others. The scans were assembled into a third order (3-mode) data tensor (30K vertices $\times$ 10 expressions $\times$ 15 identities). After N-mode SVD reduction, the resulting bilinear model offers 6 knobs for manipulating expression and 9 for identity.

Trilinear model. 16 subjects were asked to perform 5 visemes in 5 different expressions (neutral, smiling, scowling, surprised, and sad). The visemes correspond to the boldfaced sounds in **m**an, c**a**r, **ee**l, t**oo**, and **sh**e. Principal components analysis of detailed speech motion capture indicated that these five expressions broadly span the space of lip shapes, and should give a good approximate basis for all other visemes—with the possible exception of exaggerated fricatives. The resulting fourth order (4-mode) data tensor (30K vertices $\times$ 5 visemes $\times$ 5 expressions $\times$ 16 identities) was decomposed to yield a trilinear model providing 4 knobs for viseme, 4 for expression, and 16 for identity (we have kept the number of knobs large since our data sets were small).

4.2 Correspondence

Training meshes that are not placed in perfect correspondence can considerably muddle the question of how to displace vertices to change one attribute versus another (e.g. identity versus expression), and thus the multilinear analysis may not give a model with good separability. We show here how to put a set of unstructured face scans into correspondence suitable for multilinear analysis.

Despite rapid advances in automatic parameterization of meshes (e.g., [Praun and Hoppe 2003; Gotsman et al. 2003]), it took consid-

erable experimentation to place many facial scans into detailed correspondence. The principal complicating factors are that the scans do not have congruent mesh boundaries, and the problem of matching widely varied lip deformations does not appear to be well served by conformal maps or local isometric constraints. This made it necessary to mark a small number of feature points in order to bootstrap correspondence-finding across large deformations.

We developed a protocol for a template-fitting procedure [Allen et al. 2003; Sumner and Popović 2004], which seeks a minimal deformation of a parameterized template mesh that fits the surface implied by the scan. The optimization objective, minimized with gradient descent, balances overall surface similarity, proximity of manually selected feature points on the two surfaces, and proximity of reference vertices to the nearest point on the scanned surface. We manually specified 42 reference points on a reference facial mesh and on a neutral (m-viseme) scan. After rigidly aligning the template and the scan with Procrustes' alignment, we deformed the template mesh into the scan: at first, weighing the marked correspondences heavily and afterwards emphasizing vertex proximity. For the trilinear model, the remaining m-viseme (closed-mouth) scans were marked with 21 features around eyebrows and lips, rigidly aligned to upper-face geometry on the appropriate neutral scans, and then non-rigidly put into correspondence as above. Finally, all other viseme scans were similarly put into correspondence with the appropriate closed-mouth scan, using the 18 features marked around the lips.

4.3 Face Model

Equation (3) shows how to approximate the data tensor by mode-multiplying a smaller core tensor with a number of truncated orthogonal matrices. Since our goal is to output vertices as a function of attribute parameters, we can decompose the data tensor without factoring along the mode that corresponds to vertices (mode-1), changing Equation (3) to:

$$\mathcal{T} \simeq \mathcal{M} \times_2 \check{\mathbf{U}}_2 \times_3 \check{\mathbf{U}}_3 \cdots \times_N \check{\mathbf{U}}_N \,, \tag{4}$$

where $\mathcal{M}$ can now be called the *multilinear model* of face geometry. Mode-multiplying $\mathcal{M}$ with $\check{\mathbf{U}}_i$'s approximates the original data. In particular, mode-multiplying it with one row from each $\check{\mathbf{U}}_i$ reconstructs exactly one original face (the one corresponding to the attribute parameters contained in that row). Therefore, to generate an arbitrary interpolation (or extrapolation) of original faces, we can mode-multiply the model with a linear combination of rows for each $\check{\mathbf{U}}_i$. We can write

$$\mathbf{f} = \mathcal{M} \times_2 \mathbf{w_2}^\top \times_3 \mathbf{w_3}^\top \cdots \times_N \mathbf{w_N}^\top \,, \tag{5}$$

where $\mathbf{w_i}$ is a column vector of parameters (weights) for the attribute corresponding to i^{th} mode, and $\mathbf{f}$ is a column vector of vertices describing the resulting face.

4.4 Missing Data

Building the multilinear model from a set of face scans requires capturing the full Cartesian product of different face attributes, (i.e., all expressions and visemes need to be captured for each person). Producing a full data tensor is not always practical for large data sets. For example, a certain person might have trouble performing some expressions on cue, or a researcher might add a new expression to the database but be unable reach all the previous subjects. In our case, data corruption and subsequent unavailability of a subject led to an incomplete tensor. The problem becomes more evident if we add *age* as one of the attributes, where we cannot expect to scan each individual throughout their entire lives. In all these cases, we would still like to include a person's successful scans in the model, and fill in the missing ones with the most likely candidates. This process is known as imputation.

There are many possible schemes for estimating a model from incomplete data. A naive imputation would find a complete subtensor, use it to estimate a smaller model, use that to predict a missing face, use that to augment the data set, and repeat. In a more sophisticated Bayesian setting, we would treat the missing data as hidden variables to be MAP estimated (imputed) or marginalized out. Both approaches require many iterations over a huge data set; Bayesian methods are particularly expensive and generally require approximations for tractability. With MAP estimation and naive imputation, the results can be highly dependent on the order of operations. Because it fails to exploit all available constraints, the naive imputative scheme generally produces inferior results.

Here we use an imputative scheme that exploits more available constraints than the naive one, producing better results. The main intuition, which we formalize below, is that any optimization criteria can be linearized in a particular tensor mode, where it yields a matrix factorization problem with missing values. Then we leverage existing factorization schemes for incomplete matrices, where known values contribute a set of linear constraints on the missing values. These constraints are then combined and solved in the least-squares sense.

Description. Our algorithm consists of two steps. First, for each mode we assemble an *incomplete* matrix whose columns are the corresponding mode vectors. We then seek a subspace decomposition that best reconstructs the known values of that matrix. The decomposition and the known values provide a set of linear constraints for the missing values. This can be done with off-the-shelf imputative matrix factorizations (e.g., PPCA [Tipping and Bishop 1999], SPCA [Roweis 1997], or ISVD [Brand 2002]). Typically these algorithms estimate a low-rank subspace from the complete vectors of the mode and use that to predict missing values in the incomplete columns (and/or update the subspace). In our experiments we used the standard PPCA formulation for filling in missing values, which reduces to a system of linear equations that relate unknown values to known values through the estimated mean and covariance of the vectors in the mode space. Second, the linear constraints are combined through the missing elements, because they are shared across all groups of modal vectors and must be filled in with consistent values. To that end, we collect the linear equations that determine a particular missing value in all the modes, and solve them together. For example, if two missing values co-occur in some mode vector, then they must be jointly estimated. We update the mean and covariance for each decomposition and repeat the two steps until convergence.

Evaluation. Figure 4 contrasts the results of this method with faces predicted by our generalization of the simple method proposed by Blanz and colleagues [2003]. In their formulation the same displacement vectors that make one person smile are copied over onto every other identity. Because our data set includes smiles for more than one person, we extend that approach to copy their average. In this particular example, 15% of real faces where held out of the trilinear data set and predicted by our imputation scheme and the simple averaging scheme. Note how the multilinear prediction is closer to the truth in most examples, even predicting some individual idiosyncrasies in puckers and smiles. The simple averaging scheme, however, seems to do a better job at keeping the lips sealed for closed-mouth faces (bottom row of Figure 4). We could obtain better results by preferentially weighting detail around the mouth.

In our earlier trilinear experiments, we found that ISVD-based imputations predicted how faces vary from the mean with less than 9% relative error (Frobenius norm of the total error divided by the

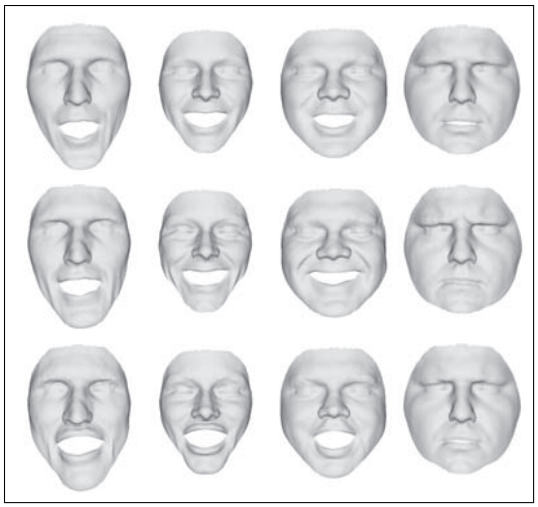


Figure 4: From top to bottom: Prediction of held-out faces with our imputation scheme (on the trilinear model), the actual face, and a simple averaging scheme.

norm of the held-out face variations) for up to 50% missing data. In general, the predictions are drawn towards the mean of the known data. Closed-mouth expressions, which are under-represented in our data and thus lie far from the mean, were not predicted as well as other expressions. That can be fixed by reweighting the data. Tests performed on synthetic data indicate that the quality of imputation increases as the data set grows in size, even if significant portions of it are missing. The reason why is that if the data is truly low-dimensional in each of the modes, the missing samples will fall within the span and density of the known ones.

Probabilistic Interpretation. The above algorithm fills in missing data by approximating the true multilinear distribution. The form of this approximation is made precise by a probabilistic interpretation, which starts from a multilinear generative model

$$\mathcal{T} = \mathcal{M} \times_2 \check{\mathbf{U}}_2 \times_3 \check{\mathbf{U}}_3 \cdots \times_N \check{\mathbf{U}}_N + \nu,$$

where $\mathcal{T}$ and $\mathcal{M}$ are the data and model tensors, $\check{\mathbf{U}}_i$ is the i-th modal subspace, and ν is a Gaussian noise source. Filling in missing data according to this model is computationally expensive. Instead, we approximate the true likelihood with a geometric average of Gaussians

$$p(\mathcal{T}|\mathcal{M}, \{\check{\mathbf{U}}_i\}_{i=2}^N) \approx \prod_{j=2}^{N} q_j(\mathcal{T}, \mathcal{M}, \{\check{\mathbf{U}}_i\}_{i=2}^N)^{1/N}.$$

Each Gaussian $q_j(\mathcal{T}, \mathcal{M}, \{\check{\mathbf{U}}_i\}_{i=2}^N) \doteq \mathcal{N}(\mathcal{T}|\check{\mathbf{U}}_j\mathbf{J}_j, \sigma_j^2)$ is found by fixing $\{\check{\mathbf{U}}_i\}_{i\neq j}$ and turning the tensor Equation (4) into matrix form: $\mathbf{T}_j = \check{\mathbf{U}}_j\mathbf{J}_j$. Here, columns of $\mathbf{T}_j$ are the mode-j vectors of $\mathcal{T}$, and the columns of $\mathbf{J}_j$ are the mode-j vectors of $\mathcal{M} \times_2 \check{\mathbf{U}}_2 \cdots \times_{j-1} \check{\mathbf{U}}_{j-1} \times_{j+1} \check{\mathbf{U}}_{j+1} \cdots \times_N \check{\mathbf{U}}_N$. The resulting likelihood becomes:

$$p(\mathcal{T}|\mathcal{M}, \{\check{\mathbf{U}}_i\}_{i=2}^N) \approx \prod_{j=2}^{N} \mathcal{N}(\mathcal{T}|\check{\mathbf{U}}_j\mathbf{J}_j, \sigma_j^2)^{1/N},$$

which can be maximized efficiently.

Taking logarithms and discarding constant factors such as N and σ_j, we seek to minimize the sum-squared error

$$\sum_{j=2}^{N} \|\mathbf{T}_j - \check{\mathbf{U}}_j\mathbf{J}_j\|_F^2$$

Each term of the summation presents a matrix factorization problem with missing values, where $\check{\mathbf{U}}_j$ and $\mathbf{J}_j$ are treated as unknown factors of the incomplete matrix $\mathbf{T}_j$, and are solved for using PPCA as described above.

5 Face Transfer

One produces animations from a multilinear model by varying the attribute parameters (the elements of the $\mathbf{w}_i$'s) as if they were dials, and generating mesh coordinates from Equation 5. The N-mode SVD conveniently gives groups of dials that separately control identity, expression and viseme. Within each group, the dials do not correspond to semantically meaningful deformations (such as smile or frown), but rather reflect the deformations that account for most variance. However, the dials can be "tuned" to reflect deformations of interest through a linear transform of each $\mathbf{w}_i$. This approach was successfully applied in [Allen et al. 2003] to make their body shape dials correspond to height and weight. A similar linear scheme was employed in [Blanz and Vetter 1999]. In general, dial-based systems are currently used on most of the deformable models in production, but only skilled animators can create believable animations (or even stills) with them. To give similar power to a casual user, we have devised a method that automatically sets model parameters from given video data. With this tool, a user can enact a performance in front of a camera, and have it automatically transferred to the model.

5.1 Face Tracking

To link the parameters of a multilinear model to video data, we use optical flow in conjunction with the weak-perspective camera model. Using the symmetric Kanade-Lucas-Tomasi formulation [Birchfield 1996], we express the frame-to-frame motion of a tracked point with a linear system:

$$\mathbf{Zd} = \mathbf{Z}(\mathbf{p} - \mathbf{p}_0) = \mathbf{e}\,. \tag{6}$$

Here, the 2-vector $\mathbf{d}$ describes the image-space motion of the point, also expressed as the difference between the point's true location $\mathbf{p}$ and its current best guess $\mathbf{p}_0$ (if we have no guess, then $\mathbf{p}_0$ is the location from the previous frame). Matrix $\mathbf{Z}$ and vector $\mathbf{e}$ contain spatial and temporal intensity gradient information in the surrounding region [Birchfield 1996].

Using a weak-perspective imaging model, the point position $\mathbf{p}$ can be expanded in terms of rigid head-motion parameters and non-rigid facial shape parameters, which are constrained by the multilinear model:

$$\mathbf{Z}(s\mathbf{R}\mathbf{f}_i + \mathbf{t} - \mathbf{p}_0) = \mathbf{e}\,, \tag{7}$$

where the rigid parameters consist of scale factor s, the first two rows of a 3D rotation matrix $\mathbf{R}$, and the image-space translation $\mathbf{t}$. The 3D shape $\mathbf{f}$ comes from the multilinear model through Equation (5), with $\mathbf{f}_i$ indicating the ith 3D vertex being tracked.

Solving for the pose and all the multilinear weights from a pair of frames using Equation (7) is not a well-constrained problem. To simplify the computation, we use a coordinate-descent method: we let only one of the face attributes vary at a time by fixing all the others to their current guesses. This transforms the multilinear problem into a linear one, as described below, which we solve with standard techniques that simultaneously compute the rigid pose along with the linear weights from a pair of frames [Bascle and Blake 1998; Brand and Bhotika 2001].

When we fix all but one attribute of the multilinear model, thereby making $\mathbf{f}$ linear, Equation (7) turns into

$$\mathbf{Z}(s\mathbf{R}\mathbf{M}_{m,i}\mathbf{w}_m + \mathbf{t} - \mathbf{p_0}) = \mathbf{e}\,, \tag{8}$$

where m is the mode corresponding to the non-fixed attribute. $\mathbf{w}_m$ is a vector of weights for that attribute, and $\mathbf{M}_{m,i}$ is the corresponding linear basis for the tracked vertex i obtained from

$$\mathbf{M}_m = \mathscr{M} \times_2 \mathbf{w}_2^\top \cdots \times_{(m-1)} \mathbf{w}_{(m-1)}^\top \times_{(m+1)} \mathbf{w}_{(m+1)}^\top \cdots \times_N \mathbf{w}_N^\top. \quad (9)$$

To get a well constrained solution for the per-frame pose (scale, rotation, and translation) as well as the model's attribute parameters (expression, identity, etc.), we track a number of vertices and stack the resulting linear equations into one big system. For each pair of neighboring video frames we assemble a set of linear systems, each one applying Equation (8) to one of the tracked vertices. If the currently tracked attribute varies from frame to frame (such as expression does), we solve the set of linear systems and proceed to the next pair of neighboring frames. If, on the other hand, the attribute is constant across all frames (like identity), we accumulate the mentioned linear systems from each pair of frames and solve them together as one combined system. Solving the entire system for a pair of frames and a thousand tracked points completes in about a millisecond. Taking into account several levels of multi-scale and several passes through the whole video, the tracking process averages at one frame per second.

5.2 Initialization

The method described above, since it is based on tracking, needs to be initialized with the first frame alignment (pose and all the weights of the multilinear model). We accomplish this by specifying a small number of feature points which are then used to position the face geometry. The correspondences can be either user-provided (which gives more flexibility and power) or automatically detected (which avoids user intervention). We have experimented with the automatic feature detector developed by [Viola and Jones 2001], and found that it is robust and precise enough in locating a number of key features (eye corners, nose tip, mouth corners) to give a good approximating alignment in most cases. Imperfect alignment can be improved by tracking the first few frames back and forth until the model *snaps* into a better location. Other more powerful automated alignment approaches that take in account texture and lighting, such as the one described in [Blanz and Vetter 1999], could also be adapted to multilinear models.

6 Results

Multilinear models provide a convenient control of facial attributes. Figures 5 and 6 show example manipulations of our bilinear and trilinear models.

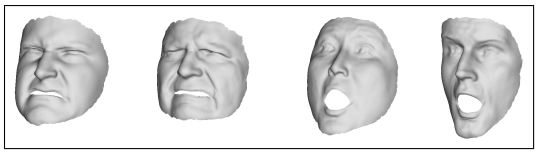


Figure 5: Several faces generated by manipulating the parameters of the bilinear model. The left two faces show our attempt of expressing disgust, an expression that was not in the database. They only differ in identity, to demonstrate the separability of our parameters. The right two faces show surprise for two novel identities, illustrating how the expression adjusts to identity.

Face Transfer infers the attribute parameters automatically by tracking the face in a video. Figure 7A shows a few example frames acquired by tracking a face outside of our data set. The resulting 3D shapes, shown below each frame, are generated by the bilinear model with the mouth shapes closed-off for texturing. Because

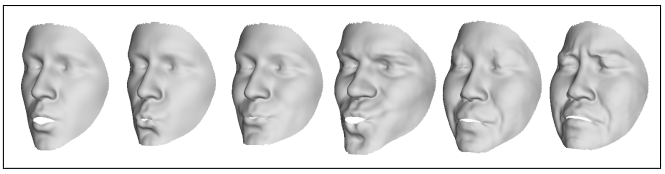


Figure 6: Faces generated by manipulating the parameters of the trilinear model. Left to right: producing the 'oo' sound, trying to whistle, breaking into a smile, two changes of identity, then adding a scowl.

our system tracks approximately a thousand vertices, the process is less sensitive to localized intensity changes (e.g., around the furrow above the lip). Once we obtain the 3D geometry, we can lift the texture from the video by assigning pixel colors to the corresponding mesh vertices. A simple performance-driven texture function can be obtained with the weighted sum of nearest neighbors.

All advantages of our system are combined in video rewrite applications, where a performance is lifted from a video, altered, and seamlessly rendered back into the video. In Figure 7B, we use the bilinear model to change a person's identity while retaining the expressions from the original performance. From left to right, the figure shows the original video, the recovered 3D shape, the modified 3D shape, and its textured overlay over the original video (without blending and with the simple texture model). Note how the style of smiling changes with identity. Figure 7C shows a post-production example, where a repeat performance is transferred onto the old footage. From left to right, we present the original video frame, a frame from a new video, the new geometry, and the final modified frame (without blending). Here, we combine the pose from the original video with the expression from the new video to modify original expressions.

Our most challenging face transfer uses the trilinear model to transfer expressions and visemes of a singing performance. In Figure 7D, the left two images show the frames from two input videos: a target video for a subject in our data set and a source video of a singing performance from a novel subject. The third image shows the final composite (along with the matching geometry as seen from two viewpoints) that combines the expressions and visemes from the source performance with the identity shown in the target video. For best visual results, we blend [Pérez et al. 2003] the face texture from the target video (constant throughout the sequence); the mouth texture from the source video; and the background texture, which includes the eyes peeking through the holes in the transferred geometry. In Figure 7E, we manually add a frown to the geometry in the final image; the frown texture was lifted from another frame of the same subject. With a similar technique, we can also combine facial attributes from several videos as shown in Figure 1, where the pose, expressions, and visemes are mixed from three different input videos.

7 Discussion

Perhaps our most remarkable empirical result is that even with a model estimated from a rather tiny data set, we can produce videorealistic results for new source and target subjects. Further improvements can be expected when we move to a wider variety of subjects and facial configurations.

We also see algorithmic opportunities to make aspects of the system more automatic and robust. Correspondence between scans might be improved with some of the methods shown in [Kraevoy and Sheffer 2004]. First-frame alignment would benefit from a successful application of a method such as [Blanz and Vetter 1999]. Optical flow-based tracking, which is inherently vulnerable

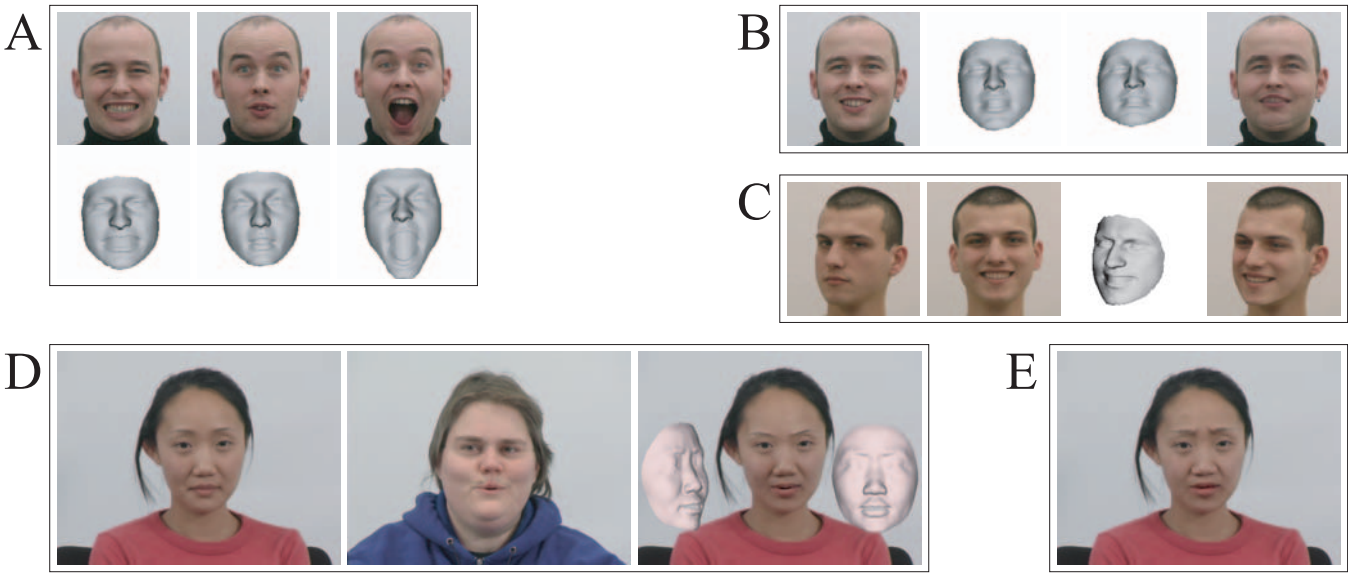


Figure 7: Several examples of our system: **(A)** A few frames of a bilinear-model tracking of a novel face and the corresponding 3D shapes below. **(B)** Changing the identity parameters of a performance tracked with the bilinear model. **(C)** Transferring a performance of a known identity from one video to another using the bilinear model. In each example the mouth gap was closed to allow for texturing. **(D)** Using the trilinear model to copy a singing performance from one video to another (left-to-right: original video, singing video, and the result). **(E)** Altering the performance from the previous example by adding a frown.

to imaging factors such as lighting, occlusions and specular reflections, can be made more robust with edge and corner constraints as demonstrated in [DeCarlo and Metaxas 1996].

In a production setting, the scan data would need to be expanded to contain shape and texture information for the ears, neck, and hair, so that we can make a larger range of head pose changes. Motions of eyes, eyelids, tongues, and teeth, are currently modeled in the texture function or not at all; this will either require more video data or a better solution. Finally, the texture function lifted from video is performance specific, in that we made no effort to remove variations due to lighting. Since we do estimate 3D shape, it may be possible to estimate and remove lighting, given sufficiently long videos.

8 Conclusion

To summarize, we have shown how to estimate a highly detailed face model from an incomplete set of face scans. The model is multilinear, and thus has the key property of separability: different attributes, such as identity and expression, can be manipulated independently. Thus we can change the identity and expression, but keep the smile. Even more useful, the new smile is in the style idiosyncratic to the new identity.

What makes this multilinear model a practical tool for animation is that we connect it directly to video, showing how to recover a time-series of poses and attribute parameters (expressions and visemes), plus a performance-driven texture function for an actor's face.

Our methods greatly simplify the editing of identity, performance, and facial texture in video, enabling video rewrite applications such as performance animation (puppetry) and actor replacement. In addition, the model offers a rich source of synthetic actors that can be controlled via video.

An intriguing prospect is that one could now build a multilinear model representing a vertex$\times$identity$\times$expression$\times$viseme$\times$*age* data tensor—without having to capture each individual's face at every stage of their life. The model would provide animators with control dials for each of the listed attributes, so they could change an actor's age along with their appearance and performance.

9 Acknowledgments

Funding for this work was provided by the MIT Oxygen Project. Our analysis was made possible by the many volunteers who agreed to have their faces scanned. The members of the MIT Graphics Group (in addition to their good looks) provided invaluable feedback throughout the entire project. Bob Sumner implemented the template-matching software used for correspondence. Ali Rahimi helped us formalize the probabilistic interpretation of our imputation algorithm. Ray Jones provided the bilateral filter code. Bryt Bradley provided a beautiful voice for our singing example. Tom Buehler shot and made all the videos.

References

ALLEN, B., CURLESS, B., AND POPOVIĆ, Z. 2003. The space of human body shapes: Reconstruction and parameterization from range scans. *ACM Transactions on Graphics 22*, 3 (July), 587–594.

BASCLE, B., AND BLAKE, A. 1998. Separability of pose and expression in facial tracking and animation. In *International Conference on Computer Vision (ICCV)*, 323–328.

BIRCHFIELD, S., 1996. KLT: An implementation of the kanade-lucas-tomasi feature tracker. http://www.ces.clemson.edu /∼stb/.

BLANZ, V., AND VETTER, T. 1999. A morphable model for the synthesis of 3D faces. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, 187–194.

BLANZ, V., BASSO, C., POGGIO, T., AND VETTER, T. 2003. Reanimating faces in images and video. *Computer Graphics Forum 22*, 3 (Sept.), 641–650.

BRAND, M. E., AND BHOTIKA, R. 2001. Flexible flow for 3D nonrigid tracking and shape recovery. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 315–322.

BRAND, M. E. 2002. Incremental singular value decomposition of uncertain data with missing values. In *European Conference on Computer Vision (ECCV)*, vol. 2350, 707–720.

BREGLER, C., COVELL, M., AND SLANEY, M. 1997. Video rewrite: Driving visual speech with audio. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, 353–360.

BREGLER, C., HERTZMANN, A., AND BIERMANN, H. 2000. Recovering non-rigid 3D shape from image streams. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 690–696.

CAO, Y., FALOUTSOS, P., AND PIGHIN, F. 2003. Unsupervised learning for speech motion editing. In *Eurographics/SIGGRAPH Symposium on Computer animation (SCA)*, 225–231.

CHAI, J.-X., XIAO, J., AND HODGINS, J. 2003. Vision-based control of 3D facial animation. In *Eurographics/SIGGRAPH Symposium on Computer Animation (SCA)*, 193–206.

CHUANG, E. S., DESHPANDE, H., AND BREGLER, C. 2002. Facial expression space learning. In *Pacific Conference on Computer Graphics and Applications (PG)*, 68–76.

DE LATHAUWER, L. 1997. *Signal Processing based on Multilinear Algebra*. PhD thesis, Katholieke Universiteit Leuven, Belgium.

DECARLO, D., AND METAXAS, D. 1996. The integration of optical flow and deformable models with applications to human face shape and motion estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 231–238.

DECARLO, D., AND METAXAS, D. 2000. Optical flow constraints on deformable models with applications to face tracking. *International Journal of Computer Vision 38*, 2, 99–127.

ESSA, I., BASU, S., DARRELL, T., AND PENTLAND, A. 1996. Modeling, tracking and interactive animation of faces and heads: Using input from video. In *Computer Animation '96*, 68–79.

EZZAT, T., AND POGGIO, T. 2000. Visual speech synthesis by morphing visemes. *International Journal of Computer Vision 38*, 1, 45–57.

FREEMAN, W. T., AND TENENBAUM, J. B. 1997. Learning bilinear models for two factor problems in vision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 554–560.

GEORGHIADES, A., BELHUMEUR, P., AND KRIEGMAN, D. 2001. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 23*, 6, 643–660.

GOTSMAN, C., GU, X., AND SHEFFER, A. 2003. Fundamentals of spherical parameterization for 3D meshes. *ACM Transactions on Graphics 22*, 3 (July), 358–363.

JONES, T. R., DURAND, F., AND DESBRUN, M. 2003. Non-iterative, feature-preserving mesh smoothing. *ACM Transactions on Graphics 22*, 3 (July), 943–949.

KOCH, R. M., GROSS, M. H., CARLS, F. R., VON BÜREN, D. F., FANKHAUSER, G., AND PARISH, Y. 1996. Simulating facial surgery using finite element methods. In *Proceedings of SIGGRAPH 96*, Computer Graphics Proceedings, Annual Conference Series, 421–428.

KRAEVOY, V., AND SHEFFER, A. 2004. Cross-parameterization and compatible remeshing of 3D models. *ACM Transactions on Graphics 23*, 3 (Aug.), 861–869.

KROONENBERG, P. M., AND DE LEEUW, J. 1980. Principal component analysis of three-mode data by means of alternating least squares algorithms. *Psychometrika 45*, 69–97.

LEE, Y., TERZOPOULOS, D., AND WATERS, K. 1995. Realistic modeling for facial animation. In *Proceedings of SIGGRAPH 95*, Computer Graphics Proceedings, Annual Conference Series, 55–62.

LI, H., ROIVAINEN, P., AND FORCHHEIMER, R. 1993. 3-D motion estimation in model-based facial image coding. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 15*, 6, 545–555.

NOH, J.-Y., AND NEUMANN, U. 2001. Expression cloning. In *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, 277–288.

PARKE, F. I. 1974. *A parametric model for human faces*. PhD thesis, University of Utah, Salt Lake City, Utah.

PARKE, F. I. 1982. Parameterized models for facial animation. *IEEE Computer Graphics & Applications 2* (Nov.), 61–68.

PENTLAND, A., AND SCLAROFF, S. 1991. Closed-form solutions for physically based shape modeling and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 13*, 7, 715–729.

PÉREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. *ACM Transactions on Graphics 22*, 3 (July), 313–318.

PIGHIN, F., HECKER, J., LISCHINSKI, D., SZELISKI, R., AND SALESIN, D. H. 1998. Synthesizing realistic facial expressions from photographs. In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, 75–84.

PIGHIN, F. H., SZELISKI, R., AND SALESIN, D. 1999. Resynthesizing facial animation through 3d model-based tracking. In *International Conference on Computer Vision (ICCV)*, 143–150.

PRAUN, E., AND HOPPE, H. 2003. Spherical parameterization and remeshing. *ACM Transactions on Graphics 22*, 3 (July), 340–349.

ROBERTSON, B. 2004. Locomotion. *Computer Graphics World* (Dec.).

ROWEIS, S. 1997. EM algorithms for PCA and SPCA. In *Advances in neural information processing systems 10 (NIPS)*, 626–632.

SIROVICH, L., AND KIRBY, M. 1987. Low dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America A 4*, 519–524.

SUMNER, R. W., AND POPOVIĆ, J. 2004. Deformation transfer for triangle meshes. *ACM Transactions on Graphics 23*, 3 (Aug.), 399–405.

TIPPING, M. E., AND BISHOP, C. M. 1999. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B 61*, 3, 611–622.

TORRESANI, L., YANG, D., ALEXANDER, E., AND BREGLER, C. 2001. Tracking and modeling non-rigid objects with rank constraints. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 493–450.

TUCKER, L. R. 1966. Some mathematical notes on three-mode factor analysis. *Psychometrika 31*, 3 (Sept.), 279–311.

VASILESCU, M. A. O., AND TERZOPOULOS, D. 2002. Multilinear analysis of image ensembles: Tensorfaces. In *European Conference on Computer Vision (ECCV)*, 447–460.

VASILESCU, M. A. O., AND TERZOPOULOS, D. 2004. Tensortextures: multilinear image-based rendering. *ACM Transactions on Graphics 23*, 3 (Aug.), 336–342.

VIOLA, P., AND JONES, M. 2001. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 511–518.

WANG, Y., HUANG, X., LEE, C.-S., ZHANG, S., LI, Z., SAMARAS, D., METAXAS, D., ELGAMMAL, A., AND HUANG, P. 2004. High resolution acquisition, learning and transfer of dynamic 3-d facial expressions. *Computer Graphics Forum 23*, 3 (Sept.), 677–686.

WATERS, K. 1987. A muscle model for animating three-dimensional facial expression. In *Computer Graphics (Proceedings of SIGGRAPH 87)*, vol. 21, 17–24.

WILLIAMS, L. 1990. Performance-driven facial animation. In *Computer Graphics (Proceedings of SIGGRAPH 90)*, vol. 24, 235–242.

ZHANG, L., SNAVELY, N., CURLESS, B., AND SEITZ, S. M. 2004. Spacetime faces: high resolution capture for modeling and animation. *ACM Transactions on Graphics 23*, 3 (Aug.), 548–558.

Performance-Driven Hand-Drawn Animation

Ian Buck[*] *Adam Finkelstein*[*] *Charles Jacobs*[‡] *Allison Klein*[*]
David H. Salesin[†‡] *Joshua Seims*[†] *Richard Szeliski*[‡] *Kentaro Toyama*[‡]

[*]Princeton University [†]University of Washington [‡]Microsoft Research

Abstract

We present a novel method for generating performance-driven, "hand-drawn" animation in real-time. Given an annotated set of hand-drawn faces for various expressions, our algorithm performs multi-way morphs to generate real-time animation that mimics the expressions of a user. Our system consists of a vision-based tracking component and a rendering component. Together, they form an animation system that can be used in a variety of applications, including teleconferencing, multi-user virtual worlds, compressed instructional videos, and consumer-oriented animation kits.

This paper describes our algorithms in detail and illustrates the potential for this work in a teleconferencing application. Experience with our implementation suggests that there are several advantages to our hand-drawn characters over other alternatives: (1) flexibility of animation style; (2) increased compression of expression information; and (3) masking of errors made by the face tracking system that are distracting in photorealistic animations.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—Display Algorithms; I.6.3 [Computer Graphics]: Methodology and Techniques—Interaction Techniques; I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Tracking

Keywords: Animation, Non-photorealistic rendering, Image morphing, Face tracking

1 Introduction

The proliferation of video cameras as standard PC peripherals expands the opportunities for synergy between computer vision and computer graphics. Many of the potential applications involve users driving graphical avatars using vision-based techniques that track facial movement. Standard video teleconferencing, for example, could be modified to display graphically generated faces instead of displaying the camera image as is. Anonymous chat rooms could be enhanced by avatars whose expressions are controlled by the participants in real time. Similarly, users could drive avatars in virtual worlds or gaming environments.

In this paper, we present an example of such a vision-driven application—a novel method for automatic animation of non-photorealistic (NPR) faces from example images. We assume we are given, as input, a set of drawings for a given character with various facial expressions, *e.g.*, 6 different mouths, 4 pairs of eyes, and 1 overall head (Figure 1). To perform one-time training of the system for a specific user, we take sample footage of the subject and manually establish correspondences between the hand-drawn elements and similar expressions on the subject's face, as shown in Figure 2. During execution, vision algorithms track the sender's expressions, which are distilled to a few parameters (we use ten 8-bit integers, or 80 bits, per frame) and sent to the renderer. Then, using various data interpolation and warping techniques, the renderer sythesizes the animated character from the appropriate pieces of artwork. Careful engineering of the components allows the system to run in real time on off-the-shelf PCs.

NPR animation of faces offers several advantages over attempts to work with photorealistic images.

First, because an illustrated character represents an abstraction of the real person, we as viewers do not expect a faithful replica of the speaker. Use of abstraction invites our imaginations to fill in the details [35, 54], as confirmed by our ability to watch hand-drawn cartoons without difficulty. Representational inaccuracies, lower frame rates, and lower temporal coherence—all of which might be unacceptable in realistic video—are perfectly acceptable with NPR animation.

Second, relaxing the constraints of realism allows us to significantly compress the information contained in an image. Our implementation, for example, requires only 10 parameters per frame to be transmitted from the face tracker to the renderer. These 10 parameters are already enough to generate convincing animations, but even increasing the number of parameters to 100 would not make significant demands on bandwidth for any network application. (The Facial Action Coding System, for example, includes only 68 parameters [1].) When combined with speech technology, such significant compression holds great promise for UI agents, instructional videos, and the like. For example, an immense amount of "talking head" video can be stored in the form of facial animation parameters plus plain ASCII text and synthesized into an NPR facial animation and voice track on the fly.

Another benefit of animated characters is that they allow the user to mask his or her true appearance, while permitting expressions and other visual cues to be perceived. This can have different value based on the application. In teleconferencing and MUD applications, it offers privacy. In a game engine, a player's facial expressions could be mapped onto the video game characters, thus enhancing the fantasy of playing the character.

Finally, an animated figure has an engaging quality that is often more fun than a live video clip. With the flexibility to render in many artistic styles and media, users can choose from a wider range of emotional contexts and appearances than with photorealistic images.

1.1 Related work

Graphical avatars driven by vision have a rich, varied history. We discuss some of this history by examining work with synthetic facial models, both 2D and 3D, photorealistic and otherwise.

In photorealistic 2D models, image blending or morphing is used to render face images [6, 18, 26, 32, 36, 55]. A typical example of this approach is a teleconferencing system [55] that stores a set of image samples of a person's face on both the transmitting and receiving computers. A face matching algorithm determines which faces among the stored samples look most like the input face, and a blend of these faces is displayed on the receiving side. The GeniMator system [52] also uses motion capture data to drive nonphotorealistic renderings, although the system is targeted more toward full-body rather than facial animation. Their system isn't targeted toward facial animation, however.

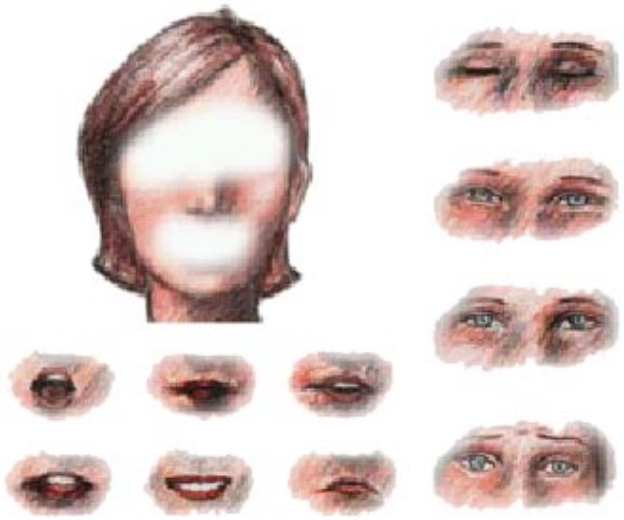


Figure 1 A full set of hand-drawn images used by our system.

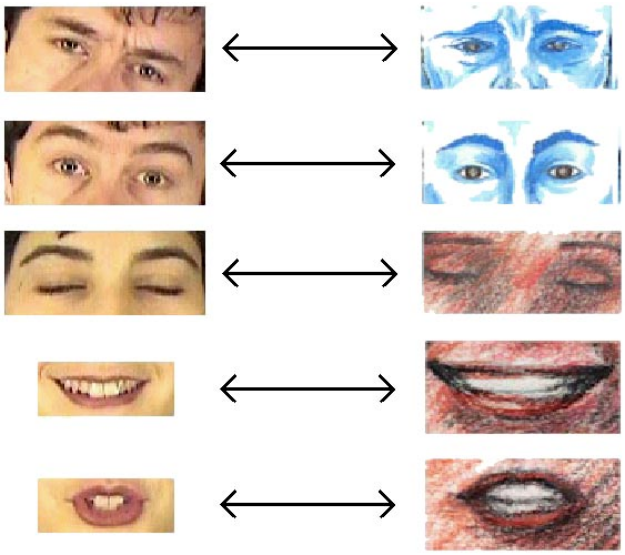


Figure 2 Sample correspondences between hand-drawn eyes and mouths and real-face equivalents. Two different users with two different corresponding sets of artwork are shown.

To drive 3D models, geometric model parameters must be recovered. In particular, facial features are tracked in the incoming images and then used to drive the movements of the 3D models, which are rendered with well-established graphics techniques [2, 12, 51].

Although facial features can be tracked by motion capture techniques to produce performance-driven animation systems [59, 42], systems requiring special markers or devices are unlikely to be adopted by the casual user. Using vision-based techniques, facial features can be tracked non-invasively. Trackers can be based on deformable patches [8, 14], edge or feature detectors [9, 26, 24, 43, 34, 53], and/or 3D models [3, 17, 23, 45]. Face tracking is currently an active area of research: robust, full-featured, real-time face tracking remains elusive. In this paper, we use a simple, color-based feature tracker (Section 2) that runs in real time.

On the rendering side, 3D facial animation is one of the most actively studied areas of computer graphics [5, 10, 11, 21, 27, 28, 30, 31, 41, 43, 44, 46, 48, 58, 61]. Generally, these techniques store a 3D mesh model of the face on which texture is overlaid. Movement of the mesh vertices, sometimes accompanied by texture changes, creates variations in expression. Truly photorealistic animation of faces, however, remains an unsolved challenge that we avoid altogether using NPR animations.

Non-photorealistic techniques come in several flavors. Haeberli [22] introduced the idea of using a reference image's pixel values to create interesting NPR effects. This idea has led to some beautiful imitations of artistic styles, such as pen and ink [50] and watercolor [15]. However, such algorithms are computationally expensive, limited to a particular artistic effect, and lack frame-to-frame coherence—a quality that is essential for animation. A totally different style of non-photorealistic rendering is to use cartoon characters, such as in Comic Chat [25]. This is most similar in spirit to our work, in that it uses combinations of hand-drawn artwork. However, we are rendering sequences of moving images, whereas the Comic Chat work concentrated on the layout of static scenes.

Inspired by work that shows how a wide range of faces (including variations in gender and age) can be synthesized by morphing among a small set of images [49], we rely on the Beier and Neely morph [4], as extended by Lee *et al.* [29] to blend multiple input images. Because we use morphing, we avoid several disadvantages of other NPR techniques: rendering performance is dependent only on the morphing process and independent of the artistic style of the drawings. Morphing can also avoid some frame-to-frame jitter caused by a stochastic rendering process. Lastly, morphing can be applied to drawing styles of any visual complexity.

Ultimately, our technique is akin to image-based rendering (IBR) approaches for non-photorealistic rendering applications. Litwinowicz and Williams [33] explored an image-based approach to NPR animation using rotoscoped lines over an image to warp it into new positions for each frame. Wood *et al.* [60] used hand-drawn artwork combined with a mobile camera to design moving background scenery for cel animation. Corrêa *et al.* [13] warped hand-drawn artwork wrapped over 3D models to attach complex textures to hand-drawn foreground characters in cel animation. Our work extends these ideas to 2D hand-drawn character animation, where we morph among many images to capture the full texture variation that can be seen across different facial expressions.

1.2 Approach

To construct a face, our system requires an initial set of hand-drawn images to blend together. An artist draws this set, divided into mouth, eyes, and background head images, all of which can be warped and blended independently. All drawings are annotated with morph control lines [4]. The control lines mark certain facial features, as illustrated in Figure 3, allowing the rendering process to morph artwork together without ghosting.

The hand-drawn images should span the range of visually distinguishable expressions and lip poses. In previous research on realistic facial rendering, only nineteen mouth images were found to be necessary for lip reading [39], and five eye images for a believable eye blink [16]. Since our goal is not to produce animations of the quality necessary for lip reading, we are able to use far fewer than this ideal number of images. Additionally, the morphing process generates many of the in-between images that would normally have to be drawn by hand. We found that just six mouth and four eye images are enough to get adequate results in a teleconferencing environment.

A training step requires manually associating each eye and mouth expression from the set of artwork with an equivalent expression from a video frame (Figure 2). This correspondence allows the system to

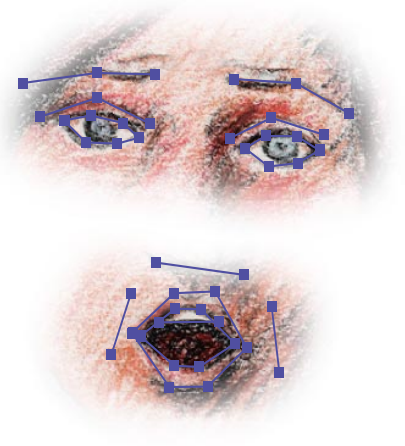


Figure 3 This image shows both eye and mouth regions with feathered masks. The purple lines are the control lines used to guide the morphing of these images.

discern which tracked measurements of the real person's face (as described in the next section) best match each hand-drawn image.

Training is required only once for a given user and set of artwork. After this initialization, a person's facial features are tracked in real time. For a teleconferencing application, these features are transmitted to a receiving computer. They are then used to compute a good blend of artwork to reconstruct a synthetic face.

As an additional feature, both our tracker and renderer are constructed to work with MPEG-4 Face Animation Parameters (FAPs) [1, 37, 38]. Therefore, our renderer could be used, with minimal changes, as a non-photorealistic renderer for MPEG-4 streams.

Section 2 describes our tracker in more detail. Section 3 describes our renderer. Section 4 describes some resulting animations. Section 5 concludes with some areas for future research.

2 Tracking

An ideal tracking system would accurately track all of the various deformations of the face in real-time, and allow us to pick a fixed set of parameters that would drive the renderer. Excellent face trackers exist (see Toyama [57] for a brief survey), but none are yet perfect. For the time being, we choose a passive, vision-based implementation that runs in real time and is non-intrusive (*i.e.*, it does not require the user to wear special devices, cosmetics, or markers).

Our particular implementation takes a frame of video and extracts ten scalar quantities, each encoded as an 8-bit integer:

- The x and y values of the midpoint of the line segment ℓ connecting the two pupils (2).
- The angle of ℓ with respect to the horizontal axis (1).
- The distance between the upper and lower eyelids of each eye (2).

Figure 4 A frame of video tracked by our system. The ten scalar quantities sent to the renderer are easily derived from the features detected by our tracker.

- The height of each eyebrow relative to the pupil (2).
- The distance between the left and right corners of the mouth (1).
- The height of the upper and lower lips, relative to the mouth center (2).

The first 3 scalars represent the head pose, while the middle 4 are used for eyes, and the final 3 for the mouth. A frame of video indicating the features tracked by our system is shown in Figure 4.

The tracking algorithms for all features rely on color information, which is relatively inexpensive to compute and somewhat resistant to illumination variations.

First, the tracker tags all pixels within a region of interest according to the output of a color-based pupil classifier. Then, for each tagged pixel, a correlation-based template match [19] is performed against a previously stored picture of the user's eye (if no pixels are tagged by the classifier, the immediate neighborhood of the previous frame's pupil location is used). If the highest score from the correlation matcher falls below an empirically determined threshold, the eye is assumed to be closed, in which case we use the previous pupil location.

Once the pupils are found, we search for the eyebrows. For all points above the pupil (within a certain range) we perform a 1D template match [9] against a stored cross section of the eyebrow, and choose the location with the highest correlation.

To find the mouth, the tracker first tags each pixel within a selected region according to a lip color classifier. Next, simple image-processing operations are applied to this set of tagged pixels to remove noise and eliminate stray pixels, and then the largest 4-connected blob is found. Finally, a many-sided polygon is fitted to

this blob, using a technique similar to Toyama's radial-spanning blob technique [56]. The mouth measurements are taken from this fitting polygon.

Once the pupils, eyebrows, and mouth have been detected, most of the 10 scalar values are easily determined. The distance between the upper and lower eyelids is computed simply by searching for the first pixel above and below the pupil that has a luminance below some threshold.

The performance of the tracker is reasonable for driving the renderer in real time. There are some remaining problems, however. The system requires a manual, per-user initialization to determine eye, eyebrow, and lip colors. The tracker is limited to a range between 0.5 and 1.5 meters from the camera, and to an approximately 30 degree rotation from an upright, frontally oriented face (about all 3 axes). The tracker assumes reasonable, fixed illumination. From time to time, it mistracks, requiring a combination of user movement and manual reinitialization to correct. And finally, some subjects do not exhibit sufficient color contrast between skin and lip color for the tracker to work. All of these problems need to be handled for a more robust system, but many of these issues remain open problems in the vision-based face tracking community. We anticipate that future research will alleviate these difficulties. For the time being, our tracker is sufficient to drive the renderer in a stably illuminated office environment.

3 Rendering

The rendering stage runs after the tracking stage (possibly on a separate machine, depending on the application). It takes the 10 tracked parameters as input and renders a synthetic animated character that mimics the user's expressions. The renderer can be divided into two components: *expression mapping*, which determines which pieces of artwork should be used in creating the final animated character for a given frame, and in what proportions; and *warping*, which combines the various pieces of artwork together using feathered masks.

3.1 Expression mapping

The problem of determining the best blend of artwork needed to mimic a given expression is really one of scattered data interpolation [40]. For now, let us consider how expression mapping is done for the mouth; the eyes are handled similarly.

Suppose we have n pieces of artwork for the mouth in different expressions $M_1, \ldots, M_n$. The training data provides an association between each mouth expression M_i and a

k-dimensional *training point* $\boldsymbol{m}_i$, whose components are the values of the k tracked parameters associated with that mouth. (Recall that for the sample art set shown in Figure 1, $n = 6$, and in our implementation, $k = 3$ for the mouths and $k = 4$ for the eyes.)

Our problem is: Given some new set $\boldsymbol{m}$ of tracked parameters for the mouth, find a set of weights $\alpha_1, \ldots, \alpha_n$ such that $\sum_i \alpha_i = 1$ and $\left\| \boldsymbol{m} - \sum_i \alpha_i \boldsymbol{m}_i \right\|$ is minimized (or at least small). We can then use these weights α_i to create the new expression by morphing together an appropriately weighted combination of the original artwork, as described in Section 3.2.

Our solution to this scattered data interpolation problem must be fast, yet accurate enough to faithfully reproduce the speaker's expression. Furthermore, for our animation to be smooth, two points $\boldsymbol{m}$ and $\boldsymbol{m}'$ that lie close to each other should produce expressions that are similar. On the other hand, there is a tradeoff between accuracy and visual clarity: the more pieces of artwork we blend together, the blurrier the imagery in the resulting animation.

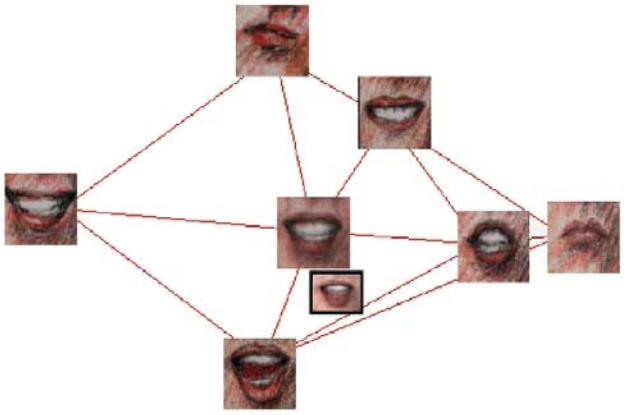


Figure 5 This image shows a sample Delaunay triangulation for mouth interpolations. Note the interpolated mouth inside one of the triangles.

A straightforward approach to this problem is to compute a k-dimensional Delaunay triangulation [20] among the training points $\boldsymbol{m}_i$. Then, for a given new point $\boldsymbol{m}$, locate that point in the triangulation and use the barycentric coordinates of the simplex it lies in as the morphing weights. These weights could then be applied to the drawings corresponding to the vertices of that simplex, as in the polymorph method described by Lee *et al.* [29]. This approach has the advantage that it provides smooth transitions between nearby expressions. However, if the number of tracked parameters k is large, the resulting morph may be blurry.

Our solution is to use the same Delaunay triangulation approach, but in a lower-dimensional space. We use a principal component analysis (PCA) [7] to choose the j largest eigenvectors that span the k-dimensional space created by the training points. We project the training set into this j-dimensional space, as well as the query point $\boldsymbol{m}$. In practice, we use $j = 2$, which appears to provide a good tradeoff between expression accuracy and image clarity. Thus, we merely locate the projection of $\boldsymbol{m}$ in its 2-D triangulation (Figure 5) and use the barycentric coordinates of the triangle vertices as morph weights for the three corresponding drawings (Figure 6). Projected points $\boldsymbol{m}$ falling outside of any Delaunay triangle are mapped to a point on the convex hull of the training points and associated with the Delaunay triangle that abuts that region of the convex hull. In our experience, if the initial correspondences are properly set up, most points projecting outside of the convex hull lie close enough to the hull that this method works well.

In our implementation, we find the PCA of the training set by doing a singular value decomposition of the matrix of training points [47]. The two dominant eigenvectors, as well as the resulting Delaunay triangulation, are saved to map feature points at runtime.

3.2 Morphing

To draw the face, we first render the warped versions of the eye and mouth regions of the face. Next, the head image, which contains "soft" alpha values in the eye and mouth areas to provide feathered masking, is placed on top of the rendered eyes and mouth. More than one such head image can be loaded, and the program will cycle through them at each frame. For the animations in the style of Bill Plympton, we use two different overall heads to achieve a shimmering quality. Finally, we apply a translation and rotation to the image in order to get the head tilt.

We describe here the process of creating a single, new mouth based on an expression mapping. (To create the eyes, we use an identical procedure.) Recall that the morphing module receives weights cor-

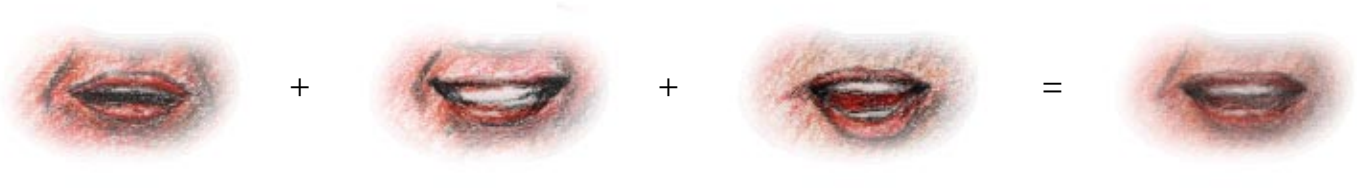


Figure 6 The three mouths on the left are warped and then blended to make the mouth on the right.

responding to the barycentric coordinates of the projected tracked parameters in a triangle whose corners correspond to three original mouth drawings in the training set. To create a new mouth, we use three-way Beier-Neely morphing [4], as generalized by Lee *et al.* [29] for *polymorphs*—morphs between more than two source images.

Consider the triangle formed by the three mouths. To create a new intermediate mouth, we first warp the mouths at the corners, yielding three mouths whose features align. Next we composite the three warped mouths using alpha blending to render the new mouth image. The blending weights are given by the aforementioned barycentric coordinates.

We employ two strategies for accelerating the morph.

First, we sample the warps over the vertices of a 30×30 quadmesh, represented as triangle strips, and use texture mapping hardware to render triangles rather than computing the warp at every pixel. Since many common PC graphics boards now come with texture mapping and alpha blending acceleration, we use this hardware to our advantage.

Second, the actual 3-way warp function is not evaluated at each mesh vertex. Instead, we approximate the correct warp function by summing together two 2-way warps. Consider the 3-way warp function $W_{ABC}(x, y, \alpha_B, \alpha_C)$, which returns a vector indicating how pixel (x, y) in image A moves when warping it toward image B by a fraction α_B and toward image C by a fraction α_C. Now, consider the 2-way warp $W_{AB}(x, y, \alpha)$, returning a vector indicating where the pixel at location (x, y) in image A moves when warping it α of the way toward image B. Our approximation of the 3-way warp is $W_{ABC}(x, y, \alpha_B, \alpha_C) \approx W_{AB}(x, y, \alpha_B) + W_{AC}(x, y, \alpha_C)$. The warp weights α_B and α_C used are simply the barycentric coordinates corresponding to the points A and B as given above. To make evaluation of the approximate warps fast, we precompute the 2-way warps at a number of discrete values of α and interpolate between these stored functions.

A more accurate method would be to sample the actual warp function at various points inside the triangle and interpolate these precomputed functions, but this would require sampling a 2-D function rather than a 1D function. In our work, we did not notice much difference between our approximation and the real warp, so we did not explore this method.

4 Implementation and observations

We implemented our algorithms on a 450MHz Pentium III processor, equipped with a high-end PC graphics card. Table 1 shows average running times rounded to the nearest millisecond for the various stages in our process. The slowest component is the facial feature tracking. Nonetheless, we can track and render simultaneously on the same computer at 25 fps. Since our video capture board can only grab frames at 15 fps, this leaves CPU cycles to spare.

Because the tracker produces 10 8-bit integers per frame, our current bandwidth requirements are 2400 baud for 30 frames per second. At 10 fps, the bandwidth requirement drops to a miserly 800 baud. Taking advantage of temporal coherence in the tracked features is likely to yield even greater compression.

Stage	*Time (ms)*
Copy video frame from camera	3
Track facial features	29
Project into feature space	< 0.1
Render mouths, eyes, and head	8
Total	40

Table 1 Running times (450MHz Pentium III) for various stages of the pipeline.

Five different styles of imagery were used to demonstrate the flexibility of the rendering scheme. Of these, four were generated from hand-drawn artwork in the manner described in Section 3 (*Monster*, *Blue*, *Wavy*, and *Straight*). These are illustrated in Figure 7. The last style (*Photoreal*), which appears on the video only, was created from actual video of one of the subjects herself based on images acquired prior to run time.

We tried rendering at 10, 15, and 30 frames per second. The animation at 30 fps was done off-line, using a previously digitized video stream, and rendered according to the methods described in Section 3. The animations at lower speeds take averages of tracked parameters collected at 30Hz over 3 (or 2) frames and render new images at 10 (or 15) fps.

Sample frames from the conversations can be seen in Figure 7. The characters shown exhibit a variety of expressions and a variety of mouth, eye, and head poses.

After using the system, we have made the following observations:

- While the output is engaging, the rendering does not achieve the quality of hand-drawn animation. This is not surprising, since our frames are generated automatically, without run-time input from a professional animator. It suggests that our technique in its current form is best suited for applications in which professional quality animation is not the goal.
- The animations rendered at 30 fps appeared jittery and anxious, whereas animations rendered at 10 and 15 fps reduced the visible jitter considerably. While this is partly due to noise in the tracking process, it may also be that the quality of animations are sometimes improved at a lower frame rate [54].
- In spite of the jitter, the 30 fps animation reproduces visual speech articulations more clearly, undoubtedly because of the high speed of mouth movement during speech.
- Apparent eye-contact is made with all of the characters. This is an advantage over standard video teleconferencing in which lack of eye contact is often cited as a major drawback.
- The four hand-drawn animations appear more compelling than the *Photoreal* style.

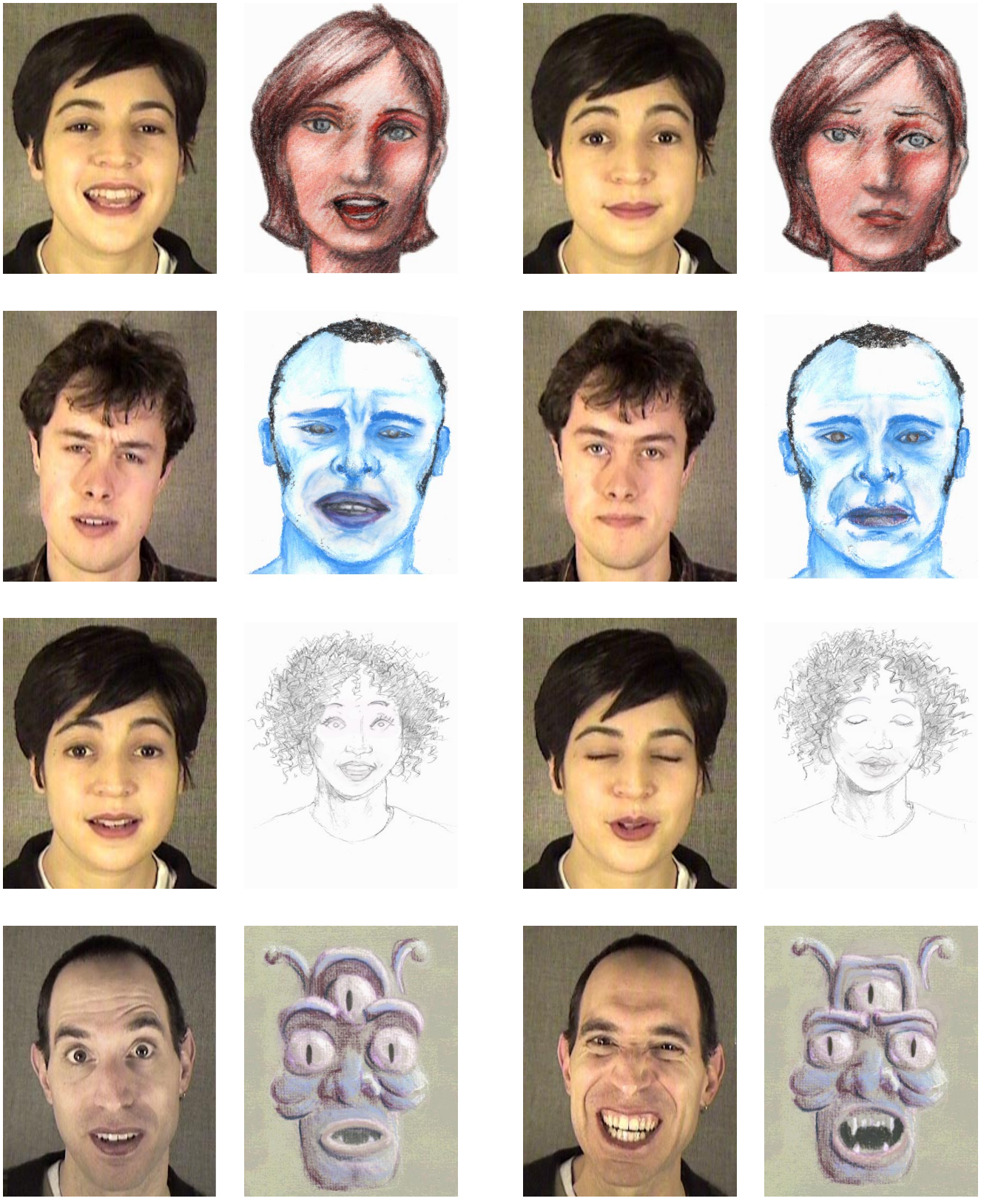


Figure 7 Snapshots of our system, shown as pairs of matching video and animated frames. Shown are a variety of artistic styles – from top to bottom, *Straight*, *Blue*, *Wavy*, and *Monster*. The system conveys the mouths and eyes in many configurations, as well as a variety of expressions such as happy, neutral, surprised, and aggressive.

This last point is interesting. Although the *Photoreal* animation more faithfully mimics a real human face, the hand-drawn animations are nevertheless perceived as more compelling. In particular, the mouth movements in the *Photoreal* style appear erratic and affected. Frequently, artifacts in the morphing process become apparent as one mouth appears to dissolve, rather than morph, into another. In contrast, although the hand-drawn animation exhibits the same technical problems, it appears more natural. We hypothesize that as observers, we are more forgiving of cartoon characters, whose abstraction and imperfection we readily accept; on the other hand, we expect absolute fidelity of photorealistic video and notice even minor departures from reality.

Finally, we discovered that different observers liked different animation styles. This highlights the flexibility of our algorithm for animation—a library of various hand-drawn faces (requiring only a few drawings per face) would allow users to pick and choose the style according to their preferences.

5 Discussion and future work

We have demonstrated how a small set of hand-drawn artwork, in conjunction with a small amount of facial tracking data, can be used to create a real-time performance-driven animation system in which animations effectively mimic the expressions and facial actions of a human speaker. Our system works in real time using a combination of a fast feature tracker and a fast novel morphing technique that paints the appropriate eyes and mouth onto a head. One component of this work is a novel face expression interpolation algorithm that projects tracking data onto a two-dimensional subspace, and then uses a Delaunay triangulation to find the three nearest expressions and to compute their blending weights.

Our system is one of the first to apply image-based rendering techniques to a collection of hand-drawn artwork to produce facial animations. Our framework has several advantages. It can accommodate a variety of artistic styles and media, limited only by the possible styles of the input artwork. It accommodates a variety of animation styles, *e.g.*, different frame rates and different amounts of flicker or blending. It is able to compress facial expression information to a handful of integers per frame. Finally, by relying on hand-drawn animations, it avoids the primary difficulty with photorealistic avatars, namely, that the rendered animations appear unnatural in some way.

There are several ways in which we plan to improve and extend our existing system. First, we plan to add the capability to both track and render additional parameters. Changes in head pose are essential for transmitting gestures such as nodding and shaking of the head. Facial creases and wrinkles will add to the expressivity of the renderings. It would also be interesting to explore using different animation styles at run-time based on the expressive content of the frame being rendered. For example, when the speaker has highly raised eyebrows indicating an extreme emotional state, random jitter and reddish shifts in color might be introduced to the rendering process to convey an additional intensity.

By shifting the focus of image-based rendering away from photorealistic reproduction towards the goal of expressive animation, we have opened up a wide range of new expressive possibilities. For example, we could use avatars without any faces *per se*, *e.g.*, a scene in which weather reflects the speaker's expression. A cloudless sunny sky could correspond to a smile, while a dark overcast or stormy sky could reflect a frown.

We believe that the combination of real-time feature tracking, performance-driven animation, and non-photorealistic rendering can form the basis for a range of exciting applications. We imagine teleconferencing applications and multi-user virtual worlds in which users can put on graphical masks that transmit their expressions without revealing identity. Another possibility is for video games, in which players could puppeteer the expressions that appear on their characters. A final application might be home animation kits, where children could shoot, edit, and replay animations of their favorite cartoon characters, driven by their own faces in real time.

References

[1] G. A. Abrantes and F. Pereira. MPEG-4 facial animation technology: Survey, implementation, and results. In *IEEE Transactions on Circuits and Systems for Video Technology*, pages 290–305, 1999.

[2] Paul M. Antoszczyszyn, John M. Hannah, and Peter M. Grant. Accurate automatic frame fitting for semantic-based moving image coding using a facial code-book. In *International Conference on Image Processing*, volume 1, pages 689–692, 1996.

[3] S. Basu, N. Oliver, and A. Pentland. 3D modeling and tracking of human lip motions. In *Proc. Int'l Conf. on Computer Vision*, pages 337–343, 1998.

[4] Thaddeus Beier and Shawn Neely. Feature based image metamorphosis. In *SIGGRAPH 92 Conference Proceedings*, pages 35–42. ACM SIGGRAPH, Addison Wesley, August 1992.

[5] Philippe Bergeron and Pierre Lachapelle. Controlling facial expressions and body movements in the computer-generated animated short "Tony De Peltrie". In *SIGGRAPH 85 Advanced Computer Animation seminar notes*. ACM SIGGRAPH, July 1985.

[6] D. Beymer, A. Shashua, and T. Poggio. Example based image analysis and synthesis. A. I. Memo 1431, Massachusetts Institute of Technology, November 1993.

[7] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.

[8] Michael Black and Yaser Yacoob. Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motion. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 374–381, 1995.

[9] Andrew Blake and Michael Isard. *Active Contours: The Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion*. Springer Verlag, 1998.

[10] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *SIGGRAPH 99 Conference Proceedings*, pages 187–194. ACM SIGGRAPH, 1999.

[11] Matthew Brand. Voice puppetry. In *SIGGRAPH 99 Conference Proceedings*, pages 21–28. ACM SIGGRAPH, 1999.

[12] Chang S. Choi, Kiyoharu, Hiroshi Harashima, and Tsuyoshi Takebe. Analysis and synthesis of facial image sequences in model-based image coding. In *IEEE Transactions on Circuits and Systems for Video Technology*, volume 4, pages 257–275, June 1994.

[13] Wagner Toledo Corrêa, Robert J. Jensen, Craig E. Thayer, and Adam Finkelstein. Texture mapping for cel animation. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 435–446. ACM SIGGRAPH, Addison Wesley, July 1998.

[14] Michele Covell. Eigen-points: control-point location using principal component analyses. In *Proc. IEEE International Conference on Automatic Face and Gesture Recognition*, pages 122–127, October 1996.

[15] Cassidy Curtis, Sean Anderson, Joshua Seims, Kurt Fleischer, and David Salesin. Computer-generated watercolor. In *SIGGRAPH 97 Conference Proceedings*, pages 421–430. ACM SIGGRAPH, Addison Wesley, August 1997.

[16] Neil D. Duffy. Animation using image samples. In *Processing Images of Faces*, pages 179–201. Ablex Publishing Corp., Norwood, NJ, 1992.

[17] I. Essa, S. Basu, T. Darrell, and A. Pentland. Modeling, tracking and interactive animation of faces and heads using input from video. In *Computer Animation Conference*, pages 68–79, June 1996.

[18] T. Ezzat and T. Poggio. Facial analysis and synthesis using image-based models. In *Proceedings of the Second International Conference on Automatic Faces and Gesture Recognition*, pages 116–121, 1996.

[19] William T. Freeman, David B. Anderson, Paul A. Beardsley, Chris N. Dodge, Michal Roth, Craig D. Weissman, William S. Yerazunis, Hiroshi Kage, Kazuo Kyuma, Yasunari Miyake, and Ken ichi Tanaka. Computer vision for interactive computer graphics. *IEEE Computer Graphics and Applications*, May/June:42–53, 1998.

[20] Jacob E. Goodman and Joseph O'Rourke. *Handbook of Discrete and Computational Geometry*. CRC Press, New York, 1997.

[21] Brian Guenter, Cindy Grimm, Daniel Wood, Henrique Malvar, and Frédéric Pighin. Making faces. In *SIGGRAPH 98 Conference Proceedings*, pages 55–66. ACM SIGGRAPH, July 1998.

[22] Paul E. Haeberli. Paint by numbers: Abstract image representations. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 207–214, August 1990.

[23] T. S. Jebara and A. Pentland. Parametrized structure from motion for 3D adaptive feedback tracking of faces. In *Proc. Computer Vision and Patt. Recog.*, 1996.

[24] M. Kass, A. Witkin, and D. Terzopoulos. Snakes, active contour models. In *First International Conference on Computer Vision*, pages 259–268, 1987.

[25] David Kurlander, Tim Skelly, and David Salesin. Comic chat. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 225–236. ACM SIGGRAPH, Addison Wesley, August 1996.

[26] A. Lanitis, C.J. Taylor, and T.F. Cootes. A unified approach to coding and interpretting faces. In *Proceedings of 5th International Conference on Computer Vision*, pages 368–373, 1995.

[27] F. Lavagetto, I.S. Pandzic, F. Kalra, and N Magnenat-Thalmann. Synthetic and hybrid imaging in the HUMANOID and VIDAS projects. In *International Conference on Image Processing*, volume 3, pages 663–666, 1996.

[28] B. Le Goff, T. Guard-Marigny, M. Cohen, and C. Benoit. Real-time analysis-synthesis and intelligibility of talking faces. In *2nd International conference on Speech Synthesis*, September 1994.

[29] Seungyong Lee, George Wolberg, and Sung Yong Shin. Polymorph: Morphing among multiple images. In *IEEE Computer Graphics and Applications*, pages 58–71, January 1998.

[30] Yuencheng Lee, Demetri Terzopoulos, and Keith Waters. Realistic modeling for facial animation. In *SIGGRAPH 95 Conference Proceedings*, pages 55–62. ACM SIGGRAPH, Addison Wesley, August 1995.

[31] H. Li, P. Roivainen, and R. Forchheimer. 3-D motion estimation in model-based facial image coding. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 545–555, 1993.

[32] Stephen E. Librande. Example-based character drawing. Master's thesis, Massachusetts Institute of Technology, August 1992.

[33] Peter Litwinowicz and Lance Williams. Animating images with drawings. In *SIGGRAPH 94 Conference Proceedings*, pages 409–412. ACM SIGGRAPH, Addison Wesley, August 1994.

[34] Katsuhiro Matsuno, Chil-Woo Lee, Satoshi Kimura, and Saburo Tsuji. Automatic recognition of human facial expressions. In *Proceedings of the IEEE*, pages 352–359, 1995.

[35] S. McCloud. *Understanding Comics*. Kitcken Sink Press, 1993.

[36] Baback Moghaddam and Alex Pentland. An automatic system for model-based coding of faces. In *IEEE Data Compression Conference*, March 1995.

[37] Committee draft of ISO/IEC 14496-2, information technology – coding of audio-visual objects: Video. Annex C contains Face object decoding tables and definitions, 1996.

[38] Coding of moving pictures and audio. ISO/IEC JTC1/SC29/WG11 N2459, International Organisation for Standardisation, October 1998. http://www.cselt.stet.it/mpeg/standards/mpeg-4/mpeg-4.htm.

[39] Ware Myers. Graphics aid the deaf. *IEEE Computer Graphics and Applications*, 2(2):100–102, March 1982.

[40] Gregory Nielson. Scattered data modeling. In *IEEE Computer Graphics and Applications*, pages 60–70, 1993.

[41] Frederic I. Parke and Keith Waters. *Computer Facial Animation*. A K Peters, Wellesley, Massachusetts, 1996.

[42] Elizabeth C. Patterson, Peter C. Litwinowicz, and Ned Greene. Facial animation by spatial mapping. In Nadia Magnenat Thalmann and Daniel Thalmann, editors, *Computer Animation 91*, pages 31–44. Springer-Verlag, Tokyo, 1991.

[43] E. Petajan and H. P. Graf. Robust face feature analysis for automatic speechreading and character animation. In *Proc. Int'l Conf. on Autom. Face and Gesture Recog.*, pages 357–362, 1996.

[44] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D. Salesin. Synthesizing realistic facial expressions from photographs. In *SIGGRAPH 98 Conference Proceedings*. ACM SIGGRAPH, 1998.

[45] Frédéric Pighin, Richard Szeliski, and David H. Salesin. Resynthesizing facial animation through 3D model-based tracking. In *Seventh IEEE International Conference on Computer Vision (ICCV '99)*, pages 143–150, 1999.

[46] S. M. Platt. Animating facial expressions. In *Computer Graphics (SIGGRAPH '81 Proceedings)*, volume 15, pages 245–252, August 1981.

[47] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing (2nd ed.)*. Cambridge University Press, Cambridge, 1992.

[48] Rao, Chen, Mersereau, and Anderson. Towards a real-time model-based-coding system. In *Proc. Workshop on Image and Multidimensional Signal Processing*, March 1996.

[49] Duncan Rowland and David Perrett. Manipulating facial appearance through shape and color. *IEEE Computer Graphics and Applications*, September:70–76, 1995.

[50] Michael Salisbury, Michael Wong, John Hughes, and David H. Salesin. Orientable textures for image-based pen-and-ink illustration. In *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 401–406. ACM SIGGRAPH, Addison Wesley, August 1997.

[51] A. Saulnier, M.-L. Viaud, and D. Geldreich. Real-time facial analysis and synthesis chain. In *Proc. Int'l Conf. on Autom. Face and Gesture Recog.*, pages 86–91, 1995.

[52] Orjan Standberg. Genimator: Applies human motion onto linedrawn cartoon characters. URL: http://home5.swipnet.se/~w-56588/GeniMator.htm.

[53] D. Terzopoulos and K. Waters. Analysis and synthesis of facial image sequences using physical and anatomical models. In *IEEE Trans. Pattern Analysis and Machine Intelligence*, pages 569–579, June 1993.

[54] Frank Thomas and Ollie Johnston. *The Illusion of Life: Disney Animation*. Hyperion Press, 1981.

[55] Sebastian Toelg and Tomaso Poggio. Towards an example-based image compression architecture for video-conferencing. In *AI Memo 1494, CBCL Paper 100*, June 1994.

[56] K. Toyama. Radial spanning for fast blob detection. In *Joint Conference on Information Sciences Proceedings*, volume 4, pages 484–487, Research Triangle Park, NC, 1998.

[57] Kentaro Toyama. Prolegomena for robust face tracking. Technical Report MSR-TR-98-65, Microsoft Research, November 1998.

[58] Keith Waters. A muscle model for animating three-dimensional facial expression. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pages 17–24, July 1987.

[59] Lance Williams. Performance-driven facial animation. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 235–242, August 1990.

[60] Daniel N. Wood, Adam Finkelstein, John F. Hughes, Craig E. Thayer, and David H. Salesin. Multiperspective panoramas for cel animation. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 243–250. ACM SIGGRAPH, Addison Wesley, August 1997.

[61] Hsi-Jung Wu et al. Tracking subspace representations of face images. In *International Conference on Image Processing*, volume 5, pages 389–392, April 1994.