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.

NPAR 2000 Annecy France

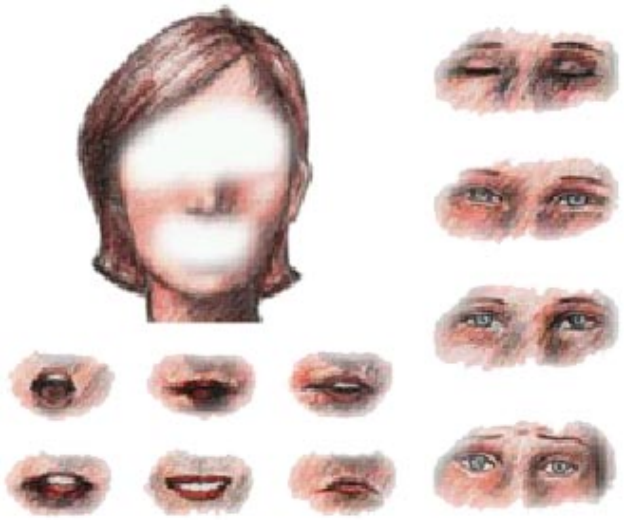

Figure 1 A full set of hand-drawn images used by our system.

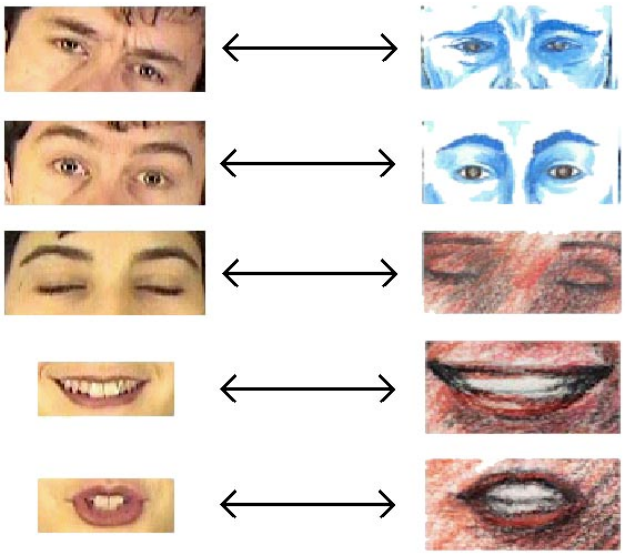

Figure 2 Sample correspondences between hand-drawn eyes and mouths and real-face equivalents. Two different users with two different corresponding sets of artwork are shown.

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

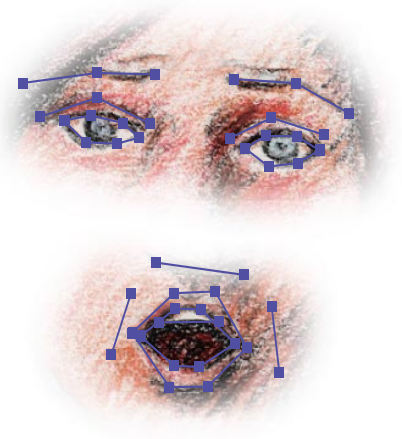

Figure 3 This image shows both eye and mouth regions with feathered masks. The purple lines are the control lines used to guide the morphing of these images.

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.

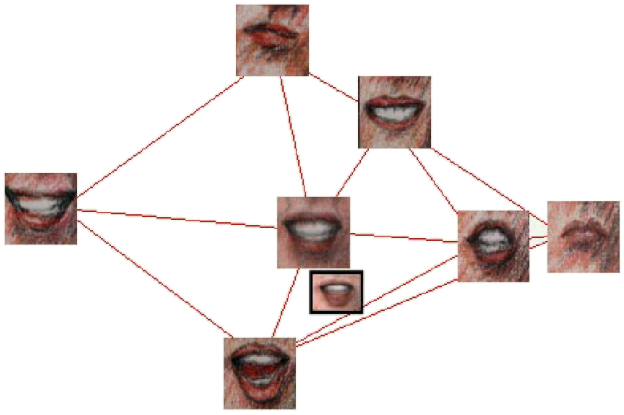

Figure 5 This image shows a sample Delaunay triangulation for mouth interpolations. Note the interpolated mouth inside one of the triangles.

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-

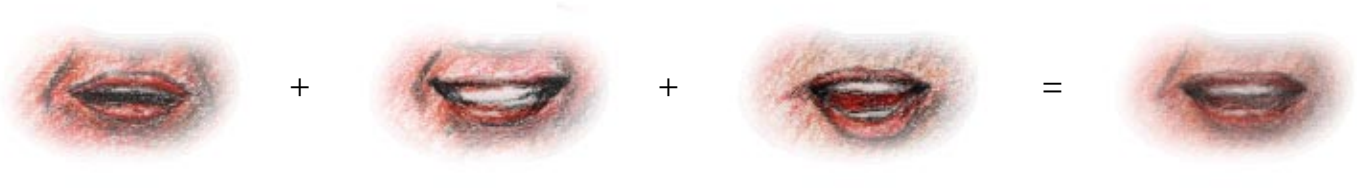

Figure 6 The three mouths on the left are warped and then blended to make the mouth on the right.

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.

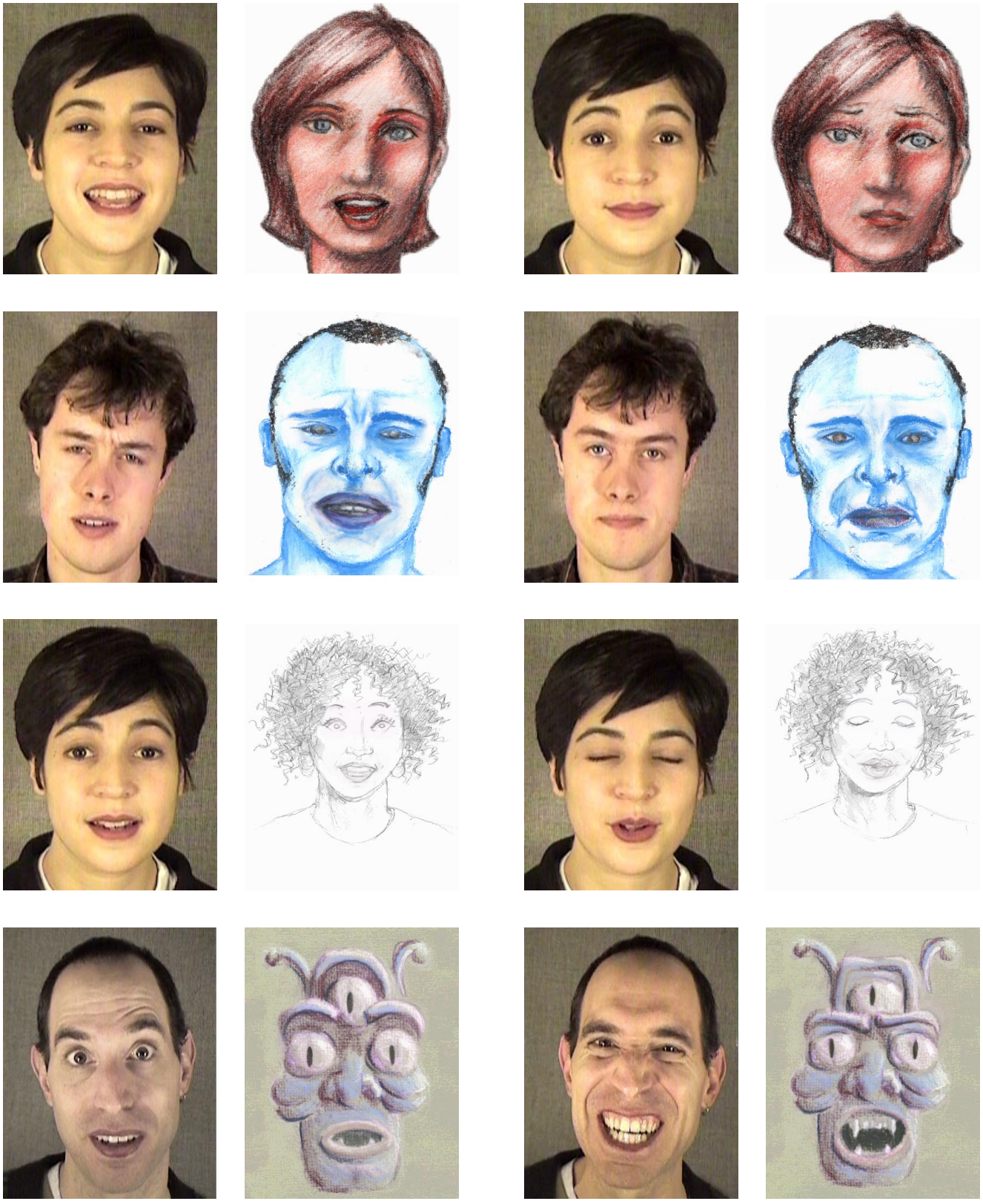

Figure 7 Snapshots of our system, shown as pairs of matching video and animated frames. Shown are a variety of artistic styles – from top to bottom, *Straight*, *Blue*, *Wavy*, and *Monster*. The system conveys the mouths and eyes in many configurations, as well as a variety of expressions such as happy, 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.