# Robust and Rapid Generation of Animated Faces From Video Images: A Model-Based Modeling Approach

Zhengyou Zhang, Zicheng Liu, Dennis Adler, Michael F. Cohen, Erik Hanson, Ying Shan[1]

http://research.microsoft.com/~zhang/
E-mail:{zhang,zliu}@microsoft.com

Technical Report
MSR-TR-2001-101

We have developed an easy-to-use and cost-effective system to construct textured 3D animated face models from videos with minimal user interaction. This is a particularly challenging task for faces due to a lack of prominent textures. We develop a robust system by following a model-based approach: we make full use of generic knowledge of faces in head motion determination, head tracking, model fitting, and multiple-view bundle adjustment. Our system first takes, with an ordinary video camera, images of a face of a person sitting in front of the camera turning their head from one side to the other. After five manual clicks on two images to indicate the position of the eye corners, nose tip and mouth corners, the system automatically generates a realistic looking 3D human head model that can be animated immediately (different poses, facial expressions and talking). A user, with a PC and a video camera, can use our system to generate his/her face model in a few minutes. The face model can then be imported in his/her favorite game, and the user sees themselves and their friends take part in the game they are playing. We have demonstrated the system on a laptop computer *live* at many events, and constructed face models for hundreds of people. It works robustly under various environment settings.

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
http://www.research.microsoft.com

[1]Current address: Sarnoff Corporation, New Jersey, USA

# Contents

# 1 Introduction

Animated face models are essential to computer games, filmmaking, online chat, virtual presence, video conferencing, etc. Generating realistic 3D human face models and facial animations has been a persistent challenge in computer vision and graphics. So far, the most popular commercially available tools have utilized laser scanners or structured lights. Not only are these scanners expensive, the data are usually quite noisy, requiring hand touchup and manual registration prior to animating the model. Because inexpensive computers and cameras are becoming ubiquitous, there is great interest in generating face models directly from video images. In spite of progress toward this goal, the available techniques are either manually intensive or computationally expensive.

The goal of our work is to develop an easy-to-use and cost-effective system that constructs textured 3D animated face models from videos with minimal user interaction in no more than a few minutes. The user interface is very simple. First, a video sequence is captured while the user turns his/her head from one side to the other side in about 5 seconds. Second, the user browses the video to select a frame with frontal face. That frame and its neighboring one (called two base images) pop up, and the user marks 5 feature points (eye corners, nose top, and mouth corners) on each of them. Optionally, the user marks 3 points under the chin on the frontal view image. After these manual steps, the program produces, in less than a minute, a 3D model having the same face and structure as the user's shows up on the screen and greets the user. The automatic process, invisible to the user, matches points of interest across images, determines the head motion, reconstructs the face in 3D space, and builds a photorealistic texture map from images. It uses many state-of-the-art computer vision and graphics techniques and also several innovative techniques we have recently developed specifically for face modeling. In this paper, we describe the architecture of our system as well as those new techniques.

The key challenge of face modeling from passive video images is the difficulty of establishing accurate and reliable correspondences across images due to the lack of texture in the facial skins. An additional challenge introduced by our choice of system setup is the variation of facial appearance. To acquire multiple views of the head, we ask the user to turn the head instead of moving the camera around the head or using multiple cameras. This greatly reduces the cost; anyone having a video camera (e.g., a webcam) can use our system. On the other hand, the appearance of the face changes in different images due to change in relative lighting condition. This makes image correspondence an even harder problem. Fortunately, as proven by an extensive set of experiments, we observe that even though it is difficult to extract dense 3D facial geometry from images, it is possible to match a sparse set of corners and use them to compute head motion and the 3D locations of these corner points. Furthermore, since faces are similar across people, the space of all possible faces can be represented by a small number of degrees of freedom. We represent faces by a linear set of deformations from a generic face model. We fit the linear face class to the sparse set of reconstructed corners to generate the complete face geometry. In this paper, we show that linear classes of face geometries can be used to effectively fit/interpolate a sparse set of 3D reconstructed points even when these points are quite noisy. This novel technique is the key to rapidly generate photorealistic 3D face models with minimal user intervention.

There are several technical innovations to make such a system robust and fast.

1. A technique to generate masking images eliminates most face-irrelevant regions based on generic face structure, face feature positions, and skin colors. The skin colors are computed on the fly for each user.

2. A robust head motion estimation algorithm takes advantage of the physical properties of the face feature points obtained from manual marking to reduce the number of unknowns.

3. We fit a set of face metrics (3D deformation vectors) to the reconstructed 3D points and markers to generate a complete face geometry. Linear face classes have already been used in (Blanz and Vetter, 1999), where a face is represented by a linear combination of physical face geometries and texture images and is reconstructed from images in a morphable model framework. We use linear classes of face metrics to represent faces and to interpolate a sparse set of 3D points for face modeling.

4. Finally, a model-based bundle-adjustment refines both the face geometry and head pose estimates by taking into account all available information from multiple images. Classical approaches first

refine 3D coordinates of isolated matched points, followed by fitting a parametric face model to those points. Our technique directly searches in the face model space, resulting in a more elegant formulation with fewer unknowns, fewer equations, a smaller search space, and hence a better posed system.

In summary, the approach we follow makes full use of rich domain knowledge whenever possible in order to make an ill-posed problem better behaved.

We provide several levels of details of our face modeling system. Section 4 provides an executive summary of our system. Section 5 describes the animated face model we use in our system. Section 6 is a guided tour, examining the system step by step. Sections 7 through 10 describes details of our technical contributions enumerated above. All of these exploit the rich generic knowledge of face structures. Section 11 provides more experimental results. Related work on face modeling is reviewed in Section 2. We give the conclusions and perspectives of our system in Section 12.

# 2 Previous Work

Facial modeling and animation has been a computer graphics research topic for over 25 years (DiPaola, 1991; Lanitis, Taylor and Cootes, 1997; Lee and Magnenat-Thalmann, 1998; Lee, Terzopoulos and Waters, 1993; Lee, Terzopoulos and Waters, 1995; Lewis, 1989; Magneneat-Thalmann, Minh, Angelis and Thalmann, 1989; Parke, 1972; Parke, 1974; Parke and Waters, 1996; Platt and Badler, 1981; Terzopoulos and Waters, 1990; Todd, Leonard, Shaw and Pittenger, 1980; Waters, 1987). The reader is referred to Parke and Waters' book (Parke and Waters, 1996) for a complete overview.

Lee et al. (Lee et al., 1993; Lee et al., 1995) developed techniques to clean up and register data generated from laser scanners. The obtained model is then animated by using a physically based approach.

DeCarlo et al. (DeCarlo, Metaxas and Stone, 1998) proposed a method to generate face models based on face measurements randomly generated according to anthropometric statistics. They were able to generate a variety of face geometries using these face measurements as constraints.

A number of researchers have proposed to create face models from two views (Akimoto, Suenaga and Wallace, 1993; Ip and Yin, 1996; Dariush, Kang and Waters, 1998). They all require two cameras which must be carefully set up so that their directions are orthogonal. Zheng (Zheng, 1994) developed a system to construct geometrical object models from image contours, but it requires a turn-table setup.

Pighin et al. (Pighin, Hecker, Lischinski, Szeliski and Salesin, 1998) developed a system to allow a user to manually specify correspondences across multiple images, and use vision techniques to compute 3D reconstructions. A 3D mesh model is then fit to the reconstructed 3D points. They were able to generate highly realistic face models, but with a manually intensive procedure.

Our work is inspired by Blanz and Vetter's work (Blanz and Vetter, 1999). They demonstrated that linear classes of face geometries and images are very powerful in generating convincing 3D human face models from images. We use a linear class of geometries to constrain our geometrical search space. One main difference is that we do not use an image database. Consequently, the types of skin colors we can handle are not limited by the image database. This eliminates the need for a fairly large image database to cover every skin type. Another advantage is that there are fewer unknowns since we need not solve for image database coefficients and illumination parameters. The price we pay is that we cannot generate complete models from a single image.

Kang et al. (Kang and Jones, 1999) also use linear spaces of geometrical models to construct 3D face models from multiple images. But their approach requires manually aligning the generic mesh to one of the images, which is in general a tedious task for an average user.

Guenter et al. (Guenter, Grimm, Wood, Malvar and Pighin, 1998) developed a facial animation capturing system to capture both the 3D geometry and texture image of each frame and reproduce high quality facial animations. The problem they solved is different from what is addressed here in that they assumed the person's 3D model was available and the goal was to track the subsequent facial deformations.

Fua et al. (Fua and Miccio, 1998; Fua and Miccio, 1999; Fua, Plaenkers and Thalmann, 1999; Fua, 2000) have done impressive work on face modeling from images, and their approach is the most similar to ours in terms of the vision technologies exploited. There are three major differences between their work

and ours: face model representation, model fitting, and camera motion estimation. They deform a generic face model to fit dense stereo data. Their face model contains a lot of free parameters (each vertex has three unknowns). Although some smoothness constraint is imposed, the fitting still requires many 3D reconstructed points. With our model, we only have about 60 parameters to estimate (see Section 5), and thus only a small set of feature points is necessary. Dense stereo matching is usually computationally more expensive than feature matching. Our system can produce a head model from 40 images in about two minutes on a PC with 366 MHz processor. Regarding camera motion estimation, they use a regularized bundle-adjustment on every image triplet, while we first use a bundle-adjustment on the first two views and determine camera motion for other views using 3D head model, followed by a model-based bundle adjustment (see Section 10).

# 3 Notation

A vector is denoted by a boldface lowercase letter such as $\mathbf{p}$ or by an uppercase letter in typewriter style such as $\mathtt{P}$. A matrix is usually denoted by a boldface uppercase letter such as $\mathbf{P}$.

We denote the homogeneous coordinates of a vector $\mathbf{x}$ by $\tilde{\mathbf{x}}$, i.e., the homogeneous coordinates of an image point $\mathbf{p} = (u, v)^T$ are $\tilde{\mathbf{p}} = (u, v, 1)^T$, and those of a 3D point $\mathtt{P} = (x, y, z)^T$ are $\tilde{\mathtt{P}} = (x, y, z, 1)^T$. A camera is described by a pinhole model, and a 3D point $\mathtt{P}$ and its image point $\mathbf{p}$ are related by

$$\lambda \tilde{\mathbf{p}} = \mathbf{A}\mathbf{P}\mathbf{M}\tilde{\mathtt{P}} \ , \tag{1}$$

where $\lambda$ is a scale, and $\mathbf{A}$, $\mathbf{P}$ and $\mathbf{M}$ are given by

$$\mathbf{A} = \begin{pmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{pmatrix} \ , \quad \mathbf{P} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \ , \quad \text{and} \quad \mathbf{M} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{pmatrix} \ .$$

The elements of matrix $\mathbf{A}$ are the intrinsic parameters of the camera and matrix $\mathbf{A}$ maps the normalized image coordinates to the pixel image coordinates (see e.g. (Faugeras, 1993)). Matrix $\mathbf{P}$ is the perspective projection matrix. Matrix $\mathbf{M}$ is the 3D rigid transformation (rotation $\mathbf{R}$ and translation $\mathbf{t}$) from the object/world coordinate system to the camera coordinate system. For simplicity, we also denote the nonlinear 3D-2D projection function (1) by function $\phi$ such that

$$\mathbf{p} = \phi(\mathbf{M}, \mathtt{P}) \ . \tag{2}$$

Here, the internal camera parameters are assumed to be known, although it is trivial to add them in our formulation.

When two images are concerned, a prime $'$ is added to denote the quantities related to the second image. When more images are involved, a subscript is used to specify an individual image.

The fundamental geometric constraint between two images is known as the *epipolar constraint* (Faugeras, 1993; Zhang, 1998a). It states that in order for a point $\mathbf{p}$ in one image and a point $\mathbf{p}'$ in the other image to be the projections of a single physical point in space, or, in other words, in order for them to be matched, they must satisfy

$$\tilde{\mathbf{p}}'^T \mathbf{A}'^{-T} \mathbf{E} \mathbf{A}^{-1} \tilde{\mathbf{p}} = 0 \ , \tag{3}$$

where $\mathbf{E} = [\mathbf{t}_r]_\times \mathbf{R}_r$ is known as the essential matrix, $(\mathbf{R}_r, \mathbf{t}_r)$ is the relative motion between the two images, and $[\mathbf{t}_r]_\times$ is a skew symmetric matrix such that $\mathbf{t}_r \times \mathbf{v} = [\mathbf{t}_r]_\times \mathbf{v}$ for any 3D vector $\mathbf{v}$.

# 4 System Overview

Figure 1 outlines the components of our system. The equipment includes a computer and a video camera connected to the computer. We assume the camera's intrinsic parameters have been calibrated, a reasonable assumption given the simplicity of calibration procedures (see e.g. (Zhang, 2000)).

The first stage is video capture. The user simply sits in front of the camera and turns his/her head from one side all the way to the other side in about 5 seconds. The user then selects an approximately

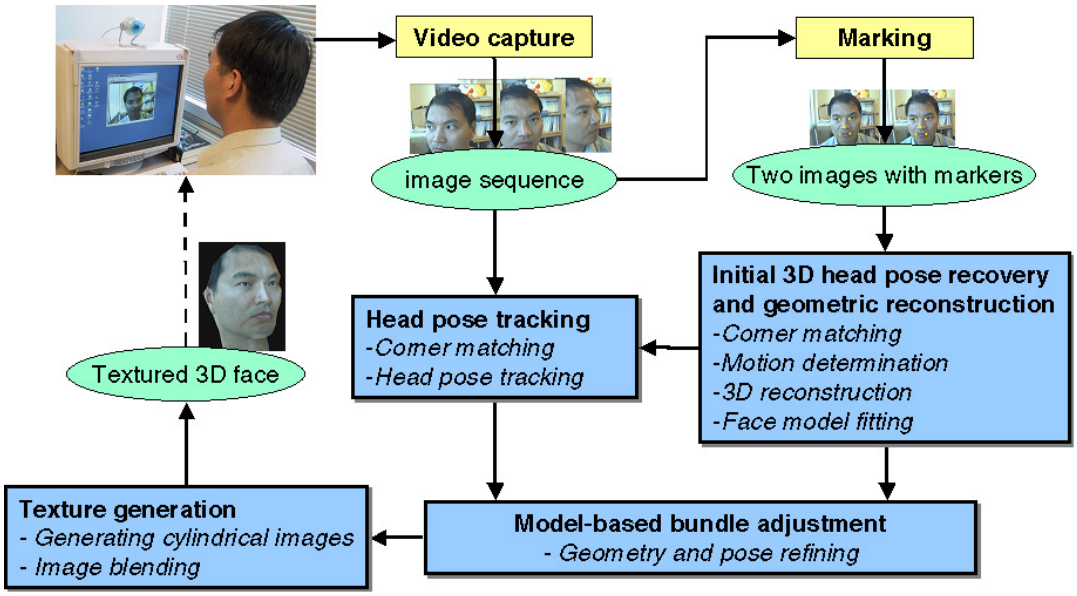



Figure 1: System overview

frontal view. This splits the video into two subsequences referred to as the *left* and *right* sequences, and the selected frontal image and its successive image are called the *base images*.

The second stage is feature-point marking. The user locates five markers in each of the two base images. The five markers correspond to the two inner eye corners, nose tip, and two mouth corners. As an optional step, the user can put three markers below the chin on the frontal view. This additional information usually improves the final face model quality. This manual stage could be replaced by an automatic facial feature detection algorithm.

The third stage is the recovery of initial face models. The system computes the face mesh geometry and the head pose with respect to the camera frame using the two base images and markers as input. This stage of the system involves corner detection, matching, motion estimation, 3D reconstruction and model fitting.

The fourth stage tracks the head pose in the image sequences. This is based on the same matching technique used in the previous stage, but the initial face model is also used for increasing accuracy and robustness.

The fifth stage refines the head pose and the face geometry using a recently developed technique called *model-based bundle-adjustment* (Shan, Liu and Zhang, 2001). The adjustment is performed by considering all point matches in the image sequences and using the parametric face space as the search space. Note that this stage is optionally. Because it is much more time consuming (about 8 minutes), we usually skip it in live demos.

The final stage blends all the images to generate a facial texture map. This is now possible because the face regions are registered by the head motion estimated in the previous stage. At this point, a textured 3D face model is available for immediate animation or other purposes (see Sect. 11.1).

In Sect. 6, we will describe in more details the techniques used in our system.

# 5 Facial Geometry Representation

Before going further, let us describe how a face is represented in our system.

Faces can be represented as volumes or surfaces, but the most commonly used representation is a polygonal surface because of the real-time efficiency that modern graphic hardware can display (Parke and Waters, 1996). A polygonal surface is typically formed from a triangular mesh. The face modeling problem

is then to determine the 3D coordinates of the vertices by using 3D digitizers, laser scanners, or computer vision techniques.

If we treat the coordinates of the vertices as free parameters, the number of unknowns is very large and we need a significant amount of data to model a face in a robust way. However, most faces look similar to each other (i.e., two eyes with a nose and mouth below them). Thus the number of degrees of freedom needed to define a wide rage of faces is limited. Vetter and Poggio (Vetter and Poggio, 1997) represented an arbitrary face image as a linear combination of a few hundred prototypes and used this representation (called linear object class) for image recognition, coding, and image synthesis. Blanz and Vetter (Blanz and Vetter, 1999) used a linear class of both images and 3D geometries for image matching and face modeling. The advantage of using a linear class of objects is that it eliminates most of the non-natural faces and significantly reduces the search space.

Instead of representing a face as a linear combination of real faces, we represent it as a linear combination of a neutral face and some number of face *metrics* where a metric is vector that linearly deforms a face in a certain way, such as to make the head wider, make the nose bigger, etc. To be more precise, let's denote the face geometry by a vector $\mathcal{S} = (\mathbf{v}_1^T, \ldots, \mathbf{v}_n^T)^T$ where $\mathbf{v}_i = (X_i, Y_i, Z_i)^T$ $(i = 1, \ldots, n)$ are the vertices, and a metric by a vector $\mathcal{M} = (\delta\mathbf{v}_1, \ldots, \delta\mathbf{v}_n)^T$, where $\delta\mathbf{v}_i = (\delta X_i, \delta Y_i, \delta Z_i)^T$. Given a neutral face $\mathcal{S}^0 = ({\mathbf{v}_1^0}^T, \ldots, {\mathbf{v}_n^0}^T)^T$, and a set of $m$ metrics $\mathcal{M}^j = ({\delta\mathbf{v}_1^j}^T, \ldots, {\delta\mathbf{v}_n^j}^T)^T$, the linear space of face geometries spanned by these metrics is

$$\mathcal{S} = \mathcal{S}^0 + \sum_{j=1}^{m} c_j \mathcal{M}^j \quad \text{subject to } c_j \in [l_j, u_j] \tag{4}$$

where $c_j$'s are the metric coefficients and $l_j$ and $u_j$ are the valid range of $c_j$. In our implementation, the neutral face and all the metrics are designed by an artist, and it is done once. The neutral face (see Figure 2) contains 194 vertices and 360 triangles. There are 65 metrics. We can easily add more metrics if we need finer control.

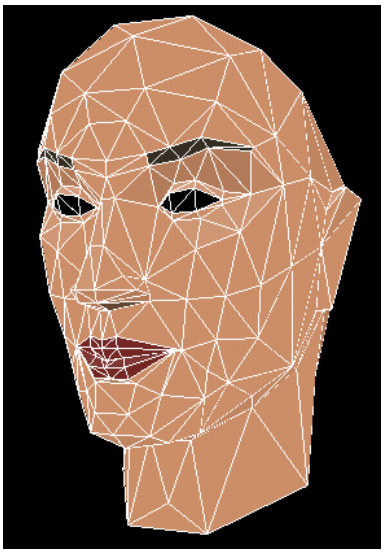


Figure 2: Neutral face.

# 6 A Tour of The System Illustrated With A Real Video Sequence

We now go through each step of the system using a real video sequence as an example. The video was captured in a normal room by a static camera while the head was moving in front. There is no control on the head motion, and the motion is unknown, of course. The video sequence can be found at `http://research.microsoft.com/~zhang/Face/duane.avi`

Figure 3: An example of two base images used for face modeling. Also shown are five manually picked markers indicated by yellow dots.

## 6.1 Marking and Masking

The base images are shown in Figure 3, together with the five manually picked markers. We have to determine first the motion of the head and match some pixels across the two views before we can fit an animated face model to the images. However, some processing of the images is necessary because there are at least three major groups of objects undergoing different motions between the two views: background, head, and other parts of the body such as the shoulder. If we do not separate them, there is no way to determine a meaningful head motion. The technique, to be described in Sect. 7, allows us to mask off most irrelevant pixels automatically. Figure 4 shows the masked base images to be used for initial face geometry recovery.

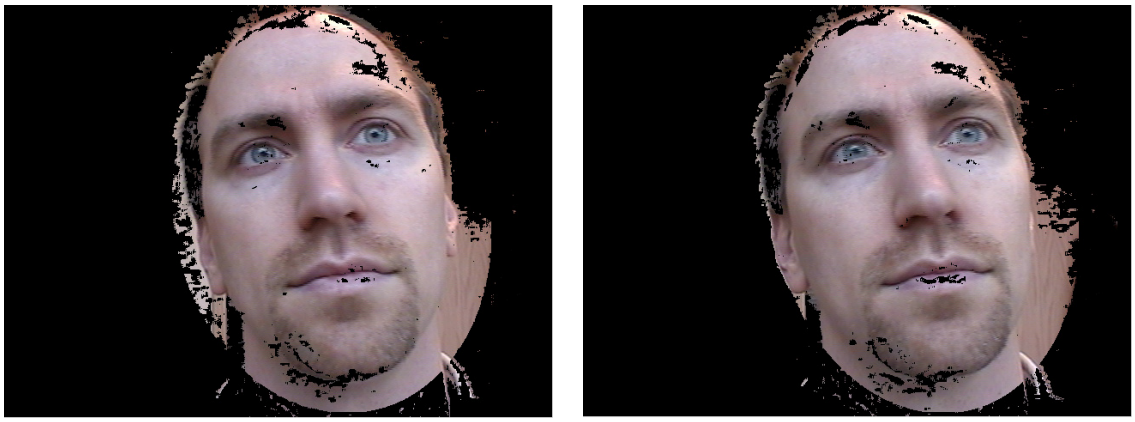


Figure 4: Maskedimages to be used in two-view image matching. See Sect. 7 for the technique used in obtaining the masked images.

Optionally, the user can also mark three points on the chin in one base image, as shown with three large yellow dots in Fig. 5. Starting from these dots, our system first tries to trace strong edges, which gives the red, green and blue curves in Fig. 5. Finally, a spline curve is fit to the three detected curves with an M-estimator. The small yellow dots in Fig. 5 are sample points of that curve. Depending on the face shape and lighting condition, the three original curves do not necessary represent accurately the chin, but the final spline represents the chin reasonably well. The chin curve will be used in face model fitting to be described in Sect. 6.5.

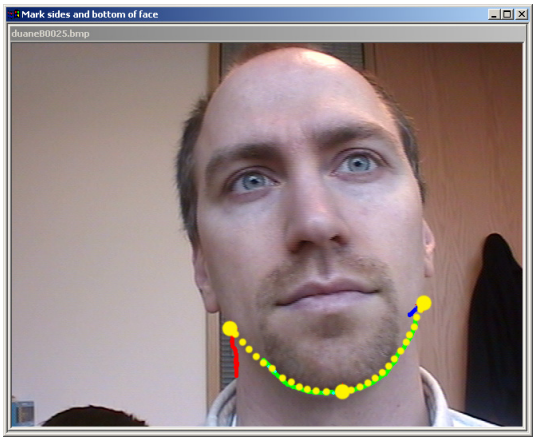



Figure 5: Marking the lower part of the face with three points (shown in large yellow dots). This is an optional step. See text for explanation.

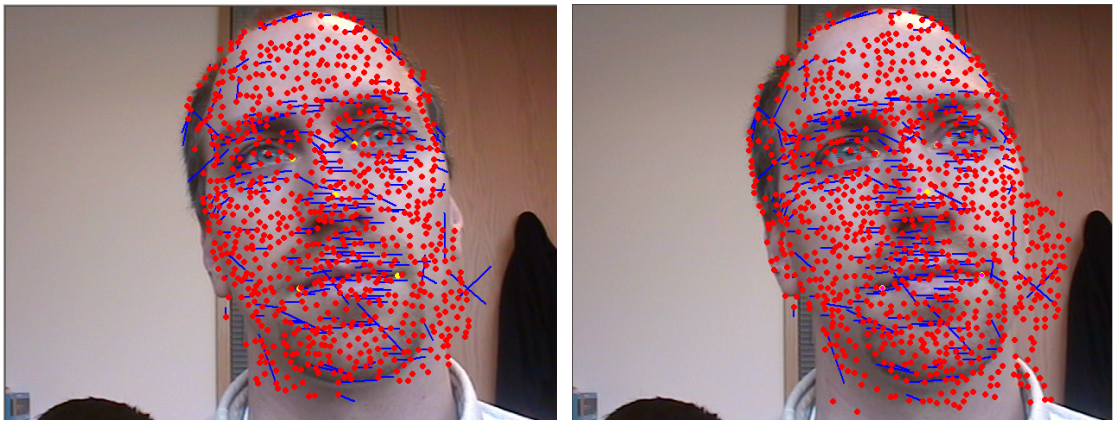


Figure 6: The set of matches established by correlation for the pair of images shown in Figure 3. Red dots are the detected corners. Blue lines are the motion vectors of the matches, with one endpoint (indicated by a red dot) being the matched corner in the current image and the other endpoint being the matched corner in the other image.

## 6.2 Matching Between the Two Base Images

One popular technique of image registration is optical flow (Horn and Schunk, 1981; Barron, Fleet and Beauchemin, 1994), which is based on the assumption that the intensity/color is conserved. This is not the case in our situation: the color of the same physical point appears to be different in images because the illumination changes when the head is moving. We therefore resort to a feature-based approach that is more robust to intensity/color variations. It consists of the following steps: (i) detecting corners in each image; (ii) matching corners between the two images; (iii) detecting false matches based on a robust estimation technique;(iv) determining the head motion; (v) reconstructing matched points in 3D space.

**Corner detection.** We use the Plessey corner detector, a well-known technique in computer vision (Harris and Stephens, 1988). It locates corners corresponding to high curvature points in the intensity surface if we view an image as a 3D surface with the third dimension being the intensity. Only corners whose pixels are white in the mask image are considered. See Figure 6 for the detected corners of the images shown in Figure 3 (807 and 947 corners detected respectively).

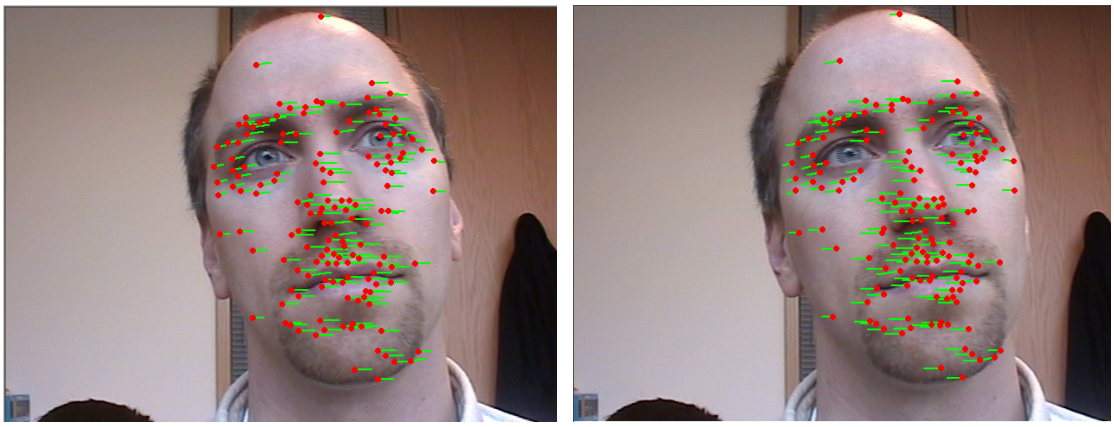


Figure 7: The final set of matches after discarding automatically false matches for the pair of images shown in Figure 3. Green lines are the motion vectors of the matches, with one endpoint (indicated by a red dot) being the matched corner in the current image and the other endpoint being the matched corner in the other image.

**Corner matching.** For each corner in the first image, we choose an $11 \times 11$ window centered on it, and compare the window with windows of the same size, centered on the corners in the second image. A zero-mean normalized cross correlation between two windows is computed (Faugeras, 1993). If we rearrange the pixels in each window as a vector, the correlation score is equivalent to the cosine angle between two intensity vectors. It ranges from -1, for two windows which are not similar at all, to 1, for two windows which are identical. If the largest correlation score exceeds a prefixed threshold (0.866 in our case), then that corner in the second image is considered to be the *match candidate* of the corner in the first image. The match candidate is retained as a *match* if and only if its match candidate in the first image happens to be *the corner being considered*. This symmetric test reduces many potential matching errors.

For the example shown in Figure 3, the set of matches established by this correlation technique is shown in Figure 6. There are 237 matches in total.

**False match detection.** The set of matches established so far usually contains false matches because correlation is only a heuristic. The only geometric constraint between two images is the epipolar constraint (3). If two points are correctly matched, they must satisfy this constraint, which is unknown in our case. Inaccurate location of corners because of intensity variation or lack of strong texture features is another source of error. We use the technique described in (Zhang, 1998a) to detect both false matches and poorly located corners, and simultaneously estimate the epipolar geometry (in terms of the essential matrix $\mathbf{E}$). That technique is based on a robust estimation technique known as the *least median squares* (Rousseeuw and Leroy, 1987), which searches in the parameter space to find the parameters yielding the smallest value for the *median* of squared residuals computed for the entire data set. Consequently, it is able to detect false matches in as many as 49.9% of the whole set of matches.

For the example shown in Figure 3, the final set of matches is shown in Figure 7. There are 148 remaining matches. Compared with those shown in Figure 6 89 matches have been discarded.

## 6.3 Robust Head Motion Estimation

We have developed a new algorithm to compute the head motion between two views from the correspondences of five feature points including eye corners, mouth corners and nose top, and zero or more other image point matches.

If the image locations of these feature points are precise, one could use a five-point algorithm to compute camera motion. However, this is usually not the case in practice since the pixel grid plus errors introduced by a human do not result in feature points with high precision. When there are errors, a five-point algorithm is not robust even when refined with a bundle adjustment technique. The key idea of our

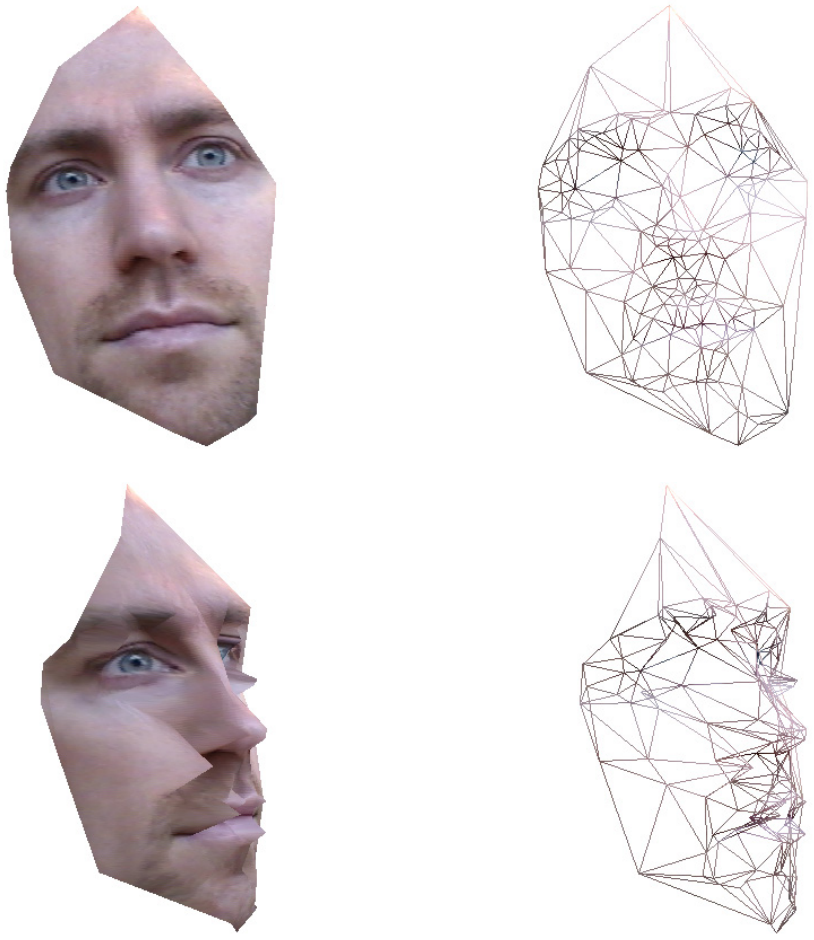


Figure 8: Reconstructed corner points. This coarse mesh is used later to fit a face model.

algorithm is to use the physical properties of the feature points to improve robustness. We use the property of symmetry to reduce the number of unknowns. We put reasonable lower and upper bounds on the nose height and represent the bounds as inequality constraints. As a result, the algorithm becomes significantly more robust. This algorithm will be described in Sect. 8.

## 6.4 3D Reconstruction.

Once the motion is estimated, matched points can be reconstructed in 3D space with respect to the camera frame at the time when the first base image was taken. Let $(\mathbf{p}, \mathbf{p}')$ be a pair of matched points, and P be their corresponding point in space. 3D point P is estimated such that $\|\mathbf{p} - \hat{\mathbf{p}}\|^2 + \|\mathbf{p}' - \hat{\mathbf{p}}'\|^2$ is minimized, where $\hat{\mathbf{p}}$ and $\hat{\mathbf{p}}'$ are projections of P in both images according to (1).

Two views of the 3D reconstructed points for the example shown in Figure 3 are shown in Figure 8. The wireframes shown on the right are obtained by perform Delaunay triangulation on the matched points. The pictures shown on the left are obtained by using the first base image as the texture map.

## 6.5 Face Model Fitting From Two Views

We now only have a set of unorganized noisy 3D points from matched corners and markers. The face model fitting process consists of two steps: fitting to the 3D reconstructed points and fine adjustment using

image information. The first consists in estimating both the *pose* of the head and the metric coefficients that minimize the distances from the reconstructed 3D points to the face mesh. The estimated head pose is defined to be the pose with respect to the camera coordinate system when the first base image was taken, and is denoted by $\mathbf{T}_0$. In the second step, we search for silhouettes and other face features in the images and use them, and also the chin curve if available from the marking step (Sect. 6.1), to refine the face geometry. Details of face model fitting are provided in Section 9.

Figure 9 shows the reconstructed 3D face mesh from the two example images (see Figure 3). The mesh is projected back to the two images. Figure 10 shows two novel views using the first image as the texture map. The texture corresponding to the right side of the face is still missing.

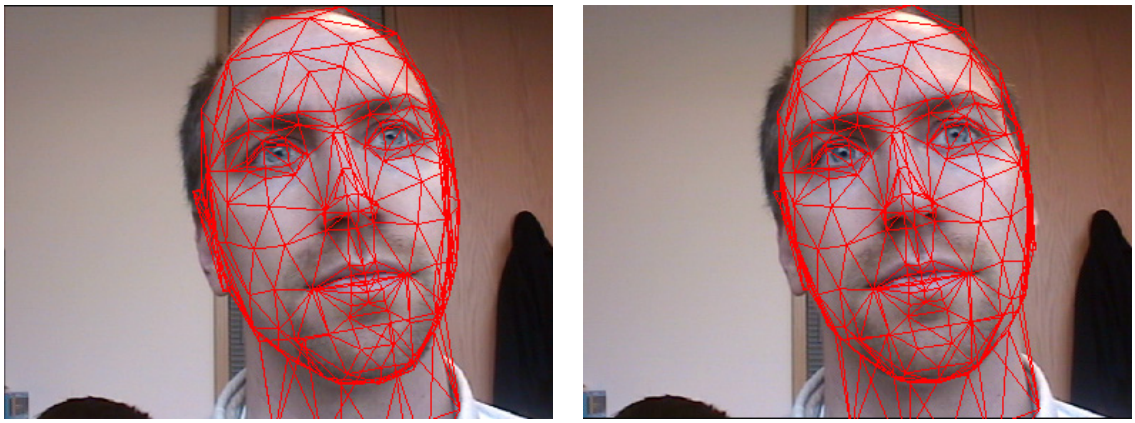


Figure 9: The constructed 3D face mesh is projected back to the two base images.

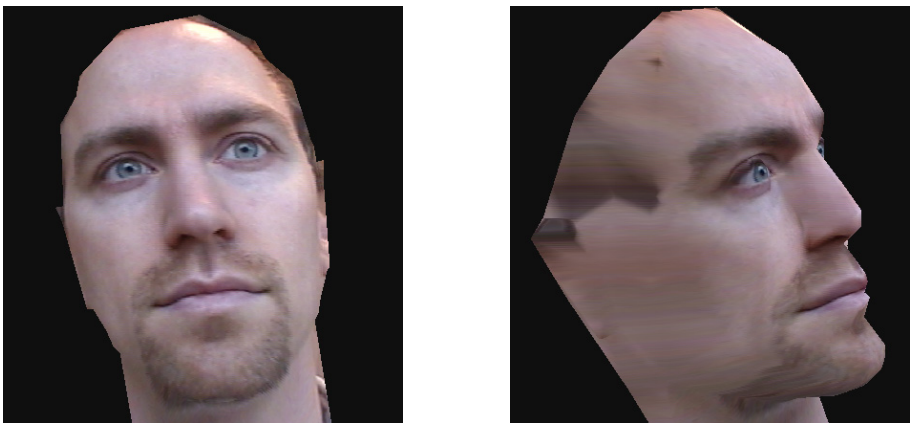


Figure 10: Two novel views of the reconstructed 3D face mesh with the the first base image as texture.

## 6.6 Determining Head Motions in Video Sequences

Now we have the geometry of the face from only two views that are close to the frontal position. As can be seen in Figure 10, for the sides of the face, the texture from the two images is therefore quite poor or even not available at all. Since each image only covers a portion of the face, we need to combine all the images in the video sequence to obtain a complete texture map. This is done by first determining the head pose for the images in the video sequence and then blending them to create a complete texture map.

Successive images are first matched using the same technique as described in Section 6.2. We could combine the resulting motions incrementally to determine the head pose. However, this estimation is quite

noisy because it is computed only from 2D points. As we already have the 3D face geometry, a more reliable pose estimation can be obtained by combining both 3D and 2D information, as follows.

Let us denote the first base image by $I_0$, the images on the video sequences by $I_1, \ldots, I_v$, the relative head motion from $I_{i-1}$ to $I_i$ by $\mathcal{R}_i = \begin{pmatrix} \mathbf{R}_{ri} & \mathbf{t}_{ri} \\ \mathbf{0}^T & 1 \end{pmatrix}$, and the head pose corresponding to image $I_i$ with respect to the camera frame by $\mathbf{T}_i$. The algorithm works incrementally, starting with $I_0$ and $I_1$. For each pair of images $(I_{i-1}, I_i)$, we first use the corner matching algorithm described in Section 6.2 to find a set of matched corner pairs $\{(\mathbf{p}_j, \mathbf{p}'_j) | j = 1, \ldots, l\}$. For each $\mathbf{p}_j$ in $I_{i-1}$, we cast a ray from the camera center through $\mathbf{p}_j$, and compute the intersection $\mathtt{P}_j$ of that ray with the face mesh corresponding to image $I_{i-1}$. According to (1), $\mathcal{R}_i$ is subject to the following equations

$$\mathbf{A}\mathbf{P}\mathcal{R}_i\tilde{\mathtt{P}}_j = \lambda_j\tilde{\mathbf{p}}'_j \quad \text{for } j = 1, \ldots, l, \tag{5}$$

where $\mathbf{A}$, $\mathbf{P}$, $\mathtt{P}_j$ and $\mathbf{p}'_j$ are known. Each of the above equations gives two constraints on $\mathcal{R}_i$. We compute $\mathcal{R}_i$ with a technique described in (Faugeras, 1993). After $\mathcal{R}_i$ is computed, the head pose for image $I_i$ in the camera frame is given by $\mathbf{T}_i = \mathcal{R}_i\mathbf{T}_{i-1}$. The head pose $\mathbf{T}_0$ is known from Section 6.5.

In general, it is inefficient to use all the images in the video sequence for texture blending, because head motion between two consecutive frames is usually small. To avoid unnecessary computation, the following process is used to automatically select images from the video sequence. Let us call the amount of rotation of the head between two consecutive frames the *rotation speed*. If $s$ is the current rotation speed and $\alpha$ is the desired angle between each pair of selected images, the next image is selected $\lfloor(\alpha/s)\rfloor$ frames away. In our implementation, the initial guess of the rotation speed is set to 1 degree/frame and the desired separation angle is equal to 5 degrees.

Figure 11 and Figure 12 show the tracking results of the two example video sequences (The two base images are shown in Figure 3). The images from each video sequence are automatically selected using the above algorithm.


Figure 11: The face mesh is projected back to the automatically selected images from the video sequence where the head turns to the left.

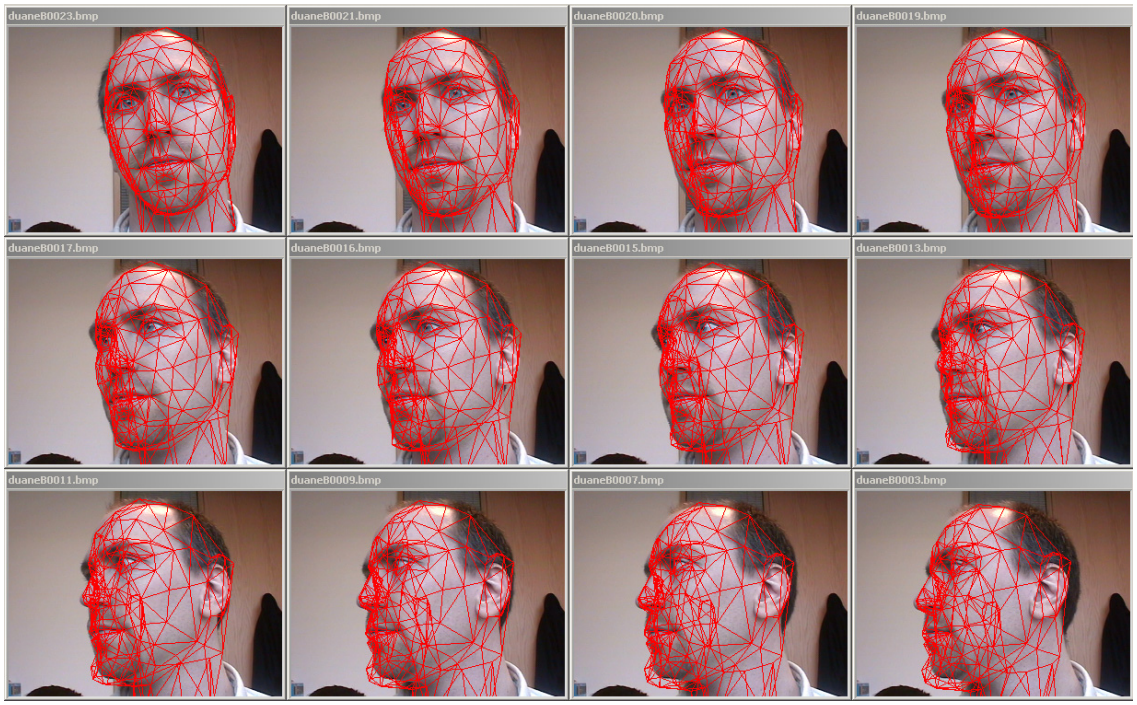



Figure 12: The face mesh is projected back to the automatically selected images from the video sequence where the head turns to the right.

## 6.7 Model-Based Bundle Adjustment

We now have an initial face model from two base images, a set of pairwise point matches over the whole video sequence, and an initial estimate of the head poses in the video sequence which is obtained incrementally based on the initial face model. Naturally, we want to refine the face model and head pose estimates by taking into account all available information simultaneously. A classical approach is to perform bundle adjustment to determine the head motion and 3D coordinates of isolated points corresponding to matched image points, followed by fitting the parametric face model to the reconstructed isolated points. We have developed a new technique called *model-based bundle adjustment*, which directly searches in the face model space to minimize the same objective function as that used in the classical bundle adjustment. This results in a more elegant formulation with fewer unknowns, fewer equations, a smaller search space, and hence a better posed system. More details are provided in Section 10.

Figure 13 shows the refined result on the right sequence, which should be compared with that shown in Fig. 12. The projected face mesh is overlaid on the original images. We can observe clear improvement in the silhouette and chin regions.

## 6.8 Texture Blending

After the head pose of an image is computed, we use an approach similar to Pighin et al.'s method (Pighin et al., 1998) to generate a view independent texture map. We also construct the texture map on a virtual cylinder enclosing the face model. But instead of casting a ray from each pixel to the face mesh and computing the texture blending weights on a pixel by pixel basis, we use a more efficient approach. For each vertex on the face mesh, we compute the blending weight for each image based on the angle between surface normal and the camera direction (Pighin et al., 1998). If the vertex is invisible, its weight is set to 0.0. The weights are then normalized so that the sum of the weights over all the images is equal to 1.0. We then set the colors of the vertexes to be their weights, and use the rendered image of the cylindrical mapped mesh as the weight map. For each image, we also generate a cylindrical texture map by rendering the cylindrical mapped mesh with the current image as texture map. Let $C_i$ and $W_i$ ($i = 1, \ldots, k$) be the

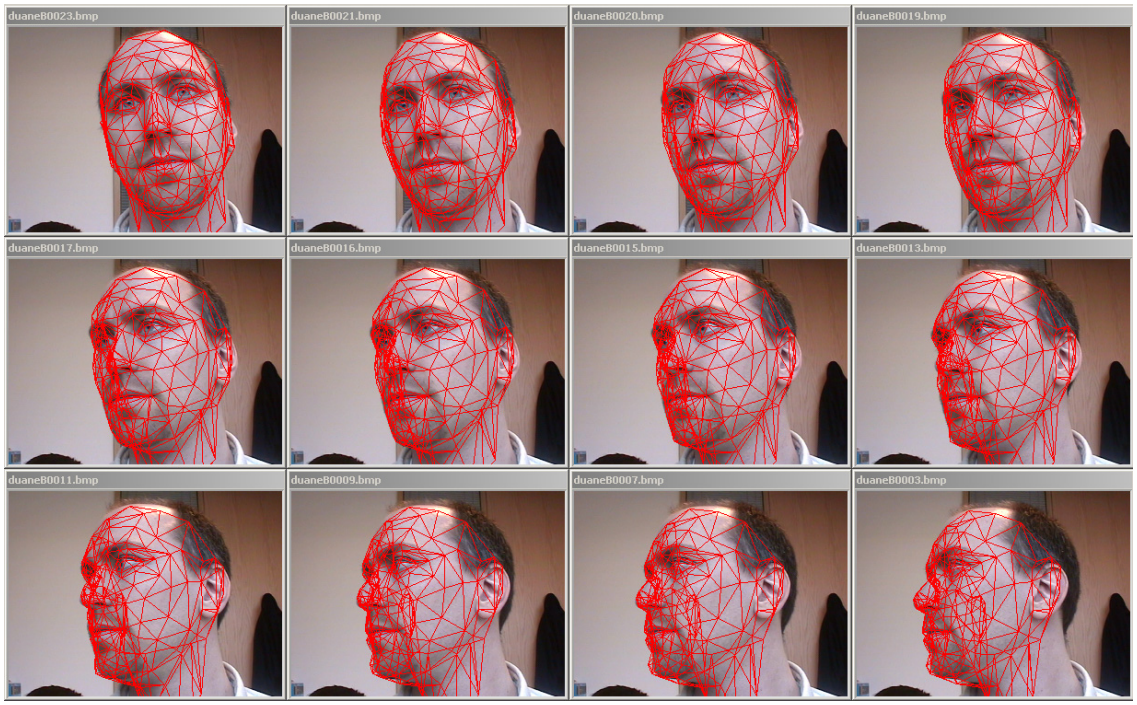



Figure 13: After model-based bundle adjustment, the refined face mesh is projected back to the automatically selected images from the video sequence where the head turns to the right.

cylindrical texture maps and the weight maps. Let $C$ be the final blended texture map. For each pixel $(u, v)$, its color on the final blended texture map is

$$C(u, v) = \sum_{i=1}^{k} W_i(u, v) C_i(u, v). \tag{6}$$

Because the rendering operations can be done using graphics hardware, this approach is very fast.

Figure 14 shows the blended texture map from the example video sequences 11 and 12. Figure 15 shows two novels views of the final 3D face model. Compared with those shown in Fig. 10, we now have a much better texture on the side.

# 7 Image Masking

As said earlier, there are at least three major groups of objects undergoing different motions between the two views: background, head, and other parts of the body such as the shoulder. If we do not separate them, there is no way to determine a meaningful head motion. Since the camera is static, we can expect to remove the background by subtracting one image from the other. However, as the face color changes smoothly, a portion of the face may be marked as background, as shown in Figure 16a. Another problem with this image subtraction technique is that the moving body and the head cannot be distinguished.

As we have marked five points on the face, we can actually build a color model of the face skin. We select pixels below the eyes and above the mouth, and compute a Gaussian distribution of their colors in the RGB space. If the color of a pixel matches this face skin color model, the pixel is marked as a part of the face. An example is shown in Figure 16b. As we can notice, some background pixels are marked as face skin.

Either union or intersection of the two mask images is not enough to locate the face because it will include either too many (e.g., including undesired moving body) or too few (e.g., missing desired eyes and mouth) pixels. As we already have information about the position of eye corners and mouth corners, we

Figure 14: The blended texture image.

define two ellipses as shown in Figure 17a. The inner ellipse covers most of the face, while the outer ellipse is usually large enough to enclose the whole head. Let $d_e$ be the image distance between the two inner eye corners, and $d_{em}$, the vertical distance between the eyes and the mouth. The width and height of the inner ellipse are set to $5d_e$ and $3d_{em}$. The outer ellipse is 25% larger than the inner one. Within the inner ellipse, the "union" operation is used. Between the inner and out ellipses, only the image subtraction is used, except for the lower part where the "intersection" operation is used. The lower part aims at removing the moving body, and is defined to be $0.6d_{em}$ below the mouth, as illustrated by the red area in Figure 17a. An example of the final mask is shown in Figure 17b.

# 8 Robust Head Motion Determination From Two Views

Our system relies on an initial face modeling from two base images, and therefore a robust head motion determination is key to extract 3D information from images. This section describes an algorithm to compute the head motion between two views from the correspondences of five feature points including eye corners, mouth corners and nose top, and zero or more other image point matches.

## 8.1 Head Motion Estimation From Five Feature Points

We use $E_1$, $E_2$, $M_1$, $M_2$, and $N$ to denote the left eye corner, right eye corner, left mouth corner, right mouth corner, and nose top, respectively (See Figure 18). Denote $E$ as the midpoint of $E_1E_2$ and $M$ the midpoint of $M_1M_2$. Notice that human faces exhibit some strong structural properties. For example, left and right sides are very close to being symmetric about the nose; eye corners and mouth corners are almost coplanar. We therefore make the following reasonable assumptions:

- $NM$ is perpendicular to $M_1M_2$,
- $NE$ is perpendicular to $E_1E_2$, and

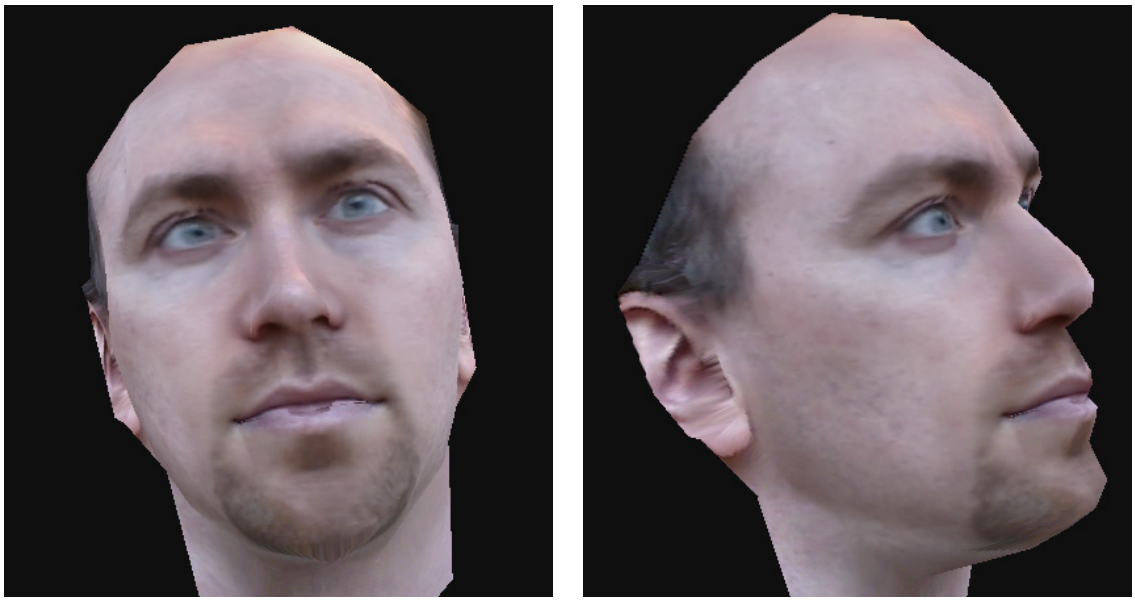


Figure 15: Two novel views of the final 3D face model.

- $E_1E_2$ is parallel to $M_1M_2$.

Let $\pi$ be the plane defined by $E_1, E_2, M_1$ and $M_2$ . Let $O$ denote the projection of point $N$ on plan $\pi$. Let $\Omega_0$ denote the coordinate system with $O$ as the origin, $ON$ as the $Z$-axis, $OE$ as the $Y$-axis. In this coordinate system, based on the assumptions mentioned earlier, we can define the coordinates of $E_1, E_2, M_1, M_2, N$ as $(-a, b, 0)^T$, $(a, b, 0)^T$, $(-d, -c, 0)^T$, $(d, -c, 0)^T$, $(0, 0, e)^T$, respectively. Thus, we only need 5 parameters to define these five points in this local coordinate system, instead of 9 parameters for generic five points.

Let $\mathbf{t}$ be the coordinates of $O$ in the camera coordinate system, and $\mathbf{R}$ the rotation matrix whose three columns correspond to the three coordinate axes of $\Omega_0$. We call $\mathbf{T} = \left(\begin{smallmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{smallmatrix}\right)$ the *head pose transform*. For each point $\mathtt{P} \in \{E_1, E_2, M_1, M_2, N\}$, its coordinates in the camera coordinate system are $\mathbf{R}\mathtt{P} + \mathbf{t}$.

Given two images of the head under two different poses (assuming the camera is static), let $\mathbf{T} = \left(\begin{smallmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{smallmatrix}\right)$ and $\mathbf{T}' = \left(\begin{smallmatrix} \mathbf{R}' & \mathbf{t}' \\ \mathbf{0}^T & 1 \end{smallmatrix}\right)$ be the corresponding head pose transforms. For each point $\mathtt{P}_i \in \{E_1, E_2, M_1, M_2, N\}$, if we denote its image point in the first view by $\mathbf{p}_i$ and that in the second view by $\mathbf{p}'_i$, according to (2), we have

$$\mathbf{p}_i = \boldsymbol{\phi}(\mathbf{T}, \mathtt{P}_i) \quad \text{and} \quad \mathbf{p}'_i = \boldsymbol{\phi}(\mathbf{T}', \mathtt{P}_i) \; . \tag{7}$$

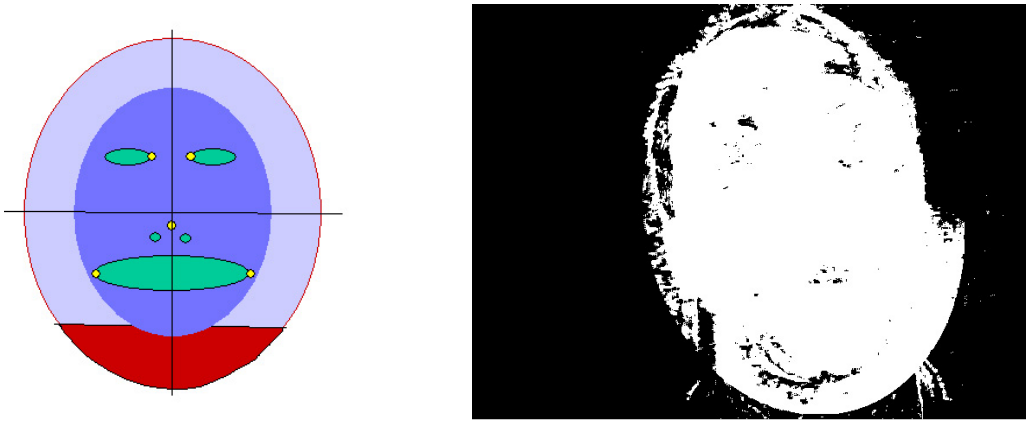
Notice that we can fix one of the $a, b, c, d, e$ since the scale of the head size cannot be determined from the images. As is well known, each pose has 6 degrees of freedom. Therefore the total number of unknowns is 16, and the total number of equations is 20. If we instead use their 3D coordinates as unknowns as in any typical bundle adjustment algorithms, we would end up with 20 unknowns, the same number of the available equations. By using the generic properties of the face structure, the system becomes over-constrained, making the pose determination more robust.

To make the system even more robust, we add an inequality constraint on $e$. The idea is to force $e$ to be positive and not too large compared to $a, b, c, d$. This is obvious since the nose is always out of plane $\pi$. In particular, we use the following inequality:

$$0 \leq e \leq 3a \tag{8}$$

(a) (b)

Figure 16: (a) Mask obtained by subtracting images (black pixels are considered as background); (b) Mask obtained by using a face skin color model (white pixels are considered as face skin).



(a) (b)

Figure 17: (a) Definition of face ellipses; (b) Final mask obtained with our preprocessing technique.

We chose 3 as the upper bound of $e/a$ simply because it seems reasonable to us. The inequality constraint is finally converted to equality constraint by using penalty function.

$$P_{\text{nose}} = \begin{cases} e^2 & \text{if } e < 0; \\ 0 & \text{if } 0 \leq e \leq 3a; \\ (e-3a)^2 & \text{if } e > 3a. \end{cases} \tag{9}$$

In summary, based on equations (7), and (9), we estimate $a, b, c, d, e, (\mathbf{R}, \mathbf{t})$ and $(\mathbf{R}', \mathbf{t}')$ by minimizing

$$\mathcal{F}_{\text{5pts}} = \sum_{i=1}^{5} w_i (\|\mathbf{p}_i - \boldsymbol{\phi}(\mathbf{T}, \mathtt{P}_i)\|^2 + \|\mathbf{p}'_i - \boldsymbol{\phi}(\mathbf{T}', \mathtt{P}_i))\|^2) + w_n P_{\text{nose}} \tag{10}$$

where $w_i$'s and $w_n$ are the weighting factors, reflecting the contribution of each term. In our case, $w_i = 1$ except for the nose term which has a weight of 0.5 because it is usually more difficult to locate the nose tip accurately than other feature points. The weight for penalty, $w_n$, is set to 10. The objective function (10) is minimized using a Levenberg-Marquardt method (More, 1977). As mentioned earlier, we set $a$ to a constant during minimization since the global head size cannot be determined from images.

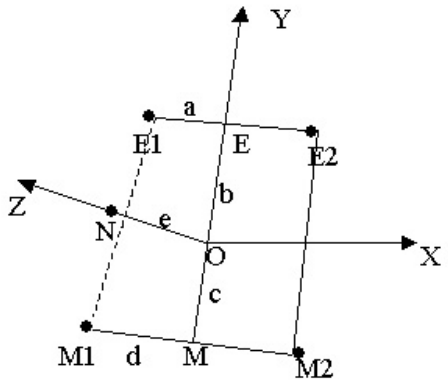



Figure 18: The new coordinate system $\Omega_0$.

## 8.2 Incorporating Image Point Matches

If we estimate camera motion using only the five user marked points, the result is sometimes not very accurate because the markers contain human errors. In this section, we describe how to incorporate the image point matches (obtained as described in Sect. 6.2) to improve precision.

Let $(\mathbf{p}_j, \mathbf{p}'_j)$ $(j = 1, \ldots, K)$ be the $K$ point matches, each corresponding to the projection of a 3D point $\mathtt{P}_j$ according to the perspective projection (7). Obviously, we have to estimate $\mathtt{P}_j$'s which are unknown. Assuming that each image point is extracted with the same accuracy, we can estimate $a, b, c, d, e$, $(\mathbf{R}, \mathbf{t})$, $(\mathbf{R}', \mathbf{t}')$ and $\{\mathtt{P}_j\}$ $(j = 1, \ldots, K)$ by minimizing

$$\mathcal{F} = \mathcal{F}_{\text{5pts}} + w_p \sum_{j=1}^{K} (\|\mathbf{p}_j - \boldsymbol{\phi}(\mathbf{T}, \mathtt{P}_i))\|^2 + \|\mathbf{m}'_j - \boldsymbol{\phi}(\mathbf{T}', \mathtt{P}_i))\|^2) \tag{11}$$

where $\mathcal{F}_{\text{5pts}}$ is given by (10), and $w_p$ is the weighting factor. We set $w_p = 1$ by assuming that the extracted points have the same accuracy as those of eye corners and mouth corners. The minimization can again be performed using a Levenberg-Marquardt method.

This is quite a large minimization problem since we need to estimate $16 + 3K$ unknowns, and therefore it is computationally expensive especially for large $K$ (usually $K > 100$). Fortunately, as shown in (Zhang, 1998b), we can eliminate the 3D points using a first order approximation. More precisely, it can be shown that if $\mathbf{p}_j$ and $\mathbf{p}'_j$ are in normalized image coordinates, we have

$$\|\mathbf{p}_j - \boldsymbol{\phi}(\mathbf{T}, \mathtt{P}_i))\|^2 + \|\mathbf{m}'_j - \boldsymbol{\phi}(\mathbf{T}', \mathtt{P}_i))\|^2 \approx \frac{(\tilde{\mathbf{p}}_j'^T \mathbf{E} \tilde{\mathbf{p}}_j)^2}{\tilde{\mathbf{p}}_j^T \mathbf{E}^T \mathbf{Z}\mathbf{Z}^T \mathbf{E} \tilde{\mathbf{p}}_j + \tilde{\mathbf{p}}_j'^T \mathbf{E}\mathbf{Z}\mathbf{Z}^T \mathbf{E}^T \tilde{\mathbf{p}}'_j}$$

where $\mathbf{Z} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$, and $\mathbf{E}$ is the essential matrix between the two images, i.e., $\mathbf{E} = [\mathbf{t}_r]_\times \mathbf{R}_r$. Here, $(\mathbf{R}_r, \mathbf{t}_r)$ is the relative motion between two views. It is easy to see that

$$\begin{aligned} \mathbf{R}_r &= \mathbf{R}' \mathbf{R}^T \ , \\ \mathbf{t}_r &= \mathbf{t}' - \mathbf{R}' \mathbf{R}^T \mathbf{t} \ . \end{aligned}$$

In summary, the objective function (11) becomes

$$\mathcal{F} = \mathcal{F}_{\text{5pts}} + w_p \sum_{j=1}^{K} \frac{(\tilde{\mathbf{p}}_j'^T \mathbf{E} \tilde{\mathbf{p}}_j)^2}{\tilde{\mathbf{p}}_j^T \mathbf{E}^T \mathbf{Z}\mathbf{Z}^T \mathbf{E} \tilde{\mathbf{p}}_j + \tilde{\mathbf{p}}_j'^T \mathbf{E}\mathbf{Z}\mathbf{Z}^T \mathbf{E}^T \tilde{\mathbf{p}}'_j} \tag{12}$$

Notice that this is a much smaller minimization problem. We only need to estimate 16 parameters as in the five-point problem (10), instead of $16 + 3K$ unknowns.

To obtain a good initial estimate, we first use only the five feature points to estimate the head motion by using the algorithm described in Section 8.1. Thus we have the following two step algorithm:

**Step 1.** Set $w_p = 0$. Solve minimization problem (12).

**Step 2.** Set $w_p = 1$. Solve minimization problem (12) using the result of step 1 as the initial estimates.

Notice that we can apply this idea to the more general cases where the number of feature points is not five. For example, if there are only two eye corners and mouth corners, we'll end up with 14 unknowns and $16 + 3K$ equations. Other symmetric feature points (such as the outside eye corners, nostrils, etc) can be added to (12) in a similar way by using the local coordinate system $\Omega_0$.

### 8.3 Experimental Results

In this section, we show some test results to compare our new algorithm with the traditional algorithms. Since there are multiple traditional algorithms, we chose to implement the algorithm as described in (Zhang, 1997). It works by first computing an initial estimate of the head motion from the essential matrix (Faugeras, 1993), and then re-estimate the motion with a nonlinear least-squares technique.

We have run both the traditional algorithm and our new algorithm on many real examples. We found many cases where the traditional algorithm fails while our new algorithm successfully produces reasonable motion estimates. Figure 19 is such an example. The motion computed by the traditional algorithm is completely bogus, and the 3D reconstruction gives meaningless results, but our new algorithm gives a reasonable result.

In order to know the ground truth, we have also performed experiments on artificially generated data. We arbitrarily select 80 vertices from a 3D face model (Figure 2) and project them on two views (the head motion is eight degrees apart). The image size is 640 by 480 pixels. We also project the five 3D feature points (eye corners, nose tip, and mouth corners) to generate the image coordinates of the markers. We then add Gaussian noise to the coordinates $(u, v)$ of the image points. The Gaussian noise has mean zero and standard deviation ranging from 0.4 to 1.2 pixels. Notice that we add noise to the markers' coordinates as well. The results are plotted in Figure 20. The blue curve shows the results of the traditional algorithm and the red curve shows the results of our new algorithm. The horizontal axis is the standard deviation of the noise distribution. The vertical axis is the difference between the estimated motion and the actual motion. The translation vector of the estimated motion is scaled so that its magnitude is the same as the actual motion. The difference between two rotations is measured as the Euclidean distance between the two rotational matrices. We compute the average of combined motion errors from 20 random trials for each noise level. We can see that as the noise increases, the error of the traditional algorithm has a sudden jump at certain noise level, indicating failures in several trials, while the errors of our new algorithm increase much more slowly.

## 9 Face Model Fitting From Two Views

The face model fitting process consists of two steps: fitting to 3D reconstructed points and fine adjustment using image information.

### 9.1 3D Fitting

Given a set of reconstructed 3D points from matched corners and markers, the fitting process searches for both the *pose* of the face and the metric coefficients to minimize the distances from the reconstructed 3D points to the face mesh. The pose of the face is the transformation $\mathbf{T} = \begin{pmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{pmatrix}$ from the coordinate frame of the neutral face mesh to the camera frame, where $\mathbf{R}$ is a $3 \times 3$ rotation matrix, $\mathbf{t}$ is a translation, and $s$ is a global scale. For any 3D vector $\mathtt{P}$, we use notation $\mathbf{T}(\mathtt{P}) = s\mathbf{R}\mathtt{P} + \mathbf{t}$.

The vertex coordinates of the face mesh in the camera frame is a function of both the metric coefficients and the pose of the face. Given metric coefficients $(c_1, \ldots, c_m)$ and pose $\mathbf{T}$, the face geometry in the camera frame is given by

$$\mathcal{S} = \mathbf{T}(\mathcal{S}^0 + \sum_{j=1}^{m} c_j \mathcal{M}^j). \tag{13}$$

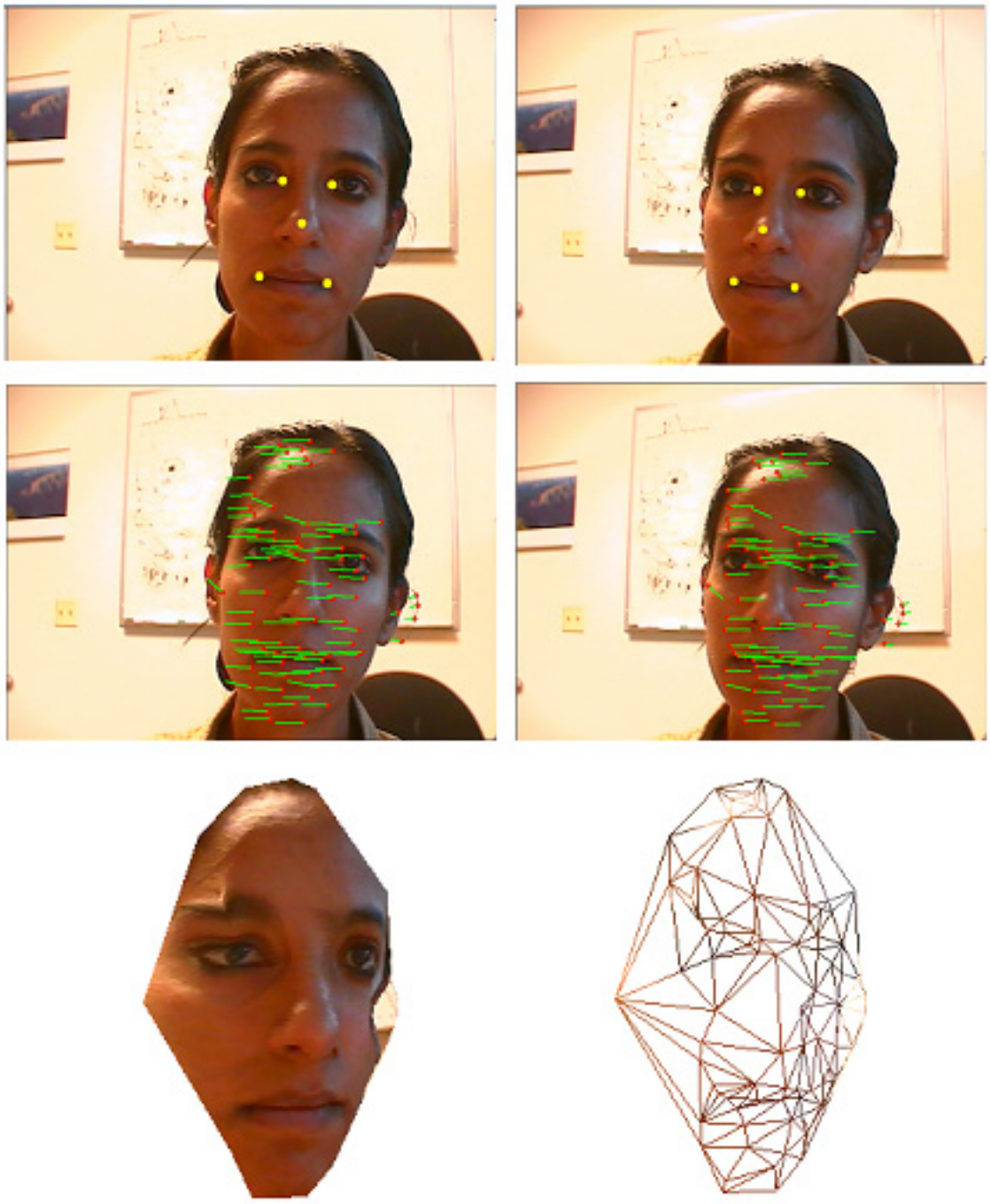


Figure 19: Top row: a pair of images with five markers. Middle row: matched image points. Bottom row: a novel view of the 3D reconstruction of the image matching points with the head motion computed by our new algorithm. Note that the traditional motion estimation failed in this example.

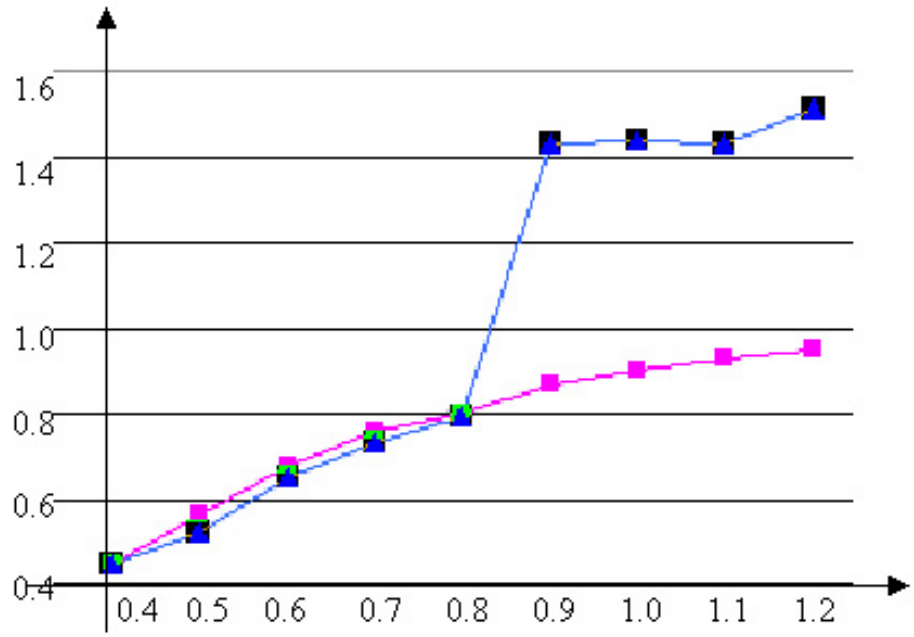



Figure 20: Comparison of the new algorithm with the traditional algorithm. The blue curve shows the results of the traditional algorithm and the red curve shows the results of our new algorithm. The horizontal axis is the standard deviation of the added noise. The vertical axis is the error of computed head motion.

Since the face mesh is a triangular mesh, any point on a triangle is a linear combination of the three triangle vertexes in terms of barycentric coordinates. So any point on a triangle is also a function of $\mathbf{T}$ and metric coefficients. Furthermore, when $\mathbf{T}$ is fixed, it is simply a linear function of the metric coefficients.

Let $(\mathtt{P}_1, \mathtt{P}_2, \ldots, \mathtt{P}_n)$ be the reconstructed corner points, and $(\mathtt{Q}_1, \mathtt{Q}_2, \ldots, \mathtt{Q}_5)$ be the reconstructed markers. Denote the distance from $\mathtt{P}_i$ to the face mesh $\mathcal{S}$ by $d(\mathtt{P}_i, \mathcal{S})$. Assume marker $\mathtt{Q}_j$ corresponds to vertex $\mathbf{v}_{m_j}$ of the face mesh, and denote the distance between $\mathtt{Q}_j$ and $\mathbf{v}_{m_j}$ by $d(\mathtt{Q}_j, \mathbf{v}_{m_j})$. The fitting process consists in finding pose $\mathbf{T}$ and metric coefficients $\{c_1, \ldots, c_n\}$ by minimizing

$$\sum_{i=1}^{n} w_i d^2(\mathtt{P}_i, \mathcal{S}) + \sum_{j=1}^{5} d^2(\mathtt{Q}_j, \mathbf{v}_{m_j}) \tag{14}$$

where $w_i$ is a weighting factor.

To solve this problem, we use an iterative closest point approach. At each iteration, we first fix $\mathbf{T}$. For each $\mathtt{P}_i$, we find the closest point $\mathtt{G}_i$ on the current face mesh $\mathcal{S}$. We then minimize $\sum w_i d^2(\mathtt{P}_i, \mathtt{G}_i) + \sum d^2(\mathtt{Q}_j, \mathbf{v}_{m_j})$. We set $w_i$ to be 1 at the first iteration and $1.0/(1 + d^2(\mathtt{P}_i, \mathtt{G}_i))$ in the subsequent iterations. The reason for using weights is that the reconstruction from images is noisy and such a weight scheme is an effective way to avoid overfitting to the noisy data (Fua and Miccio, 1999). Since both $\mathtt{G}_i$ and $\mathbf{v}_{m_j}$ are linear functions of the metric coefficients for fixed $\mathbf{T}$, the above problem is a linear least square problem. We then fix the metric coefficients, and solve for the pose. To do that, we recompute $\mathtt{G}_i$ using the new metric coefficients. Given a set of 3D corresponding points $(\mathtt{P}_i, \mathtt{G}_i)$ and $(\mathtt{Q}_j, \mathbf{v}_{m_j})$, there are well known algorithms to solve for the pose. We use the quaternion-based technique described in (Horn, 1987).

To initialize this iterative process, we first use the 5 markers to compute an initial estimate of the pose. In addition, to get a reasonable estimate of the head size, we solve for the head-size related metric coefficients such that the resulting face mesh matches the bounding box of the reconstructed 3D points. Occasionally the corner matching algorithm may produce points not on the face. In that case, the metric coefficients will be out of the valid ranges, and we throw away the point that is the most distant from the center of the face. We repeat this process until metric coefficients become valid.

### 9.2 Fine Adjustment Using Image Information

After the geometric fitting process, we have now a face mesh that is a close approximation to the real face. To further improve the result, we search for silhouettes and other face features in the images and use them, together with the chin curve if available from the marking step (Sect. 6.1), to refine the face geometry. The general problem of locating silhouettes and face features in images is difficult, and is still a very active research area in computer vision. However, the face mesh that we have obtained provides a good estimate of the locations of the face features, so we only need to perform search in a small region.

We use the snake approach (Kass, Witkin and Terzopoulos, 1988) to compute the silhouettes of the face. The silhouette of the current face mesh is used as the initial estimate. For each point on this piecewise linear curve, we find the maximum gradient location along the normal direction within a small range (10 pixels each side in our implementation). Then we solve for the vertexes (acting as control points) to minimize the total distance between all the points and their corresponding maximum gradient locations. In the case where the user chooses to put 3 markers below the chin (an optional step), the system treats these 3 markers as the silhouette points.

We use a similar approach to find the upper lips.

To find the outer eye corner (not marked), we rotate the current estimate of that eye corner (given by the face mesh) around the marked eye corner by a small angle, and look for the eye boundary using image gradient information. This is repeated for several angles, and the boundary point that is the most distant to the marked corner is chosen as the outer eye corner.

We could also use the snake approach to search for eyebrows. However, our current implementation uses a slightly different approach. Instead of maximizing image gradients across contours, we minimize the average intensity of the image area that is covered by the eyebrow triangles. Again, the vertices of the eyebrows are only allowed to move in a small region bounded by their neighboring vertices. This has worked very robustly in our experiments.

We then use the face features and the image silhouettes (including the chin curve) as constraints in our system to further improve the mesh. Notice that each vertex on the mesh silhouette corresponds to a vertex on the image silhouette. We cast a ray from the camera center through the vertex on the image silhouette. The projection of the corresponding mesh vertex on this ray acts as the target position of the mesh vertex. Let $\mathbf{v}$ be the mesh vertex and $\mathbf{h}$ the projection. We have equation $\mathbf{v} = \mathbf{h}$. For each face feature, we obtain an equation in a similar way. These equations are added to equation (14). The total set of equations is solved as before, i.e., we first fix the pose $\mathbf{T}$ and use a linear least square approach to solve the metric coefficients, and then fix the metric coefficients while solving for the pose.

## 10 Model-Based Bundle Adjustment

As mentioned in Sect. 6.7, we are interested in refining the initial face model as well as the head pose estimates by taking into account all available information from multiple images simultaneously. In this section, we briefly outline our model-based bundle adjustment. The interested reader is encouraged to read (Shan et al., 2001) for details.

### 10.1 Problem Formulation

As described in Sect. 5, a face mesh $\mathcal{S}$ is defined by $m$ metric coefficients $\{c_j | j = 1, \ldots, m\}$ according to (4), where the vertices of the mesh are evaluated in the coordinate system where the neutral mesh and metrics are designated. To complete define the face geometry in the camera coordinate system, we still need to know the head pose $\mathbf{T}$, for example, in the first base image. Thus, the face geometry given by (13), i.e., $\mathcal{S} = \mathbf{T}(\mathcal{S}^0 + \sum_{j=1}^{m} c_j \mathcal{M}^j)$, is the face mesh in the camera coordinate system. Therefore, including the 6 parameters for $\mathbf{T}$, the total number of model parameters is $M = m + 6$, and they are collectively designated by vector $\mathtt{C}$. Let us denote the surface by $\mathcal{S}(\mathtt{C})$.

Furthermore, we assume there are $Q$ semantically meaningful points (*semantic points* for short) $\{\mathtt{Q}_j | j = 1, \ldots, Q\}$ on the face. If there are no semantic points available, then $Q = 0$. In our case, $Q = 5$ because we use five semantic points: two inner eye corners, two mouth corners, and the nose tip. The relationship

between the $j^{\text{th}}$ semantic point $\mathtt{Q}_j$ and the face parameters $\mathtt{C}$ is described by

$$\mathtt{Q}_j = \mathcal{Q}(\mathtt{C}, j) \ . \tag{15}$$

Obviously, point $\mathtt{Q}_j \in \mathcal{S}(\mathtt{C})$.

We are given a set of $N$ images/frames, and a number of points of interest across images have been established as described earlier. Image $i$ is taken by a camera with unknown pose parameters $\mathtt{M}_i$ which describes the position and orientation of the camera with respect to the world coordinate system in which the face mesh is described. A 3D point $\mathtt{P}_k$ and its image point $\mathbf{p}_{i,k}$ are related by (2), i.e., $\mathbf{p}_{i,k} = \boldsymbol{\phi}(\mathtt{M}_i, \mathtt{P}_k)$.

Because of occlusion, feature detection failure and other reasons, a point on the face may be observed and detected in a subset of images. Let us call the set of image points corresponding to a single point on the face a *feature track*. Let $P$ be the total number of feature tracks, $\Theta_k$ be the set of frame numbers of the $k^{\text{th}}$ feature track, $\mathbf{p}_{i,k}$ ($i \in \Theta_k$) be the feature point in the $i^{\text{th}}$ frame that belongs to the $k$th feature track, and $\mathtt{P}_k$ be the corresponding 3D point, which is unknown, on the face surface. Furthermore, we assume that the $j^{\text{th}}$ semantic point $\mathtt{Q}_j$ is observed and detected in zero or more images. Let $\mathbf{T}_j$ be the, possibly empty, set of frame numbers in which $\mathtt{Q}_j$ are detected, and $\mathbf{q}_{l,j}$ ($l \in \Omega_j$) be the detected semantic point in the $l^{\text{th}}$ frame.

We can now state the problem as follows:

**Problem:** *Given $P$ tracks of feature points $\{\mathbf{p}_{i,k} | k = 1, \ldots, P; i \in \Theta_k\}$ and $Q$ tracks of semantic points $\{\mathbf{q}_{l,j} | j = 1, \ldots, Q; l \in \Omega_j\}$, determine the face model parameters $\mathtt{C}$ and the camera pose parameters $\mathtt{M} = [\mathtt{M}_1^T, \ldots, \mathtt{M}_N^T]^T$.*

Our objective is to solve this problem in an optimal way. By *optimal* we mean to find simultaneously the face model parameters and camera parameters by minimizing some statistically and/or physically meaningful cost function. As in the classical point-based bundle adjustment, it is reasonable to assume that the image points are corrupted by independent and identically distributed Gaussian noise because points are extracted independently from images by the same algorithm. In that case, the maximum likelihood estimation is obtained by minimizing the sum of squared errors between the observed image points and the predicted feature points. More formally, the problem becomes

$$\begin{aligned} \min_{\mathtt{M},\mathtt{C},\{\mathtt{P}_k\}} & \left( \sum_{k=1}^{P} \sum_{i \in \Theta_k} \|\mathbf{p}_{i,k} - \boldsymbol{\phi}(\mathtt{M}_i, \mathtt{P}_k)\|^2 + \sum_{j=1}^{Q} \sum_{l \in \Omega_j} \|\mathbf{q}_{l,j} - \boldsymbol{\phi}(\mathtt{M}_l, \mathtt{Q}_j)\|^2 \right) \\ & \text{subject to } \mathtt{P}_k \in \mathcal{S}(\mathtt{C}), \end{aligned} \tag{16}$$

where $\mathtt{Q}_j = \mathcal{Q}(\mathtt{C}, j)$ as defined in (15). Note that although the part for the general feature points (the first term) and the part for the semantic points (the second term) have the same form, we should treat them differently. Indeed, the latter provides stronger constraint in bundle adjustment than the former. We can simply substitute $\mathtt{Q}_j$ in the second part by $\mathcal{Q}(\mathtt{C}, j)$ while $\mathtt{P}_k$ must be searched on the surface $\mathcal{S}(\mathtt{C})$.

We observe that in (16), unknown 3D points $\{\mathtt{P}_k\}$, which correspond to feature tracks, are not involved at all in the second term. Furthermore, in the first term, the second summation only depends on each individual $\mathtt{P}_k$. We can therefore rewrite (16) as

$$\begin{aligned} \min_{\mathtt{M},\mathtt{C}} & \left( \sum_{k=1}^{P} \Big( \min_{\mathtt{P}_k} \sum_{i \in \Theta_k} \|\mathbf{p}_{i,k} - \boldsymbol{\phi}(\mathtt{M}_i, \mathtt{P}_k)\|^2 \Big) + \sum_{j=1}^{Q} \sum_{l \in \Omega_j} \|\mathbf{q}_{l,j} - \boldsymbol{\phi}(\mathtt{M}_l, \mathtt{Q}_j)\|^2 \right) \\ & \text{subject to } \mathtt{P}_k \in \mathcal{S}(\mathtt{C}). \end{aligned} \tag{17}$$

This property is reflected in the sparse structure of the Jacobian and Hessian of the objective function. As in the classical bundle adjustment, exploiting the sparse structure leads a much more efficient implementation. In (Shan et al., 2001), we use a first order approximation to eliminate the structure parameters $\{\mathtt{P}_k\}$, thus resulting in a much smaller minimization problem.

In most practical problems, not all possible values of parameters $\mathtt{C}$ are acceptable, and it is often necessary or desirable to impose constraints. There are many forms of constraints: linear or nonlinear, equality or inequality. The reader is referred to (Gill, Murray and Wright, 1981) for various techniques to deal with constrained optimization problems. An important case is when a parameter $c_m$ is subject to bounds:

$l_m \leq c_m \leq u_m$. For each such constraint, we add to (16) two penalty terms for the lower and upper bound. For the lower bound, the penalty term is defined by

$$p_m = \begin{cases} 0 & \text{if } c_m \geq l_m; \\ \rho(l_m - c_m)^2 & \text{otherwise,} \end{cases}$$

where the non-negative value $\rho$ is the penalty parameter.

### 10.2 Comparison Between CBA and MBA

Compared with the classical point-based bundle adjustment (CBA), our model-based bundle adjustment (16) (MBA) has a similar form except that we have model parameters $\mathtt{C}$ and that points $\mathtt{P}_k$ are constrained on $\mathcal{S}(\mathtt{C})$. In CBA, there are no model parameters but points $\mathtt{P}_k$ are free. Although it appears that MBA has more parameters to estimate, the real number of free parameters is usually smaller because of the constraint on points $\mathtt{P}_k$. Indeed, the total number of free parameters in CBA is equal to $6(N-1)-1+3P$ ("$-1$" is due to the fact that the global scale cannot be determined), while the total number of free parameters in MBA is equal to $6(N-1)-1+M+2P$ because each point on a surface has two degrees of freedom. As long as $P > M$ (the number of feature tracks is larger than that of model parameters), the parameter space for MBA is smaller than for CBA. In our typical setup, $P > 1500$ while $M = 71$.

### 10.3 Experimental Results

The reader is referred to (Shan et al., 2001) for a comparison between our MBA algorithm and the classical bundle adjustment. In this section, we only show how the MBA improves the initial face models.

A typical sequence contains 23 to 26 images of resolution 640x480. The number of feature tracks ranges from 1500 to 2400. There are 50 to 150 image matches between each pair of neighboring views. For each sequence, the total running time of the MBA algorithm is about 6 to 8 minutes on a 850MHz Pentium III machine. We have already seen an example in Section 6. Two more examples are shown below. For each example, we show both the initial guesses (results from the rapid face modeling system) and the final results from the MBA algorithm.

The first example is shown in Fig. 21. The left column is the initial guess. The right column is the result from the MBA algorithm. The images in the middle are the acquired images. We can see that the face of the initial guess is too narrow compared to the actual face, and there is a clear improvement with the result from the MBA algorithm.

The second example is shown in Fig. 22. We can see that the profile of the initial guess is quite different from the actual person. With MBA, the profile closely matches the profile of the actual person.

## 11 More Experimental Results

We have constructed 3D face models for well over two hundred people. We have done live demonstrations at ACM Multimedia 2000, ACM1, CHI2001, ICCV2001 and other events such as the 20th anniversary of the PC, where we set up a booth to construct face models for visitors. At each of these events, the success rate is 90% or higher. In ACM1, most of the visitors are kids or teenagers. Kids are usually more difficult to model since they have smooth skins, but our system worked very well. We observe that the main factor for the occasional failure is the head turning too fast.

We should point out that in our live demonstrations, the optional model-based bundle adjustment is not conducted because it is quite time consuming (about 6 to 8 minutes on a 850MHz Pentium III machine). Without that step, our system takes about one minute after data-capture and manual marking to generate a textured face model. Most of this time is spent on head tracking in the video sequences. All the results shown in this section were produced in this way.

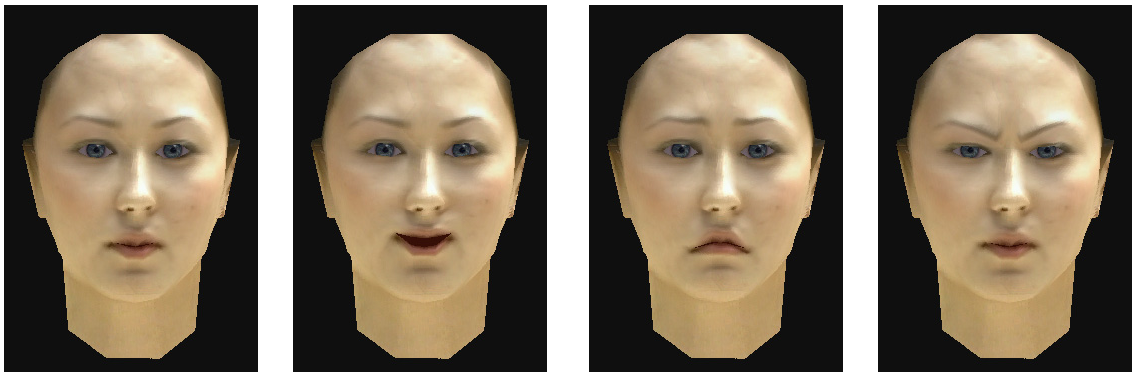
Figure 23 shows side-by-side comparisons of eight reconstructed models with the real images. Figure 24 shows the reconstructed face models of our group members immersed in a virtualized environment. In these examples, the video sequences were taken using ordinary video camera in people's offices or in live demonstrations. No special lighting equipment or background was used.

Figure 21: First textured face model. Left: initial guess; Middle: original images; Right: MBA.

Figure 22: Second textured face model. Left: initial guess; Middle: original images; Right: MBA.

Figure 23: Side by side comparison of the original images with the reconstructed models of various people in various environment settings.


Figure 24: Face models of our group members in a virtualized environment.



Neutral Happy Sad Thinking

Figure 25: Examples of facial animation with an animated face model built with our system.

### 11.1 Applications: Animation and Interactive Games

Having obtained the 3D textured face model, the user can immediately animate the model with the application of facial expressions including smiles, sad, thinking, etc. The model can also perform text to speech.

To accomplish this we have defined a set of vectors, which we call *posemes*. Like the metric vectors described previously, posemes are a collection of artist-designed displacements, corresponding approximately to the widely used action units in the Facial Action Coding System (FACTS) (Ekman and Friesen, 1978). We can apply these displacements to any face as long as it has the same topology as the neutral face. Posemes are collected in a library of actions, expressions, and visems.

Figure 25 shows a few facial expressions which can be generated with our animated face models. Note that the facial expressions shown here are the results of geometric warping, namely, the texture image is warped according to the desired displacement of the vertices. Facial expressions, however, exhibit many detailed image variations due to facial deformation. A simple expression mapping technique based on ratio images is described in (Liu, Shan and Zhang, 2001) which can generate vivid facial expression.

An important application of face modeling is interactive games. You can import your personalized face model in the games so that you are controlling the "visualized you" and your friends are seeing the "visualized you" play. This would dramatically enhance the role-playing experience offered in many games. Figure 26 shows a snapshot of an online poker game, developed by our colleague Alex Colburn, where the individualized face models are built with our system.

## 12 Conclusions and Future Work

We have developed a system to construct textured 3D face models from video sequences with minimal user intervention. With a few simple clicks by the user, our system quickly generates a person's face model which is animated right away. Our experiments show that our system is able to generate face models for people of different races, of different ages, and with different skin colors. Such a system can be potentially used by an ordinary user at home to make their own face models. These face models can be used, for example, as avatars in computer games, online chatting, virtual conferencing, etc.

Besides use of many state-of-the-art computer vision and graphics techniques, we have developed several innovative techniques including intelligent masking, robust head pose determination, low-dimensional linear face model fitting, and model-based bundle adjustment. By following the model-based modeling approach, we have been able to develop a robust and efficient modeling system for a very difficult class of objects, namely, faces.

Several researchers in computer vision are working at automatically locating facial features in images (Shakunaga, Ogawa and Oki, 1998). With the advancement of those techniques, a completely automatic face modeling system can be expected, even though it is not a burden to click just five points with

Figure 26: An online poker game with individualized face models.

our current system.

The current face mesh is very sparse. We are investigating techniques to increase the mesh resolution by using higher resolution face metrics or prototypes. Another possibility is to compute a displacement map for each triangle using color information.

Additional challenges include automatic generation of eyeballs and eye texture maps, as well as accurate incorporation of hair, teeth, and tongues. For people with hair on the sides or the front of the face, our system will sometimes pick up corner points on the hair and treat them as points on the face. The reconstructed model may be affected by them. Our system treats the points on the hair as normal points on the face.

The animation of the face models is pre-designed in our system. In many other applications, it is desirable to understand the facial expression in a real video sequence and use it to drive the facial animation. Some work has been done in that direction (Essa and Pentland, 1997; Black and Yacoob, 1997; Guenter et al., 1998; Tao and Huang, 1999; Tian, Kanade and Cohn, 2001).

## Acknowledgment

The authors would like to thank P. Anandan, R. Rashid, R. Szeliski, and Turner Whitted for their support and stimulating discussions in developing this system. They also like to thank James Mahoney for designing the generic face model and metrics database, and A. Soupliotis, A. Colburn, and many others for testing the system. The preliminary version of the system was reported in (Liu, Zhang, Jacobs and Cohen, 2000), and Chuck Jacobs contributed to that development. The robust head motion algorithm was reported in (Liu and Zhang, 2001).

## References

Akimoto, T., Suenaga, Y. and Wallace, R. S. (1993). Automatic 3d facial models, *IEEE Computer Graphics and Applications* **13**(5): 16–22.

Barron, J., Fleet, D. and Beauchemin, S. (1994). Performance of optical flow techniques, *The International Journal of Computer Vision* **12**(1): 43–77.

Black, M. and Yacoob, Y. (1997). Recognizing facial expressions in image sequences using local parameterized models of image motion, *The International Journal of Computer Vision* **25**(1): 23–48.

Blanz, V. and Vetter, T. (1999). A morphable model for the synthesis of 3d faces, *Computer Graphics, Annual Conference Series*, Siggraph, pp. 187–194.

Dariush, B., Kang, S. B. and Waters, K. (1998). Spatiotemporal analysis of face profiles: Detection, segmentation, and registration, *Proc. of the 3rd International Conference on Automatic Face and Gesture Recognition*, IEEE, pp. 248–253.

DeCarlo, D., Metaxas, D. and Stone, M. (1998). An anthropometric face model using variational techniques, *Computer Graphics, Annual Conference Series*, Siggraph, pp. 67–74.

DiPaola, S. (1991). Extending the range of facial types, *Journal of Visualization and Computer Animation* **2**(4): 129–131.

Ekman, P. and Friesen, W. (1978). *The Facial Action Coding System: A Technique for The Measurement of Facial Movement*, Consulting Psychologists Press, San Francisco.

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.

Faugeras, O. (1993). *Three-Dimensional Computer Vision: a Geometric Viewpoint*, MIT Press.

Fua, P. (2000). Regularized bundle-adjustment to model heads from image sequences without calibration data, *The International Journal of Computer Vision* **38**(2): 153–171.

Fua, P. and Miccio, C. (1998). From regular images to animated heads: A least squares approach, *European Conference on Computer Vision*, pp. 188–202.

Fua, P. and Miccio, C. (1999). Animated heads from ordinary images: A least-squares approach, *Computer Vision and Image Understanding* **75**(3): 247–259.

Fua, P., Plaenkers, R. and Thalmann, D. (1999). From synthesis to analysis: Fitting human animation models to image data, *Computer Graphics International*, Alberta, Canada.

Gill, P. E., Murray, W. and Wright, M. H. (1981). *Practical Optimization*, Academic Press.

Guenter, B., Grimm, C., Wood, D., Malvar, H. and Pighin, F. (1998). Making faces, *Computer Graphics, Annual Conference Series*, Siggraph, pp. 55–66.

Harris, C. and Stephens, M. (1988). A combined corner and edge detector, *Proc. 4th Alvey Vision Conf.*, pp. 189–192.

Horn, B. K. (1987). Closed-form Solution of Absolute Orientation using Unit Quaternions, *Journal of the Optical Society A* **4**(4): 629–642.

Horn, B. K. P. and Schunk, B. G. (1981). Determining Optical Flow, *Artificial Intelligence* **17**: 185–203.

Ip, H. H. and Yin, L. (1996). Constructing a 3d individualized head model from two orthogonal views, *The Visual Computer* (12): 254–266.

Kang, S. B. and Jones, M. (1999). Appearance-based structure from motion using linear classes of 3-d models, *Manuscript*.

Kass, M., Witkin, A. and Terzopoulos, D. (1988). SNAKES: Active contour models, *The International Journal of Computer Vision* **1**: 321–332.

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

Lee, W. and Magnenat-Thalmann, N. (1998). Head modeling from photographs and morphing in 3d with image metamorphosis based on triangulation, *Proc. CAPTECH'98*, Springer LNAI and LNCS Press, Geneva, pp. 254–267.

Lee, Y. C., Terzopoulos, D. and Waters, K. (1993). Constructing physics-based facial models of individuals, *Proceedings of Graphics Interface*, pp. 1–8.

Lee, Y. C., Terzopoulos, D. and Waters, K. (1995). Realistic modeling for facial animation, *Computer Graphics, Annual Conference Series*, SIGGRAPH, pp. 55–62.

Lewis, J. P. (1989). Algorithms for solid noise synthesis, *Computer Graphics, Annual Conference Series*, Siggraph, pp. 263–270.

Liu, Z. and Zhang, Z. (2001). Robust head motion computation by taking advantage of physical properties, *Proceedings of the IEEE Workshop on Human Motion (HUMO 2000)*, Austin, USA, pp. 73–77.

Liu, Z., Shan, Y. and Zhang, Z. (2001). Expressive expression mapping with ratio images, *Computer Graphics, Annual Conference Series*, ACM SIGGRAPH, Los Angeles, pp. 271–276.

Liu, Z., Zhang, Z., Jacobs, C. and Cohen, M. (2000). Rapid modeling of animated faces from video, *Proc. 3rd International Conference on Visual Computing*, Mexico City, pp. 58–67. Also in the special issue of *The Journal of Visualization and Computer Animation*, Vol.12, 2001. Also available as MSR technical report from http://research.microsoft.com/˜zhang/Papers/TR00-11.pdf.

Magneneat-Thalmann, N., Minh, H., Angelis, M. and Thalmann, D. (1989). Design, transformation and animation of human faces, *Visual Computer* (5): 32–39.

More, J. (1977). The levenberg-marquardt algorithm, implementation and theory, *in* G. A. Watson (ed.), *Numerical Analysis*, Lecture Notes in Mathematics 630, Springer-Verlag.

Parke, F. I. (1972). Computer generated animation of faces, *ACM National Conference*.

Parke, F. I. (1974). *A Parametric Model of human Faces*, PhD thesis, University of Utah.

Parke, F. I. and Waters, K. (1996). *Computer Facial Animation*, AKPeters, Wellesley, Massachusetts.

Pighin, F., Hecker, J., Lischinski, D., Szeliski, R. and Salesin, D. H. (1998). Synthesizing realistic facial expressions from photographs, *Computer Graphics, Annual Conference Series*, Siggraph, pp. 75–84.

Platt, S. and Badler, N. (1981). Animating facial expression, *Computer Graphics* **15**(3): 245–252.

Rousseeuw, P. and Leroy, A. (1987). *Robust Regression and Outlier Detection*, John Wiley & Sons, New York.

Shakunaga, T., Ogawa, K. and Oki, S. (1998). Integration of eigentemplate and structure matching for automatic facial feature detection, *Proc. of the 3rd International Conference on Automatic Face and Gesture Recognition*, pp. 94–99.

Shan, Y., Liu, Z. and Zhang, Z. (2001). Model-based bundle adjustment with application to face modeling, *Proceedings of the 8th International Conference on Computer Vision*, Vol. II, IEEE Computer Society Press, Vancouver, Canada, pp. 644–651.

Tao, H. and Huang, T. (1999). Explanation-based facial motion tracking using a piecewise bezier volume deformation model, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Vol. I, IEEE Computer Society, Colorado, pp. 611–617.

Terzopoulos, D. and Waters, K. (1990). Physically based facial modeling, analysis, and animation, *Visualization and Computer Animation*, pp. 73–80.

Tian, Y.-L., Kanade, T. and Cohn, J. (2001). Recognizing action units for facial expression analysis, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **23**(2): 97–115.

Todd, J. T., Leonard, S. M., Shaw, R. E. and Pittenger, J. B. (1980). The perception of human growth, *Scientific American* (1242): 106–114.

Vetter, T. and Poggio, T. (1997). Linear object classes and image synthesis from a single example image, *IEEE Transations on Pattern Analysis and Machine Intelligence* **19**(7): 733–742.

Waters, K. (1987). A muscle model for animating three-dimensional facial expression, *Computer Graphics* **22**(4): 17–24.

Zhang, Z. (1997). Motion and structure from two perspective views: From essential parameters to euclidean motion via fundamental matrix, *Journal of the Optical Society of America A* **14**(11): 2938–2950.

Zhang, Z. (1998a). Determining the epipolar geometry and its uncertainty: A review, *The International Journal of Computer Vision* **27**(2): 161–195.

Zhang, Z. (1998b). On the optimization criteria used in two-view motion analysis, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **20**(7): 717–729.

Zhang, Z. (2000). A flexible new technique for camera calibration, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**(11): 1330–1334.

Zheng, J. Y. (1994). Acquiring 3-d models from sequences of contours, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **16**(2): 163–178.