Animated Heads from Ordinary Images: A Least Squares Approach

P. Fua and C. Miccio
Computer Graphics Lab (LIG)
EPFL
CH-1015 Lausanne
Switzerland
pascal.fua@epfl.ch

To Appear in CVIU

Abstract

We show that we can effectively fit arbitrarily complex animation models to noisy data extracted from ordinary face images Our approach is based on least-squares adjustment, using of a set of progressively finer control triangulations and takes advantage of three complementary sources of information: stereo data, silhouette edges and 2–D feature points.

In this way, complete head models—including ears and hair—can be acquired with a cheap and entirely passive sensor, such as an ordinary video camera. They can then be fed to existing animation software to produce synthetic sequences.

1 Introduction

In this paper, we show that we can effectively fit a complex head animation model, including ears and hair, to images taken with simple video or CCD cameras. We use a completely passive technique, stereo, as our main source of information.

In recent years much work has been devoted to modeling faces from image and range data. There are many effective approaches to recovering face geometry. They rely on stereo [5], shading [18], structured light [25], silhouettes [27] or low-intensity lasers. However, recovering a head as a simple triangulated mesh does not suffice: To animate the face, one must further fit an actual animation model to the data.

To be suitable for animation, these models must have a large number of degrees of freedom. Some approaches use very clean data—the kind produced by a laser scanner or structured light—to instantiate them [20]. Among approaches that rely on image data alone, many require extensive manual intervention, such as supplying silhouettes in orthogonal images [19] or point correspondences in multiple images [22]. The most successful recent approaches to automating the fitting process have involved the use of optical flow [4] or appearance based techniques [16] to overcome the fact that faces have little texture and that, as a result, automatically and reliably establishing correspondences is difficult. Such techniques are best adapted to processing images taken from very similar viewpoints, such as consecutive frames in a video sequence, and make assumptions about constant illumination of the face that may

be violated as the head moves. This tends to limit the range of images that can be used, especially if the lighting is not diffuse.

In this paper, we propose using stereo instead as our main source of shape information. We will show that, given pairs, triplets or video sequences, we can robustly produce high quality and realistic face models that can then be animated. This versatility could be exploited to model faces from many different sources of imagery. The accuracy of the models increases with the number of images used. In the case of video sequences, the kind of stereo data we need can now be computed at near frame rate on ordinary computers [17]. The required manual intervention reduces to supplying the location of 5 key 2–D feature points in one image. Optionally, we can also supply additional 2–D feature points and take advantage of silhouette information which is helpful to model parts of the head that slope away from the camera planes or for which reliable stereo information can not be obtained, such as hair.

We typically start with a set of stereo image pairs or a video sequence and use a correlation-based algorithm [7] to compute disparity maps. We then derive clouds of 3–D points and fit the animation mask to these. In this work, we assume that the images are registered and that camera models are available. This assumption is reasonable because there are well established photogrammetric techniques, such as bundle-adjustment, that allow the computation of these models as needed. Furthermore, recent work in the area of autocalibration [6, 29, 23] or pose estimation [26, 14, 9] can be brought to bear to automate the process. In this paper, we take advantage of the latter.

Our contribution is twofold:

- Because we use robust fitting techniques and take advantage of our rough knowledge of a head's shape, we obtain reliable results even from noisy data acquired with a cheap and entirely passive technique.

  In particular, we will show that convincing, metrically accurate models can be constructed as well from sequences acquired with a simple hand-held video camera as from stereo pairs and triplets.

  Admittedly, the pairwise correlation technique we use could be improved, for example by matching over longer sequences of images when possible. However, its very simplicity also makes our approach very flexible and nothing precludes us from using better range data if it is available.

- The least-squares framework we have developed allows us to pool heterogeneous information sources—stereo or range, silhouettes and 2–D feature locations—without deterioration in the convergence properties of the algorithm. This is in contrast to typical optimization approaches whose convergence properties tend to degrade when using an objective function that is the sum of many incommensurate terms [13, 11].

Therefore, our approach derives convincing models from a wide variety of images: stereo pairs, triplets and video sequences.

The rest of the paper is organized as follows: We sketch out the approach, introduce our optimization framework, and discuss how the various sources of information are handled. Finally we present reconstruction and animation results on a number of different heads.

2 Approach

In this work, we use the facial animation model depicted by Figure 1. It has been developed at University of Geneva and EPFL [15]. It can produce the different facial expressions arising from speech and

emotions. Its hierarchical configuration reduces complexity and provides independent control for each level. At the lowest one, a deformation controller simulates muscle actions using rational free form deformations. At a higher level, the controller produces animations corresponding to abstract entities such as speech and emotions.

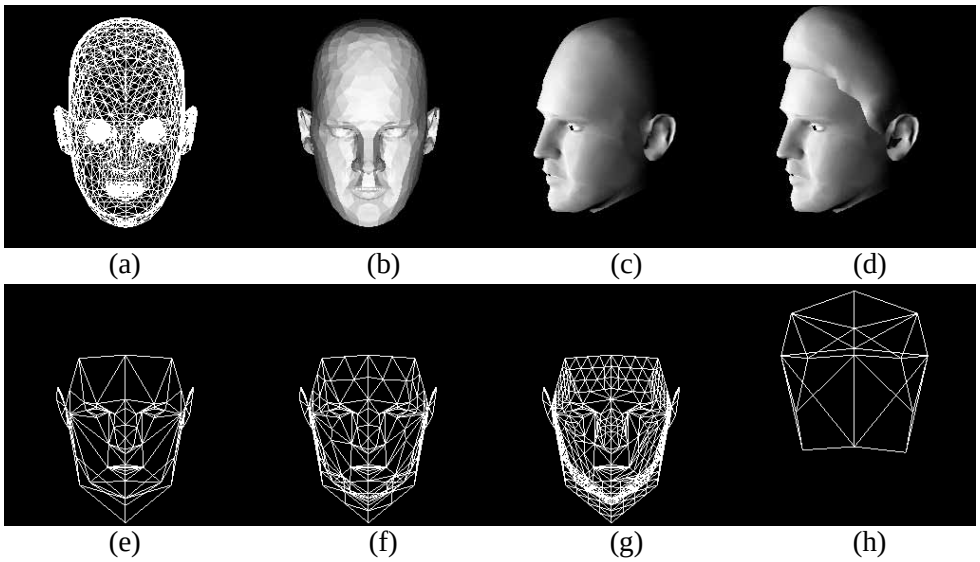


Figure 1: Animation model. (a) Model used to animate complete heads seen as a wireframe. (b,c) Shaded views. (d) The same model with an additional triangulated surface that represents the hair. (e,f,g) The three levels of refinement of the control mesh used to deform the face surface. (h) The control mesh used to deform the hair surface.

Given a set of registered images, our fitting procedure goes through the following steps.

- We compute disparity maps for each stereo pair or each consecutive pair in the video sequences, fit local surface patches to the corresponding 3–D points, and use these patches to compute a central 3–D point and a normal vector. Optionally, we also use manual or semi-automated techniques to extract silhouettes and a small number of feature points.

- We attach a coarse control mesh to the face of the animation model and perform a least squares adjustment of this control mesh so that the model matches the previously computed data. We weight the data points according to how close—in the least squares sense—they are to the model and use an iterative reweighting technique to eliminate the outliers. We then subdivide the control mesh and repeat the procedure to refine the result.

- We repeat the same procedure for the hair. Because hair is where stereo tends to fail, we typically rely more heavily on the use of silhouettes to derive an estimate of the shape than we do for the face.

- We use the original images to compute a cylindrical texture map that allows realistic rendering. This is achieved by first generating a cylindrical projection of the head model and then, for each projected point, finding the images in which it is visible and averaging the corresponding gray-levels.

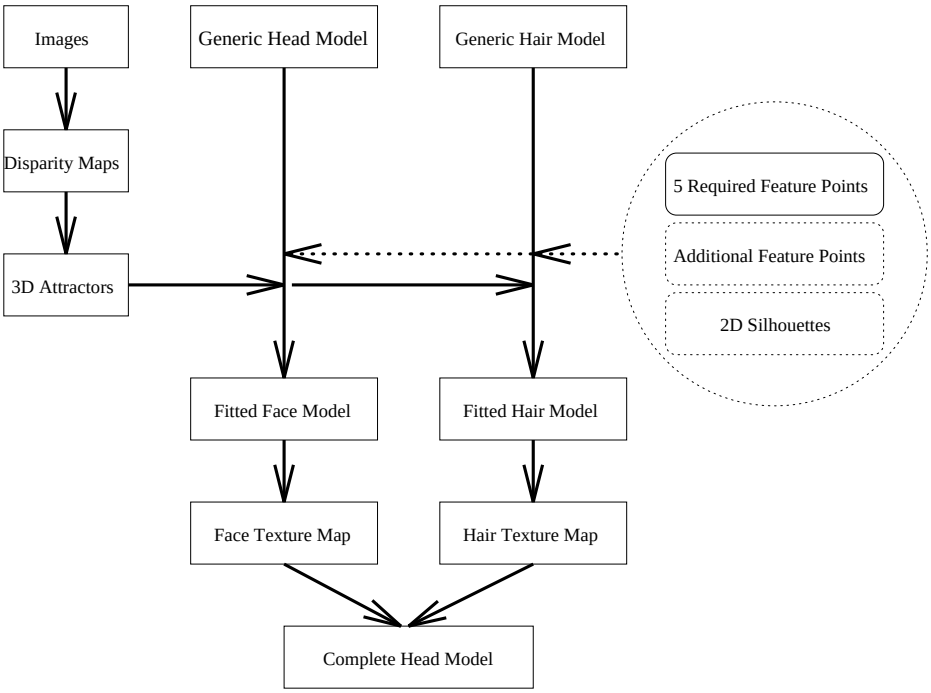
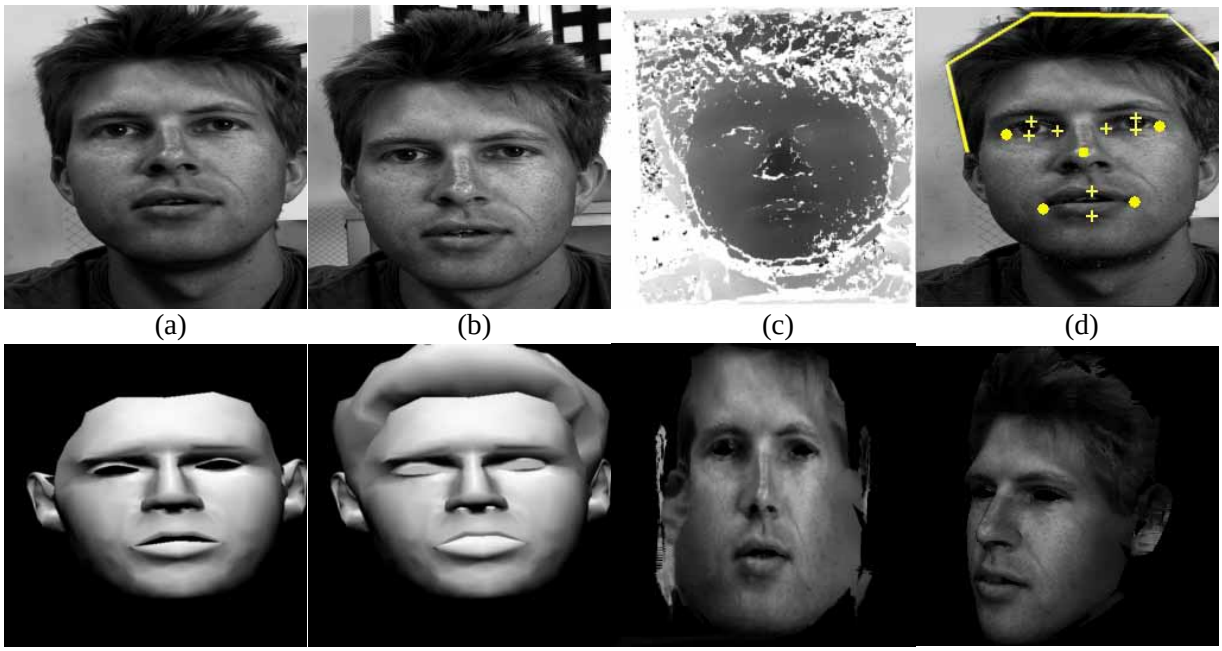


Figure 2: Face Reconstruction Procedure: The manually and semi-automatically entered data appears in the the circle. The location of 5 2-D points must be supplied, the rest is optional.

These steps are depicted by the flowchart of Figure 2 and illustrated by Figure 3. The only manual intervention that is mandatory is supplying the location in one single image of the five feature points—corners of the eyes and mouth and tip of the nose—shown in Figure 3(c) as circles. These are then matched in the other images to compute a 3–D prism that is used to position the animation mask prior to fitting. This estimate need not be very precise because we start the fitting process by computing the rigid motion that best aligns the model to the data.

In order to ensure proper animation, it is important to guarantee that important features of the model—mouth and corners of the eyes especially—project at the correct locations in the face. The system is therefore set up so that we have the option to supply the 2–D location of such points, shown as crosses in Figure 3(c), as well as a number of 2–D silhouettes.

The silhouettes are entered using either snakes or dynamic programming. As far as the 2-D points are concerned, we have implemented a simple tool that allows us to enter the 2-D location of a number of selected points—4 for each eye, 4 for the mouth, 3 for the hairline, 4 for the nose and 4 for each ear. Of course, not all these points need to be specified. One mode of operation is to run the system without them, and then to add as many of these points as needed to achieve the desired quality. For example, in the case of Figure 3, we did not supply the hairline feature points. As a result, the length of the model's forehead relative to the rest of the face remains the same as that of the generic model, which, in this case, is acceptable. If it were not, we could supply the missing points and rerun the fitting process. Because only the 2-D location of these points need to be specified, this can be done very quickly. Furthermore, there now exists a number of automated techniques that could supply some of those locations [2, 3, 14] without manual intervention.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 3: Modeling the complete head. (a,b) A stereo pair. (c) Horizontal disparity map computed by rectifying the images and running a correlation based algorithm. (d) Hand entered feature points and silhouettes. The required points are shown as circles while the optional ones appear as crosses. (e) Reconstructed face. (f) Hair optimized to conform to the outlines of (d). (g) Cylindrical texture map. (h) Texture mapped view.

3 Least Squares Framework

In this section, we introduce the framework we have developed to fit the generic animation model of Figure 1 to noisy image data. This model embodies a rough knowledge about the head's shape and can be used to constrain the search space.

We will refer to the skin surface, shown in its rest position in Figure 1(a), as the *surface triangulation*. Our goal is to deform the surface—without changing its topology, that is the connectivity of its vertices—so that it conforms to the image data. In standard least-squares fashion, we will use this data to write $nobs$ observation equations of the form

$$f_i(P) = obs_i + \epsilon_i \quad , 1 \leq i \leq nobs \quad , \tag{1}$$

where P is a parameter vector that defines the shape of the surface and ϵ_i is the deviation from the model. We will then minimize

$$\sum_{1 \leq i \leq nobs} w_i \epsilon_i^2 \quad , \tag{2}$$

where w_i is a weight associated to each observation. The optimization is then performed using the Levenberg-Marquardt algorithm [24].

In theory we could take the parameter vector P to be the vector of all x, y, and z coordinates of the surface triangulation. However, because the image data is very noisy, we would have to impose a very strong regularization constraint. For example, we have tried to treat the surface triangulation as finite

element mesh. Due to its great irregularity and its large number of vertices, we have found the fitting process to be very brittle and the smoothing coefficients difficult to adjust. Therefore, we propose the following scheme to achieve robustness.

3.1 Control Triangulation

Instead of directly modifying the vertex positions during the minimization, we introduce *control triangulations* such as the ones shown in Figure 1(e,f,g). The vertices of the surface triangulation are "attached" to the control triangulation and the range of allowable deformations of the surface triangulation is defined in terms of weighted averages of displacements of the vertices of the control triangulation. The triangulation can thus be deformed to fit image data and to produce results such as those shown in Section 5. This control triangulation reduces the number of degrees of freedom. It plays a role somewhat similar to the spline in the appearance based technique discussed in the introduction [16, 26]. However, it does not have to be completely regular.

More specifically, we project each vertex of the surface triangulation onto the control triangulation. If this projection falls in the middle of a control facet, we "attach" the vertex to the three vertices of the control facets and compute the corresponding barycentric coordinates. If this projection falls between two facets, we "attach" the vertex to the vertices of the corresponding edge. In effect, we take one of the barycentric coordinates to be zero.

Given these attachments, the surface triangulation's shape is defined by deformation vectors associated with the vertices of the control triangulation. The 3–D position P_i of vertex i of the surface triangulation is taken to be

$$P_i = P_i^0 + l_1^i \delta_{j1} + l_2^i \delta_{j2} + l_3^i \delta_{j3} \quad , \tag{3}$$

where P_i^0 is its initial position, $\delta_{j1}, \delta_{j2}, \delta_{j3}$ are the deformation vectors associated to the control triangulation vertices to which vertex i is attached, and l_1^i, l_2^i, l_3^i are the precomputed barycentric coordinates.

In this fashion, the shape of the surface triangulation becomes a function of the δ_j and the parameter vector P of Equation 1 is taken to be the vector of the x, y and z components of these δ_j. Because the control triangulations have fewer vertices that are more regularly spaced than the surface triangulation, the least-squares optimization has better convergence properties. Of course, the finer the control triangulation, the less smoothing it provides. By using a precomputed set of increasingly refined control triangulations, we implement a hierarchical fitting scheme that has proved very useful when dealing with noisy data, as shown in Section 4. The bottom row of Figure 1 shows the three levels of control mesh refinement used to deform the face and the coarsest control mesh used to deform the hair or skull.

3.2 Stiffness Matrix and Symmetry

Because there may be gaps in the image data, it is necessary to add a small stiffness term into the optimization to ensure that the δ_j of the control vertices located where there is little or no data are consistent with their neighbors. If the surface was continuous, we could take this term to be

$$\iint \left(\frac{\partial}{\partial u} \delta(u,v) \right)^2 + \left(\frac{\partial}{\partial v} \delta(u,v) \right)^2 du\, dv \quad .$$

However, because our control triangulation is discrete, we can treat its facets as C^0 finite elements and write our stiffness term as

$$E_S = \Delta_x^t K \Delta_x + \Delta_y^t K \Delta_y + \Delta_z^t K \Delta_z \tag{4}$$

where K is a stiffness matrix and Δ_x, Δ_y and Δ_z are the vectors of the x, y and z coordinates of the displacements δ. The term we actually optimize becomes

$$E = \sum_{1 \leq i \leq nobs} w_i \epsilon_i^2 + \lambda_S E_S \ , \tag{5}$$

where λ_S is a small positive constant. This is achieved very simply in the least squares framework by incrementing the appropriate elements of the matrix that appears in the normal equations by those of the stiffness matrix K.

Because there is no guarantee that the image data covers equally both sides of the head, we also add a small number of symmetry observations between control vertices on both sides of the face. They serve the same purpose as the stiffness term: Where there is no data, the shape is derived by symmetry. An alternative would have been to use a completely symmetric model with half of the degrees of freedom of the one we use. We chose not to do so because, in reality, faces are somewhat asymmetric.

3.3 Weighting the Observations

As will be discussed in Section 4, our system must be able to deal with many observations coming from different sources that may not be commensurate with each other. Formally we can rewrite the observations equations of Equation 1 as

$$f_i^{type}(P) = obs_i^{type} + \epsilon_i \ \ , 1 \leq i \leq nobs \ \ , \tag{6}$$

with weight w_i^{type}, where $type$ is one of the possible types of observations we use. In this paper, $type$ may be stereo, silhouette position or 2–D feature location. The least-squares minimization procedure involves iteratively solving normal equations and solving linear systems of the form

$$A^t A X = A^t Y$$

where the A matrix is formed by summing the elements of the gradient vectors $w_i^{type} \nabla f_i^{type}(P)$, Y is the vector of all $f_i^{type}(P) - obs_i^{type}$ and X is the unknown update vector that is added to the state vector at each iteration. To ensure that these matrices are well conditioned and that the minimization proceeds smoothly we multiply the weight w_i^{type} of the n_{type} individual observations of a given type by a global coefficient w_{type} computed as follows:

$$\begin{aligned} G_{type} &= \frac{\sqrt{\sum_{1 \leq i \leq nobs, type} w_i^{type} \|\nabla f_i^{type}(P)\|^2}}{n_{type}} \\ w_{type} &= \frac{\lambda_{type}}{G_{type}} \end{aligned} \tag{7}$$

where λ_{type} is a user supplied coefficient between 0 and 1 that indicates the relative importance of the various kinds of observations. This guarantees that, initially at least, the magnitudes of the gradient terms for the various types have the appropriate relative values.

As shown in Section 5, the final result is relatively insensitive to the exact values of the λ_{type} and we have used the same set of values to generate all of our results.

4 From Image Data to Observations

In this section, we show how the raw image data—stereo, silhouettes and 2–D feature locations, can be turned into the observations of Section 3.

4.1 Stereo Data

We use several sets of stereo pairs or triplets of a given face as our input data such as those of Figure 4. We start the process by using a simple correlation-based algorithm [7] to compute a disparity map for each pair or triplet and by turning each valid disparity value into a 3–D point. When using many stereo pairs, as in the the case of a video sequence, this could result in a very large number of such 3–D points that form a noisy and irregular sampling of the underlying global 3–D surface.

To reduce the number of points and the complexity of the subsequent computation, we replace these raw 3–D points by a smaller number of *attractors* computed as follows: We pick spatial step sizes δ_x,δ_y and δ_z along the X,Y and Z axes of an absolute referential. We use them to define a cube-shaped set of 3–D buckets. We then store the raw 3–D points into the appropriate buckets. By robustly fitting a local surface to every bucket containing enough points, we generate patches whose center is the projection of the bucket's center onto the surface and whose orientation is given by the surface's normal at that point. In the presence of very noisy data, the projection may fall outside of the bucket. In this case, we reject the patch, thereby, ensuring that there is only one patch per bucket and that the patches are regularly distributed. The right most column of Figure 4(d,h,l) depicts the result of this procedure. For additional details, we refer the interested reader to an earlier publication [8]. In effect, this procedure reduces the complexity of the data and, in Section 5, we will show that this has no adverse effects on the quality of the reconstructions.

The center of each patch can then be treated as an attractor. The easiest way to handle this is to model it as a spring attached to the mesh vertex closest to it. This, however, is inadequate if one wishes to use facets that are large enough so that attracting the vertices, as opposed to the surface point closest to the attractor, would cause unwarranted deformations of the mesh. This is especially important when using a sparse set of attractors. In our implementation, this is achieved by writing the observation equation as

$$d_i^a = 0 + \epsilon_i \quad , \tag{8}$$

where d_i^a is the orthogonal distance of the attractor to the closest facet, whose nominal value is zero. It can be computed as a function of the x,y, and z coordinates of the vertices of the facet closest to the attractor.

The normal vector to a facet can be computed as the normalized cross product of the vectors defined by two sides of that facet, and d_i^a as the dot product of this normal vector with the vector defined by one of the vertices and the attractor. Letting $(x_i, y_i, z_i)_{1 \leq i \leq 3}$ be the three vertices of a facet, consider the polynomial D defined as

$$D = \begin{vmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_a & y_a & z_a & 1 \end{vmatrix} = C_x x_a + C_y y_a + C_z z_a \tag{9}$$

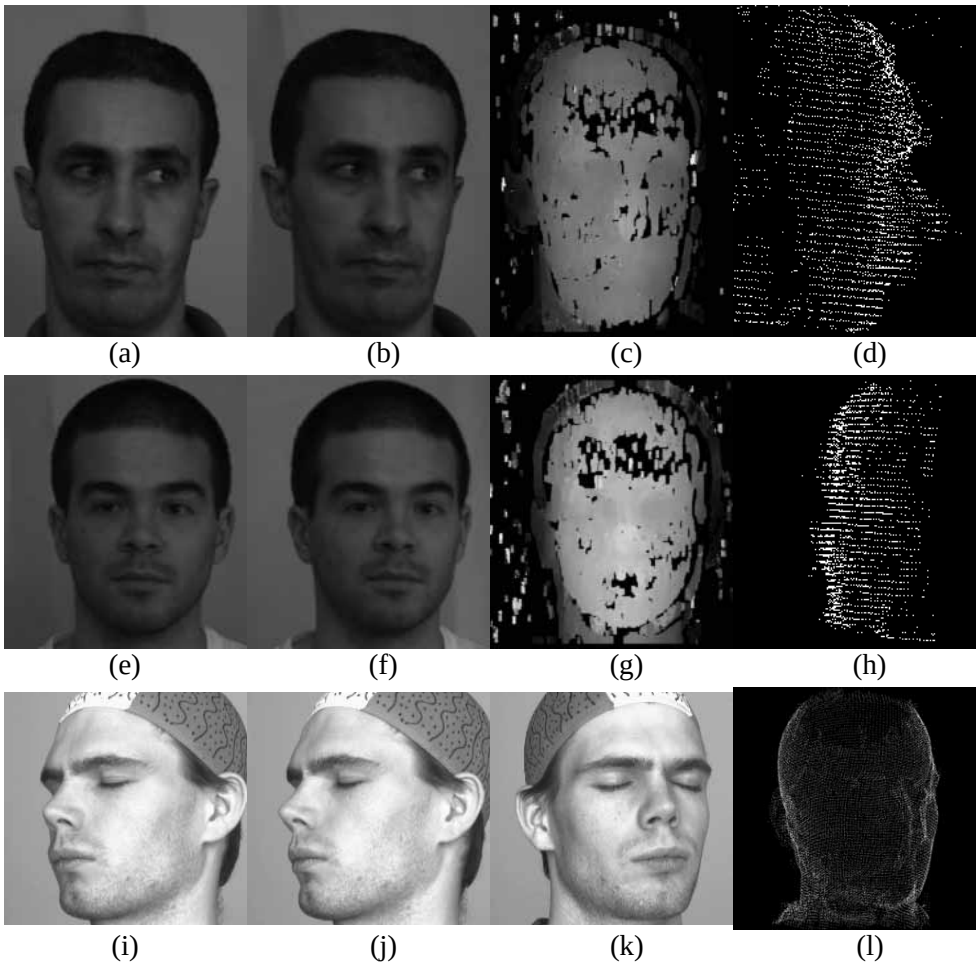


Figure 4: Sources of stereo data. (a,b) Two images of a stereo triplet. (c) Corresponding disparity map. (d) 3–D attractors derived from the disparity maps using the technique of Section 4.1. Note that this set of 3–D points is still relatively noisy. (e,f,g,h) Two images from a second stereo triplet, disparity map and 3-D attractors. (i,j,k) Three images out of a video sequence of forty (Courtesy of IGP, ETHZ). (l) 3-D attractors computed by treating each pair of consecutive images as as stereo-pair. This last set of 3–D points is much cleaner than the other two because we have used more images—40 as opposed to 3— of higher resolution—512x512 as opposed to 376x258 and 368x208.

where C_x,C_y, and C_z are polynomial functions of x_i,y_i, and z_i for $1 \leq i \leq 3$. It is easy to show that the facet normal is parallel to the vector (C_x, C_y, C_z) and that the orthogonal distance d_i^a of the attractor to the facet becomes

$$d^a = D/\sqrt{C_x^2 + C_y^2 + C_z^2}$$

Finding this "closest facet" is computationally expensive if we exhaustively search the list of facets for the one that initially minimizes the observation error of Equation 8. However, the search can be made efficient and fast if we assume that the 3–D points can be identified by their projection in an image, as is the case with stereo data. For each image, we use the Z-buffering capability of our machines to compute what we call a "Facet-ID image:" We encode the index i of each facet f_i as a unique color, and project the surface into the image plane, using a standard hidden-surface algorithm. We can then trivially look

up the facet that projects at the same place as a given point.

We recompute these attachments at each stage of the hierarchical fitting scheme of Section 3, that is each time we introduce a new control triangulation. Because some of the patches derived from stereo may be spurious, we use a variant of the Iterative Reweighted Least Squares [1] technique. Each time we recompute the attachments, we also recompute the weight w_i^{stereo} of observation i and take it to be inversely proportional to the initial distance d_i^a of the data point to the surface triangulation. More specifically we compute w_i^{stereo} as

$$w_i^{stereo} = exp(\frac{-d_i^a}{\overline{d^a}}) \quad \text{for } 1 \leq i \leq n \tag{10}$$

where $\overline{d^a}$ is the median value of the d_i. In effect, we use $\overline{d^a}$ as an estimate of the noise variance and we discount the influence of points that are more than a few standard deviations away.

Robustness can be further increased by multiplying the w_i by the dot product of the normal of the surface patches used to derive the attractors with the current estimate of the normal vector of the facet to which the facet is attached. In this manner, the influence of patches whose orientation is very different from that of the attached facet is discounted.

4.2 Silhouette Data

Contrary to 3–D edges, silhouette edges are typically 2–D features since they depend on the viewpoint and cannot be matched across images. However, they constrain the surface tangent. Each point of the silhouette edge defines a line that goes through the optical center of the camera and is tangent to the surface at its point of contact with the surface. The points of a silhouette edge therefore define a ruled surface that is tangent to the surface. In terms of our facetized representation, this can be expressed as follows. Given a silhouette point (u_s, v_s) in an image, there must be a facet with vertices $(x_i, y_i, z_i)_{1 \leq i \leq 3}$ whose image projections $(u_i, v_i)_{1 \leq i \leq 3}$, as well as (u_s, v_s), all lie on a single line. This can be enforced by writing for each silhouette point three observation equations:

$$\begin{vmatrix} u_i & u_j & u_s \\ v_i & v_j & v_s \\ 1 & 1 & 1 \end{vmatrix} = 0 + \epsilon_{ij} \quad , 1 \leq i \leq 3 \ , \ i \leq j \leq 3 \tag{11}$$

where 0 again is the nominal value, and the (u_i, v_i) are derived from the (x_i, y_i, z_i) using the camera model.

As with the 3–D attractors of Section 4.1, we can use the iterative reweighting scheme described above and the main problem is to find the "silhouette facet" to which the constraint applies. As before, we can exhaustively search the facets of the surface triangulation for those that initially are close to being parallel to the ruled surface defined by the silhouette and minimize the observations errors of Equation 11. Alternatively, we can use the Facet-ID image to speed up the process: Since the silhouette point (u_s, v_s) can lie outside the projection of the current estimate of the surface, we must search the Facet-ID image in a direction normal to the silhouette edge for a facet that minimizes the initial observation error. This, in conjunction with our coarse-to-fine optimization scheme, has proved a robust way of determining which facets correspond to silhouette points.

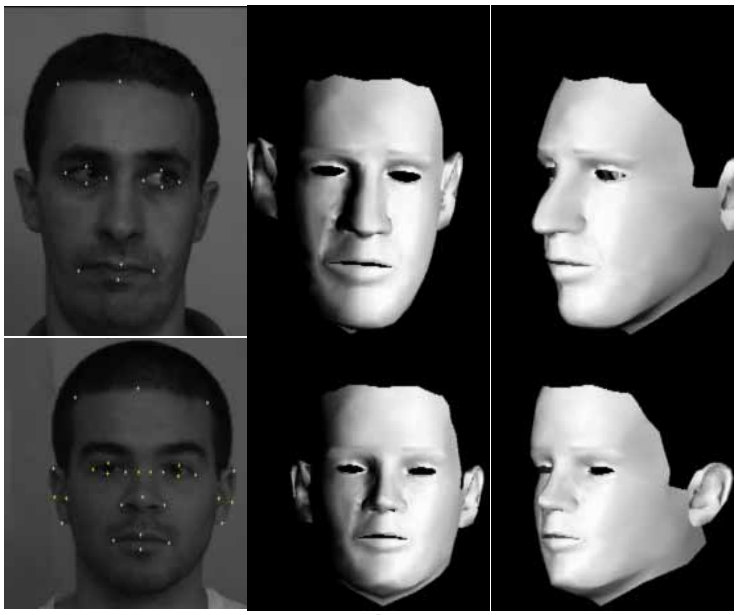
4.3 2–D Location of Features

For each vertex x_i, y_i, z_i of the *surface triangulation* whose 2–D projection u_i, v_i is known, we can write two observation equations:

$$\begin{aligned} Pr_u(x_i, y_i, z_i) &= u_i + \epsilon_i^u \\ Pr_v(x_i, y_i, z_i) &= v_i + \epsilon_i^v \quad , \end{aligned} \tag{12}$$

where Pr_u and Pr_v stand for the projection in u and v. In this way we do not need the explicit 3–D position of these feature points, only their 2–D image location.

5 Results



(a) (b) (c)

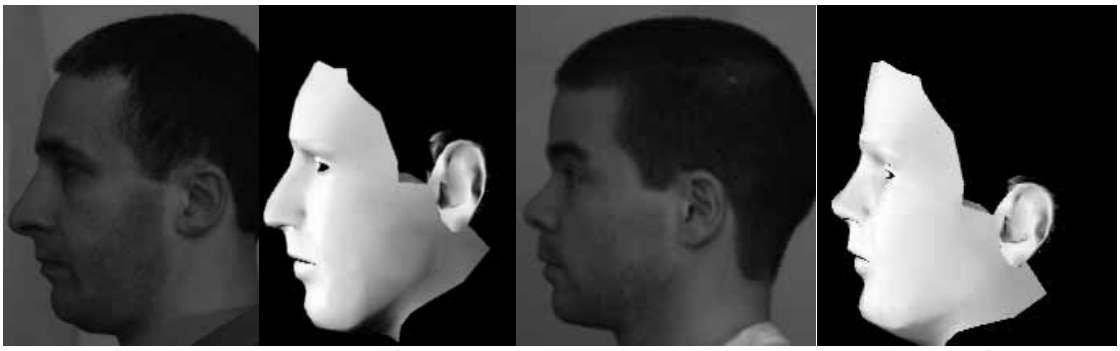
Figure 5: Face reconstruction using stereo triplets of Figure 4: (a) The full set of hand-entered 2–D points locations that our system recognizes. (b,c) For each subject, two shaded views of the recovered face model.

To demonstrate the effectiveness of our fitting procedure and the relative insensitivity to the weighting and stiffness parameters of Section 3, we have used the two stereo triplets of Figure 4. We ran the procedure of Section 2 using the 2-D point locations of Figure 5(a) in addition to the stereo data of Figure 4 to produce the face models shown in Figure 5(b,c).

These results were obtained for particular values of the stiffness parameter λ_S of Equation 5 and of the relative observation weights the λ_{type} of Section 3.3, specifically $\lambda_S = 0.007$, $\lambda_{stereo} = 1.0$ and $\lambda_{features} = 0.3$.

To evaluate these results qualitatively, in Figure 6, we show two side views of the heads that have *not* been used to perform the computation and show the reconstructed models seen in a similar pose. Note that the face outlines corresponds quite accurately except where stereo can be expected to fail

because the surface slopes away from the camera: Bottom of the nose and of the chin. One way to address this problem would be to use a model-based stereo techniques that are less sensitive to the slope of the surfaces [12, 16] In this work, we chose a simpler approach: Since we are not concerned with utmost precision but only with realism, our experience is that using additional images and stereo pairs effectively solves the problem as shown below.



(a) (b) (c) (d)

Figure 6: Evaluating the quality of the reconstructions using side views: (a) A image of the subject of Figure 4(a,b) that has *not* been used to perform the computation. (b) The corresponding face model shown in a similar pose. (c,d) Side image and face model for the subject of Figure 4(c,d). The profiles are very similar except under the nose, that is, where stereo cannot be expected to give reliable information.

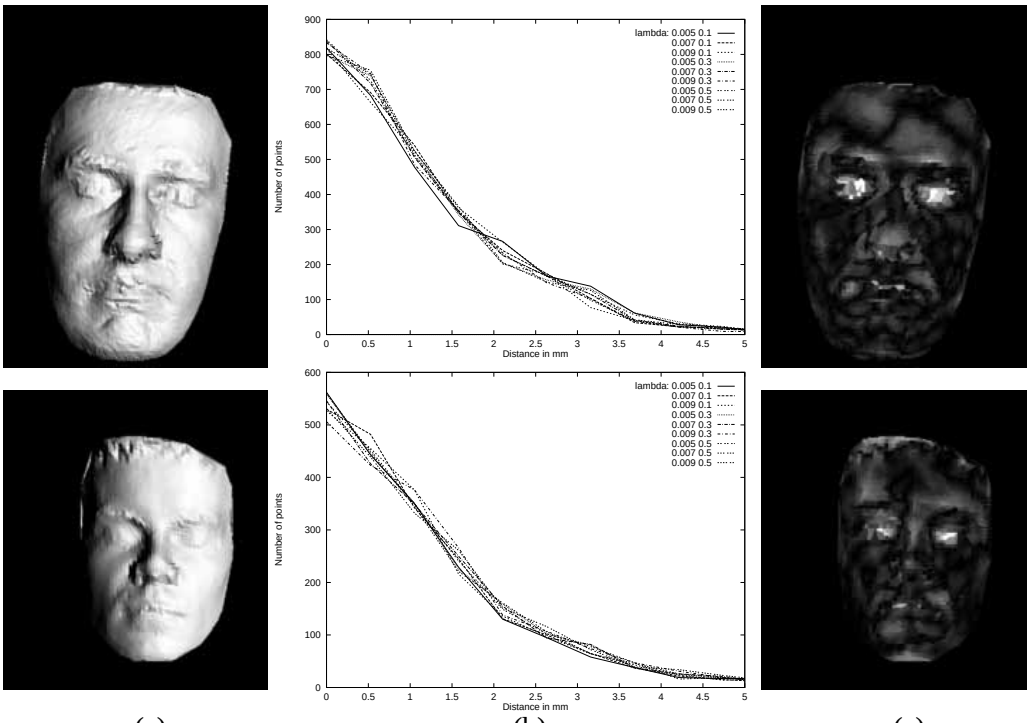
For a more quantitative evaluation of these results, we have acquired 3–D models of the faces of both subjects using a Minolta^{tm} scanner. The theoretical precision of the laser is approximately 0.3 millimeters and it can therefore be considered as a reasonable approximation of ground truth. Of course, in practice, even the laser exhibits a few artifacts. However, since the two reconstruction methods are completely independent of one another, places of agreement are very likely to be correct for both.

The images were calibrated using the INRIA camcal package[28] without accounting for lens distortion which results in a slight deformation of the reconstructed surfaces. To correct for it, we approximate the deformation by an affine transform and compute the one that best maps our model onto the laser output. We then histogram the distances of the vertices of the latter to the surface of the former. The corresponding plots appear in Figure 7(a,b). In each case we ran our fitting algorithm nine times, for all pairs of values of $\lambda_S \in \{0.05, 0.07, 0.09\}$ and $\lambda_{features} \in \{0.1, 0.3, 0.5\}$ and counted the number of 3–D points in the laser output that were within a given distance, expressed here in millimeters, of the surface model. Note that:

1. For both faces, the nine curves are essentially superposed, indicating the relative insensitivity of the fitting procedure to parameter choice.

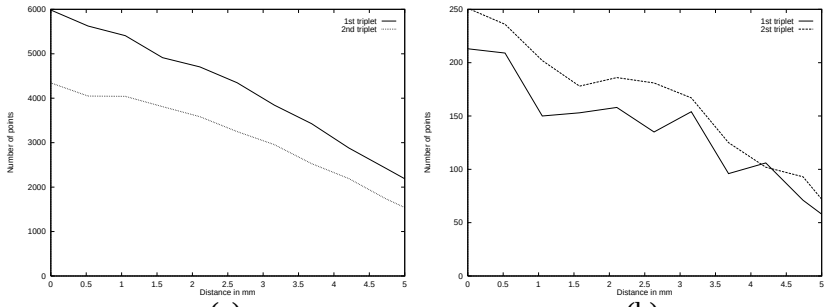
2. The median errors for both heads and the nine parameter settings used here are all very close to 1 millimeter—to be precise, 1.027 and 1.074 mm for the examples of Figure 5(b,c). Given the camera geometry and distances of the heads to the cameras, an error of one pixel in disparity corresponds to an error in measured range of 8 to 10 millimeters. The precision of the correlation based algorithm we use is in the order of half a pixel, outliers excluded [7]. We therefore conclude that our fitting algorithm performs an effective and robust averaging of the input data.

For comparison's sake, in Figure 8, we have also histogrammed the distances of the original stereo data to the outputs of the laser before and after fitting the local surface patches of Section 4.1. The



(a) (b) (c)

Figure 7: Quantitative estimation. (a) Reconstructions obtained using a Minolta^{tm} laser scanner for the first two subjects of Figure 4. (b) Histograms of the distances of the 3–D points in the laser output to the surface derived independently using our approach on the two image triplets of Figure 4 and using nine different parameter settings: $\lambda_{stereo} = 1.0$, $\lambda_S \in \{0.05, 0.07, 0.09\}$ and $\lambda_{features} \in \{0.1, 0.3, 0.5\}$. The nine curves in both histograms are almost superposed indicating the relative insensitivity of our algorithm to the exact value of those parameters. The distances are expressed in millimeters and, for the camera geometry used, a 1 millimeter error approximately corresponds to an error of 1/8 to 1/10 of a pixel in disparity. (c) Graphic depiction of the errors: the surface of the laser output is color-coded so that the white areas are those that are furthest away from the stereo model. White corresponds to an error greater than 10 millimeters.



(a) (b)

Figure 8: Quality of the raw data: (a) Distance to the laser output of the 3-D points derived directly from disparity maps. (b) Distance to the laser output of the 3-D attractors.

overall shape of the histogram, or its median value (2.78 millimeters and 2.69 millimeters for one face, 2.72 millimeters and 2.53 millimeters for the other), does not change much but the number of points is drastically reduced by the fitting process. The computation times required to perform the two key steps

of the procedure depicted by Figure 2 are given below:

Seconds on an R10000 SGI	Face at the top of Figure 5	Face at the bottom of Figure 5
Disparity maps to attractors	22.7 (376x258 images)	9.5 (368x208 images)
Fitting mask to attractors	56.9 (3186 attractors)	58.3 (2812 attractors)

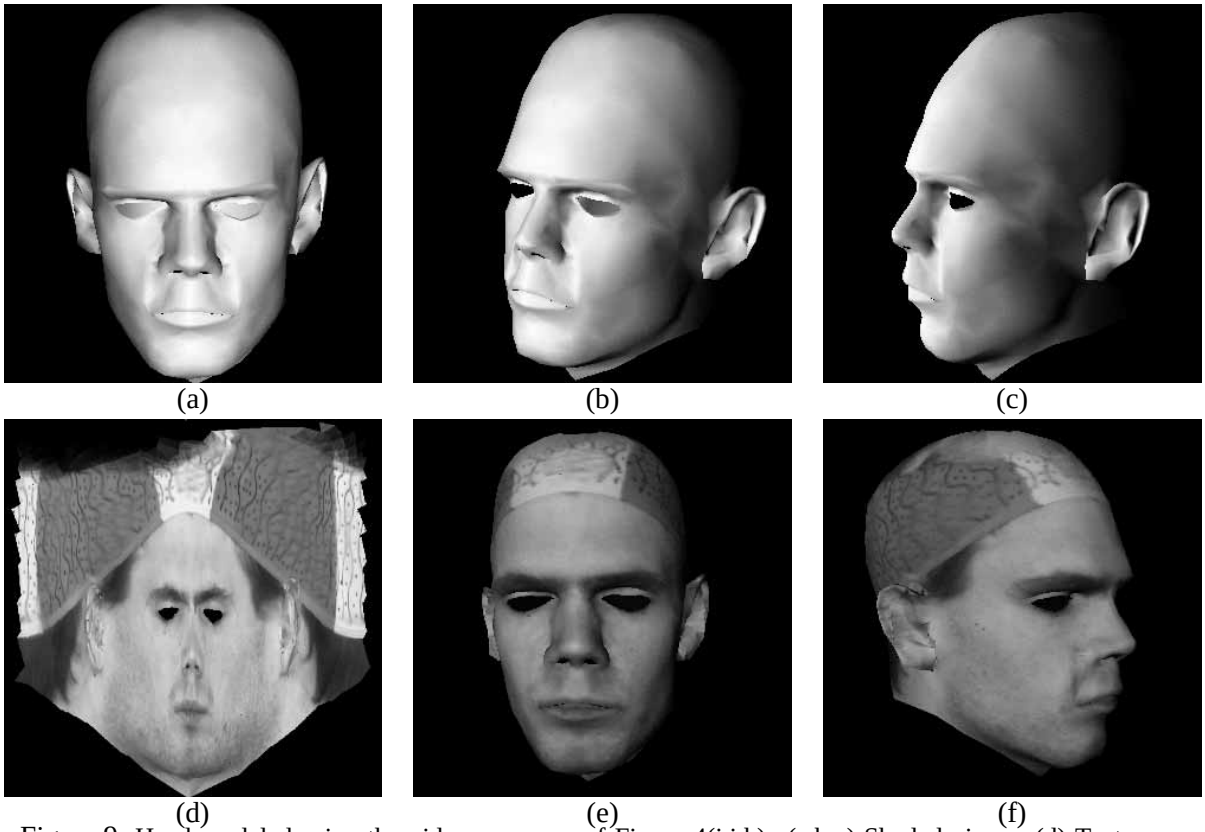


Figure 9: Head modeled using the video sequence of Figure 4(i,j,k). (a,b,c) Shaded views. (d) Texture map. (e,f) Texture mapped views.

In Figure 9, we demonstrate the use of the video sequence of Figure 4. Each pair of consecutive image is treated as a stereo pair, yielding a full reconstruction and the generation of a texture map for the whole head. In this case, because the subject was wearing a cap, the stereo data was good enough to compute both the shape of the face and of the skull.

Given this complete head model, we compute the texture map shown in Figure 9(d) and allows textured reconstruction. If we were to use color images instead of black and white ones, we can repeat the process on each of the color bands and produce color models such as the ones shown below.

Similarly to further improve upon the quality of the results shown in Figure 5(b,c), we have used two nine-image video sequences of the same people turning their heads in front of the camera. Here the motion of the head was recovered using an automated bundle adjustment algorithm [9, 10]. Following the complete procedure outlined in Section 2, we first recovered the face as before. We then deformed the skull of the generic model using both stereo and semi-automatically entered hair outlines, yielding the complete heads of Figure 10(c,d) and 11(c,d) and the corresponding texture-mapped representations. Note that the profiles of the reconstructed models are now closer to the real ones shown in Figure 6.

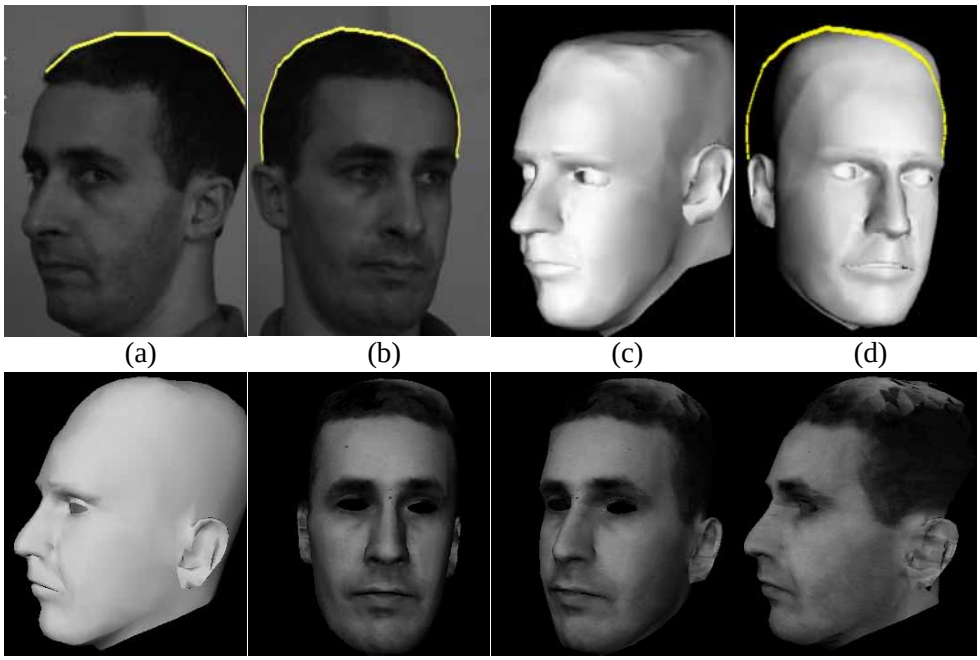


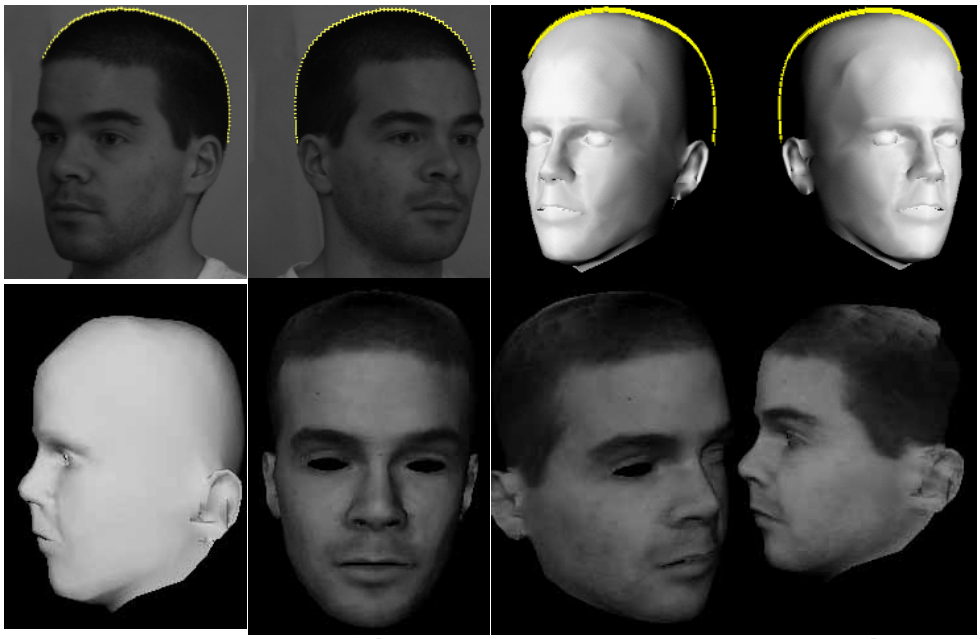
Figure 10: Using a video sequence to model the head of Figure 4(a,b): Two extreme positions of the head in a sequence of nine. The overlaid hair outlines were drawn using "intelligent scissors" [21] and were used to compute the shape of the skull in the absence of good stereo data for the hair. (c,d) The recovered head model in a similar pose. The overlaid hair outlines correspond to the silhouettes . (e) A side view to be compared to that of Figure 6(a). (f,g,h) Texture mapped views of the model.

Using color images, we can reconstruct the heads by running our stereo algorithm on one of the bands, the red one for example, and use the original images to compute colored texture maps. This yields results such as those of Figure 12 and 13. In all cases shown in this Section, the masks can be animated as shown in Figure 12(e,f) and 13(f). In both cases, we have used the original texture images to texture map not only the skin but also the eyes. To model the hair of the subject of Figure 13 we used the hair surface of Figure 1(d).

6 Conclusion

We have presented a technique that allows us to fit a complex animation model to noisy image data with very limited manual intervention. As a result, these models can be produced cheaply and fast. Furthermore, because our approach relies on the use of a coarse to fine control triangulation, it can be used to fit arbitrarily complex models whose topology is designed for animation purposes and are not necessarily well suited for surface reconstruction.

In future work, we intend to extend the approach to the modeling of dynamic faces and more importantly to the estimation not only of the parameters that control the shape of the surface but also of those that control the various facial expressions. We will also incorporate self-calibration techniques into our framework so that we can achieve this result using ordinary video sequences filmed using regular camcorders.



(a) (b) (c) (d)

Figure 11: Using a video sequence to model the head of Figure 4(e,f): Two extreme positions of the head in a sequence of nine, with overlaid hair outlines. (c,d) The recovered head model in a similar pose. (e) A side view to be compared to that of Figure 6(c). (f,g,h) Texture mapped views of the model.

Furthermore, the face model we use has been one of the starting points for the facial animation parameters defined in the MPEG-4 FBA work. When standardization is complete, it will therefore be easy to make the parameters of our model conformant with the MPEG-4 norm for facial animation and the work presented here will become directly relevant to video transmission.

Acknowledgments

We wish to thank Prof. Nadia Magnenat Thalmann, Prof. Daniel Thalmann and Dr. Prem Kalra for having made their facial animation model available to us. We are also indebted to Prof. Grün for sharing with us his insights about least-squares technology.

References

[1] A. E. Beaton and J.W. Turkey. The Fitting of Power Series, Meaning Polynomials, Illustrated on Band-Spectroscopic Data. *Technometrics*, 16:147–185, 1974.

[2] A. Blake and M. Isard. 3D Position Attitude and Shape Input Using Video Tracking of Hands and Lips. In *Computer Graphics, SIGGRAPH Proceedings*, pages 71–78, July 1994.

[3] T.F. Cootes and C.J. Taylor. Locating Objects of Varying Shape Using Statistical Feature Detectors. In *European Conference on Computer Vision*, Cambridge, England, April 1996.

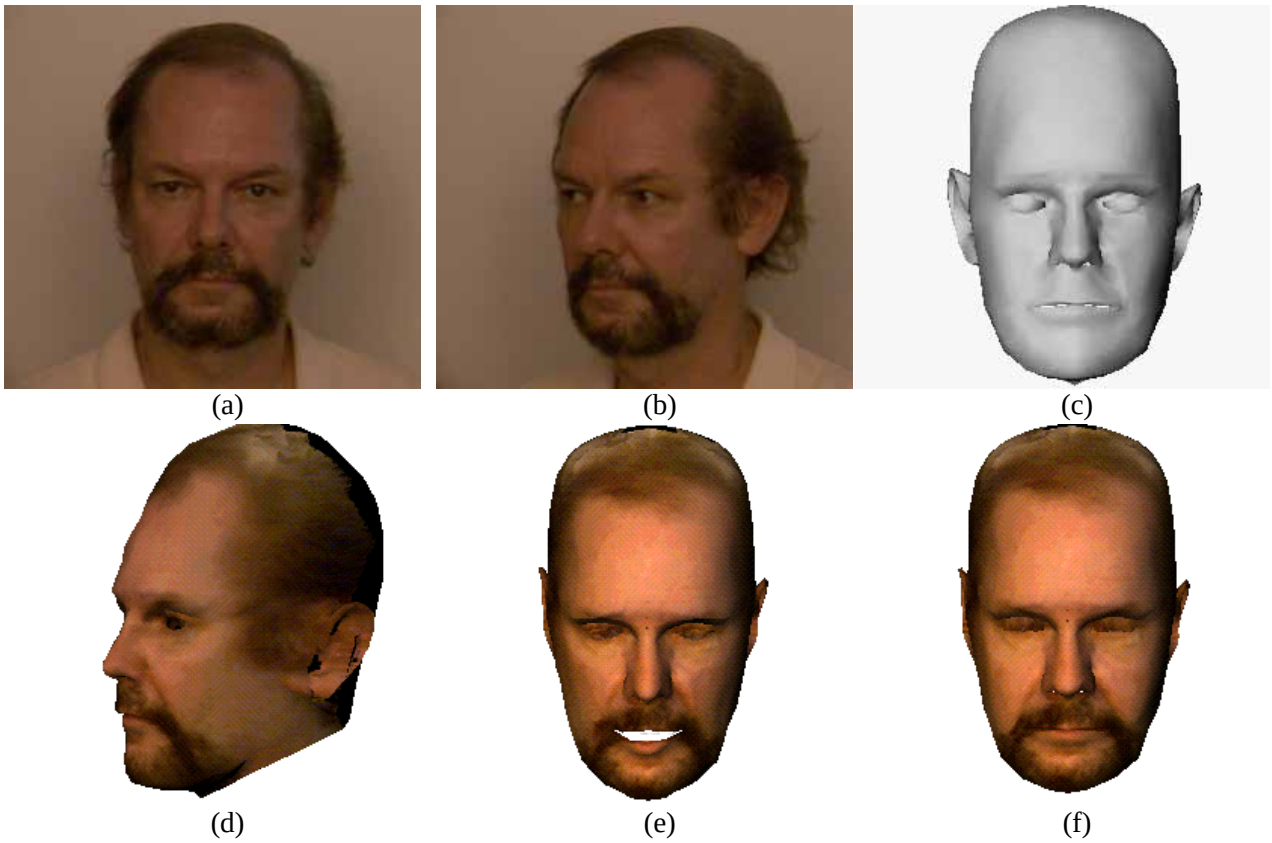


Figure 12: Reconstruction and animation using 9 color images of a video sequence: (a,b) Two of the images. (c) Recovered head model without hair. (d) Texture mapped model. (e,f) Synthetically generated opening of the mouth and closing of the eyelids.

[4] D. DeCarlo and D. Metaxas. Deformable Model-Based Shape and Motion Analysis from Images using Motion Residual Error. In *International Conference on Computer Vision*, pages 113–119, Bombay, India, 1998.

[5] F. Devernay and O. D. Faugeras. Computing Differential Properties of 3–D Shapes from Stereoscopic Images without 3–D Models. In *Conference on Computer Vision and Pattern Recognition*, pages 208–213, Seattle, WA, June 1994.

[6] O.D. Faugeras, Q.-T. Luong, and S.J. Maybank. Camera self-calibration: theory and experiments. In *European Conference on Computer Vision*, pages 321–334, Santa-Margerita, Italy, 1992.

[7] P. Fua. A Parallel Stereo Algorithm that Produces Dense Depth Maps and Preserves Image Features. *Machine Vision and Applications*, 6(1):35–49, Winter 1993.

[8] P. Fua. From Multiple Stereo Views to Multiple 3–D Surfaces. *International Journal of Computer Vision*, 24(1):19–35, August 1997.

[9] P. Fua. Face Models from Uncalibrated Video Sequences. In *Workshop on Modelling and Motion Capture Techniques for Virtual Environments*, Geneva, Switzerland, November 1998.

[10] P. Fua. From Synthesis to Analysis: Fitting Human Animation Models to Image Data. In *Computer Graphics International*, Canmore, Alberta, Canada, June 1999.

[11] P. Fua and C. Brechbühler. Imposing Hard Constraints on Deformable Models Through Optimization in Orthogonal Subspaces. *Computer Vision and Image Understanding*, 24(1):19–35, February 1997.

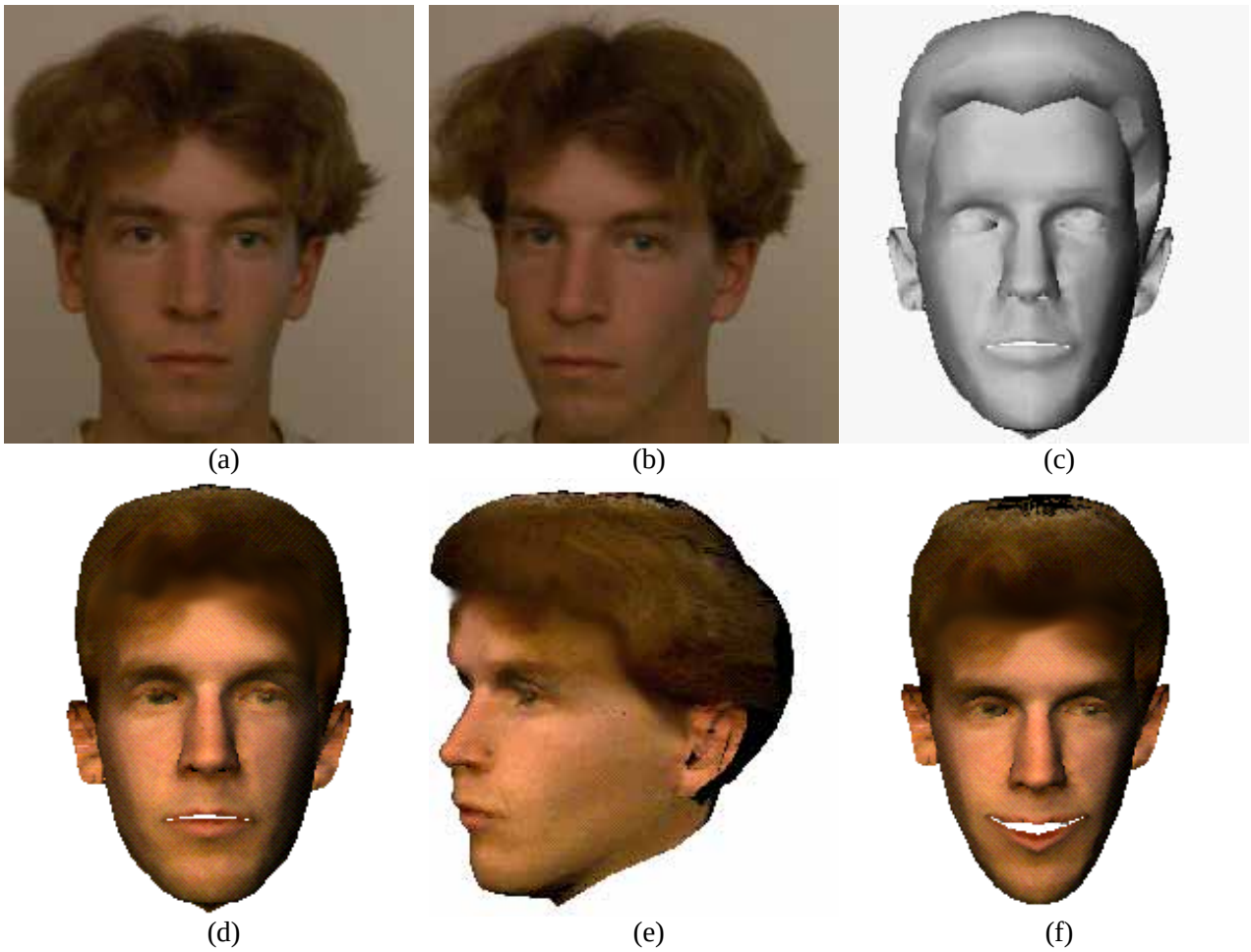


Figure 13: Reconstruction and animation using 17 color images of a video sequence: (a,b) Two of the images. (c) Recovered head model with hair surface. (d,e) Two texture mapped views. (f) Synthetically generated smile.

[12] P. Fua and Y. G. Leclerc. Object-Centered Surface Reconstruction: Combining Multi-Image Stereo and Shading. *International Journal of Computer Vision*, 16:35–56, September 1995.

[13] P.E. Gill, W. Murray, and M.H. Wright. *Practical Optimization*. Academic Press, London a.o., 1981.

[14] T.S. Jebara and A. Pentland. Parametrized Structure from Motion for 3D Adaptive Feedback Tracking of Faces. In *Conference on Computer Vision and Pattern Recognition*, pages 144–150, Porto Rico, June 1997.

[15] P. Kalra, A. Mangili, N. Magnenat Thalmann, and D. Thalmann. Simulation of Facial Muscle Actions Based on Rational Free Form Deformations. In *Eurographics*, 1992.

[16] S. B. Kang. A Structure from Motion Approach using Constrained Deformable Models and Apperance Prediction. Technical Report CRL 97/6, Digital, Cambridge Research Laboratory, October 1997.

[17] K. Konolige. Small Vision Systems: Hardware and Implementation. In *Eighth International Symposium on Robotics Research*, Hayama, Japan, October 1997.

[18] Y. G. Leclerc and A. F. Bobick. The Direct Computation of Height from Shading. In *Conference on Computer Vision and Pattern Recognition*, Lahaina, Maui, Hawaii, June 1991.

[19] W.S. Lee and N. Magnenat Thalmann. From Real Faces To Virtual Faces: Problems and Solutions. In *3IA*, Limoges, France, 1998.

[20] Y. Lee, D. Terzopoulos, and K. Waters. Realistic Modeling for Facial Animation. In *Computer Graphics, SIGGRAPH Proceedings*, pages 191–198, Los Angeles, CA, August 1995.

[21] E.N. Mortensen and W.A. Barrett. Intelligent Scissors for Image Composition. In *Computer Graphics, SIGGRAPH Proceedings*, pages 191–198, Los Angeles, CA, August 1995.

[22] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D.H. Salesin. Synthesizing Realistic Facial Expressions from Photographs. In *Computer Graphics, SIGGRAPH Proceedings*, volume 26, pages 75–84, July 1998.

[23] M. Pollefeys, R. Koch, and L. VanGool. Self-Calibration and Metric Reconstruction In Spite of Varying and Unknown Internal Camera Parameters. In *International Conference on Computer Vision*, 1998.

[24] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes, the Art of Scientific Computing*. Cambridge U. Press, Cambridge, MA, 1986.

[25] M. Proesmans, L. Van Gool, and A. Oosterlinck. Active acquisition of 3D shape for Moving Objects. In *International Conference on Image Processing*, Lausanne, Switzerland, September 1996.

[26] R. Szeliski and S.B. Kang. Recovering 3–D Shape and Motion from Image Streams Using Non Linear Least Squares. *Journal of Visual Communication and Image Representation*, 5(1):10–28, 1994.

[27] L. Tang and T.S. Huang. Analysis-based facial expression synthesis. *ICIP-III*, 94:98–102.

[28] J.P. Tarel and J.M. Vezien. Camcal Manual: A Complete Software Solution for Camera Calibration. Technical Report 0196, INRIA, September 1996.

[29] B. Triggs. Autocalibration and the Absolute Quadric. In *Conference on Computer Vision and Pattern Recognition*, pages 609–614, 1997.