3D Face Modeling From Monocular Video Sequences

Amit K. Roy-Chowdhury
Dept. of Electrical Engineering
University of California
Riverside, CA 92521, USA

Rama Chellappa
Center for Automation Research
University of Maryland
College Park, MD 20740, USA

Himanshu Gupta
ObjectVideo
11600 Sunrise Valley Dr. Suite. 290
Reston VA 20191, USA

Abstract

In this chapter we present two algorithms for 3D face modeling from a monocular video sequence. The first method is based on Structure from Motion (SfM), while the second one relies on contour adaptation over time. The SfM based method incorporates statistical measures of quality of the 3D estimate into the reconstruction algorithm. The initial multi-frame SfM estimate is smoothed using a generic face model in an energy function minimization framework. Such a strategy avoids excessively biasing the final 3D estimate towards the generic model. The second method relies on matching a generic 3D face model to the outer contours of a face in the input video sequence, and integrating this strategy over all the frames in the sequence. It consists of an edge-based head pose estimation step, followed by global and local deformations of the generic face model in order to adapt it to the actual 3D face. This contour adaptation approach is able to separate the geometric subtleties of the human head from the variations in shading and texture, and it does not rely on finding accurate point correspondences across frames. Detailed experimental evaluation of both the methods along with reconstructed 3D models is presented.

1 Introduction

Constructing 3D face models from a video sequence is one of the challenging problems of computer vision. Successful solution of this problem has applications in multimedia, computer graphics, and face recognition. In multimedia, 3D face models can be used in video conferencing applications for efficient transmission. In computer graphics applications, 3D face models form the basic building block upon which facial movements and expressions can be added. Being able to build these models automatically from video data would greatly simplify animation tasks where models are now painstakingly built with significant human intervention. By incorporating 3D models into face recognition systems, the problems arising due to pose, illumination, and expression variations can be effectively addressed. Most current commercial systems address special cases of the 3D face reconstruction problem that are well-constrained by additional information in the form of depth estimates available from multiple cameras, projecting laser or other patterns on the face, decorating the face with special textures to make inter-frame correspondences simple, or using structured light to reveal the contours of the face. All these constraints and the special hardware needed reduce the operational flexility of the system.

Various researchers have addressed the issue of 3D face modeling. In [16], the authors used an extended Kalman filter to recover the 3D structure of a face which was then used for tracking. A method for recovering non-rigid 3D shapes as a linear combination of a set of basis shapes was proposed in [4]. A factorization based method for recovering non-rigid 3D structure and motion from video was presented in [3]. In [20], the author proposes a method for self-calibration in the presence of varying internal camera parameters and reconstructs metric 3D structure. One of the common approaches to solve the problem of 3D reconstruction from a monocular video is Structure from Motion (SfM). Numerous SfM algorithms [15] that can reconstruct a 3D scene from two or more images exist in the literature. The basic idea behind most SfM algorithms is to recover the structure of 3D points on a rigid object from 2D point correspondences across images, or recover a dense depth map from optical flow. These methods have been adapted for face modeling in [11, 28], where the authors proposed solving the problem of 3D face modeling using a generic model. Their method of bundle-adjustment works by initializing the reconstruction algorithm with this generic model. Romdhani, Blanz and Vetter [22] came up with an impressive appearance-based approach where they showed that it is possible to recover the shape and texture parameters of a 3D morphable model from a single image. Shape from Contours [2] is another promising approach for 3D reconstruction. One of the strongest cues for the 3D information contained in a 2D image is the outline of an object in the image. The occluding contour (extreme boundary) in a 2D image directly reflects the 3D shape. Shape from Silhouette techniques have been used to reconstruct 3D shapes from multiple silhouette images of an object without assuming any previous knowledge of the object to be reconstructed [17]. However, it is impossible to recover the concavities in the shape of the object from the silhouettes or contours. But if we assume prior knowledge of the object being reconstructed, and use a generic model, contour information can be exploited as an important constraint for the exact shape of the object. Moghaddam et al [18] have developed a system to recover the 3D shape of a human face from a sequence of silhouette images. They used a downhill simplex method to estimate the model parameters, which are the coefficients of the eigenhead basis functions.

In this chapter, we present two methods that we have developed for 3D face modeling from monocular video sequences. The first method uses the multi-frame SfM approach to arrive at an initial estimate of the face model [27]. This is followed by a smoothing phase where the errors in this estimate are corrected by comparison with a generic face model. The method also involves a statistical quality evaluation of the input video [24], and compensating for fluctuations in quality within the SfM algorithm framework. The second approach that we present relies on matching a generic 3D face model to the outer contours of the face to be modeled and a few of its internal features [14]. Using contours separates the geometric subtleties of the human head from the variations in shading and texture. The adaptation of the generic face model is integrated over all the frames of the video sequence, thus producing a 3D model that is accurate over a range of pose variations.

2 SfM Based 3D Face Modeling

In this section, we present a method for face modeling from monocular video using SfM, with special emphasis on the incorporation of the generic face model and evaluating the quality of the input video sequence. The main features of our method are the following.

Reconstructing from a Monocular Video: This is particularly important in unregulated surveillance applications where the training data (from which the 3D model needs to be estimated) may contain a few images or a video from one view (e.g. frontal), but the probe may be another view of the person (e.g. profile). Since the motion between pairs of frames in a monocular video is usually small, we adopt the optical flow paradigm for 3D reconstruction [19]. Also, estimating the motion between the pairs of frames accurately may be a challenge in many situations because of differences in the quality of the input video. Our method learns certain statistical parameters of the incoming video data and incorporates them in the algorithm. Quality evaluation is done by estimating the statistical error covariance of the depth estimate analytically from the optical flow equations. The details of the statistical analysis can be found in [23, 24].

Avoid Biasing the Reconstruction with the Generic Model: Mathematically speaking, the introduction of the generic model is similar to introducing constraints on the solution of the 3D estimation problem. In our method, we introduce the generic model *after* obtaining the estimate using the SfM algorithm. The SfM algorithm reconstructs purely from the video data after computing the optical flow. This is unlike the methods in [11] and [28] where the generic model was used to initialize a bundle adjustment approach. The difficulty with this approach is that the algorithm often converges to a solution very near this initial value, resulting in a reconstruction which has the characteristics of the generic model, rather than that of the particular face in the video which needs to be modeled. This method may give very good results when the generic model has significant similarities with the particular face being reconstructed. However, if the features of the generic model are different from those of the face being reconstructed, the solution obtained using this approach may be unsatisfactory. We provide some experimental validation for this statement later. After the initial SfM-based estimate is obtained, we use a cost function which identifies local regions in the generic face model where there are no sharp depth discontinuities, looks for deviations in the trend of the values of the 3D estimate in these regions and then corrects for the errors. The optimization of the cost function is done in a Markov Chain Monte Carlo (MCMC) framework using a Metropolis-Hastings sampler [8]. The advantage of this method is that the particular characteristics of the face that is being modeled are not lost since the SfM algorithm does not incorporate the generic model. However, most errors (especially those with large deviations from the average representation) in the reconstruction are corrected in the energy function minimization process by comparison with the generic model.

We start by providing a brief overview of error modeling in SfM, followed by the reconstruction algorithm, including incorporation of the generic model, and finally present the results.

2.1 Error Estimation in 3D Reconstruction

Since the motion between adjacent frames in a video sequence of a face is usually small, we will adopt the optical flow framework for reconstructing the structure [19]. It is assumed that the coordinate frame is attached rigidly to the camera with the origin at the center of perspective projection and the z-axis perpendicular to the image plane. The camera is moving with respect to the face being modeled (which is assumed rigid) with translational velocity $\mathbf{V} = [v_x, v_y, v_z]$ and rotational velocity $\mathbf{\Omega} = [\omega_x, \omega_y, \omega_z]$ (this can be reversed by simply changing the sign of the velocity vector). Using the small-motion approximation to the perspective projection model for motion field analysis, and denoting by $p(x, y)$ and $q(x, y)$, the horizontal and vertical velocity fields of a point (x, y) in the image plane, we can write the equations relating the object motion and scene depth by [19]

$$\begin{aligned} p(x,y) &= (x - fx_f)h(x,y) + \frac{1}{f}xy\omega_x - (f + \frac{1}{f}x^2)\omega_y + y\omega_z \\ q(x,y) &= (y - fy_f)h(x,y) + (f + \frac{1}{f}y^2)\omega_x - \frac{1}{f}xy\omega_y - x\omega_z, \end{aligned} \tag{1}$$

where f is the focal length of the camera, $(x_f, y_f) = (\frac{v_x}{v_z}, \frac{v_y}{v_z})$ is known as the *focus of expansion* (FOE), and $h(x, y) = \frac{v_z}{z(x,y)}$ is the scaled inverse scene depth. We will assume that the FOE is known over a few frames of the video sequence. Under the assumption that the motion between adjacent frames in a video is small, we compute the FOE from the first two or three frames and then keep it constant over the next few frames [30]. For N corresponding points, using subscript i to represent the above defined quantities at the i^{th} point, we define (similar to [30])

$$\begin{aligned}
\mathbf{h} &= (h_1, h_2, ..., h_N)^T_{N\times 1} \\
\mathbf{u} &= (p_1, q_1, p_2, q_2, ..., p_N, q_N)^T_{2N\times 1} \\
\mathbf{r}_i &= (x_i y_i, -(1+x_i^2), y_i)^T_{3\times 1} \\
\mathbf{s}_i &= (1+y_i^2, -x_i y_i, -x_i)^T_{3\times 1} \\
\mathbf{\Omega} &= (w_x, w_y, w_z)^T_{3\times 1} \\
\mathbf{Q} &= \begin{bmatrix} r_1 & s_1 & r_2 & s_2 & ... & r_N & s_N \end{bmatrix}^T_{2N\times 3} \\
\mathbf{P} &= \begin{bmatrix} x_1 - x_f & 0 & \cdots & 0 \\ y_1 - y_f & 0 & \cdots & 0 \\ 0 & x_2 - x_f & \cdots & 0 \\ 0 & y_2 - y_f & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & x_N - x_f \\ 0 & 0 & \cdots & y_N - y_f \end{bmatrix}_{2N\times N}
\end{aligned} \tag{2}$$

$$\begin{aligned}
\mathbf{A} &= [\mathbf{P} \quad \mathbf{Q}]_{2N\times(N+3)} \\
\mathbf{z} &= \begin{bmatrix} \mathbf{h} \\ \mathbf{\Omega} \end{bmatrix}_{(N+3)\times 1}.
\end{aligned} \tag{3}$$

Then (1) can be written as

$$\mathbf{Az} = \mathbf{u}. \tag{4}$$

Our aim is to compute $\mathbf{z}$ from $\mathbf{u}$ and to obtain a quantitative idea of the accuracy of the 3D reconstruction $\mathbf{z}$ as a function of the uncertainty in the motion estimates $\mathbf{u}$. Let us denote by $\mathbf{R_u}$ the covariance matrix of $\mathbf{u}$ and by C the cost function

$$C = \frac{1}{2}||\mathbf{Az} - \mathbf{u}||^2 = \frac{1}{2}\sum_{i=1}^{n=2N} C_i^2(u_i, \mathbf{z}). \tag{5}$$

In [24], using the implicit function theorem [33], we proved the following result.

Theorem 1 *Define*

$$\begin{aligned}
\mathbf{A}_{\bar{i}p} &= [0 \quad \cdots \quad 0 \quad -(x_{\bar{i}} - x_f) \quad 0 \quad \cdots \quad 0 \quad -x_{\bar{i}} y_{\bar{i}} \quad (1+x_{\bar{i}}^2) \quad -y_{\bar{i}}], \\
&= [-(x_{\bar{i}} - x_f)\mathbf{I}_{\bar{i}}(N)| - \mathbf{r}_{\bar{i}}] = [A_{\bar{i}ph}|A_{\bar{i}pm}] \\
\mathbf{A}_{\bar{i}q} &= [0 \quad \cdots \quad 0 \quad -(y_{\bar{i}} - y_f) \quad 0 \quad \cdots \quad 0 \quad -(1+y_{\bar{i}}^2) \quad x_{\bar{i}} y_{\bar{i}}(N) \quad x_{\bar{i}}], \\
&= [-(y_{\bar{i}} - y_f)\mathbf{I}_{\bar{i}}(N)| - \mathbf{s}_{\bar{i}}] = [A_{\bar{i}qh}|A_{\bar{i}qm}]
\end{aligned} \tag{6}$$

where $\bar{i} = \lceil i/2 \rceil$ is the ceiling of i ($\bar{i}$ will then represent the number of feature points N and $i = 1, ..., n = 2N$) and $\mathbf{I}_n(N)$ denotes a 1 in the n^{th} position of the array of length N and zeros elsewhere. The subscript p in $\mathbf{A}_{\bar{i}p}$ and q in $\mathbf{A}_{\bar{i}q}$ denotes that the elements of the respective vectors are derived from the p^{th} and q^{th} components of the motion in (1). Then

$$\mathbf{R_z} = \mathbf{H}^{-1}\left(\sum_i \frac{\partial C_i^T}{\partial \mathbf{z}}\frac{\partial C_i}{\partial \mathbf{u}}\mathbf{R_u}\frac{\partial C_i^T}{\partial \mathbf{u}}\frac{\partial C_i}{\partial \mathbf{z}}\right)\mathbf{H}^{-T} \tag{7}$$

$$= \mathbf{H}^{-1}\left(\sum_{\bar{i}=1}^{N}\left(\mathbf{A}_{\bar{i}p}{}^T\mathbf{A}_{\bar{i}p}R_{u\bar{i}p} + \mathbf{A}_{\bar{i}q}{}^T\mathbf{A}_{\bar{i}q}R_{u\bar{i}q}\right)\right)\mathbf{H}^{-T}, \tag{8}$$

and

$$\mathbf{H} = \sum_{\bar{i}=1}^{N} \left(\mathbf{A}_{\bar{i}p}{}^{T} \mathbf{A}_{\bar{i}p} + \mathbf{A}_{\bar{i}q}{}^{T} \mathbf{A}_{\bar{i}q} \right), \tag{9}$$

where $\mathbf{R_u} =$ *diag* $[R_{u1p}, R_{u1q}, ..., R_{uNp}, R_{uNq}]$.

Because of the partitioning of $\mathbf{z}$ in (3), we can write

$$\mathbf{R_z} = \begin{bmatrix} \mathbf{R_h} & \mathbf{R_{hm}} \\ \mathbf{R_{hm}}^T & \mathbf{R_m} \end{bmatrix}. \tag{10}$$

We can then show that for N points and M frames, the average distortion in the reconstruction is

$$D_{avg}(M,N) = \frac{1}{MN^2} \sum_{j=1}^{M} \text{trace}(\mathbf{R}_{\mathbf{h}}^{j}), \tag{11}$$

where the superscript in the index to the frame number. We will call (11) the multi-frame SfM (MFSfM) rate-distortion function, hereafter referred to as the video rate-distortion (VRD) function. Given a particular tolerable level of distortion, the VRD specifies the minimum number of frames necessary to achieve that level. In [23, 25] we proposed an alternative information theoretic criterion for evaluating the quality of a 3D reconstruction from video and analyzed the comparative advantages and disadvantages. The above result does not require the standard assumptions of Gaussianity of observations and is thus an extension of the error covariance results presented in [34].

2.2 SfM Algorithm for Face Reconstruction

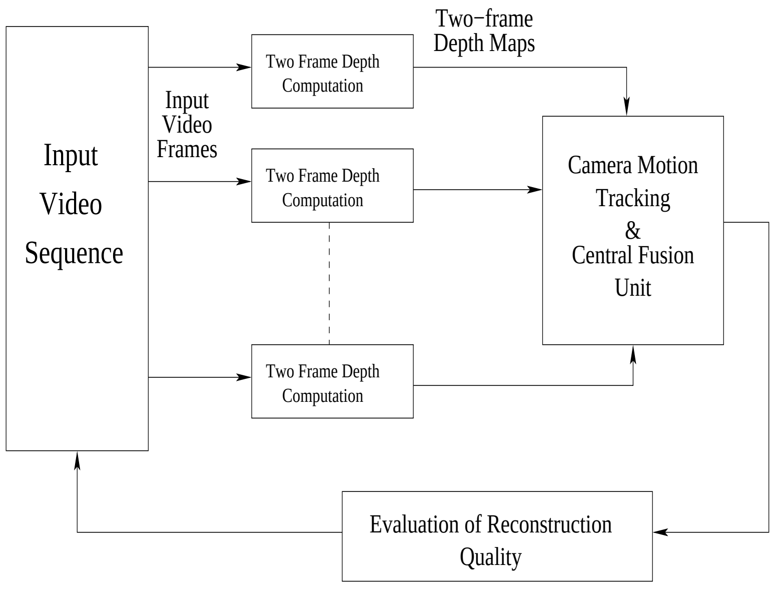

Two−frame Depth Maps
Two Frame Depth Computation
Input Video Frames
Input Video Sequence
Two Frame Depth Computation
Camera Motion Tracking & Central Fusion Unit
Two Frame Depth Computation
Evaluation of Reconstruction Quality

Figure 1: Block diagram of the 3D reconstruction framework.

Figure 1 shows a block-diagram schematic of the complete 3D face reconstruction framework using SfM. The input is a monocular video sequence. We choose an appropriate two-frame depth reconstruction strategy [30]. The depth maps are aligned to a single frame of reference and the aligned depth maps are fused together using stochastic approximation.

Let $\mathbf{s}^i \in \mathbf{R}^3$ represent the structure, [1] computed for a particular point, from i-th and $(i+1)$-st frame, $i = 1, ..., K$, where the total number of frames is $K+1$. [2] Let the fused structure sub-estimate at the i-th frame be denoted by $\mathbf{S}^i \in \mathbf{R}^3$. Let $\mathbf{\Omega}^i$ and $\mathbf{V}^i$ represent the rotation and translation of the camera between the i-th and $(i+1)$-st frames. Note that the camera motion estimates are valid for all the points in the object in that frame. The 3×3 rotation matrix $\mathbf{P}^i$ describes the change of coordinates between times i and $i+1$, and is orthonormal with positive determinant. When the rotational velocity $\mathbf{\Omega}$ is held constant between time samples, $\mathbf{P}$ is related to $\mathbf{\Omega}$ by $\mathbf{P} = e^{\hat{\mathbf{\Omega}}}$.[3] The fused sub-estimate $\mathbf{S}^i$ can now be transformed as $T^i(\mathbf{S}^i) = \mathbf{P}^i\mathbf{S}^i + \mathbf{V}^{iT}$. But in order to do this, we need to estimate the motion parameters $\mathbf{V}$ and $\mathbf{\Omega}$. Since we can determine only the direction of translational motion $(v_x/v_z, v_y/v_z)$, we will represent the motion components by the vector $\mathbf{m} = [\frac{v_x}{v_z}, \frac{v_y}{v_z}, \omega_x, \omega_y, \omega_z]$. Thus, the problems at stage $(i+1)$ will be to i) reliably track the motion parameters obtained from the two-frame solutions, and ii) fuse $\mathbf{s}^{i+1}$ and $T^i(\mathbf{S}^i)$. If $\{l^i\}$ is the transformed sequence of inverse depth values with respect to a common frame of reference, then the optimal value of the depth at the point under consideration is obtained as

$$u^* = \arg\min_u \text{median}_i \left(w_l^i (l^i - u)^2 \right), \tag{13}$$

where $w_l^i = (\mathsf{R}_{\mathbf{h}}^i(l))^{-1}$, with $\mathsf{R}_{\mathbf{h}}^i(l)$ representing the covariance of l^i (which can be obtained from (10)). However, since we will be using a recursive strategy, it is not necessary to align all the depth maps to a common frame of reference *a priori*. We will use a Robbins-Monro stochastic approximation (RMSA) [21] algorithm (refer to [24] for details) where it is enough to align the fused sub-estimate and the two-frame depth for each pair of frames and proceed as more images become available.

For each feature point, we compute $X^i(u) = w_l^i(l^i - u)^2, u \in \mathcal{U}$. At each step of the RM recursion, the fused inverse depth, $\hat{\theta}^{k+1}$, is updated according to [24]

$$\hat{\theta}^{k+1} = \hat{T}^k(\hat{\theta}^k) - a^k(p^k(\hat{\theta}^k) - 0.5), \tag{14}$$

where a^k is determined by a convergence condition , $p^k(\hat{\theta}^k) = \mathbf{I}_{[X^k \le \hat{T}^k(\hat{\theta}^k)]}$, $\mathbf{I}$ represents the indicator function, $\hat{T}^k$ is the estimate of the camera motion. When $k = K$, we obtain the fused inverse depth $\hat{\theta}^{K+1}$, from which we can get the fused depth value $\mathbf{S}^{K+1}$. The camera motion $\hat{T}$ is estimated using a tracking algorithm as described in [24].

The Reconstruction Algorithm: Assume that we have the fused 3D structure $\mathbf{S}^i$ obtained from i frames and the two-frame depth map $\mathbf{s}^{i+1}$ computed from the i-th and $(i+1)$-st frames. Figure 2 shows a block diagram of the multi-frame fusion algorithm. The main steps of the algorithm are:

Track Estimate the camera motion using the camera motion tracking algorithm [24].

Transform Transform the previous model $\mathbf{S}^i$ to the new reference frame.

Update Update the transformed model using $\mathbf{s}^{i+1}$ to obtain $\mathbf{S}^{i+1}$ from (14).

Evaluate Reconstruction Compute a performance measure for the fused reconstruction from (11).

Iterate Decide whether to stop on the basis of the performance measure. If not, set $i = i+1$ and go back to Track.

[1] In our description, subscripts will refer to feature points and superscripts will refer to frame numbers. Thus x_i^j refers to the variable x for the i-th feature point in the j-th frame.

[2] For notational simplicity, we use i and $(i+1)$-st frames to explain our algorithm. However, the method can be applied for any two frames provided the constraints of optical fbw are not violated.

[3] For any vector $\mathbf{a} = [a_1, a_2, a_3]$, there exists a unique skew-symmetric matrix

$$\hat{a} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}. \tag{12}$$

The operator $\hat{a}$ performs the vector product on $\mathbf{R}^3$: $\hat{a}X = \mathbf{a} \times X, \forall X \in \mathbf{R}^3$.
With an abuse of notation, the same variable is used for the random variable and its realization.

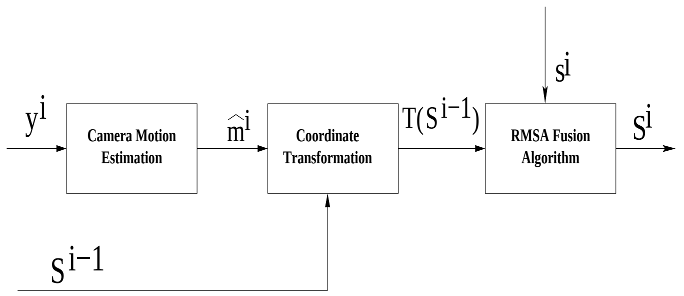


Figure 2: Block diagram of the multi-frame fusion algorithm.

2.3 Incorporating the Generic Face Model

The Optimization Function: The MFSfM estimate is smoothed using the generic model in an energy minimization framework. Both the generic model and the 3D estimate have a triangular mesh representation with N vertices and the depth at each of these vertices is known. (We will explain how this can be obtained later). Let $\{d_{g_i}, i = 1, ..., N\}$ be the set of depth values of the generic mesh for each of the N vertices of the triangles of the mesh. Let $\{d_{s_i}, i = 1, ..., N\}$ be the corresponding depth values from the SfM estimate. We wish to obtain a set of values $\{f_i, i = 1, ..., N\}$ which are a smoothed version of the SfM model, after correcting the errors on the basis of the generic mesh.

Since we want to retain the specific features of the face we are trying to model, our error correction strategy works by comparing local regions in the two models and smoothing those parts of the SfM estimate where the *trend* of the depth values is significantly different from that in the generic model, e.g. a sudden peak on the forehead will be detected as an outlier after the comparison and smoothed. This is where our work is different from previous work [11, 28], since we do not intend to fuse the depth in the two models but to correct errors based on local geometric trends. Towards this goal, we introduce a line process on the depth values. The line process indicates the borders where the depth values have sudden changes, and is calculated on the basis of the generic mesh, since it is free from errors. For each of the N vertices, we assign a binary number indicating whether or not it is part of the line process (see Figure 3(b)). This concept of the line process is borrowed from the seminal work of Geman and Geman [12] on stochastic relaxation algorithms in image restoration.

The optimization function we propose is

$$\begin{aligned} E(f_i, l_i) &= \sum_{i=1}^{N} (f_i - d_{s_i})^2 + (1-\mu) \sum_{i=1}^{N} (f_i - d_{g_i})^2 + \\ &\quad \mu \sum_{i=1}^{N} (1 - l_i) \sum_{j \in \mathcal{N}_i} (f_i - f_j)^2 \, \mathbf{1}_{d_s \neq d_g}, \end{aligned} \tag{15}$$

where $l_i = 1$ if the i^{th} vertex is part of a line process and μ is a combining factor which controls the extent of the smoothing. $\mathcal{N}_i$ is the set of vertices which are neighbors of the i^{th} vertex. $\mathbf{1}_{d_s \neq d_g}$ represents the indicator function which is 1 if the line process of d_s is not equal to the line process of d_g, else 0. (If $d_s = d_g$ point to point, their line processes will be equal, though the converse is not true). We will now explain the performance of the optimization function in the normal operating condition when the SfM estimate is “close” to the true face, but still contains some errors (mostly unwanted peaks and valleys). We have found experimentally that this is an usual characteristic of the SfM solution obtained from the first part of our modeling algorithm. In order to understand the importance of (15), consider the third term. When $l_i = 1$, the i^{th} vertex is part of a line process and should not be smoothed on the basis of the values in $\mathcal{N}_i$; hence this term is switched off. Any errors in the value of this particular vertex will be corrected on the basis of the first two terms, which control how close the final smoothed mesh will be to the generic one and the SfM estimate. When $l_i = 0$, indicating that the i^{th} vertex is not part of a line process, its final value in the smoothed mesh is determined by the neighbors as well as its corresponding values in the generic model and SfM estimate. The importance of each of these terms is controlled by the factor $0 < \mu < 1$. Note that μ is a function of d_s and d_g, as explained later when we discuss the choice of μ. This ensures that the optimization function converges to the true model in the ideal case that the SfM solution, d_s, computes the true model. For this case, the third term would be zero as the line processes would be the same, since they compare the trend in the depth values. The line process l_i has a value of 1 or 0, depending on whether there is a sudden change in the depth in the generic model at the particular vertex i. Obtaining such changes relies on derivative computations, which are known to be noise prone. Hence, in our optimization scheme, we allow the line process to be perturbed slightly around its nominal value computed from the generic model.

Besides, the normal operating condition and the ideal case discussed above, there are two other cases that may occur, though both are highly impractical. In the case where $d_s = d_g$, the second term is switched off because of μ and the third term is switched off because of the indicator function. The smoothed mesh can be either d_s or d_g as a result of (15). The other case arises if $d_s \neq d_g$, but their line processes are equal, i.e. $\mathbf{1}_{d_s \neq d_g} = 0$. In this case again the second and third terms are switched off and the optimization will converge to d_s.

The design of the cost function follows conventional ideas of regularization theory [7], whereby the energy function usually consists of a data term requiring the solution to be close to the data, and a regularizer which imposes a smoothness on the solution. Various other forms of the different terms in (15) could be considered. One interesting variation would be to impose a penalty term for the discontinuities. The graduated non-convexity algorithm of [1] is an appropriate method for solving such problems. It works by first finding the minimum of a convex approximation to the non-convex function, followed by minimization of a sequence of functions, ending with the true cost function. However, based on experimental analysis of the solution of our reconstruction algorithm, we decided that we could work with the simpler version of (15), which does not have the penalty term. This is because, during the optimization, the line process does not move very far from its nominal value, and hence, the penalty term does not have any significant contribution.

Equation (15) can be optimized in various ways. If the l_i are fixed, this is equivalent to solving a sparse linear system of equations (Chapters 9 and 10 of [13]). In the parlance of classical deterministic optimization, we then assume that we have perfect information about the loss function and that this information is used to determine the search directions in a deterministic manner in every step of the algorithm. However, as explained before, one of the major sources of noise in (15) is the estimate of l_i. If the l_i are not known perfectly, we need to optimize over this variable also. The cost function can no longer be represented as a linear system of equations. More complicated optimization schemes need to be investigated for this purpose.

We use the technique of simulated annealing built upon the Markov Chain Monte Carlo (MCMC) framework [8]. MCMC is a natural method for solving energy function minimization problems [7]. The MCMC optimizer is essentially a Monte Carlo integration procedure in which the random samples are produced by evolving a Markov chain. Let $T_1 > T_2 > ... > T_k > ...$ be a sequence of monotone decreasing temperatures in which T_1 is reasonably large and $\lim_{T_k \to \infty} = 0$. At each such T_k, we run N_k iterations of a Metropolis-Hastings (M-H) sampler [8] with the target distribution represented as $\pi_k(f,l) \propto \exp\{-E(f,l)/T_k\}$. As k increases, π_k puts more and more of its probability mass (converging to 1) in the vicinity of the global maximum of E. Since minimizing $E(f,l)$ is equivalent to maximizing $\pi(f,l)$, we will almost surely be in the vicinity of the global optimum if the number of iterations N_k of the M-H sampler is sufficiently large. The steps of the algorithm are:

- Initialize at an arbitrary configuration f_0, l_0 and initial temperature level T_1. Set $k = 1$.
- For each k, run N_k steps of MCMC iterations with $\pi_k(f,l)$ as the target distribution. Consider the following update strategy. For the line process, consider all the vertices (say $L < N$) for which the nominal value, $l_{i,nominal} = 1$, and their individual neighborhood sets, $N_1, ..., N_L$. For each $l_i = 1$, consider the neighborhood set among $N_1, ..., N_L$ that it lies in, randomly choose a vertex in this neighborhood set whose value is not already set to 1, and switch the values of l_i and this chosen vertex.

Starting from $l_i = l_{i,nominal}$, this process ensures that the values of l_i do not move too far from the nominal values. In fact, only the vertices lying in the neighborhood sets $N_1, ..., N_L$ can take a value of $l_i = 1$. Next, randomly determine a new value of f, using a suitable transition function [8]. With the new values, f_{new}, l_{new} of f, l, compute $\delta = E(f_{new}, l_{new}) - E(f, l)$. If $\delta < 0$, i.e. the energy decreases with this new configuration, accept f_{new}, l_{new}; else, accept with a probability ρ. Pass the final configuration of f, l to the next iteration.

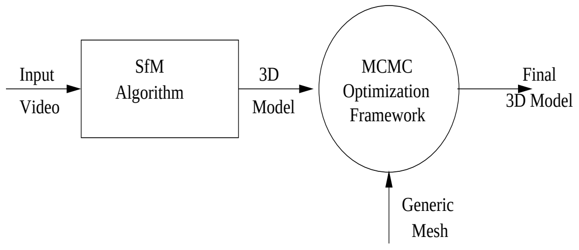
- Increase k to $k + 1$.

Mesh Registration: The optimization procedure described above requires a one-to-one mapping of the vertices $\{d_{s_i}\}$ and $\{d_{g_i}\}$. Once we obtain the estimate from the SfM algorithm, a set of corresponding points between this estimate and the generic mesh is identified. This can be done manually as in [11] or [28] or automatically as described next. This is then used to obtain a registration between the two models. Thereafter, using proper interpolation techniques, the depth values of the SfM estimate are generated corresponding to the (x, y) coordinates of the vertices of the triangles in the generic model. By this method, we obtain the meshes with the same set of N vertices, i.e. the same triangulation.

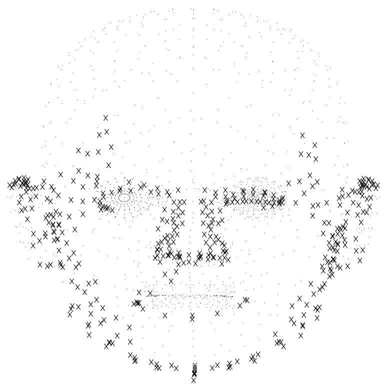
If we want to perform the registration automatically, we can follow a simple variant of our method for registering wide baseline images [26]. In that paper, we showed that it is possible to register two face images obtained from different viewing directions by considering the similarity of the shape of important facial features (e.g. eyes, nose etc.) and compensating for the variability of the shape with viewing direction by considering prior information about it. Applying it to this problem is actually simpler because the two meshes are from the same viewing angle. We can consider the 2D projection of the generic mesh and identify the shape of some important facial features a priori. Once the 3D estimate from SfM is obtained, we can take its 2D projection from the same viewing direction (e.g. from the front view), automatically extract the shape of the important features (as described in the paper by using a corner-finder algorithm and k-means clustering [9]) and then register by computing the similarity of the set of two shapes.

Choice of μ: There exists substantial literature on how to optimally choose the constant μ for energy functions similar to (15) (Chapter 3 of [7]). For our problem, we decided to choose the value based on a qualitative analysis of (15). When $l_i = 1$, the third term in the optimization is switched off. These are the points where there are sharp changes in the depth (see Figure 4(a)). These changes are important characteristics particular to a person's face and need to be retained. However, any errors in these regions should also be corrected. Thus μ is computed by comparing the line process obtained from d_s with that pre-computed from d_g. This gives an approximate idea about the goodness of the SfM estimate. From these considerations, the value of μ turns out to be between 0.7 and 0.8 (the correlation between the two line processes was usually in this range). When $l_i = 0$, the most important term is the third term of (15), which controls local smoothing. The errors should be corrected by comparing with neighboring values of the 3D estimate, rather than by fusing with the depth values of the vertices of the generic model. For this case, the above choice of μ is again reasonable and the generic model is not given undue importance leading to over-smoothing. This process of choosing μ also ensures that, in the ideal case, if d_s is the true face, the optimization converges to this value as μ will be equal to 1.

The Final Algorithm: The main steps of the algorithm for incorporating the generic mesh are are as follows.
1. Obtain the 3D estimate from the given video sequence using SfM (output of the reconstruction algorithm of Section 2.1).
2. Register this 3D model with the generic mesh and obtain the depth estimate whose vertices are in one-to-one correspondence with the vertices of the generic mesh.
3. Compute the line processes and to each vertex i assign a binary value l_i.



(a)

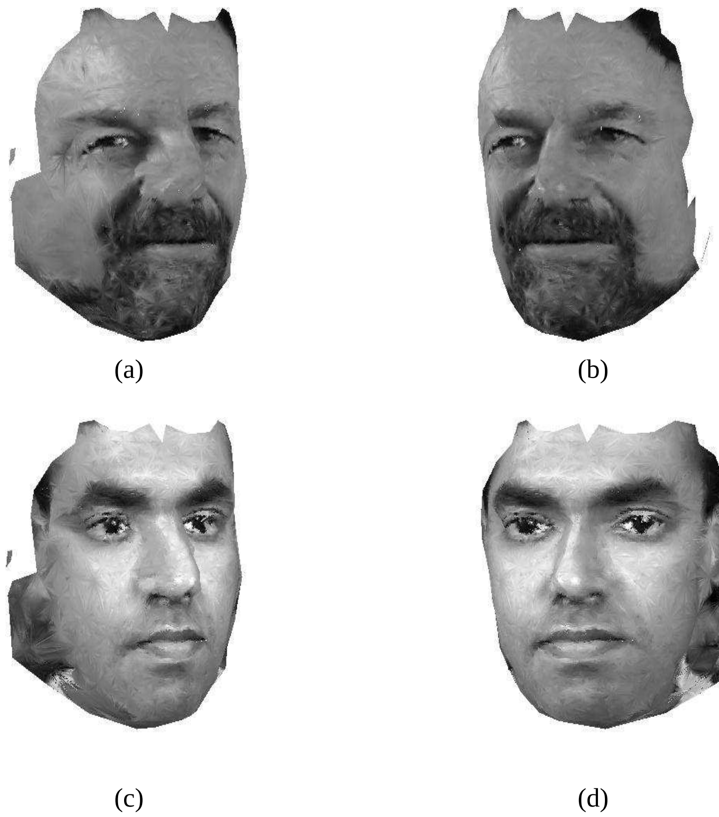


(b)

Figure 3: (a): A block diagram representation of the complete 3D modeling algorithm using the generic mesh and the SfM algorithm. (b): The vertices which form part of the line processes indicating a change in depth values are indicated with black 'x's.

4. Obtain the smoothed mesh f from the optimization function in (15).
5. Map the texture onto f from the video sequence.

The complete 3D reconstruction paradigm is composed of a sequential application of the two algorithms (3D Reconstruction Algorithm and the Generic Mesh Algorithm) we have described in Sections 2.1 and 2.3 (see Figure 3(a)). Some examples of 3D reconstruction are shown in Figure 4.



(a) (b)

(c) (d)

Figure 4: Different views of the two 3D models after texture mapping.

2.4 Experimental Evaluation

The SfM technique computed structure from optical flow using two frames [30] and then integrated the multiple two-frame reconstructions over the video sequence using robust estimation techniques. The error covariance of the optical flow was estimated a priori over the first few frames of the video sequence, which were not used in the reconstruction. It was done over a sampled grid of points (rather than the dense flow) so as to simplify calculations. The technique used was similar to the gradient-based method of [31], except that, for more accurate results, it was repeated for each of these initial frames and the final estimate was obtained using bootstrapping techniques [10]. This is the stage where the quality of the video data is estimated and incorporated into the algorithm. Assuming that the statistics remain stationary over the frames used in the reconstruction, the error covariance of the 3D reconstruction, $\mathbf{R_z}$, in (8), was computed. The quality evaluation of the fusion algorithm was done using the rate-distortion function of (11). The model obtained from the SfM algorithm is shown in Figure 5(b).

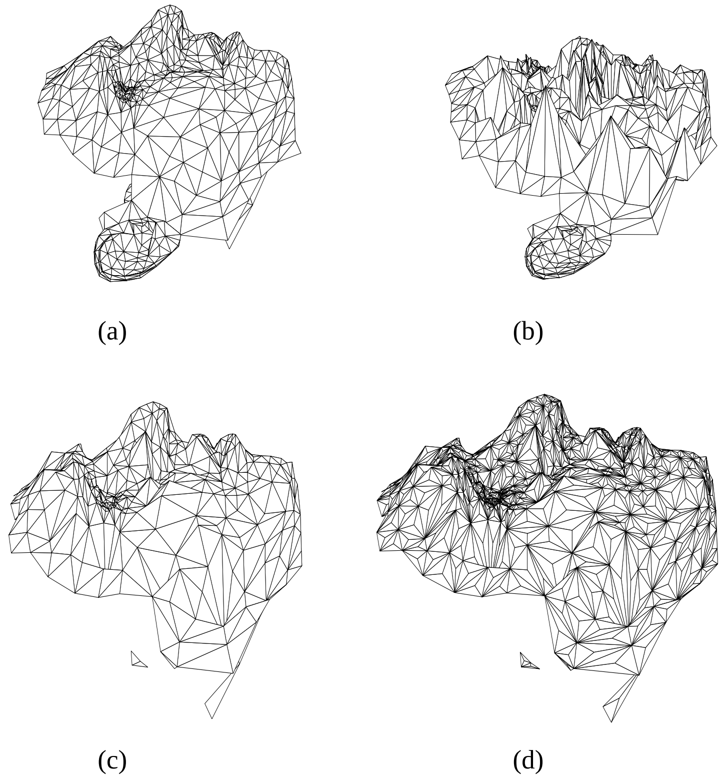


Figure 5: Mesh representations of the 3D models obtained at different stages of the algorithm. (a) represents the generic mesh, (b) the model obtained from the SfM algorithm (the ear region is stitched on from the generic model in order to provide an easier comparison between the different models), (c) the smoothed mesh obtained after the optimization procedure, (d) a fi ner version of the smoothed mesh for the purpose of texture mapping.

The combinatorial optimization function in (15) was implemented using the simulated annealing procedure based on a Metropolis-Hastings sampler. At each temperature we carried out 100 iterations and this was repeated for a decreasing sequence of 20 temperatures. Although this is much below the optimal annealing schedule suggested by Geman and Geman [12] (whereby the temperature T_k should decrease sufficiently slowly as $\mathcal{O}(\log(\sum_{i=1}^{k} N_i)^{-1}$, N_i being the total number of iterations at temperature T_i), it does give a satisfactory result for our face modeling example. This is because the optimization algorithm was initialized with the SfM reconstruction, which is usually a reasonable good estimate of the actual 3D model. We used a value of $\mu = 0.7$ in (15). The final smoothed model is shown in Figure 5(c).

Next, we map the texture onto the smoothed model in Figure 5(c). This is done using an image from the video sequence corresponding to the front view of the face. Direct mapping of the texture from the image is not possible since the large size of the triangles smears the texture over its entire surface. In order to overcome this problem, we split each of the triangles into smaller ones. This is done only at the final texture mapping stage. The initial number of triangles is enough to obtain a good estimate of the depth values, but not to obtain a good texture mapping. This splitting at the final stage helps us save a lot of computation time, since the depth at the vertices of the smaller triangles is obtained by interpolation, not by the optimization procedure [4]. The fine mesh onto which the texture is mapped is shown in Figure 5(d). Different views of the 3D model after the texture mapping are shown in Figure 4.

2.4.1 Computational Load

The computational complexity of the entire system can be analyzed by its individual parts. For the two-frame algorithm [30], given a $N \times N$ flow field, the complexity is $\mathcal{O}(N^2 \log N)$. For M frames, the complexity of the fusion algorithm is $\mathcal{O}(N^2 M)$. If the statistics are computed at $N' < N^2$ points, it takes computational power of the order of $\mathcal{O}(N'^2)$. The computational time for the final optimization will be determined by the actual annealing schedule chosen.

We have a running demonstration of the face modeling software. The software runs on a 2.6 Gigahertz P4 PC with 1 Gigabyte memory. A JVC DVL9800 video camera is attached to the PC to capture the video. Using a combination of C and MATLAB implementations, the entire reconstruction (from capturing the video sequence, pre-processing it to creating of a final 3D Graphics model) takes about 3-4 minutes. This can be substantially reduced by optimizing the code, converting it entirely to C and automating certain pre-processing stages (like identifying the relevant part of the input video sequence).

2.4.2 Accuracy Analysis of 3D Reconstructions

We have applied our algorithm to several video sequences. We present here the results on three such sequences. We will name the three people as subjects A, B and C. Reconstructed 3D models of Subjects A and C are shown in Figure 4. Since the line process and the neighborhood set are calculated from the generic model, they are pre-computed. We computed the accuracy of the 3D reconstruction for these three cases by comparing the projections of the 3D model with the images from the original video sequence. Figure 6 plots the root mean square (RMS) projection errors as a percentage of the actual values in the original images. In order to depict the change of the error with the viewing angle, the horizontal axis of the figure represents this angle, with 0 indicating the front view. We considered all the combinations between the three subjects, i.e. A-A, A-B, A-C, B-B, B-C and C-C. From the plots, we see that the average error in the reconstruction at the front view is about 1%. The error increases with viewing direction, as would be expected. Also there are certain preferred viewing directions, a fact which has been reported before in the literature [35]. In order to obtain an understanding of the role the depth (as opposed to the texture) plays in projections, we considered the case where the texture of B and C are overlayed on the model of A and the projections are compared to the images of B and C, respectively. The experiment was also repeated for the models of B and C with similar results; however, for the sake of clarity, the plots are not shown in Figure 6. The separation between the error curves for the different combinations holds out hope that 3D models can be used for recognition across pose variations. For example, given the model of Subject A with its proper texture, the projection errors, at any viewing angle, with other subjects is much more than it is with itself [5].

[4]This step may not be necessary with some of the recent computer graphics software.

[5]There are many issues that contribute to the error in face recognition across pose. Accuracy of the 3D model is only one of them. Others are issues of registration of the projections of the model with the image, changes of illumination etc. This is a separate research problem by itself and is one of our future directions of work. For our experiments, most of these issues were taken care of manually so that the error values represented in Figure 6 are mostly due to errors in 3D models (though errors due to other sources cannot be completely eliminated).

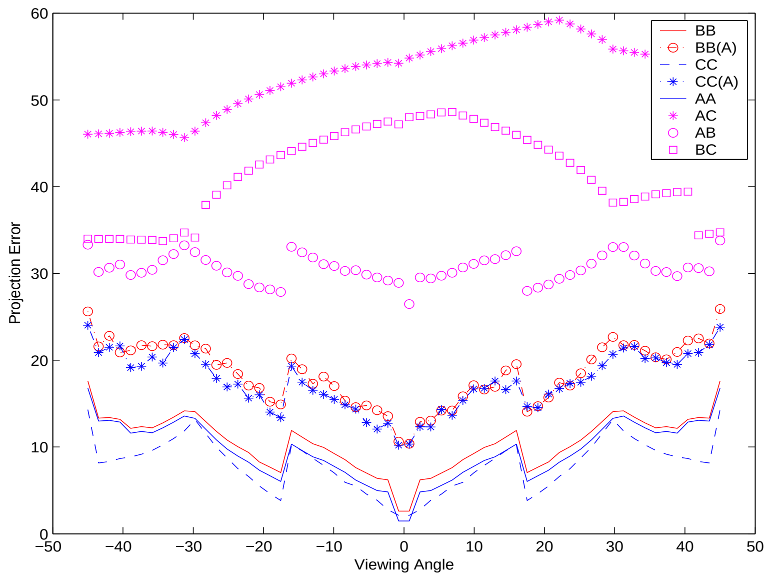


Figure 6: Percentage errors between the projections of the 3D models of subjects A, B and C and their images obtained from the video sequences, i.e. AA represents the error between the projections of 3D model of A and the images of A; similarly for AB, AC, BB, BC and CC. BB(A) represents the projection error when B's texture is overlayed on A's model and the projections are compared to B's images; similarly for CC(A). The error is plotted as a function of the viewing angle which is represented on the x-axis.

Table 1: Average Percentage Difference of the 3D models from the generic model

Subject Index	Percentage Difference
1 (frame 001)	10.2
2 (frame 002)	8.5
3 (frame 003)	4.6
4 (frame 004)	6.6
5 (frame 005)	6.9

2.4.3 Comparative Evaluation of Algorithm

In order to analyze the accuracy of the 3D reconstruction directly (as opposed to comparing the 2D projections), we require the ground truth of the 3D models. We experimented with a publicly available database of 3D models obtained from a Minolta 700 range scanner. The data is available on the World Wide Web at http://sampl.eng.ohio-state.edu/ sampl/data/3DDB/RID/minolta/faces-hands.1299/index.html. We will report numerical results from our algorithm on some of the data available here, though we will not publish the images or 3D models of the subjects.

In order to perform an accurate analysis of our methods, we require a video sequence of the person and the 3D depth values. This, however, is not available on this particular database or on any other that we know of. Thus we had to generate a sequence of images in order to apply our algorithm [6]. This was done using the 3D model and the texture map provided on the web-site. Given these images we performed the following experiments:

[6]The optical flow computed with the generated sequences may be more accurate than in a normal setting. However, for all the methods that are compared, the images used are the same. Hence, it is reasonable to assume that the effect of the image quality would be similar in all three cases. Hence the comparison of the 3D reconstruction accuracy, keeping all other factors constant, should still be useful.

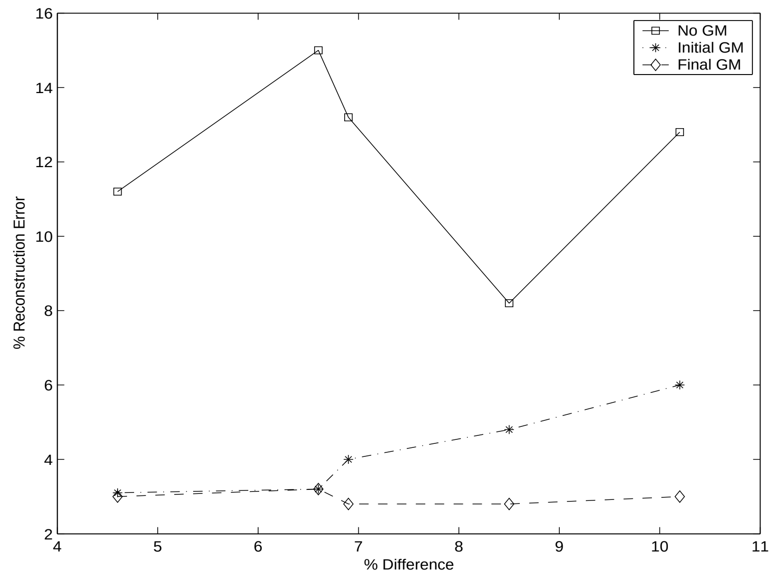


Figure 7: The error in 3D reconstruction when a) no generic model (GM) is used; b) the generic model is used to initialize the reconstruction algorithm; c) the generic model is used later as described in this section. The error is plotted as a function of the difference of the specifi c 3D model with the generic one. Five subjects were considered in this experiment.

- Obtain the 3D reconstruction without using the generic model (Section 2.1).
- Introduce the generic model at the beginning of the 3D reconstruction algorithm, by initializing (13) with the the values at the vertices of the generic model [7]. Note that the statistical error analysis of the video data is still done.
- Apply the algorithm in this section, which postpones the introduction of the generic model.

We considered the error in the 3D estimate in all these methods compared to the actual 3D values. Figure 7 plots the percentage RMS errors (percentage taken with respect to the true value) as a function of the percentage difference of the specific 3D model (as obtained from the website) and the generic one. The percentages on the horizontal axis are calculated with respect to the generic model, while those on the vertical axis are computed with respect to the ground truth of the 3D model of the particular subject. The first five subjects on that website, referred to as “frame001” to “frame005”, were considered. The percentage differences of the specific 3D models with the generic one are tabulated in Table 1. From the figure, it is clear that if the generic model is introduced at an early stage of the algorithm, the error in the reconstruction increases as the model of the subject deviates from the generic one. On the other hand, if the generic model is introduced later (as in our algorithm), the error in the reconstruction remains approximately constant. However, the reconstructions for the case where the generic model is introduced earlier (e.g. [11] and [28]) are visually very pleasing.

An idea of the progress of the optimization can be obtained from the following experiment, the results of which are shown in Figure 8. The experiment was done on Subject 1 of Table 1, which is an interesting case since the face is very different from the generic one. We considered ten significant points on the face (similar to fiducial points). They are the left eye, the bridge between the eyes, the right eye, the left extreme part of the nose, the tip of the nose, the right extreme part of the nose, the left and right ends of the lips and the center of the left and right cheeks. We considered a window around each of these points and computed the average depth in each of them for the various reconstructions. Figure 8 plots the average depth at each of

[7]We cannot compare it directly with [11] or [28] because of substantial differences in the input data.

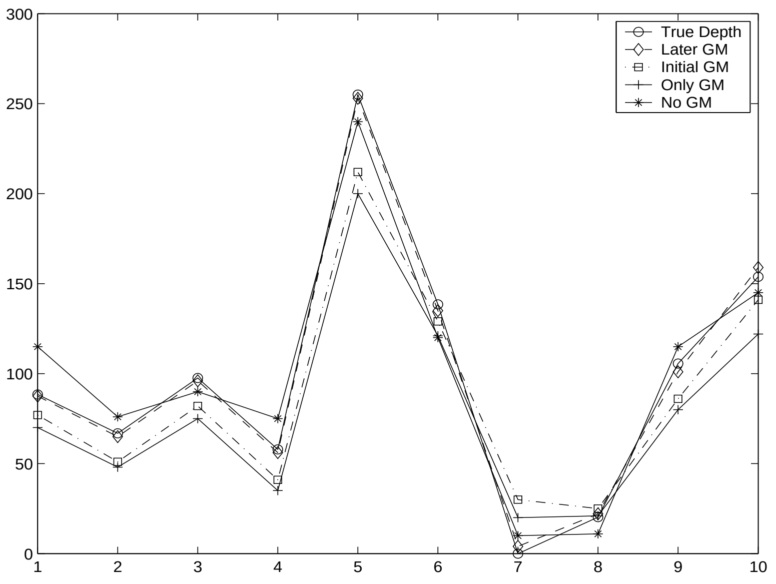


Figure 8: Plots of a) the true depth, b) the depth with generic model introduced later (our algorithm), c) the depth with generic model used as initial value, d) the depth of the generic model and e) the SfM reconstruction with no generic model. The depths are computed in local neighborhoods around a set of fi ducial points on the face for Subject 1.

these points for the following cases: true depth, depth with generic model introduced later (our algorithm), depth with generic model used as the initial value, depth of the generic model and the SfM reconstruction with no generic model. The depth values are normalized between 0 and 255, as in a depth map image. The plots provide some very interesting insights. When the reconstruction is initialized with the generic mesh, the solution at these points do not move very far from the initial value. On the other hand, the SfM estimate with no generic mesh gives a solution not very far from the true value, which is improved further by considering the generic model. Moreover, we find that the average error of the final reconstruction in these fiducial regions is less than the overall average error of Figure 7. This is very interesting since it shows that these significant regions of the face are reconstructed accurately.

3 Contour Based 3D Face Modeling

In this section, a novel 3D face modeling approach from a monocular video captured using a conventional camera is presented. The algorithm relies on matching a generic 3D face model to the outer contours of the face to be modeled and a few of its internal features. At the first stage of the method, the head pose is estimated by comparing the edges extracted from video frames with the contours extracted from a generic face model. Next, the generic face model is adapted to the actual 3D face by global and local deformations. An affine model is used for global deformation. The 3D model is locally deformed by computing the optimal perturbations of a sparse set of control points using a stochastic search optimization method. The deformations are integrated over a set of poses in the video sequence, leading to an accurate 3D model. Figure 9 shows a block diagram of the proposed algorithm.

3.1 Pose Estimation

In order to estimate the head pose without any prior knowledge about the 3D structure of the face, a generic 3D face model is used. Human shape variability is highly limited by both genetic and environmental constraints, and is characterized by a high degree of symmetry and approximate invariance of body lengths and ratios. Since the facial anthropometric measurements of the generic face are close to average, the algorithm can

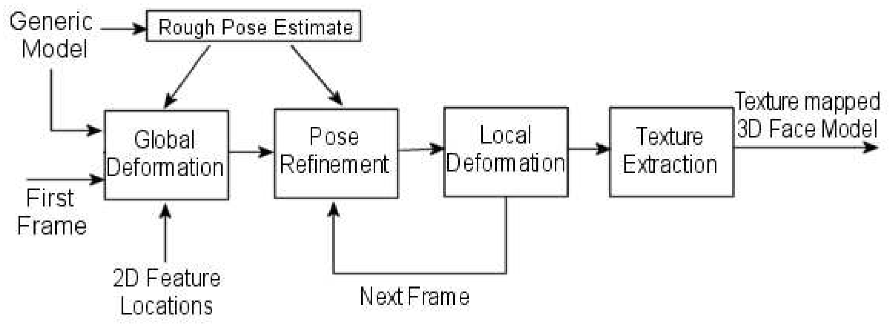


Figure 9: 3D face modeling algorithm

estimate the pose robustly, unless the person whose head pose is being estimated has a hugely deformed face. The problem of pose estimation along the azimuth angle is most commonly encountered in video sequences, and the system is currently limited to this case. The algorithm described in this section can be easily extended to incorporate more complex head motion (including roll and elevation). The addition of each degree of freedom of the head motion causes the search space for the face pose to grow exponentially, thus slowing down the 3D reconstruction algorithm at the pose estimation and pose refinement stages.

The frames extracted from the video are sub-sampled such that successive frames have a distinct pose variation. A simple image-difference method is used to detect the background pixels in the video. All edges in the background are removed to make sure they do not adversely affect the pose estimation algorithm. The pose estimation algorithm requires the coordinates of the nose tip in the image, because the edge maps obtained from the 2D projection of the 3D model and from the video frames are aligned at the nose tip. The Kanade-Lucas tracker [32] is used to automatically track the nose tip across multiple frames.

For pose estimation, an average human texture is mapped onto the generic 3D face model. An average 3D face shape and texture data was obtained from [22]. The texture mapped generic face model is rotated along the azimuth angle, and edges are extracted using the Canny edge detector [6]. The projection of the average 3D model also has edges which result from the boundaries of the 3D model (e.g. top of forehead, bottom of neck). These edges are not the result of natural contours of the human face, and are therefore automatically removed using a modified average texture. The edge maps are computed for 3D model rotation along the azimuth angle from -90° to $+90^\circ$ in increments of 5°. These edge maps are computed only once, and stored offline in an image array to make the procedure fast.

To estimate the head pose in a given video frame, the edges of the image are extracted using the Canny edge detector. Each of the scaled 3D model edge maps is compared to this frame edge map to determine which pose results in the best overlap of the edge maps. To compute the disparity between these edge maps, the Euclidean Distance Transform (DT) of the current video frame edge map is computed. For each pixel in the binary edge map, the distance transform assigns a number that is the distance between that pixel and the nearest nonzero pixel of the edge map. Figure 10 shows the binary edge map of a video frame, and the corresponding distance transform.

Each of the 3D model edge maps is aligned at the nose tip in the video frame, and the value of the cost function, F, is computed. The cost function, F, which measures the disparity between the 3D model edge map and the edges of the current video frame is of the form:

$$F = \frac{\sum_{(i,j)\in A_{EM}} DT(i,j)}{N} \tag{16}$$

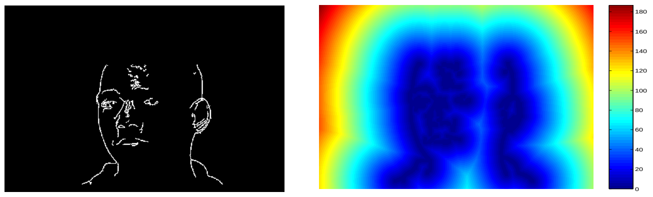


Figure 10: Left: Edge Map. Right: Distance Transform

where $A_{EM} \triangleq \{(i,j) : EM(i,j) = 1\}$ and N is the cardinality of set A_{EM} (total number of nonzero pixels in the 3D model edge map EM). F is the average distance transform value at the nonzero pixels of the binary 3D model edge map. The pose for which the corresponding 3D model edge map results in the lowest value of F is the estimated head pose for the current video frame. Figure 11 shows the head pose estimation results for a few video frames of a subject.

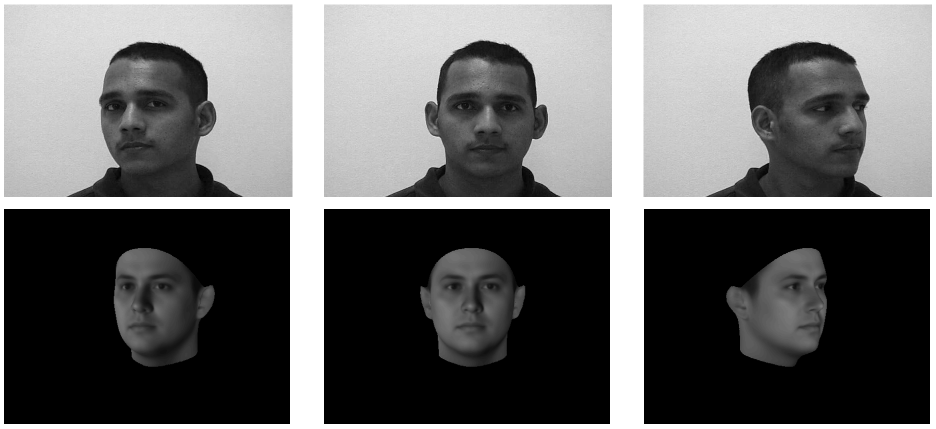


Figure 11: Top Row: Video Frames. Bottom Row: Generic model at estimated head pose

3.2 3D Face Model Reconstruction

In this section, the contour-based algorithm for 3D face reconstruction from a monocular video is described. A generic 3D face model is assumed to be the initial estimate of the true 3D face model. The generic face model is globally and locally deformed to adapt itself to the actual 3D face of the person.

3.2.1 Registration and Global Deformation

Once the pose is estimated using the method described in Section 3.1, the next step is to perform a global deformation and register the 3D model to the 2D image. A scaled orthographic projection model for the camera, and an affine deformation model for the global deformation of the generic face model is used. The coordinates of four feature points (left eye, right eye, nose tip, and mouth center) are used to determine a solution for the affine parameters. Since these feature point locations need to be known for just the first frame, currently these are marked manually. The 3D coordinates of the corresponding feature points on the generic face model are available beforehand.

The generic 3D model is globally deformed only for the first frame. The following affine model is used for global deformation:

$$\begin{bmatrix} X_{\text{gb}} \\ Y_{\text{gb}} \\ Z_{\text{gb}} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ 0 & 0 & \frac{a_{11}+a_{22}}{2} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ 0 \end{bmatrix}, \tag{17}$$

where (X, Y, Z) are the 3D coordinates of the vertices of the generic model, and subscript "gb" denotes global deformation. The affine model appropriately stretches/shrinks the 3D model along the the X, and Y axes and also takes into account the shearing in the X-Y plane. Considering the basic symmetry of human faces, the affine parameters contributing to shearing (a_{12}, a_{21}) are very small, and can often be neglected. Since an orthographic projection model is used, we can not have an independent affine deformation parameter for the Z-coordinate. The affine deformation parameters are obtained by minimizing the reprojection error of the 3D feature points on the rotated deformed generic model, and their corresponding 2D locations in the current frame. The 2D projection (x_f, y_f) of the 3D feature points (X_f, Y_f, Z_f) on the deformed generic face model is given by

$$\begin{bmatrix} x_f \\ y_f \end{bmatrix} = \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \end{bmatrix}}_{R_{12}} \begin{bmatrix} a_{11}X_f + a_{12}Y_f + b_1 \\ a_{21}X_f + a_{22}Y_f + b_2 \\ (\frac{a_{11}+a_{22}}{2})Z_f \end{bmatrix}, \tag{18}$$

where R_{12} is the matrix containing the top two rows of the rotation matrix corresponding to the estimated head pose for the first frame. Using the coordinates of the four feature points, (18) can be reformulated into a linear system of equations. The affine deformation parameters $\mathbf{P} = [a_{11}, a_{12}, a_{21}, a_{22}, b_1, b_2]^T$ can be determined by obtaining a Least-Squares (LS) solution of the system of equations. The generic mesh is globally deformed according to these parameters. This process ensures that the 3D face model matches the approximate shape of the face and the significant internal features are properly aligned.

3.2.2 Local Deformation

To adapt the 3D model to a particular individual's face more accurately from the video sequence, local deformations are introduced in the globally deformed model. The algorithm for local deformation of the face model is described in this section.

Control Points The globally deformed dense face mesh is sampled at a small number of points to obtain a sparse mesh shown in Figure 12. Each of the vertices of this sparse mesh is a control point in the optimization procedure. Each control point is imparted a random perturbation in the X, Y, and Z direction. The perturbation for each of the vertices of the dense face mesh is computed from the random perturbations of control points using triangle-based linear interpolation. The computed perturbations are imparted to the vertices of the face mesh to obtain a locally-deformed mesh. The outer contours obtained from the locally-deformed face mesh are compared with the edges from the current video frame. The optimum local perturbations of the control points are determined using a Direct Random Search method.

Direct Random Search Direct Random Search methods [29] are based on exploring the domain D in a random manner to find a point that minimizes the cost function L. They are "direct" in the sense that the algorithms use minimal information about the cost function $L = L(\boldsymbol{\theta})$. The minimal information is essentially only the input-output data of the following form: input $= \boldsymbol{\theta}$, output $= L(\boldsymbol{\theta})$. In the Global Random Search method, we repeatedly sample over D such that the current sampling for $\boldsymbol{\theta}$ does not take into account the previous samples. The domain D is a hypercube (a p-fold Cartesian product of intervals on the real line, where p is the problem dimension), and we use uniformly distributed samples. There exists a convergence proof [29] which shows that the two-step (after initialization) algorithm described below converges almost surely to the global minimum $\boldsymbol{\theta}^*$.

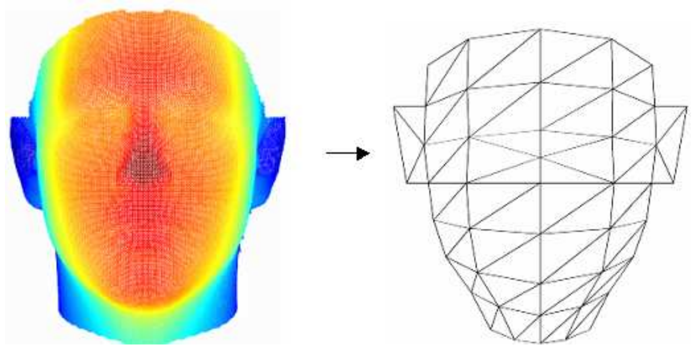


Figure 12: Sparse mesh of control points

Global Random Search Algorithm:
Step 0 (initialization): Generate an initial value of $\boldsymbol{\theta}$, say $\hat{\boldsymbol{\theta}}_0 \in D$, according to the uniform probability distribution on the domain D. Calculate $L(\hat{\boldsymbol{\theta}}_0)$. Set $k = 0$.
Step 1: Generate a new independent value of $\boldsymbol{\theta} \in D$, say $\boldsymbol{\theta}_{new}(k+1)$, according to the uniform probability distribution. If $L(\boldsymbol{\theta}_{new}(k+1)) < L(\hat{\boldsymbol{\theta}}_k)$, set $\hat{\boldsymbol{\theta}}_{k+1} = \boldsymbol{\theta}_{new}(k+1)$. Else take $\hat{\boldsymbol{\theta}}_{k+1} = \hat{\boldsymbol{\theta}}_k$.
Step 2: Stop if the maximum number of L evaluations has been reached; else return to Step 1 with $k = k+1$.

In our application of the Global Random Search algorithm to achieve the optimum local deformation of the face model, the cost function L is the disparity between the outer contours obtained from the 3D face model after applying the local deformation $(\boldsymbol{\theta})$ and the edges from the current video frame. The stochastic optimization algorithm determines the value of $\boldsymbol{\theta}$ that minimizes the cost function L. The parameter vector $\boldsymbol{\theta}$ whose optimum value is being sought is of the form:

$$\boldsymbol{\theta} = [\Delta X_c \; \Delta Y_c \; \Delta Z_c]_{P\times 3}, \tag{19}$$

where ΔX_c, ΔY_c, and ΔZ_c are the perturbations in the X, Y, and Z directions of the P control points. The perturbations for all vertices of the dense face mesh, $[\Delta X_{\text{vt}} \; \Delta Y_{\text{vt}} \; \Delta Z_{\text{vt}}]$, are computed from the perturbations of these control points using a triangle-based linear interpolation method. The coordinates of the vertices of the locally deformed face model are given by,

$$\begin{bmatrix} X_{\text{lc}} \\ Y_{\text{lc}} \\ Z_{\text{lc}} \end{bmatrix} = \begin{bmatrix} X_{\text{gb}} \\ Y_{\text{gb}} \\ Z_{\text{gb}} \end{bmatrix} + \begin{bmatrix} \Delta X_{\text{vt}} \\ \Delta Y_{\text{vt}} \\ \Delta Z_{\text{vt}} \end{bmatrix}, \tag{20}$$

where the subscript "lc" denotes local deformation. Let $EM_{\boldsymbol{\theta}}$ be the binary edge map (outer contours) of the 2D projection of the 3D model after applying local deformation $\boldsymbol{\theta}$. The unwanted edges due to the boundaries of the 3D model are removed by the method described earlier in Section 3.1. Let DT be the distance transform of the edge map of the current video frame. The cost function, L, is of the form

$$L(\boldsymbol{\theta}) = \frac{\sum_{(i,j)\in A_{EM_{\boldsymbol{\theta}}}} DT(i,j)}{N}, \tag{21}$$

where $A_{EM_{\boldsymbol{\theta}}} \triangleq \{(i,j) : EM_{\boldsymbol{\theta}}(i,j) = 1\}$ and N is the cardinality of set $A_{EM_{\boldsymbol{\theta}}}$. The structure of this cost function is similar to the one used in (16). In Section 3.1, the edge maps were compared to determine the head pose, but here these are compared to determine the optimal local deformation for the globally deformed generic face model.

Multi-Resolution Search The random perturbations are applied to the face mesh at two different resolutions. Initially, M iterations of large random perturbation are performed to scan a large search area at a coarse

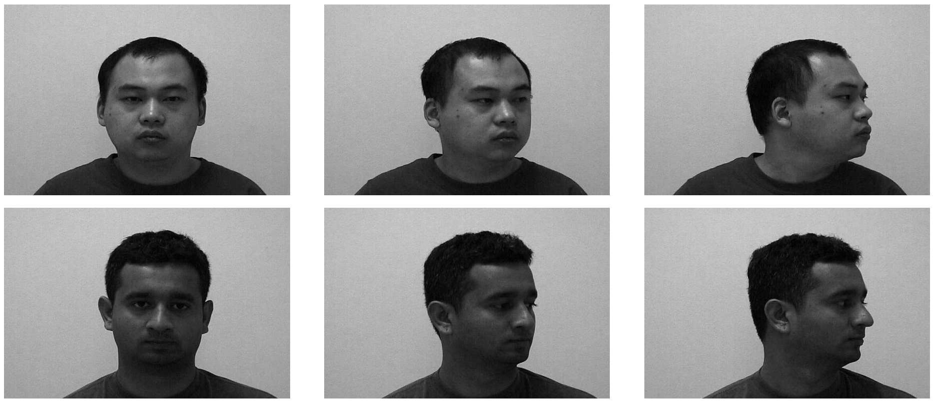


Figure 13: Video Frames. Top Row: Subject D. Bottom Row: Subject E.

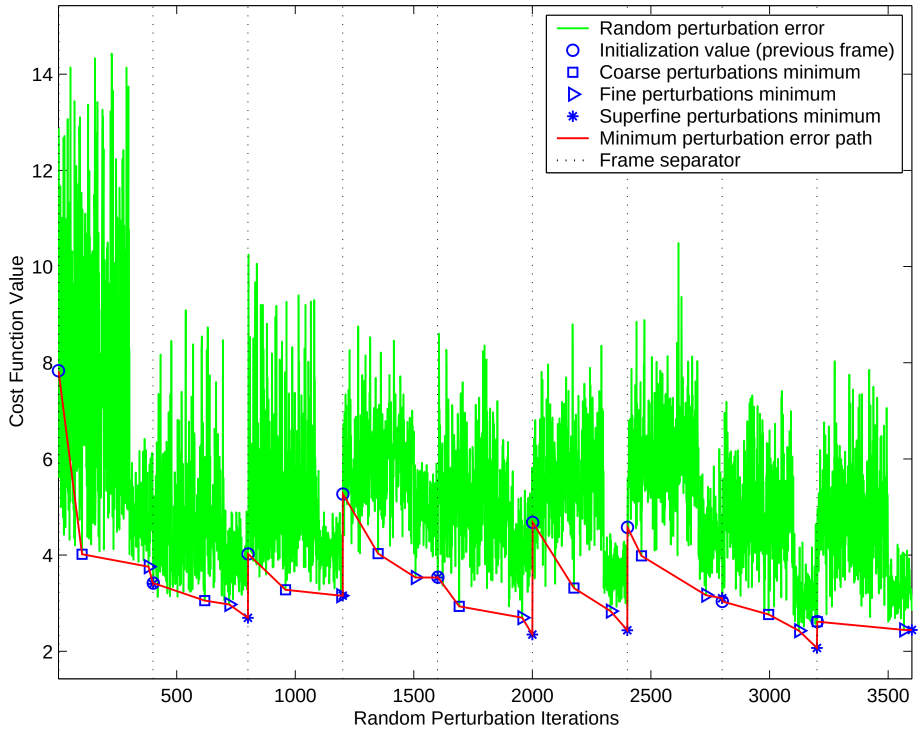


Figure 14: Perturbation errors at each stage of the multi-resolution search for subject D

resolution. The final estimate at the end of M coarse-level iterations is denoted by $\hat{\boldsymbol{\theta}}^{c}_{M}$. Once a coarsely deformed mesh which is close to the global minimum is available, it is used as the starting estimate for the fine resolution iterations, i.e. $\hat{\boldsymbol{\theta}}^{c}_{M} = \hat{\boldsymbol{\theta}}^{f}_{0}$. N small random perturbations are applied to the coarsely deformed mesh to get even closer to the actual solution. The final estimate at the end of N fine-level iterations is denoted by $\hat{\boldsymbol{\theta}}^{f}_{N}$. This coarse to fine resolution search helps the algorithm converge to the solution much faster, compared to a search strategy where the search space is sampled at a fixed fine resolution. As the number of iterations performed for the coarse and fine resolution searches is increased, the performance of the algorithm gets better, but, at the cost of computational time. It was observed that the algorithm converges to a good solution in 300 iterations of large (coarse) random perturbations, and 100 iterations of small (fine) random perturbations.

The final step to fine tune the control point perturbations estimate is an exhaustive search for the optimal perturbation, $\hat{\boldsymbol{\theta}}^{sf}_{N}(i)$, for each individual control point, i, in a small neighborhood of the solution, $\hat{\boldsymbol{\theta}}^{f}_{N}(i)$, obtained from fine resolution iterations. The perturbations of all other control points are fixed to the value determined by $\hat{\boldsymbol{\theta}}_N^f$. The perturbation that results in the minimum value of the cost function L, is chosen to be the optimal perturbation $\hat{\boldsymbol{\theta}}_N^{sf}(i)$ for that particular control point. The superscript "*sf*" denotes super-fine estimate. Ideally, the exhaustive search sould be performed in a combinatorial manner (instead of individually for each control point) over the entire search space, but the computational complexity for this strategy would be prohibitive.

Constraints A few constraints are imposed on the values of perturbations that are imparted to each of the control points. Without any constraints, the algorithm will also search in the domain of unrealistic faces, thus unnecessarily slowing down the algorithm. The following constraints are imposed on the perturbations of the control points:

- Symmetry along the vertical axis passing through nose tip, based on the fact that most human faces are symmetric about this axis.
- The maximum possible perturbation for the control points is defined.
- The perturbation of a few control points is dependent on the perturbations of their neighboring control points.
- Only control points whose movement might alter the contours of the 3D model (and hence change the cost function) are perturbed.

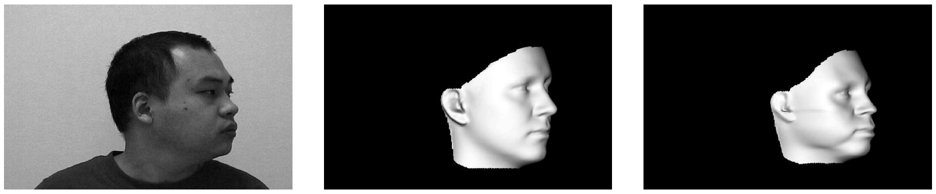


Figure 15: Left: Video frame of subject. Middle: Generic 3D model. Right: Reconstructed 3D model

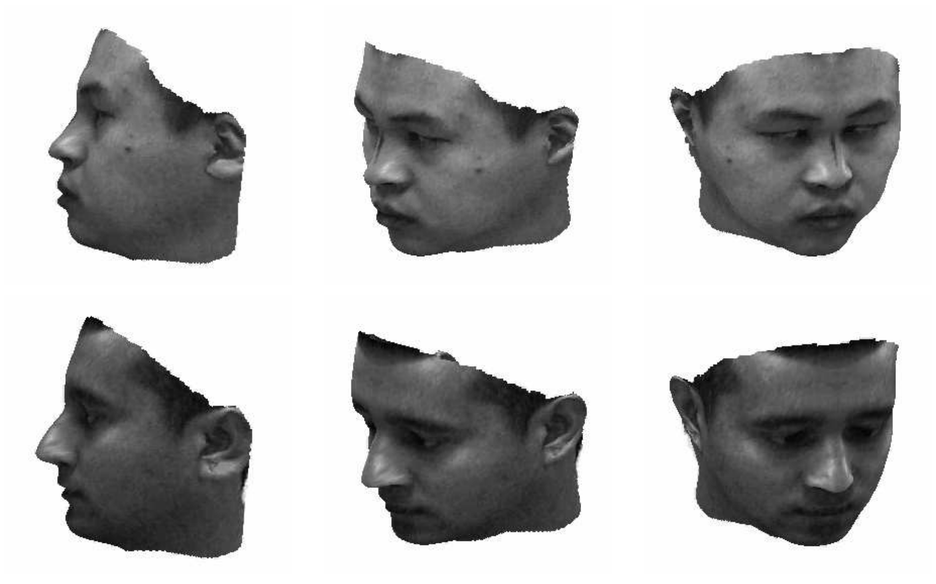


Figure 16: Reconstructed 3D face model. Top Row: Subject D. Bottom Row: Subject E.

3.2.3 Pose Refinement and Model Adaptation across time

Once an adapted 3D model from a particular frame is available, the rough pose estimate (obtained using a generic model) of the next frame is refined to obtain a more accurate head pose estimate. The method used for this pose refinement is similar to the method described in Section 3.1, except that the contours to be compared to the edges of the current frame are extracted from the adapted 3D model upto that stage, instead of the generic face model. For the first video frame, the adapted 3D model used for pose refinement is just the globally deformed generic model. But, for later frames, the globally and locally adapted 3D model is used for pose refinement. This pose refinement step is critical to the shape estimation procedure because the described contour-based algorithm is very sensitive to the head pose estimate. If the pose estimate is not accurate, the 3D model (at the wrong pose) will adapt itself in an inappropriate manner so that its contours conform with the edges extracted from the video frame. To refine the head pose estimate, contours are obtained from the adapted 3D face model rotated about azimuth angles in a small neighborhood of the rough pose estimate (obtained using the method in Section 3.1). The angle which results in 3D model contours closest to the edges extracted from the current frame is chosen as the refined pose estimate. The similarity criterion used to compare the edge maps is the Distance Transform based measure described in Equation (16).

Once the refined pose estimate and the adapted (global and local) 3D face model are available from the first video frame, the algorithm for pose refinement and local deformation described earlier is applied recursively to all the subsequent frames to improve the quality of the reconstructed 3D face model. Integrating the algorithm across several frames makes the system robust to any noise in the contours extracted from a particular frame. The adapted 3D model from the previous frame is used as the starting estimate for the next frame. As more frames with varying head poses become available, the system gets valuable cues to model certain aspects of the face more accurately (e.g. the side view models the structure of the nose better than the front view). Figure 15 shows a video frame, the initial generic 3D model and the final reconstructed 3D face model of the subject.

3.2.4 Texture Extraction

Once the pose refinement and the 3D model adaptation are completed for all the frames, the texture of the person's face is extracted and mapped onto the adapted 3D model for visualization. A single frame is used for texture extraction, because it was found from our experiments that texture sampling across multiple frames smears the texture slightly. The smearing occurs because of the fact that the registration (using estimated pose) across multiple frames is not exact. Now the question arises as to which frame to choose for texture extraction? Based on studies that face recognition systems perform best on 3/4 profile view [5], we believe that this view is most representative of the person. Hence, the frame closest to 3/4 profile view ($\pm 45^\circ$) was chosen for texture extraction. In a 3/4 profile view, part of the face would be occluded for one half of the face. To get the texture for this half of face (part of which is occluded), symmetry of face texture about the vertical axis passing through the nose tip is assumed.

3.3 Experimental Results

In this section, the experimental results of the contour-based 3D face modeling algorithm are presented. We have a current implementation of the algorithm in MATLAB, using video frames of size 720x480, on a 1.66 GHz Pentium 4 machine. No effort has been made yet to optimize the algorithm. Since the head motion is assumed to be only along the azimuth angle, the top and bottom pixels of the face region were marked out in the first frame, and were assumed to be constant across all video frames. All the edge maps extracted from the average 3D model were resized according to this scale. Also, only the edges in the region between the marked out top and bottom pixels of the face region are retained, thus removing the hair and shoulder edges. Figure 13 shows a few video frames of two subjects whose face modeling results have been presented. Figure 14 shows the plot of perturbation errors (value of cost function L) at each stage of the multi-resolution search across all frames of the video sequence for Subject D. It can be observed from the plot that, within each frame, the minimum perturbation error decreases as the search resolution gets finer. Figure 16 shows the reconstructed texture mapped 3D face model of the subjects from different viewpoints. Extensive experimentation on a number of real video sequences has been conducted, with good results.

4 Conclusions

In this chapter, two algorithms for 3D face modeling from a monocular video have been presented. The first method works by creating an initial estimate using multi-frame SfM, which is then refined by comparing against a generic face model. The comparison is carried out using an energy function optimization strategy. Statistical measures of the quality of the input video are evaluated and incorporated into the SfM reconstruction framework. Bias of the final reconstruction towards the generic model is avoided by incorporating it in the latter stages of the algorithm. This is validated through experimental evaluation. Results of the 3D reconstruction algorithm, along with an analysis of the errors in the reconstruction, are presented. The second method presented in this chapter reconstructs a face model by adapting a generic model to the contours of the face over all the frames of a video sequence. The algorithm for pose estimation and 3D face reconstruction relies solely on contours, and the system does not require knowledge of rendering parameters (e.g light direction and intensity). Results and anlysis of this algorithm, which does not rely on finding accurate point correspondences across frames, is presented.

References

[1] A. Blake and A. Zisserman. *Visual Reconstruction*. MIT Press, 1987.

[2] M. Brady and A.L. Yuille. An extremum principle for shape from contour. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(3):288–301, May 1984.

[3] M. Brand. Morphable 3D models from video. In *Proc. of IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, pages II:456–463, 2001.

[4] C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3D shape from image streams. In *Proc. of IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, pages II:690–696, 2000.

[5] V. Bruce, T. Valentine, and A. Baddeley. The basis of the 3/4 view advantage in face recognition. *Applied Cognitive Psychology*, 1:109–120, 1987.

[6] J. Canny. A computational approach to edge detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8:679–698, November 1986.

[7] J.J. Clark and A.L. Yuille. *Data Fusion for Sensory Information Processing Systems*. Kluwer, 1990.

[8] A. Doucet, N. deFreitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

[9] R. Duda, P. Hart, and D. Stork. *Pattern Classification (2nd Edition)*. John Wiley and Sons, 2001.

[10] B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, 1993.

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

[12] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 6(6):721–741, November 1984.

[13] G. Golub and C.V. Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, 1989.

[14] H. Gupta, A. Roy-Chowdhury, and R. Chellappa. Contour based 3d face modeling from a monocular video. In *British Machine Vision Conference*, 2004.

[15] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[16] T.S. Jebara and A.P. Pentland. Parameterized structure from motion for 3d adaptive feedback tracking of faces. In *Proc. of IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, pages 144–150, 1997.

[17] W. Matusik, C. Buehler, R. Raskar, S.J Gortler, and L. McMillan. Image-based visual hulls. In *Computer Graphics Proceedings, Siggraph 2000, New Orleans, LA*, pages 369–374, July 2000.

[18] B. Moghaddam, J. Lee, H. Pfister, and R. Machiraju. Model-based 3-D face capture with shape-from-silhouettes. In *IEEE International Workshop on Analysis and Modeling of Faces and Gestures, Nice, France*, pages 20–27, October 2003.

[19] Vishwjit Nalwa. *A Guided Tour of Computer Vision*. Addison-Wesley, 1993.

[20] M. Pollefeys. *Self-calibration and metric 3D reconstruction from uncalibrated image sequences*. PhD Thesis, ESAT-PSI, K.U.Leuven, 1999.

[21] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[22] S. Romdhani, V. Blanz, and T. Vetter. Face identification by fitting a 3d morphable model using linear shape and texture error functions. In *Proc. of European Conference on Computer Vision*, page IV: 3 ff, 2002.

[23] A. Roy Chowdhury. *Statistical Analysis of 3D Modeling From Monocular Video Streams*. PhD Thesis, Univeristy of Maryland, College Park, 2002.

[24] A. Roy Chowdhury and R. Chellappa. Stochastic approximation and rate-distortion analysis for robust structure and motion estimation. *International Journal of Computer Vision*, 55(1):27–53, October 2003.

[25] A. Roy-Chowdhury and R. Chellappa. An information theoretic criterion for evaluating the quality of 3D reconstructions from video. *IEEE Trans. on Image Processing*, pages 960–973, July 2004.

[26] A. Roy Chowdhury, R. Chellappa, and T. Keaton. Wide baseline image registration with application to 3D face modeling. *IEEE Transactions on Multimedia*, pages 423–434, June 2004.

[27] A.K. Roy Chowdhury and R. Chellappa. Face reconstruction from monocular video using uncertainty analysis and a generic model. *Computer Vision and Image Understanding*, 91:188–213, July 2003.

[28] Y. Shan, Z. Liu, and Z. Zhang. Model-based bundle adjustment with application to face modeling. In *Proc. of International Conf. on Computer Vision*, pages 644–651, 2001.

[29] J.C. Spall. *Introduction to Stochastic Search and Optimization*. Wiley, 2000.

[30] S. Srinivasan. Extracting structure from optical flow using fast error search technique. *International Journal of Computer Vision*, 37:203–230, 2000.

[31] Z. Sun, V. Ramesh, and A.M. Tekalp. Error characterization of the factorization method. *Computer Vision and Image Understanding*, 82:110–137, May 2001.

[32] C. Tomasi and J. Shi. Good features to track. In *Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recogition, Jerusalem, Israel*, pages 593–600, October 1994.

[33] R. Walter. *Principles of Mathematical Analysis, 3rd Edition*. McGraw-Hill, 1976.

[34] G.S. Young and R. Chellappa. Statistical analysis of inherent ambiguities in recovering 3D motion from a noisy flow field. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14:995–1013, October 1992.

[35] W. Zhao, R. Chellappa, P.J. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *ACM Transactions*, 35:399–458, December 2003.