Multimed Tools Appl
DOI 10.1007/s11042-011-0925-8

3D face and motion estimation from sparse points using adaptive bracketed minimization

Varin Chouvatut · Suthep Madarasmi · Mihran Tuceryan

Abstract This paper presents a novel method for estimating camera motion and reconstructing human face from a video sequence. The coarse-to-fine method is applied via combining the concepts of Powell's minimization with gradient descent. Sparse points defining the human face in every frame are tracked using the active appearance model. The case of occluded points, even for self-occlusion, does not pose a problem in the proposed method. Robustness in the presence of noise and 3D accuracy using this method is also demonstrated. Examples of face reconstruction using other methods including trifocal tensor, Powell's minimization, and gradient descent are also compared to the proposed method. Experiments on both synthetic and real faces are presented and analyzed. Also, different camera movement paths are illustrated. All real-world experiments used an off-the-shelf digital camera carried by a human walking without using any dolly to demonstrate the robustness and practicality of the proposed method.

Keywords Camera pose · Gradient descent · Model reconstruction · Powell's multidimensional minimization

1 Introduction

Three-dimensional (3D) face model provides an obvious advantage in the field of face recognition since unseen viewpoints of the face can be rendered after having its 3D model. Model reconstruction and camera pose (position and orientation) estimation can be viewed

V. Chouvatut (✉) · S. Madarasmi
Computer Engineering Department, King Mongkut's University of Technology Thonburi, Bangkok, Thailand
e-mail: varin@cpe.kmutt.ac.th

S. Madarasmi
e-mail: suthep@kmutt.ac.th

M. Tuceryan
Computer and Information Science Department, Indiana University-Purdue University Indianapolis, Indianapolis, IN 46202-5132, USA
e-mail: tuceryan@cs.iupui.edu

Published online: 10 November 2011

as two problems: 1) 3D model reconstruction and 2) estimation of camera pose. Camera pose is often estimated [13, 15, 21] by placing artificial markers with known 3D-positions in the scene. Unfortunately, placing markers on a human face is impractical while measuring the relative 3D-position of markers cannot be done precisely. Thus, the method proposed here uses natural feature points that appear on the face without requiring prior knowledge of 3D-positions. The feature points are tracked using the Active-Appearance Model (AAM) [7].

Auto-calibration [10] is a commonly used approach to computing structure from motion using points without known 3D-positions. Here, the fundamental matrix is used when given only two frames [10]; the trifocal tensor given three [1]; the quadrifocal tensor given four [11]; and the factorization approach given multiple frames [14]. Metric reconstruction must be achieved in order to recover the structure of the target object. To obtain metric reconstruction, projective reconstruction must be solved first, after which the affine reconstruction may also be needed. Metric reconstruction may not be accurate or even fail, if the projective or affine reconstruction provides erroneous results. Furthermore, missing or occluded image-points between the frames must be identified. Due to the complexity from the multiple steps and strategies needed for such correspondence search, the method proposed in this paper achieves 3D-reconstruction where missing points including self-occlusion do not pose a problem.

Gradient descent is known to be a robust optimization approach to many applications including estimating camera motion [20], training neural networks [12], and searching for motion of an image point [4, 17]. Generic gradient descent requires calculating the conjugate direction [4] or the derivative of the cost function to define the direction and size for variable updating. The derivative may suggest a step-size and direction that causes the overall system to jump in the wrong direction with a bad data set propagated in the iterations to follow, making the system take rather long to converge. Thus, our proposed method applies gradient descent without requiring a derivative calculation while the execution time is even shorter.

Multidimensional problems can be solved by Powell's minimization with paraboloid bracketing as in [18, 22]. The concept of line minimization used in Powell's method is used in this paper. The multidimensional problem with many unknown variables can be viewed as search for the minimum point in the valley of a 3D-paraboloid. When considering a single dimension or line, the deepest point along the line may not lead to the global minimum solution. Thus, instead of walking deep down until the minimization of each dimension is found like in the line minimization approach, the proposed method chooses to climb down the valley with only a small step-size in a limited bracket corresponding to the appropriate line direction, providing much better results as shown in section 4.

For face reconstruction, Zheng et al. [24] reconstructed 3D-face from stereo images and a reference 3D face model. The reference model was used since the traditional stereo approach based on intensity image could not provide a good result. To find correspondences between the stereo images, non-linear deformations and camera registration need to be solved. Later, Park et al. [16] used a single image with an average 3D face model generated from 3D training images. A triangular mesh of the 3D-face is created via Delaunay triangulation, an approach also used in this paper. Reconstructing the 3D-model from a single image would appear to provide a big advantage; however, a basis 3D-model of the target object must first be available. Zheng and Wang [23] used only one frontal image to generate a 3D-face but a database of the detected image feature point depths is required.

Clearly, using more available information enables a more optimal solution that is more robust to noise and occlusion as shown in sections 3.2 and 3.3. If only a few images were used, self-occlusion and noisy data can pose a problem as demonstrated in section 4 (comparison to the trifocal tensor [11]).

2 Methodology

2.1 The proposed method

In this paper, the only input required is a video sequence recording the target object(s) to be reconstructed. The output is a 3D-model of the object, the camera motion, and one focal length. All videos are recorded with the frame rate of 15 frames/s. Videos of synthetic experiments are generated from 3D Studio Max whereas the real-world videos are recorded by an off-the-shelf digital camera carried by a human walking around the object without using any dolly. The 3D reconstruction results composed of 3D-objects and several cones representing camera poses are displayed in the OpenGL environment.

For all real-world experiments, feature points defining the target object seen in video frames are tracked with sub-pixel precision using AAM. Since AAM cannot solve the problem of missing points, a video sequence is separated into two or more parts so that each point is visible in all frames of a part. For example, the first part of the video may show only the left side of a face while the second part shows only the right side. The random sample consensus (RANSAC) [8, 11] based on the relationship between the fundamental matrix $F_{3\times3}$ and point correspondences $\mathbf{x}'_i, \mathbf{x}_i$ of point i in 3D space is further used to eliminate outliers as:

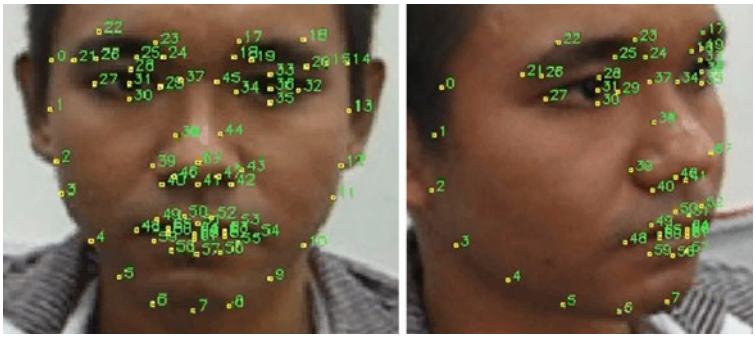
$$\mathbf{x}'^{T}_{i} F \mathbf{x}_i = 0. \tag{1}$$

The RANSAC algorithm can be outlined as:

1) Repeat for K samples.
   a) Select a sample of seven correspondences and compute F from (1). There will be one or three real solutions.
   b) Calculate the displacement d of each correspondence.
   c) Count the number of inliers where $d<\varepsilon$, for a given threshold ε.
   d) If there are three solutions, choose the one with the most number of inliers.
2) Choose F with the most number of inliers. In the case of a tie, choose the one with the lowest standard deviation σ of inliers.

Normally, RANSAC will remove an equal number of outliers out of both frame1 and frame2 to keep the number of points in both frames equal. Here, the inliers and outliers categorized by RANSAC will be defined as two groups and marked with a flag ℓ where $\ell \in \{0, 1\}$. All inliers will have $\ell = 1$ whereas the outliers will have $\ell = 0$. An outlier can get flag $\ell_i = 1$ if and only if the outlier i has flag $\ell_i = 1$ in either the previous or the next frame, e.g. the outlier i in frame1 is an inlier in frame0 or the outlier i in frame2 is an inlier in frame3. Note that the feature of allowing an outlier of frame j to get flag $\ell = 1$ must not be propagated to frames $j \notin [j-1, j+1]$, i.e. the original categories from RANSAC should be used to consider this feature.

A total of 68 sparse points are used to define a human face as shown in Fig. 1(a). An example of points missing due to the self-occlusion is shown in Fig. 1(b). The 2D-points defining the target object seen in video frames can be called image coordinates. Generally, projecting a 3D-



(a) 68 points defining facial parts (b) Self-occlusion example

Fig. 1 Feature points used to define facial parts and an example of self-occlusion. **a** 68 points defining facial parts. **b** Self-occlusion example

point in the real-world to a 2D-point can be done by: 1) from the 3D world-coordinate to 3D camera-coordinate, and 2) from the 3D camera-coordinate to 2D image-coordinate. Let the world coordinates of a point be $\mathbf{W} = \begin{bmatrix} X_w & Y_w & Z_w & 1 \end{bmatrix}^T$, the point's camera coordinates be $\mathbf{C} = \begin{bmatrix} X_c & Y_c & Z_c & 1 \end{bmatrix}^T$, and its image coordinates be $\mathbf{S} = \begin{bmatrix} u & v & 1 \end{bmatrix}^T$. Transforming the world coordinates to the camera coordinates can be written as:

$$\mathbf{C} = M^{w \to c}\mathbf{W} = [R|\mathbf{T}]\mathbf{W} \tag{2}$$

The transformation matrix $M^{w \to c}$ is generated from

$$M^{w \to c} = R_x^{-1} R_y^{-1} R_z^{-1} T^{-1} \tag{3}$$

where R and T are rotation and translation matrices, respectively.

Projecting a camera coordinate to its image coordinate is given by:

$$\mathbf{S} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c/Z_c \\ Y_c/Z_c \\ 1 \end{bmatrix} \tag{4}$$

where f is the focal length.

Projecting a camera coordinate to its image coordinate can be rewritten using an arbitrarily user-defined unit such as centimeters for the 3D-geometry:

$$\begin{bmatrix} u(pixel) \\ v(pixel) \\ 1 \end{bmatrix} = \begin{bmatrix} f(pixel) \times \{X_c(cm)/Z_c(cm)\} \\ f(pixel) \times \{Y_c(cm)/Z_c(cm)\} \\ 1 \end{bmatrix} \tag{5}$$

From (5), f is implied from:

$$\begin{bmatrix} u(pixel) \\ v(pixel) \\ 1 \end{bmatrix} = \begin{bmatrix} q(cm) \times \left\{ \dfrac{\mathrm{Img}_w(pixel)}{CCD_x(cm)} \right\} \times \left\{ \dfrac{X_c(cm)}{Z_c(cm)} \right\} \\ q(cm) \times \left\{ \dfrac{\mathrm{Img}_h(pixel)}{CCD_y(cm)} \right\} \times \left\{ \dfrac{Y_c(cm)}{Z_c(cm)} \right\} \\ 1 \end{bmatrix} \tag{6}$$

where Im g_w and Im g_h are the image's width and height, CCD_x and CCD_y are the width and height of camera's aperture size, and q is the length of the focal point of the camera's lens measured in world coordinates.

Given a point seen in a frame, two parameters of the image point (u, v) provide two equations from (2)–(5). Thus, given N points visible in M frames, $2MN$ equations are available. The unknowns include six variables per frame for camera pose (θ_x, θ_y, θ_z, t_x, t_y, t_z), three per point for world coordinates (X_w, Y_w, Z_w), and one for focal length (f). Thus, the $6M + 3N + 1$ unknowns can be solved if $2MN \geq 6M + 3N + 1$.

The proposed method aims to minimize the errors calculated from the displacement between the observed image-points and the reprojected ones. The error of a point can be calculated from:

$$d_i^j = \sqrt{\ell_i \times \left(u_i^j - \widetilde{u}_i^j\right)^2 + \ell_i \times \left(v_i^j - \widetilde{v}_i^j\right)^2} \tag{7}$$

where d_i^j is displacement between the observed point $\begin{bmatrix} u & v & 1 \end{bmatrix}^T$ and the reprojected point $\begin{bmatrix} \widetilde{u} & \widetilde{v} & 1 \end{bmatrix}^T$ of the point i seen in frame j.

In this paper, all average errors are calculated using Root Mean Square (RMS). Thus, the average error in the unit of pixels per point per frame (missing points must be excluded from this calculation) is the global energy to be minimized. The RMS errors ($\overline{d}$) used as the local and global energies are calculated from:

$$\overline{d} = \sqrt{\frac{\sum\limits_{i=0;j=0}^{i=N;j=M} \left(d_i^j\right)^2}{M \times N}} \tag{8}$$

The system's convergence can be checked from:

$$\sqrt{\frac{\sum\limits_{i=0;j=0}^{i=N;j=M} \left(d_i^j - \overline{d}\right)^2}{M \times N}} < 1.0. \tag{9}$$

The process for the proposed reconstruction starts from motion interpolation explained in section 2.3. Then, the coarse results of reconstruction estimation from section 2.3 will be further calculated by the following main steps to obtain a fine reconstructed solution.

The main steps of the proposed method are:

1) Initialize all variables' step-sizes as (10). The value initialized to all step-sizes is the bracket of variation for all variables. The methodology for applying the bracketing method to one-dimensional search problems can be found in [3].

$$\begin{aligned} & p_t - \alpha_t < p_{t+1} < p_t + \alpha_t \\ & \alpha_t = \gamma \end{aligned} \tag{10}$$

where $\gamma \in \{1.0, 2.0\}$; p_t and α_t are the variable and its step-size, respectively, at time t. The γ should not be too large a number so as to control the volume of 3D space as normalized but significant, so that quick convergence can be achieved. However, this does not mean that the reconstructed volume will be limited to γ.

2) Repeat steps 3–5 for each variable until the global convergence or a maximum number of iterations.

3) Try updating the variable using its step-size by

$$\begin{aligned} p_{t+1}[1] &= p_t + \alpha_t \\ p_{t+1}[2] &= p_t - \alpha_t. \end{aligned} \tag{11}$$

Choose one of the two p_{t+1}'s which gives larger decrease of local energy after checking both of them in step 4. This concept makes the system try to move to the direction which gives larger decrease of system's energy to speed up the convergence while the largest decrease should be avoided as recommended in Powell's bracketing [18]. Defining a small value for γ in (10) also plays an important role for this feature. Equations (11) and (14) avoid wasting time and memory in trying bracketing and line minimization over all previous directions that require the use of multidimensional arrays such as in Powell's method [18].

4) Check the local energy from:

a) If the energy decreases,

$$\alpha_{t+1} = \alpha_t. \tag{12}$$

b) Otherwise:

$$\begin{aligned} p_t &= p_{t-1} \\ \alpha_{t+1} &= b \times \alpha_t \end{aligned} \tag{13}$$

where $0.0 < b < 1.0$. Then, back to step 3 if $\alpha_{t+1} > \varepsilon$.

5) If $\alpha_{t+1} \leq \varepsilon$,

$$\alpha_{t+1} = \gamma. \tag{14}$$

For clarification, some important aspects of the algorithm are further explained here. Section 2.2 explains this further. From step 1, each variable has an individual step-size, the maximum of 1.0 (or 2.0) to keep the system distance and parameter's variation small. Keeping the distance and variation small helps fast convergence. Though the calculated distance of the system is small, it can be rescaled later by the known real-world distance. Note that coordinate transformation explained in section 2.4 needs to be done for error measurement. Since all relative distances can be controlled to the unit size, the focal length is initialized to a large but reasonable value. From (5) and (6), q is very small in the real-world but f, which is calculated by q multiplied by Img_w/CCD_x, is large. The camera pose of the first frame can be initialized to $[R|\mathbf{T}] = [I|\mathbf{0}]$. Camera pose of the other frames will be relative to the first frame, so those camera parameters can also be initialized to 0's.

The step-size represents a small step of walking down the valley of a paraboloid to the system convergence, in accordance with Powell's line minimization with paraboloid bracketing [18]. Instead of finding a bracket and the bottom of the paraboloid from line minimization which may be calculated from a trivial data set (not yet the optimal solution), our proposed method finds a small and limited bracket (discarding the direction of the largest decrease of energy) calculated from the current data set. Consequently, a good initial guess for variables is not critical for our approach to work well. Since the derivative directions are not used as in the gradient descent, both positive and negative step-sizes are tried in step 3 to control the promising direction to the system convergence. However, we still search for the largest possible step-size (but still in a limited bracket) that can decrease the energy in step 4. Resetting the step-size to its maximum bracket may be needed at times as in step 5 to prevent the system from getting stuck at a local minimum.

From step 2, the order of variables for line minimization is also considered to speed up the system's convergence to a globally optimal solution. Variables with a tendency of high variation will be considered first. So the variables with a smaller variation will maintain their low value (close to zero) while the variables with higher variation are first updated. Here, the world coordinates of all points are considered first: X_w, Y_w, and Z_w. Next, the focal length, f. After that, the camera pose: θ_y, t_x, t_z, θ_x, t_y, and θ_z. We use a right-hand coordinate system for the camera coordinates. One may notice that the camera usually has a large range of orientation in the y axis, especially when the camera pans the object from side to side. Panning the camera to a face usually involves a rotation in the y direction and translations in the x and z directions. The next important parameter is the orientation around the x-axis as the camera may roll slightly. When the camera rolls, a translation in y direction may also occur to keep the face present in the frames. Finally, the variable that usually has the least variation is the orientation around the z-axis.

2.2 Constraints for the proposed method

After updating a variable, certain criteria to double-check constraints on each variable are required to prevent unacceptable values for the next iteration in step 2.

1) *The camera pose* of any two contiguous frames should have a small difference, i.e.

$$\left| p_t^j - p_t^{j-1} \right| < \beta \tag{15}$$

where p_t^j is the camera's variable of frame j at time t and β is the threshold. For example, thresholds for R and T may be 30.0 and 20.0, respectively. If (15) does not meet,

$$\begin{aligned} p_t^j &= p_t^{j-1} \\ \alpha_{t+1} &= \gamma \end{aligned} \tag{16}$$

where γ is defined in (10).

2) *The 3D-point* in the world coordinate is insured to be in front of the camera by forcing:

$$Z_w = -|Z_w|. \tag{17}$$

3) *The focal length*, f, must be positive:

$$\begin{aligned} &\text{c) if } f < 0.0; \quad f = |f|, \\ &\text{d) if } f = 0.0; \quad f = 1.0. \end{aligned} \tag{18}$$

2.3 Motion interpolation

Since the camera usually has small changes in motion between contiguous frames of a video sequence, the parameters of camera motion including R and $\mathbf{T}$ can be initially estimated from interpolation. Using interpolation, the system can save hundreds to thousands of iterations for estimating camera parameters. Consequently, the system can reach much lower global energy than not using any interpolation, given the same processing time. Linear interpolation (Lerp) and cubic-spline interpolation are then applied to the geometry reconstruction of camera motion.

Assuming that the interpolation is done in the Lagrange form [18] in this section, given a set of N data points (x_i, y_i) where $i = 0, \ldots, N-1$ and no two x_i are the same; a linear combination of the Lagrange formula can be written as

$$L(x) \equiv \sum_{i=0}^{N-1} y_i l_i(x) \tag{19}$$

where

$$l_i(x) \equiv \prod_{\substack{0 \le m < N \\ m \ne i}} \frac{x - x_m}{x_i - x_m}. \tag{20}$$

Lerp is a simple form of interpolation and it is often used due to its low complexity [2]. The Lerp used in this paper is

$$y = y_i + \left(\frac{x - x_i}{x_{i+1} - x_i}\right)(y_{i+1} - y_i). \tag{21}$$

Cubic spline interpolation is an interpolation that is smooth in the first derivative and continuous in the second derivative [18]. Since the second derivative of the cubic spline is not known in this case, the natural cubic spline with the zero curvature condition at the endpoints of the spline, i.e. $y_0'' = y_{N-1}'' = 0.0$, is used. The condition $y_0'' = 0.0$ is then used to find y_1'', the calculated y_1'' will then be used to find y_2'', and so on. Specifically, $y_i'' = F(y_{i-1}'')$. On the other hand, y_{i+1}'' will be back-substituted to y_i'' also, specifically, $y_i'' = F(y_{i-1}'')$. The spline function used is

$$\begin{aligned} y &= Ay_i + By_{i+1} + Cy_i'' + Dy_{i+1}'' \\ A &\equiv \frac{x_{i+1} - x}{x_{i+1} - x_i} \\ B &\equiv 1 - A = \frac{x - x_i}{x_{i+1} - x_i} \\ C &\equiv \frac{1}{6}\left(A^3 - A\right)(x_{i+1} - x_i)^2 \\ D &\equiv \frac{1}{6}\left(B^3 - B\right)(x_{i+1} - x_i)^2. \end{aligned} \tag{22}$$

The first and second derivatives derived from (22) with respect to x are:

$$\frac{dy}{dx} = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} - \frac{3A^2 - 1}{6}(x_{i+1} - x_i)y_i'' + \frac{3B^2 - 1}{6}(x_{i+1} - x_i)y_{i+1}'', \tag{23}$$

$$\frac{d^2y}{dx^2} = Ay_i'' + By_{i+1}''. \tag{24}$$

As mentioned that the first derivative must be continuous between two intervals, Eq. (23) evaluated for $x = x_i$ in the interval (x_{i-1}, x_i) must be equal to the equation evaluated for $x = x_i$ in (x_i, x_{i+1}) for $i = 1, \ldots, N-1$; this gives

$$\frac{x_i - x_{i-1}}{6}y_{i-1}'' + \frac{x_{i+1} - x_{i-1}}{3}y_i'' + \frac{x_{i+1} - x_i}{6}y_{i+1}'' = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} - \frac{y_i - y_{i-1}}{x_i - x_{i-1}} \tag{25}$$

and is used to solved for N unknowns $y_i'', i = 0, \ldots, N-1$. However, the natural cubic spline has $y_0'' = y_{N-1}'' = 0.0$.

The process of interpolation is as the main steps explained in section 2.1 except:

1) Add the first step for initialization to all parameters at step 1 of section 2.1 so

$$p_t = 0.0. \tag{26}$$

All p_t's are initialized as (26) except

$$\begin{aligned} Z_w &= -1.0 \\ f &= \beta \end{aligned} \tag{27}$$

where $\beta > 0.0$. Note that, β should be a reasonable number as implied from (5)-(6). Assigning a constant value to Z_w, as shown in (27), means all points in 3D space are put on a plane in front of a camera, parallel to the camera's plane.

2) The bracket of variation γ from (10) for camera's parameters $[R|\mathbf{T}]$ is allowed a wider range as

$$\gamma = a \times \gamma \tag{28}$$

where $a \in \Re^+$, e.g. if a=50.0 is used, it means that the camera can move and orient by 50.0 times wider between two contiguous key frames than between two contiguous frames. The problem of key frame selection has already been studied and can be found in [9, 19].

3) Main loop of step 2 in section 2.1 is done for all parameters but the camera's parameters are applied only at the key frames. The intermediate frames are obtained by the interpolation explained previously.

4) The local energy (step 4 in section 2.1) of a key-frame parameter, $E(p_k)$ where $k = 0, \ldots, N-1$, is calculated from all of its intermediate frames i's, i.e.

$$E(p_k) = \sum_{k-1<i<k+1} E(p_i) \tag{29}$$

for the intermediate key-frames and

$$E(p_k) = \begin{cases} \sum_{0 \le i < k+1} E(p_i) \\ \sum_{k-1 < i \le N-1} E(p_i) \end{cases} \tag{30}$$

for the key frames at boundaries.

2.4 Coordinate transformation

As explained earlier, the normalized 3D geometry of the reconstructed result $\mathbf{p}$ can be transformed to the real-world geometry $\widetilde{\mathbf{p}}$ by

$$\widetilde{\mathbf{p}}_{i,j} = \mathbf{p}_{i,j} \times \left\| \frac{\mathbf{b}_1 - \mathbf{b}_0}{\mathbf{a}_1 - \mathbf{a}_0} \right\| \tag{31}$$

where $\mathbf{a}_i$ is 3D coordinate of the ith obtained point, $\mathbf{b}_i$ is 3D coordinate of the ith real-world point, and $\mathbf{p}_{i,j} = \mathbf{a}_i - \mathbf{a}_j$.

The final geometry of the results obtained from this proposed method is in a Euclidean geometry whose origin can be anywhere. Converting the obtained coordinate to an arbitrary coordinate can be done by

$$\widetilde{\mathbf{t}}_i = \mathbf{t}_i - \mathbf{t}_0 \tag{32}$$

where $\mathbf{t}_i$ is the ith position corresponding to the coordinate origin. The new coordinate with position $\widetilde{\mathbf{t}}$ can then be oriented in the order of z, y, and x axis respectively to the final coordinate. The following is the transformation for orientation according to the first camera coordinate $\mathbf{C}_0$ in the order of roll, yaw, and pitch for α, φ, and ϑ degrees, respectively, to the coordinate $\mathbf{t}'_i$ by:

$$\begin{aligned} t'_1 &= t_1 \cos\varphi \cos\alpha + t_2 \cos\varphi \sin\alpha - t_3 \sin\varphi \\ t'_2 &= t_1(\sin\vartheta \sin\varphi \cos\alpha - \cos\vartheta \sin\alpha) \\ &\quad + t_2(\sin\vartheta \sin\varphi \sin\alpha + \cos\vartheta \cos\alpha) + t_3 \sin\vartheta \cos\varphi \\ t'_3 &= t_1(\cos\vartheta \sin\varphi \cos\alpha + \sin\vartheta \sin\alpha) \\ &\quad + t_2(\cos\vartheta \sin\varphi \sin\alpha - \sin\vartheta \cos\alpha) + t_3 \cos\vartheta \cos\varphi. \end{aligned} \tag{33}$$

The obtained coordinate can then be transformed to a new coordinate of the first camera $\mathbf{C}''_\mathbf{0}$ with pitch, yaw, and roll of θ, ϕ, and ψ, respectively. So

$$\begin{aligned} t''_1 &= t_1 \cos\psi \cos\phi - t_2(\sin\psi \cos\theta - \cos\psi \sin\phi \sin\theta) \\ &\quad + t_3(\sin\psi \sin\theta + \cos\psi \sin\phi \cos\theta) \\ t''_2 &= t_1 \sin\psi \cos\phi + t_2(\cos\psi \cos\theta + \sin\psi \sin\phi \sin\theta) \\ &\quad - t_3(\cos\psi \sin\theta - \sin\psi \sin\phi \cos\theta) \\ t''_3 &= -t_1 \sin\phi + t_2 \cos\phi \sin\theta + t_3 \cos\phi \cos\theta. \end{aligned} \tag{34}$$

Finally, the new coordinate from (34) may also be applied to $\mathbf{C}'''_0$ whose $\mathbf{T} = \mathbf{t}''_0$, so the final coordinate will be:

$$\mathbf{t}'''_i = \mathbf{t}_i + \mathbf{t}'''_0. \tag{35}$$

3 Experiments

3.1 Human faces

The facial features for reconstruction are represented by 68 sparse points. After computing the 3D-values of these 68 points, a 3D triangular mesh is generated by Delaunay triangulation followed by a texture mapping. Three face texture images including the frontal, left-side, and right-side views of faces are used. Figure 2 shows example frames from videos of a synthetic face and four real faces. Reprojection errors after 3D reconstruction are shown in Table 1. In Table 1, the length of the video is shown in column "# Frames", the execution time is column "Time (s)", and the number of image points seen in the video excluding occluded points is column "# Points seen in *M* frames." From Table 1, the reprojection result from the synthetic face is very good and the results from real faces are also good, since RMS errors are all low.

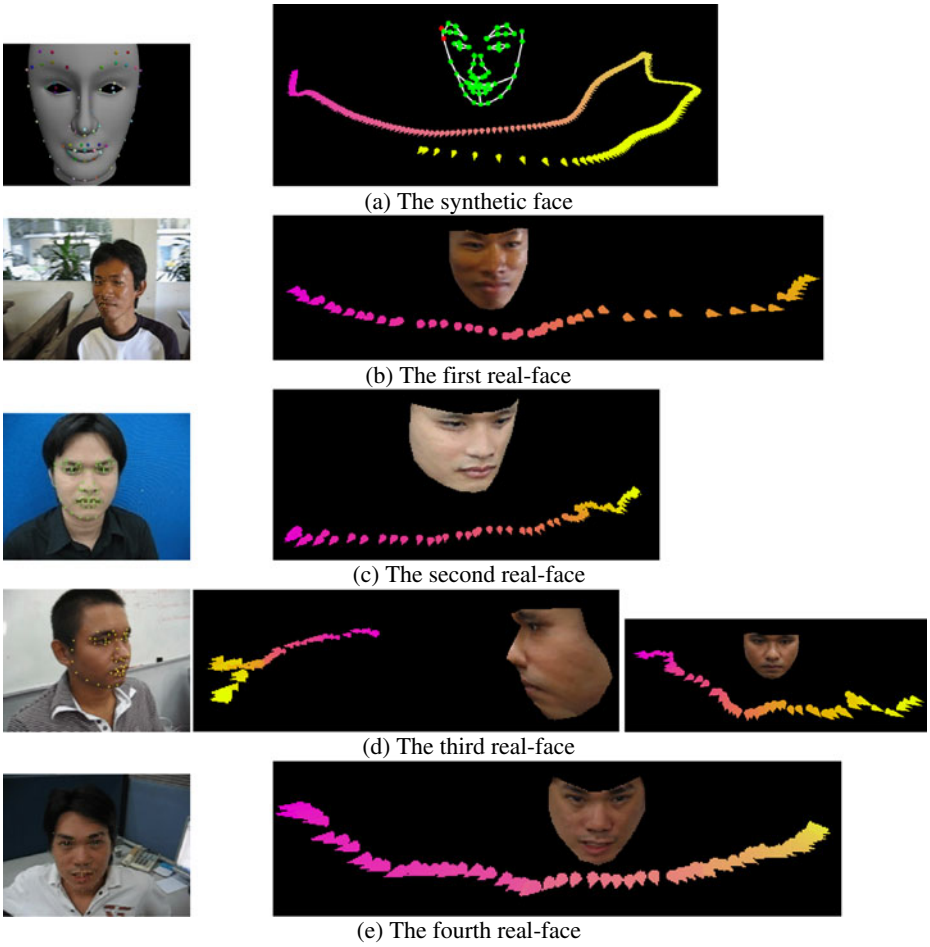


Fig. 2 Human faces after tracking and the 3D-reconstruction results along with camera path. **a** The synthetic face. **b** The first real-face. **c** The second real-face. **d** The third real-face. **e** The fourth real-face

Different views of the 3D-reconstructed results for all five experiments after texture mapping are shown in Fig. 2. From Fig. 2, all face-model and camera-motion results are very good. The camera for the synthetic face moves from the right of the face to its left and

Table 1 Reprojection errors of human faces

Face No.	Image		# Frames	# Points seen in M frames	Time (s)	RMS
	Width	Height				
Synthetic	640	480	250	17,000	390	5.13×10^{-2}
1st Real	640	480	47	2,956	152	1.97
2nd Real	320	240	72	4,143	158	1.00
3rd Real	640	480	66	3,936	303	1.90
4th Real	640	480	78	4,490	126	2.19

then back to the front of the face. The cameras in all remaining experiments move from the right to the left of the faces. The camera from Fig. 2(b) moves more quickly than in other faces so the cone representing the camera position of each frame is far from the contiguous cone. From the front view of Fig. 2(d), the camera motion has higher variation than other faces. Since the 3D-geometry ground truth is known for the synthetic face, the 3D-errors can be calculated as shown in Table 4, showing very low 3D-errors.

3.2 The presence of noise

The efficiency and robustness of the proposed method are demonstrated by adding noise to the image-coordinate of feature points. Since the graphical errors can be easily observed for a restricted camera motion, a synthetic video from a camera panning around a house and a hexagonal prism in a perfectly circular motion as used in [6] is created. The video has 101 frames each with 640×480 pixels. The house is represented by 10 points and the prism by 24 points, as illustrated in Fig. 3(a).

To add noise, image points are shifted from their true image-coordinates in alternate frames. The number of 3D-points, whose image coordinate is shifted, and the noise amounts used for shifting are shown in Table 2. Five cases are tested; i.e. no noise for the 1st case, varying amounts of noise added to all points in alternate frames for the 2nd–5th cases. From Table 2, columns "Δu" and "Δv" represent the noise amounts in the x and y directions of image coordinate, respectively. The "RMS" column represents reprojection errors after 3D-reconstruction. The results from 3D-reconstructions are also shown in Fig. 3. From Table 2 and Fig. 3, the 2nd-5th test cases show very good graphical and numerical results, similar to the 1st case.

From the experiments, after applying the two perturbations of the variable's brackets at the same time in (11) together with the motion interpolation in section 2.3 for a coarse reconstruction to the main process proposed in this paper, the system can tolerate the noise

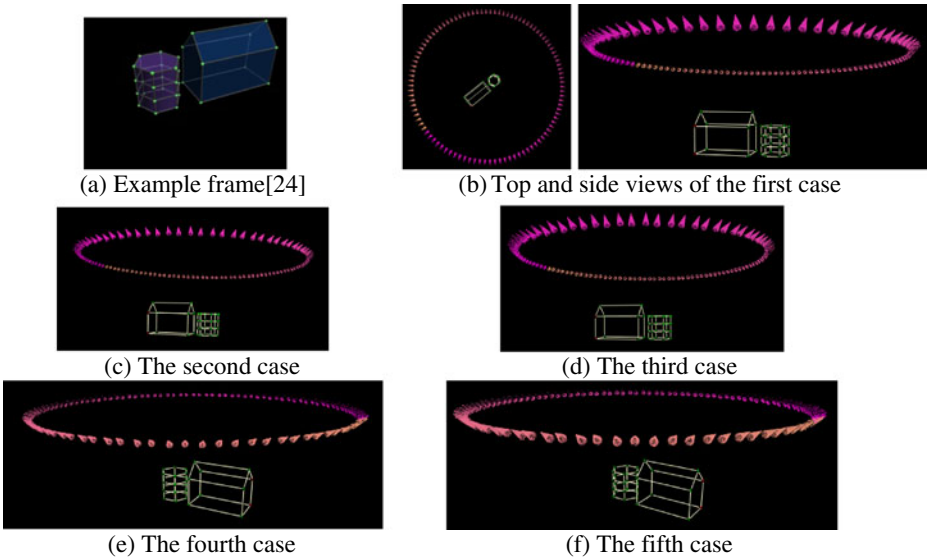


Fig. 3 3D-reconstructed results from the proposed method with noise added. **a** Example frame [6]. **b** Top and side views of the first case. **c** The second case. **d** The third case. **e** The fourth case. **f** The fifth case

Table 2 Reprojection errors of noise effect

Case	# Points with Noise	# Frames with Noise	# Points seen in M frames	Δ_u	Δ_v	Time (s)	RMS
1	0	0	3,434	0.0	0.0	58	5.76×10^{-3}
2	34	51	3,434	25.0	25.0	63	2.38×10^{-1}
3	34	51	3,434	30.0	30.0	65	2.84×10^{-1}
4	34	51	3,434	35.0	35.0	81	6.57×10^{-1}
5	34	51	3,434	40.0	40.0	81	7.47×10^{-1}

amounts up to $(\Delta u, \Delta v) = (44, 44)$. Without the motion interpolation, the system can tolerate to less noise amounts, i.e. $(\Delta u, \Delta v) = (34, 34)$. If the perturbation is applied to the variable updating for only one perturbation at each iteration and the motion interpolation is not used for estimating a coarse reconstruction as used in [6], the system can tolerate to the noise for only $(\Delta u, \Delta v) = (27, 27)$. In [5, 6], the positive perturbation $p_{t+1}[1]$ from (11) is always chosen first; if $p_{t+1}[1]$ cannot decrease the energy, $p_{t+1}[2]$ will then be tried.

3.3 Analysis of 3D accuracy

To analyze 3D accuracy of the proposed method, different paths of camera motion are tested using image size of 640×480 pixels. Experiments on a box with a camera moving in six different controlled paths shown in Fig. 4 are created in a synthetic environment. 3D errors for camera motion and the box's dimensions, whose ground truth is 60 cm×30 cm×30 cm, can be measured after rescaling the 3D-reconstruction results.

The 1st path, Fig. 4(a), is a full-circular path where the box is in the center. The 2nd path, Fig. 4(b), is a linear path where a camera does not pan towards the box, but instead points straight, while moving linearly. The 3rd path, Fig. 4(c), is a linear path where the camera pans towards the box's center while moving. The 4th path, Fig. 4(d), is a linear path where the camera moves towards to the box. The 5th path, Fig. 4(e), is a semicircular path where noise is added to the camera orientation. The noise added to the 5th path is generated by allowing 3D position of the camera's target to randomly vary from the box's center by ±10 cm in x, y, and z. Finally, the 6th path, Fig. 4(f), is a curved path where the box is not in the center of the arc.

From Table 3, reprojection errors of all paths are very low. From Table 4, 3D-errors from the circular, semicircular, and arc paths are very low even when noise is added to the semicircular path. Errors of the box's depth (Z_w) from the linear paths are still high because the linear paths cannot provide enough parallax to recover the ratio between the focal length and the distance in the z direction. So fixing the focal length to a known value (assuming camera calibration) can help solve this problem as seen in Table 4 under caption "Fix f." The 3D-reconstruction results of all controlled paths with fixed f for the linear paths are also shown in Fig. 4 where all paths are perfectly computed.

Based on the synthetic experiments, the proposed method can provide very good results even when the focal length is unknown if the camera moves in a curved path. Further real-world experiments on a camera moving in an approximately circular and arc paths are also tested, demonstrating the practicality and robustness of this method. It is difficult to precisely measure the 3D-errors of a real object. In the real-world experiments, the box with a label "TOP" shown in Fig. 5(a) is used with ground-truth

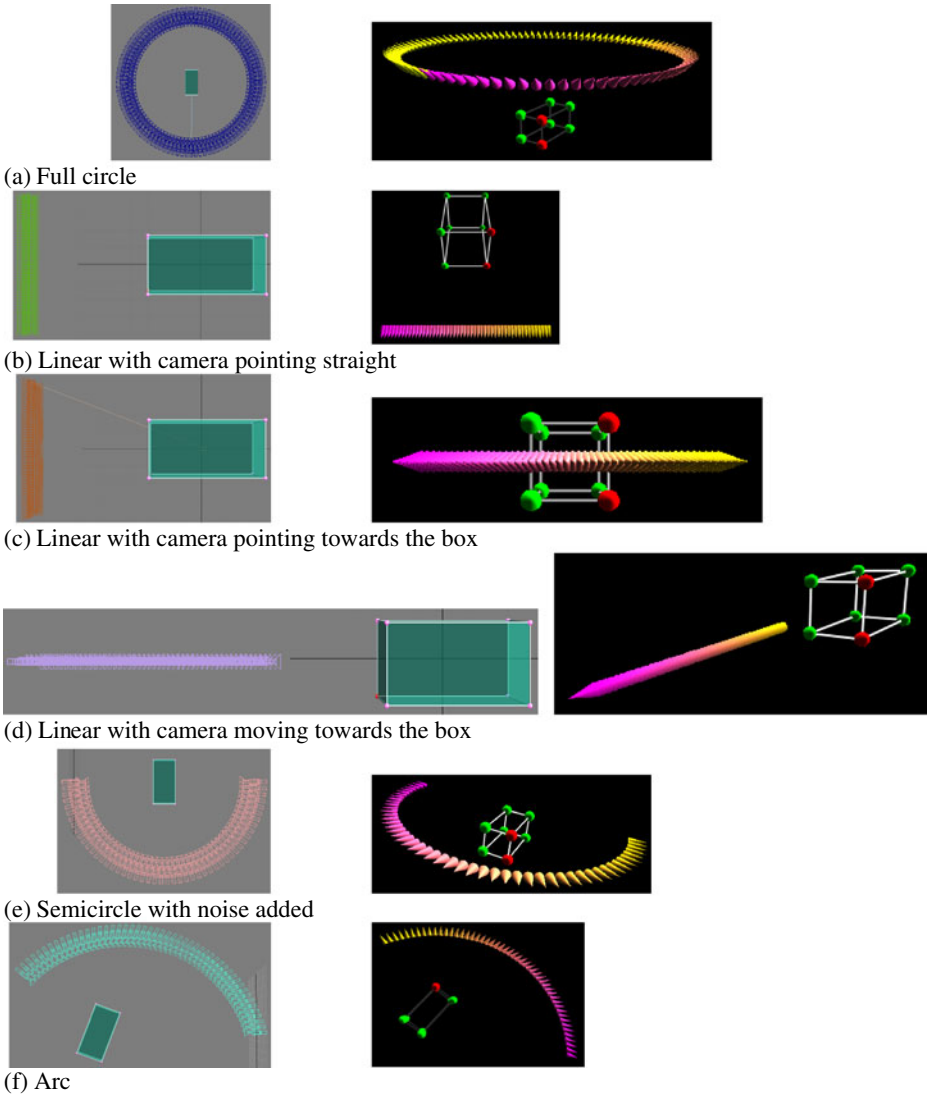


Fig. 4 Different controlled paths and their 3D-reconstructed results. **a** Full circle. **b** Linear with camera pointing straight. **c** Linear with camera pointing towards the box. **d** Linear with camera moving towards the box. **e** Semicircle with noise added. **f** Arc

dimensions shown in Table 5. Example frames together with their corresponding reprojection-results after 3D-reconstruction from the real experiments on a circular path and an arc path are shown in Fig. 5(b) and (d), respectively, where each plus mark (+) represents the points tracked by AAM and each crossed mark (×) represents the reconstructed points.

Figure 5(c) shows that the full circular path and the box are reconstructed very well. The ring of camera motion seems to have a little noise because the box is not in the exact center of the table while recording the video. Although the arc path is not a purely curved path, the

Table 3 Reprojection errors of controlled paths

Path	# Frames	# Points seen in M frames	Time (s)	RMS
Full Circle	101	686	16	9.85×10^{-3}
Point Straight	50	400	90	0.00
Point Straight (Fix f)	50	400	22	2.40×10^{-4}
Point Towards	50	260	20	3.58×10^{-1}
Point Towards (Fix f)	50	260	31	3.56×10^{-4}
Move Closer	50	400	201	0.00
Move Closer (Fix f)	50	400	5	0.00
Semicircle with Noise	50	338	9	3.44×10^{-4}
Arc	50	345	14	2.15×10^{-4}
Full Circle (Real Box)	130	871	13	1.33
Arc (Real Box)	119	811	21	1.57

reconstruction result is still very good as shown in Fig. 5(e) where the camera moves closer to the box at first and then it moves in a curved path. The 3D-errors of the box's dimensions can still be measured directly after rescaling with the known ground-truth as shown in Table 5. The 3D-errors of the camera motion are measured from the reprojection function as shown in Table 3 under column "Real Box." Tables 3 and 5 show that errors of the two experiments are very low. From the calculated errors and 3D-graphical results, it can be concluded that this proposed method is practical and also robust.

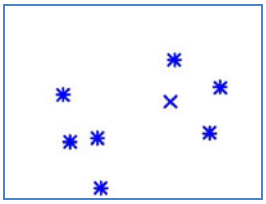
4 Comparison with other methods

In this section, an example auto-calibration (trifocal tensor [11]) and two commonly used methods for system optimization (conjugate gradient descent and Powell's line minimization [18]) are compared with the proposed method. To compare the proposed method with

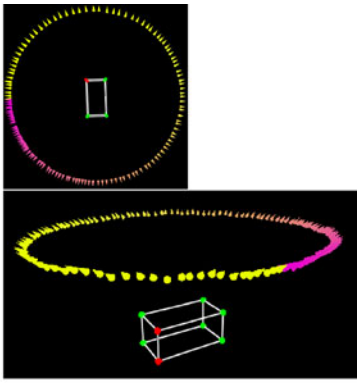
Table 4 Average 3D errors

Case	Synthetic Face	Full Circle Path	Linear Path						Semi-circle Path	Arc Path
			Point Straight		Point Towards		Move Closer			
				(Fix f)		(Fix f)		(Fix f)		
θ_x	$4.12e^{-1}$	$1.01e^{-2}$	$4.00e^{-6}$	0.00	$4.94e^{-4}$	$5.00e^{-6}$	0.00	0.00	$8.70e^{-2}$	$5.73e^{-3}$
θ_y	$9.77e^{-2}$	$5.91e^{-3}$	$2.40e^{-5}$	$3.39e^{-4}$	2.15	$1.12e^{-1}$	0.00	0.00	$7.69e^{-4}$	$1.35e^{-3}$
θ_z	$4.19e^{-1}$	$4.88e^{-3}$	$2.00e^{-6}$	0.00	$1.12e^{-1}$	$1.12e^{-1}$	0.00	0.00	$4.87e^{-2}$	$8.52e^{-4}$
t_x	$3.74e^{-2}$	$1.46e^{-2}$	$7.10e^{-5}$	$1.37e^{-3}$	1.08	$2.25e^{-3}$	0.00	0.00	$1.51e^{-2}$	$9.89e^{-3}$
t_y	$1.31e^{-2}$	$3.36e^{-2}$	0.00	0.00	$5.34e^{-4}$	0.00	0.00	0.00	$4.45e^{-3}$	$1.64e^{-3}$
t_z	$2.51e^{-2}$	$1.99e^{-2}$	$2.10e^{-5}$	$3.86e^{-4}$	$9.55e^{-1}$	$3.65e^{-4}$	$1.06e^{-3}$	$5.67e^{-4}$	$7.76e^{-3}$	$2.50e^{-3}$
X_w	$5.69e^{-3}$	$3.29e^{-3}$	$2.70e^{-5}$	$1.76e^{-4}$	2.36	$6.55e^{-4}$	0.00	$3.70e^{-5}$	$9.56e^{-4}$	$8.08e^{-4}$
Y_w	$2.41e^{-3}$	$3.18e^{-3}$	$1.20e^{-5}$	$1.76e^{-4}$	1.11	$2.33e^{-4}$	0.00	$3.70e^{-5}$	$3.26e^{-4}$	$3.24e^{-4}$
Z_w	$7.67e^{-3}$	$6.79e^{-3}$	8.90	$1.48e^{-3}$	3.63	$2.36e^{-3}$	9.64	$2.09e^{-4}$	$4.53e^{-4}$	$2.86e^{-4}$

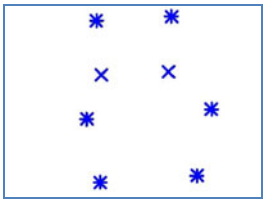
(a) Defined corners of the real box



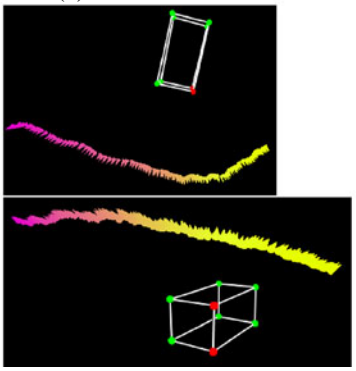
(b) Example frame and its result from a circular path



(c) 3DResults



(d) Example frame and its result from an arc path



(e) 3D Results

Fig. 5 Experiments on a real box. **a** Defined corners of the real box. **b** Example frame and its result from a circular path. **c** 3D Results. **d** Example frame and its result from an arc path. **e** 3D Results

trifocal tensor, one synthetic face from Fig. 2(a) and two example real-faces from [5] and Fig. 2(c) are used. The trifocal tensor with camera-resectioning approach, detailed in [11], is used. Note that, the 3D-model can be directly obtained from the trifocal tensor but not the six parameters defining camera pose in each frame. Thus, only the 3D-model will be compared here. The feature points on a human face are often occluded by the face itself. However, the trifocal tensor cannot work if any point is occluded. We must, hence, provide an approximate value for the missing points. Reconstruction results from the trifocal tensor can be called *valid* only if elements of the diagonal matrix calculated from the Absolute Dual Quadric (ADQ) decomposition are *all* positive or *all* negative.

Table 5 Ground truth and 3D-errors of a real box

Dimensions (Point to Point)	Height ($\times 10^{-1}$)				Width ($\times 10^{-1}$)				Depth ($\times 10^{-1}$)				RMS ($\times 10^{-1}$)
	0→1	2→3	4→5	6→7	0→4	1→5	2→6	3→7	0→3	1→2	4→7	5→6	
Ground Truth	119	119	120	118	269	266	266	267	139	141	137	140	–
Full Circle	0.39	−0.95	0.30	0.44	−1.07	−2.49	0.00	2.34	−0.81	−0.20	0.67	0.55	1.14
Arc	4.48	2.06	0.44	2.27	−8.07	−3.12	0.55	0.36	0.00	4.40	4.47	0.08	3.47

The test case using trifocal tensor provides as good a 3D-model (Fig. 6(a)) as the one obtained from the method proposed here (Fig. 2(a)). From Fig. 6(b), 3D-model of the first real-face obtained from the trifocal tensor is not as good as the model obtained from our method (which gives as good results as shown in [5]). For trifocal tensor, reordering the image sequence is also tested here with results of the first real-face shown in Fig. 6(c) where the trifocal tensor failed to reconstruct a 3D-model. To be fair, we reconstructed half of the face at a time to prevent the problem of point occlusion inherent in the trifocal tensor with 3D-results shown in Fig. 6(d) which is still rather distorted. Combining the half face and reordering provides results shown in Fig. 6(e) where the frontal view of the 3D result looks very good but the side view shows that the face is still rather distorted. The 3D results of the 2nd real-face from the similar experiments considering about a full and a half face are shown in Fig. 6(f)-(g). The 3D-model in Fig 6(f) looks good for the front view but is too flat for the side view. In Fig. 6(g), the 3D-model is almost good but it is completely flat looking from side view; meaning, the trifocal tensor failed to estimate the depth of the model. From the numerical results, it is clear that only the synthetic face yields a valid ADQ-decomposition.

In conclusion, the trifocal tensor cannot provide good 3D-model for real-world experiments if the problem of missing points is not solved, but our proposed method can. Furthermore, reordering image sequence in the trifocal tensor is important whereas our proposed method does not require it.

To compare the proposed method with conjugate gradient descent and Powell's line minimization with paraboloid bracketing approaches (details can be found in [18]),

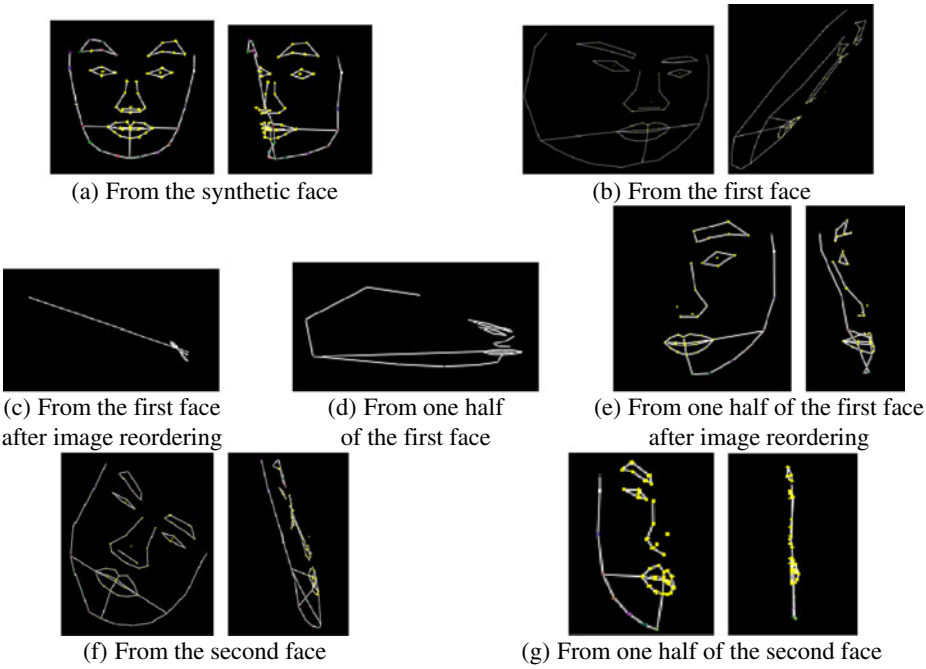


Fig. 6 3D-reconstructed results of human faces obtained from trifocal tensor. **a** From the synthetic face. **b** From the first face. **c** From the first face after image reordering. **d** From one half of the first face. **e** From one half of the first face after image reordering. **f** From the second face. **g** From one half of the second face

experiments on the synthetic face of Fig. 2(a) are used so that the effect of noise and lens distortion generally present in real-world experiments can be avoided. We compare our proposed method with the other two approaches in terms of execution time and RMS reprojection-errors. Graphs comparing the average reprojection-errors (pixels per point per frame) after system convergence amongst the three approaches are shown in Fig. 7. In each graph, the x-axis denotes index of the video frame and the y-axis denotes the average error. For conjugate gradient descent, the original method is applied to reduce the computational time for system convergence. That is, the system's derivative is forced to be recalculated each time after a maximum number of iterations in the main loop.

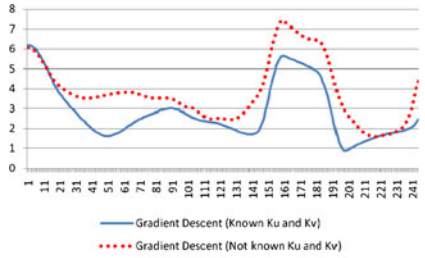
Let (6) be rewritten as

$$\begin{bmatrix} u(pixel) \\ v(pixel) \\ 1 \end{bmatrix} = \begin{bmatrix} q \times k_u \times (X_c/Z_c) \\ q \times k_v \times (Y_c/Z_c) \\ 1 \end{bmatrix} \tag{36}$$

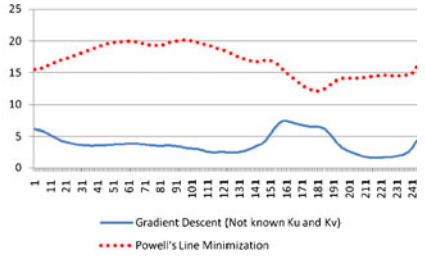
where $k_u = \mathrm{Img}_w/CCD_x$ and $k_v = \mathrm{Img}_h/CCD_y$.

The following settings must be done using the conjugate gradient descent to achieve the system convergence:

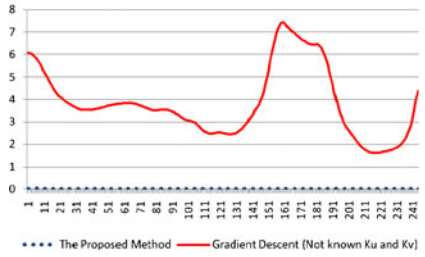
1) k_u and k_v must be known. For our proposed method, known values of the two parameters are not required.



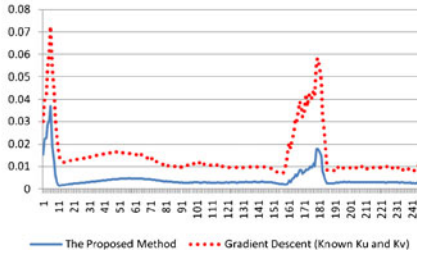
(a) Gradient descent with and without known intrinsic parameters



(b) Gradient descent without known intrinsic parameters and Powell's minimization



(c) The proposed method and gradient descent without known intrinsic parameters



(d) The proposed method and gradient descent with known intrinsic parameters

Fig. 7 Average reprojection-errors in the unit of pixels (y-axis) per frame (x-axis) for comparison. **a** Gradient descent with and without known intrinsic parameters. **b** Gradient descent without known intrinsic parameters and Powell's minimization. **c** The proposed method and gradient descent without known intrinsic parameters. **d** The proposed method and gradient descent with known intrinsic parameters

2) Z_w of all points must be initialized with a good guess, e.g. −25.0 cm (ground truth). The proposed method can just use −1.0 cm, resulting in quicker convergence without concern to the initialization value.
3) q must be initialized to a value close to the ground truth. Due to the small value of q and since the derivative used for parameter updating may be calculated from the trivial data, parameter q may easily escape far from the ground truth. Here, q is initialized to 10.0 cm (the ground truth is 4.3456 cm). Our proposed method can find parameter f with an even larger range of variation.

Setting the three parameter sets to the same values as used in our proposed method failed to find an optimal solution. This implies that the conjugate gradient descent is sensitive to the initial guess whereas the proposed method is not.

The comparison between the gradient descent with and without known camera's intrinsic parameters (CCD_x and CCD_y) is shown in Fig. 7(a). Though when k_u and k_v are not known requires longer convergence-time than the other case, its errors are still higher. Execution time for the case with and without k_u and k_v parameters is 1,137 s and 2,527 s, respectively.

Next, Powell's line minimization with paraboloid bracketing is compared to the gradient descent without known k_u and k_v. Even if a good initial guess for the three parameter sets is used in Powell's approach, its average errors are still very high compared with the gradient descent as seen in Fig. 7(b). The time consumed by Powell's method is 626 s. From our experiments, Powell's method will work well only if all parameters (not only the three sets) are initialized to a very good guess, and it takes only 65 s. Since a very good guess for parameter initialization in the real-world environment is not possible, Powell's approach will not be compared further in this paper.

Comparisons between our proposed method and the two cases of gradient descent are shown in Fig. 7(c)–(d). The average reprojection-errors of our proposed method are lower than both cases of the gradient descent in all frames. For the case of gradient descent with known k_u and k_v shown in Fig. 7(d), the method takes about 13 hours and 47 minutes to achieve almost as same results as of the proposed method whereas the proposed method takes only 390 s.

In conclusion, our proposed method consumes less time than all other compared methods while a very good result is still obtained since our method uses the bracketing techniques and other constraints explained in section 2 instead of calculating the derivative as done in the other two methods.

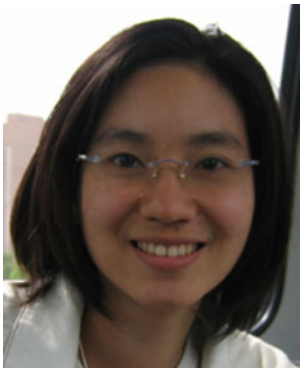
5 Conclusion

All real-world experiments in this paper are recorded by an off-the-shelf digital camera carried by a human walking around the object without using any dolly. Using feature points seen in images provides a big advantage when compared to using markers with known world-coordinates. Missing points including the case of self-occlusion which is generally found in experiments on real-faces does not pose a problem using our method. Without derivative calculation as required in the conjugate gradient descent approach, the proposed method consumes less time to achieve the optimal solution while still robust to noise. Using the limited bracket of step-sizes for reducing energy instead of using the values calculated from Powell's line minimization (which requires higher time complexity) prevents the proposed method from walking deep down the wrong direction and getting stuck at a local minimum. All parameters of

camera motion can be obtained directly and simultaneously with the face-model so one need not retrieve these parameters from the transformation matrices as needed using auto-calibration approach. The experiments and discussions in this paper demonstrate that the proposed approach is simple, practical, and robust.

References

1. Avidan S, Shashua A (1998) Novel view synthesis by cascading trilinear tensors. IEEE Trans Vis Comput Graph 4(4):293–306, Oct-Dec 1998
2. Bozek J, Grgic M, Delac K (2010) Comparative analysis of interpolation methods for bilateral asymmetry. IEEE Int Symp ELMAR, pp. 1–7, 15–17 Sep 2010.
3. Brent RP (2002) Algorithms for minimization without derivatives, 1st ed. Dover, 2002.
4. Chen OT-C (2000) Motion estimation using a one-dimensional gradient descent search. IEEE Trans Circuits Syst Video Technol 10(4):608–616, Jun 2000
5. Chouvatut V, Madarasmi S, Tuceryan M (2009) Face reconstruction and camera pose using multi-dimensional descent. Proc Int Conf Comput Electr Syst Sci Eng (CESSE), pp. 730–735, 25–27 Dec 2009.
6. Chouvatut V, Madarasmi S, Tuceryan M (2010) 3D reconstruction and camera pose from video sequence using multi-dimensional descent. 4th Int Conf Inf Syst Tech Manag (ICISTM), pp. 282–292, 10–12 Mar 2010.
7. Edwards GJ, Taylor CJ, Cootes TF (1998) Interpreting face images using active appearance models. 3rd IEEE Int Conf Autom Face Gesture Recognit, pp. 300–305, 14–16 Apr 1998.
8. Fischler MA, Bolles RC (1981) Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun Assoc Comput Mach (ACM) 24(6):381–395, Jun 1981
9. Galpin F, Morin L (2002) Sliding adjustment for 3D video representation. EURASIP J Appl Signal Process 2002(10):1088–1101, 2002
10. Hartley RI (1994) Projective reconstruction and invariants from multiple images. IEEE Trans Pattern Anal Mach Intell 16(10):1036–1041, Oct 1994
11. Hartley R, Zisserman A (2006) Multiple view geometry in computer vision, 2nd ed. Cambridge, 2006.
12. Karayiannis NB (2000) Reformulated radial basis neural networks trained by gradient descent. IEEE Trans Neural Netw 10(3):657–671, May 2000
13. Kato H, Billinghurst M (1999) Marker tracking and HMD calibration for a video-based augmented reality conferencing system. Proc 2nd IEEE and ACM Int Workshop Augment Real, pp. 85–94, Oct 1999.
14. Li J, Chellappa R (2005) A factorization method for structure from planar motion. IEEE Workshop Motion Video Comput (WACV/MOTIONS) 2:154–159, Jan 2005
15. Okuma T, Sakaue K, Takemura H, Yokoya N (2000) Real-time camera parameter estimation from images for a mixed reality system. IEEE Proc 15th Int Conf Pattern Recognit 4:482–486, 3–7 Sep 2000
16. Park SW, Heo J, Savvides M (2008) 3D face reconstruction from a single 2D face image. IEEE Comput Soc Conf Comput Vis Pattern Recognit Workshops (CVPR), pp. 1–8, 23–28 Jun 2008.
17. Po LM, Ng KH, Cheung KW, Wong KM, Uddin Y, Ting CW (2009) Novel directional gradient descent searches for fast block motion estimation. IEEE Trans Circuits Syst Video Technol 19(8):1189–1195, Aug 2009
18. Press WH, Teukolsky SA, Vetterling WT, Flannery BP (2007) Numerical recipes – the art of scientific computing, 3rd ed. Cambridge, 2007.
19. Repko J, Pollefeys M (2005) 3D models from extended uncalibrated video sequences: addressing key-frame selection and projective drift. Int Conf 3-D Digit Imaging Model, 2005.
20. Smolic A (2002) Robust generation of 360-degree panoramic views from consumer video sequences. 4th EURASIP-IEEE Reg 8 Int Symp Video/Image Process Multimed Commun (VIPromCom), pp. 431–435, 16–19 Jun 2002.
21. Tsai RY (1987) A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV camera and lenses. IEEE J Robot Autom RA-3(4):323–344, Aug 1987
22. Xu X, Dony RD (2004) Differential evolution with powell's direction set method in medical image registration. IEEE Int Symp Biomed Imaging: Nano to Micro 1:732–735, 15–18 Apr 2004
23. Zheng Y, Wang Z (2008) Robust depth estimation for efficient 3D face reconstruction. 15th IEEE Int Conf Image Process, pp. 1516–1519, 12–15 Oct 2008.
24. Zheng Y, Chang J, Zheng Z, Wang Z (2007) 3D face reconstruction from stereo: a model based approach. IEEE Int Conf Image Process (ICIP) 3:III-65–III-68, 16 Sep 2007 – 19 Oct 2007



Varin Chouvatut received the B.Eng. (Honor) and M.Eng. degrees in Computer Engineering from King Mongkut's University of Technology Thonburi (KMUTT) in 2002 and 2005, respectively. During 2008–2009, she stayed in Indiana University – Purdue University Indianapolis (IUPUI), IN, USA, to research estimation of camera pose, augmented reality, and 3D reconstruction. She is now a Ph.D. student at computer engineering department of KMUTT.

Suthep Madarasmi received B.S. and M.S. degrees in Computer Science from Michigan State University in 1984 and 1986, respectively. He later received his Ph.D. from the University of Minnesota in 1993. He has been a lecturer at King Mongkut's University of Technology Thonburi, Thailand since 1995 where he is now an associate professor. He was also head of Computer Engineering Department from 2004–2008.

Mihran Tuceryan received the S.B. degree in Computer Science and Engineering from the Massachusetts Institute of Technology in 1978 and his Ph.D. degree from the University of Illinois Urbana-Champaign in 1986. He has been with the Michigan State University, European Computer-Industry Research Centre (ECRC), and Texas Instruments, Inc. prior to joining his current position as Associate Professor at the IUPUI in 1997.