3D Face and Motion from Feature Points Using Adaptive Constrained Minima

Varin CHOUVATUT\textsuperscript{(a)}, \textit{Student Member}, Suthep MADARASMI\textsuperscript{(b)}, and Mihran TUCERIAN\textsuperscript{(c), Nonmembers}

SUMMARY This paper presents a novel method for reconstructing 3D geometry of camera motion and human-face model from a video sequence. The approach combines the concepts of Powell’s line minimization with gradient descent. We adapted the line minimization with bracketing used in Powell’s minimization to our method. However, instead of bracketing and searching deep down a direction for the minimum point along that direction as done in their line minimization, we achieve minimization by bracketing and searching for the direction in the bracket which provides a lower energy than the previous iteration. Thus, we do not need a large memory as required by Powell’s algorithm. The approach to moving in a better direction is similar to classical gradient descent except that the derivative calculation and a good starting point are not needed. The system’s constraints are also used to control the bracketing direction. The reconstructed solution is further improved using the Levenberg Marquardt algorithm. No average face model or known-coordinate markers are needed. Feature points defining the human face are tracked using the active appearance model. Occluded points, even in the case of self occlusion, do not pose a problem. The reconstructed space is normalized where the origin can be arbitrarily placed. To use the obtained reconstruction, one can rescale the computed volume to a known scale and transform the coordinate system to any other desired coordinates. This is relatively easy since the 3D geometry of the facial points and the camera parameters of all frames are explicitly computed. Robustness to noise and lens distortion, and 3D accuracy are also demonstrated. All experiments were conducted with an off-the-shelf digital camera carried by a person walking without using any dolly to demonstrate the robustness and practicality of the method. Our method does not require a large memory or the use of any particular expensive equipment.

\textit{key words:} camera pose, gradient descent, model reconstruction, Powell's multidimensional minimization

1. Introduction

A three dimensional (3D) face model provides an obvious advantage to face recognition since unseen viewpoints of the face can be rendered using the 3D model. Model reconstruction and camera pose (position and orientation) estimation can be viewed as two problems. In estimating camera pose, an obviously efficient method can be done by placing artificial markers with known 3D-positions in the scene \cite{1}--\cite{4}. Unfortunately, for 3D face reconstruction, placing markers on a human face is impractical. Also, measuring the relative 3D-position of markers cannot be done precisely. Thus, the proposed method finds a better strategy that uses only natural feature points appearing on the face without requiring prior knowledge of 3D positions. The active-appearance model (AAM) \cite{5}, \cite{6} is employed for natural feature tracking. However, the true image coordinate of a target object is difficult to observe accurately under usual circumstances and this is even more difficult when occlusion affects the appearance of the target object \cite{7}. The target object such as human face is not typically able to avoid this occlusion problem, especially the self-occlusion explained later, thus recovering its 3D model is difficult in usual circumstances while the occlusion problem is still presented. Even AAM cannot solve this self-occlusion problem.

Some researchers tried to avoid the self-occlusion problem using various approaches for the 3D face model reconstruction. Zheng et al. \cite{8} reconstructed 3D-face from stereo images and a reference face model. Since the traditional stereo approach based on intensity images cannot provide a good result, they included a reference model which must first be prepared for initialization. Furthermore, non-linear deformations and camera registration need to be solved to find correspondences between the stereo images. Later, Park et al. \cite{9} used a single image with an average face model generated from 3D training images. In their proposed method, a triangular mesh of the face model was created via Delaunay triangulation, an approach also used in this paper. Similarly, Zheng and Wang \cite{10} used only one frontal image to generate a face model though their approach requires a database of the detected image point depths. Reconstructing the face model from a single image would appear to provide a big advantage. However, a basis 3D-model of the target object and the 3D training images must first be available which is rather impractical.

An alternative approach commonly used to compute structure from motion using points without known 3D-positions or the basis 3D-model of the target object is auto-calibration \cite{11}. In auto-calibration, the fundamental matrix is often used when given only two frames \cite{11}; the trifocal tensor given three frames \cite{12}; the quadrifocal tensor given four frames \cite{13}; and the factorization approach given multiple frames \cite{14}. The main drawback of these approaches is in the three reconstruction levels required: namely, the projective, affine, and metric reconstructions. The absolute dual quadric (ADQ) that must first be computed in order to obtain the metric reconstruction is often invalid if the prior reconstruction steps are erroneous due to noisy data from the

Manuscript received February 18, 2011. Manuscript revised June 6, 2011.

\textsuperscript{1}The authors are with the Computer Engineering Department, King Mongkut’s University of Technology Thonburi, Bangkok 10140 Thailand.
\textsuperscript{2}The author is with the Computer and Information Science Department, Indiana University-Purdue University Indianapolis, Indianapolis, IN 46202-5132 USA.

a) E-mail: varin@cpe.kmutt.ac.th
b) E-mail: suthep@kmutt.ac.th
c) E-mail: tuceryan@cs.iupui.edu

DOI: 10.1587/transfun.E94.A.2207

Copyright © 2011 The Institute of Electronics, Information and Communication Engineers
error propagation. In addition, defining the plane at infinity requires the presence of three sets of parallel lines in three perpendicular planes. This is impractical as most scenes do not contain such information. The problems of occlusion, noise, and radial distortions in the image points led to noisy data input to the reconstruction process must first be solved. Furthermore, the Levenberg-Marquardt optimization process [15] is required after each reconstruction level to prevent noisy reconstructed results from propagating to the next, higher level. Without this optimization, the system will fail to find a valid reconstructed solution. Due to the complexity from the multiple steps and strategies needed for such a correspondence search, auto-calibration is considered an impractical approach. Nevertheless, auto-calibration’s advanced concept of normalizing the 3D-geometry volume of the reconstructed results which can later be rescaled to the desired coordinates is still applied to our proposed method whereas the noisy data and missing points including self-occlusion do not pose a problem in our method as presented below.

The 3D-reconstruction problem usually includes a large number of parameters and can be viewed as a multidimensional problem. In optimization approaches to a multidimensional problem, gradient descent approaches use derivative calculations known as conjugate directions [16] to search for the appropriate direction towards a solution. Although gradient descent can be applied to many applications such as estimating camera motion [17], training neural networks [18], and searching for motion of an image point [16],[19], the derivative may suggest a step size and direction that cause the overall system to jump in the wrong direction with poor data propagated in the iterations that follow. Consequently, the system may invariably take a much longer time to converge or even get stuck at a local minimum, especially when a good starting point is not provided. In addition, the 3D-reconstruction problem involves a highly complex system of equations; thus, errors in the derivative computation can easily happen. Furthermore, a good choice of the factor to be multiplied by the conjugate direction, often called the step size, is required to expedite convergence. In our proposed approach, the strong point of classical gradient descent in moving in better direction towards a valid and optimal solution is still maintained, while the weakness in requiring the use of derivatives is abandoned.

Some researchers view the multidimensional problem as a search for the minimum point in a valley of a 3D paraboloid. Powell’s minimization with paraboloid bracketing [20],[21] uses the direction sets initialized by unit vectors to move from a starting position in the solution space to a new position with lower energy. For each direction set, a line minimization with bracketing is performed to move the current position in the energy terrain to a minimal new position along the given direction set. Unfortunately, the previous direction sets (with the set size equal to the number of adjustable parameters) will also be kept and reused in the next search iterations. These previous direction sets are stored in a two dimensional array, requiring a large memory, especially when the number of adjustable parameters is large. Reusing all previous direction sets and keeping track of them may help in rapid descent (i.e., descent with the largest possible perturbation) towards the bottom of the valley. However, the risk of getting stuck at a local minimum is higher. To jump out of a local minima solution, back tracking along some sets of previously maintained directions is needed. From our experiments, Powell’s method can converge very quickly, but only if a good start position is provided. Thus, our proposed method uses line minimization with bracketing initialized by a unit value (or even a double value for speed), but we do not minimize deep down the valley along the current direction with the largest possible perturbation to decrease the greedy nature of the algorithm. In order to find each bracket, the system’s equality and inequality constraints are employed to control the system model not to get lost in a path towards an incorrect reconstruction. Consequently, the proposed method requires less time and space complexity due to the multidimensional array than the Powell’s method. To prevent getting stuck at a local minimum, we provide the system a chance of resetting the bracket to its starting position of larger perturbation when the bracket becomes too small so that the system can jump out of the current position.

The advantages in our proposed method may be useful for various application domains. Missing point correction is needed in applications using AAM or other correspondence search programs. In an augmented reality (AR) system, the AR’s 3D system can be directly generated from the computed 3D model along with camera’s parameters in all the frames [4]. For video surveillance applications, a 3D model of a face can help to convert a non-frontal face view into a frontal one, aiding the recognition process. In addition, if the police or security guards need a 3D facial model, our approach can easily find one by simply panning an off-the-shelf video camera to view a subject’s face. The 3D face model will be obtained as output. Our robust proposed method may also be used for other application domains such as in the medical field, virtual reality (VR), architecture, mechanics, device assembly, and etc.

Our approach does not require a good starting position or any previously known coordinate markers (or models) to the system. Yet, it can rapidly achieve convergence to a good solution. Allowing the origin of the reconstructed space to be anywhere and keeping the space volume normalized but not limited to unity helps achieve fast convergence.

2. Methodology

The only input required in this calculation is a video sequence of the target object(s) to be reconstructed. The reconstruction results are a model of the object in 3D space, the camera motion in 3D space, and the camera’s focal length. In this paper, the 3D space of the reconstruction will be displayed in OpenGL’s 3D environment where each frame’s camera pose is represented by a cone. All videos are recorded with the frame rate of 15 frames/s. Videos of syn-
user-defined unit such as centimeters in 3D geometry as

$$
S = \text{diag}(f, f, 1) \begin{bmatrix}
\frac{C_d (cm)}{C_d (cm)} & \frac{C_d (cm)}{C_d (cm)} & 1
\end{bmatrix}^T.
$$

(3)

From Eq. (3), \( f \) is implied from

$$
\begin{bmatrix}
S_u \\
S_v
\end{bmatrix} = \begin{bmatrix}
q (cm) \times \frac{\text{img}_w (pixels)}{\text{CCD}_w (cm)} \\
q (cm) \times \frac{\text{img}_h (pixels)}{\text{CCD}_h (cm)}
\end{bmatrix}
$$

(4)

where \( \text{img}_w \) and \( \text{img}_h \) are the image’s width and height, \( \text{CCD}_w \) and \( \text{CCD}_h \) are the camera’s aperture size in cm. From Eqs. (3) and (4), one may see that \( q \) is very small in the real-world but \( f \) is large.

Given \( N \) points seen in \( M \) frames, \( 2MN \) equations are provided from Eqs. (1)–(3). There are \( 6M + 3N + 1 \) unknowns in total, including six per frame for camera pose \((\theta_1, \theta_2, \theta_3, t_1, t_2, \text{and} t_3)\), three per point for world coordinates \((W_1, W_2, \text{and} W_3)\), and one for focal length \((f)\). In short, the core problem is to fit \( MN \) data points \( S = (u, v) \) to a model that has \( 6M + 3N + 1 \) adjustable parameters \( p \), providing the relationship [20]:

$$
\mathcal{F}(u, v) = \mathcal{F}(u, v | p_0 \ldots p_{6M + 3N}).
$$

(5)

If each data point has its own standard deviation \( \sigma \), we want to get fitted values for the \( p \)’s by minimizing the constraint from image formation given by:

$$
\mathcal{G}^2 = \sum_{i=0}^{MN-1} \left( \mathcal{F}(\tilde{u}_i, \tilde{v}_i) - \mathcal{F}(u_i, v_i | p_0 \ldots p_{6M + 3N}) \right)^2
$$

(6)

where \( \tilde{S} \) is the observed data.

From Eqs. (3) and (6), the proposed method aims to find the extremum of the energy function \( \zeta_w \), subject to the system’s constraints explained in Sect. 2.3, from a paraboloid problem as shown in Eq. (7). The minimum of this paraboloid is at its origin \((0, 0, 0)\). Our objective is to find this minimum.

$$
\zeta_w = \sum_{i,j} \left( \zeta_u^2 + \zeta_v^2 \right),
$$

$$
\zeta_u = \ell (u_i - \bar{u}_i),
$$

$$
\zeta_v = \ell (v_i - \bar{v}_i),
$$

(7)

where \( \ell \) is defined in Sect. 2.2.

The local and global energies \( \mathcal{R} \) to be used are calculated from

$$
\mathcal{R} = \sqrt{\frac{\zeta_w}{MN}}.
$$

(8)

The system’s convergence can be assumed when the following condition is met for a few iterations:

$$
\left[ \frac{\sum_{i,j} \left( \sqrt{\zeta_u^2 + \zeta_v^2} - \mathcal{R} \right)^2}{MN} \right]^{\frac{1}{2}} \leq \eta
$$

(9)
where $\eta \in \mathbb{R}^+$ is a small threshold.

The main steps for reconstruction are:

- Initialize all parameters and their perturbations using Eqs. (10) and (11). The value initialized to all perturbations is the bracket of variation for all parameters. The methodology for applying the bracketing method to the one-dimensional search problems can be found in [22].

$$
\begin{align*}
p_t - \alpha_t &< p_{t+1} < p_t + \alpha_t, \\
\alpha_t &= \gamma, \\
p_t &= 0.0, \\
\end{align*}
$$

where $\gamma \in \{1.0, 2.0\}$; $p_t$ and $\alpha_t$ are the parameter and its perturbation, respectively, at time $t$. The $\gamma$ should not be too large a number so as to control the volume of 3D space as normalized but significant, in order to achieve quick convergence. However, this does not mean that the reconstructed volume will be limited to $\gamma$. All $p_t's$ are initialized as in Eq. (10) with

$$
W_3 = -1.0,
$$

$$
f = \beta.
$$

where $\beta \in \{0.0, \ldots, 0.0\}$. Note that $\beta$ should not be a negligible number as implied from Eqs. (3) and (4). Assigning a constant value to $W_3$, as shown in Eq. (11), means all points in 3D space are put on a plane in front of a camera, parallel to the camera’s image plane.

- Repeat Steps 3-5 for each parameter until global convergence or a maximum number of iterations is attained.

- Apply two perturbations to a parameter by

$$
\begin{align*}
p_{t+1}[1] &= p_t + \alpha_t, \\
p_{t+1}[2] &= p_t - \alpha_t, \\
\end{align*}
$$

Equations (12) and (15) 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 [20], [23].

- Check the local energy from:

  - If the energy decreases,

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

  In the case that both $p_{t+1}[1]$ and $p_{t+1}[2]$ from Eq. (12) can decrease the energy, the parameter $p_{t+1}$ perturbation which results in a lower energy will be chosen.

  - Otherwise,

  $$
  \begin{align*}
p_{t+1} &= p_t, \\
\alpha_{t+1} &= b\alpha_t, \\
\end{align*}
$$

where $b \in (0.0, 1.0)$. Go back to Step 3 if $\alpha_{t+1} > \epsilon$ where $\epsilon \in \{0.0, \ldots, 0.0\}$. A good setting of $b$ is half the value from the previous $t$ because the bias against the current direction towards a solution can be increased by a fair value. In other words, the probabilities of the brackets (shown in Eq. (10)) between the current and the new set of values which may provide a correct direction to the solution are equal. This setting of $b$ plays an important role in discarding the direction of largest decrease utilized in Powell’s method.

- If $\alpha_{t+1} \leq \epsilon$, the system may get stuck and Eq. (15) should be used to jump out of the local minima:

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

Once a solution to the reconstruction problem is obtained, the Levenberg Marquardt (LM) [15], [20] algorithm explained in Sect. 2.4 can be applied to the reconstruction. Note that coordinate transformation explained in Sect. 2.5 must be done to convert the volume of 3D space to its correct, real-world dimension.

For clarification, some important aspects of the algorithm are further explained here. The perturbation represents a range of small directions for walking down the parabolic valley towards the bottom where system convergence is achieved in accordance with Powell’s line minimization with paraboloid bracketing [20], [22], [23]. In Step 2, the order of the parameters to be considered for line minimization is also considered to speed up the system’s convergence to a globally optimal solution; i.e., variables that tend to have higher variation are considered sooner. Thus, the variables with smaller variation will maintain their low values (close to zero) while the other variables with higher variation get updated. Here, the world coordinates are first considered; followed by the focal length; and, finally, the camera pose in each frame.

### 2.2 Active Appearance Model (AAM) with Outliers

AAM [5], [6] is an optimization process for interpreting images by adjusting parameters efficiently so that a synthetic example that matches the new image most closely can be generated. A vector of appearance parameters used for controlling its shape and gray levels is called the statistical model. The statistical model for human face can be generated from a training set of annotated-face images available in [24], [25].

First of all, a video sequence should be separated into two or more sequences so that each point is visible in all frames of each sequence. For example, the first sequence of the video may show only the frontal face, the second may show the left pose of face, and the third may show only the right pose. After applying AAM to each sequence, they can be recombined into one video. Then, the random sample consensus (RANSAC) [13], [26] based on the relationship between the fundamental matrix $F_{3x3}$ and point correspondences $x_i, x_i$ of point $i$ in 3D space is further used to eliminate outliers via:

$$
x_i^T F x_i = 0.
$$
The RANSAC algorithm can be outlined as:

- Repeat for $K$ samples.
  - Select a sample of seven correspondences and compute $F$ from Eq. (16). There will be one or three real solutions.
  - Calculate the displacement $d$ of each correspondence.
  - Count the number of inliers where $d < \varepsilon$, for a given threshold $\varepsilon$.
  - If there are three solutions, choose the one with the most number of inliers.
- Choose $F$ with the most number of inliers. In the case of a tie, choose the one with the lowest standard deviation $\sigma$ 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 $\ell$ 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. For example, 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.

2.3 Equality and Inequality Constraints

After applying two perturbations to a parameter in Eq. (12), certain criteria to double check constraints on the parameter are required to prevent unacceptable reconstruction.

*The camera pose* $[R \mid T]$ of any two contiguous frames should have a small difference, i.e.

$$\left| p_i^t - p_i^{t-1} \right| < \delta p$$  \hspace{1cm} (17)

where $p_i^t$ is the camera’s parameter of frame $j$ at time $t$ and $\delta p$ is the threshold. If Eq. (17) does not hold, then set:

$$p_i^t = p_i^{t-1},$$

$$\alpha_{t+1} = \gamma,$$  \hspace{1cm} (18)

where $\gamma$ is defined in Eq. (10).

*The point* $W$ must be in front of the camera in a Cartesian coordinate system. This can be done by setting:

$$W_3 = -|W_3|.$$  \hspace{1cm} (19)

*The focal length* $f$ must be positive to conform with Eq. (19). If not, it will be reset by:

$$f_i = \begin{cases} \lfloor f_i \rfloor, & \text{if } f_i < 0.0 \\ \nu, & \text{if } f_i = 0.0 \end{cases}$$  \hspace{1cm} (20)

where $\nu \in [\mathbb{R}^+ \mid \nu \neq 0.0]$.

2.4 Levenberg Marquardt (LM) Algorithm

The LM algorithm [15], [20] outlined below varies between the inverse-Hessian method and the steepest descent method. When the solution is still far from the fitting function, the steepest descent method will be used. The inverse-Hessian method will later be used as the fitting function is approached. Unfortunately, this LM method can work well only if a good starting point to the solution is given. Thus, the LM algorithm can be used to improve the solution obtained in Sect. 2.1, whose global energy is now near to the bottom of its paraboloid, as required.

- Given the set of fitted parameter set $p$, compute $\chi^2(p)$ of the model as shown in Eqs. (5) and (6).
- Set a discrete factor $\lambda$.
- Solve equations for the increment set $\delta p$ to give it to the next approximation $\chi^2(p + \delta p)$.
- Compare the current approximation $\chi^2(p)$ to the next approximation $\chi^2(p + \delta p)$:
  - If the next approximation decreases, set
    $$\lambda_{t+1} = \lambda_t - \delta \lambda_t,$$
    $$p_{t+1} = p_t + \delta p.$$  \hspace{1cm} (21)
  - Otherwise, set
    $$\lambda_{t+1} = \lambda_t + \delta \lambda_t.$$  \hspace{1cm} (22)
- Repeat Step 3 until $\chi^2(p)$ decreases by a negligible amount $\varepsilon$, for a few iterations in a row.

2.5 Coordinate Transformation

As explained earlier in Sect. 2.1, the normalized geometry of the reconstruction $p$ can be transformed to the real-world geometry $\tilde{p}$ by

$$\tilde{p}_{i,j} = p_{i,j} \times \frac{\|b_i - b_j\|}{\|a_i - a_j\|}.$$  \hspace{1cm} (23)

where $a_i$ is 3D coordinate of the $i$th reconstructed point, $b_i$ is 3D coordinate of the $i$th real-world point, and $p_{i,j} = a_i - a_j$.

The final geometry of the reconstruction proposed in this paper is in a 3D space whose origin can be anywhere, not fixed to any point in the image. Converting the obtained coordinate to an arbitrary coordinate can be done by

$$\tilde{t}_i = [I \mid -t_0] \, t_i,$$  \hspace{1cm} (24)

where $t_i$ is the $i$th homogeneous coordinate with respect to the reconstructed origin. The coordinate computed from Eq. (24) will then be oriented according to the coordinate of the first camera $C_0$ in the order of roll, yaw, and pitch for $\alpha$, $\varphi$, and $\theta$ degrees, respectively, to the coordinate $t_i$ by:

$$t_1 = t_1\cos\varphi\cos\alpha + t_2\cos\varphi\sin\alpha - t_3\sin\varphi,$$
$$\begin{align*}
t_2 &= t_1(\sin\theta \sin\varphi \cos\alpha - \cos\theta \sin\alpha) \\
    &+ t_2(\sin\theta \sin\varphi \sin\alpha + \cos\theta \cos\alpha) + t_3 \sin\theta \cos\varphi, \\
\hat{t}_3 &= t_1(\cos\theta \sin\varphi \cos\alpha + \sin\theta \sin\alpha) \\
    &+ t_2(\cos\theta \sin\varphi \sin\alpha - \sin\theta \cos\alpha) + t_3 \cos\theta \cos\varphi. \quad (25)
\end{align*}$$

Next, the coordinate system from Eq. (25) can now be transformed to a new coordinate system of the first camera \( \hat{C}_0 \) with pitch, yaw, and roll of \( \theta, \phi, \) and \( \psi \), respectively. So the new coordinate \( \hat{t}_i \) will be:

$$\begin{align*}
\hat{t}_1 &= t_1 \cos \psi \cos \phi - t_2 (\sin \psi \cos \theta - \cos \psi \sin \phi \sin \theta) \\
    &+ t_3 (\sin \psi \sin \theta + \cos \psi \sin \phi \cos \theta), \\
\hat{t}_2 &= t_1 \sin \psi \cos \phi + t_2 (\cos \psi \cos \theta + \sin \psi \sin \phi \sin \theta) \\
    &- t_3 (\cos \psi \sin \theta - \sin \psi \sin \phi \cos \theta), \\
\hat{t}_3 &= -t_1 \sin \phi + t_2 \cos \phi \sin \theta + t_3 \cos \phi \cos \theta. \quad (26)
\end{align*}$$

Finally, the new coordinate from Eq. (26) may also be applied to \( \hat{C}_0 \) whose \( \hat{T} = \hat{t}_i \), so the final coordinate \( \hat{t}_i \) will be:

$$\hat{t}_i = [I | \hat{t}_0] t_i. \quad (27)$$

3. Experimental Results

3.1 Human Faces

After the face model was successfully reconstructed from Sect. 2, a 3D triangular mesh is generated by Delaunay triangulation [27] followed by a texture mapping. The texture mapping can be done by using three images of the frontal, left, and right views of the face.

Figure 2 shows example frames and reconstruction results from videos of one synthetic face and four real faces. The results of all face models and camera motion are very good. The cameras in all experiments move from the right to the left side of the faces except Fig. 2(c) where the camera starts from the frontal face. The camera motion in Fig. 2(e) has higher variation than in the other faces.

Reprojection errors after reconstruction are shown in Table 1. The length of each video is shown in column “Frms”, the execution time in column “Time”, and the number of points seen in the video excluding the occluded points in column “Pnts.” The result from the synthetic face is excellent while the results from real faces are also good, since the RMS (root mean square) errors computed by reprojection in pixels per point per frame are all low. Since the ground truth of the 3D geometry for the synthetic face is known, the geometry errors are shown in Table 5 under column “Face”, showing very low errors in all the dimensions.

3.2 The Presence of Noise and Lens Distortion

The efficiency and robustness of the proposed method are demonstrated by adding noise and lens distortion to the feature points. Since the errors from reconstruction can be observed easily for a restricted camera motion, the experiments with a camera panning a house and a hexagonal prism in a perfectly circular motion are presented. The video obtained from the camera has 101 frames, each with 640×480 pixels. The house and the hexagonal prism are represented by 10 and 24 points, respectively, as illustrated in Fig. 3(a).

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Reprojection errors of human faces.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Face no.</td>
<td>Image</td>
</tr>
<tr>
<td>Synthetic</td>
<td>640×480</td>
</tr>
<tr>
<td>1st real</td>
<td>640×480</td>
</tr>
<tr>
<td>2nd real</td>
<td>320×240</td>
</tr>
<tr>
<td>3rd real</td>
<td>640×480</td>
</tr>
<tr>
<td>4th real</td>
<td>640×480</td>
</tr>
</tbody>
</table>

![Fig. 2](image)

(a) Five faces with annotation shown  
(b) Output for synthetic face  
(c) Output for the 1st real face  
(d) Output for the 2nd real face  
(e) Output for the 3rd real face

(f) Output for the 4th real face

Fig. 2 Human faces after tracking. The results of the 3D geometric reconstruction along with 3D camera motion paths are shown.
To add noise, image points are shifted from their true coordinates in alternate frames. The number of points, whose coordinate is shifted, and the noise amount used for shifting are shown in Table 2. The noise amounts are shown in column \((\Delta u, \Delta v)\). The noise added to this section is known as positional noise [7], [28] which may be independent from frame to frame due to jitter in the camera motion. The jitter may occur because the camera moves on a bumpy road, for example, or when the camera is carried by human walking around the target object without using any dolly, causing the center of viewpoint to randomly deviate from the object’s center. Positional noise may be present due to other reasons [7] such as rapid camera movement, low frame rate, incorrect feature tracking, and etc. Thus, the noise amount is the positional noise presented in each image coordinate of Eq. (2). In other words, the noise amount is the displacement between the true and the tracked image coordinates. These noisy data usually cause a serious problem in the auto-calibration approaches as shown in Sect. 4.1. However, since positional noise is common in tracking, it is essential that the algorithm to compute 3D reconstruction and camera motion be robust against such noise. Later we will demonstrate that our algorithm performs well in spite of such noise whereas other algorithms fail miserably.

The three cases in this section include no noise for the 1st case, and varying amounts of noise added to all points in alternate frames for the 2nd and the 3rd cases. Reconstruction results of the three cases are illustrated in Fig. 3. From Table 2 and Fig. 3, the 2nd and the 3rd cases show very good reconstruction results, similar to the 1st case. The maximum amount of noise tolerated by the proposed method is about 8% (per image coordinate) of the image’s diagonal. For a camera with resolution of about 178 pixels/cm, a \((\Delta u, \Delta v) = (44, 44)\) in radius around each image coordinate is the maximum amount tolerable. Nevertheless, if all image coordinates are not affected by a noise amount greater than the maximum amount, i.e. \((\Delta u_i, \Delta v_i) > (44, 44); i = 0, \ldots, N-1\), the method can still tolerate the noise.

To add lens distortion, the second order of the radial and tangential distortions is added to the image coordinates of all frames. The undistorted image coordinate \(\mathbf{S}\) of Eq. (2) may be rewritten as

\[
\mathbf{S} = \text{diag}(s_u, s_v) \begin{bmatrix} \hat{u} \\ \hat{v} \end{bmatrix} 1^T. \tag{28}
\]

Let \(k_1\) and \(k_2\) be the first and second radial distortion coefficients, \(p_1\) and \(p_2\) be the first and second tangential distortion coefficients. The four coefficients are applied to \(\mathbf{S}\) as

\[
\hat{u} = u \left(1 + k_1 r^2 + k_2 r^4 + 2p_1 u \hat{v} + p_2 (r^2 + 2u^2)\right), \\
\hat{v} = v \left(1 + k_1 r^2 + k_2 r^4 + 2p_2 u \hat{v} + p_1 (r^2 + 2v^2)\right), \\
r^2 = \hat{u}^2 + \hat{v}^2. \tag{29}
\]

Thus, the distorted image coordinate \(\hat{\mathbf{S}}\) to be input to our system will then be

\[
\hat{\mathbf{S}} = \text{diag}(s_u, s_v) \begin{bmatrix} \hat{u} \\ \hat{v} \end{bmatrix} 1^T. \tag{30}
\]

The different amounts of \((k_1, k_2)\) and \((p_1, p_2)\) used to distort the image coordinates are shown in Table 3. If the distortion coefficients are much larger than presented in Table 3 and a reconstruction solution cannot be found, the lens distortion correction approach proposed in [29] should first be used to recover the undistorted image coordinates. Table 3 shows that the RMS geometry errors from various distortions are very low in all dimensions. The reconstruction results from all distortions shown in Table 3 are compared in a graph format for each dimension in Fig. 4. From Fig. 4, the geometry errors are low in all dimensions even when large distortions are added. For each graph dimension, the RMS errors are similar even when different distortion types and amounts are applied.

From all of the explained experimental results, even when different amounts of noisy data and lens distortions are included, the system is still able to tolerate the noise and lens distortions, providing good 3D geometry and camera path reconstruction results.

### 3.3 Analysis of 3D Accuracy

The experiments on a synthetic box with the dimensions 60 cm × 30 cm × 30 cm and six different controlled camera motion paths are shown in Fig. 5 for an image size of 640×480 pixels. The 1st camera path, Fig. 5(a), is a full circular path where the box is in the center. The 2nd path,
Table 3  RMS camera path and object geometry errors (°, cm) for lens distortion (with \( k_1, k_2; p_1 = p_2 = 0.001 \)).

<table>
<thead>
<tr>
<th>((k_1, k_2))</th>
<th>((0.0, 0.0))</th>
<th>((0.1, -0.05))</th>
<th>((0.2, -0.1))</th>
<th>((0.3, -0.15))</th>
<th>((0.4, -0.2))</th>
<th>((0.1, 0.05))</th>
<th>((-0.2, 0.1))</th>
<th>((-0.3, 0.15))</th>
<th>((-0.4, 0.2))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\theta_1)</td>
<td>1.28e-2</td>
<td>1.14e-1</td>
<td>2.16e-1</td>
<td>3.17e-1</td>
<td>4.12e-1</td>
<td>9.85e-2</td>
<td>2.06e-1</td>
<td>3.15e-1</td>
<td>4.27e-1</td>
</tr>
<tr>
<td>(\theta_2)</td>
<td>5.99e-3</td>
<td>7.85e-2</td>
<td>1.49e-1</td>
<td>2.20e-1</td>
<td>2.90e-1</td>
<td>7.18e-2</td>
<td>1.45e-1</td>
<td>2.19e-1</td>
<td>2.95e-1</td>
</tr>
<tr>
<td>(\theta_3)</td>
<td>7.72e-3</td>
<td>9.26e-2</td>
<td>1.78e-1</td>
<td>2.63e-1</td>
<td>3.48e-1</td>
<td>9.17e-2</td>
<td>1.80e-1</td>
<td>2.72e-1</td>
<td>3.64e-1</td>
</tr>
<tr>
<td>(t_1)</td>
<td>2.44e-3</td>
<td>7.06e-1</td>
<td>1.17e-1</td>
<td>1.01e-1</td>
<td>1.12e-1</td>
<td>5.58e-1</td>
<td>9.07e-1</td>
<td>1.10</td>
<td>1.38</td>
</tr>
<tr>
<td>(t_2)</td>
<td>1.48e-3</td>
<td>8.88e-1</td>
<td>1.62e-1</td>
<td>1.98e-1</td>
<td>2.50e-1</td>
<td>7.96e-1</td>
<td>1.49</td>
<td>2.10</td>
<td>1.24</td>
</tr>
<tr>
<td>(t_3)</td>
<td>1.15e-3</td>
<td>3.95e-1</td>
<td>7.59e-1</td>
<td>1.12e-1</td>
<td>1.48e-1</td>
<td>3.66e-1</td>
<td>7.54e-1</td>
<td>1.15</td>
<td>1.53</td>
</tr>
<tr>
<td>(W_1)</td>
<td>6.48e-4</td>
<td>3.17e-1</td>
<td>6.65e-1</td>
<td>1.01e-1</td>
<td>1.34e-1</td>
<td>4.01e-1</td>
<td>7.71e-1</td>
<td>1.15</td>
<td>1.53</td>
</tr>
<tr>
<td>(W_2)</td>
<td>5.00e-5</td>
<td>1.40e-1</td>
<td>2.84e-1</td>
<td>4.25e-1</td>
<td>5.63e-1</td>
<td>1.61e-1</td>
<td>3.16e-1</td>
<td>4.76e-1</td>
<td>6.40e-1</td>
</tr>
<tr>
<td>(W_3)</td>
<td>4.85e-4</td>
<td>2.30e-1</td>
<td>4.56e-1</td>
<td>6.75e-1</td>
<td>8.88e-1</td>
<td>2.42e-1</td>
<td>4.89e-1</td>
<td>7.43e-1</td>
<td>1.00</td>
</tr>
</tbody>
</table>

![Comparison graphs for camera path and object geometry errors (°, cm) for various lens distortions.](image)

Fig. 4  Comparison graphs for camera path and object geometry errors (°, cm) for various lens distortions.

Fig. 5(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. 5(c), is similar to the 2nd path except that the camera pans towards the box as if to track the box while moving. The 4th path, Fig. 5(d), is a linear path where the camera heads towards the box. The 5th path, Fig. 5(e), is a semicircular path where noise is also added to the camera orientation by allowing the camera’s target (point being tracked) to randomly vary from the box’s center by ±10 cm in all dimensions. Finally, the 6th path, Fig. 5(f), is a curved path where the box is intentionally set off-center to the arc. Reprojection errors after reconstruction of all paths are shown in Table 4. All errors are very low. From Table 5, the 3D geometry errors of the circular, semicircular, and arc paths are very low even when noise is added to the semicircular path. The errors for all linear paths can be found in

Table 4  Reprojection errors for various controlled paths.

<table>
<thead>
<tr>
<th>Path</th>
<th>Frms</th>
<th>Pts</th>
<th>Time(s)</th>
<th>RMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full circle</td>
<td>101</td>
<td>680</td>
<td>21</td>
<td>9.85e-3</td>
</tr>
<tr>
<td>Point straight</td>
<td>50</td>
<td>400</td>
<td>76</td>
<td>2.40e-4</td>
</tr>
<tr>
<td>Gaze towards</td>
<td>50</td>
<td>260</td>
<td>45</td>
<td>3.56e-4</td>
</tr>
<tr>
<td>Move closer</td>
<td>50</td>
<td>400</td>
<td>5</td>
<td>0.00</td>
</tr>
<tr>
<td>Semicircle with noise</td>
<td>50</td>
<td>338</td>
<td>10</td>
<td>3.44e-4</td>
</tr>
<tr>
<td>Arc</td>
<td>50</td>
<td>345</td>
<td>16</td>
<td>2.15e-4</td>
</tr>
<tr>
<td>Full circle (real box)</td>
<td>130</td>
<td>871</td>
<td>19</td>
<td>1.33</td>
</tr>
<tr>
<td>Arc (real box)</td>
<td>119</td>
<td>811</td>
<td>29</td>
<td>1.57</td>
</tr>
</tbody>
</table>
Table 5  RMS camera path and object geometry errors (° cm).

<table>
<thead>
<tr>
<th>Case</th>
<th>Synthetic</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Face</td>
<td>Circle</td>
</tr>
<tr>
<td>θ1</td>
<td>1.42e-3</td>
<td>1.01e-2</td>
</tr>
<tr>
<td>θ2</td>
<td>4.55e-2</td>
<td>5.91e-3</td>
</tr>
<tr>
<td>θ3</td>
<td>9.12e-2</td>
<td>4.88e-3</td>
</tr>
<tr>
<td>t1</td>
<td>1.24e-3</td>
<td>1.46e-2</td>
</tr>
<tr>
<td>t2</td>
<td>1.16e-2</td>
<td>3.36e-2</td>
</tr>
<tr>
<td>t3</td>
<td>5.01e-2</td>
<td>1.99e-2</td>
</tr>
<tr>
<td>W1</td>
<td>2.06e-3</td>
<td>3.29e-3</td>
</tr>
<tr>
<td>W2</td>
<td>2.86e-3</td>
<td>3.18e-3</td>
</tr>
<tr>
<td>W3</td>
<td>3.21e-3</td>
<td>6.79e-3</td>
</tr>
</tbody>
</table>

Table 6  RMS camera path and object geometry errors (° cm) for linear paths.

<table>
<thead>
<tr>
<th>Case</th>
<th>Point straight</th>
<th>Gaze towards</th>
<th>Move closer</th>
</tr>
</thead>
<tbody>
<tr>
<td>θ1</td>
<td>0.00</td>
<td>5.00e-6</td>
<td>0.00</td>
</tr>
<tr>
<td>θ2</td>
<td>3.39e-4</td>
<td>1.12e-1</td>
<td>0.00</td>
</tr>
<tr>
<td>θ3</td>
<td>1.37e-3</td>
<td>2.25e-3</td>
<td>0.00</td>
</tr>
<tr>
<td>t1</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>t2</td>
<td>3.86e-4</td>
<td>3.65e-4</td>
<td>5.67e-4</td>
</tr>
<tr>
<td>t3</td>
<td>1.76e-4</td>
<td>6.55e-4</td>
<td>3.70e-5</td>
</tr>
<tr>
<td>W1</td>
<td>1.76e-4</td>
<td>2.33e-4</td>
<td>3.70e-5</td>
</tr>
<tr>
<td>W2</td>
<td>1.48e-3</td>
<td>2.36e-3</td>
<td>2.09e-4</td>
</tr>
</tbody>
</table>

Table 7  Ground truth and object 3D geometry errors (x10^-1 cm) for a real box in case of circular and arc camera paths.

<table>
<thead>
<tr>
<th>Dimensions</th>
<th>Box</th>
<th>Ground</th>
<th>Circle path</th>
<th>Arc path</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>side</td>
<td>truth</td>
<td>Err. % Err.</td>
<td>Err. % Err.</td>
</tr>
<tr>
<td>Height</td>
<td>0</td>
<td>119</td>
<td>0.39</td>
<td>3.28e-3</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>119</td>
<td>-0.95</td>
<td>7.98e-1</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>120</td>
<td>0.30</td>
<td>2.50e-1</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>118</td>
<td>0.44</td>
<td>3.73e-1</td>
</tr>
<tr>
<td>Width</td>
<td>0</td>
<td>269</td>
<td>-1.07</td>
<td>3.98e-1</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>266</td>
<td>-2.49</td>
<td>9.36e-1</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>266</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>267</td>
<td>2.34</td>
<td>8.76e-1</td>
</tr>
<tr>
<td>Depth</td>
<td>0</td>
<td>139</td>
<td>-0.81</td>
<td>5.83e-1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>141</td>
<td>-0.20</td>
<td>1.42e-1</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>137</td>
<td>0.67</td>
<td>4.89e-1</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>140</td>
<td>0.55</td>
<td>3.93e-1</td>
</tr>
<tr>
<td>RMS</td>
<td>1.14</td>
<td>5.40e-1</td>
<td>3.47</td>
<td>2.08</td>
</tr>
</tbody>
</table>

Table 6. The reconstruction results of all paths are perfectly computed.

Real-world experiments on a camera moving in approximately circular and arc paths are also tested, demonstrating the practicality and robustness of this method. Since a precise measurement cannot be done in the real world, a box with label “TOP” illustrated in Fig. 6(a) with dimensions specified in Table 7 is used in the two experiments. Example frames together with their corresponding reprojections after reconstruction of the circular and the arc paths are shown in Figs. 6(b) and 6(c), respectively. Each plus mark (+) represents the point obtained from Sect. 2.2 and each crossed mark (×) represents the point after reconstruction (the “×” is overlap of “+” and “×”). Note that the reconstruction is so accurate that the “+” and “×” overlap to appear as
‘*’ for all the visible points.

Figure 6(d) shows that the full circular path is very well reconstructed. 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 this video. Although the arc path is not an exact curve, the reconstruction is still very good as shown in Fig. 6(e) where the camera moves closer to the box at first and then it moves in a curved path. The reconstruction errors in the box’s dimensions are measured after the transformation explained in Sect. 2.5 with the errors shown in Table 7. The reprojection errors are shown in Table 4 under caption “Real Box.” Tables 4 and 7 show that errors computed from the two experiments are very low.

All the experimental results demonstrate that this proposed method is practical and robust.

4. Comparison with Other Methods

In this section, an example auto-calibration (trifocal tensor [13], [30]) and two commonly used methods for system optimization (conjugate gradient descent and Powell’s line minimization [20], [23]) are compared with the proposed method.

4.1 Trifocal Tensor with Camera Resectioning

One synthetic face and two real faces, including the 1st and 2nd real faces from Sect. 3.1 are used here for comparison. Details of the trifocal tensor and the camera resectioning approaches for auto-calibration can be found in [13]. The 3D face model can directly be obtained from the trifocal tensor but not the six parameters defining the camera pose. Thus, only the 3D model will be compared in this section. The feature points on a face are often occluded by the face itself but, unlike our approach, the trifocal tensor cannot work if even a single point is occluded. We must, hence, provide an approximate value for each missing point. This is a serious drawback of the auto-calibration algorithm. The obtained reconstruction can be called valid if and only if the absolute dual quadric (ADQ) is positive (or negative) semi-definite. This is another complexity since sometimes a solution may not be found.

For a better illustrated comparison, the face models reconstructed using our proposed method from Figs. 2(b)–(d) are shown again here in Figs. 7(a), (c), and (h), respectively. The reconstructed model of the synthetic face using the trifocal tensor shown in Fig. 7(b) is as good as the one in Fig. 7(a). Figure 7(d) shows that the model from the first real face is not as good as our model shown in Fig. 7(c). Thus, the approach to reordering image sequence for trifocal tensor is considered. The reconstruction is shown in Fig. 7(e), where the trifocal tensor failed to reconstruct a model. To be fair, we tried reconstructing half of the face at a time to prevent the occlusion problem inherent in the auto-calibration approach, but the reconstructed model is still rather distorted as shown in Fig. 7(f). Combining the half face and the reordering approach provides the reconstructed face whose frontal view looks very good but the side view shows heavy face distortion in Fig. 7(g). The reconstructed models of the second real face considering a full and a half face are shown in Figs. 7(i) and 7(j), respectively. The front view of Fig. 7(i) looks good whereas the side view lacks depth. Figure 7(j) shows an even worse result where the model is completely flat looking from the side; meaning, the trifocal tensor failed to estimate the relative depth of the model.

In conclusion, the auto-calibration approach cannot provide good reconstruction for real world experiments if the problem of missing points is not solved or if the data is noisy, but our proposed method can. Furthermore, reordering the image sequence in the trifocal tensor is necessary whereas our proposed method does not require it.

4.2 Optimization Approaches for Multidimensional Search

In this section, our proposed method is compared with the conjugate gradient descent and Powell’s line minimization with bracketing approaches (details can be found in [20], [22], [23]). The synthetic face from Sect. 3.1 is used so that the effects of noise and lens distortion generally present in real-world experiments can be avoided. Graphs comparing the RMS reprojection errors amongst the three approaches...
Fig. 7  Reconstruction results of human faces obtained from the trifocal tensor to compare with the results from our proposed method (a), (c), (b).

are shown in Fig. 8, where the x-axis of each graph denotes frame number and the y-axis denotes the RMS error.

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

- \( k_u \) and \( k_c \) must be known. For our proposed method, known values of the two parameters are not required.
- \( W_j \) must be initialized to a good guess. Our method can provide a solution with quicker convergence without requiring such initial value.
- \( q \) must be initialized to a value close to the ground truth. Due to the small value of \( q \) and since the derivative may be calculated from the trivial data set, parameter \( q \) may easily escape far from the ground truth. Our proposed method, instead, successfully finds parameter \( f \) that has 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.

First, the comparison between the gradient descent with and without known \( (k_u, k_c) \) is shown in Fig. 8(a). Though when \( k_u \) and \( k_c \) are not known requires longer con-
vergence time than when known, the errors are still higher. The execution times for the case with and without \((k_u, k_v)\) are 712 s and 1,408 s, respectively.

Second, Powell’s method is compared to the gradient descent without known \((k_u, k_v)\). Even if a good initial guess for the parameters is used in Powell’s approach, its errors are still higher than the gradient descent as illustrated in Fig. 8(b). The time consumed by Powell’s method is 633 s. From our experiments, Powell’s method will work well only if all parameters are initialized to a good guess. It then takes only 47 s. Since a very good guess for parameter initialization in the real-world environment is not possible, Powell’s method will not be compared further in this paper.

Third, the two cases of gradient descent are compared to the proposed method in Figs. 8(c) and 8(d). The errors of our method are lower than both gradient descent cases in all frames. The case of gradient descent with known \((k_u, k_v)\) takes about 11 hours and 15 minutes to achieve almost the same result as of the proposed method as seen in Fig. 8(d). However, our method takes only 407 s for the entire image sequence. Also, our method does not require \(k_u\) and \(k_v\) to be known.

In conclusion, our proposed method consumes less time than all other compared methods and requires no initial guess while providing a very good result.

5. Conclusion

From the experiments and discussions throughout this paper, we have shown that our proposed simple, practical, and robust approach works well without requiring conjugate directions, a large memory, many levels of reconstruction, or even a good starting point. Missing points, including those from self-occlusion common to real faces, do not pose a problem in our approach. The method consumes less time to achieve an optimal solution when the geometry volume is normalized and the bracketing approach is controlled via the system model constraints. Without using a conjugate direction, our method can still find the optimal solution while being robust to noise including lens distortion in the real-world environment as all the videos in this paper are recorded by an off-the-shelf digital camera carried by a human walking around the object without using a dolly or any other special equipment. Using only feature points seen in images provides a big advantage when compared to using markers or average face models with known world-coordinates. Most optimization approaches can work well only if a good starting point to the solution is provided; whereas, our method does not require that. Using the limited brackets instead of keeping all previous brackets as done in Powell’s method (which has higher time and space complexity) prevents the proposed method from walking deep down the valley in the wrong direction and getting stuck at a local minimum. However, discarding the direction with the largest decrease in energy as in Powell’s algorithm is still applied in order to reduce the greedy nature of the algorithm. All parameters of camera motion can be obtained directly and simultaneously along with the 3D face model, avoiding the need to retrieve these parameters from the metric camera matrices common in the auto-calibration approaches. Since the origin of the world coordinates can be placed arbitrarily and the geometric volume is normalized, fast convergence can be achieved even with a long video sequence. The geometric volume can later be rescaled using a known absolute distance as a unit. The origin of the world coordinates can also be changed by the coordinate transformation explained earlier.

References


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.

Varin Chouvut received B.Eng. (Honor) and M.Eng. degrees in Computer Engineering from King Mongkut’s University of Technology Thonburi (KMU TT), Thailand in 2002 and 2005, respectively. During 2007–2008, she spent a year at Indiana University - Purdue University Indianapolis (IUPUI), USA, as a research scholar to research estimation of camera pose, augmented reality, and 3D reconstruction. She is now a Ph.D. student at Computer Engineering Department of KMUTT.