New Calibration-free Approach for Augmented Reality Based on Parameterized Cuboid Structure

Chu-Song Chen, Chi-Kuo Yu, Yi-Ping Hung
Institute of Information Science, Academia Sinica, Nankang 115, Taipei, Taiwan
song@iis.sinica.edu.tw

Abstract

A new method called PCS (parameterized cuboid structure) is presented for augmented reality. In particular, our method can insert animated virtual objects into a static scene, with geometric consistency, and also allow the user to interactively position and rotate the virtual objects with respect to a world coordinate system in a physically (or intuitively) meaning way. Such capability can not be achieved by using the existing calibration-free methods. To achieve this goal, we develop a new method for estimating camera parameters, which uses a cuboid structure (or more generally, a parallelepiped structure) as the reference object. The reference cuboid structure can be either explicit or implicit — implicit in the sense that the cuboid structure can be inferred by human perception even though it does not appear explicitly in the image. This method can determine the sizes of the cuboid (or parallelepiped) as well as the intrinsic and extrinsic parameters of the camera. To insertt a virtual object into a single un-calibrated image, some human interaction is unavoidable. We have implemented an AR authoring system based on the proposed PCS method which provides an auxiliary line and a refinement criterion to assist human interaction. Experimental results have demonstrated that our method can successfully insert virtual objects into both static and dynamic scenes with highly convincing geometric consistency.

1. Introduction

A fundamental problem in augmented reality (AR) is to insert virtual objects into real images. One of the major issues in AR is the geometric consistency [1]. Existing vision-based approaches for solving the AR problems can be classified into two types. The first type assumes that some 3D calibration points can be observed in the image sequence [1][8] or that the intrinsic camera parameters are pre-calibrated and a known 3D model [2] of the scene is given in advance [14]. The major research issues remained in this type of approaches are tracking and re-computing the pose of camera. Another type of approaches assumes that no prior 3D information is available [4][9][10][12], and hence is more difficult. Since approaches of this type do not require the calibration of camera parameters (either intrinsic or extrinsic parameters) in advance using a known 3D calibration object, they are referred to as the *calibration-free approaches* in this paper. To solve this type of problems, one way is to compute the camera parameters and 3D structures in the Euclidean space directly from the information contained in the image sequences. In [10], Ong et al. developed an AR approach by first reconstructing some unorganized 3D points using the structure-from-motion method proposed by Tomasi and Kanade [16] via the assumption that the camera model is orthographic. In [4], Faugeras showed some AR examples realized by using the *self-calibration* technique for computing the camera parameters. Another way is to perform the AR task in the affine or projective space rather than in the Euclidean space, which is referred to as the *calibration-free augmentation in other spaces* (CFAOS) in this paper [9][12]. However, in the CFAOS approaches, the virtual objects to be inserted have to be manually drawn in the both of the first two frames of the image sequence.

In this paper, we focus on the AR problems without using prior 3D information such as calibration points or known 3D models. To solve this type of problems, almost all the existing approaches require to select some reference points on the real objects appeared in the image sequences for computing the relative poses that is required for rendering virtual objects. However, a common drawback of these approaches is that they can not be applied to inserting moving virtual objects relative to the *static* real

[1] Another issue is the *photometric consistency*, but which is not the main concern of this paper.

[2] In this paper, a known 3D model refers to a wire-frame or curved model whose shape information (including size) is known.

objects or static background ("static" relative to the camera). For example, consider an image sequence containing a moving truck shown in Fig. 7. This sequence was captured with a fixed camera and the truck is the only moving real object. We may want to insert a virtual sedan on top of the truck with consistent motion (referred to as problem A) and insert another virtual convertible which moves to the right relative to the background (referred to as the problem B), as shown in Fig. 7. To perform the above AR task, we can treat the truck as a moving reference object and the ground to be a static reference object. Existing calibration-free methods can solve problem A but can not solve problem B. It is because that, with those methods, either the camera parameters have to be estimated from the disparities between two or more images [4][10], or the animated virtual object has to be manually-drawn in a static scene frame by frame carefully [9][12]. However, in the above example, the ground is static in all images of the sequence, and hence no disparity between images is available for computing the pose of the ground plane with respect to the camera.

The above drawback of the calibration-free approaches constrains their application to the AR problem [3]. In fact, it is still hard and tedious to inserting animated or moving 3D virtual object into a single 2D image (or panorama) since a user may have to carefully manipulate the virtual object frame by frame in order to generate convincing results — convincing in the sense that the inserted object is visually consistent with the real scene in terms of geometric relation. To overcome this difficulty, we develop an appropriate way to insert an animated virtual object into a static scene. In particular, the essential issue is to develop an effective method for inserting virtual objects into one *single image*.

In this paper, we propose a new technique for augmented reality based on a *parameterized cuboid structure* (PCS). The proposed method can be used for inserting virtual objects not only in one single image but also in a sequence of images which contains static and/or dynamic scenes. Our main idea for solving this problem is that, if the user can identify a cuboid structure in the scene, then the computer can insert arbitrary virtual objects in the scene automatically and allows the user to interactively control the motion of the virtual objects relative to the cuboid coordinate system, in an intuitively simple and user-friendly way.

This paper is organized as follows: Section 2 introduces the PCS method. Section 3 describes how to perform AR tasks using the PCS method. Some experimental results are shown in Section 4. Finally, Section 5 gives some conclusions and discussion.

2. Camera Parameter Estimation Based on PCS

Inserting an animated objects in a single image (or panorama) requires the estimation of the camera parameters with respect to some reference objects in the scene. However, estimation of the camera parameters from a single image is an ill-posed problem if there is no additional constraints on the reference objects [4]. In this paper, we use "information a priori less constraining to obtain" [4] instead of using calibration points with known coordinates which information is unlikely to have for many real applications. The basic idea came from the Façade system [2] in which a parameterized CAD model was used. A parameterized CAD model can inherently provide information about the shape constraints of an object (such as parallelism, orthogonality of lines, equal length of line segments etc.). The simplest parameterized CAD models is the parameterized cuboid structure (PCS) adopted in this paper. We will show that the camera parameters can be estimated from 2D image coordinates of a specific set of six corner points of a 3D parallelepiped. The idea of using shape constraints for camera parameter estimation is not new. However, most existing approaches either assume that the intrinsic parameters are given in advance [2][5][13][18], or require more than one images [4], and hence are not suitable for the purpose of inserting virtual objects into a single un-calibrated image.

The camera equation used in this paper is

$$\lambda p = K[R_{3\times3} \mid t_{3\times1}]P \tag{1}$$

and

$$K = \begin{bmatrix} f_u & s & u \\ 0 & f_v & v \\ 0 & 0 & 1 \end{bmatrix}, \tag{2}$$

where P is the homogeneous coordinates of a 3D point, p is the homogeneous coordinates of its 2D image point, K is the matrix consists of the intrinsic parameters, R and t are rotation and translation with respect to the world coordinate system, and λ is the *relative depth*.

Lemma 1: Given a parallelogram in 3D space with known image coordinates of four corner points, we can determine the relative depths of the four corner points.

---

[3] For example, inserting animated virtual characters into a static scene is useful for developing more interesting video games.

[4] Although the self-calibration methods do not require reference objects, at least two images have to be used [11].

***pf*:** Consider the four points $\{\boldsymbol{P_0}, \boldsymbol{P_1}, \boldsymbol{P_4}, \boldsymbol{P_2}\}$ which forms a parallelogram as shown in Fig. 1. The camera equations of the four corner points are

$$\lambda_0 p_0 = K(RP_0 + t), \quad \lambda_1 p_1 = K(RP_1 + t),$$
$$\lambda_4 p_4 = K(RP_4 + t), \quad \lambda_2 p_2 = K(RP_2 + t).$$

Since $P_0 + P_4 = P_1 + P_2$ is the property of a parallelogram for any world coordinate systems, we can derive that

$$\lambda_0 p_0 + \lambda_4 p_4 = \lambda_1 p_1 + \lambda_2 p_2$$

and hence yield the following three homogeneous linear equations:

$$\begin{bmatrix} u_0 \\ v_0 \\ 1 \end{bmatrix} \lambda_0 = \begin{bmatrix} u_1 & -u_4 & u_2 \\ v_1 & -v_4 & v_2 \\ 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_4 \\ \lambda_2 \end{bmatrix} \equiv A \begin{bmatrix} \lambda_1 \\ \lambda_4 \\ \lambda_2 \end{bmatrix},$$

where the homogeneous coordinates of the image points are denoted by $p_i=[u_i\ v_i\ 1]^T$ for $i=0,1,2,4$. Since the matrix A is nonsingular iff the three points p_0, p_1, p_2 are not co-linear, we have

$$[\ \lambda_1\ \lambda_4\ \lambda_2]^T = A^{-1}[u_0\ \ v_0\ \ 1]^T \lambda_0\ . \qquad (3)$$

That is, the relative depths λ_1 / λ_0, λ_4 / λ_0, λ_2 / λ_0 can be computed from the image points using (3). ■

Notice that the property of lemma 1 holds without knowing the intrinsic parameters K. However, only knowing the relative depths is insufficient for many AR applications. Before further derivation, some assumptions should be made because the matrix defined in (2) is too general to be solved without ambiguity. Often it can be assumed that there is no skew and that the aspect ratio is tuned to one, i.e. $f_u=f_v$ and $s=0$. This assumption has also been adopted by many self-calibration approaches [6][11]. Hence, (2) reduces to (4).

$$K = \begin{bmatrix} f & 0 & u \\ 0 & f & v \\ 0 & 0 & 1 \end{bmatrix} \qquad (4)$$

According to our experience, camera parameter estimation with this assumption is accurate enough for many AR applications. In our approach, a parameterized parallelepiped defined by the lengths of its three edges, $\{a, b, c\}$, as shown in Fig. 1, is used for camera parameter estimation.

Property 1: Given a parameterized parallelepiped in 3D space, if we know the three angles between its adjacent edges, $\theta_{ab}, \theta_{bc}, \theta_{ac}$ and the image coordinates of the six corner points of its two adjacent faces, then we can determine the rigid motion $\{R, t\}$, the intrinsic parameters $\{f, u, v\}$, and the size parameters $\{a, b, c\}$, by solving a polynomial equation of at most fourth degree.

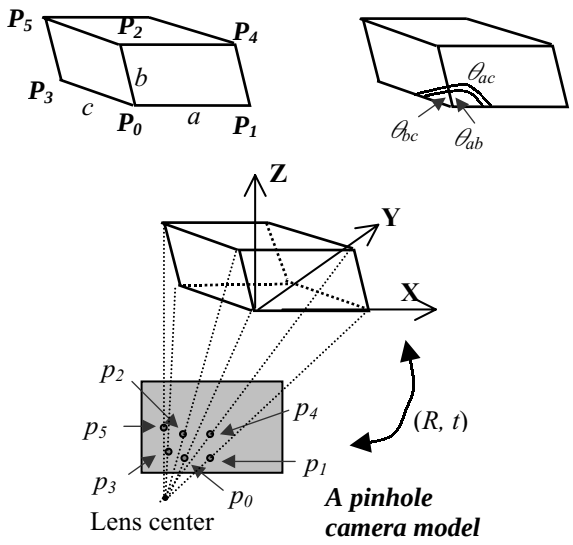

Figure 1. A parallelepiped with edge lengths a, b, c. Assume that the three angles between its adjacent edges are $\theta_{ab}, \theta_{bc}, \theta_{ac}$.

***pf*:** see Appendix 1.

It is worth comparing our fundamental method (property 1) to the method using perspective trihedral angle constraint [18] (referred to as the P3A method). The P3A method assumes that the angles between three lines passing through a common point are given and the intrinsic parameters are known in advance. Then, the camera pose (i.e., extrinsic parameters) can be determined. In fact, the P3A problem is equivalent to our formulation using parameterized parallelepiped if only four corner points $\{P_0, P_1, P_2, P_3\}$ are utilized (see Fig. 1). The closed-form solution of the P3A problem requires to solve a fifth-degree polynomial equation. Our method is more powerful because we can solve both the intrinsic and the extrinsic parameters, but two more corner points $\{P_4, P_5\}$ are required. In addition, our method only needs to solve a fourth-degree polynomial equation which has analytic solutions. In the past, Faugeras [3] has also shown that the knowledge of the angle between two lines in the scene puts a polynomial constraint of degree 2. However, that method requires at least two images.

Basically, Property 1 gives general solutions to a class of camera parameter estimation problems. However, when putting them into practical use of inserting a virtual object into a single un-calibrated image, a common problem is that some control points of the parallelepiped may not be seen in the images. That is, the parallelepiped structure may not be completely and explicitly observed in the image, even though the missing control points can be

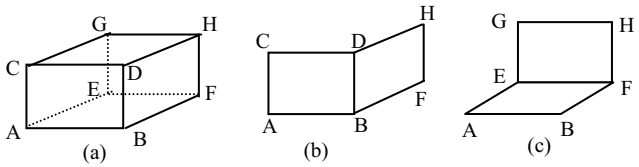

Figure 2. (a) A cuboid with ABFE as the ground plane. (b) Control points of convex type. (c) Control points of concave type.

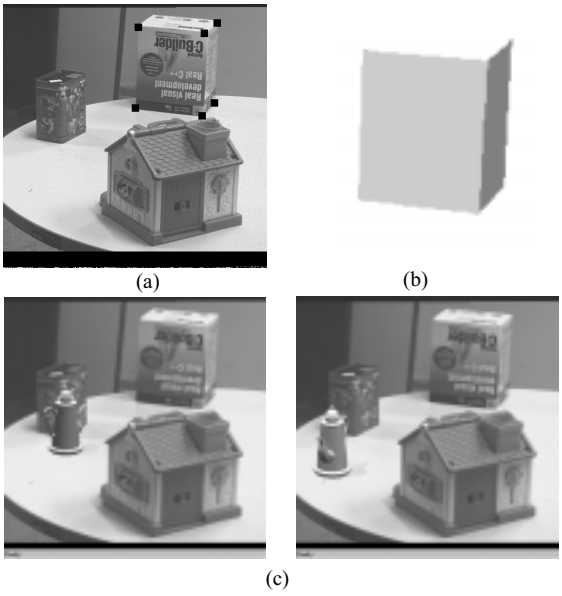

Figure 3. An example of augmenting a virtual object into a single un-calibrated image using the PCS method. (a) Select the control points (here the control points of convex type are used). (b) The computed 3D structure. (c) Two instances of augmenting a virtual teapot on the table surface.

inferred by human perception based on the image content. To overcome the above problem, human interaction is often used. [5] That is, if a user can manually specify the six control points of a parallelepiped in the scene, then the computer can integrate arbitrary animated virtual objects automatically and allow the user to interactively control its motion with respect to the world coordinate system defined by the parallelepiped. This operation is similar to that of the CFAOS approach [9][12] in which the virtual objects have to be put in the first two images of an sequence. However, our method can allow the user to insert virtual objects directly in the Euclidean space rather than in other less intuitive spaces. In particular, to have better human-machine interaction, we choose a special kind of the parallelepiped in which all the three angles, $\theta_{ab}, \theta_{bc}, \theta_{ac}$, are 90^o. This choice reduces the general parallelepiped to a *cuboid*, and the corresponding method for camera parameter estimation is referred to as the *PCS* method in this paper. The reason we choose the cuboid structure for human-machine interaction is that it is intuitive to human perception and can be found in many real scenes (e.g., see Fig. 5(b)). Also, the problem of solving a general fourth-degree polynomial equation can be simplified to a problem of solving a linear polynomial equation when a cuboid structure is used (as described below).

Property 2: Given a parameterized cuboid in 3D space, if we know the image coordinates of the six corner points of its two adjacent faces, then we can determine the rigid motion $\{R, t\}$, the intrinsic parameters $\{f, u, v\}$, and the size parameters $\{a, b, c\}$, by solving a linear system.

pf: see Appendix 2.

3. Augmented Reality Using PCS Method

We have built a human-machine interactive system which allows the user to insert virtual objects into a single un-calibrated image. If the user has specified the six control points of a cuboid in the image, our system can automatically compute the 3D cuboid structure (up to a scale) and the camera parameters, and then a virtual object can be inserted into the image automatically and a user can interactively control its motion with respect to the cuboid coordinate system. Because many applications require to insert virtual objects moving on a ground plane, we have adopted two types of control points selected from a cuboid for more flexible insertion. Fig. 2(b) shows the control points of convex type and Fig. 2(c) shows the control points of concave type. In our system, real cuboids contained in the scene can no doubt be served as useful reference objects, as shown in Fig. 3. However, our approach does not require real cuboid structures to appear in the scene completely and explicitly, as described in Section 2. An important reason is that we can give the user good hints to identify the human-drawn structure, as introduced below.

3.1 Auxiliary Line

In the CFAOS approach [9], epipolar line constraints are derived to help the user to put virtual objects into the first two images. In our approach, an auxiliary line can also be derived, which constrains the position of $\boldsymbol{P_5}$ if the positions of $\boldsymbol{P_0}$, $\boldsymbol{P_1}$, $\boldsymbol{P_2}$, $\boldsymbol{P_3}$, and $\boldsymbol{P_4}$ are given. Remember that the two faces $\{P_0, P_1, P_4, P_2\}$ and $\{P_0, P_2, P_5, P_3\}$ are both parallelograms and are used to solve the relative depths of $P_0, P_1, ..., P_5$ in the proof of property 1 (in Appendix 1). The fact that these two parallelograms have a common point, P_2 (in addition to P_0), generates an additional constraint that "the same λ_2 should be obtained by applying lemma 1 to these two parallelograms." Suppose that $P_5=[u_5\ v_5\ 1]$ has not been determined yet

[5] For example, an interaction tool was built in [7] for allowing users to tour in a single image.

while the other five control points have been specified by the user. Using the parallelogram $\{P_0, P_1, P_4, P_2\}$ we can first obtain the relative depths $\lambda_1, \lambda_4, \lambda_2$ via Lemma 1, and then write the equation related to $\{P_0, P_2, P_5, P_3\}$ as follows:

$$\begin{bmatrix} u_0 \\ v_0 \\ 1 \end{bmatrix} = \begin{bmatrix} u_2 & -u_5 & u_3 \\ v_2 & -v_5 & v_3 \\ 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} \lambda_2 \\ \lambda_5 \\ \lambda_3 \end{bmatrix}. \quad (5)$$

Re-writing (5) as the following equation:

$$\begin{bmatrix} 1-\lambda_2 \\ u_0 - \lambda_2 u_2 \\ v_0 - \lambda_2 v_2 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ u_3 & -u_5 \\ v_3 & -v_5 \end{bmatrix} \begin{bmatrix} \lambda_3 \\ \lambda_5 \end{bmatrix} \quad (6)$$

In (6), the necessary and sufficient condition of the existence of a solution of (λ_3, λ_5) is that the three vectors $[1\ -1]^T$, $[u_3\ \ -u_5]^T$, and $[v_3\ \ -v_5]^T$ are linearly dependent. This property induces that $(u_5\ v_5)$ should fall within a line, which is called the *auxiliary line* in this paper. For example, Fig. 4 shows some different auxiliary lines generated by dragging the control point $\boldsymbol{P_4}$ to different image positions. The auxiliary lines can help users to put the control points more correctly.

3.2 Image Sequence

An image sequence may contain two types of scenes: dynamic and static. We can use the approach introduced above to insert a moving virtual object such that its appearance looks geometrically consistent with respect to the static scene in an image sequence. To insert virtual objects whose motion is consistent with a moving object in a dynamic scene, we have developed a tracking procedure. First, six corner points of a cuboid structure in the first frame are selected manually. The size parameters *a, b, c* of the cuboid structure can be computed using the PCS method, and thus the object coordinates (or world coordinates) of the six corner points are also given. Assume that the cuboid structure is rigid (i.e., the sizes *a, b, c* are not varied), we can then treat the cuboid as a calibration object and re-calibrate the camera parameters by keeping tracking the calibration object. To solve the tracking problem, we formulate an energy function similar to that proposed in [14]. By minimizing the energy function we can keep tracking the calibration object and also re-calibrate the camera parameters simultaneously.

4. Experimental Results

We have implemented our method on a PC and using the OpenGL software for graphics rendering. We can manually set some light sources and generates planar shadows in the environment. Fig. 5 shows an example of computing the 3D structure by selecting the six control points of concave type. The user can then interactively move a virtual object to the desired positions and orientations on an intuitively meaningful ground plane, and generate a composite view as shown in Fig. 5(c). Fig. 6 shows the result of inserting a moving car into a single un-calibrated image. The motion of the virtual car was controlled with a mouse. This functionality can be used to develop more interesting video games on PC. Fig. 7 shows an example of applying our method to the insertion in an image sequence. As stated in Section 1, inserting a virtual object whose motion is relative to a moving real object (the problem A) can be achieved by using some existing calibration-free methods. However, inserting a virtual object whose motion is relative to a static real object (the problem B) can not be achieved using those calibration-free methods. Our method can solve both problems, as demonstrated in Fig. 7.

Figure 4 Different auxiliary lines (in which the fifth control point $\boldsymbol{P_5}$ should be lying on) generated by dragging the control point $\boldsymbol{P_4}$ to different image positions. The control points of concave type are used in this example.

5. Conclusions and Discussion

The major contribution of this paper is to propose a new method for estimating the camera parameters associated with an image, based on a parameterized cuboid structure (or a parallelepiped structure) seen in the image. The proposed PCS method can determine the sizes of the cuboid (or parallelepiped) as well as the intrinsic and extrinsic parameters of the camera. The PCS method has been successfully used for inserting virtual objects not

only in one single image but also in a sequence of images which contains static and/or dynamic scenes. In particular, the PCS method can be used for inserting moving or animated virtual objects into one single un-calibrated image, which can not be achieved by the existing methods. The PCS method can also be used for inserting virtual objects into image sequences if the cuboid structure keeps on appearing in the scene either explicitly or "implicitly" — i.e., can be inferred by human perception. To insert virtual objects into a single un-calibrated image, some human interaction is unavoidable. Our method provides an auxiliary line to help manually selecting the control points.

Inserting virtual objects in an image sequence remains to be a research issue. In fact, all the existing methods proposed to solve this problem has some limitations. For example, the method assuming an orthographic projection [10] can not be used if the real object is close to the camera center. The method using self-calibration [4] can not be applied if the motion of the real object contained in the scene is *critical* (e.g., pure translation, plane movements, rotation movements around a point, etc.) [4][14]. The CFAOS methods [9][12] require the user to draw the virtual objects in both of the first two frames of the image sequence in a geometrically consistent way. The proposed PCS method requires that a cuboid structure exists in the scene, either explicitly or implicitly. In our opinion, no methods can successfully handle all the cases, and hence a hybrid approach may be required. The PCS method proposed in this paper has greatly enriched the tools for handling the vision-based AR applications.

Appendix 1 (proof of Property 1)

Assume that the six points of two adjacent faces are {$\boldsymbol{P_0}$, $\boldsymbol{P_1}$, $\boldsymbol{P_2}$, $\boldsymbol{P_3}$, $\boldsymbol{P_4}$, $\boldsymbol{P_5}$} as shown in Fig. 1. The homogeneous coordinates of their image points are p_0, p_1, p_2, p_3, p_4, p_5, respectively. Based on the parallelepiped, we define an orthogonal world coordinate system that (i) the origin is $\boldsymbol{P_0}$, (ii) the **X** axis is along the direction from $\boldsymbol{P_0}$ to $\boldsymbol{P_1}$ (iii) the **Y** axis lies on the plane formed by {$\boldsymbol{P_0}$, $\boldsymbol{P_1}$, $\boldsymbol{P_3}$} and is orthogonal to the **X** axis, and (iv) the *Z* axis is orthogonal to both **X** and **Y** axes as shown in Fig. 1. We use P_i to represent the coordinate of $\boldsymbol{P_i}$ w. r. t. the world coordinate system defined above (in particular, $P_0 = [0\ 0\ 0]^T$). Then, the corresponding six camera equations are

$$\lambda_0 p_0 = Kt,\ \ \lambda_1 p_1 = K(RP_1 + \mathrm{t}),\ \ \lambda_2 p_2 = K(RP_2 + \mathrm{t})$$
$$\lambda_3 p_3 = K(RP_3 + \mathrm{t}),\ \ \lambda_4 p_4 = K(RP_4 + \mathrm{t}),\ \ \lambda_5 p_5 = K(RP_5 + \mathrm{t}) \quad \text{(A1)}$$

Since there always exists a scale factor which can not be calibrated, we set $\lambda_0 = 1$ (i.e., the distance from the lens center to P_0 is the *unit length*) and it will not affect the camera parameter estimation results. Because the two faces $\{P_0, P_1, P_4, P_2\}$ and $\{P_0, P_2, P_5, P_3\}$ are both parallelograms. Their depths $\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5$ can all be computed via lemma 1. So we can also compute all the V_i for *i=1,2,3,4,5,* where V_i is defined as

$$V_i \equiv \lambda_i p_i - Kt = \lambda_i p_i - \lambda_0 p_0 \quad \text{(A2)}$$

Via (A2), the eq. (A1) can be rewritten as $V_i = KRP_i$, or equivalently, $K^{-1}V_i = RP_i$, for *i=1,2,3,4,5.*

Consider the matrix A defined as

$$A \equiv \begin{bmatrix} V_1^T \\ V_2^T \\ V_3^T \end{bmatrix} K^{-T}K^{-1}\begin{bmatrix} V_1 & V_2 & V_3 \end{bmatrix} = \begin{bmatrix} P_1^TP_1 & P_1^TP_2 & P_1^TP_3 \\ P_2^TP_1 & P_2^TP_2 & P_2^TP_3 \\ P_3^TP_1 & P_3^TP_2 & P_3^TP_3 \end{bmatrix} \quad \text{(A3)}$$

Since

$$P_1^TP_2 = P_2^TP_1 = ab\cos(\theta_{ab}),\ P_2^TP_3 = P_3^TP_2 = bc\cos(\theta_{bc})$$
$$P_1^TP_3 = P_3^TP_1 = ac\cos(\theta_{ac}).$$

Also, $P_1^TP_1 = a^2$, $P_2^TP_2 = b^2$, $P_3^TP_3 = c^2$.

Hence, (A3) can be re-written as

$$V^TMV = \begin{bmatrix} a^2 & abC_{ab} & acC_{ac} \\ abC_{ab} & b^2 & bcC_{bc} \\ acC_{ac} & bcC_{bc} & c^2 \end{bmatrix} \quad \text{(A4)}$$

where $V \equiv [V_1\ V_2\ V_3]$,

$C_{ab} \equiv \cos(\theta_{ab}), C_{bc} \equiv \cos(\theta_{bc}), C_{ac} \equiv \cos(\theta_{ac})$, and

$$M \equiv K^{-T}K^{-1} = \begin{bmatrix} \alpha^2 & 0 & \alpha\beta \\ 0 & \alpha^2 & \alpha\gamma \\ \alpha\beta & \alpha\gamma & \beta^2+\gamma^2+1 \end{bmatrix}$$

($\alpha \equiv f^{-1}, \beta \equiv -f^{-1}u, \gamma \equiv -f^{-1}v$).

Notice that the physical meaning of *det(V)* is the volume of the parallelepiped, and hence V is nonsingular. So, (A4) can be equivalently written as

$$\begin{bmatrix} \alpha^2 & 0 & \alpha\beta \\ 0 & \alpha^2 & \alpha\gamma \\ \alpha\beta & \alpha\gamma & \beta^2+\gamma^2+1 \end{bmatrix} = U^T \begin{bmatrix} a^2 & abC_{ab} & acC_{ac} \\ abC_{ab} & b^2 & bcC_{bc} \\ acC_{ac} & bcC_{bc} & c^2 \end{bmatrix} U \quad \text{(A5)}$$

where $U \equiv V^{-1}$. Eq. (A5) has six unknowns α, β, γ and *a, b, c,* and totally six constrained equations can be listed from it. Hence it is possible to solve the problem now. Remember that the left side of (A5) is the matrix M. By using the constraints that $M_{12}=0$ and $M_{11} - M_{22}=0$ (where

M_{ij} is the ijth entry of M), we can obtain two quadratic equations in terms of a, b, c:

$$C_1a^2+C_2b^2+C_3c^2+C_4ab+C_5ac+C_6bc=0 \quad \text{(A6)}$$

$$C'_1a^2+C'_2b^2+C'_3c^2+C'_4ab+C'_5ac+C'_6bc=0 \quad \text{(A7)}$$

where C_i and C'_i ($i=1,...,6$) are constants which can be computed from U and C_{ab}, C_{bc}, C_{ac} using the right side of (A5). Let $x=b/a$ and $y=c/a$, then (A6) and (A7) can be written as

$$a^2 (C_2x^2+C_6xy +C_3y^2+C_4x+C_5y+C_1)=0 \quad \text{(A8)}$$

$$a^2 (C'_2x^2+C'_6xy +C'_3y^2+C'_4x+C'_5y+C_1)=0 \quad \text{(A9)}$$

Since the edge length $a \neq 0$, we have two quadratic equations consisting of two variables x, y in (A8) and (A9). The number of the joint points of two quadratic curves are maximal four, and these points can be solved by using a fourth-degree polynomial equation constructed by first solving y in terms of x (solving a quadratic equation) in (A8) and then replacing y in (A9). After solving x and y, the values α / a, β / a, γ / a can be computed by substituting x and y in the right side of (A5) and using the terms $M_{11}=\alpha^2$, $M_{13}=\alpha\beta$, and $M_{23}=\alpha\gamma$. Then, by using the terms $M_{33}=\beta^2+\gamma^2+1$, $1/a^2$ can be obtained. After that, the size parameters $a, b, c,$ and the intrinsic parameters f, u, v can all be easily obtained by tracing back their relations to the solved terms. The translation t is simply $K^{-1}P_0$. To solve the rotation matrix R, notice that it satisfies $R[P_1\ P_2\ P_3] = K^{-1}[V_1\ V_2\ V_3]$, and $[P_1\ P_2\ P_3]$ is an upper-triangular matrix according to the definitions of the world coordinate system. Hence, R can be obtained via the QR-decomposition of the matrix $K^{-1}[V_1\ V_2\ V_3]$. ■

Appendix 2 (proof of Property 2)

Since C_{ab}, C_{bc}, C_{ac} are all zeros for a cuboid, the eqs. (A6) and (A7) will be reduced to

$$C_1a^2+C_2b^2+C_3c^2=0 \text{ and } C'_1a^2+C'_2b^2+C'_3c^2=0 \quad \text{(A10)}.$$

By letting $x=b/a$, $y=c/a$, (A10) can be re-written to two linear equations in terms of x^2 and y^2.

$$C_2x^2+C_3y^2+ C_1=0 \text{ and } C'_2x^2+C'_3y^2+ C'_1=0 \quad \text{(A11)}.$$

After solving the linear systems (A11), x^2 and y^2 can be obtained. Then, other parameters can be solved by simple arithmetic as described in the proof of property 1. Notice that to solve R, it is not required to use the QR-decomposition now. Since $[P_1\ P_2\ P_3]$ is a diagonal matrix, the three columns of R is simply the unit vectors of the three columns of $K^{-1}[V_1\ V_2\ V_3]$, respectively. ■

References

[1] Y. Cho and U. Neumann, "Multi-ring Color Fiducial Systems for Scalable Fiducial Tracking Augmented Reality," *Proc. VRAIS98*, March, 1998.

[2] P. Debevec, C. J. Taylor, and J. Malik, "Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-based Approach," *Proc. SIGGRAPH'96*, pp. 11-21, 1996.

[3] O. Faugeras, "Stratification of Three-dimensional Vision: Projective, Affine, and Metric Representations," *J. Opt. Soc. Am. A*, Vol. 12, pp. 465-484, 1995.

[4] O. Faugeras, "From Geometry to Variational Calculus: Theory and Applications of Three-Dimensional Vision," *Proc. CVVRHC'98*, pp. 52-71, 1998.

[5] R. Haralick, *Computer and Robot Vision, Vol. II*, Addison-Wesley, New York, 1993.

[6] A. Heyden and K. Astrom, "Euclidean Reconstruction from Image Sequences with Varying and Unknown Focal Length and Principle Point," *Proc. CVPR'97*, pp. 438-443, 1997.

[7] Y. Horry, K. Anjyo, and K. Arai, "Tour into the Picture," *Proc. SIGGRAPH'97*, pp. 225-232, 1997.

[8] D. Koller, et al., "Real-time Vision-Based Camera Tracking for Augmented Reality Applications," *Proc. VRST'97*, pp. 87-94, 1997.

[9] K. N. Kutulakos and J. R. Vallino, "Calibration-free Augmented Reality," *IEEE Trans. Visul. and Graph.*, Vol. 4, pp. 1-20, 1998.

[10] K. C. Ong, H. C. Teh, and T. S. Tan, "Resolving Occlusion in Image Sequence Made Easy," *The Visual Computer,* Vol. 14, pp. 153-165, 1998.

[11] M. Pollefeys, R. Koch, and L. V. Gool, "Self-Calibration and Metric Reconstruction in spite of Varying and Unknown Internal Camera Parameters," *Proc. ICCV'98*, pp. 90-95, 1998.

[12] Y. Seo, M. H. Ahn, and K. S. Hong, "Video Augmentation by Image-based Rendering under the Perspective Camera Model," *Proc. ICPR'98*, pp. 1694-1697, 1998.

[13] H. Y. Shum, et al., "Interactive 3D Modeling from Multiple Images Using Scene Regularities," *Proc. SMILE'98, Lecture Notes in Comp. Sci., Vol. 1506*, pp. 236-252, 1998.

[14] G. Simon and M. O. Berger, "A Two-Stage Robust Statistical Method for Temporal Registration from Features of Various Type," *Proc. ICCV'98,* pp. 261-266, 1998.

[15] P. Sturm, "Critical Motion Sequences for Monocular Self-Calibration and Uncalibrated Euclidean Reconstruction," *Proc. CVPR'97*, pp. 1100-1105, 1997.

[16] C. Tomasi and T. Kanade, "Shape and Motion from Image Streams Under Orthography: A Factorization Method," *IJCV*, Vol. 9, pp. 137-154, 1992.

[17] R. Y. Tsai, "A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-The-Shelf TV Cameras and Lenses," *IEEE Trans. Robo. and Auto.*, Vol. 3, pp. 323-344, 1987.

[18] Y. Wu et al., "A New General Computational Framework for Finding Object Orientation Using Perspective Trihedral Angle Constraint," *IEEE Trans. PAMI*, Vol. 16, pp. 961-975, 1994.

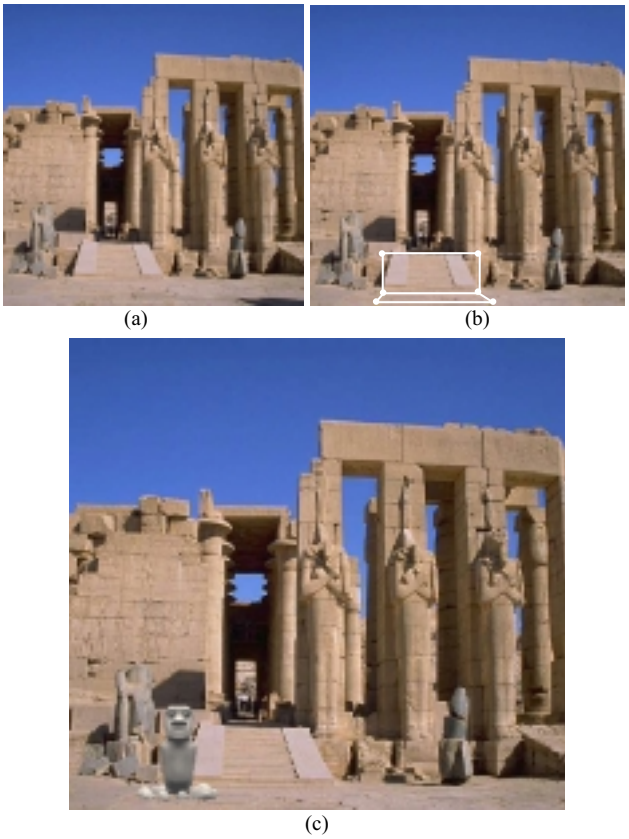

Figure 5. An example of inserting a single virtual object. (a) The original image. (b) The selected control points of two adjacent faces of a cuboid (which is of concave type). After performing the PCS method, a virtual object is inserted into the image, as shown in (c).

Figure 6. An example of inserting a moving object into a single un-calibrated image. An image sequence is generated by inserting a virtual car driving along the road. Notice that we only have to perform the camera parameter estimation once, and then the motion of the virtual car can be controlled with a mouse in an easy and intuitive way.

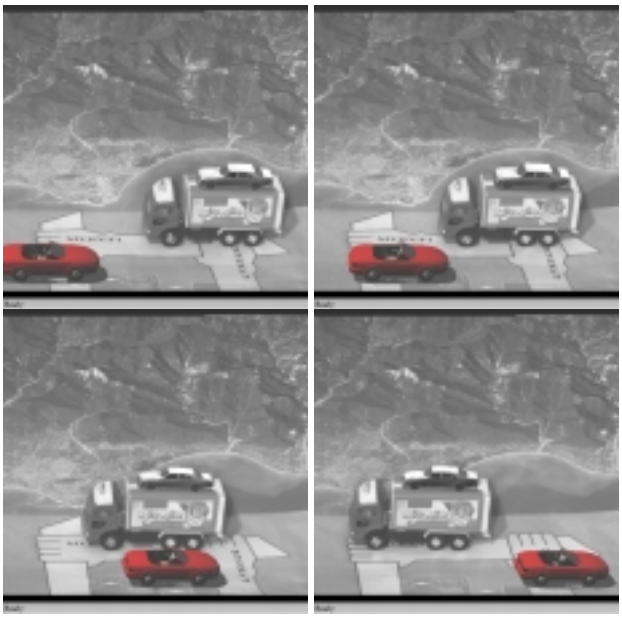

Figure 7. An example of inserting virtual objects into an image sequence containing a moving truck and a static background. A virtual convertible is inserted on the ground, driving toward the opposite direction of the truck's motion. Another virtual sedan was inserted on top of the truck with consistent motion.