On Pose Recovery for Generalized Visual Sensors

Chu-Song Chen, *Member*, *IEEE*, and Wen-Yan Chang

Abstract—With the advances in imaging technologies for robot or machine vision, new imaging devices are being developed for robot navigation or image-based rendering. However, to satisfy some design criterion, such as image resolution or viewing ranges, these devices are not necessarily being designed to follow the perspective rule and, thus, the imaging rays may not pass through a common point. Such generalized imaging devices may not be perspective and, therefore, their poses cannot be estimated with traditional techniques. In this paper, we propose a systematic method for pose estimation of such a generalized imaging device. We formulate it as a nonperspective n point (NP*n*P) problem. The case with exact solutions, $n = 3$, is investigated comprehensively. Approximate solutions can be found for $n > 3$ in a least-squared-error manner by combining an initial-pose-estimation procedure and an orthogonally iterative procedure. This proposed method can be applied not only to nonperspective imaging devices but also perspective ones. Results from experiments show that our approach can solve the NP*n*P problem accurately.

Index Terms—Computer vision, camera pose estimation, generalized imaging device (GID), perspective n point problem (P*n*P), nonperspective n point problem (NP*n*P).

1 Introduction

How to estimate the pose of a camera by determining the rigid transformation, which relates images to known geometry, is a problem of fundamental importance to machine and robot vision. In the past, many methods were developed for solving the pose estimation problem for perspective imaging devices, where the imaging rays are assumed to intersect at a common point. For some applications such as telepresence and image-based virtual reality, the perspective property has to be taken into account, because the generated images are supposed to be presented to humans. However, for some other applications such as automatic visual surveillance and mobile robot guidance, the imaging system need not comply with the perspective rule.

In fact, many new types of imaging methodologies or devices, violating the perspective construction in machine and robot vision, were designed. That is, the imaging rays of each of them may not intersect at a common point. For example, Rademacher and Bishop introduced the concept of images with multiple centers of projection, which were applied to image-based rendering [26]. A linear pushbroom camera [13] contains multiple focal centers distributed in a line and is thus nonperspective as well. In addition, Huang et al. [11] proposed a polycentric panorama model, where the focal centers were distributed in circles. In fact, it is also possible to acquire a nonperspective image in a single shot. For instance, wide-angle lens systems including severe projective distortions may have a locus of viewpoints [24]. An omnidirectional vision sensor combining a camera and a conic mirror, which was employed for collision avoidance of robotics, is another example of a nonperspective imaging device [33].

However, systematic methods for pose estimation of a nonperspective imaging device are lacking. In this paper, we propose a pose estimation method for an arbitrary imaging device that can be perspective or nonperspective. The rest of the paper is organized as follows: Section 2 gives a formulation of the problems. Sections 3 and 4 introduce the nonperspective pose estimations when three and n ($n > 3$) point correspondences are given, respectively. Simulated and experimental results are shown in Sections 5 and 6, respectively. Finally, some conclusions and discussion are given in Section 7.

2 Problem Formulation

First, we formulate the model of imaging devices considered in this paper. In essence, an imaging device captures the rays of lights in 3D space. These rays are occluded by the physical occupation of the imaging device itself, with which the end points of these rays are inherently determined. Hence, an imaging device can be generally formulated by the three components, (*I, CCS, L*), as defined below.

1. $I(\cdot,\cdot)$: $D_I \rightarrow R \times G \times B$ is an image map ($D_I \subset \mathcal{R}^2$ is the domain of image *I*), and R, G, B are the sets consisting of the three primitive colors.
2. *CCS*: an arbitrary Euclidean coordinate frame selected in 3D space, which is referred to as the *camera coordinate system* (CCS).
3. $L(\cdot,\cdot)$: $D_I \rightarrow \mathcal{R}^3 \times \mathcal{R}^3$ is a mapping from an image point, say (i, j), to the 3D ray represented as $(\boldsymbol{c}, \boldsymbol{v})$ with respect to CCS, which consists of all the 3D points that can be imaged at (i, j), where $\boldsymbol{c} \in \mathcal{R}^3$ is the end point and $\boldsymbol{v} \in \mathcal{R}^3$ is the unit directional vector of this ray.

• *The authors are with the Institute of Information Science, Academia Sinica, 128 Academic Rd., Sec. 2, Nankang 115, Taipei, Taiwan. E-mail: {song, wychang}@iis.sinica.edu.tw.*

Manuscript received 15 Mar. 2002; revised 12 Nov. 2002; accepted 9 June 2003.
Recommended for acceptance by S. Dickinson.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 116094.

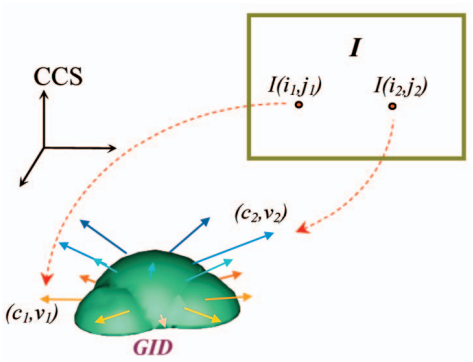

Fig. 1. An illustrative example of a generalized imaging device (GID).

The model formulated above is called a *generalized imaging device* (GID) in this paper. Fig. 1 gives an illustration of the GID.[1]

Given a GID $\mathbf{G}$, let $\Gamma(\mathbf{G}) = \{l$: a 3D line $|\exists(\boldsymbol{c}, \boldsymbol{v})$ s.t. the ray specified by $(\boldsymbol{c}, \boldsymbol{v})$ is contained in the full line $l\}$. If all of the lines contained in $\Gamma(\mathbf{G})$ intersect at a common point, then $\mathbf{G}$ is called *perspective*. Otherwise, $\mathbf{G}$ is *nonperspective*.

For example, a common video camera is usually modeled as a perspective GID. An omnidirectional imaging system combining a hyperbolic mirror and a camera, where the lens center of the camera is placed in one of the focal points of the hyperbolic mirror, is another example of a perspective GID [34]. On the other hand, a linear pushbroom camera and some wide-angle lens systems, which have been introduced in Section 1, are examples of nonperspective GIDs. In essence, the concept of the GID is suitable for formulating the geometrical relation of optical apparatuses designed for capturing images in a 3D environment. Considering such a general definition of imaging devices, a basic problem would be as follows: *Given a set of 3D points with respect to a world coordinate system (WCS) and their projecting points in the image plane of a GID, how can the rigid transformation between the world and the camera coordinate systems be computed?* Such a fundamental problem is called the *perspective n point problem* (PnP) for perspective imaging devices [8], [9], [14], [22], [23], [35]. This study investigates a more general problem for the GIDs, which we call *nonperspective n point problem* (NPnP) because the GIDs considered herein need not be perspective [4].

The PnP problem has been well investigated. Closed-form solutions have been formulated if three or four 3D-2D correspondences are adopted [9], [14], [15]. However, if more correspondences are used, there are no closed-form solutions available. Lowe [22] and Yuan [35] used the Newton-Raphson method for pose estimation under the assumption that approximate initial poses were provided. The DeMenthon and Davis approach [8] first assumed the camera model as a scaled orthographic projection. It obtains the rigid transformation by solving a linear system, and then uses a POSIT procedure to refine the result iteratively. Lu et al. [23] also proposed an iteration method using a scaled orthographic projection for initial estimation.

However, the NPnP problem has not been well researched in past studies. In Sections 3 and 4, we will propose a systematic method for solving the NPnP problem. Since the PnP problem is a special case of the NPnP problem, our method can be used for solving the PnP problem as well.

3 Nonperspective Three-Point (NP3P) Problem

In an NPnP problem, n points with known coordinates with respect to a WCS in the 3D space, e.g., $P_1, P_2, \ldots, P_n$, are supposed to be imaged with a GID. Assume that their 2D image points are $(i_1, j_1), (i_2, j_2), \ldots, (i_n, j_n)$, respectively, where $(i_k, j_k) \in \mathrm{D_I}$ for all $k = 1, \ldots, n$. When the intrinsic model of the GID is known, we want to find the rigid transformation between CCS and WCS so that $\boldsymbol{Q}_k = \boldsymbol{R} \cdot \boldsymbol{P}_k + \boldsymbol{t}$, where $\boldsymbol{R}$ is a 3×3 rotation matrix, $\boldsymbol{t}$ is a 3×1 translation vector, and $\boldsymbol{Q}_k$ is a point in CCS that can be represented as $\boldsymbol{Q}_k = s_k \cdot \boldsymbol{v}_k + \boldsymbol{c}_k$, in which $(\boldsymbol{c}_k, \boldsymbol{v}_k) = L(i_k, j_k)$ and s_k is a scale factor for all $k = 1, \ldots, n$.

First, we investigate the problem when $n = 3$, which is the minimum number of 3D-2D correspondences needed to allow the solutions to be identified exactly. The induced problem is called the NP3P problem in this study.

3.1 Solutions of the NP3P Problem

When $n = 3$, the three points P_1, P_2, and P_3 with known coordinates with respect to the WCS form a triangle. Since the coordinates of P_1, P_2, and P_3 are known, the length of the three edges of the triangle, a, b, c, can be determined. Consider $\boldsymbol{Q}_1$, $\boldsymbol{Q}_2$, and $\boldsymbol{Q}_3$, the transformed points of P_1, P_2, and P_3, lying on the corresponding rays, $\boldsymbol{l}_1$, $\boldsymbol{l}_2$, and $\boldsymbol{l}_3$, respectively, as shown in Fig. 2a, where $\boldsymbol{l}_k = (\boldsymbol{c}_k, \boldsymbol{v}_k), k = 1, 2, 3$ are three lines of a GID with known intrinsic model (and, thus, $\boldsymbol{c}_k$ and $\boldsymbol{v}_k, k = 1, 2, 3$ are all known to the CCS of the GID). Denote l_k to be the full line containing the ray $\boldsymbol{l}_k, k = 1, 2, 3$, and let l_{ij} be the common orthogonal line between l_i and $l_j, i, j = 1, 2, 3$ and $i \neq j$, respectively. Denote $\boldsymbol{q}_{ij}$ and $\boldsymbol{q}_{ji}$ to be the intersecting points between l_i and l_{ij}, and l_j and l_{ij}, respectively. Let the distance between $\boldsymbol{q}_{ij}$ and $\boldsymbol{q}_{ji}$ be d_{ij}. Without loss of generality, consider the coordinate system whose origin is $\boldsymbol{q}_{12}$ and whose **x**-axis is defined to be along the same direction of l_{12}. In addition, the **y**-axis of this coordinate system is defined to be along the direction of l_1, and the **z**-axis is defined to be the cross product of **x** and **y** axes. By using this coordinate system, $\boldsymbol{Q}_1$ and $\boldsymbol{Q}_2$ can be represented as $\boldsymbol{Q}_1 = (0, t_1, 0)$ and $\boldsymbol{Q}_2 = (d_{12}, t_2 cos\theta_{12}, t_2 sin\theta_{12})$, where t_1 is the distance between $\boldsymbol{q}_{12}$ and $\boldsymbol{Q}_1, t_2$ is the distance between $\boldsymbol{q}_{21}$ and $\boldsymbol{Q}_2$, and θ_{12} is the angle between the directions of l_1 and l_2 (i.e.,$\theta_{12} = acos(\boldsymbol{v}_1, \boldsymbol{v}_2)$). Because the distance between $\boldsymbol{Q}_1$ and $\boldsymbol{Q}_2$ is a, we have

$$\begin{aligned} a^2 &= d_{12}^2 + (t_2 cos\theta_{12} - t_1)^2 + (t_2 sin\theta_{12})^2 \\ &= d_{12}^2 + (t_2^2 cos^2\theta_{12} - 2t_1 t_2 cos\theta_{12} + t_1^2) + (t_2^2 sin^2\theta_{12}) \\ &= d_{12}^2 - 2t_1 t_2 cos\theta_{12} + t_1^2 + (t_2^2 cos^2\theta_{12} + t_2^2 sin^2\theta_{12}) \\ &= d_{12}^2 - 2t_1 t_2 cos\theta_{12} + t_1^2 + t_2^2. \end{aligned}$$

1. In [10], Grossberg and Nayar presented a more general imaging model in which an image point corresponds to a bundle of rays, and it is useful for identifying the point spread function for each pixel. Since we focus on the geometrical calibration of the imaging devices in this paper, the imaging model is formulated by associating an image point with a single ray, which considerably simplifies the problem for estimating the parameters for rigid transformations.

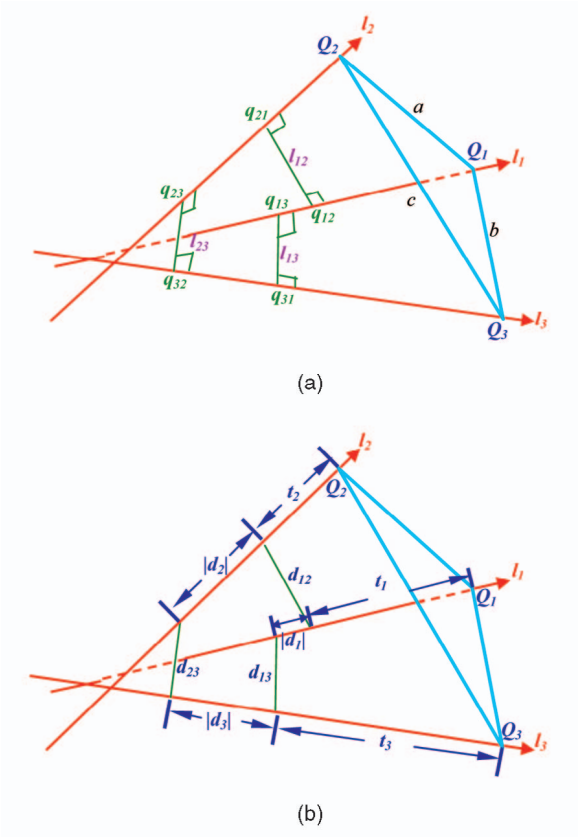

Fig. 2. Illustration of the definitions associated with the NP3P problem.

Hence,

$$t_2^2 - 2t_1t_2cos\theta_{12} + t_1^2 = a^2 - d_{12}^2. \quad (1)$$

Note that the parameters used for describing (1), including the distance between $\boldsymbol{q}_{12}$ (or $\boldsymbol{q}_{21}$) and $\boldsymbol{Q}_1$ (or $\boldsymbol{Q}_2$) and the angle between v_1 and v_2, are all independent of the coordinate systems being selected. Hence, consider the line pair (l_1, l_3), we also have the following constraint by using the property that the distance between $\boldsymbol{Q}_1$ and $\boldsymbol{Q}_3$ is b:

$$t_3^2 - 2(t_1')t_3cos\theta_{13} + (t_1')^2 = b^2 - d_{13}^2, \quad (2)$$

where t_3 is the distance between $\boldsymbol{q}_{31}$ and $\boldsymbol{Q}_3$, and t_1' is the distance between $\boldsymbol{q}_{13}$ and $\boldsymbol{Q}_1$. When $d_1 = t_1 - t_1'$ (and, thus, the distance between $\boldsymbol{q}_{12}$ and $\boldsymbol{q}_{13}$ is $|d_1|$), then the following equation holds:

$$t_3^2 - 2(t_1 - d_1)t_3cos\theta_{13} + (t_1 - d_1)^2 = b^2 - d_{13}^2. \quad (3)$$

Similarly, consider the line pair (l_2, l_3), then we also have:

$$(t_3 - d_3)^2 - 2(t_3 - d_3)(t_2 - d_2)cos\theta_{23} + (t_2 - d_2)^2 = c^2 - d_{23}^2, \quad (4)$$

where $t_3 - d_3$ (or $t_2 - d_2$) is the distance between $\boldsymbol{q}_{32}$ (or $\boldsymbol{q}_{23}$) and $\boldsymbol{Q}_3$ (or $\boldsymbol{Q}_2$).

Equations (1), (3), and (4) give three constraints on the three unknowns, t_1, t_2, and t_3. Generally, since each of (1), (3), and (4) is a quadratic polynomial equation associated with two unknowns, the solutions can be obtained by solving eighth-order polynomial equations with a single variable, as shown in the following: From (1) and (3), t_2 and t_3 can both be represented as t_1, respectively:

$$t_2 = t_1cos\theta_{12} \pm (a^2 - d_{12}^2 - t_1^2sin^2\theta_{12})^{1/2}, \quad (5)$$

$$t_3 = (t_1 - d_1)cos\theta_{13} \pm [b^2 - d_{13}^2 - (t_1 - d_1)^2sin^2\theta_{13}]^{1/2}. \quad (6)$$

Substituting (5) and (6) into (4), we can derive an equation of the following form:

$$A_2 \pm A_1(B_2)^{1/2} = \pm B_1(C_2)^{1/2} \pm 2cos\theta_{23}(B_2C_2)^{1/2}, \quad (7)$$

where

$$\begin{aligned}
B_2 &= a^2 - d_{12}^2 - t_1^2sin^2\theta_{12},\\
C_2 &= b^2 - d_{13}^2 - (t_1 - d_1)^2sin^2\theta_{13},\\
A_1 &= 2B - 2Acos\theta_{23},\\
A_2 &= A^2 + B^2 - c^2 + d_{23}^2 - 2ABcos\theta_{23} + B_2 + C_2,\\
B_1 &= 2Bcos\theta_{23} - 2A,\\
A &= (t_1 - d_1)cos\theta_{13} - d_3,
\end{aligned}$$

and

$$B = t_1cos\theta_{12} - d_2.$$

Taking the square of both sides of (7), we obtain

$$\begin{aligned}
&A_2^2 + A_1^2B_2 \pm 2A_2A_1(B_2)^{1/2}\\
&= B_1^2C_2 + 4cos^2\theta_{23}B_2C_2 \pm 4cos\theta_{23}B_1C_2(B_2)^{1/2}
\end{aligned}$$

or, equivalently:

$$\begin{aligned}
&A_2^2 + A_1^2B_2 - B_1^2C_2 - 4cos^2\theta_{23}B_2C_2\\
&= 2(\pm 2cos\theta_{23}B_1C_2 \pm A_2A_1)(B_2)^{1/2}.
\end{aligned} \quad (8)$$

Taking the square of both sides of (8) yields eighth-order polynomial equations in terms of t_1.

Although there is no analytic way to solve a polynomial equation of the eighth order, its solution is not difficult to find numerically. Polynomial root finding is a traditional topic in numerical analysis. Three of the popular general-purpose polynomial root finders include the method of computing the eigenvalues of a companion matrix (which was used in the EISPACK routine [29]), the Jenkins/Traub method [18] that works with the polynomial itself, and the method consisting of a combination of Muller's and Newton's method [25], [19]. According to [19], all these three methods yield comparable results regarding speed and accuracy when working with low degree ($n < 20$) polynomials and, thus, are suitable for solving t_1 in our problem ($n = 8$). Then, the other coefficients, t_2 and t_3, can be obtained by substituting the solution of t_1 into (5) and (6). The a priori knowledge of the 3D points located on the positive direction of the corresponding rays, i.e., $s_k > 0, k = 1, 2, 3$, can be used to eliminate inappropriate solutions as well.

Two singular cases (i.e., cases with infinite solutions) of the NP3P problem are discussed as follows: The first case happens when the three points are all in a line, and is referred to as the colinear case. This is because the transformation obtained by coupling a solution with any rotation around this line remains a solution. The second case, referred to as the parallel-ray case, occurs when all three rays are parallel to each other, as will be analyzed in

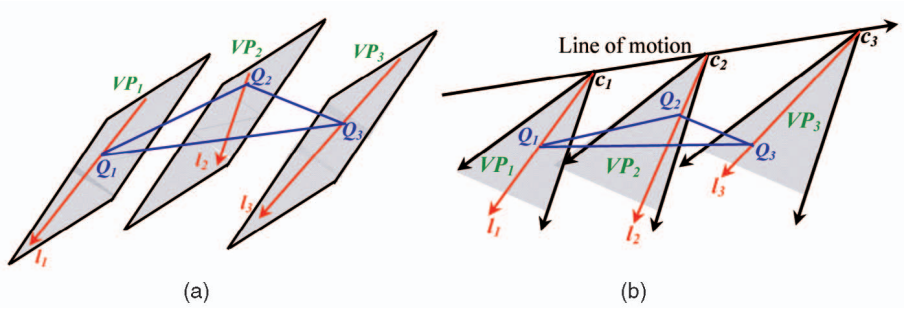

Fig. 3. (a) Illustration of the parallel plane case. (b) Illustration of the linear pushbroom camera model.

the orthographic-projection case (i.e., Case 5) in Section 3.2. These two singular cases will be employed for removing unsuitable triples of 3D points in finding an initial estimation for the NPnP problem, as shown in Section 4.1. A more formal analysis of singular cases of the general NPnP problem will be given in Section 4.2.

3.2 Special Cases of the NP3P Problem

To solve a general NP3P problem requires solving the eighth-order polynomial equations as described above. In this section, we investigate some special cases of the NP3P problem whose solutions can be obtained by solving polynomial equations whose orders are at most four instead of eight. Since there are analytical representations of the solutions of a fourth-order polynomial equation, the solutions of these special cases can be expressed in closed forms.

Case 1 [Partially Parallel]. We call an NP3P problem *partially parallel* if any two of the three lines, l_1, l_2, l_3, are parallel. Without loss of generality, assume that l_1 and l_2 are parallel, i.e., $\theta_{12} = 0$ and, thus, $sin\theta_{12} = 0$. The term B_2 in (8) then becomes $B_2 = a^2 - d_{12}^2 - t_1^2 sin^2\theta_{12} = a^2 - d_{12}^2$, which is a constant term irrelevant with t_1. Hence, (8) becomes fourth-order polynomial equations in terms of t_1. Therefore, analytical solutions can be obtained for the partially parallel model.

Case 2 [Perspective]. In this case, the three lines, l_1, l_2, and l_3, intersect at a common point, which forms the standard P3P problem which has been investigated in previous works [9], [14]. According to our formulation, $d_1 = d_2 = d_3 = 0$ and $d_{12} = d_{13} = d_{23} = 0$ in this case, and (1), (3), and (4) degenerate to the same forms of (1), (2), and (3) of [14]. It has been shown in [14] that a fourth-order polynomial equation can be derived from those three equations. Recently, an algorithm, CASSC, was proposed to find complete and robust real-number solutions to the P3P problem [12].

Case 3 [Parallel Plane]. We refer to the NP3P problem as the case of *parallel plane* if there are three parallel planes, VP_1, VP_2, and VP_3, such that the three rays, $\boldsymbol{l}_1 = (\boldsymbol{c}_1, \boldsymbol{v}_1)$, $\boldsymbol{l}_2 = (\boldsymbol{c}_2, \boldsymbol{v}_2)$, and $\boldsymbol{l}_3 = (\boldsymbol{c}_3, \boldsymbol{v}_3)$ lie on VP_1, VP_2, and VP_3, respectively, as shown in Fig. 3a (however, the three rays are not necessarily parallel to each other). This case cannot be analyzed easily by using the coordinate system as defined in Section 3.1. Therefore, a new coordinate system is used where the origin is $\boldsymbol{c}_1$, the **x**-axis is along $\boldsymbol{v}_1$ (i.e., the direction of $\boldsymbol{l}_1$), the **z**-axis is along the normal of VP_1, and the **y**-axis is the cross product of **z** and **x** axes, respectively. Since VP_1 is the **x-y** plane of this coordinate system and both VP_2 and VP_3 are parallel to VP_1, $\boldsymbol{c}_k$ can be represented as (c_{kx}, c_{ky}, L_k) and $\boldsymbol{v}_k$ can be represented as $(v_{kx}, v_{ky}, 0)$ for $k = 2$ and 3, where $v_{kx}, v_{ky}, c_{kx}, c_{ky} \in \mathcal{R}$ and L_k is the distance between VP_1 and VP_k, $k = 2$ or 3. Hence, the coordinates of the transformed points, $\boldsymbol{Q}_1$, $\boldsymbol{Q}_2$, and $\boldsymbol{Q}_3$ in this coordinate system are $\boldsymbol{Q}_1 = (d, 0, 0)$, $\boldsymbol{Q}_2 = (c_{2x} + s_1 v_{2x}, c_{2y} + s_1 v_{2y}, L_2)$, and $\boldsymbol{Q}_3 = (c_{3x} + s_2 v_{3x}, c_{3y} + s_2 v_{3y}, L_3)$, respectively, where $s_1, s_2 \in \mathcal{R}$, and d is the distance between $\boldsymbol{c}_1$ and $\boldsymbol{Q}_1$. According to the three edge lengths $\{a, b, c\}$ of the triangle formed by $\boldsymbol{Q}_1$, $\boldsymbol{Q}_2$, and $\boldsymbol{Q}_3$, we have the following equations in terms of three unknowns s_1, s_2, and d:

$$(c_{2x} + s_1 v_{2x} - d)^2 + (c_{2y} + s_1 v_{2y})^2 + L_2^2 = a^2, \tag{9}$$

$$(c_{3x} + s_2 v_{3x} - d)^2 + (c_{3y} + s_2 v_{3y})^2 + L_3^2 = b^2, \tag{10}$$

$$(c_{2x} + s_1 v_{2x} - c_{3x} - s_2 v_{3x})^2 + (c_{2y} + s_1 v_{2y} - c_{3y} - s_2 v_{3y})^2 + (L_2 - L_3)^2 = c^2. \tag{11}$$

By eliminating the unknown d in (9) and (10), the following equation can be derived:

$$\begin{aligned} & c_{2x} + s_1 v_{2x} \pm (a' - \alpha^2)^{1/2} = c_{3x} + s_2 v_{3x} \pm (b' - \beta^2)^{1/2} \\ \Rightarrow\ & (c_{2x} + s_1 v_{2x} - c_{3x} - s_2 v_{3x})^2 = b' - \beta^2 + a' - \alpha^2 \pm \\ & \quad 2([b' - \beta^2][a' - \alpha^2])^{1/2}, \end{aligned} \tag{12}$$

where $a' = a^2 - L_2^2$, $b' = b^2 - L_3^2$, $\alpha = c_{2y} + s_1 v_{2y}$, and $\beta = c_{3y} + s_2 v_{3y}$. By substituting $(c_{2x} + s_1 v_{2x} - c_{3x} - s_2 v_{3x})^2$ in (11) with the right-hand side of (12), we have $b' - \beta^2 + a' - \alpha^2 \pm 2([b' - \beta^2][a' - \alpha^2])^{1/2} + (\alpha - \beta)^2 = c'$, where $c' = c^2 - (L_2 - L_3)^2$. Hence,

$$\begin{aligned} & b' + a' \pm 2([b' - \beta^2][a' - \alpha^2])^{1/2} = c' + 2\alpha\beta \\ \Rightarrow\ & \pm 2([b' - \beta^2][a' - \alpha^2])^{1/2} = d' + 2\alpha\beta \\ & (\text{where } d' = c' - b' - a') \\ \Rightarrow\ & 4([b' - \beta^2][a' - \alpha^2]) = (d' + 2\alpha\beta)^2 \\ \Rightarrow\ & 4a'b' - 4a'\beta^2 - 4b'\alpha^2 + 4\alpha^2\beta^2 = d'^2 + 4d'\alpha\beta + 4\alpha^2\beta^2 \\ \Rightarrow\ & 4a'\beta^2 + 4b'\alpha^2 + 4d'\alpha\beta = 4a'b' - d'^2. \end{aligned} \tag{13}$$

In addition, by substituting $s_1 = (\alpha - c_{2y})/v_{2y}$ and $s_2 = (\beta - c_{3y})/v_{3y}$ into (11), we have another equation in terms of α and β.

$$(c_{2x} + (v_{2x}/v_{2y})(\alpha - c_{2y}) - c_{3x} - (v_{3x}/v_{3y})(\beta - c_{3y}))^2 + (\alpha - \beta)^2 = c'. \tag{14}$$

Since both (13) and (14) are quadratic equations in terms of α and β, a fourth-order polynomial equation in terms of a single variable α (or β) can be derived by eliminating another unknown β (or α) from (13) and (14). Hence, the pose estimation problem of the parallel planes case can be solved analytically as well.

Case 4 [Linear Pushbroom]. All the rays of a linear pushbroom camera [11] are emitted from a line (referred to as the line of motion) and lie on parallel planes (referred to as the view planes), as shown in Fig. 3b. The linear pushbroom camera model can be used to model X-ray imageries and local behaviors of satellite images. It is obvious that the linear pushbroom camera model is a special case of the parallel plane case analyzed above and, thus, can be solved analytically as well.

Case 5 [Orthographic]. In orthographic projection, all rays are parallel, which is both a special case of the partially parallel model and the parallel plane model discussed above. In this case, $cos\theta_{12} = cos\theta_{13} = cos\theta_{23} = 1$ and $d_1 = d_2 = d_3 = 0$. Therefore, (1), (3), and (4) become $(t_2 - t_1)^2 = a^*$, $(t_3 - t_1)^2 = b^*$, and $(t_3 - t_2)^2 = c^*$, respectively, where $a^* = a^2 - d_{12}^2$, $b^* = b^2 - d_{13}^2$, and $c^* = c^2 - d_{23}^2$. They can be further simplified to the following three linearly dependent equations in terms of t_1, t_2, and t_3: $t_2 - t_1 = \pm a''$, $t_1 - t_3 = \pm b''$, and $t_3 - t_2 = \pm c''$, where $(a'')^2 = a^*$, $(b'')^2 = b^*$, and $(c'')^2 = c^*$. Hence, the solutions exist if and only if $\pm a'' \pm b'' \pm c'' = 0$, and in this case there are infinite number of solutions, $t_2 = t_1 \pm a''$ and $t_3 = t_1 \pm b''$, for all t_1. The existence of an infinite number of solutions is because the projected image remains the same under orthographic projection when an object is translated along the ray direction.

4 Nonperspective *N* Point (NP*n*P) Problem

The analysis in Section 3 shows that exact solutions can be identified for the NP*n*P problem when $n = 3$. However, when $n > 3$, exact solutions may not exist due to image noises. It is therefore necessary to find approximate solutions. In this paper, we developed a systematic method that finds an initial estimate of the approximate solutions first, as introduced in Section 4.1. Then, an iterative optimization procedure is proposed for refining the solutions, as introduced in Section 4.2.

4.1 Initialization for the NP*n*P Problem

The idea of our approach to initialization of the NP*n*P problem is to exploit the solutions of the three-point case. In general, there are $n!/(3!(n-3)!)$ triples of 3D-2D correspondences that can be served as initial estimates. The desire is to find that one that is better than the others (better in the sense that the triple gives a more accurate estimate than the other triples). The initialization procedure is shown as follows.

Algorithm 1: Consider n 3D points, $\boldsymbol{P}_1$, $\boldsymbol{P}_2$, ... $\boldsymbol{P}_n$, with known coordinates with respect to a WCS, and the corresponding 2D image points $\boldsymbol{p}_1$, $\boldsymbol{p}_2$, ..., $\boldsymbol{p}_n$, where $\boldsymbol{p}_m \in \mathrm{D_I}$ for all $m = 1, \ldots, n$. Assume that the rays associated with these points are $(\boldsymbol{c}_1, \boldsymbol{v}_1)$, $(\boldsymbol{c}_2, \boldsymbol{v}_2)$, ..., $(\boldsymbol{c}_n, \boldsymbol{v}_n)$, and the full lines containing these rays are $l_1, l_2, \ldots, l_n$, with respect to the CCS, respectively.

Step 1. Repeat *Steps 1.1-1.4 K* times, where *K* is a positive integer.

1.1. Select three points in $\{\boldsymbol{P}_1, \boldsymbol{P}_2, \ldots, \boldsymbol{P}_n\}$ randomly. Assume that they are $\boldsymbol{P}_i, \boldsymbol{P}_j$, and $\boldsymbol{P}_k$.

1.2. If one of the following conditions happens, then go back to *Step 1*: 1) $\max(|\boldsymbol{v}_i^T \boldsymbol{v}_j|, |\boldsymbol{v}_j^T \boldsymbol{v}_k|, |\boldsymbol{v}_i^T \boldsymbol{v}_k|)$ is smaller than a given threshold. 2) The angle between the two vectors, $\boldsymbol{P}_{ji}$ and $\boldsymbol{P}_{ki}$, where $\boldsymbol{P}_{ji} = \boldsymbol{P}_j - \boldsymbol{P}_i$ and $\boldsymbol{P}_{ki} = \boldsymbol{P}_k - \boldsymbol{P}_i$, is too small. 3) The area of the triangle formed by the three selected points in the image, $\boldsymbol{p}_i$, $\boldsymbol{p}_j$, and $\boldsymbol{p}_k$, is too small.

1.3. Compute the rigid transformations between WCS and CCS, which are associated with the three point-line pairs, $(\boldsymbol{P}_i, l_i)$, $(\boldsymbol{P}_j, l_j)$, and $(\boldsymbol{P}_k, l_k)$.

1.4. For each rigid transformation computed in Step *1.3*, say, $(\boldsymbol{R}, \boldsymbol{t})$, transform all the other 3D points with this rigid transformation by $\boldsymbol{P}'_m = \boldsymbol{R}\boldsymbol{P}_m + \boldsymbol{t}$, where $m = 1, \ldots, n$. Compute the sum of squared distances (SSD) between $\boldsymbol{P}'_m$ and $l_m, m = 1, \ldots, n$. Record both the SSD value *e* and its corresponding rigid transformation $(\boldsymbol{R}, \boldsymbol{t})$.

Step 2. Let *e* be the smallest among all the SSD values recorded in *Step 1.3*, and let $(\boldsymbol{R}, \boldsymbol{t})$ be the recorded rigid transformation corresponding to *e*. Output $(\boldsymbol{R}, \boldsymbol{t})$.

The above algorithm computes a rigid transformation with the smallest error among several selections of the triples of point-line correspondences. In *Step* 1.2, some triples of point-line correspondences randomly selected from *Step* 1.1 are dropped because they are not appropriatefor pose estimation. Rule 1) drops the triples whose three rays are too parallel to each other. Rule 2) drops the triples whose three world points are too co-linear. Both 1) and 2) drop the triples that are close to the singular cases shown in the end of Section 3.1, which lead to infinite approximate solutions and, thus, are not suitable to be employed. Rule 3) in *Step* 1.2 drops the triples whose three image points are too close to each other. This is employed because the image domain is discrete in practice and the three image points of the triples that are too close to each other is easily affected by quantization noises. In our experience, the pose computed by Algorithm 1 can serve as a good initial estimate for the NP*n*P problem. The iteration time, *K*, is selected by balancing between accuracy and time. The larger the *K*, the more the triples are used and, thus, the higher the chance of having a more accurate solution. Too large a *K* leads to a long period of execution time for Algorithm 1, but may also induce a faster convergence speed for Algorithm 2 that will be introduced in Section 4.2. A simulation showing the influence of *K* on the accuracy of the estimated pose is given in Section 5. In addition, note that Algorithm 1 copes with the case in which 3D-2D correspondences are available. By augmenting it with some further processing, this algorithm can be extended to a RANSAC algorithm [9] that can handle outliers of 3D-2D correspondences.

Another issue worthy of being addressed is the computation of rigid transformations associated with the three pairs, $(\boldsymbol{P}_i, l_i)$, $(\boldsymbol{P}_j, l_j)$, and $(\boldsymbol{P}_k, l_k)$, in *Step 1.3*. A

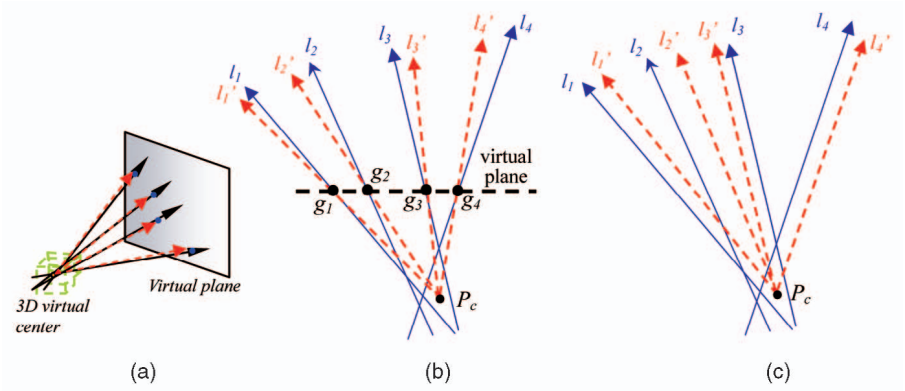

Fig. 4. (a) Approximate a GID with a persepctive camera containing a virtual center and a virtual image plane. (b) A 2D example of the virtual center, virtual plane, and approximated lines. (C) A 2D illustration of the infinite-plane approximation strategy.

generally effective way is to use the method for solving the NP3P problem as introduced in Section 3.1, which obtains rigid transformations that transforms P_i, P_j, and P_k to points lying on l_i, l_j, and l_k, respectively. However, the computational efficiency may be diminished due to the following two reasons. 1) It requires finding all the real-number solutions of several eighth-order polynomial equations which can only be solved numerically. 2) There may be many solutions satisfying a NP3P problem, and all of them need to be further processed in *Step 1.4*.

To increase the efficiency of this algorithm, we suggest using a perspective camera to approximate the GID being considered. As a result, the computations involved in *Step 1.3* are reduced to finding solutions of a P3P problem instead of a NP3P one. The number of real-number solutions of a P3P problem is at most four, which is much smaller than that of the NP3P one. Although the solution obtained by solving a P3P problem is an approximated one compared with its NP3P counterpart, it usually suffices as an initial estimate for a NPnP problem, by considering that the NP3P solution itself serves as an approximation to the NPnP problem.

In the remainder of this section, we will focus on how to approximate a GID with a perspective camera. This problem is equivalent to finding a virtual center and a virtual image plane, as shown in Fig. 4a. First, the virtual center is obtained as a 3D point, P_c, that satisfies the following criterion:

$$\boldsymbol{P}_c = \arg\min_{P} \frac{1}{\boldsymbol{n}} \sum_{\boldsymbol{m}=1}^{\boldsymbol{n}} dist^2(\boldsymbol{P}, l_m), \boldsymbol{P} \in \mathcal{R}^3, \tag{15}$$

where $dist(\boldsymbol{P}, l_m)$ is the distance between $\boldsymbol{P}$ and l_m. Since the objective function to be minimized in (15) is a quadratic polynomial in terms of P_x, P_y, and P_z (where $\boldsymbol{P} = (P_x, P_y, P_z)$), it has a closed-form solution that can be solved via standard LMS analysis.

Now, consider the selections of the virtual image plane. As seen in the 2D illustrative example shown in Fig. 4b, when a virtual plane is selected, the intersection points, $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_n$, of this plane and all the rays of the GID can be computed. The line l'_m passing through $\boldsymbol{P}_c$ and $\boldsymbol{g}_m$ then serves as an approximated line of l_m for all $m = 1, \ldots, n$.

Hence, given a 3D point $\boldsymbol{P}$ lying in l'_m, its distance to l_m is, however, dependent on how far $\boldsymbol{P}$ is away from g_m. The farther that $\boldsymbol{P}$ is away from g_m, the larger $dist(\boldsymbol{P}, l_m)$ becomes. Consequently, the accuracy of such an approximation is distance-dependent. When $\boldsymbol{P}$ is away from the virtual plane (e.g., $\boldsymbol{P}$ is a distant 3D point), the approximation is likely to be very poor. Since no prior knowledge about the locations of the 3D points to be imaged with a GID is given, we propose an *infinite-plane approximation* strategy for approximating the 3D rays of GID by using the rays of a perspective imaging device. In this strategy, the infinite plane is selected as the virtual plane, and line l'_m is thus parallel to line l_m so that it approximates for all $m = 1, \ldots, n$. Hence, the distance between l_m and l'_m is a constant, $dist(\boldsymbol{P}_c, l_m)$. In other words, the distance from any point $\boldsymbol{P}$ on the approximated line l'_m to the original line l_m remains fixed by using this strategy. In contrast, if the virtual plane is selected "not" distant to the virtual center $\boldsymbol{P}_c$, the distance from a point $\boldsymbol{P}$ on the approximated line l'_m to the original line l_m will vary with the distance from $\boldsymbol{P}$ to $\boldsymbol{P}_c$, which then leads to a case in which the accuracies of virtual-plane approximation are not even for the 3D points in space. A 2D illustrative example of the infinite-plane strategy is shown in Fig. 4c. Therefore, the advantage of this approximation strategy is that the accuracy of the approximation is independent of the locations of the 3D points, which allows the 3D points to be treated evenly in the pose-estimation process.

After approximating the GID to be processed with a perspective imaging model as described above, the method for solving the P3P problem as introduced in Section 3.2 can then be used to find the required rigid transformations in *Step 1.3* of Algorithm 1.

4.2 Convergent Iterations for NP*n*P

Given an initial estimate of the rigid transformation between WCS and CCS, we further refine it by minimizing an objective function iteratively. Consider that the projection of a point $\boldsymbol{P}$ onto a ray $\boldsymbol{l} = (\boldsymbol{c}, \boldsymbol{v})$ can be represented as:

$$Proj(\boldsymbol{P}; \boldsymbol{l}) = \boldsymbol{v}\boldsymbol{v}^T(\boldsymbol{P} - \boldsymbol{c}) + \boldsymbol{c}. \tag{16}$$

The orthogonal vector from $\boldsymbol{P}$ to $Proj(\boldsymbol{P}; \boldsymbol{l})$ is thus

$$Proj(\boldsymbol{P}; \boldsymbol{l}) - \boldsymbol{P} = (\boldsymbol{v}\boldsymbol{v}^T - I)(\boldsymbol{P} - \boldsymbol{c}), \tag{17}$$

where I is the 3 by 3 identity matrix. The length of the vector defined in (17) is the distance between $\boldsymbol{P}$ and $\boldsymbol{l}$. The

objective function being minimized in our approach is $E = \min_{R,t} e(R,t)$, where $e(R,t)$ is defined as

$$\begin{aligned} e(R,t) &= \frac{1}{n}\sum_{m=1}^{n} ||(v_m v_m^T - I)(RP_m + t - c_m)||^2 \\ &= \frac{1}{n}\sum_{m=1}^{n} ||RP_m + t - Proj(RP_m + t; l_m)||^2. \end{aligned} \tag{18}$$

To find the optimal solution (R^*, t^*) of (18), we adopt the iterative-closest point (ICP) principle introduced by Besl and McKay [3]. Algorithms based on the ICP principle always converge monotonically to minimum values of a mean-square distance metric. This rate of convergence is more rapid than that of generic nonlinear optimization methods (such as the Gauss-Newton method). Although the ICP principle does not guarantee that the global minimum can always be found, it does suggest that the global minimum (or a very approximate local minimum) can be obtained from a very broad range of initial guesses (our simulation result, as shown in Section 5, also supports this phenomenon). The ICP principle was widely adopted for the registration of partially overlapping range images [5], [28]. In the past, the ICP principle has also been adopted for solving pose estimation problems for the perspective case [32], [6]. Lu et al. [23] have recently proposed a method that is very similar to the ICP principle for solving the PnP problem as well, which has also shown that such a scheme is faster and globally convergent.

An algorithm based on the ICP principle contains the iteration of the following two stages: 1) Find the closest point in the corresponding line for each 3D point. 2) Find a rigid transformation that transforms the 3D points to their closest points in a least-squared-error manner. The iterative refinement method proposed for solving the NPnP problem based on the ICP principle is shown in the following.

Algorithm 2: The same variables defined in Algorithm 1 are used

Step 0. Let (R_0, t_0) be the initial rigid transformation estimated using Algorithm 1. Set $E_{old} \leftarrow \infty$.

Step 1. Compute $P_m^* = R_0 P_m + t_0$ for all $m = 1, \ldots, n$.

Step 2. For each point P_m^*, find its closest point, P'_m, in l_m. That is, $P'_m = Proj(P_m^*; l_m) = vv^T(P_m^* - c) + c$ for all $m = 1, \ldots, n$.

Step 3. Find the rigid transformation that minimizes the sum of squared distances between P_m and $P'_m, m = 1, \ldots, n$. That is, find (R_{new}, t_{new}) that minimizes $E = \frac{1}{n}\sum_{m=1}^{n} ||R_{new}P_m + t_{new} - P'_m||^2$.

Step 4. If $(E_{old} - E)/E_{old}$ is smaller than a predefined positive threshold, then **stop.** Otherwise, set $E_{old} \leftarrow E$, $R_0 \leftarrow R_{new}$, $t_0 \leftarrow t_{new}$, and go to *Step 1*.

In *Step 3* of Algorithm 2, the least-squared-error transformation between two sets of 3D points has closed-form solutions, which can be solved via the quaternion approach [16], the SVD approach [1], the orthonormal-matrices approach [17], or the dual-quaternion approach [31]. In our work, the SVD approach proposed by Arun et al. [1] was adopted because it has been shown to have the best overall accuracy among these methods [21].

In the following, some limitations of our method are formally analyzed. Consider that our method consists of two phases, the initial-estimation phase (Algorithm 1) and the ICP phase (Algorithm 2). The major step of Algorithm 1 is to compute the solutions of the NP3P problem, and that of Algorithm 2 is to find the least-squared-error (LSE) rigid transformation between two 3D point sets. We have shown in Section 3.1 that the parallel-ray and the colinear cases are both singular cases of the NP3P problem. Note that the colinear case is also a singular case for estimating the LSE rigid transformation between two 3D point sets [21]. By joining the singular cases of both phases, we can see that the two singular cases of the NP3P problem remain singular for the NPnP problem. We demonstrate this by giving a formal analysis of the error function (18) as shown below.

First, consider the case that all the rays are parallel to each other. In this case, all the unit vectors v_m in (18) are the same. Without loss of generality, let $v_m = u$ for all m, then (18) becomes

$$\begin{aligned} E &= \min_{R,t} \Sigma_m ||(uu^T - I)(RP_m + t - c_m)||^2 \\ &= \min_{R,t} \Sigma_m ||(uu^T - I)(RP_m + t - c_m) + \sigma uu^T u - \sigma u||^2 \end{aligned}$$

for all $\sigma \in \mathcal{R}$ because u is a unit vector. Hence,

$$\begin{aligned} E &= \min_{R,t} \Sigma_m ||(uu^T - I)(RP_m + t - c_m) + (uu^T - I)\sigma u||^2 \\ &= \min_{R,t} \Sigma_m ||(uu^T - I)(RP_m + (t + \sigma u) - c_m)||^2, \end{aligned}$$

which shows that if (R,t) is a solution, then $(R, t + \sigma u)$ is also a solution for all $\sigma \in \mathcal{R}$.

Second, consider the case where all the 3D points are in a line. In this case, the point P_m can be represented as $Q_0 + \sigma_m w$ for all m, where w is a unit vector along this line, Q_0 is an arbitrary point on this line, and $\sigma_m \in \mathcal{R}$, respectively. Note that any rotation can be represented as rotating an angle θ about some axis. Let R_θ be the rotation of rotating θ around the ray (Q_0, w). Then, R_θ satisfies that $R_\theta(Q_0 + \sigma_m w) = Q_0 + \sigma_m w$, and (29) becomes

$$\begin{aligned} E &= \min_{R,t} \Sigma_m ||(v_m v_m^T - I)(R(Q_0 + \sigma_m w) + t - c_m)||^2 \\ &= \min_{R,t} \Sigma_m ||(v_m v_m^T - I)(RR_\theta(Q_0 + \sigma_m w) + t - c_m)||^2. \end{aligned}$$

Hence, if (R,t) is a solution, then (R_θ, t) is also a solution for all θ in this case. The above analyses show that the pose of the GID cannot be determined exactly with our method in the above two cases.

In our experience, the ICP method performs well for solving the NPnP problem. Global minimum (or a very approximate local minimum) can be found from a broad range of initial estimates, as will be shown in the simulation in Section 5. Nevertheless, ICP does not guarantee finding the global minimum for all cases and, therefore, may at times get stuck at a strong local minimum. To solve this problem, a common strategy is to use multiple initial estimates for restarting and running the ICP algorithm multiple times [7], [27]. Note that our method can be easily modified to implement this idea. In Algorithm 1, since K rigid transformations have been found by solving

K distinct P3P problems, each transformation can serve as an initial estimate to restart the ICP method in Algorithm 2. By doing so, there is more opportunity to seek out a better solution when the ICP method fails to find an accurate one in previous trials.

5 Simulations

Some simulations were performed to evaluate the accuracy of the proposed method for solving the NPnP problem. First, a GID containing 50 rays is defined in 3D space. The end point $\boldsymbol{c}$ of each ray is randomly selected in a disk whose radius is 10 and the directional vector $\boldsymbol{v}$ of this ray is selected randomly in space. After a GID is defined in this way, a 3D point is picked randomly on each ray. Then, these 50 points are translated and rotated via a rigid transformation $\boldsymbol{T}_{sim} = (\boldsymbol{R}_{sim}, \boldsymbol{t}_{sim})$. Our purpose is to estimate the rigid transformation $\boldsymbol{T}_{sim}$, given the pairs of rays and transformed 3D points. In our simulation, $\boldsymbol{R}_{sim}$s are synthesized using Euler angles. Several rotations whose angles are within a range $[0, W_{sim}]$ are selected, and the translations $\boldsymbol{t}_{sim}$s are synthesized randomly. In our simulation, W_{sim} is selected from 20° to 140°, as shown in Fig. 5a. With Algorithm 1 (K is set to 100) and the approximated GID as introduced in Section 4.1, an estimated rigid transformation $\boldsymbol{T}_{sim}$ can be obtained from a selected triple of pairs of point-ray correspondences for each $(\boldsymbol{R}_{sim}, \boldsymbol{t}_{sim})$. By applying the estimated transformation, the 3D-distance error (as defined to be the squared root of (18)) was obtained for each $(\boldsymbol{R}_{sim}, \boldsymbol{t}_{sim})$, and Fig. 5a shows an average of the errors for each range specified by W_{sim}.

Meanwhile, the rigid transformation is obtained by means of only three pairs. It can then be further refined by including more pairs of point-ray correspondences using our approach. By using this transformation as an initial estimation for iterative refinement with Algorithm 2, a new rigid transformation can be obtained from all 50 pairs of point-ray correspondences. Fig. 5a also shows the 3D-distance error (the squared root of E) by applying the newly generated rigid transformation. As can be seen, no matter how large the range of the rotation angles are, our method (the combination of Algorithms 1 and 2) can always converge to the correct solutions that correspond to $E = 0$. This shows that our method can converge to the global minimum within a very broad range of initial guesses.To make a comparison, two widely adopted iterative refinement methods, the Gauss-Newton (G-N) method and the Levenberg-Marquardt (L-M) method,[2] are also used to refine the initial rigid transformation obtained via Algorithm 1, as shown in Fig. 5b. In this simulation, the maximal number of iterations (MaxIter) allowed is set to be 10,000, and the tolerance of the error value for stopping (TolFun) is set as 1e-5, for both G-N and L-M, respectively. Fig. 5b shows the 3D-distance error. As can be seen, our proposed method converges more accurately than the other two. In general, the efficiency of an iterative process highly depends on the number of iterations. Fig. 5c further shows the number of iterations required for G-N, L-M, and our Algorithm 2, respectively. It reveals that our method, based on the ICP principle, is far more efficient than the other two.

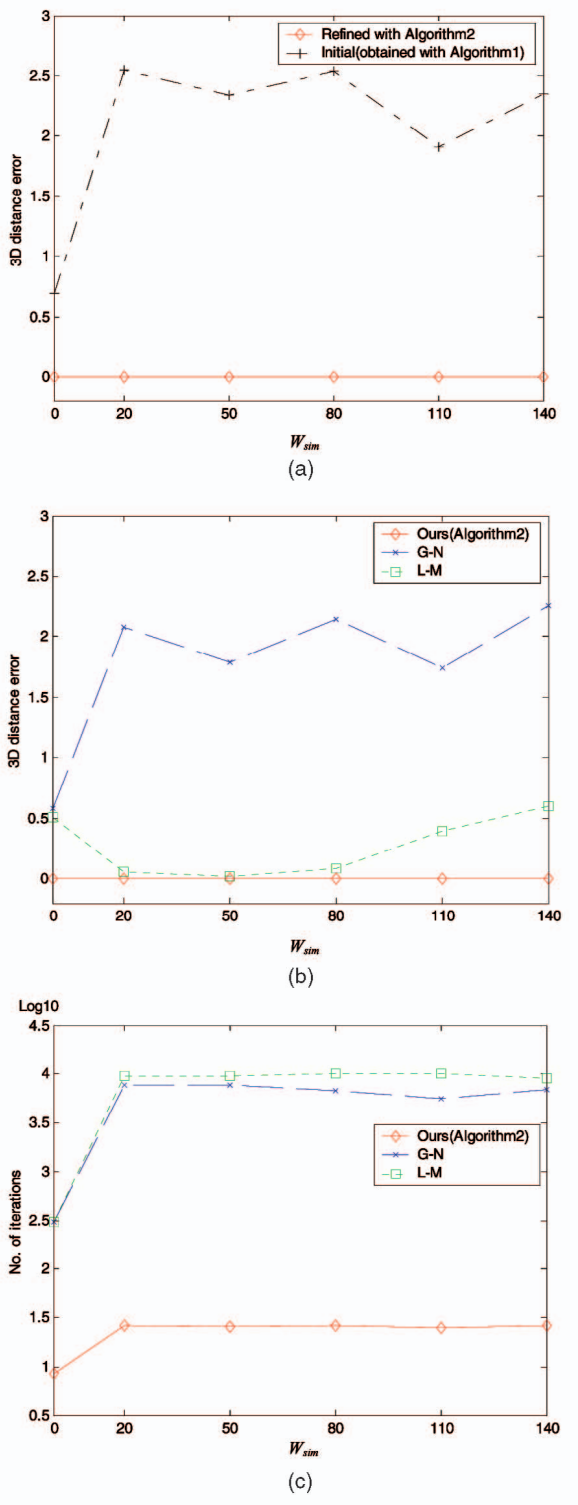

Fig. 5. (a) Average 3D-distance errors obtained using Algorithm 1 and Algorithm 2. (b) Average 3D-distance error obtained using G-N, L-M, and our (Algorithm 2) methods. (c) Number of iterations associated with each method.

We have also varied some other factors in our method to see their influences. First, we show the influences of the

2. They were implemented with MATLAB via the command "lsqnonlin" in our simulation.

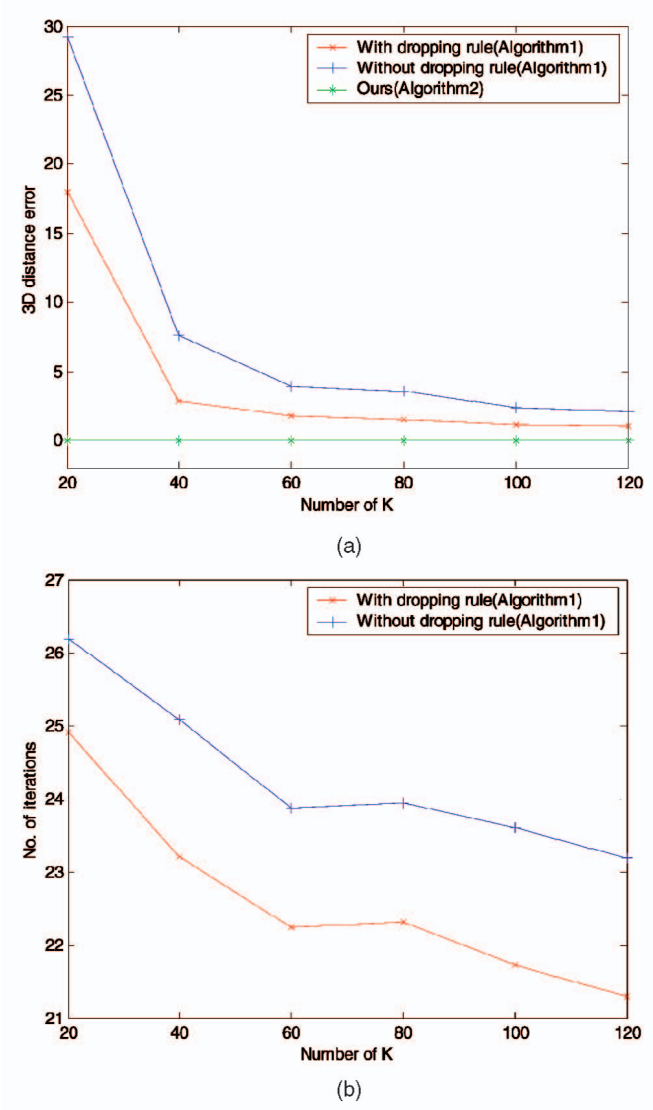

Fig. 6. (a) Average 3D-distance errors obtained by using Algorithm 1 (with and without using dropping rules) and Algorithm 2 in association with K, the number of iterations of Algorithm 1. (b) Number of iterations of Algorithm 2 in association with K.

iterative times, *K*, in Algorithm 1, by varying *K* from 20 to 120 (where W_{sim} is fixed to $50°$). The 3D-distance error of the initial transformation, obtained by using Algorithm 1, is shown as a red line in Fig. 6a. As can be seen, the pose-estimation error decreases when *K* increases, which is very reasonable because the more trials that are being done, the better the chances are to find a better solution. Nevertheless, an interesting result is that, no matter what *K* is, an initial pose can always be iteratively refined via the ICP principle of Algorithm 2 to an exactly correct pose, as shown with the green line in Fig. 6a. Hence, *K* has an influence on the accuracy of the initial poses but is without influences on the final poses when our method is being used in this simulation. Secondly, we investigated the influence of the dropping rules of *Step 1.2* of Algorithm 1. The blue line in Fig. 6a shows the average 3D-distance error of the initial poses obtained by using Algorithm 1, where the dropping rules in *Step 1.2* were not used. It is clear that the dropping rules are useful for finding a better initial estimation of the GID pose. Also, similar to the case of varying the rotation angles and *K*, exactly correct poses can always be found by further refining with Algorithm 2, whether the dropping rules have been applied or not (green line of Fig. 6a). The above simulations reveal that the global minimum (or a very approximate local minimum) can be obtained from a very broad range of initial guesses, by using the ICP principle in Algorithm 2. Although the dropping rules have no significant influences on the final poses obtained with our method in the simulation, they are helpful to reduce the iteration time of Algorithm 2 because better initial estimates have been found with Algorithm 1 by using these rules, as shown in Fig. 6b.

In the above, ideal GID models without imaging noise were considered and, thus, the error could be minimized to zero. In the following simulation, the same GID model as defined in the above was used, and some noises are added to the GID. In this simulation, a 3D point is randomly picked on each ray such that the distance between this 3D point and the end point is within a range [10, 500]. Hence, such a configuration simulates a situation where the width of the GID is about 10 centimeters, and it is used to take images of objects that are within a range of 5 meters away from it. These 50 points are translated and rotated via a rigid transformation $\boldsymbol{T}_{sim} = (\boldsymbol{R}_{sim}, \boldsymbol{t}_{sim})$, where the Euler angles of $\boldsymbol{R}_{sim}$ are selected within $[0°, 50°]$ and $\boldsymbol{t}_{sim}$ is selected randomly. Then, we add some imaging noises to the simulation process. The noise is added by varying the directional vector of each ray within a stereo angle. To determine reasonable noise ranges (i.e., ranges of the stereo angles) for simulation, we assume that the angle of viewing scope of the GID is *W*, and it has roughly 512 pixels in a row. If the imaging error is about 3 pixels, then the ray direction deviates roughly $3W/512$ degrees, which is smaller than $2.2°(\approx 3 \times 360°/512)$ because *W* is at most $360°$. According to the above analysis, the noise of the stereo angle is set to range from $S_{sim} = 0.5$ degrees to 5 degrees (i.e., about 0.7 to 6.8 pixel error if it is the case considered above). Within each noise range $[0, S_{sim}]$, several trials were performed. To save the simulation time and taking practical situations into account, the maximal number of function evaluations allowed (Maxfunevals) for both G-N and L-M are set as 1,000 in this simulation. The average errors versus noise ranges thus obtained are shown in Fig. 7a, and it shows the better performance of our method. In addition, we found that the number of function evaluations required for both G-N and L-M are always 1,000. Comparing the number of function evaluations required for different methods as shown in Fig. 7b, our method also demonstrates a greater efficiency in the presence of image noise. Looking at the influence of the iterative time *K* of Algorithm 1 when image noise is presented, Fig. 7c further shows that the 3D-distance errors obtained by setting $S_{sim} = 1°$. Similarly, the larger is *K*, the more accurate is the estimated pose via triple of points with Algorithm 1. Nevertheless, no matter the size of *K* is, Algorithm 2 can always converge to a pose with the same accuracy from all initial poses, as shown in Fig. 7c.

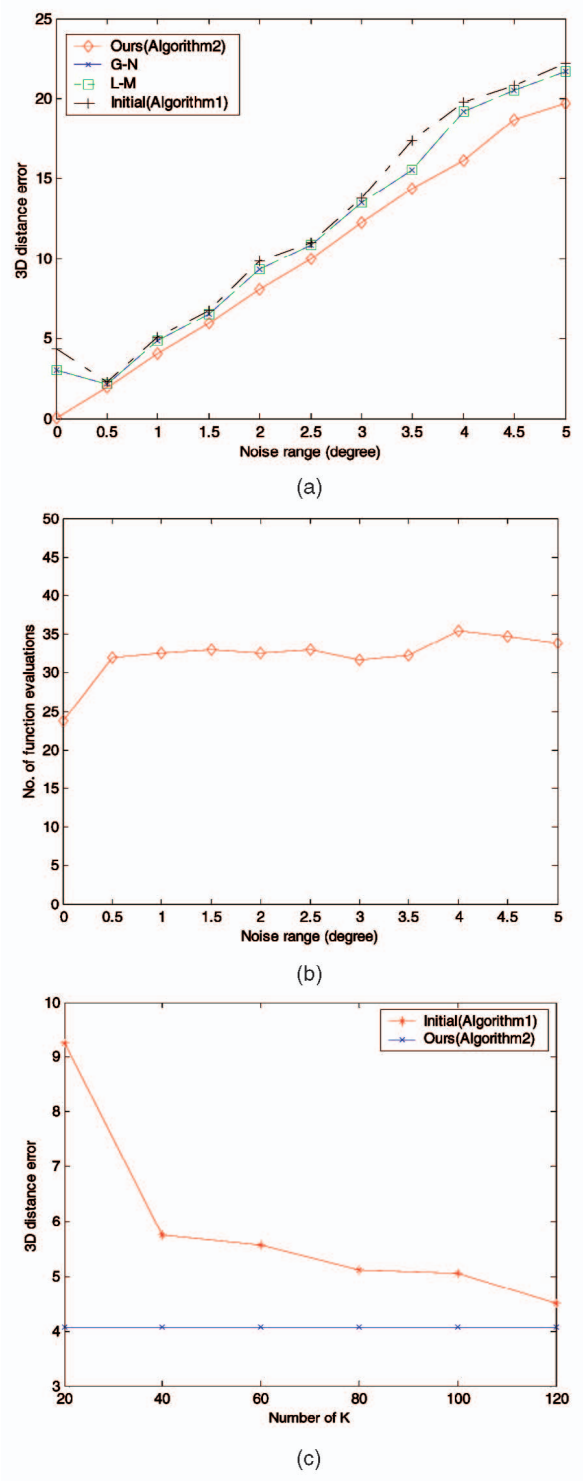

Fig. 7. (a) Average 3D-distance error versus noise range. (b) Number of function evaluations versus noise range using our method. (c) Average 3D-distance errors obtained using Algorithm 1 and Algorithm 2 in association with K when noise is presented ($S_{sim} = 1°$).

6 Experimental Results

An omnidirectional camera, composed of a lens and a curved mirror, is used in our experiment, as shown in Fig. 8. The reflection curve of this camera is designed to maximize the average image resolutions within a range of viewing angles, but not deliberately to satisfy the single view-point constraint [2]. Hence, such an imaging device is a nonperspective GID, but with higher image resolutions and better point-spread properties than those designed to satisfy the single view-point constraint. In the following, an implementation issue of investigating the intrinsic model is first discussed in Section 6.1. Then, the experimental results are shown in Section 6.2.

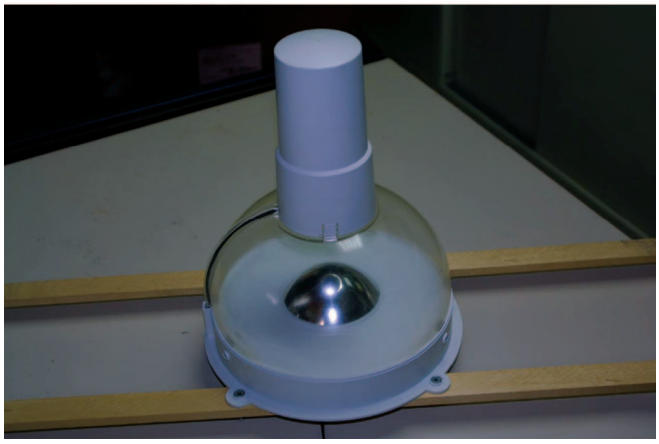

Fig. 8. The omnidirectional camera used in our experiment, which is nonperspective.

6.1 Intrinsic Model Investigation

This paper focuses on the pose-estimation problem of a GID under the situation that the intrinsic model of a GID is given. That is, for all $(i, j) \in D_I$, the corresponding ray, $L(i, j)$, with respect to a CCS of the GID is known. In the following, we will introduce the method used in our experiment for investigating the intrinsic model. The idea behind it is similar to that in [10].

Inspired by an image-based rendering method [20], we use two disjointed surfaces for establishing the intrinsic model $L(\cdot, \cdot)$ in our work. Fig. 9a shows a 2D illustration of our method. Suppose that S_1 and S_2 are two parametric surfaces (e.g., planes or cylinders) with known parameters with respect to a CCS. Hence, each point contained in S_1 or S_2 has known 3D coordinates with respect to the CCS. In addition, assume that each ray of the GID to be investigated intersects some points in S_1 and some other points in S_2, respectively. For example, let $L(i, j)$ intersect S_1 in P_1 and S_2 in P_2, respectively. If we can identify P_1 and P_2 in the image, then $L(i, j)$ can be set as the ray starting from P_1 and passing through P_2. Here, P_1 and P_2 are referred to as the starting and passing point of (i, j), respectively.

In practice, two opaque calibration surfaces, S_1 (inner) and S_2 (outer), are used, both of which contain some calibration marks with known coordinates as shown in Fig. 9b. However, it is worth noting that the two calibration surfaces need not necessary be similar, and that the marks can be randomly located on each surface. The only critical restriction for these two surfaces is that the 3D coordinates of the calibration marks on them have to be known with reference to a unified coordinate system. First, the GID takes an image of S_1. Then, S_1 is taken off and the GID takes an image of S_2. Assume that these two images are I_1 and I_2, respectively. Given an $(i, j) \in D_I$, if it happens to be the

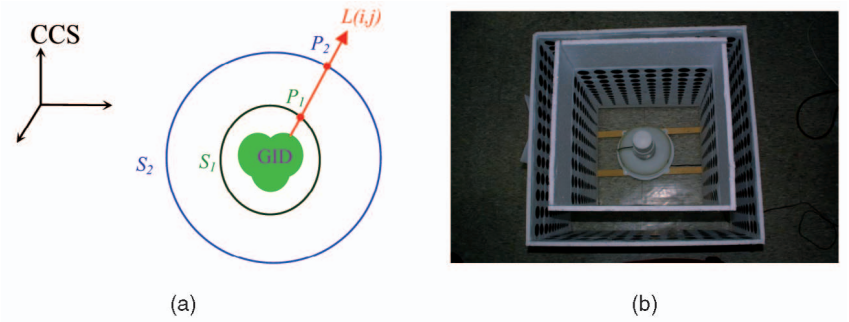

Fig. 9. (a) Two-dimensional illustration of the investigation of the intrinsic model of a GID, where S_1 and S_2 are two surfaces consisting of 3D points of known coordinated with respect to the CCS. (b) The two surfaces, S_1 (inner) and S_2 (outer), used for estimating the intrinsic model.

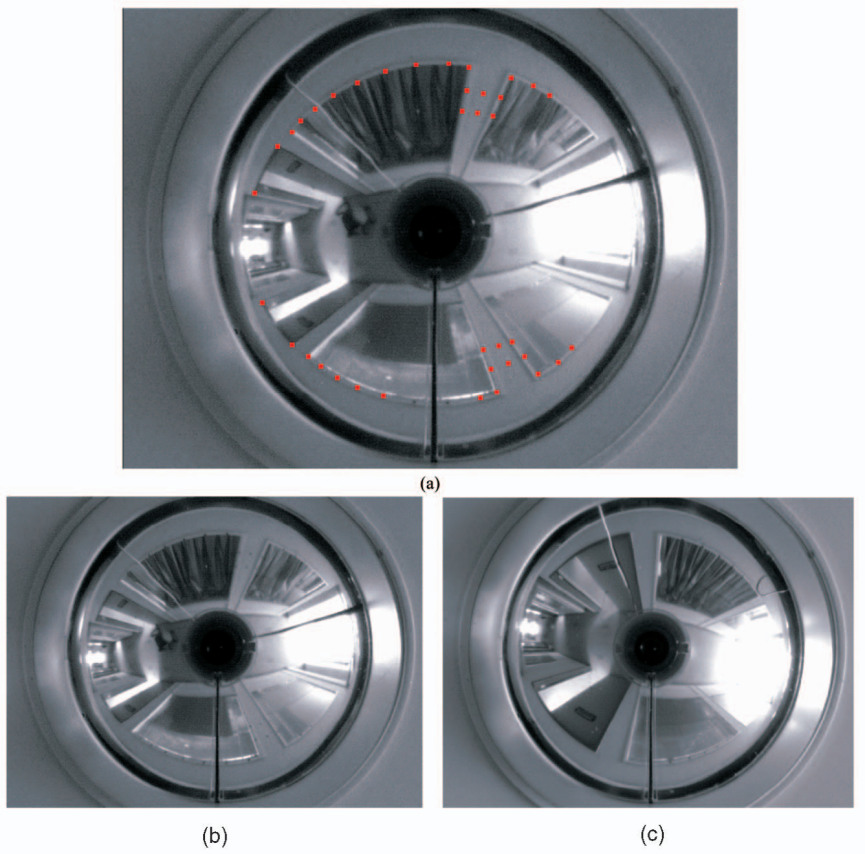

Fig. 10. The images captured with the camera shown in Fig. 8. (a) The red points are the 3D points used for pose estimation. (b) A stereo pair taken with the omnidirectional cameras.

imaging positions of some calibration mark, M, in I_1, as well as some other calibration mark, M', in I_2, then the 3D coordinate of M can serve as the starting point, and that of M' can serve as the passing point, of (i, j), respectively. $L(i, j)$ can then be set as the ray from the starting point toward the passing point. Otherwise, if (i, j) is not the imaging positions of any calibration marks in either I_1 or I_2, then the 3D coordinates of some calibration marks imaged in a neighborhood of (i, j) are used to interpolate the starting and the passing points. The cubic spline [25] is used here to interpolate the starting and the passing points.

6.2 Results

In practice, the quantization error, image correspondence error, and the estimation error of the intrinsic model all can generate errors to the estimated positions and orientations. To verify the accuracy of our method, two such omnidirectional GIDs were used and, thus, a nonperspective stereo pair was formed. We put this stereo setup in an indoor environment, and some 3D points (total of 38 points) in this environment were measured in advance and employed for pose estimation, as shown in Fig. 10. After using the method introduced above, the poses of both imaging devices were estimated. Hence, a calibrated stereo pair of omnidirectional cameras was constructed, which can help us compute the 3D coordinate of any other point in this environment if its corresponding image points have been identified in both images. In this way, we computed the coordinates of some 3D points in this environment and used them to verify the accuracy of the poses estimated with our method. The left part of Table 1 lists the errors measured for

TABLE 1
Errors for Some Right Angles and Length Ratios in Experiment 1

	Real angle	Estimated angle	Relative Error	Real length ratio	Estimated length ratio	Relative Error
Case 1	90	89.9967	0.36*e-4	0.7550	0.7384	2.19*e-2
Case 2	90	90.0017	0.20*e-4	0.8926	0.9005	0.88*e-2
Case 3	90	90.0070	0.78*e-4	0.0624	0.0623	0.16*e-2
Case 4	90	90.0138	1.54*e-4	0.1812	0.1809	0.16*e-2
Case 5	90	90.0164	1.83*e-4	0.0990	0.1043	5.35*e-2

Fig. 11. The illustrations for estimations of some right angles and length ratios in experiment 1. (a) The angles used for right-angle results. (b) The lines used for the length-ratio results, where line 0 is the unit length.

some right angles as shown in Fig. 11a, while the right part of Table 1 lists the errors measured for some length ratios, where line 0 shown in Fig. 11b serves as the unit length. The relative error is the ratio of the difference between real and estimated values to the real value.

In addition, since a stereo pair is formed, the correspondence of a point selected in one image should lie on a curve in the other image, as illustrated in Fig. 12. It is called the epipolar line in the perspective case, and is referred to as the *matching curve* here. Fig. 13a shows some points selected in one image. If no errors occur, their associated matching curves should pass through the corresponding points in the other image. Fig. 13b shows the matching curves of the points shown in Fig. 13a. As can be seen, these matching curves all pass through the corresponding points in a close manner. Table 2 shows the distances from the corresponding points to their associated matching curves, where these corresponding points are selected manually. As can be seen, these distances are all very small (less than one pixel).

In the next experiment, in order to test our method in a more complicated environment, a GID stereo pair is formed such that one GID has a slant motion (where the slant angle is about $10°$) with the other GID, as illustrated in Fig. 14a. In addition, with more objects being added, the scene contains points with more 3D variations than those in the previous experiment, as shown in Figs. 14b and 14c. After computing the GID poses with our method, Table 3 lists distances from some points to their associated matching curves (shown in Fig. 14d). The results of this experiment show again the effectiveness of our method.

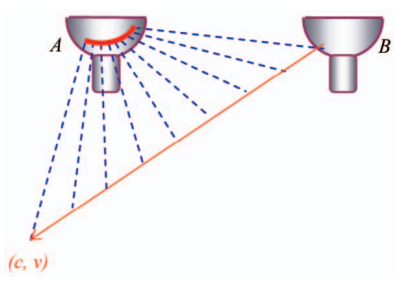

Fig. 12. Illustration of the matching curve.

7 Conclusions and Discussion

A systematic method for pose estimations of generalized imaging devices is proposed in this paper. Since the imaging devices considered in our framework may not be perspective, their pose estimation problem is referred to as the NPnP problem. We have investigated the case when $n = 3$ and presented how to find its exact solutions. Some particularly useful special cases, such as the parallel plane, linear pushbroom, and partially parallel camera models, have also been investigated, and they were shown to have closed-form solutions.

A random-selection strategy, which finds an initial approximation of the rigid transformation via a better triple of 3D-2D correspondences, was then developed for solving the NPnP problem when $n > 3$. To increase the efficiency of the initial-estimation stage, a perspective camera model was also proposed and used for approximating a GID. An iterative refinement algorithm, based on the ICP principle that always converges monotonically to a minimum value, was developed for refining the rigid transformation initially estimated in a highly effective and efficient way. Although a nonperspective imaging device was used in our experiment, the proposed method can be applied not only to nonperspective imaging devices, but to perspective ones as well. The

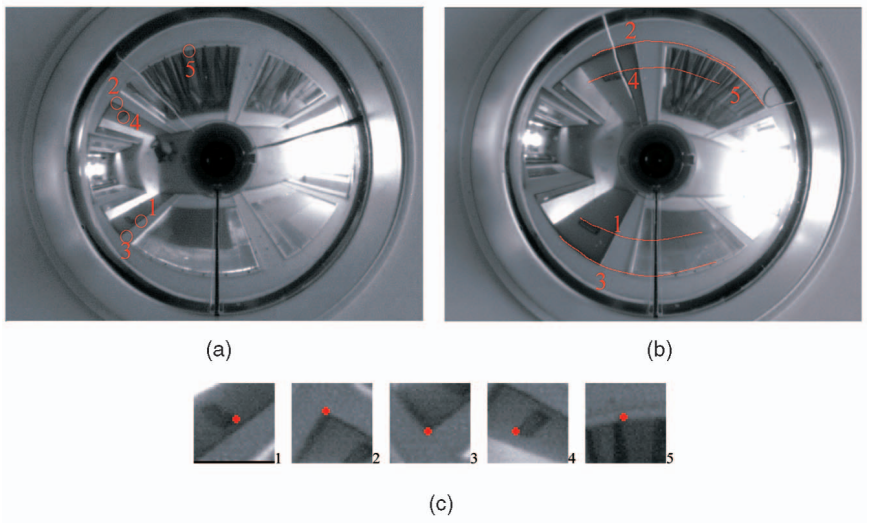

Fig. 13. (a) Five points selected in one image of the stereo GID pair used in experiment 1, where their enlarged local views are shown in (c). (b) The matching curves of these points in the other image of the stereo pair.

TABLE 2
Distances to the Associated Matching Curves in Experiment 1

	Case 1	Case 2	Case 3	Case 4	Case 5
Distance to the matching curve (in pixel)	0.378	0.153	0.378	0.300	0.209

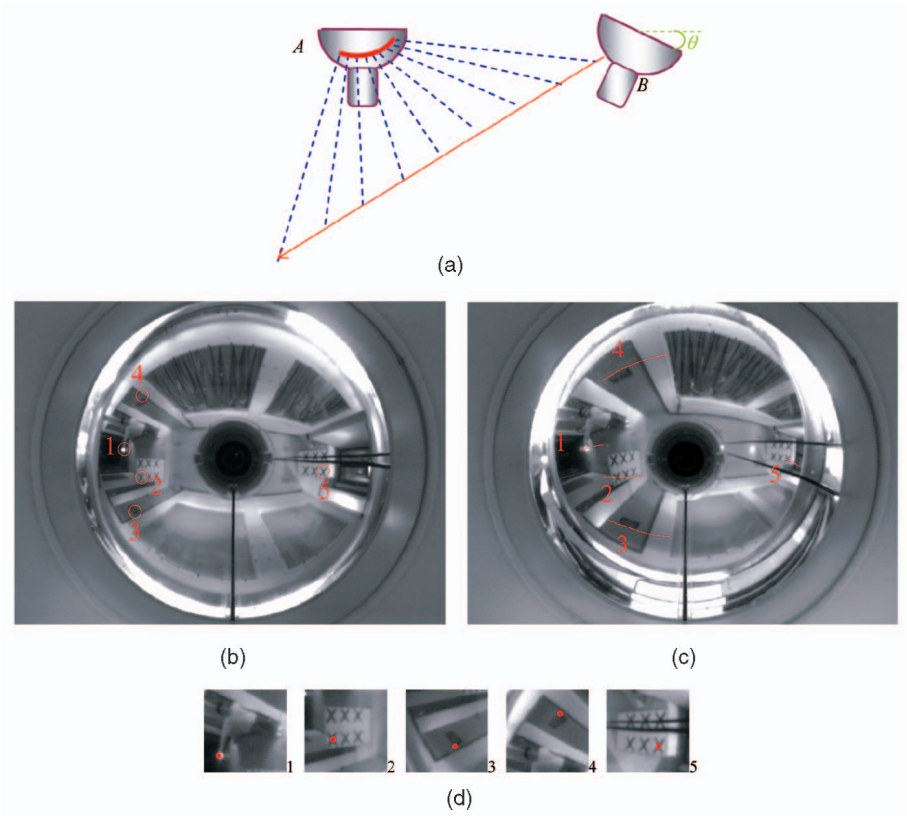

Fig. 14. (a) A slanted stereo GID pair. (b) Five points selected in one image of the stereo GID pair used in experiment 2, where their enlarged local views are shown in (d). (c) The matching curves of these points in the other image of the stereo pair.

TABLE 3
Distances to the Associated Matching Curves in Experiment 2

	Case 1	Case 2	Case 3	Case 4	Case 5
Distance to the matching curve (in pixel)	0.403	0.177	0.410	0.201	0.347

developed method also has potential to be used for pose estimation of a mounted multicamera system by modeling this system with a single GID consisting of all the camera rays. We conclude that our approach provides an effective way for pose estimation of general imaging devices.

REFERENCES

[1] K.S. Arun, T.S. Huang, and S.D. Blostein, "Least-Squares Fitting of Two 3-D Point Sets," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 9, no. 5, pp. 698-700, 1987.

[2] S. Baker and S.K. Nayar, "A Theory of Catadioptric Image Formation," *Proc. IEEE Int'l Conf. Computer Vision,* pp. 35-42, 1998.

[3] P.J. Besl and N.D. McKay, "A Method for Registration of 3D Shapes," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 14, no. 5, pp. 239-256, May 1992.

[4] C.S. Chen and W.Y. Chang, "Pose Estimation for Generalized Imaging Device via Solving Non-Perspective *N* Point Problem," *Proc. IEEE Int'l Conf. Robotics and Automation,* pp. 2931-2937, 2002.

[5] C.S. Chen, Y.P. Hung, and J.B. Cheng, "Ransac-Based Darces: A New Approach to Fast Automatic Registration of Partially Overlapping Range Images," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 21, no. 11, pp. 1229-1234, Nov. 1999.

[6] C.S. Chen et al., "Integrating Virtual Objects into Real Images for Augmented Reality," *Proc. ACM Virtual Reality Software and Technology Conf.,* pp. 1-8, 1998.

[7] D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek, "The Trimmed Iterative Closest Point Algorithm," *Proc. IEEE Int'l Conf. Pattern Recognition,* pp. 545-548, 2002.

[8] D.F. DeMenthon and L.S. Davis, "Model-Based Object Pose in 25 Line of Code," *Int'l J. Computer Vision,* vol. 15, pp. 123-141, 1995.

[9] M.A. Fischler and R.C. Bolls, "Random Sample Consensus: A Paradigm for Model Fitting with Application to Image Analysis and Automated Cartography," *Comm. ACM,* vol. 24, pp. 381-395, 1981.

[10] M.D. Grossberg and S.K. Nayar, "A General Imaging Model and a Method for Finding Its Parameters," *Proc. IEEE Int'l Conf. Computer Vision,* vol. 2, pp. 108-115, July 2001.

[11] F. Huang, S.K. Wei, and R. Klette, "Geometrical Fundamentals of Polycentric Panoramas," *Proc. IEEE Int'l Conf. Computer Vision,* vol. 1, pp 560-565, July 2001.

[12] X.S. Gao et al., "Complete Solution Classification for the Perspective-Three-Point Problem," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 25, no. 8, pp. 930-943, Aug. 2003.

[13] R. Gupta and R. Hartley, "Linear Pushbroom Cameras," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 9, pp. 963-975, Sept. 1997.

[14] R.M. Haralick et al., "Analysis and Solutions of the Three Point Perspective Pose Estimation Problem," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition,* pp. 592-598, 1991.

[15] R. Horaud, B. Canio, and O. Leboullenx, "An Analytic Solution for the Perspective 4-Point Problem," *Computer Vision, Graphics, and Image Understanding,* no. 1, pp. 33-44, 1989.

[16] B.K.P. Horn, "Closed-Form Solution of Absolute Orientation Using Unit Quaternions," *J. Optical Soc. Am., A,* vol. 4, pp. 629-642, 1987.

[17] B.K.P. Horn, H.M. Hilden, and S. Negahdaripour, "Closed-form Solution of Absolute Orientation Using Orthonormal Matrices," *J. Optical Soc. Am., A,* vol. 5, no. 7, pp. 1127-1135, 1988.

[18] M.A. Jenkins, "Algorithm 493: Zeros of a Real Polynomial," *ACM Trans. Math. Software,* vol. 1, p. 178, June 1975.

[19] M. Lang and B.-C. Frenzel, "Polynomial Root Finding," *IEEE Signal Processing Letter,* 1994.

[20] M. Levoy and P. Hanrahan, "Light Field Rendering," *Proc. SIGGRAPH'96,* pp. 31-42, Aug. 1996.

[21] A. Lorusso, D.W. Eggert, and R.B. Fisher, "A Comparison of Four Algorithms for Estimating 3-D Rigid Transformations," *Proc. Sixth British Machine Vision Conf.,* pp. 237-246, 1995.

[22] D.G. Lowe, "Robust Model-Based Motion Tracking through the Integration of Search and Estimation," *Int'l J. Computer Vision,* vol. 8, no. 2, pp. 113-122, 1992.

[23] C.P. Lu, G.D. Hager, and E. Mjolsness, "Fast and Globally Convergent Pose Estimation from Video Images," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 6, pp. 610-622, June 2000.

[24] K. Miyamoto, "Fish Eyes Lens," *J. Optical Soc. of Am.,* vol. 54, pp. 1060-1061, 1994.

[25] W.H. Press et al., *Numerical Recipes in C: The Art of Scientific Computing,* second ed., chapter 3 and chapter 9, Cambridge Univ. Press, 1999.

[26] P. Rademacher and G. Bishop, "Multiple-Center-of-Projection Images," *Proc. SIGGRAPH'98,* pp. 199-206, July 1998.

[27] S. Rusinkiewicz and M. Levoy, "Efficient Variant of the ICP Algorithm," *Proc. Third Int'l Conf. 3D Digital Imaging and Modeling,* pp. 145-152, 2001.

[28] G.C. Sharp, S.W. Lee, and D.K. Wehe, "ICP Registration Using Invariant Features," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 1, pp. 90-102, Jan. 2002.

[29] B.T. Smith et al., *Matrix Eigensystem Routines—EISPACK Guide,* vol. 6, Lecture Notes in Computer Science, Springer, 1976.

[30] R. Swaminathan, M.D. Grossberg, and S.K. Nayar, "Caustics of Catadioptric Cameras," *Proc. IEEE Int'l Conf. Computer Vision,* vol. 2, pp. 2-9, July 2001.

[31] W.M. Walker, L. Shao, and R.A. Volz, "Estimating 3-D Location Parameters Using Dual Number Quaternions," *CVGIP: Image Understanding,* vol. 54, no. 4, pp. 358-367, 1991.

[32] P. Wunsch and G. Hirzinger, "Registration of CAD-Models to Images by Iterative Perspective Matching," *Proc. Int'l Conf. Pattern Recognition,* pp. 78-83, 1996.

[33] Y. Yagi, S. Kaato, and T. Tsuji, "Collision Avoidance Using Omnidirectional Image Sensor (COPIS)," *Proc. IEEE Int'l Conf. Robotics and Automation,* pp. 910-915, Apr. 1991.

[34] K. Yamazawa, Y. Yagi, and M. Yachida, "Obstacle Detection with Omnidirectional Image Sensor: HyperOmni Vision," *Proc. IEEE Int'l Conf. Robotics and Automation,* pp. 1062-1067, 1995.

[35] J.S.C. Yuan, "A General Photogrammetric Method for Determining Object Position and Orientation," *IEEE Trans. Robotics and Automation,* vol. 5, pp. 129-142, 1989.

Chu-Song Chen received the BS degree in control engineering from National Chiao-Tung University, Hsing-Chu, Taiwan, in 1989. He received the MS degree in 1991 and the PhD degree in 1996, respectively, both from the Department of Computer Science and Information Engineering, National Taiwan University. From 1997 to 1999, he has been a postdoctoral fellow of the Institute of Information Science, Academia Sinica, and has become an assitant research fellow since 1999. He has been an adjunct assistant professor in the Department of CSIE, National Taipei University of Technology since 2000. He has also been a joint appointment assistant professor in the Department of Graphic Arts and Communication at the National Taiwan Normal University since 2001. His research interests include pattern recognition, computer vision, signal/image processing, and computer graphics. He is a member of the IEEE and the IEEE Computer Society.

Wen-Yan Chang received the BS degree in computer and information science from Tunghai University, Taichung, Taiwan, in 1998, and the MS degree in computer science and information engineering from National Cheng Kung University, Tainan, Taiwan, in 2000. He is currently a research assistant at the Institute of Information Science, Academia Sinica, Taipei, Taiwan. He is also pursuing a doctorate degree in computer science and information engineering at National Taiwan University. His research interests include image processing, computer graphics, biomedical image analysis, and computer vision.