Pose Estimation from Line Correspondences: A Complete Analysis and a Series of Solutions

Chi Xu, Lilian Zhang, Li Cheng, and Reinhard Koch

Abstract—In this paper we deal with the camera pose estimation problem from a set of 2D/3D line correspondences, which is also known as PnL (Perspective-n-Line) problem. We carry out our study by comparing PnL with the well-studied PnP (Perspective-n-Point) problem, and our contributions are three-fold: (1) We provide a complete 3D configuration analysis for P3L, which includes the well-known P3P problem as well as several existing analyses as special cases. (2) By exploring the similarity between PnL and PnP, we propose a new subset-based PnL approach as well as a series of linear-formulation-based PnL approaches inspired by their PnP counterparts. (3) The proposed linear-formulation-based methods can be easily extended to deal with the line and point features simultaneously.

Index Terms—Perspective-3-Line, perspective-n-line, configuration analysis, camera pose estimation

1 Introduction

In a broad range of applications including computer vision [1], robotics [2] and augmented reality [3], [4], it is crucial to determine the pose of a calibrated camera. This is usually performed by analyzing n correspondences between 3D reference features and their 2D projections, where the features are either points or lines. When the point features are employed, it becomes the well-studied Perspective-n-Point (PnP) problem [5], [6], [7], [8]. Meanwhile for line features, it corresponds to the Perspective-n-Line (PnL) problem, which remains a challenging topic. Recent works such as [9] manifest that line feature carries rich information, thus plays an indispensable role in camera pose estimation. The aim of this paper is dedicated to studying the PnL problem.

When the number of line correspondences is three ($n = 3$), it is known as the Perspective-3-Line (P3L) problem, and its solution plays a fundamental role in dealing with the general PnL problem. Given three 2D/3D line correspondences, we first derive a generic P3L polynomial by parameterizing the camera orientation with respect to the model coordinate frame as two rotation angles. It is known that the order of the general P3L polynomial is 8 [10], higher than that of the 4th order P3P polynomial. To study the complexity of the P3L polynomial, it is necessary to carry out a complete 3D configuration analysis. Our analysis reveals that the order of the P3L polynomial can be reduced to 4, 2 or 1 depending on the 3D line configurations. We also show that a number of existing analyses can be included as addressing special cases of the complete picture. In particular, when three lines lie on a plane and form a triangle, the P3L problem degenerates to a P3P problem and produces a 4th order polynomial.

When $n \geq 4$, there are many open issues: (a) The small line sets are sensitive to noise; (b) The computational complexity to discover the global optimum is expensive; (c) The outliers of 2D/3D line correspondences should be effectively detected and removed. To tackle these issues, a series of new PnL approaches are proposed in this paper, which can be classified into the following two categories:

(i) Subset-Based PnL Approach. An Accurate Subset-based PnL approach (**ASPnL**) is proposed here inspired by RPnP [7]. In this framework, lines are separated into triplets by selecting a rotation axis, then a 16th order cost function is constructed from a set of P3L polynomials, and an optimum solution is retrieved from its local minima. In the presence of an uncentred small set of n lines, the performance of **ASPnL** is significantly better than the state-of-the-art methods which are more prone to local minima. Nonetheless it is very sensitive to outliers. **ASPnL** is extended from our previous work [11].

(ii) A Series of Linear-Formulation-Based Approaches. By exploring the similarity between PnL and PnP in terms of linear formulation, we propose a series of linear-formulation-based PnL approaches (**LPnL**) inspired by the state-of-the-art PnP methods including EPnP [5] and REPPnP [8]. Although the linear-formulation based method is not robust to small line-set, it is efficient and able to deal with large line-set with up to 50 percent of outliers with the help of a built-in outlier removal module. Furthermore, the proposed linear-formulation-based approaches can be easily extended to deal with line and point correspondences simultaneously.

2 Related Work

The problem of estimating camera pose from 2D/3D line correspondences has been addressed for more than two

- *C. Xu and L. Cheng are with Bioinformatics Institute, A*STAR, Singapore 138632, Singapore. E-mail: {xuchi, chengli}@bii.a-star.edu.sg.*
- *L. Zhang is with the Department of Automatic Control, College of Mechatronics and Automation, National University of Defense Technology, Changsha 410073, China. E-mail: lilianzhang@nudt.edu.cn.*
- *R. Koch is with the Institute of Computer Science, University of Kiel, Kiel 24118, Germany. E-mail: rk@mip.informatik.uni-kiel.de.*

Manuscript received 7 Oct. 2015; revised 2 Feb. 2016; accepted 2 June 2016. Date of publication 19 June 2016; date of current version 12 May 2017. Recommended for acceptance by R. Yang.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPAMI.2016.2582162

0162-8828 © 2016 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

decades. There are 6 degrees of freedom for camera pose in 3D space. To get a finite set of camera pose solutions, at least three correspondences should be given because each correspondence offers two dimensions of constraint on the pose parameters [12]. In one of the earliest works, Dhome et al. [13] proposed a closed-form solution for the P3L problem by a polynomial approach. To derive the P3L polynomial, 3D lines were transformed into a model coordinate system and 2D lines were transformed into a virtual image plane in the viewer coordinate system. Later, Chen [10] proposed to derive the P3L polynomial by introducing a canonical configuration. Unfortunately, the description of the overall rotation was unclear because three rotation axes in Eq. (4) of [10] were neither aligned with the camera coordinate frame nor aligned with the world coordinate frame, and some extra rotations were required to align the coordinate systems. For three lines in certain configurations, there exist special geometric properties which can be applied to reduce the complexity of the P3L problem. Caglioti [14] addressed a special case of the P3L problem where three lines lie in a common plane and intersect at a common point, which is called the planar 3-line junction perspective problem. Qin and Zhu [15] proposed a solution of another special situation of the P3L problem, where three lines form a Z-shape in space, i. e., two lines are parallel and the 3rd line intersects with both of them. Recently, Zhang and Koch [16] addressed the problem of estimating vanishing point (i.e., camera orientation) in a Manhattan world where lines should be either parallel or orthogonal to each other. In this work, we will provide a complete analysis of all the special line configurations under a unified framework.

One well-known fact is that the solutions to the P3L problem are not uniquely determined [10]. Thus at least four line correspondences should be established for a unique pose solution. Generally, there are two kinds of approaches (iterative and non-iterative algorithms) to solve the pose estimation problem when a large number of correspondences are given. To optimize the pose solution, geometric errors (such as reprojection error) or algebraic errors should be minimized.

Liu et al. [17] proposed to estimate the camera orientation and then the translation in an iterative manner. Kumar and Hanson [18] proposed an improved iterative algorithm that estimates the camera orientation and translation simultaneously (named as R_and_T). The iterative methods of [19], [20] can estimate the camera pose with either a weak perspective or a para-perspective camera model. These iterative algorithms require an initialization and may converge to a local minimum. Besides, in the absence of a good initialization, the computational cost of the iterative algorithms is generally high.

For non-iterative approaches, the most straight-forward algorithm is the Direct-Linear-Transformation (DLT) algorithm [17], [21] which requires six or more line correspondences as input. An improved algorithm was presented in [22] that works for four or more line correspondences, by employing a lifting approach to convert the polynomial system to a linear system of the elements from the rotation matrix. This algorithm has $O(n^2)$ computational complexity and its performance is very sensitive to image noises. Phong et al. [23] presented a method for optimally estimating the camera pose by representing rotation and translation with a dual number quaternion. Recently, Mirzaei and Roumeliotis [24] presented an algebraic approach to estimate the global optimum of the camera pose. The algorithm is non-iterative and possesses $O(n)$ computational complexity. The rotation is represented by the Cayley-Gibbs-Rodriguez (CGR) parametrization in their work. After relaxing the constraints on the rotation matrix, three cubic equations with three unknowns are generated to form a polynomial system with 27 candidate solutions. In their latter work [25], they showed that the original polynomial system (i e., without relaxation) consists of three 5th order equations and one cubic equation with four unknowns, which yields 40 candidate solutions. In their work, the polynomial system was solved using algebraic geometry techniques and the optimal solution is chosen out of the candidates in a least-square sense. Nevertheless, the polynomial system is computationally very expensive, mainly due to the construction of the 120×120 Macaulay matrix. It also returns too many candidate solutions.

PnL is closely related to the PnP problem. Kneip et al. [26] proposed a novel P3P solution which computes the 3D transformation in one step. Zheng et al. [27] proposed an global optimal O(n) solver using quaternion parametrization whose two-fold symmetry property was exploited. Kneip et al. [28] proposed an closed-form solution for generalized PnP, which unified the properties such as global optimality, linear complexity, completeness and singularity-free. Inspired by the success of the RPnP approach [7] for the PnL problem, the RPnL approach was proposed in [11] which retrieved the optimal solution from a 16th order cost function.

Most of the proposed algorithms are designed to robustly estimate the camera pose when image noise exists. Yet, the problem of dealing with outliers in the 2D/3D line correspondences has not been directly handled. An independent preliminary step based on RANSAC [12] is generally required. The main drawback of this two stage strategy is that the efficiency of the PnL algorithm is not fully exploited. Most recently, Ferraz et al. [8] proposed an efficient PnP solution by integrating the outlier rejection within the pose estimation pipeline. We seek to similarly address the more challenging PnL problem in this paper.

In what follows we start by presenting the P3L problem.

3 The Perspective-3-Line Problem

3.1 The General P3L Polynomial

The coordinate frames are defined as follows: The camera coordinate frame $O^cX^cY^cZ^c$, the world coordinate frame $O^wX^wY^wZ^w$, and the model coordinate frame $O^mX^mY^mZ^m$ (which is an auxiliary coordinate frame sharing its origin with $O^wX^wY^wZ^w$). The rotation of the camera frame with respect to the model frame is denoted as $\mathbf{R}_m^c$, the rotation of the model frame with respect to the world frame is $\mathbf{R}_w^m$, and similarly $\mathbf{R}_w^c$ denotes the rotation of the camera frame with respect to the world frame, as $\mathbf{R}_w^c = \mathbf{R}_m^c\mathbf{R}_w^m$, with all these three rotations being 3×3 real matrices.

Given a calibrated camera and three reference lines L_i $(i \in \{0, 1, 2\})$ with their corresponding 2D projections on the image plane as l_i, the camera pose can be solved by

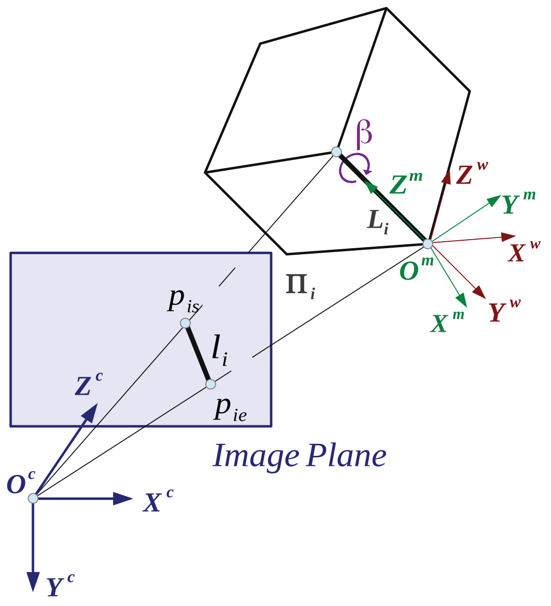

Fig. 1. The geometry of the P3L problem: the 2D/3D line correspondences and the three coordinate frames.

utilizing these known 3D/2D correspondences ($L_i \Leftrightarrow l_i$). The problem consists of two parts: estimating the rotation matrix $\mathbf{R}_w^c$ and the translation vector $\mathbf{t} \in \mathbb{R}^3$. As will become clear later, the rotation matrix $\mathbf{R}_w^c$ is solved by exploiting the geometric constraints between the lines. In general, it involves solving nonlinear equations of 8th order polynomials. The translation $\mathbf{t}$ is then obtained by solving linear equations.

As illustrated in Fig. 1, let $L_i = (\mathbf{v}_i, P_i)$ be a 3D line, where $\mathbf{v}_i \in \mathbb{R}^3$ is the normalized vector giving the direction of the line and $P_i \in \mathbb{R}^3$ is any point on the line. The coordinates of the line vector $\mathbf{v}_i$ in the world, model and camera coordinate frames are denoted as $\mathbf{v}_i^w$, $\mathbf{v}_i^m$ and $\mathbf{v}_i^c$, respectively. Here $\mathbf{v}_i^w$ is known from the input, $\mathbf{v}_i^m = \mathbf{R}_w^m \mathbf{v}_i^w$, and $\mathbf{v}_i^c = \mathbf{R}_m^c \mathbf{v}_i^m$. Similarly, the coordinates of P_i in the world, model and camera coordinate frames are denoted as P_i^w, P_i^m and P_i^c, respectively. P_i^w is known from the input, $P_i^m = \mathbf{R}_w^m P_i^w$, and $P_i^c = \mathbf{R}_m^c P_i^m + \mathbf{t}$. Denote the projection of L_i on the image plane as l_i, in which $\mathbf{p}_{is}$ and $\mathbf{p}_{ie}$ are the endpoints of l_i.[1] For a given l_i, its projection plane Π_i is determined by the projection center O^c, l_i and L_i. The normal of Π_i is denoted as $\mathbf{n}_i$. The coordinate of $\mathbf{n}_i$ in the camera frame is denoted as $\mathbf{n}_i^c$ which can be easily calculated from the input l_i.

By selecting a 3D line L_0, we form the model coordinate frame ($O^m X^m Y^m Z^m$) whose Z^m-axis aligns with $\mathbf{v}_0$ and whose origin is located at the origin of the world frame. The rotation matrix from the world frame to the model frame can be determined as $\mathbf{R}_w^m = [\mathbf{e}_1, \mathbf{e}_2, \mathbf{v}_0^w]$ where $\mathbf{e}_1 \in \mathbb{R}^3$ and $\mathbf{e}_2 \in \mathbb{R}^3$ are one pair of the orthogonal bases of the null space formed by the linear system $\mathbf{v}_0^{wT}\mathbf{e}_i = 0$ $(i \in \{1, 2\})$.

Considering the constraint that the line L_0 lies on the projection plane Π_0, the unknown rotation matrix $\mathbf{R}_m^c$ can be parameterized as

1. The endpoints $\mathbf{p}_{is}$ and $\mathbf{p}_{ie}$ can be chosen as any two points on line l_i.

$$\mathbf{R}_m^c = \mathbf{R}' \, Rot(X, \alpha) Rot(Z, \beta), \tag{1}$$

in which $\mathbf{R}' = \begin{bmatrix} r'_{11} & r'_{12} & r'_{13} \\ r'_{21} & r'_{22} & r'_{23} \\ r'_{31} & r'_{32} & r'_{33} \end{bmatrix}$ is an arbitrary orthogonal rotation matrix whose first column $[r'_{11}, r'_{21}, r'_{31}]^T$ equals to $\mathbf{n}_0^c$, $Rot(X, \alpha)$ denotes a rotation around the X-axis, and $Rot(Z, \beta)$ denotes a rotation around the Z-axis. Hence, $\mathbf{R}_m^c$ can be determined by the two rotation angles α and β.

By using the geometric constraint that $\mathbf{v}_i^c$ $(i = 1, 2)$ should be perpendicular to the normal $\mathbf{n}_i^c$ of the plane Π_i, we have the following two constraints:

$$\begin{cases} \mathbf{n}_1^{cT}\mathbf{v}_1^c = \mathbf{n}_1^{cT}\mathbf{R}_m^c\mathbf{v}_1^m = 0 \\ \mathbf{n}_2^{cT}\mathbf{v}_2^c = \mathbf{n}_2^{cT}\mathbf{R}_m^c\mathbf{v}_2^m = 0. \end{cases} \tag{2}$$

Letting $\mathbf{n}_1^{cT}\mathbf{R}' = [n'_{x1}, n'_{y1}, n'_{z1}]$, $\mathbf{n}_2^{cT}\mathbf{R}' = [n'_{x2}, n'_{y2}, n'_{z2}]$, $\mathbf{v}_1^m = [v_{x1}, v_{y1}, v_{z1}]^T$ and $\mathbf{v}_2^m = [v_{x2}, v_{y2}, v_{z2}]^T$, and substituting Eq. (1) into Eq. (2), we have

$$\begin{cases} \sigma_1 \cos\beta + \sigma_2 \sin\beta + \sigma_3 = 0 \\ \sigma_4 \cos\beta + \sigma_5 \sin\beta + \sigma_6 = 0, \end{cases} \tag{3}$$

where

$$\begin{cases} \sigma_1 = v_{y1}\, n'_{y1} \cos\alpha + v_{y1}\, n'_{z1} \sin\alpha + v_{x1}\, n'_{x1} \\ \sigma_2 = v_{x1}\, n'_{y1} \cos\alpha + v_{x1}\, n'_{z1} \sin\alpha - v_{y1}\, n'_{x1} \\ \sigma_3 = v_{z1}\, n'_{z1} \cos\alpha - \ v_{z1}\, n'_{y1} \sin\alpha \\ \sigma_4 = v_{y2}\, n'_{y2} \cos\alpha + v_{y2}\, n'_{z2} \sin\alpha + v_{x2}\, n'_{x2} \\ \sigma_5 = v_{x2}\, n'_{y2} \cos\alpha + v_{x2}\, n'_{z2} \sin\alpha - v_{y2}\, n'_{x2} \\ \sigma_6 = v_{z2}\, n'_{z2} \cos\alpha - \ v_{z2}\, n'_{y2} \sin\alpha. \end{cases}$$

By solving Eq. (3), we have

$$\cos\beta = \frac{\sigma_2\sigma_6 - \sigma_3\sigma_5}{\sigma_1\sigma_5 - \sigma_2\sigma_4}, \quad \sin\beta = \frac{\sigma_3\sigma_4 - \sigma_1\sigma_6}{\sigma_1\sigma_5 - \sigma_2\sigma_4}. \tag{4}$$

By substituting Eq. (4) into $\cos^2\beta + \sin^2\beta = 1$, we have

$$(\sigma_2\sigma_6 - \sigma_3\sigma_5)^2 + (\sigma_3\sigma_4 - \sigma_1\sigma_6)^2 = (\sigma_1\sigma_5 - \sigma_2\sigma_4)^2. \tag{5}$$

By substituting $\sin^2\alpha = 1 - \cos^2\alpha$ into Eq. (5) and rearranging the terms, we have

$$\sum_{k=0}^{4} u_k \cos^k\alpha = \sin\alpha \sum_{k=0}^{3} v_k \cos^k\alpha, \tag{6}$$

where $u_k \in \mathbb{R}$ and $v_k \in \mathbb{R}$ are coefficients which can be computed from n'_{xi}, n'_{yi}, n'_{zi}, v_{xi}, v_{yi} and v_{zi} for $i = 1, 2$. Taking the squares of both sides of Eq. (6) and letting $x = \cos\alpha$, an 8th order polynomial can be constructed as:

$$f(x) = \sum_{k=0}^{8} \delta_k x^k = 0, \tag{7}$$

where $\delta_k \in \mathbb{R}$ can be computed from u_k and v_k:

$$\begin{cases} \delta_0 = u_0^2 - v_0^2 \\ \delta_1 = 2(u_0u_1 - v_0v_1) \\ \delta_2 = u_1^2 + 2u_0u_2 + v_0^2 - v_1^2 - 2v_0v_2 \\ \delta_3 = 2(u_0u_3 + u_1u_2 + v_0v_1 - v_1v_2 - v_0v_3) \\ \delta_4 = u_2^2 + 2u_0u_4 + 2u_1u_3 + v_1^2 + 2v_0v_2 - v_2^2 - 2v_1v_3 \\ \delta_5 = 2(u_1u_4 + u_2u_3 + v_1v_2 + v_0v_3 - v_2v_3) \\ \delta_6 = u_3^2 + 2u_2u_4 + v_2^2 - v_3^2 + 2v_1v_3 \\ \delta_7 = 2(u_3u_4 + v_2v_3) \\ \delta_8 = u_4^2 + v_3^2. \end{cases} \tag{8}$$

Eq. (7) is called the P3L polynomial. Although for lines in some special configurations (such as orthogonal, parallel or intersection), a lower order of polynomial may be derived as discussed in the sequel (Section 3.5), the P3L polynomial will not be lower than 8th order for three spatial lines in general configurations. It is worth to point out that we decompose the overall rotation from the world frame to the camera frame into a sequence of simple rotations by an approach different from [10], [13]. Our decomposition is simpler than theirs despite the fact that the same constraints are enforced: the orthogonal constraint that the plane normal is orthogonal to the line direction, and the triangular constraint that the square sum of sine and cosine equals to 1.

3.2 Solving Translation Vector

After solving the rotation matrix, the translation vector $\mathbf{t}$ can be solved using the constraint that P_i^c should be perpendicular to $\mathbf{n}_i^c$, we have

$$\mathbf{n}_i^{cT}(\mathbf{R}_m^c P_i^m + \mathbf{t}) = 0 \ (i = 0, 1, 2),$$

based on which we have

$$\mathbf{N}\,\mathbf{t} = -\begin{bmatrix} \mathbf{n}_0^{cT}\mathbf{R}_m^c P_0^m \\ \mathbf{n}_1^{cT}\mathbf{R}_m^c P_1^m \\ \mathbf{n}_2^{cT}\mathbf{R}_m^c P_2^m \end{bmatrix}, \tag{9}$$

where $\mathbf{N} = [\mathbf{n}_0^c \ \mathbf{n}_1^c \ \mathbf{n}_2^c]$. In general cases, $Rank(\mathbf{N}) = 3$, the translation vector can be solved from Eq. (9). In a specific case that the three lines pass through a joint point in 3D space, or the projections of the lines form a junction, $\det(\mathbf{N}) = 0$ and $\mathbf{t}$ has infinite amount of candidate solutions.

In the P3L problem, the rotation and translation are solved sequentially. Note that the rotation can be estimated even when the translation cannot be solved. Besides, the Euclidean transformation defined by $(\mathbf{R}_w^c, \mathbf{t})$ may transform the scene in front of the camera or behind it. Only the candidate solutions which transform the scene in front of the camera are of interest.

3.3 Comparing the P3P and P3L Problems

The well-known P3P problem is equivalent to a specific P3L sub-case that lines lie on a plane and form a triangle (refer to Section 3.5, case A.2.c). As can be seen in Fig. 2, three lines L_i $(i \in \{0, 1, 2\})$ lie on a plane, and their intersections are P_{01}, P_{12} and P_{02}. The intersections of l_i are p_{01}, p_{12} and p_{02}. It

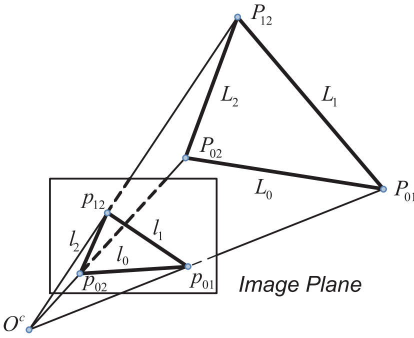

Fig. 2. Geometry of the P3P problem.

can be easily proven that the feasible solutions of $P3L(L_i, l_i)$ satisfy the constraints of $P3P(P_{ij}, p_{ij})$, and vice versa.

It is known that the maximum number of candidate solutions of P3L is 8, while that of P3P is 4. In order to discuss P3P and P3L in the same framework, let's consider the P3P equation system first [29]:

$$\begin{cases} t_0^2 + t_1^2 - t_0t_1\cos\alpha_{01} = d_{01}^2 \\ t_1^2 + t_2^2 - t_1t_2\cos\alpha_{12} = d_{12}^2 \\ t_2^2 + t_0^2 - t_2t_0\cos\alpha_{20} = d_{20}^2, \end{cases} \tag{10}$$

in which t_i denotes the distance of point $P_i (i = 0, 1, 2)$ to camera centre, d_{ij} denotes the length of edge $\overline{P_iP_j}$, and α_{ij} denotes the angle $\angle P_iO^cP_j$. According to Bezout Theorem [30], the P3P equation system has at most eight candidate solutions. It is noteworthy that the candidate solutions of P3P are symmetric. For any $\mathbf{t} = [t_0, t_1, t_2]^T$ being the solution of Eq. (10), $\mathbf{t} = -\mathbf{t}$ is also a solution. By utilizing the symmetric property, the parameter space of P3P can be divided into two parts ($t_0 > 0$ and $t_0 < 0$), and the complexity of Eq. (10) can be easily reduced to form a 4th order polynomial [31].

Inspired by the symmetric property of P3P, let's consider a question: *Can we find a symmetric structure in the P3L parameter space to reduce the order of the P3L polynomial?* The symmetric property of P3L is closely related to the parameterization of the solution and the 3D configuration of lines, which is discussed in what follows.

3.4 Symmetric Property of the P3L Problem

To investigate the symmetric property of the P3L problem, the solution of P3L can be parameterized into two different forms:

(1) *Rotation matrix* $\mathbf{R}_w^c$: Parameterize the rotation matrix. For example, $\mathbf{R}_w^c$ can be parameterized as Euler angles, Quaternion or CGR parameter [24], [32]. In our solution for the general P3L polynomial, two rotation angles α and β are used to parameterize $\mathbf{R}_w^c$.

(2) *Direction of the lines* $\mathbf{V}^c$: Parameterize the line direction vectors $\mathbf{V}^c = [\mathbf{v}_0^c, \mathbf{v}_1^c, \mathbf{v}_2^c]$ in the camera frame.

$\mathbf{R}_w^c$ is the final solution of P3L, while $\mathbf{V}^c$ is the intermediate solution which contributes to understand the geometry of the solution intuitively. Denoting the line direction vectors in the world frame as $\mathbf{V}^w = [\mathbf{v}_0^w, \mathbf{v}_1^w, \mathbf{v}_2^w]$, according to the euclidean transform, we have $\mathbf{V}^c = \mathbf{R}_w^c\mathbf{V}^w$. Since the direction of a line is symmetric, i. e., $(\mathbf{v}_i, P_i)$ and $(-\mathbf{v}_i, P_i)$

represent the same line, we have

$$\mathbf{V}^c\mathbf{M} = \mathbf{\Omega}\mathbf{V}^w,$$

where $\mathbf{\Omega}$ is a possible rotation solution, and $\mathbf{M}$ is the symmetric kernel which has eight possible combinations:

$$\mathbf{M} \in \left\{ \begin{array}{c} diag[\ 1,\ 1,\ 1],\ diag[\ 1,-1,-1], \\ diag[-1,-1,\ 1],\ diag[-1,\ 1,-1], \\ diag[-1,\ 1,\ 1],\ diag[\ 1,\ 1,-1], \\ diag[1,-1,\ 1],\ diag[-1,-1,-1] \end{array} \right\}. \tag{11}$$

Here, $diag[m_1, m_2, m_3]$ means a diagonal matrix. Assuming that $\mathbf{V}^w$ is a full rank matrix, the candidate solution for $\mathbf{R}_w^c$ is

$$\mathbf{\Omega} = \mathbf{V}^c\mathbf{M}(\mathbf{V}^w)^{-1}. \tag{12}$$

In different line configurations, only a subset of matrices in Eq. (11) ensures that $\mathbf{\Omega}$ is a rotation matrix in $SO(3)$ [33], i.e.,

$$SO(3) = \{\mathbf{\Omega} \in \Re^{3\times 3} | \mathbf{\Omega}^T\mathbf{\Omega} = \mathbf{I}, \det(\mathbf{\Omega}) = 1\}.$$

From Eq. (12), we have

$$\det(\mathbf{\Omega}) = \det(\mathbf{V}^c\mathbf{M}(\mathbf{V}^w)^{-1}) = \det(\mathbf{M})\det(\mathbf{R}_w^c).$$

Since the determinant of a rotation matrix equals 1, we have $\det(\mathbf{M}) = 1$. Therefore, the number of possible $\mathbf{M}$ is reduced to 4:

$$\mathbf{M} \in \left\{ \begin{array}{c} diag[\ 1,\ 1,\ 1],\ diag[1,-1,-1], \\ diag[-1,-1,1],\ diag[-1,1,-1] \end{array} \right\}. \tag{13}$$

The 3D configuration of $\mathbf{V}^w$ is closely related to the symmetric kernels $\mathbf{M}$ in Eq. (12). Our primary results are as follows, and the proofs of following lemmas can be found in Section I of the supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2016.2582162:

Lemma 1. *When $\mathbf{v}_0^w \perp \mathbf{v}_1^w$, $\mathbf{v}_0^w \perp \mathbf{v}_2^w$ and $\mathbf{v}_1^w \perp \mathbf{v}_2^w$, the symmetric kernels in Eq. (12) are exactly the four possible candidates given in Eq. (13).*

Lemma 2. *When $rank(\mathbf{V}^w) = 3$, $\mathbf{v}_0^w \perp \mathbf{v}_1^w$, $\mathbf{v}_0^w \perp \mathbf{v}_2^w$ and $\mathbf{v}_1^w \not\perp \mathbf{v}_2^w$, the symmetric kernel in Eq. (12) belongs to a further reduced set $\mathbf{M} \in \{diag[1,1,1],\ diag[1,-1,-1]\}$.*

Lemma 3. *When $rank(\mathbf{V}^w) = 3$, $\mathbf{v}_0^w \not\perp \mathbf{v}_2^w$, $\mathbf{v}_1^w \not\perp \mathbf{v}_2^w$, the symmetric kernel in Eq. (12) can only be $\mathbf{M} = diag[1,1,1]$.*

Lemma 4. *When $rank(\mathbf{V}^w) = 2$ and $\mathbf{v}_0^w \nparallel \mathbf{v}_1^w \nparallel \mathbf{v}_2^w$, two full rank matrices can be constructed as:*

$$\hat{\mathbf{V}}^c = [\mathbf{v}_0^c \times \mathbf{v}_1^c,\ \mathbf{v}_0^c,\ \mathbf{v}_1^c],\quad \hat{\mathbf{V}}^w = [\mathbf{v}_0^w \times \mathbf{v}_1^w,\ \mathbf{v}_0^w,\ \mathbf{v}_1^w].$$

The rotation solution is $\mathbf{R}_w^c = \hat{\mathbf{V}}^c\mathbf{M}(\hat{\mathbf{V}}^w)^{-1}$ with symmetric kernel $\mathbf{M} \in \{diag[1,1,1],\ diag[1,-1,-1]\}$.

3.5 A Complete 3D Configuration Analysis

The complexity of the P3L polynomial and the geometrical attribute of the P3L solution are closely related to the 3D configuration of lines. In this section, we will provide a complete 3D configuration analysis from three aspects:

(1) Junction: The complexity of the P3L polynomial is closely related to the condition that whether the three lines pass through a spatial joint point, or the projections of the lines form a junction.

(2) $Rank(\mathbf{V}^w)$: The rank of $\mathbf{V}^w = [\mathbf{v}_0^w, \mathbf{v}_1^w, \mathbf{v}_2^w]$ has a direct impact on the number of P3L candidate solutions. When $Rank(\mathbf{V}^w) = 1$, all three lines are parallel in space and the P3L problem has infinite solutions. Therefore, we consider only two valid configurations: (i) $Rank(\mathbf{V}^w) = 3$ and (ii) $Rank(\mathbf{V}^w) = 2$.

(3) Spatial relationship: The spatial relationships (orthogonal or parallel) between the 3D lines directly affect the symmetric property of the P3L solution space.

In what follows, we first divide the 3D configuration space of P3L into two classes according to whether the three lines form a junction. Each class is further split into two subspaces based on $Rank(\mathbf{V}^w)$. Each subspace is then partitioned into several cases, where each case addresses a distinct scenario hinged on the relationship among the 3D lines. For each case, we analyze the complexity of the P3L polynomial and the maximum number of candidate solutions of $\mathbf{R}_w^c$ by parameterizing the rotation matrix. In the following classification, the line direction vectors are expressed as $\mathbf{v}_i$, $\mathbf{v}_j$ or $\mathbf{v}_k$, where $i, j, k \in \{0, 1, 2\}$ are indexes of the lines.

To emphasize the main results of the configuration analysis, in Table 1 we first summarize the maximum number of candidate solutions of the rotation matrix $\mathbf{R}_w^c$ and the order of the simplified P3L polynomial. Detailed derivation can be found in the rest of this section as well as in Sections II & III of the supplementary material, available online.

Class A. No Junction. In this class, three lines do not form a junction in space or on the projection plane. Here the rotation matrix and the translation vector can be solved. The cases discussed in class A are listed in the first part of Table 1. To illustrate the main strategy employed in our analysis, the solution of $\mathbf{R}_w^c$ for case A.1.a is presented below.

The solution of $\mathbf{R}_w^c$: Since three lines are orthogonal to each other in case A.1.a, we can form a model coordinate frame in which $\mathbf{v}_0^m$ aligns to Z^m-axis, $\mathbf{v}_1^m$ aligns to X^m-axis and $\mathbf{v}_2^m$ aligns to Y^m-axis. We have $\mathbf{v}_1^m = [1,0,0]^T$ and $\mathbf{v}_2^m = [0,1,0]^T$. The constraint equation (3) can be simplified to

$$\begin{cases} \sigma_1\cos\beta + \sigma_2\sin\beta = 0 \\ \sigma_4\cos\beta + \sigma_5\sin\beta = 0, \end{cases} \tag{14}$$

with

$$\begin{cases} \sigma_1 = n'_{x1} \\ \sigma_2 = n'_{y1}\cos\alpha + n'_{z1}\sin\alpha \\ \sigma_4 = n'_{y2}\cos\alpha + n'_{z2}\sin\alpha \\ \sigma_5 = -n'_{x2}. \end{cases}$$

By solving Eq. (14), we have

$$\sigma_1\sigma_5 - \sigma_2\sigma_4 = 0. \tag{15}$$

TABLE 1
Summary of the 3D Configuration Analysis

Classes	Configurations	Case ID	Spatial Relationship	#P3L	$\#\mathbf{R}_w^c$
No Junction	$Rank(\mathbf{V}^w) = 3$	A.1.a	$\mathbf{v}_i^w \perp \mathbf{v}_j^w, \mathbf{v}_i^w \perp \mathbf{v}_k^w, \mathbf{v}_j^w \perp \mathbf{v}_k^w$	2	8
		A.1.b	$\mathbf{v}_i^w \perp \mathbf{v}_k^w, \mathbf{v}_j^w \perp \mathbf{v}_k^w, \mathbf{v}_i^w \not\perp \mathbf{v}_j^w$	4	8
		A.1.c	$\mathbf{v}_i^w \not\perp \mathbf{v}_k^w, \mathbf{v}_j^w \not\perp \mathbf{v}_k^w$	8	8
	$Rank(\mathbf{V}^w) = 2$	A.2.a	$\mathbf{v}_i^w \parallel \mathbf{v}_j^w, \mathbf{v}_i^w \perp \mathbf{v}_k^w$	1	4
		A.2.b	$\mathbf{v}_i^w \parallel \mathbf{v}_j^w, \mathbf{v}_i^w \not\perp \mathbf{v}_k^w$	2	4
		A.2.c	$\mathbf{v}_i^w \nparallel \mathbf{v}_j^w \nparallel \mathbf{v}_k^w$	4	8
Junction	$Rank(\mathbf{V}^w) = 3$	B.1.a	$\mathbf{v}_i^w \perp \mathbf{v}_j^w, \mathbf{v}_i^w \perp \mathbf{v}_k^w, \mathbf{v}_j^w \perp \mathbf{v}_k^w$	1	8
		B.1.b	$\mathbf{v}_i^w \perp \mathbf{v}_k^w, \mathbf{v}_j^w \perp \mathbf{v}_k^w, \mathbf{v}_i^w \not\perp \mathbf{v}_j^w$	2	8
		B.1.c	$\mathbf{v}_i^w \not\perp \mathbf{v}_k^w, \mathbf{v}_j^w \not\perp \mathbf{v}_k^w$	4	8
	$Rank(\mathbf{V}^w) = 2$	B.2.a	$\mathbf{v}_i^w \parallel \mathbf{v}_j^w, \mathbf{v}_i^w \perp \mathbf{v}_k^w$	1	4
		B.2.b	$\mathbf{v}_i^w \parallel \mathbf{v}_j^w, \mathbf{v}_i^w \not\perp \mathbf{v}_k^w$	1	4
		B.2.c	$\mathbf{v}_i^w \nparallel \mathbf{v}_j^w \nparallel \mathbf{v}_k^w$	2	8

"#P3L" denotes the order of the simplified P3L polynomial, "$\#\mathbf{R}_w^c$" denotes the maximum number of candidate solutions of $\mathbf{R}_w^c$. The indexes of lines are $i, j, k \in \{0, 1, 2\}$.

By substituting $\sin^2\alpha = 1 - \cos^2\alpha$ into it, we have

$$u_2 \cos^2\alpha + u_0 = v_1 \cos\alpha \sin\alpha.$$

Taking the squares of both sides and defining $x = \cos^2\alpha$, a 2nd order polynomial can be constructed as:

$$f(x) = \delta_2 x^2 + \delta_1 x + \delta_0 = 0.$$

There exist at most two feasible solutions for $\cos^2\alpha$ and at most four feasible solutions for $\cos\alpha$. For each candidate solution of α, there exist two symmetric solutions for β as Eq. (14) is homogeneous. Therefore, $\mathbf{R}_w^c$ has at most $2 \times 2 \times 2 = 8$ feasible solutions.

The P3L solution space for the other cases in class A can be analyzed similarly following the same strategy as above. Detailed analysis of these cases can be found in Section II of the supplementary material, available online.

Class B: Junction. In this class, three lines all pass through a spatial joint point, or the projections of the lines meet at a junction. The rotation matrix can be solved, however the translation vector cannot be determined.

The parameterization of $\mathbf{R}_w^c$ for the junction class is similar to that of the non-junction class. The difference is that here we introduce a virtual camera C' (similar to the virtual camera in [34]) to simplify the complexity of the P3L polynomial. Let p_{joint}^c denote the junction of the three projected lines on the image plane. Now we create a virtual camera frame C' whose image plane is perpendicular to the ray from the camera center O^c to the joint point p_{joint}, and its $Y^{c'}$-axis aligns to Π_0 as shown in Fig. 3. The virtual camera C' shares its origin with C. The projection of the joint point in C' is denoted as $p_{joint}^{c'}$ which is at the principal point of the virtual camera. Let the normal vector of Π_i in C' be $\mathbf{n}_i^{c'}$, we have $\mathbf{n}_0^{c'} = [1, 0, 0]^T$, $\mathbf{n}_1^{c'} = [n'_{x1}, n'_{y1}, 0]^T$, $\mathbf{n}_2^{c'} = [n'_{x2}, n'_{y2}, 0]^T$, because the optical axis of C' passes though the junction. The rotation matrix $\mathbf{R}_{c'}^c$ and $\mathbf{n}_i^{c'}$ can be directly calculated from the input.

Similar to Section 3.1, a model coordinate frame $O^m X^m Y^m Z^m$ is created whose Z^m-axis aligns with $\mathbf{v}_0$. The solution is $\mathbf{R}_w^c = \mathbf{R}_{c'}^c \mathbf{R}_m^{c'} \mathbf{R}_w^m$, where $\mathbf{R}_{c'}^c$ and $\mathbf{R}_w^m$ can be calculated easily. To solve $\mathbf{R}_m^{c'}$, we note that it can be parameterized as

$$\mathbf{R}_m^{c'} = Rot(X, \alpha) Rot(Z, \beta), \tag{16}$$

where $Rot(X, \alpha)$ denotes a rotation around the X-axis, and $Rot(Z, \beta)$ denotes a rotation around the Z-axis. By using the geometric constraint that $\mathbf{v}_i^{c'} (i = 1, 2)$ should be perpendicular to the normal of Π_i, we have

$$\begin{cases} \mathbf{n}_1^{c'} \cdot \mathbf{v}_1^{c'} = \mathbf{n}_1^{c'T} \mathbf{R}_m^{c'} \mathbf{v}_1^m = 0 \\ \mathbf{n}_2^{c'} \cdot \mathbf{v}_2^{c'} = \mathbf{n}_2^{c'T} \mathbf{R}_m^{c'} \mathbf{v}_2^m = 0. \end{cases} \tag{17}$$

The primary difference between class B and class A is that the 3rd element of $\mathbf{n}_i^{c'}$ equals zero (i. e., $n'_{zi} = 0$) in class B, so that the complexity of the P3L polynomial can be effectively simplified. The cases discussed in class B are listed in the second part of Table. 1. Detailed analysis of the P3L solution for this class is provided in Section III of the supplementary material, available online.

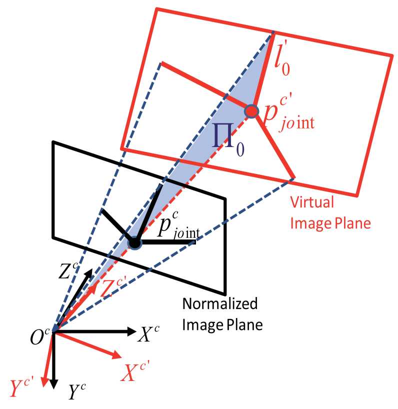

Fig. 3. Illustration of the virtual camera frame. The ray $O^c p_{joint}^c$ is perpendicular to the virtual image plane and l_0' is parallel to $Y^{c'}$-axis.

3.6 Discussion of the P3L Solution

In this section we provide a complete analysis on three spatial lines configurations, which also connects the previous approaches under a unified framework: In [15], the situation of three lines in Z-shape configuration is addressed which turns to be a special situation of case A.2.b. Our derivation is also much simpler than the method in [15]. When the three lines form a triangle in space [35], it can be considered as a special situation of case A.2.c. In our case A.2.c, three lines are parallel to a common plane, but they are not necessary lying on the same plane. The method in case A.2.c is more general than P3P, but with the same complexity (a quartic polynomial), which offers an alternative approach for the P3P problem. Caglioti [14] focuses on the situation of three lines lying on the same plane and intersecting at a point to form a junction, which is a special situation of case B.2.c. In our case B.2.c, we find the maximum number of feasible rotation solutions is eight, while in [14] only two candidate solutions of the rotation matrix are obtained as it ignores the symmetric property of the line direction.

It must be pointed out that in our analysis, we only consider the line configurations in 3D space, their projections in the image plane are not completely addressed because elaborating all the situations would be cumbersome. In [29], the authors showed how complex it could end up with by enumerating all the configurations of three points and their projections for P3P, and they even gave up to present all their results of the two step geometric approach (cf. Section 5 of [29]). Since P3P is only a very special case of the more general P3L problem considered in our context, one can imagine how tedious it will be to enumerate all the configurations of lines both in 2D and 3D. Furthermore, at this point only the configuration of *three* lines are considered. [36] discussed some critical sets of $n(>3)$ line configurations and their maximum number of candidate solutions. In what follows we will focus on the more general scenarios of $n > 3$.

4 Perspective-n-Line Algorithms

In this section, we propose an Accurate Subset-based PnL solution (**ASPnL**), a series of linear-formulation-based PnL solutions, and an efficient $O(1)$ camera pose refinement method.

4.1 Accurate Subset-Based PnL Solution

Select a Rotation Axis to Form the Model Frame. Given n reference lines L_i $(i = 1, \ldots, n)$ which are projected onto the normalized image plane as l_i, we first select a line L_{i0} from $\{L_i\}$ as a rotation axis $\mathbf{v}_0$, based on which the model coordinate frame $O^m X^m Y^m Z^m$ is created (see Fig. 1). Usually the line with the longest projection length $|\mathbf{p}_{is}\mathbf{p}_{ie}|$ is selected as the rotation axis because longer edges are less affected by noise on their endpoints. Furthermore, the line with the 2nd longest projection length is selected as an auxiliary line L_{i1}, so as to construct a polynomial equation system together with the rest of lines. The auxiliary line is introduced to keep the complexity of the proposed ASPnL algorithm *linear* in the number of line correspondences.

Determining the Rotation Axis. The line set $\{L_i\}$ can be divided into $n-2$ triplets $\{L_{i0}L_{i1}L_j | j = 1, \ldots, n; j \neq i0 \,\&\, j \neq i1\}$. According to Eq. (7), each triplet yields an 8th order polynomial:

$$\begin{cases} f_1(x) = \sum_{k=0}^{8} \delta_{1k} x^k = 0 \\ f_2(x) = \sum_{k=0}^{8} \delta_{2k} x^k = 0 \\ \cdots \\ f_{n-2}(x) = \sum_{k=0}^{8} \delta_{(n-2)k} x^k = 0. \end{cases} \tag{18}$$

Instead of directly solving the nonlinear equation system (18) by the linearization technique [22] that might lead to an inconsistent result from redundant equations, we explore the local minima of the system in terms of least-square residuals: A cost function F is defined as a square sum of the polynomials in Eq. (18), $F = \sum_{i=1}^{n-2} f_i^2(x)$. The minima of F can be identified by finding the roots of its derivative $F' = \sum_{i=1}^{n-2} f_i(x) f_i'(x) = 0$. F' is a 15th order polynomial which can be easily solved by the eigenvalue method [37].

Remark 1. The 16th order polynomial F has at most eight local minima.

Proof. Assuming F has m stationary points, in which there are m_1 minima and m_2 maxima, $m_1 + m_2 \leq m$. As there exists at least one maximum between two minima, we have $m_1 - 1 \leq m_2$. As the stationary points of F are the real roots of F', we have $m \leq 15$. Therefore, $2m_1 - 1 \leq m_1 + m_2 \leq 15$, and we have $m_1 \leq 8$. □

In general, there are only a few real roots among the minima. These real roots are chosen as the candidate solutions. As soon as x is solved, the rotation angle α around the X-axis in Eq. (1) can be calculated and the rotation axis $\mathbf{v}_0$ in the camera frame can be determined, i.e., $\mathbf{v}_0^c = \mathbf{R}' Rot(X, \alpha)\,[0, 0, 1]^T$. The remaining unknown variables are the rotation angle β around the axis $\mathbf{v}_0$ and the translation vector $\mathbf{t}$.

Solving the Rotation Angle and the Translation Vector. When the rotation axis $\mathbf{v}_0$ (i.e., Z_m axis of the model coordinate frame) is determined, from Eq. (1), the rotation matrix from the camera to the model coordinate frame $\mathbf{R}_m^c$ can be expressed as:

$$\mathbf{R}_m^c = \bar{\mathbf{R}}\, Rot(Z, \beta) = \begin{bmatrix} \bar{r}_{11} & \bar{r}_{12} & \bar{r}_{13} \\ \bar{r}_{21} & \bar{r}_{22} & \bar{r}_{23} \\ \bar{r}_{31} & \bar{r}_{32} & \bar{r}_{33} \end{bmatrix} \begin{bmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{19}$$

with $\bar{\mathbf{R}} = \mathbf{R}' Rot(X, \alpha)$, $c = \cos\beta$ and $s = \sin\beta$. Similar to the solution of P3L in Section 3.1, we have the following constraints:

$$\mathbf{n}_i^{cT} \mathbf{R}_m^c \mathbf{v}_i^m = 0, \quad \mathbf{n}_i^{cT} (\mathbf{R}_m^c P_i^m + \mathbf{t}) = 0, \tag{20}$$

where $\mathbf{t} = [t_x, t_y, t_z]^T$. For n lines, by substituting Eq. (19) into Eq. (20) and stacking these constraints, we get $2n$ homogeneous linear equations with parameter vector $[c, s, t_x, t_y, t_z, 1]^T$. Letting $\mathbf{n}_i^c = [n_{xi}, n_{yi}, n_{zi}]^T$, $(\mathbf{n}_i^{cT}\bar{\mathbf{R}}) = [\bar{n}_{xi}, \bar{n}_{yi}, \bar{n}_{zi}]$, $P_i^m = [p_{xi}, p_{yi}, p_{zi}]^T$ and $\mathbf{v}_i^m = [v_{xi}, v_{yi}, v_{zi}]^T$, we have:

$$[\mathbf{U}_1 \quad \mathbf{U}_2]\begin{bmatrix} c & s & t_x & t_y & t_z & 1 \end{bmatrix}^T = 0, \tag{21}$$

in which

$$\mathbf{U}_1 = \begin{bmatrix} \cdots & \cdots \\ \bar{n}_{xi}v_{xi} + \bar{n}_{yi}v_{yi} & \bar{n}_{yi}v_{xi} - \bar{n}_{xi}v_{yi} \\ \bar{n}_{xi}p_{xi} + \bar{n}_{yi}p_{yi} & \bar{n}_{yi}p_{xi} - \bar{n}_{xi}p_{yi} \\ \cdots & \cdots \end{bmatrix},$$

$$\mathbf{U}_2 = \begin{bmatrix} \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \bar{n}_{zi}v_{zi} \\ -n_{xi} & -n_{yi} & -n_{zi} & \bar{n}_{zi}p_{zi} \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix}.$$

The rotation angle β and the translation vector $\mathbf{t}$ can be estimated by solving this linear system in Eq. (21) with SVD. Then, the rotation matrix from the camera frame to the world frame is computed as $\mathbf{R}_w^c = \mathbf{R}_m^c \mathbf{R}_w^m$. The estimated rotation matrix $\mathbf{R}_w^c$ is further refined using the method described in Section 4.3 and the translation vector $\mathbf{t}$ can be determined by Eq. (27). Then we have a few candidate solutions corresponding to the minima of the polynomial system in Section 4.1. For each candidate, we evaluate its orthogonal error

$$E_{or} = \sum_{i=1}^{n} \left(\mathbf{n}_i^{cT} \mathbf{R}_w^c \mathbf{v}_i^w \right)^2 \tag{22}$$

and select the one with the smallest E_{or} as our final solution.

Discussion. In Section 3.5, a series of P3L polynomials are derived for distinct configurations. If the line configuration is known, we can simplify the complexity of the polynomial system. For example, for Manhattan world configuration where all spatial lines are either parallel or orthogonal to each other, the polynomials of case A.1.a and case A.2.a can be employed to simplify the cost function F which leads to at most two local minima. For planar configuration where all spatial lines lie on a plane, the polynomial of case A.2.c can be employed to reduce the maximum number of local minima to four. For the joint configuration, where all lines form a junction, the polynomial of case B.1.c can be employed to produce at most four local minima. In general, without constricting the spatial lines to any specific configuration, the general P3L polynomial of 8th order can be used to form the cost function F and to find the global minimum as the final solution.

The philosophy of the proposed ASPnL approach is closely related to the previous works for perspective-n-point problem [35], [38], [39], in which n points are divided into a set of triplets, then unreasonable solutions are eliminated by examining consistency among the triplets' solutions. This idea can be regarded as a "bottom up" approach. In [35], [38], all 3-point subsets are solved individually, then a random sampling scheme or the Hough transform method are used to find the most consistent solution. Meanwhile [38] and [35] fail to incorporate the global information of all the n points in an equation system for a more accurate result. In [39], the P3L polynomials of the subsets are stacked together to form a system of nonlinear equations, and a linearization technology is used to solve the equations, however the local minimum problem stemming from the underlying nonlinear equations is not considered.

On the other hand, the *key* parameter directly related to the multiple candidate solutions of P3L and the local

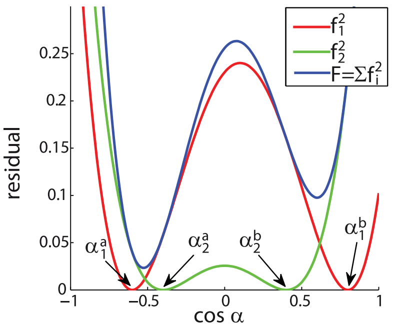

Fig. 4. A simplified illustration of the local minima of the PnL problem with respect to α. f_i denotes the P3L polynomial in Eq. (18). The curves of f_i^2 and $F = \Sigma f_i^2$ are plotted. The local minima of F are related to the solutions of f_i.

minima of PnL is the rotation angle α. To illustrate the relationship between the candidate solutions of the P3L polynomial f_i in Eq. (18) and the local minima of the cost function $F = \Sigma f_i^2$, let's consider a simplified case which contains only two 3-line subsets with corresponding P3L polynomials f_1 and f_2. Assuming that each f_i has only two local solutions: the solutions of f_1 are (α_1^a, α_1^b), and the solutions of f_2 are (α_2^a, α_2^b). The curves of f_1^2, f_2^2 and $F = \Sigma f_i^2$ are plotted in Fig. 4. Clearly the local minima of F depend on the solutions of f_i. Moreover, the local minimum of F locating between α_1^a and α_2^a is lower than the other local minimum locating between α_1^b and α_2^b, despite the fact that the individual values of $f_1(\alpha_1^a)$, $f_1(\alpha_1^b)$, $f_2(\alpha_2^a)$, and $f_2(\alpha_2^b)$ are all the same. This is because the distance between α_1^a and α_2^a is smaller, which indicates these two triplets are more agreeable to each other. In practice, f_i has at most eight possible solutions, and $F = \Sigma f_i^2$ has at most eight local minima accordingly (see Remark 1). These local minima of F are examined one by one, so that an optimal solution can be ensured as the output of our ASPnL method.

4.2 Linear-Formulation-Based PnL Solutions

PnL problem is closely related to the well studied PnP problem. Recently, the state-of-the-art PnP solver OREPnP [8] has been shown to be robust against not only image noise but also outliers. Meanwhile most existing PnL solvers are not as stable to outliers. This inspires us to propose a series of approaches based on linearized PnL formulation, among them several are quite robust to outliers. We start by revisiting the similarity between PnL and PnP problem.

PnP Linear Formulation. Let a 3D reference point be $\mathbf{P}_i^w = [x_i^w \; y_i^w \; z_i^w]^T$ in the world frame, which is denoted as $\mathbf{P}_i^c = [x_i^c \; y_i^c \; z_i^c]^T$ in the camera frame. The corresponding 2D projection of the point on the image plane is $\mathbf{u}_i = [u_i \; v_i]^T$. By perspective projection constraint we have

$$\begin{bmatrix} \lambda_i u_i \\ \lambda_i v_i \\ \lambda_i \end{bmatrix} = \mathbf{A} \begin{bmatrix} x_i^c \\ y_i^c \\ z_i^c \end{bmatrix}, \tag{23}$$

where $\lambda_i \in \mathbb{R}$ is an unknown scaling factor, and $\mathbf{A}$ is a 3×3 projection matrix. By substituting the 3rd row of (23) into the first two rows, the scaling factor λ_i can be eliminated and we are left with two constrains equations for each point correspondence $\mathbf{P}_i \leftrightarrow \mathbf{u}_i$. To solve this problem, the unknown camera pose can be parametrized in two forms:

(i) The camera pose can be parameterized by a 3×3 rotation matrix $\mathbf{R}_w^c$ and a 3×1 translation vector $\mathbf{t}$, which leads to a 3D point $\mathbf{P}_i^c = \mathbf{R}_w^c \mathbf{P}_i^w + \mathbf{t}$. The elements in $\mathbf{R}_w^c$ and $\mathbf{t}$ can be vectorized to form an unknown vector $\mathbf{x} = [r_1\ r_2 \cdots r_9\ t_x\ t_y\ t_z]^T$ of size 12×1 (where r_i are the elements of $\mathbf{R}_w^c$ in column-major order). By concatenating constraint equations of n point correspondences, a linear system is obtained

$$\mathbf{M}\,\mathbf{x} = \mathbf{0}, \tag{24}$$

where $\mathbf{M}$ is a $2n \times 12$ matrix, and every two rows of $\mathbf{M}$ corresponds to a projection constraint defined in Eq. (23). The result of Eq. (24) is obtained with the standard **DLT** method [21].

(ii) The camera pose can also be expressed as four control points $\{\mathbf{C}_j^c\},\ j = 1 \cdots 4$ as in [5], and each 3D point $\mathbf{P}_i^c$ can be expressed as barycentric coordinates α_{ij}, i.e., $\mathbf{P}_i^c = \sum_{j=1}^4 \alpha_{ij} C_j^w$. By concatenating the coordinates of control points $\{\mathbf{C}_j^c\}$ we have a vector $\mathbf{x} = [\mathbf{C}_1^T \cdots \mathbf{C}_4^T]^T$ of size 12×1. It yields a linear system with the same form of Eq. (24) that uncovers the linear formulation of [5], [8].

PnL Linear Formulation. Let the endpoints of the i_{th} 3D reference line be $(\mathbf{P}_i^w, \mathbf{Q}_i^w)^2$ in the world frame and $(\mathbf{P}_i^c, \mathbf{Q}_i^c)$ in the camera frame. Camera observation of the line can be expressed as a projection plane passing through the camera center and the line. Let the normal of the i_{th} projection plane be $\mathbf{n}_i$, and we have the two constraints for each line

$$\begin{cases} \mathbf{n}_i^T \mathbf{P}_i^c = 0 \\ \mathbf{n}_i^T \mathbf{Q}_i^c = 0. \end{cases} \tag{25}$$

Similar to PnP problem, we have the following two options to construct a linear system for PnL:

(i) The camera pose can be parameterized as rotation $\mathbf{R}$ and translation $\mathbf{t}$. We have $\mathbf{P}_i^c = \mathbf{R}_w^c \mathbf{P}_i^w + \mathbf{t}$ and $\mathbf{Q}_i^c = \mathbf{R}_w^c Q_i^w + \mathbf{t}$. By vectorizing the elements in $\mathbf{R}_w^c$ and $\mathbf{t}$, and concatenating the constraint equations of n line correspondences, we have a linear system of the same form as Eq. (24).

(ii) Inspired by [5], [8], we express the end points of a line by using barycentric coordinates $\mathbf{P}_i^c = \sum_{j=1}^4 \alpha_{ij} \mathbf{C}_j^w$ and $\mathbf{Q}_i^c = \sum_{j=1}^4 \beta_{ij} \mathbf{C}_j^w$. By concatenating the coordinates of control points $\{\mathbf{C}_j^c\}$ we have an unknown vector $\mathbf{x} = [\mathbf{C}_1^T \cdots \mathbf{C}_4^T]^T$ of size 12×1, which yields a linear system in the same form as Eq. (24).

Null Space and Solution. The solution of the linear system Eq. (24) resides in the null space of $\mathbf{M}$. In noise free case, the rank of null-space of $\mathbf{M}$ is one for $n \geq 6$. Meanwhile in real-world application, the rank of null-space of $\mathbf{M}$ could be 0 due to noise and outliers. Furthermore, the solution is supposed to satisfy the geometric constraints among the elements of $\mathbf{x}$. There exist several strategies to solve $\mathbf{M}\,\mathbf{x} = \mathbf{0}$, with the two commonly used ones listed below:

(a) The eigenvector corresponding to the smallest eigenvalue is considered as the solution of $\mathbf{x}$, which is a least square solution of Eq. (24) without considering the geometric constraints among the elements of $\mathbf{x}$, i. e., the same as **DLT** [21].

(b) A subset of eigenvectors $\{\Gamma_k\}$ with the smallest eigenvalues are considered as the "effective null space" [5]. With the size of effective null space being 1, 2, 3 or 4, the solution is then expressed as $\mathbf{x} = \sum_k \eta_k \Gamma_k$ as a linear combination of the effective null space. The coefficients η_k can be determined by an optimization scheme to enforce the geometric constraints among the elements of $\mathbf{x}$, i.e., in [5] the constraints are the pair distances between the control points. A discussion about the size of effective null space can be found in Section V of the supplemental material, available online. A maximum number of four eigen vectors are suitable for PnL problem. It is interesting to mention that for PnP problem of multi-camera system, the maximum number of five null space vectors yields good result, and more details can be found in [40].

Outlier Removal. Inspired by [8], we design an outlier removal scheme for PnL problem, where the camera pose is iteratively optimized by minimizing the following objective

$$\mathbf{x} = \arg\min_{\mathbf{x}} ||\mathbf{W}(\mathbf{x})\mathbf{M}\,\mathbf{x}||^2 \quad s.t.\ ||\mathbf{x}||^2 = 1,$$

where $\mathbf{W}(\mathbf{x}) = \mathrm{diag}(\cdots w_i, w_i, \cdots)$ is a diagonal matrix of size $2n \times 2n$ whose diagonal elements depend on $\mathbf{x}$. w_i can be either 1 or 0, where $w_i = 1$ means that the corresponding 2D/3D line correspondence is an inlier and $w_i = 0$ means the correspondence is an outlier. Let the residual vector be $\mathbf{M}\,\mathbf{x} = [\tau_1 \cdots \tau_{2n}]^T$, and $\epsilon_i(\mathbf{x}) = ||[\tau_{2i-1}\ \tau_{2i}]||$ be the residual of i_{th} line correspondence given $\mathbf{x}$. The elements of $\mathbf{W}(\mathbf{x})$ are updated as follows:

$$w_i = \begin{cases} 1 & if\ \epsilon_i(\mathbf{x}) \leq \tau^* \\ 0 & otherwise, \end{cases}$$

where τ^* is a threshold to determine whether a correspondence is an inlier or an outlier. We set $\tau^* = \rho Q_{\kappa\%}(\epsilon_1 \cdots \epsilon_n)$, where ρ is a real number coefficient, and $Q_{\kappa\%}$ denotes a value to threshold off the lowest κ percent data from the majority $1 - \kappa$ percent data. Empirically we found that $\rho = 2$ and $\kappa = 30$ yield satisfying robustness. $\mathbf{x}$ and $\mathbf{W}(\mathbf{x})$ are updated iteratively until the target function $||\mathbf{W}(\mathbf{x})\mathbf{M}\,\mathbf{x}||^2$ cannot be further minimized. Empirically the algorithm often converges within five iterations.

A series of new Linear-formulation-based PnL approaches are introduced here by combining the options mentioned above.[3] They are

(1) **LPnL_DLT_LS**: Here $\mathbf{x}$ is parameterized using $\mathbf{R}_w^c$, $\mathbf{t}$, and the linear system $\mathbf{M}\,\mathbf{x} = \mathbf{0}$ is solved by the least square solver. This approach corresponds to the standard DLT approach for the PnP problem [21].

(2) **LPnL_DLT_ENull**: Here $\mathbf{x}$ is parameterized using $\mathbf{R}_w^c$, $\mathbf{t}$, and then be solved by the effective null space solver.

2. The endpoints $(\mathbf{P}_i^w, \mathbf{Q}_i^w)$ can be chosen as any two points on line L_i.

3. Please refer to supplemental material, available online, Section IV for pseudo-code.

(3) **LPnL_Bar_LS**: Here **x** is parameterized with the barycentric coordinates, and the linear system is solved by the least square solver.

(4) **LPnL_Bar_ENull**: Here **x** is parameterized with the barycentric coordinates, and the linear system is solved by the effective null space solver. This approach corresponds to EPnP+GN [5] for the PnP problem.

(5) **RLPnL_LS**: This is the robust variant of **LPnL_Bar_LS**[4] with outlier removal module. The geometric constraints among the elements of **x** are not enforced in the iterations as in [8]. This approach is corresponding to REPPnP [8] for the PnP problem.

(6) **RLPnL_ENull:** This is the robust variant of **LPnL_Bar_ENull** with outlier removal module. The geometric constraints of **x** are enforced in each iteration.

Extension to Lines and Points. As the linear formulations of PnL and PnP have the same form of Eq. (24), our LPnL method can be easily extended to deal with lines and points simultaneously by concatenate the linear system of lines and points. Then the problem can be solved the same as mentioned above.

4.3 Camera Pose Refinement

It is more accurate (and unfortunately also more time consuming) to optimize the rotation and translation parameters simultaneously than to optimize the rotation only [18]. By expressing the translation vector in term of the rotation parameters in a least-square sense, the optimization can be efficiently performed on only three rotation parameters. Moreover, it can be used for both n-lines and n-points. We parameterize **x** using rotation and translation, and the linear system can be split into two parts

$$[\mathbf{A} \quad \mathbf{B}]\begin{bmatrix}\mathbf{r}\\ \mathbf{t}\end{bmatrix} = \mathbf{0}. \tag{26}$$

Eq. (26) can be written in the following form $\mathbf{A}\,\mathbf{r} = -\mathbf{B}\,\mathbf{t}$, and the least-square solution of $\mathbf{t}$ is

$$\mathbf{t} = -(\mathbf{B^T B})^{-1}\mathbf{B^T A}\,\mathbf{r}, \tag{27}$$

Substituting Eq. (27) into Eq. (26), we have

$$\mathbf{D}\,\mathbf{r} = \mathbf{0}, \tag{28}$$

where $\mathbf{D} = \mathbf{A} - (\mathbf{B^T B})^{-1}\mathbf{B^T A}$ is a $2n \times 9$ matrix.

Let the rotation $\mathbf{R}_w^c = \tilde{\mathbf{R}}_w^c \Delta\mathbf{R}$, where $\tilde{\mathbf{R}}_w^c$ is the initial rotation, and $\Delta\mathbf{R}$ is the compensation rotation. We express $\Delta\mathbf{R}$ using CGR parameterization [24], [32]. It is known that the CGR parameter degenerates when the rotation angle is $\pi/2$, whereas it is stable in our application as $\tilde{\mathbf{R}}_w^c$ is supposed to be near $\mathbf{R}_w^c$ and the rotation angle of $\Delta\mathbf{R}$ is small.

Let $\mathbf{s} = [s_1\ s_2\ s_3]^T$ be the CGR parameter vector,[5] we have $\Delta\mathbf{R} = \Delta\mathbf{R}/(1 + \mathbf{s^T s})$, where the elements of $\Delta\mathbf{R}$ are vectorized in column major order as $[s_1^2 - s_2^2 - s_3^2 + 1,\ 2s_1s_2 + 2s_3,\ 2s_1s_3 - 2s_2,\ 2s_1s_2 - 2s_3,\ -s_1^2 + s_2^2 - s_3^2 + 1,\ 2s_2s_3 + 2s_1,\ 2s_1s_3 + 2s_2,\ 2s_2s_3 - 2s_1,\ -s_1^2 - s_2^2 + s_3^2 + 1]$. Then a cost function F is constructed as the squared sum of Eq. (28)

$$F(s_1, s_2, s_3) = \mathbf{r^T D^T D\,r}.$$

As $\mathbf{r}$ can be expressed using the CGR parameter, F contains three unknowns $(s_1\ s_2\ s_3)$ of order 4, thus the solution can be obtained by optimizing the cost function F. The optimization process can be efficiently performed using the Newton method: (1) initialize with $\mathbf{s}_0$; (2) for each iteration k, compute the gradient vector ∇F and the Hessian matrix $\mathbf{H}$; (3) iteratively update $\mathbf{s}_{k+1} = \mathbf{s}_k - \mathbf{H}^{-1}\nabla F$ until convergence or the iteration number exceeds a threshold. This iterative refinement process can be carried out in a very short time, because no matter how large the number of lines n is, the size of $\mathbf{D}^T\mathbf{D}$ is constant, and the complexity for each iteration is $O(1)$.

5 EXPERIMENTS

5.1 Experiments with Synthetic Data

Given a virtual perspective camera with image size 640×480 pixels and focal length 800 pixels, the 3D reference lines are randomly generated in the camera coordinate frame. In the *centred case*, the 2D projections of the end-points of the 3D lines spread in region $[0, 640] \times [0, 480]$ pixels on image plane. In the *uncentred case*, the 2D projections of the end-points of the 3D lines scatter in region $[0, 320] \times [0, 240]$ pixels on image plane. The depth of the lines are randomly distributed in range $[4, 8]$ m in the camera coordinate frame. Different levels of random noise are added to the end point locations of the projected image lines. The outliers are incorrect 3D/2D correspondences generated by randomly selecting correspondences between the 3D and 2D features. All the plots are created by running 500 independent simulations. The implementations of our proposed methods are available on website.[6]

The error metric is defined the same as in [7], [8]. $\mathbf{R}_{true}$ and $\mathbf{t}_{true}$ denotes the ground truth rotation and translation, and $\mathbf{R}$, $\mathbf{t}$ the estimated results. Rotation error is defined as $Err_{\mathbf{R}} = \max\{angle(\mathbf{R}_{true}(:, i), \mathbf{R}(:, i)\}\ (i = 1, 2, 3)$, where $:$ follows the Matlab indexing notation, and $angle(\cdot, \cdot)$ denotes the measured angle distance between the i_{th} column vectors of $\mathbf{R}_{true}$ and $\mathbf{R}$. Translation error $Err_{\mathbf{t}} = ||\mathbf{t} - \mathbf{t}_{true}||/||\mathbf{t}_{true}||$. An estimation is counted as correct when $Err_R < 5\ deg$, and $Err_t < 5$ percent.

The following methods are compared: **Ansar** [22], **Bronislav** [41], **Mirzaei** [25], **ASPnL** described in Section 4.1, and the Linear-formulation-based solutions described in Section 4.2.

(i). Centred and Uncentred Cases: The accuracy of the compared methods in outlier-free case is shown in row a and b of Fig. 5. Generally speaking, **ASPnL** is the most accurate algorithm in both centred and uncentred cases. The performances of **ENull** based methods come in second. Experimental results show that the accuracies of **LPnL_Bar_ENull** and **LPnL_DLT_ENull** are similar. For the **LS** based methods, **LPnL_Bar_LS** outperforms

4. Generally speaking, the performances of **Bar** based methods are similar/better than that of **DLT** based methods. Therefore, in the robust variant **RLPnL** we consider only **Bar** based methods for concise.

5. For the unified PnP problem, Kneip et al. [28] proposed a geometrically optimal quaternion-based formulation which can be extended for PnL refinement. Details can be found in Section VI of the supplemental material, available online.

6. http://xuchi.weebly.com/rpnl.html

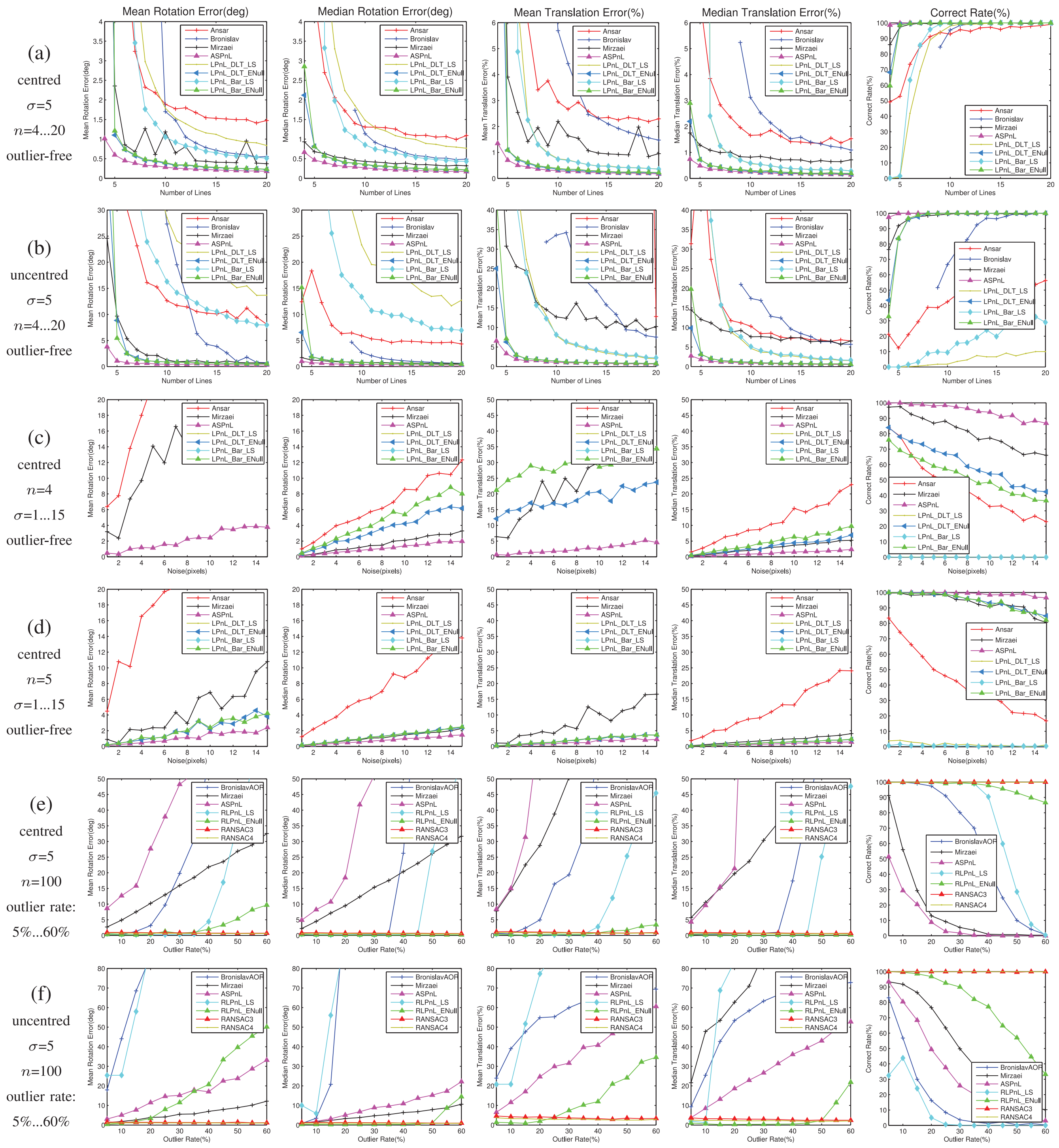

Fig. 5. Synthetic experimental results. n denotes the number of lines, and σ denotes the standard deviation of image noise. In row (a) and (b), the accuracies of the compared methods are tested in centred and uncentred cases by varying n from 4 to 20. In row (c) and (d), the robustness against noisy small line-set is tested with $n = 4$ or 5, and σ varies from 1 to 15 pixels. In row (e) and (f), the robustness against outliers is tested in centred and uncentred cases by varying outlier rate from 5 to 60 percent.

LPnL_DLT_LS. The observation suggests that the least square solution of the barycentric coordinate based equation system is better than that of the **DLT** based equation system. On the other hand, the effective null space of both **Bar** and **DLT** based equation system are similar. The accuracies of the **LS** based methods degenerate significantly in uncentred case, but the **ENull** based methods perform well in both centred and uncentred cases, and it is as accurate as **ASPnL** when $n > 10$. In uncentred case, the primary difficulty lies in the local minima problem. Experimental results suggest that the least-square solver is prone to local minima, while the effective null space solver is an effective tool to boost the performance of linear-formulation-based PnL solver against local minima problem. We also observe that **Bronislav** achieves relatively higher rotation accuracy than other **LS** based methods in uncentred case when $n > 16$, and it may due to the advantage of using of Plucker Coordinates. While **Bronislav** needs at least nine lines.

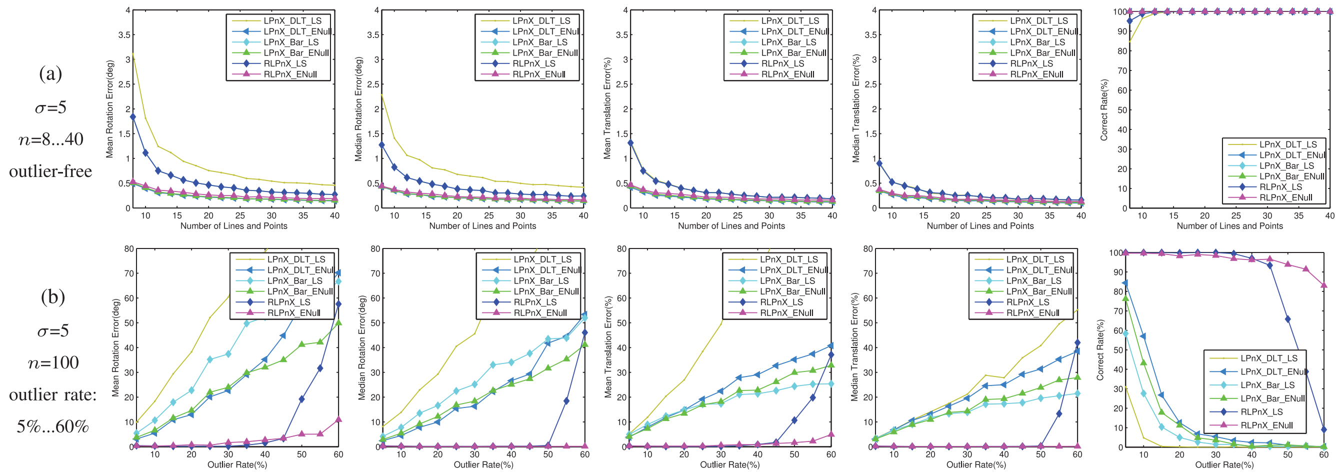

Fig. 6. Camera pose estimation using lines and points. n denotes the number of features in which 50 percent are lines and 50 percent are points. σ denotes the standard deviation of image noise. The experiments are tested in centred case. In row (a), the accuracy is tested by varying n from 8 to 40. In row (b), the robustness against outliers is tested by varying outlier rate from 5 to 60 percent.

(ii). Noisy Small Line-set: The PnL solutions are prone to be unstable when the inputs are noisy small line-sets, i.e., $n = 4$ *or* 5. As can be seen in row c and d of Fig. 5, when $n = 4$, the performance of **ASPnL** is significantly better than others, and **Mirzaei** comes in 2nd, meanwhile the rest methods fail. The correct rate of **ASPnL** is around 94 percent when the image noise σ reaches 10 pixels, and that of **Mirzaei** is about 77 percent. When $n = 5$, **ASPnL**, **Mirzaei** and the **ENull** based methods perform robustly, in which **ASPnL** is the best in both accuracy and correct rate. Note that **Bronislav** cannot deal with small line-set as it needs at least nine lines.

(iii). Outliers: In the outliers case, the following methods are compared: **RANSAC3**, **RANSAC4**, **ASPnL**, **BronislavAOR** [41], **Mirzaei**, **RLPnL_LS** and **RLPnL_ENull**. In **RANSAC3** our P3L solver is used in the RANSAC strategy, and in **RANSAC4** ASPnL with $n = 4$ is used. As can be seen in row e and f of Fig. 5, **ASPnL** and **Mirzaei** are not robust to outliers, but **RLPnL_ENull** achieves satisfying performance with reasonable amount of outliers. In the centred case, the correct rate of **RLPnL_ENull** is 93 percent with 50 percent outliers. In the uncentred case, the performance of **RLPnL_ENull** is negatively affected, and its correct rate is 90 percent with 30 percent outliers. **RLPnL_LS** is robust with up to 35 percent outliers in the centred case, but it cannot deal with the uncentred case. **Bronislav** achieves reliable result within 15 percent outliers in centred case, but it is not stable in uncentred case. The performances of **RANSAC3** and **RANSAC4** are the best, both have 100 percent correct rate with upto 60 percent outliers, but they are not as efficient as the Linear-formulation-based methods.

(iv). Efficiency: The running time of the compared methods in outlier-free case is shown (a) of Fig. 7. We can see that the linear-formulation-based methods and **Bronislav** are very efficient, and the running time of **ASPnL** and **Mirzaei** grows linear with respect to n. **Ansar** is time consuming for large line-set as the algorithm is $O(n^2)$. In outlier case, as shown in b of Fig. 7, the linear-formulation-based methods are the most efficient in the comparison list. **RANSAC3** with our P3L solver is efficient when the outlier rate ≤ 40 percent, and its running time increases to 138 ms when the outlier rate reaches 60 percent.

5.2 Using Lines and Points

The experiments for pose estimation from lines and points are shown in Fig. 6. PnX denotes Perspective-n-X, where "X" denotes both line and point features. The compared methods are: **LPnX_DLT_LS**, **LPnX_DLT_ENull**, **LPnX_Bar_LS**, **LPnX_Bar_ENull**, **RLPnX_LS**, and **RLPnX_ENull**. These methods are corresponding to their PnL counterparts. The number of features evaluated is n, and in our test 50 percent are lines and 50 percent are points. Fig. 6a shows that in outlier-free case **LPnX_Bar_LS** outperforms **LPnX_DLT_LS** as the Barycentric coordinate is used, while the performances of **LPnX_DLT_ENull** and **LPnX_Bar_ENull** are similar. Fig. 6b shows the robustness of the compared methods against outliers. The correct rate of **RPnX_Enull** slowly decreases when the outlier rate increases. We observe that it is helpful to improve the pose estimation performance by combining point features: In Fig. 5e, **RLPnL_LS** endures 35-40 percent outliers using only line features, but in Fig. 6b, **RLPnX_LS** endures upto 45 percent outliers by using points and lines simultaneously.

5.3 Experiments with Real Images

In order to compare the PnL solutions in real situations, we apply the algorithms on a set of images with known 3D line model. For each image, we extract lines using the LSD line

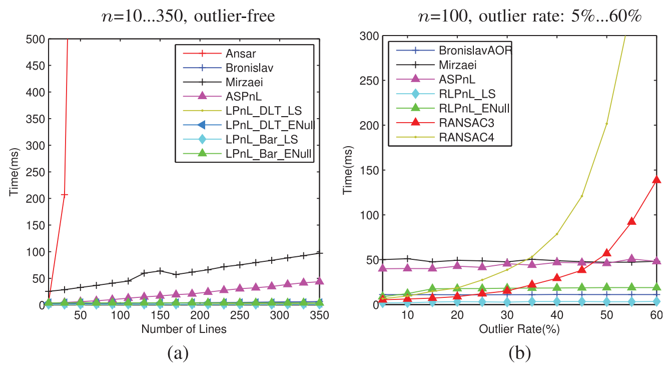

Fig. 7. Running Time. In (a), running time is tested by varying n from 10 to 350 in outlier-free case. In (b), $n = 100$ and the outlier rate varies from 5 to 60 percent.

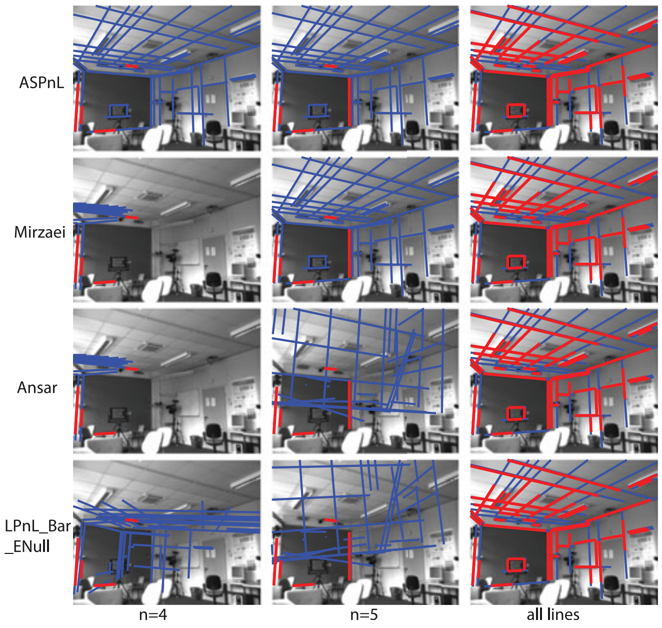

Fig. 8. Performance evaluation on real image with small line set. The experimental results illustrate the compared PnL solutions for real images when using 4, 5 or all the available line correspondences. The red lines are the used 2D line segments. The blue lines are the projection of the 3D line model using the estimated camera pose.

detector [42], then establish the correspondences between the image lines and the 3D line model. We test the compared algorithms by using 4, 5 and all the available line correspondences. In order to demonstrate the accuracy of the results, we reproject the 3D line model into the image by using the estimated camera pose. Fig. 8 shows the reprojection results on a sampled image from the image set. As shown in Fig. 8, when using only four pairs of 2D/3D line correspondences, only **ASPnL** robustly estimates the camera pose, i. e., the 3D line model is correctly projected into the image. When using five pairs of correspondences, both **ASPnL** and **Mirzaei** estimate the camera pose accurately. If all the available line correspondences are used, then all the compared methods can accurately estimate the camera pose. In Section VII of supplementary material, available online, we also test the performance of the compared methods when outliers exist and the performance of the proposed algorithm in real time applications.

6 Conclusion

In this paper, we first derive a generic 8th order P3L polynomial to characterize the basic building block of PnL involving only three spatial lines. The solution of this P3L subproblem is discussed by a complete investigating of three spatial lines in all possible configuration scenarios. In our analysis, the 3D configuration of the P3L problem is divided into 12 cases, and the order of the P3L polynomial can be further reduced to 4, 2 or 1 in different cases. The analysis includes a number of previous works as special P3L configurations in a unified framework. The well-know P3P problem can also be regarded as a special sub-case of case A.2.c in Section 3.5.

For the n-line problem, we propose a series of PnL solutions, and all of them have their advantages and disadvantages: (i) **ASPnL** is accurate and efficient to the noisy small line set ($3 < n \leq 5$) in both centred and uncentred cases as it explores the local minima. But this solution is not robust to outliers. (ii) **RLPnL_LS** is robust and very efficient for large line set ($n \geq 100$) with upto 35 percent outliers in centred case, but it is not stable in uncentred case. **RLPnL_ENull** is significantly more robust and accurate than **RLPnL_LS** in both centred and uncentred cases as it explores the effective null space. But **RLPnL** based methods are not suitable for small line-set. (iii) Besides, though the **RLPnL** based methods are efficient and robust to outlier, they are not guaranteed to converge to a global optimum, and cannot guaranty 100 percent correct rate when the outlier rate is more than 40 percent. Meanwhile the **P3L** based **RANSAC** method achieves 100 percent correct rate with even upto 60 percent outliers, and this observation again highlights the significance of the P3L solution analysis.

In practice, we suggest a combination of the above mentioned methods to fully utilize their advantages: When $n < 10$ use **ASPnL**, otherwise use **RLPnL_ENull**. If the outlier rate is high, **P3L** based **RANSAC** is necessary.

The work presented in this paper is of both theoretical and practical interests. The solution analysis of the P3L problem reveals the complexity of the problem for different cases with respect to the maximum number of solutions ($\mathbf{R}_w^c$) and the order of the simplified P3L polynomial. The proposed PnL algorithms are suitable for applications that need to handle either a small or a large number of line correspondences, such as the line feature based object tracking or camera localization in augmented reality.

Acknowledgments

Lilian Zhang is supported by National Natural Science Foundation of China (No. 61503403, No. 61573371). Chi Xu and Li Cheng are partially supported by A*STAR JCO grant 1431AFG120. The authors would like to thank the anonymous reviewers for their help comments and suggestions. The authors would like to thank Prof. Xavier Binefa and Luis Ferraz for their helpful discussion. Lilian Zhang is the corresponding author.

References

[1] D. Dementhon and L. Davis, “Model-based object pose in 25 lines of code,” *Int. J. Comput. Vis.*, vol. 15, no. 1–2, pp. 123–141, 1995.
[2] J. Yuan, “A general photogrammetric method for determining object position and orientation,” *IEEE Trans. Robot. Autom.*, vol. 5, no. 2, pp. 129–142, Apr. 1989.
[3] L. Vacchetti, V. Lepetit, and P. Fua, “Combining edge and texture information for real-time accurate 3D camera tracking,” in *Proc. 3rd IEEE/ACM Int. Symp. Mixed Augmented Real.*, 2004, pp. 48–57.
[4] F. Zhou, H.-L. Duh, and M. Billinghurst, “Trends in augmented reality tracking, interaction and display: A review of ten years of ISMAR,” in *Proc. 7th IEEE/ACM Int. Symp. Mixed Augmented Real.*, 2008, pp. 193–202.
[5] V. Lepetit, F. Moreno-Noguer, and P. Fua, “EPnP: An accurate O(n) solution to the PnP problem,” *Int. J. Comput. Vis.*, vol. 81, no. 2, pp. 155–66, 2009.
[6] J. Hesch and S. Roumeliotis, “A direct least-squares (DLS) method for PnP,” in *Proc. Int. Conf. Comput. Vis.*, 2011, pp. 383–390,.
[7] S. Li, C. Xu, and M. Xie, “A robust O(n) solution to the perspective-n-point problem,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 7, pp. 1444–1450, Jul. 2012.
[8] L. Ferraz, X. Binefa, and F. Moreno-Noguer, “Very fast solution to the PnP problem with algebraic outlier rejection,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 501–508.
[9] C. L. Zitnick and P. Dollár, “Edge boxes: Locating object proposals from edges,” in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 391–405.

[10] H. Chen, "Pose determination from line-to-plane correspondences: Existence condition and closed-form solutions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 6, pp. 530–541, Jun. 1991.
[11] L. Zhang, C. Xu, K.-M. Lee, and R. Koch, "Robust and efficient pose estimation from line correspondences," in *Proc. 11th Asian Conf. Comput. Vis.-Vol. Part III*, 2013, pp. 217–230.
[12] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge, U.K.: Cambridge University Press, 2004.
[13] M. Dhome, M. Richetin, J. Lapreste, and G. Rives, "Determination of the attitude of 3D objects from a single perspective view," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 12, pp. 1265–1278, Dec. 1989.
[14] V. Caglioti, "The planar three line junction perspective problem with application to the recognition of polygonal patterns," *Pattern Recognit.*, vol. 26, no. 11, pp. 1603–1618, 1993.
[15] L. Qin and F. Zhu, "A new method for pose estimation from line correspondences," *Acta Autom. Sin.*, vol. 34, no. 2, pp. 130–134, 2008.
[16] L. Zhang, H. Lu, X. Hu, and R. Koch, "Vanishing point estimation and line classification in a Manhattan world with a unifying camera model," *Int. J. Comput. Vis.*, vol. 117, pp. 111–130, 2015.
[17] Y. Liu, T. Huang, and O. Faugeras, "Determination of camera location from 2-D to 3-D line and point correspondences," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 1, pp. 28–37, Jan. 1990.
[18] R. Kumar and A. Hanson, "Robust methods for estimating pose and a sensitivity analysis," *Comput. Vis. Image Underst.*, vol. 60, no. 3, pp. 313–342, 1994.
[19] S. Christy and R. Horaud, "Iterative pose computation from line correspondences," *Comput. Vis. Image Underst.*, vol. 73, no. 1, pp. 137–144, 1999.
[20] F. Dornaika and C. Garcia, "Pose estimation using point and line correspondences," *Real-Time Imaging*, vol. 5, no. 3, pp. 215–230, 1999.
[21] Y. Abdel-Aziz, "Direct linear transformation from comparator coordinates in close-range photogrammetry," presented at the ASP Symp. Close-Range Photogrammetry, Illinois, USA, 1971.
[22] A. Ansar and K. Daniilidis, "Linear pose estimation from points or lines," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 5, pp. 578–589, May 2003.
[23] T. Q. Phong, R. Horaud, A. Yassine, and P. D. Tao, "Object pose from 2-D to 3-D point and line correspondences," *Int. J. Comput. Vis.*, vol. 15, no. 3, pp. 225–243, 1995.
[24] F. M. Mirzaei and S. I. Roumeliotis, "Globally optimal pose estimation from line correspondences," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 5581–5588.
[25] F. M. Mirzaei and S. I. Roumeliotis, "Optimal estimation of vanishing points in a Manhattan world," in *Proc. Int. Conf. Comput. Vis.*, 2011, pp. 2454–2461.
[26] L. Kneip, D. Scaramuzza, and R. Siegwart, "A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2011, pp. 2969–2976.
[27] Y. Zheng, Y. Kuang, S. Sugimoto, and K. Astrom, "Revisiting the PnP problem: A fast, general and optimal solution," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 2344–2351.
[28] L. Kneip, H. Li, and Y. Seo, "UPnP: An optimal O(n) solution to the absolute pose problem with universal applicability," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 127–142.
[29] X. Gao, X. Hou, J. Tang, and H. Cheng, "Complete solution classification for the perspective-three-point problem," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 8, pp. 930–943, Aug. 2003.
[30] D. A. Cox, J. Little, and D. O'shea, *Using Algebraic Geometry*, 2nd ed. New York, NY, USA: Springer, 2005.
[31] S. Li and C. Xu, "A stable direct solution of perspective-three-point problem," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 25, no. 5, pp. 627–642, 2011.
[32] M. Shuster, "A survey of attitude representations," *J. Astronaut. Sci.*, vol. 41, pp. 439–517, 1993.
[33] R. Hartley, J. Trumpf, Y. Dai, and H. Li, "Rotation averaging," *Int. J. Comput. Vis.*, vol. 103, no. 3, pp. 267–305, 2013.
[34] S. Li and C. Xu, "Efficient lookup table based camera pose estimation for augmented reality," *Comput. Animat. Virtual Worlds*, vol. 22, no. 1, pp. 47–58, 2011.
[35] S. Linnainmaa, D. Harwood, and L. S. Davis, "Pose determination of a three dimensional object using triangle paris," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 10, no. 5, pp. 634–47, Sep. 1988.
[36] N. Navab and O. Faugeras, "Monocular pose determination from lines: Critical sets and maximum number of solutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 1993, pp. 254–260.
[37] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes: The Art of Scientific Computing*. Cambridge, U.K.: Cambridge Univ. Press, 2007.
[38] M. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.
[39] L. Quan and Z. Lan, "Linear N-point camera pose determination," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 8, pp. 774–780, Aug. 1999.
[40] L. Kneip, P. Furgale, and R. Siegwart, "Using multi-camera systems in robotics: Efficient solutions to the NPnP problem," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2013, pp. 3770–3776.
[41] B. Přibyl, P. Zemčík, and M. Čadík, "Camera pose estimation from lines using Plücker coordinates," in *Proc. Brit. Mach. Vis. Conf.*, Sep. 2015, pp. 45.1–45.12.
[42] R. von Gioi, J. Jakubowicz, J. Morel, and G. Randall, "LSD: A fast line segment detector with a false detection control," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 4, pp. 722–732, Apr. 2010.

Chi Xu received the PhD degree from the Huazhong University of Science and Technology, Wuhan, China, in 2011. He is a senior postdoctoral research fellow with Bioinformatics Institute (BII), A*STAR, Singapore. His research interests are computer vision, machine vision, and machine learning.

Lilian Zhang received the BS degree from the National University of Defense Technology (NUDT), China, in 2007, and the PhD degree from the Institute of Computer Science, Christian-Albrechts-University of Kiel, Germany, in 2013. He is a lecturer at the National University of Defense Technology, China. His current research is on image processing, structure from motion and vision navigation.

Li Cheng received the PhD degree in computer science from the University of Alberta, Canada. He is a principal investigator with Bioinformatics Institute (BII), A*STAR, Singapore. Prior to joining BII July of 2010, he worked at Statistical Machine Learning group of NICTA, Australia, TTI-Chicago, USA, and University of Alberta, Canada. His research expertise is mainly on computer vision and machine learning.

Reinhard Koch received the graduate degree from the University of Hannover, Germany, and the PhD (Dr.-Ing.) on the topic of 3D scene reconstruction from stereoscopic image sequences in 1996. From 1996-1999 he worked with Prof. Luc van Gool in Leuven, Belgium. Since 1999 he is professor of computer science at the Christian-Albrechts-University Kiel, Germany. His research field is multimedia information processing, with emphasis on 3D scene reconstruction from images and video. His current research interests include underwater imaging and 3D reconstruction, high-quality multi view scene capture, Lightfields and autostereoscopic 3D-TV, and deformable 3D scene analysis from range video sequences.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.