A Stable Algebraic Camera Pose Estimation for Minimal Configurations of 2D/3D Point and Line Correspondences*

Lipu Zhou[1], Jiamin Ye[2], and Michael Kaess[1]

[1] Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA
{lipuz,kaess}@andrew.cmu.edu,
[2] Institute of Engineering Thermophysics, Chinese Academy of Sciences
yejiamin@iet.cn

Abstract. This paper proposes an algebraic solution for the problem of camera pose estimation using the minimal configurations of 2D/3D point and line correspondences, including three point correspondences, two point and one line correspondences, one point and two line correspondences, and three line correspondences. In contrast to the previous works that address these problems in specific geometric ways, this paper shows that the above four cases can be solved in a generic algebraic framework. Specifically, the orientation of the camera is computed from a polynomial equation system of four quadrics, then the translation can be solved from a linear equation system. To make our algorithm stable, the key is the polynomial solver. We significantly improve the numerical stability of the efficient three quadratic equation system solver, E3Q3 [17], with a slight computational cost. The simulation results show that the numerical stability of our algorithm is comparable to the state-of-the-art Perspective-3-Point (*P3P*) algorithm [14], and outperforms the state-of-the-art algorithms of the other three cases. The numerical stability of our algorithm can be further improved by a rough estimation of the rotation matrix, which is generally available in the Localization and Mapping (SLAM) or Visual Odometry (VO) system (such as the pose from the last frame). Besides, this algorithm is applicable to real-time applications.

Keywords: Minimal solution · Pose estimation · SLAM.

1 Introduction

Estimating the pose of a camera from a set of 2D/3D point and correspondences has many applications in computer vision and robotics, such as robot autonomous navigation, Augmented Reality (AR) [35], SLAM [23, 30, 29] and VO [20, 13]. Recent studies [26, 25, 34] show that jointly using point and line features for pose estimation give improved results. As there may exist false matchings in the real scenarios, RANSAC algorithm [5] is generally used to point out

* This work was partially supported by Autel Robotics under award number A020215.

these outliers. The solution of the minimal problem is an essential part of the RANSAC algorithm. This paper focuses on solving the minimal configurations of the 2D/3D point and line correspondences.

There are four minimal configurations for the 2D/3D point and line correspondences. Existing works generally focus on finding a specific solution for each of these minimal configurations. The similarity between the 2D/3D line correspondence and the 2D/3D point correspondence has been used in the literature. In [31], they apply such similarity to solve the least-squares problem of 2D/3D line and point correspondences. Kuang *et al.* [16] propose a minimal solution to estimate the pose of a camera with unknown focal length by points, directions and lines. Direct algebraic solution is generally adopted for pose estimation when the camera intrinsic parameters are unknown, because it is hard to get the 3D information of points and lines in the image plane without the intrinsic parameters. However, it is not clear whether the direct algebraic solution using the basic constraints is comparable to the methods based on well-designed geometric transformations when the intrinsic parameters are known. The specific geometric transformation can eliminate the unknown or even get lower order equation. Such simplification is thought to probably result in a more stable algorithm. This paper shows that directly solving the basic constraints can result in more stable or comparable results. We significantly improve the stability of the efficient three quadrics solver, E3Q3 [17], by selecting a proper unknown elimination order. This can benefit the vision tasks that resort to a three quadrics solver. We compare our algorithm with the previous algorithms by simulations. The results show that our algebraic algorithm is comparable to the state-of-the-art *P3P* algorithm [14], and is superior to the state-of-the-art algorithms of the other three cases in terms of stability. In addition, our algorithm is efficient and can be applied in real-time applications.

2 Related Work

The four minimal problems mentioned above have been solved case by case in the literature.

P3P problem Calculating the camera pose from three 2D/3D point correspondences is known as the *P3P* problem, which have been extensively studied in the literature. The first solution for the *P3P* problem dates back to 1841 presented by Grunert [7]. This algorithm applies the law of cosines to generate three quadratic equations about the lengths between the three 3D points and the camera origin. This is a specific quadratic polynomial system without first order monomials, which can result in a quartic equation with a closed-form solution. Several works [4, 22, 18] follow this formulation with difference in the specific variable elimination approach used to get the quartic equation. Haralick *et al.* [9] present a detailed comparison about these algorithms. More general approaches are also used to explore this specific quadric system. Quan *et al.* [27] apply the Sylvester resultant [19] to solve the quadric system. Gao *et al.* [6] employ Wu-Ritts zero decomposition algorithm [32] to systematically study this equation system, and provide a complete analytical solution. They also give criteria to determine the number of solutions and the number of real physical solutions. The drawback of this series of algorithms is that they need to solve a 3D/3D point alignment problem [1] to get the pose. This increases the computational time. Additionally, due to the finite representation of a digital number, the numerical error accumulated in the extra step may degrade the accuracy. Kneip *et al.* [15] and Masselli *et al.* [21] address this problem by introducing the intermediate coordinate frame to eliminate the variable. Most recently, Ke *et al.* [14] give an algebraic solution to directly compute the camera pose. Due to avoiding extra transformations, this algorithm is efficient and accurate. These approaches make use of the specific property of the *P3P* problem. Therefore, they can not be generalized to the other three minimal problems.

Two point and one line, and one point and two line correspondences These two cases have not been studied thoroughly in the literature. Ramalingam *et al.* [28] give a solution to both problems. They apply the collinearity of the 2D/3D point correspondence, and the coplanarity of the 2D/3D line correspondence to construct constraints on the camera pose. They design two intermediate coordinate systems for each problems to eliminate the variables. Their transformations involve tangent function. This may cause numerical problem. Our algorithm also uses the collinearity and coplanarity constraints. But our algorithm does not require any transformation. This can avoid numerical error propagation, thus can increase accuracy. Besides, their algorithm needs to calculate the Singular-Value Decomposition (SVD) of a relative large matrix. This reduces the speed of the algorithm.

P3L problem Determining the camera pose by three line correspondences is known as the Perspective-3-Line (*P3L*) problem. Several solutions [3, 2, 33] have been proposed for this problem. Chen [2] analyzes the degenerate condition of the *P3L* problem. Xu *et al.* [33] study the number of potential solutions of the *P3L* problem. These methods adopt the similar methodology. They introduce intermediate coordinate systems to make one of the constraints on the rotation matrix automatically satisfied. Two transformations are required by [3], and one transformation is needed by [33, 2]. The simplified problem then can be solved by using the elementary linear algebra and the trigonometric identity. Our algorithm does not need such transformation, thus reduces the numerical error accumulation.

3 Notation and Geometrical Constraints

In this paper, we use italic, boldfaced lowercase and boldfaced uppercase letters to represent scalars, vectors and matrices, respectively. The aim of this paper is to calculate the rotation $\mathbf{R}$ and translation $\mathbf{t}$ between a world frame $O^wX^wY^wZ^w$ and a camera frame $O^cX^cY^cZ^c$ from the minimal configurations of 2D/3D point and line correspondences, including three point correspondences, two point and one line correspondences, one point and two line correspondences, and tree line correspondences. As mentioned above, determining the camera pose from three

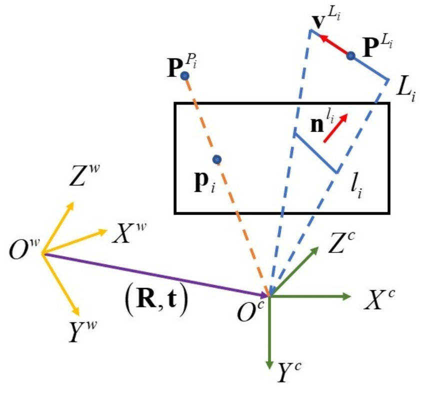


Fig. 1: Geometric constraints from one 2D/3D point correspondence and one 2D/3D line correspondence.

2D/3D point correspondences and three 2D/3D line correspondences are known as the *P3P* and *P3L* problem, respectively. To simplify the notation, we call determining the camera pose from two point and one line correspondences as the Perspective-2-Point-and-1-Line (*P2P1L*) problem, and determining the pose from one point and two line correspondences as the Perspective-1-Point-and-2-Line (*P1P2L*) problem. This section describes the notation and geometrical constraints yielded by one 2D/3D point correspondence and one 2D/3D line correspondence, illustrated in Figure 1.

3.1 2D/3D Point Correspondence

In this paper, we use a quaternion $\mathbf{q} = [w, x, y, z]^T$ [12] to represent the rotation matrix $\mathbf{R}$ as:

$$\begin{bmatrix} w^2+x^2-y^2-z^2 & 2xy-2wz & 2wy+2xz \\ 2wz+2xy & w^2-x^2+y^2-z^2 & 2yz-2wx \\ 2xz-2wy & 2wx+2yz & w^2-x^2-y^2+z^2 \end{bmatrix} \tag{1}$$

Let $\mathbf{P}^{P_i}$ denote a 3D point and $\mathbf{p}_i$ the back-projection ray of its image. To avoid extra transformation, we do not adopt the law of cosines widely used in the *P3P* problem [9]. As $\mathbf{p}_i$ is collinear with $\mathbf{P}^{P_i}$, we have:

$$\mathbf{p}_i \times \left(\mathbf{R}\mathbf{P}_i^P + \mathbf{t}\right) = \mathbf{0} \tag{2}$$

where $\times$ represents the cross product, which can be calculated as:

$$[\mathbf{p}_i]_\times \left(\mathbf{R}\mathbf{P}_i^P + \mathbf{t}\right) = \mathbf{0} \tag{3}$$

where $[\mathbf{p}_i]_\times$ is a skew-symmetric matrix having the form:

$$[\mathbf{p}_i]_\times = \begin{bmatrix} p_{i1} \\ p_{i2} \\ p_{i3} \end{bmatrix}_\times = \begin{bmatrix} 0 & -p_{i3} & p_{i2} \\ p_{i3} & 0 & -p_{i1} \\ -p_{i2} & p_{i1} & 0 \end{bmatrix} \tag{4}$$

Substituting (1) and (4) into (3), we have the following three quadric equations:

$$\begin{aligned}
&c^{p_i}_{1,1}x^2 + c^{p_i}_{1,2}y^2 + c^{p_i}_{1,3}z^2 + c^{p_i}_{1,4}w^2 + c^{p_i}_{1,5}xy + c^{p_i}_{1,6}xz \\
&+c^{p_i}_{1,7}xw + c^{p_i}_{1,8}yz + c^{p_i}_{1,9}yw + c^{p_i}_{1,10}zw - p_{i3}t_2 + p_{i2}t_3 = 0, \\
&c^{p_i}_{2,1}x^2 + c^{p_i}_{2,2}y^2 + c^{p_i}_{2,3}z^2 + c^{p_i}_{2,4}w^2 + c^{p_i}_{2,5}xy + c^{p_i}_{2,6}xz \\
&+c^{p_i}_{2,7}xw + c^{p_i}_{2,8}yz + c^{p_i}_{2,9}yw + c^{p_i}_{2,10}zw + p_{i3}t_1 - p_{i1}t_3 = 0, \\
&c^{p_i}_{3,1}x^2 + c^{p_i}_{3,2}y^2 + c^{p_i}_{3,3}z^2 + c^{p_i}_{3,4}w^2 + c^{p_i}_{3,5}xy + c^{p_i}_{3,6}xz \\
&+c^{p_i}_{3,7}xw + c^{p_i}_{3,8}yz + c^{p_i}_{3,9}yw + c^{p_i}_{3,10}zw - p_{i2}t_1 + p_{i1}t_2 = 0
\end{aligned} \tag{5}$$

where t_i $(i = 1, 2, 3)$ are the three components of $\mathbf{t}$. Define

$$\mathbf{r} = \left[x^2, y^2, z^2, w^2, xy, xz, xw, yz, yw, zw\right]^T \tag{6}$$

We can simplify the j-th equation of the i-th point correspondences in (5) as

$$\mathbf{c}^{p_i}_j \cdot \mathbf{r} + \mathbf{n}^{p_i}_j \cdot \mathbf{t} = 0, \quad j = 1, 2, 3 \tag{7}$$

where $\cdot$ represents the dot product, $\mathbf{c}^{p_i}_j$ is a 10×1 vector and $\mathbf{n}^{p_i}_j$ is a 3×1 vector. As $[\mathbf{p}_i]_\times$ is a rank-2 matrix, this equation system only provides 2 linear independent constraints.

3.2 2D/3D Line Correspondence

Let L_i and l_i represent a 3D line and its corresponding 2D line, respectively. Denote the direction of L_i as $\mathbf{v}^{L_i}$ and a 3D point on L_i as $\mathbf{P}^{L_i}$. The back-projection of l_i is a plane π_i that passes through the origin of the camera coordinate system. Denote the normal of π_i as $\mathbf{n}^{l_i}$. Since L_i should be on π_i, we get the following constraints:

$$\begin{aligned}
&\mathbf{n}^{l_i} \cdot \mathbf{R}\mathbf{v}^{L_i} = 0 \\
&\mathbf{n}^{l_i} \cdot \left(\mathbf{R}\mathbf{P}^{L_i} + \mathbf{t}\right) = 0
\end{aligned} \tag{8}$$

Substituting (1) into (8) and using the definition of (6), we obtain two quadrics:

$$\begin{aligned}
&\mathbf{c}^{l_i}_1 \cdot \mathbf{r} = 0, \\
&\mathbf{c}^{l_i}_2 \cdot \mathbf{r} + \mathbf{n}^{l_i} \cdot \mathbf{t} = 0
\end{aligned} \tag{9}$$

4 Minimal Solution

4.1 P3P

We give a new approach for the *P3P* problem. As we seek to give a generic framework for all the minimal configurations of 2D/3D point and line correspondences, we avoid adopting the specific property of the *P3P* problem used by the previous works [9, 14]. As mentioned above, each 2D/3D correspondence provides two constraints. Without loss of generality, we pick up the first two equations of (3) from the first two correspondences, and the first and the last equations from the third correspondence. According to (7), we have the following quadratic equation system:

$$\begin{array}{l} \mathbf{c}_1^{p_1} \cdot \mathbf{r} + \mathbf{n}_1^{p_1} \cdot \mathbf{t} = \mathbf{c}_1^{p_2} \cdot \mathbf{r} + \mathbf{n}_1^{p_2} \cdot \mathbf{t} = \mathbf{c}_1^{p_3} \cdot \mathbf{r} + \mathbf{n}_1^{p_3} \cdot \mathbf{t} = 0 \\ \mathbf{c}_2^{p_1} \cdot \mathbf{r} + \mathbf{n}_2^{p_1} \cdot \mathbf{t} = \mathbf{c}_2^{p_2} \cdot \mathbf{r} + \mathbf{n}_2^{p_2} \cdot \mathbf{t} = \mathbf{c}_3^{p_3} \cdot \mathbf{r} + \mathbf{n}_3^{p_3} \cdot \mathbf{t} = 0 \end{array} \tag{10}$$

Divide this equation system into two parts, so that the first part contains the first 3 equations and the second part contains the remaining ones. Then we have:

$$\begin{array}{l} \mathbf{C}_1\mathbf{r} + \mathbf{N}_1\mathbf{t} = \mathbf{0}, \quad \mathbf{C}_2\mathbf{r} + \mathbf{N}_2\mathbf{t} = \mathbf{0} \\ \mathbf{C}_1 = \left[\mathbf{c}_1^{p_1}, \mathbf{c}_1^{p_2}, \mathbf{c}_1^{p_3}\right]^T, \ \mathbf{N}_1 = \left[\mathbf{n}_1^{p_1}, \mathbf{n}_1^{p_2}, \mathbf{n}_1^{p_3}\right]^T \\ \mathbf{C}_2 = \left[\mathbf{c}_2^{p_1}, \mathbf{c}_2^{p_2}, \mathbf{c}_3^{p_3}\right]^T, \ \mathbf{N}_2 = \left[\mathbf{n}_2^{p_1}, \mathbf{n}_2^{p_2}, \mathbf{n}_3^{p_3}\right]^T \end{array} \tag{11}$$

where $\mathbf{C}_1$ and $\mathbf{C}_2$ are 3×10 matrices, $\mathbf{N}_1$ and $\mathbf{N}_2$ are 3×3 matrices. Using $\mathbf{C}_2\mathbf{r} + \mathbf{N}_2\mathbf{t} = \mathbf{0}$ in (11), we get a closed-form solution for $\mathbf{t}$ as

$$\mathbf{t} = -(\mathbf{N}_2)^{-1}\mathbf{C}_2\mathbf{r} \tag{12}$$

Other choices are also valid, if the coefficient matrix of $\mathbf{t}$ is invertible. Replace $\mathbf{t}$ in $\mathbf{C}_1\mathbf{r} + \mathbf{N}_1\mathbf{t} = \mathbf{0}$ in (11). Together with the norm one constraint of $\mathbf{q}$, we get four quadratic equations for the four elements in $\mathbf{q}$ as:

$$\begin{array}{l} \mathbf{A}\mathbf{r} = 0 \\ w^2 + x^2 + y^2 + z^2 = 1 \end{array} \tag{13}$$

where

$$\mathbf{A} = \mathbf{C}_1 - \mathbf{N}_1(\mathbf{N}_2)^{-1}\mathbf{C}_2 \tag{14}$$

We will show that the other three minimal cases also have the same quadric forms of $\mathbf{q}$. Therefore, we will give the solution to it at the end of this section.

4.2 P2P1L

For the two 2D/3D point correspondences, we chose the first two equations of (7). Together with constraints in (9) from the line correspondence, we can obtain the following equation system:

$$\begin{array}{l} \mathbf{c}_1^{l_1} \cdot \mathbf{r} = \mathbf{c}_1^{p_1} \cdot \mathbf{r} + \mathbf{n}_1^{p_1} \cdot \mathbf{t} = \mathbf{c}_1^{p_2} \cdot \mathbf{r} + \mathbf{n}_1^{p_2} \cdot \mathbf{t} = 0 \\ \mathbf{c}_2^{l_1} \cdot \mathbf{r} + \mathbf{n}^{l_1} \cdot \mathbf{t} = \mathbf{c}_2^{p_1} \cdot \mathbf{r} + \mathbf{n}_2^{p_1} \cdot \mathbf{t} = \mathbf{c}_2^{p_2} \cdot \mathbf{r} + \mathbf{n}_2^{p_2} \cdot \mathbf{t} = 0 \end{array} \tag{15}$$

There are 5 equations in $\mathbf{t}$. Without loss of generality, we choose one equation involving $\mathbf{t}$ from each correspondence to solve $\mathbf{t}$. To simplify the notation, we use the same notation as (11). Rearranging (15), we have

$$\begin{array}{l} \mathbf{c}_1^{l_1} \cdot \mathbf{r} = 0, \quad \mathbf{C}_1\mathbf{r} + \mathbf{N}_1\mathbf{t} = 0, \quad \mathbf{C}_2\mathbf{r} + \mathbf{N}_2\mathbf{t} = 0 \\ \mathbf{C}_1 = \left[\mathbf{c}_1^{p_1}, \mathbf{c}_1^{p_2}\right]^T, \ \mathbf{N}_1 = \left[\mathbf{n}_1^{p_1}, \mathbf{n}_1^{p_2}\right]^T \\ \mathbf{C}_2 = \left[\mathbf{c}_2^{l_1}, \mathbf{c}_2^{p_1}, \mathbf{c}_2^{p_2}\right]^T, \ \mathbf{N}_2 = \left[\mathbf{n}^{l_1}, \mathbf{n}_2^{p_1}, \mathbf{n}_2^{p_2}\right]^T \end{array} \tag{16}$$

where $\mathbf{C}_1$ is a 2×10 matrix, and $\mathbf{N}_1$ is a 2× 3 matrix. Using $\mathbf{C}_2\mathbf{r} + \mathbf{N}_2\mathbf{t} = \mathbf{0}$, we can obtain a closed-form solution for $\mathbf{t}$ as (12). Substituting the expression of $\mathbf{t}$ into $\mathbf{C}_1\mathbf{r} + \mathbf{N}_1\mathbf{t} = \mathbf{0}$, we get a quadric equation system as (13), with

$$\mathbf{A} = \left[\mathbf{c}_1^{l_1}, \left(\mathbf{C}_1 - \mathbf{N}_1(\mathbf{N}_2)^{-1}\mathbf{C}_2\right)^T\right]^T \tag{17}$$

4.3 P1P2L

Given one 2D/3D point and two 2D/3D line correspondences, according to (7) and (9), we can have the following equations:

$$\begin{array}{l} \mathbf{c}_1^{l_1} \cdot \mathbf{r} = \mathbf{c}_1^{l_2} \cdot \mathbf{r} = \mathbf{c}_1^{p_1} \cdot \mathbf{r} + \mathbf{n}_1^{p_1} \cdot \mathbf{t} = 0 \\ \mathbf{c}_2^{l_1} \cdot \mathbf{r} + \mathbf{n}^{l_1} \cdot \mathbf{t} = \mathbf{c}_2^{l_2} \cdot \mathbf{r} + \mathbf{n}^{l_2} \cdot \mathbf{t} = \mathbf{c}_2^{p_1} \cdot \mathbf{r} + \mathbf{n}_2^{p_1} \cdot \mathbf{t} = 0 \end{array} \tag{18}$$

Here we use the first two equations of (7). Other choices are also valid. Each line correspondence provides one constraint on $\mathbf{t}$. Together with another constraint from the point correspondence, we can obtain three linear equations with respect to $\mathbf{t}$. Rearranging the equations, we can have:

$$\begin{array}{l} \mathbf{c}_1^{l_1} \cdot \mathbf{r} = \mathbf{c}_1^{l_2} \cdot \mathbf{r} = \mathbf{c}_1^{p_1} \cdot \mathbf{r} + \mathbf{n}_1^{p_1} \cdot \mathbf{t} = 0 \\ \mathbf{C}_2\mathbf{r} + \mathbf{N}_2\mathbf{t} = 0 \\ \mathbf{C}_2 = \left[\mathbf{c}_2^{l_1}, \mathbf{c}_2^{l_2}, \mathbf{c}_2^{p_1}\right]^T, \ \mathbf{N}_2 = \left[\mathbf{n}^{l_1}, \mathbf{n}^{l_2}, \mathbf{n}_2^{p_1}\right]^T \end{array} \tag{19}$$

Using (12), we can get $\mathbf{t}$. Substituting (12) into $\mathbf{c}_1^{p_1} \cdot \mathbf{r} + \mathbf{n}_1^{p_1} \cdot \mathbf{t} = 0$, we can get a quadratic equation system the same as (13) with

$$\mathbf{A} = \left[\mathbf{c}_1^{l_1}, \mathbf{c}_1^{l_2}, \mathbf{c}_1^{p_1} - \left((\mathbf{N}_2)^{-1}\mathbf{C}_2\right)^T \mathbf{n}_1^{p_1}\right]^T \tag{20}$$

4.4 P3L

Given three line correspondences, we can have the following quadratic equation system according to (9):

$$\begin{array}{l} \mathbf{c}_1^{l_1} \cdot \mathbf{r} = \mathbf{c}_1^{l_2} \cdot \mathbf{r} = \mathbf{c}_1^{l_3} \cdot \mathbf{r} = 0 \\ \mathbf{c}_2^{l_1} \cdot \mathbf{r} + \mathbf{n}^{l_1} \cdot \mathbf{t} = \mathbf{c}_2^{l_2} \cdot \mathbf{r} + \mathbf{n}^{l_2} \cdot \mathbf{t} = \mathbf{c}_2^{l_3} \cdot \mathbf{r} + \mathbf{n}^{l_3} \cdot \mathbf{t} = 0 \end{array} \tag{21}$$

It is clear that the first three quadrics only involving the quaternion $\mathbf{q}$. Combining with the norm one constraint of $\mathbf{q}$, we have a form the same as (13) with

$$\mathbf{A} = \left[\mathbf{c}_1^{l_1}, \mathbf{c}_1^{l_2}, \mathbf{c}_1^{l_3}\right]^T \tag{22}$$

Besides, $\mathbf{t}$ can be computed from the last three equations of (21) using (12) with

$$\mathbf{C}_2 = \left[\mathbf{c}_2^{l_1}, \mathbf{c}_2^{l_2}, \mathbf{c}_2^{l_3}\right]^T, \ \mathbf{N}_2 = \left[\mathbf{n}^{l_1}, \mathbf{n}^{l_2}, \mathbf{n}^{l_3}\right]^T \tag{23}$$

4.5 Solve the Rotation Matrix

As mentioned above, in all of the four minimal configurations, $\mathbf{R}$ can be obtained by solving a quadratic equation system with the form (13). It seems that there are 16 solutions according to the Bézout's Theorem [19]. However, as (1) only includes the degree 2 monomials, the sign of the unknowns does not impact on the value of $\mathbf{R}$. Thus, there are at most 8 real solutions for $\mathbf{R}$. In this section, we show how to solve this quadratic equation system.

Assume w is not equal to 0. Let

$$x = aw, \quad y = bw, \quad z = cw \tag{24}$$

Divide both side of $\mathbf{Ar} = \mathbf{0}$ in (13) by w. We can have

$$\mathbf{a}_i \left[a^2, b^2, c^2, ab, ac, bc, a, b, c, 1\right]^T = 0, \quad i = 1, 2, 3 \tag{25}$$

where $\mathbf{a}_i$ is the i-th row of $\mathbf{A}$ with coefficient permutation. It is easy to find that $[a, b, c]^T$ is the intersection of three quadrics. This can be solved by the E3Q3 algorithm [17].

For completeness, we briefly introduce the E3Q3 algorithm. By regarding a as a constant, we get three equations about b and c. Dividing the six monomials of b and c into two parts, *i.e.*, $\{b^2, c^2, bc\}$ and $\{b, c, 1\}$, and rearranging the equations, we can obtain:

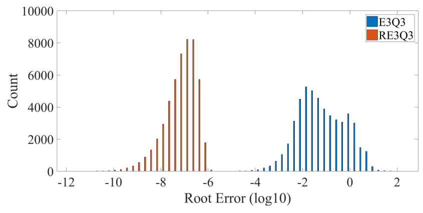
$$\mathbf{H}\begin{bmatrix} b^2 \\ c^2 \\ bc \end{bmatrix} = \begin{bmatrix} p_{11}(x), p_{12}(x), p_{13}(x) \\ p_{21}(x), p_{22}(x), p_{23}(x) \\ p_{31}(x), p_{32}(x), p_{33}(x) \end{bmatrix} \begin{bmatrix} b \\ c \\ 1 \end{bmatrix} \tag{26}$$

Assume $\mathbf{H}$ is invertible. Multiplying $\mathbf{H}^{-1}$ to both side of (26), we get the relationship between $\{b^2, c^2, bc\}$ and $\{b, c, 1\}$. Using this relationship and the identities $(b^2)c = (bc)b$, $(bc)c = (c^2)b$, and $(bc)(bc) = (b^2)(c^2)$, we can get a homogeneous linear system whose coefficients $\mathbf{M}(a)$ are polynomials in a. According to the linear algebra, the homogeneous linear system has a non-trivial solution, if and only if the determinant of $\mathbf{M}(a)$ is zero. This results in a degree 8 polynomial in a. Solve this for a, then back-substitute a into the linear system to get b and c.

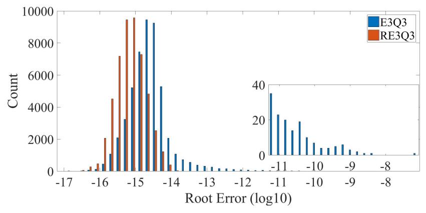
Given a, b and c, w^2 can be obtained from the norm one constraint of the quaternion by $w^2 = 1/\left(a^2 + b^2 + c^2 + 1\right)$. We do not need to calculate w, as all the monomials in (1) has degree two. Substituting (24) into (1) and using w^2, we can easily obtain $\mathbf{R}$.

There are two assumptions for computing $\mathbf{R}$. The first is $\mathbf{H}$ is invertible, and the second is w is not zero. Therefore, singularity occurs when the assumptions do not satisfy. We address both singularities in the following two sections.

Robust E3Q3 (RE3Q3) Kukelova *et al.* [17] find that there are 8 degenerate configurations when $\mathbf{H}$ is rank deficient, and they give the solution for each of them. However, this method is hard to handle the situation where $\mathbf{H}$ approximates singularity, which will significantly degrade the performance of the algorithm as shown in Figure 2a.



(a) Degenerate situation



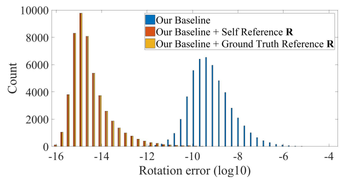
(b) General situation

Fig. 2: Compare RE3Q3 with E3Q3 in degenerate situation (a) and in general situation (b). We randomly generate the coefficients of (25) except for the constants. Then we randomly generate a solution, and substitute it into (25) to calculate the constants. For the degenerate cases, we get $\mathbf{H}$ in (26) and randomly set the smallest singular value within $\left(0, 10^{-6}\right)$. It is clear that RE3Q3 outperforms E3Q3. We run the algorithm 50,000 times.

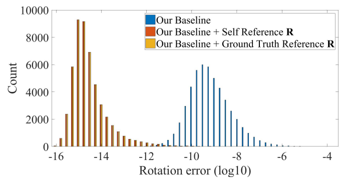
As we can treat any of a, b and c as a constant, and the other two as unknowns in (25), there actually exist three choices for $\mathbf{H}$. Let $\mathbf{H}_a$, $\mathbf{H}_b$ and $\mathbf{H}_c$ represent the coefficient matrices obtained by choosing a, b and c as a constant, respectively. If the coefficient matrix of the second order monomials in (25) is nondegenerate, it is probable that when $\mathbf{H}_a$ is ill-conditioned, but $\mathbf{H}_b$ or $\mathbf{H}_c$ is still in good condition. We try to find the one with the best condition. As the condition number of a matrix describes to what extent a matrix approaches singularity. The larger the condition number is, the closer the matrix approaches singularity. We calculate the condition number of $\mathbf{H}_a$, $\mathbf{H}_b$ and $\mathbf{H}_c$, and choose the one with the minimal condition number to replace $\mathbf{H}$ in (25). This just needs to interchange the coefficient of (25), and do not need to implement different algorithm for different choice. Thus, it is efficient. We call this approach Robust E3Q3 (RE3Q3). Figure 2a shows that RE3Q3 is much more stable than the original E3Q3 algorithm [17] in the degenerate configuration. Besides, in the general situation, RE3Q3 still improves the stability of E3Q3, as shown in Figure 2b.

Reference Rotation When w in $\mathbf{q}$ is small, according to (24), a, b and c are probably greater than 1. Thus, they may amplify the estimation error of w when we compute x, y and z. This effect increases, when w gets smaller. The performance of our algorithm will degrade if w is a very tiny value, as shown in Figure 3, where $w \in (0, 10^{-6})$. If we have a reference rotation represented as a quaternion $\mathbf{q}_{ref}$, which gives a rough estimation of the rotation, we can easily solve this problem. Given a $\mathbf{q}_{ref}$, we can exchange w with the element that has the maximum absolute value in $\mathbf{q}_{ref}$ to get $\mathbf{q}'$. This makes a, b and c all smaller than 1. Exchanging the element of $\mathbf{q}'$ equals to permute the coefficients in (25), and the computational cost is negligible. When $\mathbf{q}'$ is calculated, we can get the original $\mathbf{q}$ by applying the exchange again.

In the application, we can generally have a $\mathbf{q}_{ref}$. For example, in the SLAM system, camera pose is sequentially estimated. Therefore, the last rotation can



(a) *P3L* rotation error



(b) *P3P* rotation error

Fig. 3: Rotation error for *P3L* and *P3P* when $w \in (0, 10^{-6})$, which is degenerate for the baseline algorithm. We generate 3 extra points to find the most accurate rotation $\mathbf{R}_{self}$. Then we use $\mathbf{R}_{self}$ as a reference rotation to calculate the rotation again (labeled as Self Reference $\mathbf{R}$). We run the algorithm 50,000 times. This method gives almost the same result as using the ground truth as the reference matrix. It is clear that our algorithm can provide valid reference rotation even in the degenerate situation.

be used as the reference rotation. In addition, the minimal solution is generally used in the RANSAC algorithm [5], the current optimal rotation estimation can be a reference rotation. One question is that whether our algorithm can generate a valid reference matrix in the degenerate configuration. To verify this, we run our *P3L* and *P3P* algorithm on 50,000 randomly generated degenerate configurations where $w \in (0, 10^{-6})$. Three additional points are generated to select the most accurate rotation, denoted as $\mathbf{R}_{self}$. $\mathbf{R}_{self}$ is then used as the reference rotation. We also use the ground truth rotation $\mathbf{R}_{gt}$ as the reference rotation. The experimental results in Figure 3 show that $\mathbf{R}_{self}$ and $\mathbf{R}_{gt}$ gives almost the small results. As we only use the relative order of $\mathbf{q}_{ref}$, $\mathbf{q}_{ref}$can be rather rough. This makes our algorithm stable even in the degenerate case.

4.6 Algorithm Summary

As mentioned above, the rotation matrix $\mathbf{R}$ of all the four minimal configurations can be obtained by solving a quadric equation system having the form of (13). Given $\mathbf{R}$, the translation $\mathbf{t}$ can be calculated from a linear system (12). One 2D/3D point correspondence gives three equations in (3). We use one of them for $\mathbf{R}$ estimation. Given $\mathbf{R}$, we find that using the remaining 2 equations for the estimation of $\mathbf{t}$ is more accurate in our simulations. Our algorithm is summarized in Algorithm 1.

5 Simulation Results

As the previous works [14, 9, 6, 15, 28, 3, 2] , we compare our algorithm with the state-of-the-art algorithms by simulations. We can evaluate the algorithms by a large number of configurations in the simulation. As the same input will generate the same result, the simulation results will unfold the performance of different algorithms in real applications.

Given the estimation $(\hat{\mathbf{R}}, \hat{\mathbf{t}})$ and the ground truth $(\mathbf{R}_{gt}, \mathbf{t}_{gt})$, the estimation error of $\hat{\mathbf{R}}$ is measured by the absolute angle of the axis-angle representation of

Algorithm 1: Solve the minimal problems of 2D/3D point and line correspondences

```
1. Calculate the coefficient matrix $\mathbf{A}$ of the quadric equations, i.e., using (14) for P3P, (17) for P2P1L, (20) for P1P2L, (22) for P3L.
2. If $\mathbf{q}_{ref}$ is available, find the max absolute element in $\mathbf{q}_{ref}$. Exchange it with $w$, and rearrange the coefficients accordingly.
3. Solve (25) by the RE3Q3 algorithm. If $\mathbf{q}_{ref}$ is used, rearrange the solution accordingly.
4. Use the norm one constraint of the quaternion and the definition (1) to get $\mathbf{R}$.
5. Solve $\mathbf{t}$ from the linear system including all the remaining constraints on $\mathbf{t}$.
```

$\hat{\mathbf{R}}\mathbf{R}_{gt}^{-1}$ as [17], and the estimation error of $\hat{\mathbf{t}}$ is measured by $\left\|\hat{\mathbf{t}} - \mathbf{t}_{gt}\right\|_2 / \|\mathbf{t}_{gt}\|_2$. We randomly generate the rotation matrix by Euler angle. The position of the camera is within a cube $[-5, 5]^3$. The virtual camera has image size 640×480 and focal length 800. A line is generated by two random points. The depth of the 3D point is within $[2, 8]$. We also study the behavior of our algorithm with or without a reference rotation. As shown in Figure 3, the rotation matrix calculated by our algorithm is as valid as the ground truth. Thus, we use the ground truth rotation matrix as the reference. Our algorithm without a reference rotation is labeled as the baseline in the following simulations. The following results are obtained from 50,000 trials. Table 1 lists the mean, standard deviation, median, and maximal estimation errors. It shows that our baseline algorithm is comparable to the state-of-the-art *P3P* algorithm [14], and outperforms the previous algorithms of the other three problems. Besides, a reference rotation can further increase the performance.

5.1 Results of P3P Problem

We compare our algorithm with Ke's algorithm [14], Kneip's algorithm [15] and Gao's algorithm [6]. For fairness, we do not apply root polishing for Ke's algorithm. As all of these algorithms have publicly available c++ implementation, we also implement our algorithm in c++. We use Hartley's Sturm sequences [10] implementation to solve the eighth degree polynomial equation. The relative error is set to 10^{-14} as [17].

The histograms of rotation and translation errors are shown in Figure 4. Table 1 lists the quantitative results of different algorithms. It is clear that the reference rotation can increase the stability of our algorithm. Ke's algorithm is better than our algorithm in rotation. Our algorithm gives a slightly better result in translation, as we use more equations for the translation estimation, and solve it in the least-squares manner. Our algorithm outperforms other *P3P* algorithms. This is because both Ke's algorithm and our algorithm avoid the unnecessary intermediate transformation, therefore reduce the numerical error accumulation.

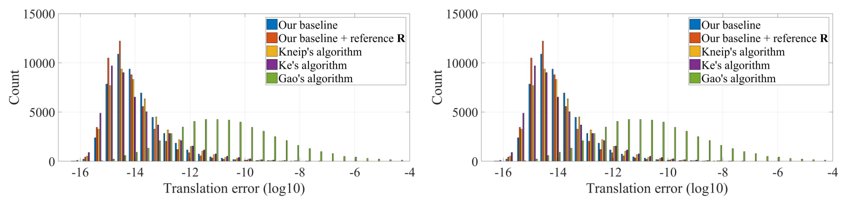


Fig. 4: Histograms of rotation errors (left) and translation errors (right) for *P3P* algorithms.

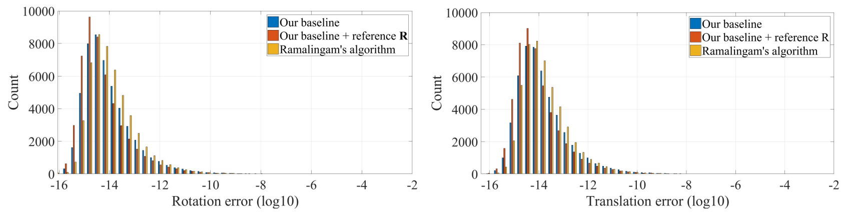


Fig. 5: Histograms of rotation errors (left) and translation errors (right) for different *P2P1L* algorithms.

5.2 Results of P2P1L and P1P2L problem

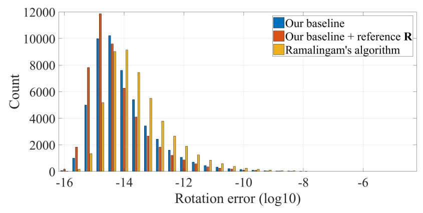
We compare our algorithm with Ramalingam's algorithm [28] for the *P2P1L* and *P1P2L* problems. The error histograms are shown in Figures 5 and 6. Table 1 gives the statistics of the estimation error. It is obvious that our algorithm outperforms Ramalingam's algorithm in terms of accuracy. Ramalingam's algorithm requires two intermediate transformations. Numerical errors accumulated in these transformations potentially decrease the accuracy. Besides, their transformations involve tangent function, which may case numerical issue.

5.3 Results of P3L problem

As mentioned in the section 2, The *P3L* algorithms [33, 3, 2] are similar. We compare our algorithm with the latest *P3L* algorithm [33]. Figure 7 shows the results of different algorithms. In the area of very small rotation error (the first two bins in Figure 7a), Xu's algorithm has a higher probability than our algorithm. However, as shown by the sub-windows in Figure 7, Xu's algorithm has a very long tail. Besides, the statistics listed in Table 1 also verify that our algorithm is more stable than Xu's algorithm. The maximal rotation and translation errors of Xu's algorithm approximate 0.1 rad and 0.8, respectively. Our maximal rotation and translation errors are much smaller than theirs.

5.4 Computational time

Our algorithm is implemented in C++ using Eigen linear algebra library [8] for the *P3P* problem. We use the OpenCV's [24] implementation of Ke's algorithm [14]. For the other minimal problems, we compare the time using the Matlab implementations. Here, we only list our running time with reference rotation, as the running of our baseline algorithm is very similar to our algorithm



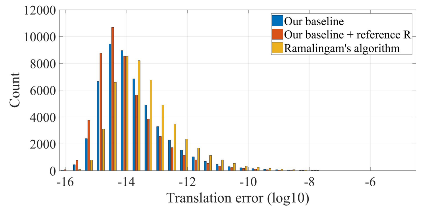
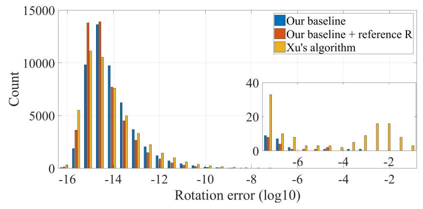


Fig. 6: Histograms of rotation errors (left) and translation errors (right) for *P1P2L* algorithms.



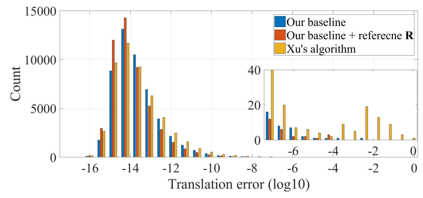


Fig. 7: Histograms of rotation errors (left) and translation errors (right) for *P3L* algorithms.

with reference rotation. As all the four minimal problems are solved in a uniform framework, the computational time of the other three cases in c++ should be similar to the time of the *P3P* problem. In the application, we can use the reference rotation to reduce the computational time of the translation. For fairness, we compute all the eight solutions of translation here. All the results are obtained by 50,000 trials on a laptop with a 2.9 GHZ intel core i7 CPU.

Table 2 gives the results. Compared to E3Q3, RE3Q3 slightly increases the running time. It is not surprising that our algorithm is slower that Ke's algorithm [14], as we need to solve an eighth degree equation for the rotation and eight linear equation systems for the translation, but they only need to solve a quartic equation for the rotation and four linear systems for the translation. For the *P3L* problem, Xu's algorithm [33] is slightly faster than ours. This is because they use a transformation to directly eliminate one of the rotation variable. Ramalingam's algorithm [28] is slower than our algorithm. This is because their algorithm needs to compute SVD of a 6×8 matrix for the *P2P1L* problem, and 6×9 matrix for the *P1P2L* problem. The SVD is time-consuming. Although our algorithm is slower than Ke's algorithm, it is still efficient for a real-time application. Minimal solution is generally used in the RANSAC algorithm [5]. Suppose the ratio of the outlier is 0.5. To ensure with a probability, such as 0.99, that at least one of the random minimal samples is without outliers, we need at least 35 trials [11]. This will be finished within 0.8 ms.

6 Conclusion

In this paper, we propose a algebraic algorithm for the four minimal configurations of 2D/3D point and line correspondences. This is useful for many robotics

Table 1: Mean, standard deviation (STD), median, and max of the pose errors. The best result is highlighted by the boldface.

	Algorithm	Rotation				Translation			
		Mean	STD	Median	Max	Mean	STD	Median	Max
P3P	Our baseline	1.2e-09	2.1e-07	5.4e-15	4.6e-05	1.2e-09	2.0e-07	7.4e-15	4.4e-05
	Baseline+ref. **R**	6.6e-10	6.9e-08	3.2e-15	1.3e-05	**1.2e-09**	**1.6e-07**	**4.5e-15**	**3.4e-05**
	Kneip	1.6e-08	3.2e-06	6.3e-15	7.1e-04	2.2e-08	4.3e-06	8.3e-15	9.5e-04
	Ke	**1.4e-10**	**2.0e-08**	**2.5e-15**	**4.4e-06**	7.9e-09	1.1e-06	5.5e-15	1.7e-04
	Gao	1.9e-04	1.8e-02	7.7e-12	2.7	4.1e-04	3.4e-02	1.5e-11	5.4
P2P1L	Our baseline	2.2e-08	4.3e-06	5.5e-15	9.5e-04	3.0e-08	6.0e-06	9.0e-15	1.3e-03
	Baseline+ref. **R**	**7.9e-09**	**1.2e-06**	**3.3e-15**	**2.6e-04**	**9.0e-09**	**1.4e-06**	**5.6e-15**	**3.1e-04**
	Ramalingam	1.6e-07	3.4e-05	7.8e-15	7.6e-03	8.7e-08	1.9e-05	1.1e-14	4.2e-03
P1P2L	Our baseline	9.1e-10	1.2e-07	5.6e-15	2.6e-05	1.2e-09	1.3e-07	1.0e-14	2.6e-05
	Baseline+ref. **R**	**3.0e-10**	**2.9e-08**	**3.3e-15**	**5.2e-06**	**4.3e-10**	**4.7e-08**	**6.1e-15**	**9.5e-06**
	Ramalingam	1.9e-09	1.9e-07	1.6e-14	3.6e-05	2.3e-09	2.1e-07	2.6e-14	3.1e-05
P3L	Our baseline	2.0e-08	3.6e-06	4.6e-15	8.0e-04	6.6e-08	1.4e-05	1.3e-14	3.1e-03
	Baseline+ref. **R**	**6.6e-10**	**7.8e-08**	**2.5e-15**	**1.2e-05**	**3.0e-09**	**3.5e-07**	**6.9e-15**	**5.1e-05**
	Xu	1.1e-05	6.6e-04	3.4e-15	9.1e-02	3.9e-05	3.8e-03	1.2e-14	8.0e-01

Table 2: Computational time comparison. RE3Q3, E3Q3, and *P3P* algorithms are tested by c++. Others are tested by Matlab.

	Algorithm	Time (μs)
Solver	RE3Q3	11
	E3Q3	8.3
P3P	Our baseline + reference **R**	21
	Ke's algorithm	3.1
P2P1L	Our baseline + reference **R**	290
	Ramalingam's algorithm	429
P1P2L	Our baseline + reference **R**	279
	Ramalingam's algorithm	439
P3L	Our baseline + reference **R**	261
	Xu's algorithm	202

and computer vision applications. Our algorithm directly uses the collinearity and coplanarity constraints to construct the equation system, and does not need any intermediate transformation. This can avoid numerical error accumulation. We increase the stability of our algorithm by a reference matrix which is generally available in real applications. We present the RE3Q3 algorithm which significantly increases the stability of the E3Q3 algorithm [17]. The simulation results show that our baseline algorithm is comparable to the state-of-the-art *P3P* algorithm [14], and outperforms the stat-of-the-art algorithms of the other three minimal cases. A reference rotation matrix, which is generally available in the SLAM or VO system, can further increase the numerical stability of our algorithm. Additionally, our algorithm is efficient for the real-time application.

References

1. Arun, K.S., Huang, T.S., Blostein, S.D.: Least-squares fitting of two 3-d point sets. IEEE Transactions on pattern analysis and machine intelligence (5), 698–700 (1987)
2. Chen, H.H.: Pose determination from line-to-plane correspondences: existence condition and closed-form solutions. IEEE Transactions on Pattern Analysis and Machine Intelligence **13**(6), 530–541 (Jun 1991). https://doi.org/10.1109/34.87340
3. Dhome, M., Richetin, M., Lapreste, J.T., Rives, G.: Determination of the attitude of 3d objects from a single perspective view. IEEE Transactions on Pattern Analysis and Machine Intelligence **11**(12), 1265–1278 (Dec 1989). https://doi.org/10.1109/34.41365
4. Finsterwalder, S., Scheufele, W.: Das rückwärtseinschneiden im raum. Verlag d. Bayer. Akad. d. Wiss. (1903)
5. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM **24**(6), 381–395 (1981)
6. Gao, X.S., Hou, X.R., Tang, J., Cheng, H.F.: Complete solution classification for the perspective-three-point problem. IEEE transactions on pattern analysis and machine intelligence **25**(8), 930–943 (2003)
7. Grunert, J.A.: Das pothenotische problem in erweiterter gestalt nebst über seine anwendungen in der geodäsie. Grunerts archiv für mathematik und physik **1**(238-248), 1 (1841)
8. Guennebaud, G., Jacob, B., et al.: Eigen v3. http://eigen.tuxfamily.org (2010)
9. Haralick, B.M., Lee, C.N., Ottenberg, K., Nölle, M.: Review and analysis of solutions of the three point perspective pose estimation problem. International journal of computer vision **13**(3), 331–356 (1994)
10. Hartley, R., Li, H.: An efficient hidden variable approach to minimal-case camera motion estimation. IEEE transactions on pattern analysis and machine intelligence **34**(12), 2303–2314 (2012)
11. Hartley, R., Zisserman, A.: Multiple view geometry in computer vision. Cambridge university press (2003)
12. Horn, B.K.: Closed-form solution of absolute orientation using unit quaternions. JOSA A **4**(4), 629–642 (1987)
13. Jose Tarrio, J., Pedre, S.: Realtime edge-based visual odometry for a monocular camera. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 702–710 (2015)
14. Ke, T., Roumeliotis, S.I.: An efficient algebraic solution to the perspective-three-point problem. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4618–4626 (July 2017). https://doi.org/10.1109/CVPR.2017.491
15. Kneip, L., Scaramuzza, D., Siegwart, R.: A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. pp. 2969–2976. IEEE (2011)
16. Kuang, Y., Åström, K.: Pose estimation with unknown focal length using points, directions and lines. In: ICCV. pp. 529–536 (2013)
17. Kukelova, Z., Heller, J., Fitzgibbon, A.: Efficient intersection of three quadrics and applications in computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1799–1808 (2016)

18. Linnainmaa, S., Harwood, D., Davis, L.S.: Pose determination of a three-dimensional object using triangle pairs. IEEE Transactions on Pattern Analysis and Machine Intelligence **10**(5), 634–647 (1988)
19. Little, J.B., O'Shea, D.: Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra. Springer (2007)
20. Lu, Y., Song, D.: Robust rgb-d odometry using point and line features. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3934–3942 (2015)
21. Masselli, A., Zell, A.: A new geometric approach for faster solving the perspective-three-point problem. In: Pattern Recognition (ICPR), 2014 22nd International Conference on. pp. 2119–2124. IEEE (2014)
22. Merritt, E.: Explicit three-point resection in space. Photogrammetric Engineering **15**(4), 649–655 (1949)
23. Mur-Artal, R., Montiel, J.M.M., Tardos, J.D.: Orb-slam: a versatile and accurate monocular slam system. IEEE Transactions on Robotics **31**(5), 1147–1163 (2015)
24. OpenCV: Open source computer vision library. https://github.com/itseez/opencv (2017)
25. Proença, P.F., Gao, Y.: Splode: Semi-probabilistic point and line odometry with depth estimation from rgb-d camera motion. arXiv preprint arXiv:1708.02837 (2017)
26. Pumarola, A., Vakhitov, A., Agudo, A., Sanfeliu, A., Moreno-Noguer, F.: Pl-slam: Real-time monocular visual slam with points and lines. In: Robotics and Automation (ICRA), 2017 IEEE International Conference on. pp. 4503–4508. IEEE (2017)
27. Quan, L., Lan, Z.: Linear n-point camera pose determination. IEEE Transactions on pattern analysis and machine intelligence **21**(8), 774–780 (1999)
28. Ramalingam, S., Bouaziz, S., Sturm, P.: Pose estimation using both points and lines for geo-localization. In: Robotics and Automation (ICRA), 2011 IEEE International Conference on. pp. 4716–4723. IEEE (2011)
29. Sola, J., Vidal-Calleja, T., Civera, J., Montiel, J.M.M.: Impact of landmark parametrization on monocular ekf-slam with points and lines. International journal of computer vision **97**(3), 339–368 (2012)
30. Taylor, C.J., Kriegman, D.J.: Structure and motion from line segments in multiple images. IEEE Transactions on Pattern Analysis and Machine Intelligence **17**(11), 1021–1032 (1995)
31. Vakhitov, A., Funke, J., Moreno-Noguer, F.: Accurate and linear time pose estimation from points and lines. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) Computer Vision – ECCV 2016. pp. 583–599. Springer International Publishing, Cham (2016)
32. WU, W.: Basic principles of mechanical theorem proving in elementary geometries. In: Selected Works Of Wen-Tsun Wu, pp. 195–223. World Scientific (2008)
33. Xu, C., Zhang, L., Cheng, L., Koch, R.: Pose estimation from line correspondences: A complete analysis and a series of solutions. IEEE transactions on pattern analysis and machine intelligence **39**(6), 1209–1222 (2017)
34. Yang, S., Scherer, S.: Direct monocular odometry using points and lines. arXiv preprint arXiv:1703.06380 (2017)
35. Zhou, F., Duh, H.B.L., Billinghurst, M.: Trends in augmented reality tracking, interaction and display: A review of ten years of ismar. In: Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality. pp. 193–202. IEEE Computer Society (2008)