An Efficient and Accurate Algorithm for the Perspecitve-n-Point Problem

Lipu Zhou and Michael Kaess

Abstract— **In this paper, we address the problem of pose estimation from N 2D/3D point correspondences, known as the Perspective-n-Point (PnP) problem. Although many solutions have been proposed, it is hard to optimize both computational complexity and accuracy at the same time. In this paper, we propose an accurate and simultaneously efficient solution to the PnP problem. Previous PnP algorithms generally involve two sets of unknowns including the depth of each pixel and the pose of the camera. Our formulation does not involve the depth of each pixel. By introducing some intermediate variables, this formulation leads to a fourth degree polynomial cost function with 3 unknowns that only involves the rotation. In contrast to previous works, we do not address this minimization problem by solving the first-order optimality conditions using the off-the-shelf Gröbner basis method, as the Gröbner basis method may encounter numeric problems. Instead, we present a method based on linear system null space analysis to provide a robust initial estimation for a Newton iteration. Experimental results demonstrate that our algorithm is comparable to the start-of-the-art algorithms in terms of accuracy, and the speed of our algorithm is among the fastest algorithms.**

I. INTRODUCTION

The PnP problem is to compute the camera pose relative to a world frame from N 3D points and their corresponding 2D pixels in the image plane, which is an important problem in compute vision and robotics. It has many applications, such as structure from motion (SfM), simultaneous localization and mapping (SLAM), augmented reality (AR) and visual relocalization. Due to its importance, many algorithms has been proposed to solve this problem. The primary goal of previous works is to balance the computational complexity, accuracy and scalability. Although many PnP algorithms have been proposed, it is still a challenge for an algorithm to achieve accuracy and high speed at the same time. The algorithms based on linearization [1], [2] are fast, but may not be applied to a small N and not robust to noise. On the other hand, algorithms based on nonlinear [3], [4] formulation could be more accurate at the cost of speed. This paper focuses on addressing this problem. We propose an efficient and accurate solution to the PnP problem.

In previous works, the PnP problem is generally formulated by two sets of unknowns, *i.e.* the camera pose and the depth of each point. Our formulation does not involve the depths of the points, which significantly reduces the number of unknowns. However, it leads to a sixth degree polynomial cost function in six unknowns, which is hard to solve. We introduce three intermediate variables to reduce the degree of the polynomial cost function from 6 to 4. These intermediate variables can also decouple the rotation from the translation which results in a fourth degree polynomial with 3 unknowns involving the rotation only. In contrast to the previous works, we do not solve this minimization problem by finding the stationary points using the Gröbner basis method, as the Gröbner basis method may encounter numeric problems [5]. Therefore, algorithms based on Gröbner basis method may be unstable for certain configurations. Although the previous works based on the Gröbner basis method are thought as non-iterative, these methods are actually iterative algorithms, as the Gröbner basis method itself requires to address an eigenvalue problem that is solved by iterative methods. As the cost function is a fourth degree polynomial of 3 unknowns, we can efficiently minimize it by the Newton's method. The result of an iterative algorithm depends on the initial solution. Motivated by [1], we present a method based on linear system null space analysis to provide a robust initial estimation for the damped Newton's method. Our null space analysis method is based on the rotation matrix instead of the control points in EPnP [1]. Besides, EPnP uses linearization [6] to solve for the combination coefficients. The result is not accurate especially when N is small. We use the hidden variable method [7] to calculate the combination coefficients. Experimental results verify that our algorithm is comparable to the start-of-the-art algorithms in terms of accuracy. For the speed, our algorithm is among the fastest algorithms.

The authors are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA `zhoulipu@outlook.com,kaess@cmu.edu`

II. RELATED WORK

The PnP problem has finite solutions when there are at least 3 points. Estimating the camera pose from 3 2D/3D point correspondences are known as the P3P problem. The first solution for the P3P problem is given by Grunert [8] in 1841. This algorithm applies the law of cosines to generate three quadric equations about the lengths between the three 3D points and the camera origin. Several later works [9], [10], [11] adopt this formulation and provide different polynomial solver. Haralick *et al.* [12] give a detailed review of these algorithms. Quan *et al.* [13] apply the Sylvester resultant [14] to solve P3P problem. Gao *et al.* [15] systematically study the P3P problem, and provide a complete analytical solution. Kneip *et al.* [16] and Masselli *et al.* [17] solve the P3P problem by introducing the intermediate coordinate frame to eliminate the variable. Most recently, Ke *et al.* [18]. give an very efficient solution to the P3P problem.

A lot of algorithms have been proposed to solve the PnP problem for $N \geq 4$. Some algorithms are based on the iterative algorithm. In [19], the PnP problem is formulated as a semi-definite positive program problem. In [20], they proposed an orthogonal iterative algorithm to solve the PnP problem. These algorithms are generally computationally demanding. Thus they are not suitable for real-time applications.

Polynomial cost functions are constructed in the literature for the PnP algorithm. Li *et. al.* [21] use the minimal solution to solve the PnP problem. They divide the N points into a set of 3-point subsets. Each 3-point subset is used to generate a quadric equation by [22]. They minimize the sum of the square of these quadric equations. Hesch and Roumeliotis [23] apply the Cayley–Gibbs–Rodriguez (CGR) parametrization to represent the rotation matrix. Their formulation yields three third order polynomial equations as the first-order optimality conditions. Zheng *et al.* [4] use the scaled quaternion to represent the rotation matrix, whose first-order optimality conditions are four three-degree polynomials. They use the two-fold symmetry of the polynomial system to reduce the computational complexity. Laurent *et al.* provide a universal algorithm for the PnP problem. It can be used to different configurations of the points, and different types of cameras. They also use the quaternion to represent the rotation matrix. They introduce $||\mathbf{q}||^2 = 1$ to the optimality conditions of the cost function, rather than apply the Lagrangian formulation for the constraint $||\mathbf{q}||^2 = 1$. This leads to an efficient polynomial solver. In [23], [4], [3], the first-order optimality conditions of a cost functions are calculated. The Gröbner basis method is adopted to solve these polynomial systems. As the Gröbner basis method has numeric problems that are hard to be completely addressed as mentioned in [5], the algorithms based on the Gröbner basis method may be unstable for certain configurations. Furthermore, the speed of algorithms based on the Gröbner basis method is not comparable to the algorithms based on linearization.

Linearization is also used to solve the PnP problem. Quan *et al.* [13] use the minimal solution to generate a linear equation system to solve the P4P and P5P problems. Ansar *et al.* solve the PnP problem by linearizing the quadratic equation system. Lepetit *et. al.* [1] provide the first $O(N)$ solution for the PnP problem. They introduce the control points to simplify the computation. The coordinates of the control points are estimated by computing the weights of the null space vector. Then the camera pose is recovered by fitting 2 sets of 3D points. This algorithm is adopted in the ORB-SLAM system [24]. Ferraz *et. al.* [2] introduce an efficient outlier elimination strategy. This algorithm can generate accurate results in situations with up to 50% of outliers. As the linearization may not well approximate the original cost function, this may result in suboptimal solution.

III. Notation and Problem Definition

Throughout this paper we use italic, boldfaced lowercase and boldfaced uppercase letters to represent scalars, vectors and matrices, respectively.

Suppose that we have N feature points $\mathbf{p}_i = [x_i, y_i]^T$ (such as SIFT [25] or ORB [26] features) and the corresponding 3D points $\mathbf{P}_i = [X_i, Y_i, Z_i]^T$. The PnP problem is to calculate the pose, including the rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{t}$, between the world frame and the camera fame from N ($N \geq 3$) 2D/3D point corresponding $\mathbf{p}_i \Leftrightarrow \mathbf{P}_i$, as shown in Fig. 1. $\mathbf{R} \in SO(3)$ subjects to the constraints $\mathbf{R}^T\mathbf{R} = \mathbf{I}_3$ and $\det(\mathbf{R}) = 1$, where $\det(\mathbf{R})$ represents the determinant of $\mathbf{R}$. To eliminate the constraints of $\mathbf{R}$, we use the Cayley–Gibbs–Rodriguez (CGR) parametrization [23] to represent $\mathbf{R}$ as

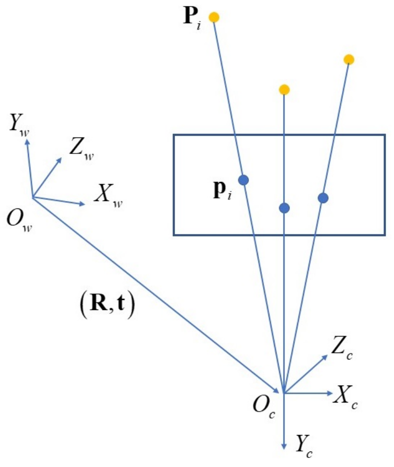


Fig. 1: Schematic of the PnP Problem.

$$\mathbf{R} = \frac{\bar{\mathbf{R}}}{1 + \mathbf{s}^T\mathbf{s}}, \bar{\mathbf{R}} = \left(\left(1 - \mathbf{s}^T\mathbf{s}\right)\mathbf{I}_3 + 2[\mathbf{s}]_\times + 2\mathbf{s}\mathbf{s}^T\right), \quad (1)$$

where $\mathbf{s} = [s_1; s_2; s_3]$ is a 3-dimensional vector and $[\mathbf{s}]_\times = \begin{bmatrix} 0 & -s_3 & s_2 \\ s_3 & 0 & -s_1 \\ -s_2 & s_1 & 0 \end{bmatrix}$. The CGR parameterization does not require additional constraints.

Using the pinhole camera model, the relationship between $\mathbf{p}_i$ and $\mathbf{P}_i$ can be formulated as

$$\begin{bmatrix} \mathbf{p}_i \\ 1 \end{bmatrix} \sim \mathbf{C}\left[\mathbf{R}, \mathbf{t}\right]\begin{bmatrix} \mathbf{P}_i \\ 1 \end{bmatrix}, \quad (2)$$

where $\mathbf{C}$ is the camera intrinsic matrix. To simplify the notation, we multiply both side of (2) by $\mathbf{C}^{-1}$ before solving the PnP problem, and adopt the normalized pixel coordinates $\mathbf{u}_i = [u_i, v_i]^T$ [27], where $[\mathbf{u}_i; 1] = \mathbf{C}^{-1}[\mathbf{p}_i; 1]$ in the following description.

IV. Our Algorithm

In this section, we detail our algorithm. We first introduce our formulation of the PnP problem. This leads to a fourth degree polynomial cost function in three unknowns, then we describe how to calculate the initial solution for this cost function.

A. *Problem Formulation*

In previous works, depth of each pixel $\mathbf{u}_i$ is generally used to formulate the PnP problem [23], [4], [3]. This will introduce additional N unknowns. Our formulation does not involve the depth of each pixel. Substituting the definition of $\mathbf{u}_i$ into (2), we can obtain following equations:

$$u_i = \frac{\mathbf{r}_1\mathbf{P}_i + t_1}{\mathbf{r}_3\mathbf{P}_i + t_3}, v_i = \frac{\mathbf{r}_2\mathbf{P}_i + t_2}{\mathbf{r}_3\mathbf{P}_i + t_3}, \tag{3}$$

where $\mathbf{r}_i$ $(i = 1, 2, 3)$ are the three rows of $\mathbf{R}$, and t_i $(i = 1, 2, 3$ $)$ are the three elements of $\mathbf{t}$. Let us multiply both sides of the two equations in (3) by $\mathbf{r}_3\mathbf{P}_i + t_3$. After some elementary math operations, we have

$$\begin{array}{l}(\mathbf{r}_1\mathbf{P}_i + t_1) - u_i(\mathbf{r}_3\mathbf{P}_i + t_3) = 0,\\ (\mathbf{r}_2\mathbf{P}_i + t_1) - v_i(\mathbf{r}_3\mathbf{P}_i + t_3) = 0.\end{array} \tag{4}$$

If we substitute the CGR parametrization (1) into (4), and multiply $1 + \mathbf{s}^T\mathbf{s}$ to the both sides of (4), we get

$$\begin{array}{l}e_i^x = \left(\bar{\mathbf{r}}_1\mathbf{P}_i + \left(1 + \mathbf{s}^T\mathbf{s}\right)t_1\right) - u_i\left(\bar{\mathbf{r}}_3\mathbf{P}_i + \left(1 + \mathbf{s}^T\mathbf{s}\right)t_3\right) = 0,\\ e_i^y = \left(\bar{\mathbf{r}}_2\mathbf{P}_i + \left(1 + \mathbf{s}^T\mathbf{s}\right)t_2\right) - v_i\left(\bar{\mathbf{r}}_3\mathbf{P}_i + \left(1 + \mathbf{s}^T\mathbf{s}\right)t_3\right) = 0,\end{array} \tag{5}$$

where $\bar{\mathbf{r}}_i$ are the three rows of $\bar{\mathbf{R}}$. As the inevitable noise in the real applications, equations in (5) will not be exactly satisfied. Therefore, we consider the following least-squares problem:

$$\hat{\mathbf{s}}, \hat{\mathbf{t}} = \arg\min_{\mathbf{s},\mathbf{t}} C(\mathbf{s}, \mathbf{t}),\ C(\mathbf{s}, \mathbf{t}) = \sum_i (e_i^x)^2 + (e_i^y)^2. \tag{6}$$

e_i^x and e_i^y in (5) are third-order polynomials in $\mathbf{s}$ and $\mathbf{t}$. Therefore, $C(\mathbf{s}, \mathbf{t})$ is a polynomial cost function of degree 6. In general, we can calculate the first-order optimality conditions of (6) which will yield a system of 6 fifth-order polynomial equations, and then solve it by the Gröbner basis method. Theoretically, the maximum number of solutions of this equation system is $5^6 = 15625$ according to the Bézout theorem [7]. Solving a high order polynomial system is computationally demanding and numerically unstable. In addition, it will take a plenty of time to find the global minimizer from these solutions, even if we are able to correctly calculate these stationary points.

The difficulty of solving (6) is from the terms $\left(1 + \mathbf{s}^T\mathbf{s}\right)t_i$, $i = 1, 2, 3$ in (5). These terms are third-order polynomials, and make $\mathbf{R}$ and $\mathbf{t}$ correlated. To reduce the degree of the cost function in (6) and decouple $\mathbf{R}$ and $\mathbf{t}$, we introduce an intermediate vector

$$\mathbf{T} = [T_1, T_2, T_3]^T, \quad T_i = \left(1 + \mathbf{s}^T\mathbf{s}\right)t_i. \tag{7}$$

Using this definition, we can rewrite (5) as

$$\begin{array}{l}E_i^x = (\bar{\mathbf{r}}_1\mathbf{P}_i + T_1) - u_i(\bar{\mathbf{r}}_3\mathbf{P}_i + T_3) = 0,\\ E_i^y = (\bar{\mathbf{r}}_1\mathbf{P}_i + T_2) - v_i(\bar{\mathbf{r}}_3\mathbf{P}_i + T_3) = 0.\end{array} \tag{8}$$

E_i^x and E_i^y are second-order polynomials in $\mathbf{s}$ and linear in $\mathbf{T}$. This formulation reduces the degrees of the polynomials in (5). Furthermore, we can decouple $\mathbf{R}$ and $\mathbf{t}$. Define

$$\bar{\mathbf{r}} = [\bar{\mathbf{r}}_1, \bar{\mathbf{r}}_2, \bar{\mathbf{r}}_3]^T, \tag{9}$$

which is a 9-dimentional vector. Then we can rewrite (8) as

$$\begin{array}{l}\mathbf{w}_i\mathbf{T} = \mathbf{v}_i\bar{\mathbf{r}},\\ \mathbf{w}_i = \left(\begin{array}{ccc} 1 & 0 & -u_i\\ 0 & 1 & -v_i\end{array}\right), \mathbf{v}_i = -\left(\begin{array}{ccc}\mathbf{P}_i^T & \mathbf{0}_{3\times1} & u_i\mathbf{P}_i^T\\ \mathbf{0}_{3\times1} & \mathbf{P}_i^T & v_i\mathbf{P}_i^T\end{array}\right)\end{array} \tag{10}$$

Given N 2D/3D correspondences, we can stack the coefficients $\mathbf{w}_i$ and $\mathbf{v}_i$ to generate a linear system as

$$\mathbf{W}\mathbf{T} = \mathbf{V}\bar{\mathbf{r}}, \tag{11}$$

where $\mathbf{W} = [\mathbf{w}_1; \mathbf{w}_2; \cdots \mathbf{w}_n]$ and $\mathbf{V} = [\mathbf{v}_1; \mathbf{v}_2; \cdots \mathbf{v}_n]$.

Given $\bar{\mathbf{r}}$, this becomes an unconstrained linear least-squares problem for $\mathbf{T}$. The closed-from expression of $\mathbf{T}$ [3] in terms of $\bar{\mathbf{r}}$ is

$$\mathbf{T} = \left(\mathbf{W}^T\mathbf{W}\right)^{-1}\mathbf{W}^T\mathbf{V}\bar{\mathbf{r}}. \tag{12}$$

Substituting (12) into (11), we obtain an equation system with respect to $\bar{\mathbf{r}}$ as

$$\mathbf{K}\bar{\mathbf{r}} = \mathbf{0},\ \mathbf{K} = \mathbf{W}\left(\mathbf{W}^T\mathbf{W}\right)^{-1}\mathbf{W}^T\mathbf{V} - \mathbf{V}. \tag{13}$$

According to the definition of $\bar{\mathbf{r}}$, we know that every element of $\bar{\mathbf{r}}$ is a function of the $\mathbf{s}$ defined in (1). Considering the noise, we can define the least-squares problem for $\mathbf{s}$ as

$$\hat{\mathbf{s}} = \arg\min_{\mathbf{s}} C(\mathbf{s}), \quad C(\mathbf{s}) = \bar{\mathbf{r}}^T\mathbf{K}^T\mathbf{K}\bar{\mathbf{r}}. \tag{14}$$

As each element of $\mathbf{K}\bar{\mathbf{r}}$ is a quadric polynomial in $\mathbf{s}$, cost function (14) is a fourth-order polynomial only involving $\mathbf{s}$, including the monomials $[s_0^4, s_0^3s_1, s_0^3s_2, s_0^2s_1^2, s_0^2s_1s_2, s_0^2s_2^2, s_0s_1^3, s_0s_1^2s_2, s_0s_1s_2^2, s_0s_2^3, s_1^4, s_1^3s_2, s_1^2s_2^2, s_1s_2^3, s_2^4]^T$. The corresponding first-order optimality conditions of (14) are three polynomial equations of degree three. There are at most 27 solutions for this polynomial system based on the Bézout theorem. We can apply the Gröbner basis method to solve this polynomial system to get all the stationary points, then find the one with the smallest cost as the solution, like previous works [23], [4], [3]. But this paper does not take this way, since the Gröbner basis method may encounter numerical problems [5], as demonstrated in Fig. 2. Besides, the Gröbner basis method is not efficient. We introduce a null space analysis method to efficiently calculate an initial estimation $\hat{\mathbf{s}}_0$ of (14), then we refine $\hat{\mathbf{s}}_0$ by minimizing (14) using the Newton's method.

B. Null Space Analysis

The cost function (14) is derived from the equation system (13). Here, we calculate an initial solution for (14) from the equation system (13). The central idea of our algorithm is to simplify the cost function (14) through the null space analysis.

According to (1) and the definition of $\bar{\mathbf{r}}$ in (9), we have $\bar{\mathbf{r}} = \left(1 + \mathbf{s}^T\mathbf{s}\right)\mathbf{r}$, where $\mathbf{r}$ contains the 3 rows of $\mathbf{R}$. Substituting this equation into (13), we get $\mathbf{K}\mathbf{r} = \mathbf{0}$ This equation is a homogeneous equation system of $\mathbf{r}$. Without noise, the solution of $\mathbf{K}\mathbf{r} = \mathbf{0}$ should lie in the null space of $\mathbf{K}$ and can be expressed as

$$\hat{\mathbf{r}} = \sum_{i=1}^{m}\alpha_i\mathbf{v}_i, \tag{15}$$

where $\mathbf{v}_i$ $(i = 1, \cdots, m)$ are the m right-singular vectors for the m smallest singular values of the singular value decomposition (SVD) of $\mathbf{K}$. We also have $||\mathbf{v}_i||_2 = 1$. For

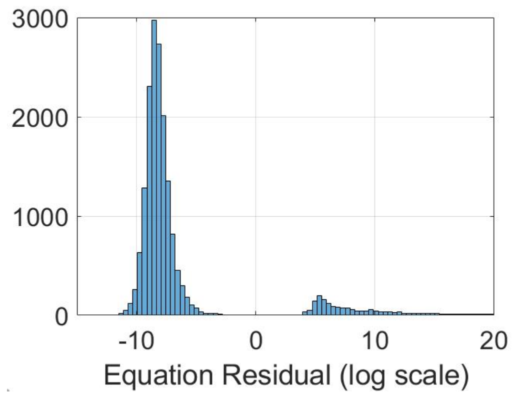


Fig. 2: The residual of polynomial equation system (14) using the Gröbner basis polynomial solver. We use [28] to generate the the Gröbner basis polynomial solver. The coefficient of the polynomial system is uniformly distributed within $[0,1]$. We randomly generate a solution within $[-50,50]$. We then calculate the constant term of each equation to ensure a polynomial system has that solution. The results are from 5000 independent trials. The long tail of the residual shows that the Gröbner basis polynomial solver is not stable.

$N=4$, the dimension of the null space of $\mathbf{K}$ will be 4. Therefore, we consider that the dimension of the null space of $\mathbf{K}$ can vary from 1 to 4 as EPnP [1]. But unlike EPnP that uses the distances between 4 control points to establish equations, we consider computing the coefficient α_i by the constraints of $\mathbf{R}$. We describe how to calculate the coefficient α_i when m varying from 1 to 4 as follows:

Case 1: Because (13) is a homogeneous equation, the solution can only be determined up to scale. In this case, the solution can be represented as $\hat{\mathbf{r}}=\alpha_1\mathbf{v}_1$. As $||\hat{\mathbf{r}}||_2$ should be equal to 3 and $||\mathbf{v}_1||_2=1$, substituting $\hat{\mathbf{r}}=\alpha_1\mathbf{v}_1$ into $||\hat{\mathbf{r}}||_2=3$, we get $\hat{\mathbf{r}}=\sqrt{3}\mathbf{v}_1$.

Case 2: In this case, we have $\hat{\mathbf{r}}=\alpha_1\mathbf{v}_1+\alpha_2\mathbf{v}_2$. Let $\hat{\mathbf{r}}_1$, $\hat{\mathbf{r}}_2$, and $\hat{\mathbf{r}}_3$ be the three vectors in $\hat{\mathbf{r}}$ corresponding to the three rows of a rotation matrix. We split $\mathbf{v}_1$ and $\mathbf{v}_2$ in the same manner. Let $\hat{\mathbf{v}}_i^j$ ($i=1,2$ and $j=1,2,3$) are the resulting vectors. Then $\hat{\mathbf{r}}_i$ can be represented as

$$\hat{\mathbf{r}}_i=\alpha_1\hat{\mathbf{v}}_1^j+\alpha_2\hat{\mathbf{v}}_2^j, j=1,2,3. \tag{16}$$

Using the constraints of the rotation matrix, we obtain 6 equations as

$$\begin{aligned}&f_1=\hat{\mathbf{r}}_1\cdot\hat{\mathbf{r}}_1=1,\ f_2=\hat{\mathbf{r}}_2\cdot\hat{\mathbf{r}}_2=1,\ f_3=\hat{\mathbf{r}}_3\cdot\hat{\mathbf{r}}_3=1,\\ &f_4=\hat{\mathbf{r}}_1\cdot\hat{\mathbf{r}}_2=0,\ f_5=\hat{\mathbf{r}}_1\cdot\hat{\mathbf{r}}_3=0,\ f_6=\hat{\mathbf{r}}_2\cdot\hat{\mathbf{r}}_3=0.\end{aligned} \tag{17}$$

Substituting (16) into (17), we have,

$$\mathbf{A}\mathbf{a}=\mathbf{b}, \tag{18}$$

where $\mathbf{a}=\left[\alpha_1^2,\alpha_1\alpha_2,\alpha_1^2\right]^T$, $\quad\mathbf{b}=[1,1,1,0,0,0]^T$.

This equation system can be solved by the linearization technology [6]. Specifically, the linearization method treats $\alpha_1^2,\alpha_1\alpha_2,\alpha_1^2$ as three independent unknowns. Then the three unknowns are solved as a linear least-squares problem. As this method ignores the relationship between the monomials, this may result in a suboptimal solution. Here we introduce a new method to solve (18). We seek to find α_1 and α_2 that minimize the least-squares cost function

$$[\hat{\alpha}_1,\hat{\alpha}_2]=\arg\min_{\alpha_1,\alpha_2} C(\alpha_1,\alpha_2)=\|\mathbf{A}\mathbf{a}-\mathbf{b}\|_2^2. \tag{19}$$

$C(\alpha_1,\alpha_2)$ is a fourth-order polynomial in α_1 and α_2. To find its global minimizer, we calculate its first-order optimality conditions as

$$g_1=\frac{\partial C(\alpha_1,\alpha_2)}{\partial\alpha_1}=0,\ g_2=\frac{\partial C(\alpha_1,\alpha_2)}{\partial\alpha_2}=0. \tag{20}$$

The two equations in (20) are of degree 3 with the monomials $\left[\alpha_1^3,\alpha_2^2\alpha_2,\alpha_1\alpha_2^2,\alpha_2^3,\alpha_1,\alpha_2\right]$. Since the Gröbner basis method has numerical problems, we use the hidden variable method [7] to solve this equation. In the hidden variable method, one unknown is treated as a constant. This unknown is called the hidden variable. Without loss of the generality, we treat α_2 as the hidden variable. This will generate an equation system from (20) as

$$g_i=c_{i1}\alpha_1^3+c_{i2}(\alpha_2)\alpha_1^2+c_{i3}(\alpha_2)\alpha_1+c_{i4}(\alpha_2)=0, \tag{21}$$

where $c_{ij}(\alpha_2)$ ($i=1,2,\ j=2,3,4$) are polynomials of α_2. In the following description, we use c_{ij} to represent $c_{ij}(\alpha_2)$ for simplification. Then according to [7], the solution of α_2 should satisfy

$$f(\alpha)=\det\begin{pmatrix}c_{11}&0&0&c_{21}&0&0\\c_{12}&c_{11}&0&c_{22}&c_{21}&0\\c_{13}&c_{12}&c_{11}&c_{23}&c_{22}&c_{21}\\c_{14}&c_{13}&c_{12}&c_{24}&c_{23}&c_{22}\\0&c_{14}&c_{13}&0&c_{24}&c_{23}\\0&0&c_{14}&0&0&c_{24}\end{pmatrix}=0. \tag{22}$$

This is a ninth-order polynomial equation in α_2 with the form as

$$k_9\alpha_2^9+k_7\alpha_2^7+k_5\alpha_2^5+k_3\alpha_2^3+k_1\alpha_2=0. \tag{23}$$

$\alpha_2=0$ is equal to **case 1** mentioned above. Here, we only consider the nontrivial solution. We divide both side of (23) by α_2. This leads to a eighth-order polynomial equation with only even terms. If we define $\beta=\alpha_2^2$, the resulting polynomial equation is a quartic,

$$k_9\beta^4+k_7\beta^3+k_5\beta^2+k_3\beta+k_1=0. \tag{24}$$

Equation (24) has a close-form solution. After we get α_2, we substitute α_2 into (21). α_1 can be solved from the resulting third order equations.

Case 3: The resulting combination in this case will be $\hat{\mathbf{r}}=\alpha_1\mathbf{v}_1+\alpha_2\underline{\mathbf{v}}_2+\alpha_3\mathbf{v}_3$. Substituting $\hat{\mathbf{r}}$ into the 6 constraints of the rotation matrix mentioned in (17), we can obtain a system of 6 equations as (18), except that we have $\mathbf{a}=\left[\alpha_1^2,\alpha_2^2,\alpha_3^2,\alpha_1\alpha_2,\alpha_1\alpha_3,\alpha_2\alpha_3\right]$. Let us consider the first 3 equations $f_1=1,f_2=1,f_3=1$ in (17). We define

$$g_1=f_1-f_2=0,\quad g_2=f_1-f_3=0. \tag{25}$$

Then $g_1=0, g_2=0, f_4=0, f_5=0, f_6=0$ are combined to generate a new equation system

$$\mathbf{B}\mathbf{a}=\mathbf{0}, \mathbf{a}=\left[\alpha_1^2,\alpha_2^2,\alpha_3^2,\alpha_1\alpha_2,\alpha_1\alpha_3,\alpha_2\alpha_3\right]. \tag{26}$$

We define

$$\alpha_2=k_1\alpha_1, \alpha_3=k_2\alpha_1. \tag{27}$$

Substituting (27) into (26) and dividing α_1 to both side of (26), we can obtain

$$\mathbf{B}\mathbf{k}=\mathbf{0}, \mathbf{k}=\left[k_1^2,k_2^2,k_1k_2,k_1,k_2,1\right]^T. \tag{28}$$

Similar to (19) in **Case 2**, we can define the cost function for k_1, k_2 as

$$\left[\hat{k}_1,\hat{k}_2\right]=\arg\min_{k_1,k_2} C\left(k_1,k_2\right)=\|\mathbf{B}\mathbf{k}\|_2^2. \tag{29}$$

We can solve (29) using the similar method as solving (19). After we get k_1, k_2, we can substitute (27) into f_1 to calculate α_1. Then we can compute α_2 and α_3 using the definition in (27).

Case 4: This case is more complicated than the above 3 cases. In this case, the form of the solution will be $\hat{\mathbf{r}}=\alpha_1\mathbf{v}_1+\alpha_2\mathbf{v}_2+\alpha_3\mathbf{v}_3+\alpha_4\mathbf{v}_4$. The 6 constraints of the rotation matrix mentioned in (17) will generate 6 equations for $\alpha_i, i=1,2,3,4$. But there are 10 possible combinations of $\alpha_i a_j$, *i.e.*, $\left[\alpha_1^2,\alpha_2^2,\alpha_3^3,\alpha_4^2,\alpha_1\alpha_2,\alpha_1\alpha_3,\alpha_1\alpha_4,\alpha_2\alpha_3,\alpha_2\alpha_4,\alpha_3\alpha_4\right]$. The relinearization method [29] was adopted to solve this problem in [1]. Here we introduce a new solution for this problem.

We first consider the last 3 equations of (17), *i.e.*, $f_4=0, f_5=0, f_6=0$. They are homogeneous equations. Let us define

$$\alpha_2=k_1\alpha_1, \alpha_3=k_2\alpha_1, \alpha_4=k_3\alpha_1. \tag{30}$$

Substituting (30) into $f_4=0, f_5=0, f_6=0$ and eliminating α_1, we can obtain

$$\begin{aligned}&\mathbf{C}\mathbf{x}=\mathbf{0},\\&\mathbf{x}=\left[k_1^2,k_2^2,k_3^2,k_1k_2,k_1k_3,k_2k_3,k_1,k_2,k_3,1\right]^T.\end{aligned} \tag{31}$$

They are three quadratic equations in three unknowns. We can efficiently solve this equation system by [30]. For completeness, we briefly introduce this method below. We treat k_1 as a constant, and split the remaining monomials into two parts k_2^2, k_3^2, k_2k_3 and $k_2, k_3, 1$. Then (31) can be rewritten as

$$\mathbf{M}\begin{bmatrix}k_2^2\\k_3^2\\k_2k_3\end{bmatrix}=\mathbf{N}\left(k_1\right)\begin{bmatrix}k_2\\k_3\\1\end{bmatrix}. \tag{32}$$

Each element of $\mathbf{N}\left(k_1\right)$ is a polynomial in k_1. If we treat k_2^2, k_3^2, k_2k_3 as independent variables, we can have a closed form expression of them as

$$\begin{bmatrix}k_2^2\\k_3^2\\k_2k_3\end{bmatrix}=\mathbf{T}\left(k_1\right)\begin{bmatrix}k_2\\k_3\\1\end{bmatrix}, \mathbf{T}\left(k_1\right)=\left(\mathbf{M}^T\mathbf{M}\right)^{-1}\mathbf{M}^T\mathbf{N}\left(k_1\right) \tag{33}$$

Then three identities $k_2^2\times k_3=k_2\times k_2k_3$, $k_2k_3\times k_3=k_2\times k_3^2$, $k_2k_3\times k_2k_3=k_2^2\times k_3^2$ are used to generate three constraints for k_1, k_2 and k_3. Substituting the expression of k_2^2, k_3^2, k_2k_3 in (33) into these three identities, we can get a homogeneous system as

$$\mathbf{M}\left(k_1\right)\begin{bmatrix}k_2\\k_3\\1\end{bmatrix}=\begin{bmatrix}0\\0\\0\end{bmatrix}. \tag{34}$$

where $\mathbf{M}(k_1)$ is 3×3 matrix with polynomials of k_1 as its elements. Denote the determinant of $\mathbf{M}(k_1)$ as $\det\left(\mathbf{M}(k_1)\right)$. According to the linear algebra theory, $\det\left(\mathbf{M}(k_1)\right)$ should equal to zero when (34) has non-trivial solutions. $\det\left(\mathbf{M}(k_1)\right)$ is an eighth-order polynomial in k_1. We can solve $\det\left(\mathbf{M}(k_1)\right)=0$ for k_1, then back-substitute k_1 into the linear system (34) to get k_2 and k_3. After we get k_1, k_2 and k_3, we substitute (30) into the first three equations of (17). Then we can solve for α_1 by minimizing the following cost function.

$$\hat{\alpha}_1=\arg\min_{\alpha_1}\sum_{i=1}^{3}\|f_i-1\|_2^2. \tag{35}$$

This cost function only involves α_1. Its first-order optimality condition is a third-order polynomial equation. It has closed-form solution. After we get k_1, k_2, k_3 and α_1, α_2, α_3, α_4 can be estimated from the definition (30).

Planar Case: For the planar case, a 3D point could be represented as $[x,y,0]^T$. This can be used to simplify the computation. Let $\mathbf{c}_i$ $(i=1,2,3)$ be the three columns of $\mathbf{R}$. As the z coordination of a planar point could be zero, we can cancel the last column $\mathbf{c}_3$ of $\mathbf{R}$. As the non-planar case, we use the constraints of $\mathbf{R}$ to calculate the combination coefficients of the right singular vectors of the resulting equitation system for $\mathbf{c}_1$ and $\mathbf{c}_2$. Specifically, $\mathbf{c}_1$ and $\mathbf{c}_2$ satisfy $\mathbf{c}_1\cdot\mathbf{c}_1=1$, $\mathbf{c}_2\cdot\mathbf{c}_2=1$ and $\mathbf{c}_1\cdot\mathbf{c}_2=0$. As here we reduce the elements of $\mathbf{R}$, we only consider m = 1 and 2 for the planar case.

C. Recover the CGR Parameters

After we get $\hat{\mathbf{r}}$, we can obtain the rotation $\hat{\mathbf{R}}$ by rearranging the elements of $\hat{\mathbf{r}}$. $\hat{\mathbf{R}}$ may be an invalid rotation matrix because of the noise. Assume the SVD of $\hat{\mathbf{R}}$ is $\hat{\mathbf{R}}=\mathbf{U}\mathbf{S}\mathbf{V}^T$. Then the closest rotation matrix $\bar{\mathbf{R}}$ for $\hat{\mathbf{R}}$ in the Frobenius sense is $\bar{\mathbf{R}}=\mathbf{U}\mathbf{V}^T$ [31]. Then s can be calculated by transforming $\bar{\mathbf{R}}$ into the CGR parameters.

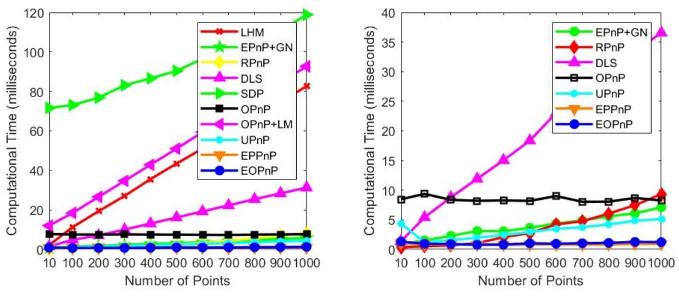
D. Translation Estimation

When we get $\hat{\mathrm{s}}$, the scaled translation $\hat{\mathbf{T}}$ in (7) can be calculated by (12). Then the translation can be recovered by $\hat{\mathbf{t}}=\hat{\mathbf{T}}\big/\left(\mathbf{1}+\hat{\mathrm{s}}^T\hat{\mathrm{s}}\right)$ using the definition in (7).

E. Estimation Refinement

After we get the estimations from the above steps, we can obtain an initial solution. Then we refine the initial solution by using the damped Newton method.

The cost function (14) is a polynomial function. Therefore, we can efficiently calculate its Hessian matrix and gradient



(a) (b)

Fig. 7: Computational time. (a) shows the computational time of all the methods. (b) gives a closeup.

vector. The iterative algorithm starts from $\hat{\mathbf{s}}_0$. For the nth iteration, we first compute the Hessian matrix $\mathbf{H}_n$ and the gradient vector $\nabla \mathbf{C}_n(\mathbf{s})$ of (14). Then we update the solution by $\hat{\mathbf{s}}_{k+1} = \hat{\mathbf{s}}_k - (\mathbf{H} + \lambda \mathbf{I}_3)^{-1} \nabla \mathbf{C}_n(\mathbf{s})$. λ is adjusted according to the LM algorithm [32] to ensure the cost (14) is reduced.

F. Algorithm Summarization

We first calculate $\mathbf{K}$ in (13). Then we compute the SVD of $\mathbf{K}$ to get the right singular vectors. We next solve the 4 cases to get an initial estimation. This initial solution is then refined by minimizing (14) using the damped Newton's method. Finally, we get $\mathbf{t}$ using (12) and (7). We summarize our algorithm as follows:

Input: N $\mathbf{P}_i \Leftrightarrow \mathbf{p}_i$ $(N \geq 4)$
Output: Camera pose relative to the world frame

1) Construct the matrices $\mathbf{W}$ and $\mathbf{V}$ in (11), then compute the coefficient matrix $\mathbf{K}$ of the linear system (13).
2) Calculate the SVD of $\mathbf{K}$ to get the four right singular vector $\mathbf{v}_i$, $i = 1, 2, 3, 4$ corresponding to the four smallest singular value.
3) Calculate the initial solution $\hat{\mathbf{s}}_0$ by computing the 4 possible combinations of the null space of $\mathbf{K}$.
4) Refine $\hat{\mathbf{s}}_0$ by minimizing (14) using the damped Newton's method.
5) Compute the rotation $\hat{\mathbf{R}}$ from (1) and the translation $\hat{\mathbf{t}}$ from (11) and (7).

V. Experimental Results

In this section, we compare the accuracy and the running time of our algorithm, referred to as **EOPnP**, with the state-of-the-art PnP algorithms, including the method based on minimal solution **RPnP** [21], the method based on semi-definite positive program **SDP** [19] and orthogonal iterative algorithm **LHM** [20], the Gröbner basis based algorithms **OPnP** [4], **UPnP**[3] and **DLS** [23], and the methods based on null space combination **EPnP+GN** [1] and **EPPnP** [2]. Besides, we also consider the iterative solution from minimizing the reprojection error **OPnP+LM** [4], which is known as the gold standard algorithm [27].

A. Experiment with Synthetic Data

Our simulation covers all possible rotations. We uniformly sample the whole range of Euler angles α, β, γ ($\alpha, \gamma \in [0°, 360°]$ and $\beta \in [0°, 180°]$). The 3D points are distributed within a $[-2, 2] \times [-2, 2] \times [4, 8]$ box. We choose the origin of the world frame at the centroid of these 3D points as [4].

The result of each experiment is obtained from 500 independent trials. Denote the estimated rotation and translation as $\hat{\mathbf{R}}$ and $\hat{\mathbf{t}}$, and the ground truth as $\mathbf{R}_{gt}$ and $\mathbf{t}_{gt}$. We evaluate the rotation error by the maximum angle between each column of $\hat{\mathbf{R}}$ and $\mathbf{R}_{gt}$, *i.e.* $\max_{k=1}^{3} acos(\mathbf{r}_{gt}^k, \mathbf{r}^k)$, and the translation error by $\left\|\mathbf{t}_{gt} - \hat{\mathbf{t}}\right\|_2 / \left\|\mathbf{t}_{gt}\right\|_2$ as [4].

Varying Number of Points Denote the standard deviation of a zero mean Gaussian noise as δ. In the first experiment, the number of points N varies from 4 to 15 with a fixed standard deviation $\delta = 2$ pixels. Fig. 3 gives the results. Fig. 3 (a) and (b) include the results of all the methods. Fig. 3 (c) and (d) compares the algorithms with similar results. **DLS** is not stable and out of the range. Generally, **UPnP** is stable, but it generates a large estimation error at $N = 10$. This may be caused by the numeric problem of the Gröbner basis method. The result from **EPnP+GN** and **EPPnP** are with large errors, especially when N is small. This is the reason why we need to introduce the hidden variable solution. **LHM** is not stable when N is small. Our algorithm (**EOPnP**), **OPnP** and **SDP** are the most stable ones among these algorithms. Furthermore, our algorithm gives a comparable result with **OPnP+LM** which gives the maximum likelihood estimation under the Gaussian noise. This is probably because our algebraic error is generated from the reprojection error.

Varying Noise Level This experiment considers the performance of different algorithms under increasing noise level. We set the number of points to 10. The standard deviation of the Gaussian noise varies from 0.5 to 5 pixels. Fig. 4 demonstrates the results. Like the first experiments, Fig. 4 (a) and (b) show the results of all the algorithms. (c) and (d) compare the algorithms with similar performance. The results of **SDP**, **OPnP**, **LHM** and our algorithm are more accurate than other algorithms.

Computational Time We compare the computational time of different algorithms. Fig. 7 gives the results. **EPPnP** and our algorithm **EOPnP** are the fastest ones among these algorithms. **EPPnP** is slightly faster than our algorithm, as our cost function of the finial iterative step is more complicated than **EPPnP**. But our algorithm is much more accurate than **EPPnP**. Our algorithm is about 8 times faster than **OPnP**. The LM algorithm [32] based on the reprojection error is time-consuming, and can not be applied to a large-scale problem with real-time requirement.

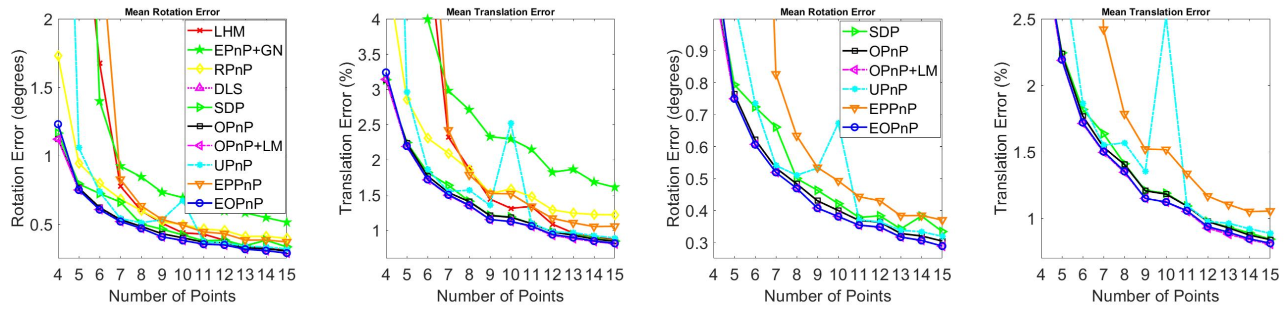


Fig. 3: Experiment results w.r.t. varying point numbers. (a) and (b) show the results of all the methods. (c) and (d) compare the algorithms with close results

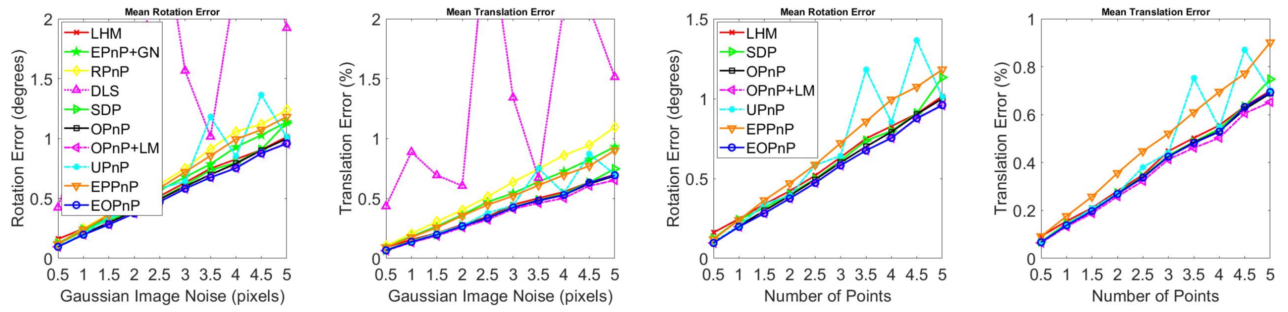
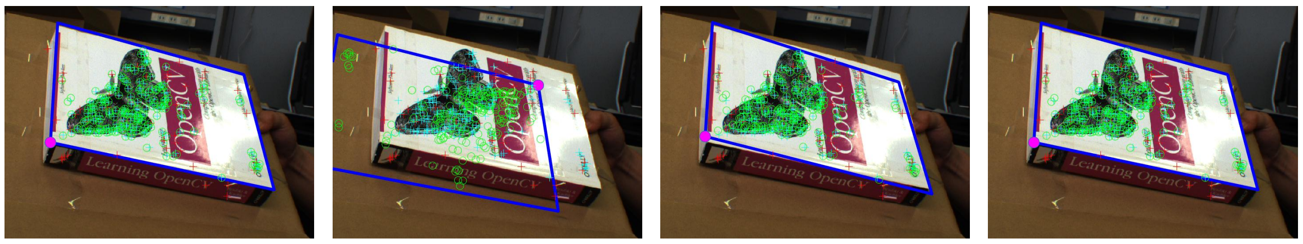


Fig. 4: Experiment results w.r.t. varying noise levels. (a) and (b) show the results of all the methods. (c) and (d) compare the algorithms with close results



(a) **EOPnP** (b) **EPnP** (c) **UPnP** (d) **OPnP**

Fig. 5: Experiments with the book cover.

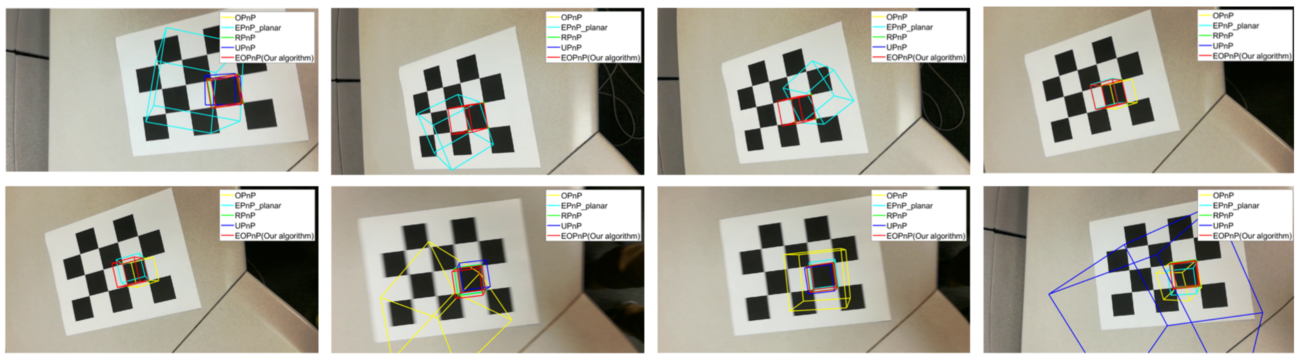


Fig. 6: Experiments with a checkerboard.

B. Experiment with Real Data

We test our algorithm on real images. The book cover image is from [4]. The feature points in the test image is matched with the reference image by the SIFT [25] feature. Fig. 5 gives one example. Our algorithm gives visually accurate result. We also use **EPnP**, **UPnP** and **OPnP** to compute the pose. The results from **EPnP** and **UPnP** are not accurate. Our algorithm and **OPnP** provide similar results.

Planar and uncentered (points are within in a small region) configurations are challenging for the PnP algorithm. We evaluated the performance of different algorithms under this configuration. 4 vertexes of one block of a checkerboard are used to estimate the pose between the checkerboard frame and the camera frame. A cube is put on the checkerboard to visually evaluate the PnP algorithms. Fig. 6 shows some results. It is obviously that our algorithm gives more accurate results.

VI. Conclusion

In this paper, we propose an efficient and accurate solution for the PnP problem. It is a challenging task to achieve an accurate solution as well as maintain the efficiency. Our algorithm avoids involving the depth of each point into the formulation. This results in a sixth-order polynomial with six unknowns. We simplify this problem by introducing three intermediate variables. This leads to a fourth degree polynomial with three unknowns. We do not adopt the Gröbner basis method to compute the stationary points. Although this strategy is optimal in theory, the Gröbner basis method may encounter numeric problems. We present a null space combination method to provide stable initial estimation for the damped Newton iteration. The experimental results verify that our algorithm is comparable to previous state-of-the-art algorithms in terms of accuracy, and is among the fastest algorithms.

References

[1] V. Lepetit, F. Moreno-Noguer, and P. Fua, “Epnp: An accurate o (n) solution to the pnp problem,” *International journal of computer vision*, vol. 81, no. 2, p. 155, 2009.

[2] L. Ferraz, X. Binefa, and F. Moreno-Noguer, “Very fast solution to the pnp problem with algebraic outlier rejection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 501–508.

[3] L. Kneip, H. Li, and Y. Seo, “Upnp: An optimal o (n) solution to the absolute pose problem with universal applicability,” in *European Conference on Computer Vision*. Springer, 2014, pp. 127–142.

[4] Y. Zheng, Y. Kuang, S. Sugimoto, K. Astrom, and M. Okutomi, “Revisiting the pnp problem: A fast, general and optimal solution,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2344–2351.

[5] M. Byröd, K. Josephson, and K. Åström, “Fast and stable polynomial equation solving and its application to computer vision,” *International Journal of Computer Vision*, vol. 84, no. 3, pp. 237–256, 2009.

[6] A. Ansar and K. Daniilidis, “Linear pose estimation from points or lines,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 578–589, 2003.

[7] D. A. Cox, J. Little, and D. O’shea, *Using algebraic geometry*. Springer Science & Business Media, 2006, vol. 185.

[8] J. A. Grunert, “Das pothenotische problem in erweiterter gestalt nebst über seine anwendungen in der geodäsie,” *Grunerts archiv für mathematik und physik*, vol. 1, no. 238-248, p. 1, 1841.

[9] S. Finsterwalder and W. Scheufele, *Das rückwärtseinschneiden im raum*. Verlag d. Bayer. Akad. d. Wiss., 1903.

[10] E. Merritt, “Explicit three-point resection in space,” *Photogrammetric Engineering*, vol. 15, no. 4, pp. 649–655, 1949.

[11] S. Linnainmaa, D. Harwood, and L. S. Davis, “Pose determination of a three-dimensional object using triangle pairs,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 5, pp. 634–647, 1988.

[12] B. M. Haralick, C.-N. Lee, K. Ottenberg, and M. Nölle, “Review and analysis of solutions of the three point perspective pose estimation problem,” *International journal of computer vision*, vol. 13, no. 3, pp. 331–356, 1994.

[13] L. Quan and Z. Lan, “Linear n-point camera pose determination,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 21, no. 8, pp. 774–780, 1999.

[14] J. B. Little and D. O’Shea, *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra*. Springer, 2007.

[15] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng, “Complete solution classification for the perspective-three-point problem,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 25, no. 8, pp. 930–943, 2003.

[16] 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 *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 2969–2976.

[17] A. Masselli and A. Zell, “A new geometric approach for faster solving the perspective-three-point problem,” in *Pattern Recognition (ICPR), 2014 22nd International Conference on*. IEEE, 2014, pp. 2119–2124.

[18] T. Ke and S. I. Roumeliotis, “An efficient algebraic solution to the perspective-three-point problem,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 4618–4626.

[19] G. Schweighofer and A. Pinz, “Globally optimal o (n) solution to the pnp problem for general camera models.” in *BMVC*, 2008, pp. 1–10.

[20] C.-P. Lu, G. D. Hager, and E. Mjolsness, “Fast and globally convergent pose estimation from video images,” *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 6, pp. 610–622, 2000.

[21] S. Li, C. Xu, and M. Xie, “A robust o (n) solution to the perspective-n-point problem,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 7, pp. 1444–1450, 2012.

[22] S. Li and C. Xu, “A stable direct solution of perspective-three-point problem,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 25, no. 05, pp. 627–642, 2011.

[23] J. A. Hesch and S. I. Roumeliotis, “A direct least-squares (dls) method for pnp,” in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 383–390.

[24] R. Mur-Artal and J. D. Tardós, “Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras,” *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[25] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[26] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “Orb: An efficient alternative to sift or surf,” in *Computer Vision (ICCV), 2011 IEEE international conference on*. IEEE, 2011, pp. 2564–2571.

[27] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[28] Z. Kukelova, M. Bujnak, and T. Pajdla, “Automatic generator of minimal problem solvers,” in *European Conference on Computer Vision*. Springer, 2008, pp. 302–315.

[29] A. Kipnis and A. Shamir, “Cryptanalysis of the hfe public key cryptosystem by relinearization,” in *Annual International Cryptology Conference*. Springer, 1999, pp. 19–30.

[30] Z. Kukelova, J. Heller, and A. Fitzgibbon, “Efficient intersection of three quadrics and applications in computer vision,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1799–1808.

[31] N. J. Higham, *Matrix nearness problems and applications*. Citeseer, 1988.

[32] J. J. Moré, “The levenberg-marquardt algorithm: implementation and theory,” in *Numerical analysis*. Springer, 1978, pp. 105–116.