Global Optimal PnP Algorithm with Reverse Projection Error

Xuhao Jin(✉)

North China University of Technology, Beijing 100043, China
jxh2453020228@163.com

Abstract. A non-iterative global optimal PnP (perspective-n-point) algorithm with time complexity O(n) is proposed. The basic idea is to use camera internal parameters to reverse project 2D points into space, use the relationship between 3D points in world coordinates and 2D points to transform PnP problem into minimization problem, and use non-unit quaternion parameterization rotation matrix to transform PnP problems into unconstrained minimization problem, and finally solve the polynomial equation system composed of its first-order partial derivative. And the proposed method can deal with the minimization problem of PnP problem, P3P problem. Experimental results show that the proposed method can accurately process the all configuration of 3D points, and compared with the state-of-the-art PnP algorithm, the proposed method can provide comparable or better accuracy, and the computational efficiency is higher.

Keywords: PnP algorithm · Global optimal · Reverse projection · Non-unit quaternion · Minimization

1 Introduction

Pose estimation is a fundamental problem in computer vision, with wide applications in areas such as navigation [1], robotics [2], and virtual reality [3]. Pose refers to the combination of position and orientation, and pose estimation aims to find the transformation relationship between the target coordinate system and the world coordinate system. This transformation is typically represented by a rotation matrix R (a 3×3 matrix satisfying orthogonality constraints $RR^T = I$ and determinant constraints $\det(R) = 1$) and a translation matrix T. The most commonly used sensors for pose estimation include IMU (Inertial Measurement Unit), LiDAR, and cameras. IMU can provide high-frequency motion information but suffers from significant drift over time. LiDAR can effectively detect the surrounding environment but is expensive and sensitive to weather conditions. Cameras, on the other hand, have the advantages of low cost and rich sensing information, making visual-based pose estimation highly advantageous.

A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 594, pp. 192–203, 2024.
https://doi.org/10.1007/978-3-031-63992-0_12

Pose estimation is a core problem in many theoretical research and practical applications. For example, in the front-end visual odometry of a visual SLAM system [4], the camera is used as the sole or primary sensor. The feature-based approach is employed, where feature points are first extracted from the images, then feature matching is performed to find corresponding features between two images, and finally, the camera pose is estimated using the matched feature points. There are three methods for camera pose estimation based on feature points: 1) 2D-2D: When the camera is monocular, only 2D point coordinates are available. The camera motion is estimated by solving the epipolar geometry problem using two sets of matched 2D points [5]. 2) 3D-3D: When the camera is binocular or RGB-D, or when distance information is obtained through some means, the camera motion is estimated based on two sets of 3D points using the ICP algorithm [6]. 3) 3D-2D: When both the 3D points and their 2D projections on the image are known, the camera pose is estimated by solving the PnP problem. In this case, a minimum of three point correspondences is required to estimate the camera motion.

Perspective-n-Point (PnP) problem for target pose estimation based on feature points. In this problem, given the camera intrinsic parameters, the coordinates of n ($n \geq 3$) feature points in the target coordinate system (often referred to as the world coordinate system) and their corresponding pixel coordinates in the image pixel coordinate system, we aim to compute the target coordinates in the camera coordinate system and thus determine the transformation between the camera coordinate system and the world coordinate system using the perspective projection relationship.

In this paper, we propose a non-iterative PnP algorithm. Our basic idea is to use the perspective projection relationship to reverse project 2D points into 3D points (3D points in the camera coordinate system). At this point, the depth of the reverse projected 3D points is unknown. However, the depth can be represented by the unknown rotation matrix and its translation, as well as the known 3D point coordinates in the world coordinate system. Due to the presence of noise, the 3D points in the two coordinate systems are not exactly equal. We exploit this error to transform the PnP problem into a minimization problem, which can be solved using the method of Gröbner bases.

Our method has the following advantages: (1) It uses non-unit quaternion to represent the rotation matrix, which eliminates the constraints on the rotation matrix. As a result, the PnP problem is formulated as an unconstrained optimization problem. Compared to the polynomials formed by unit quaternion with constraints, our method is simpler to solve when dealing with polynomials. (2) Unlike iterative methods that may converge to local optima, our method can obtain the global optimum. (3) It can handle the minimization problem in PnP (P3P). (4) Unlike OPnP and UpnP that return multiple solutions, our method only returns one solution.

2 Related Work

The PnP algorithm has been widely used in fields such as robot localization and virtual reality. Over the past few decades, researchers have proposed many methods to solve the PnP problem more accurately and efficiently. These methods can be roughly divided into iterative methods and non-iterative methods.

Iterative methods transform the PnP problem into a nonlinear least squares problem, typically using reprojection error as the error function, and solve it using iterative optimization methods such as Gauss-Newton or Levenberg-Marquardt [7]. Compared to non-iterative methods, iterative methods achieve higher accuracy but require longer computation time and better initial values, otherwise they may converge to local minima. Lu et al. [8] proposed a method that uses 3D error as the error function and alternately optimizes the rotation matrix and translation matrix T to enhance the orthogonality of the rotation matrix. Although it converges quickly, it still suffers from the problem of getting stuck in local minima. In this paper, a non-iterative method is used, so only a brief description of the iterative methods is provided here.

Direct Linear Transform (DLT) [9,10] is a traditional approach in non-iterative PnP algorithms. The DLT method solves the pose by exploiting the linear relationship between at least six sets of corresponding points. However, in the solution process, the pose is directly treated as twelve unknowns, ignoring the constraints between them (the rotation matrix R needs to satisfy $RR^T = I$ and $\det(R) = 1$). The rotation matrix obtained using DLT, denoted as R, is a general 3×3 matrix that may not satisfy the constraints of a rotation matrix, resulting in lower accuracy. Quan et al. [11] proposed a linear solution to the PnP problem, which can obtain a unique solution when n = 4 or 5. However, it is still not accurate enough when n is small, and it becomes time-consuming when n is large.

To solve these problems, Lepetit et al. [12] proposed EPnP, which uses four control points to represent the coordinates of 3D points in world coordinates. By solving for the coordinates of the four control points in camera coordinates, the coordinates of other 3D points in camera coordinates can be obtained. EPnP is the first non-iterative method with a time complexity of O(n) and has been widely used. However, EPnP is not accurate for n = 4 or 5, and it cannot handle the case of n = 3.

To improve accuracy, Li et al. [13] proposed RPnP, a non-iterative method with a time complexity of O(n). The basic idea is to divide the n 3D points into n-2 sets and construct a quartic polynomial. The PnP problem is then solved by minimizing the sum of squares of the quartic polynomial. RPnP can still obtain accurate solutions for n = 4 or 5.

Hesch et al. [14] proposed the DLS method with a time complexity of O(n). They formulated the PnP problem as a nonlinear least squares function represented by a multivariate polynomial. By using multiplication matrices, all solutions can be obtained. However, due to the use of Cayley parameterization, the accuracy of DLS is unstable and it becomes singular when the rotation is 180°. To address the singularity issue, Zheng et al. [15] used non-unit quaternion to

represent the rotation matrix and solved the PnP problem using the Gröbner basis method [16]. They also considered the symmetry of the solutions. However, in the solving process, the polynomials were treated as unconstrained general polynomials, neglecting the geometric relationship of the PnP problem. Kneip et al. [17] proposed the UPnP method, which takes into account the geometric relationship between known variables in the polynomials when solving with the Gröbner basis. Compared to OPnP, UPnP can obtain fewer solutions while ensuring global optimality of the solutions.

In practical applications, noise is inevitably present, which introduces errors in the projection equations. Some methods, such as EPnP, ignore this error during computation, while our method takes it into account. Unlike other global optimization methods like OpnP and UPnP, our method only returns a single global optimal solution, making it more convenient for practical use. Experimental results demonstrate that our method can robustly handle all experimental scenarios and achieve accuracy comparable to state-of-the-art algorithms.

3 Proposed Method

3.1 Preliminaries of the PnP Problem

The objective of the PnP problem is to determine the rotation matrix R and translation matrix t of a camera, given the known camera intrinsic matrix K, the 3D coordinates $P_i = \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}^T$ of n points in world coordinates, and their corresponding 2D projection points $\begin{bmatrix} u_i^c & v_i^c & 1 \end{bmatrix}^T$. The relationship between the 3D points and 2D points can be expressed by the following equation:

$$\lambda_i \begin{bmatrix} u_i^c \\ v_i^c \\ 1 \end{bmatrix} = K \left(RP_i + t \right), i = 1, 2, \ldots, n \tag{1}$$

where λ_i denotes the depth factor of the i-th point.

3.2 The Minimization Problem

When we have the coordinates of 3D points in world coordinates, the coordinates of their corresponding 2D projected points in camera coordinates, and the camera's intrinsic matrix, we can leverage the projection relationship between the 3D points in camera coordinates and the 2D projected points. This allows us to convert the known coordinates of the 2D projected points into the coordinates of the 3D points in camera coordinates.By solving this projection relationship, we can estimate the camera's pose, which includes its position and orientation in the world coordinate system. we can leverage the projection relationship between the 3D points in camera coordinates and the 2D projected points to transform the known coordinates of the 2D projected points into the coordinates of the 3D

points in camera coordinates.

$$\lambda_i K^{-1} \begin{bmatrix} u_i^c \\ v_i^c \\ 1 \end{bmatrix} = RP_i + t, i = 1, 2, \ldots, n \tag{2}$$

The camera intrinsic matrix K is known.

$$K = \begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

K^{-1} can be obtained by taking the inverse of (3).

$$K^{-1} = \begin{bmatrix} 1/fx & 0 & -cx/fx \\ 0 & 1/fy & -cy/fy \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

When the variables K^{-1} and $\begin{bmatrix} u_i^c & v_i^c & 1 \end{bmatrix}^T$ on the left-hand side of (2) are known, the resulting vector after multiplication still has a last dimension of 1. Therefore, the multiplied form can be expressed as:

$$\lambda_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} r_1^T \\ r_2^T \\ r_3^T \end{bmatrix} P_i + \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}, i = 1, 2, \ldots, n \tag{5}$$

The left-hand side of (5) represents the coordinates of the 3D point in camera coordinates.The (5) can be used to obtain $\lambda_i = r_3^T p_i + t_3$. By substituting λ_i into (5), we can obtain the following expression.

$$\begin{matrix} u_i r_3^T p_i + u_i t_3 = r_1^T p_i + t_1 \\ v_i r_3^T p_i + u_i t_3 = r_2^T p_i + t_2 \end{matrix}, i = 1, 2, \ldots, n \tag{6}$$

Make the right side of the (6) equal to 0.

$$\begin{matrix} r_1^T p_i - u_i r_3^T p_i + t_1 - u_i t_3 = 0 \\ r_2^T p_i - v_i r_3^T p_i + t_2 - u_i t_3 = 0 \end{matrix}, i = 1, 2, \ldots, n \tag{7}$$

To better illustrate the relationship between the unknown variables and the known point coordinates, express (7) in the form of a Kronecker product and write the 3×3 rotation matrix as vector.

$$\left(\begin{bmatrix} 1 & 0 & -u_i \end{bmatrix} \otimes p_i^T \right) \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} + \begin{bmatrix} 1 & 0 & -u_i \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} = 0, i = 1, 2, \ldots, n \tag{8}$$

$$\left(\begin{bmatrix} 1 & 0 & -v_i \end{bmatrix} \otimes p_i^T \right) \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} + \begin{bmatrix} 1 & 0 & -v_i \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} = 0, i = 1, 2, \ldots, n \tag{9}$$

(8) and (9) involve only the unknown variables $\begin{bmatrix} r_1^T & r_2^T & r_3^T \end{bmatrix}^T$ and $\begin{bmatrix} t_1 & t_2 & t_3 \end{bmatrix}^T$, while the rest are known variables. Therefore, we can combine the two equations and explain them in simpler terms.

$$\begin{bmatrix} 1 & 0 & -u_i \\ 0 & 1 & -v_i \end{bmatrix} \otimes p_i^T \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} + \begin{bmatrix} 1 & 0 & -u_i \\ 0 & 1 & -v_i \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} = 0, i = 1, 2, \ldots, n \tag{10}$$

Let's denote the coefficient matrices of $\begin{bmatrix} r_1^T & r_2^T & r_3^T \end{bmatrix}^T$ and $\begin{bmatrix} t_1 & t_2 & t_3 \end{bmatrix}^T$ as M and N, respectively. We can denote X as the $\begin{bmatrix} r_1^T & r_2^T & r_3^T \end{bmatrix}^T$ and t as the $\begin{bmatrix} t_1 & t_2 & t_3 \end{bmatrix}^T$. Equation (10) can be written as follows:

$$MX + Nt = 0 \tag{11}$$

In the previous derivations, we have been working with (1), which represents an ideal case where the equation holds exactly. However, in practical situations, there will always be some level of error or noise present, which means that the equation is not strictly equal. In previous PnP algorithms proposed by researchers, some methods are based on calculations in the ideal case without considering the presence of errors. In our calculations, we take into account the existence of errors and represent the error function as:

$$\begin{aligned} \varepsilon = & \ (MX + Nt)^T (MX + Nt) \\ = & \ X^T M^T MX + \ X^T M^T Nt + \ t^T N^T MX + \ t^T N^T Nt \end{aligned} \tag{12}$$

The equation above has X and t as unknowns. We can consider the error ε as a function of X and t. Taking the partial derivative of the equation with respect to t, $\frac{D_\varepsilon}{D_t} = 2N^T MX + 2N^T Nt$. The expression of t in terms of X can be obtained by setting the derivative to zero.

$$t = \ -\left(N^T N\right)^{-1} N^T MX \tag{13}$$

Substituting Eq. (13) into Eq. (12)

$$\varepsilon = \ X^T \left(M^T M - M^T N \left(N^T N\right)^{-1} N^T M\right) X \tag{14}$$

To facilitate representation, letting $L = M^T M - M^T N \left(N^T N\right)^{-1} N^T M$. (14) can be represented as

$$\varepsilon = X^T LX \tag{15}$$

3.3 Rotation Parameterization

For a rotation matrix R, it must satisfy the orthogonality constraint $RR^T = I$ and the determinant constraint $\det(R) = 1$. There are several parameterization methods for rotation matrices, such as Euler angles, rotation axis-angle, and

quaternion. Here, we adopt the representation of rotation matrices using non-unit quaternion. For regular quaternion$q = [a, b, c, d]^T$, they must satisfy the constraint $q^T q = 1$, while non-unit quaternion are unconstrained and can satisfy the constraints of a regular rotation matrix. The representation of rotation matrices using non-unit quaternion is as follows:

$$R = \frac{1}{s}\begin{bmatrix} a^2+b^2-c^2-d^2 & 2bc-2ad & 2bd+2ac \\ 2bc+2ad & a^2-b^2+c^2-d^2 & 2cd-2ab \\ 2bd-2ac & 2cd+2ab & a^2-b^2-c^2+d^2 \end{bmatrix} \tag{16}$$

In (16), where $s = a^2+b^2+c^2+d^2$ is a four-dimensional vector of unknowns without any constraints. Here,X is a nine-dimensional vector, and it can be observed that each term in is quadratic. We can make the substitution , letting $\tilde{X} = \left[a^2, ab, ac, ad, b^2, bc, bd, c^2, cd, d^2\right]^T$, and perform appropriate row and column transformations, resulting in a 10×10 matrix. By using non-unit quaternions, we transform (15) from a constrained solution to an unconstrained one. (15) can be expressed as follows:

$$\min_{a,b,c,d} \varepsilon = \tilde{X}^T L \tilde{X} \tag{17}$$

3.4 Global Optimization Solution

The error ε is always positive. When ε is minimized, the corresponding solution $\tilde{X}$ to the PnP problem can be obtained by taking partial derivatives of ε with respect to a, b, c, d, resulting in a system of polynomial equations.

$$\frac{\partial \varepsilon}{\partial a} = 0, \frac{\partial \varepsilon}{\partial b} = 0, \frac{\partial \varepsilon}{\partial c} = 0, \frac{\partial \varepsilon}{\partial d} = 0 \tag{18}$$

Solving the system of polynomial equations can yield all the solutions. When solving (18), the same method as UPnP [17] is used, resulting in eight solutions, similar to UPnP. However, we ultimately want to obtain only one globally optimal solution and need to filter the solutions. By substituting the solutions into (17), it is observed that the values obtained from the first four solutions are much smaller than those obtained from the last four solutions. When $n > 3$, the solution with the minimum reprojection error among the first four solutions can be selected as the optimal solution. When $n = 3$, it is noticed that in some cases, there are two different solutions with reprojection errors close to zero. In such cases, we return the solution with a reprojection error close to zero, allowing the user to choose the correct solution based on the actual tracking scenario.

3.5 Root Polishing

To further improve the accuracy, we can optimize the obtained solution using a first-order Newton method. In this case, we calculate the rotation matrix R, replace the original 3D points in world coordinates P_i with RP_i, and recompute

the matrix L. The solution obtained at this point is close to the identity matrix. Therefore, we can use the Cayley parameterization $c = \begin{bmatrix} c_1 & c_2 & c_3 \end{bmatrix}^T$ to represent the rotation matrix. The Cayley parameterization has singularity when the rotation is 180°, but in this case, the rotation angle is close to zero. We can represent $\tilde{X}$ as follows:

$$\tilde{X} = \frac{1}{1 + c_1^2 + c_2^2 + c_3^2} \left[1, c_1^2, c_2^2, c_3^2, c_1, c_2, c_3, c_1c_2, c_1c_3, c_2c_3\right]^T \in R^{10} \tag{19}$$

It is almost trivial and very efficient to compute the 3×1 Jacobian J_ε and the 3×3 Hessian H_ε of (17) around c = 0. Optimizing R through $R \leftarrow R - \delta R$ is obtained by calculating the Jacobian matrix and Hessian matrix through $\delta R = (H_\varepsilon)^{-1} J_\varepsilon$.

4 Experiment Results

In this section, we compare our method, referred to as "RPPnP," with several other PnP algorithms using simulated experiments. We compare our method with the following approaches: EPnP+GN [12] (EPnP when 3D points lie on a plane), RPnP [13], OPnP [15], UPnP [17], LHM [8] (SP+LHM when 3D points lie on a plane or near-singular configurations), and DLS [14].

4.1 Simulated Data

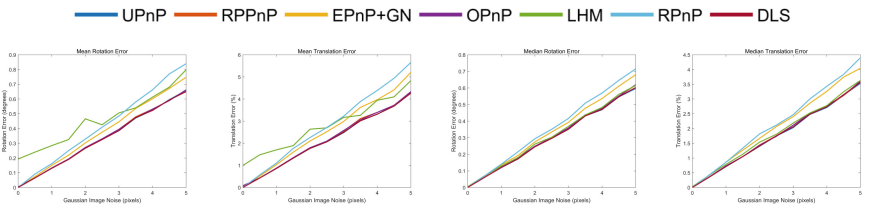
Assuming the focal length of the virtual camera is 800, we generate the coordinates of n 3D points in the world coordinate system. These points are then transformed into the camera coordinate system using randomly generated rotation and translation matrices(R_{true} and T_{true}). Finally, these 3D points are projected onto a two-dimensional plane to obtain their 2D coordinates. Gaussian noise is added to the 2D coordinates based on specific experimental requirements.

We use three configurations for selecting 3D points in world coordinates when evaluating the PnP algorithm:

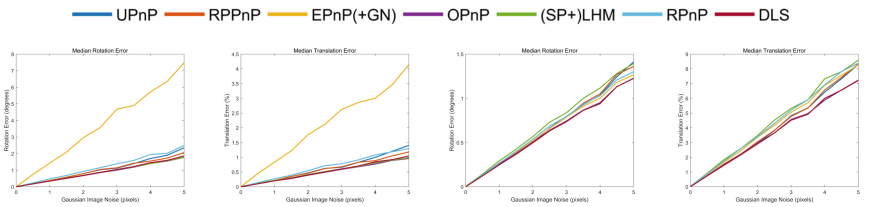
(1) Ordinary Case: $rank(W^TW) = 3$ and the smallest eigenvalue is not close to 0. The range of 3D points is [–2, 2] × [–2, 2] × [4, 8].
(2) Planar Case: $rank(W^TW) = 2$, the range of 3D points is [–2, 2] × [–2, 2] × [0, 0].
(3) Quasi-Singular Case:$rank(W^TW) = 3$ and the largest eigenvalue is close to 0. The range of 3D points is [1, 2] × [1, 2] × [4, 8].

The error between the true values (R_{true} and t_{true})and the estimated values (R and t) of the rotation matrix and translation matrix is calculated as follows:

$$\begin{aligned} e_{rot}(degrees) &= cos^{-1}\left(R_{true}^T R\right) \times \frac{180}{\pi} \\ e_{trans}(\%) &= \frac{||T_{true} - T||}{||T_{true}||} \times 100 \end{aligned} \tag{20}$$

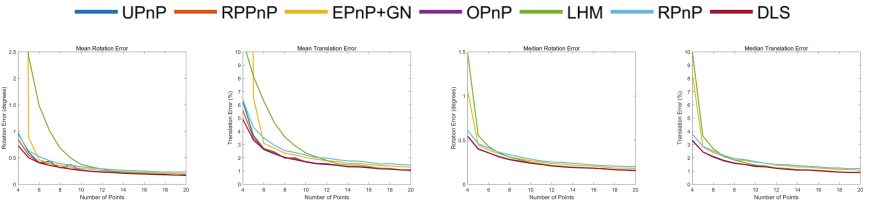


(a) The mean error of R, the mean error of t, the median error of R, and the median error of t when the 3D points are in the ordinary case

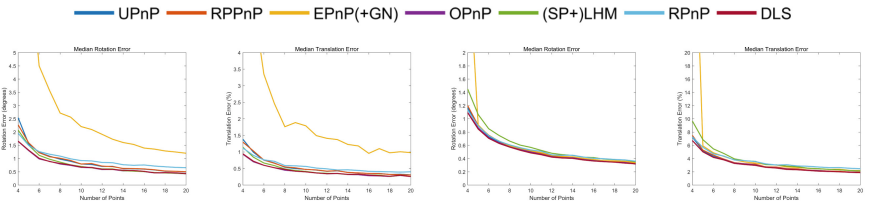


(b) The median error of R and t when 3D points are within the planar case, and the median error of R and t when 3D points are in the quasi-singular case.

Fig. 1. Experiment results with respect to varying noise levels (n = 10) in case of ordinary-3D, quasi-singular and planar point configura-tion, shown in (a) and (b) respectively.



(a) The mean error of R, the mean error of t, the median error of R, and the median error of t when the 3D points are in the ordinary case



(b) The median error of R and t when 3D points are within the planar case, and the median error of R and t when 3D points are in the quasi-singular case.

Fig. 2. Experiment results with respect to varying point numbers (Gaussian Image Noise is 2) in case of ordinary-3D, quasi-singular and planar point configuration, shown in (a) and (b) respectively.

4.2 The Effect with the Varying Noise

This experiment tested the effects of noise on the accuracy of all methods.We fixed $n = 10$ and the noise level added to the 2D point coordinates gradually increased from 0 to 5 (with an increment of 0.5 for each iteration). Figure 1 shows the mean and median errors of the rotation and translation matrices corresponding to the 3D points in the ordinary, quasi-singular, and planar cases. As shown in Fig. 1, as the noise increases, the accuracy of all PnP algorithms decreases, following a roughly linear distribution. Even for $n = 10$, EPnP+GN and RPnP are not accurate enough. Due to the presence of local optima, LHM and SP+LHM have larger mean errors in the general case. RPPnP and UPnP have similar errors in the planar and quasi-singular cases, but they exhibit slightly lower accuracy compared to OPnP and LHM. In the ordinary case, RPPnP and OPnP have similar errors, with a clear advantage in terms of mean error.

4.3 The Effect with the Varying Number of Points

This experiment investigated the performance of all methods with the varying number of points.We add zero-mean Gaussian noise with fixed deviation 2 pixels onto the image projections and gradually increasing the number of generated 3D points from 4 to 20 (increasing by 1 point per iteration). Figure 2 shows the mean and median errors of the rotation and translation matrices corresponding to the 3D points in the general, quasi-singular, and planar cases. As shown in Fig. 2, EPnP is sensitive to the change in the number of points and cannot achieve accurate poses when the number of points is small. LHM has lower accuracy when there are fewer points, possibly due to local minima. RPnP has relatively stable results for both small and large numbers of points, but with lower accuracy. UPnP, as a nonlinear optimization method, achieves high accuracy. RPnP and UPnP have similar errors in the general, planar, and quasi-singular cases.

In the mentioned experiments, only the median error is reported in the planar case and quasi-singular case. This is because in special cases, the error can fluctuate significantly. Additionally, both the PnP problem and its minimization problem, such as the P3P problem, can be handled by RPPnP. This inevitably leads to a decrease in robustness in special cases. Moreover, by looking at the median error, we can observe that RPPnP can achieve high accuracy solutions in most cases.

4.4 Computational Efficiency

In this experiment, the number of points, n, was gradually increased from 6 to 1006 in increments of 100. Each value of n was tested 500 times, and the average time was calculated. Figure 3 shows the computational efficiency of different PnP algorithms, and it is evident that our method has a significant advantage in terms of computational efficiency. Our method is 10 times faster than OPnP. Both our method and UPnP use mex files for implementation.

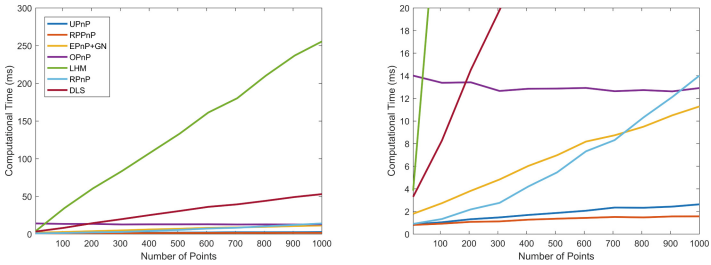


Fig. 3. Running time with respect to varying number of points.

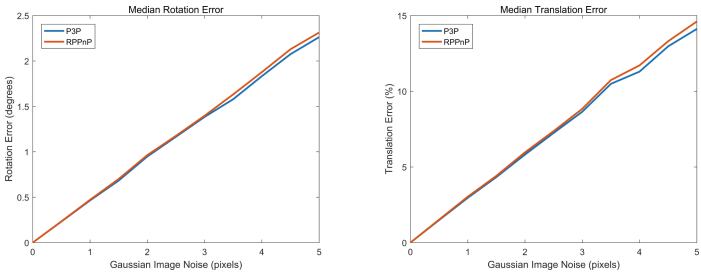


Fig. 4. The median rotation error and the median translation error with respect to varying noise levels in the minimal n = 3.

4.5 Comparison with the P3P Solver

We compared RPnP with the P3P [18] method. Figure 4 shows the median errors of R and t when the noise is increased from 0 to 5. It can be observed that RPnP and the compared P3P method achieve similar accuracy.

5 Conclusion

In this paper, we propose a non-iterative PnP algorithm that establishes a minimization problem based on the relationship between 3D points and their 2D projections. We parameterize the rotation matrix using non-unit quaternions and solve the resulting system of polynomial equations composed of the first-order partial derivatives to obtain the pose. Experimental results demonstrate that our method maintains high accuracy even with a small number of points and remains computationally efficient as the number of points increases.

References

1. Wei, W., Tan, L., Jin, G., Lu, L., Sun, C.: A survey of UAV visual navigation based on monocular SLAM. In: 2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC), pp. 1849–1853. Chongqing, China (2018). https://doi.org/10.1109/ITOEC.2018.8740355

2. O'Mahony, N., et al.: Adaptive multimodal localisation techniques for mobile robots in unstructured environments : a review. In: 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), pp. 799–804. Limerick, Ireland (2019). https://doi.org/10.1109/WF-IoT.2019.8767330
3. Marchand, E., Uchiyama, H., Spindler, F.: Pose estimation for augmented reality: a hands-on survey. IEEE Trans. Vis. Comput. Graph. **22**(12), 2633–2651 (2016)
4. Mur-Artal, R., Tardos, J.D.: ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D cameras. IEEE Trans. Robot. **33**(5), 1255–1262 (2017)
5. Kneip, L., Sweeney, C., Hartley, R.: The generalized relative pose and scale problem: view-graph fusion via 2D-2D registration. In: 2016 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 1–9. Lake Placid, NY, USA (2016). https://doi.org/10.1109/WACV.2016.7477656
6. Yang, J., Li, H., Campbell, D., Jia, Y.: Go-ICP: a globally optimal solution to 3D ICP point-set registration. IEEE Trans. Pattern Anal. Mach. Intell. **38**(11), 2241–2254 (2016)
7. Pedregal, P.: Introduction to Optimization. TAM, vol. 46. Springer, New York (2004). https://doi.org/10.1007/b97412
8. Lu, C.P., Hager, G.D., Mjolsness, E.: Fast and globally convergent pose estimation from video images. IEEE Trans. Pattern Anal. Mach. Intell. **22**(6), 610–622 (2000)
9. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, UK (2003)
10. Abdel-Aziz, Y.I., Karara, H.M., Hauck, M.: Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry. Photogramm. Eng. Remote Sens. **81**(2), 103–107 (2015)
11. Quan, L., Lan, Z.: Linear N-point camera pose determination. IEEE Trans. Pattern Anal. Mach. Intell. **21**(8), 774–780 (1999)
12. Lepetit, V., Moreno-Noguer, F., Fua, P.: EPnP: an accurate O(n) solution to the PnP problem. Int. J. Comput. Vis. **81**, 155–166 (2009)
13. Li, S., Xu, C., Xie, M.: A robust O(n) solution to the perspective-n-point problem. IEEE Trans. Pattern Anal. Mach. Intell. **34**(7), 1444–1450 (2012)
14. Hesch, J.A., Roumeliotis, S.I.: A direct least-squares (DLS) method for PnP. In: 2011 International Conference on Computer Vision, pp. 383–390. Barcelona, Spain (2011). https://doi.org/10.1109/ICCV.2011.6126266
15. Zheng, Y., Kuang, Y., Sugimoto, S., Astrom, K., Okutomi, M.: Revisiting the PnP problem: a fast, general and optimal solution. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 2344–2351. Sydney, Australia (2013). https://doi.org/10.1109/ICCV.2013.291
16. Kukelova, Z., Bujnak, M., Pajdla, T.: Automatic generator of minimal problem solvers. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008. LNCS, vol. 5304, pp. 302–315. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-88690-7_23
17. Kneip, L., Li, H., Seo, Y.: UPnP: an optimal $O(n)$ solution to the absolute pose problem with universal applicability. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8689, pp. 127–142. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10590-1_9
18. Banno, A.: A P3P problem solver representing all parameters as a linear combination. Image Vis. Comput. **70**, 55–62 (2018)