applied sciences

MDPI

Article

An Accurate, Efficient, and Stable Perspective-n-Point Algorithm in 3D Space

Rui Qiao [1], Guili Xu [1,*], Ping Wang [2], Yuehua Cheng [1] and Wende Dong [1]

[1] College of Automation, Nanjing University of Aeronautics and Astronautics, Nanjing 211100, China
[2] College of Electrical and Information Engineering, Lanzhou University of Technology, Lanzhou 730050, China
* Correspondence:guilixu2002@163.com

Abstract: The Perspective-n-Point problem is usually addressed by means of a projective imaging model of 3D points, but the spatial distribution and quantity of 3D reference points vary, making it difficult for the Perspective-n-Point algorithm to balance accuracy, robustness, and computational efficiency. To address this issue, this paper introduces Hidden PnP, a hidden variable method. Following the parameterization of the rotation matrix by CGR parameters, the method, unlike the existing best matrix synthesis technique (Gröbner technology), does not require construction of a larger matrix elimination template in the polynomial solution phase. Therefore, it is able to solve CGR parameter rapidly, and achieve an accurate location of the solution using the Gauss–Newton method. According to the synthetic data test, the PnP algorithm solution, based on hidden variables, outperforms the existing best Perspective-n-Point method in accuracy and robustness, under cases of Ordinary 3D, Planar Case, and Quasi-Singular. Furthermore, its computational efficiency can be up to seven times that of existing excellent algorithms when the spatially redundant reference points are increased to 500. In physical experiments on pose reprojection from monocular cameras, this algorithm even showed higher accuracy than the best existing algorithm.

Keywords: Perspective-n-Point problem; CGR parameter matrix; Gröbner technology; hidden PnP

Citation: Qiao, R; Xu, G.; Wang, P.; Cheng, Y.; Dong, W. An Accurate, Efficient, and Stable Perspective-n-Point Algorithm in 3D Space. *Appl. Sci.* **2023**, *13*, 1111. https://doi.org/10.3390/app13021111

Academic Editors: Miguel Cazorla, Félix Escalona Moncholí and Francisco Gomez-Donoso

Received: 5 December 2022
Revised: 7 January 2023
Accepted: 11 January 2023
Published: 13 January 2023

1. Introduction

In the field of computer vision, the Perspective-n-Point(PnP) problem refers to the determination of the relative position and gesture relation between the target and the camera by combining the image coordinates obtained from the camera with the coordinates of n known feature points of the target in the world coordinate system [1]. With the boom in robotics and space technology in recent years, the PnP problem has been widely studied and applied among fields including autonomous robot navigation [2], robot positioning [3], hand-eye calibration [4], spacecraft rendezvous and docking [5], target recognition and tracking [6], augmented reality, and SLAM [7].

The PnP problem is solvable when $n \geq 3$, and it cannot be solved when $n \leq 2$. When $n = 3$, P3P can be regarded as a minimal subset of the PnP problem. To be exact, when $n \geq 4$, it is called a general PnP problem, which exhibits higher pose estimation accuracy than P3P because of the existence of more redundant information points. The ideal PnP algorithm features accuracy, stability, high efficiency, and universality. Instead, due to the difference of scenes,the PnP algorithm fails to achieve high-precision computational results with high efficiency and stability in actual practice.

1.1. Pnp Problems Solved by Linear and Non-Linear Methods

Table 1 shows that PnP algorithms can be divided into two main categories. The first is the linear method for the PnP problem, including the Tsai method [8], the HOMO method [9], and the direct linear transformation method (DLT) [10], etc. The linear method performs with fast solving speed, as the mapping relation between spatial reference points

and images is adopted to establish the equation, and the linear method is used to solve the equation directly. Nevertheless, such a method exhibits larger error and is prone to noise interference when the number of spatial reference points n $\geq$ 4 and the spatial reference points are not in the same plane, which can be mostly explained by the failure to consider non-linear constraints in the imaging process. The corresponding P4P and P5P algorithms, when n = 4 and n = 5, proposed by some scholars lack the universal feature of the PnP algorithm. To avoid the defects of the linear method for the PnP problem, the non-linear method is adopted for some other PnP algorithms, such as in the literature in [11–19]. Among the examples from the literature, refs. [11,12] employed the SoftPOSIT method and the orthogonal iteration method, respectively. The results demonstrated that the LHM algorithm outperformed the iterative methods for PnP problem. The LHM algorithm reduces the number of unknowns by parameterizing the imaging model, takes the residual sum of squares of the reference coordinates in the world coordinate system as the loss function, and obtains the parameters of the rotation matrix and translation matrix by means of iterative solution. Despite its better performance than the linear algorithm, in terms of computational accuracy and stability, the iterative algorithm may increase the computation burden, and even lead to non-convergence and local optimum of the calculation results if its initial value is beyond a certain range of the real solution; thus, failing the algorithm. In addition, the iterative method performs low computational accuracy and stability if the distribution of spatial reference points is similar to Quasi Singular and there are few redundant feature points in the distribution.

1.2. A Non-Iterative Approach to Optimally Solving PnP Problems

Table 1 shows that scholars have used non-iterative methods to optimally solve the PnP problem in order to overcome the shortcomings of the iterative method. Firstly, the imaging model undergoes parameterization, simplification, and deformation to construct non-linear equations that can be used for optimization. Then, the Gröbner technique (Figure 1) is adopted to solve the non-linear equations (see [20,21] in detailed steps of Gröbner tool), and the required pose parameters are obtained. The first non-iterative PnP algorithm is the EPnP algorithm [13], which takes four spatial virtual points as reference points and uses these four points to represent the reference points in the rest of the space, and, finally, obtains the pose information through optimization. The EPnP algorithm requires less computational time, compared with the iterative method. However, the linear method it uses leads to unsatisfactory computational accuracy and poor stability when there is noise and to fewer spatial redundant points. The DLS algorithm [14], also a non-iterative method, first introduces the CGR (Cayley–Gibbs–Rodriguez) parameterized rotation matrix, and uses the matrix decomposition technique to process the rotation matrix. The processed rotation matrix parameterizes the translation vector through the relation derivation of the imaging model equation, leaving only three unknown parameters in the whole solution process. Finally, the pose estimation problem is transformed into an unconstrained square minimization problem by constructing a cost function. The introduction of the CGR parameterized rotation matrix and the application of the matrix decomposition technique in the solution process may incur singular value and complex computation, which greatly affects the accuracy, speed, and stability of the final solution pose. Among the non-iterative methods, the RPnP algorithm [15], which is different to the DLS algorithm, can be generally regarded as a phased solution method. First, the PnP problem is transformed into the P3P problem, that is, the spatial reference points are divided into groups of three, so as to obtain polynomial sets with the highest order of fourth. Then, the fourth-order polynomial sets, also non-linear equations, are transformed into linear equations through the linearization technique. Finally, the required pose parameters are obtained by solving the polynomial sets. The whole solution process of RPnP, though being a progressive and cohesive strategy, introduces linearization technology and adopts the non-global optimization method, which explains its fast computational speed, but less favorable stability and accuracy.

Table 1. The development of each PnP algorithm. (Note: Y means adopted, N means not adopted, L means Low, M means Median, H means Hight, + means better than this level, but worse than the next level.).

Algorithm Name	DLT	HOMO	Tsai	LHM	POSIT	EPnP	DLS	RPnP	ASPnP	OPnP	UPnP	optDLS	SRPnP	RDLT	WIEPnP
Time	1971	1981	1987	2000	2002	2009	2011	2012	2013	2013	2014	2015	2018	2020	2020
Linear Method	Y	Y	Y	N	N	N	N	N	N	N	N	N	N	Y	N
Optimization method	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y
Iterative method	N	N	N	Y	Y	N	N	N	N	N	N	N	Y	N	Y
Accuracy	L	L	L	H	M+	M	M+	H	H+	H+	H	H+	H+	H+	H+
Efficiency	H+	H+	H	L	L	H	M	H	M	M	M+	M+	H	H	H

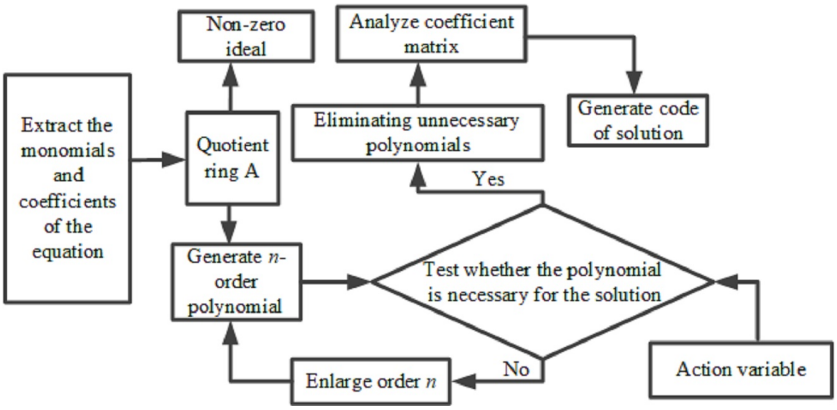


Figure 1. Schematic diagram of Gröbner generation framework.

In addition, some PnP algorithms not only absorb the advantages of iterative methods, but also overcome their proneness to local optimum, including the ASPnP algorithm [16] and the OPnP [17] algorithm. Both of the aforementioned algorithms introduce the quaternion to represent the rotation matrix, express the translation matrix by the quaternion parameters through the imaging model equation, and directly construct the objective function for global optimization by means of such mathematical skills as rewriting the matrix form and introducing new variables. What merits attention, is that the method employs the Gröbner technique, instead of the Gauss–Newton method commonly used in the iterative method, to avoid local optimum when solving the third-order polynomial sets of the objective function. Nevertheless, the above two PnP algorithms construct a large elimination template matrix when using the Gröbner technique, which increases computational burden.

Some scholars have made continuous efforts to improve existing PnP algorithms. For example, Kneip et al. proposed the UPnP in 2014 [18], and Nakano proposed the optDLS [19] PnP algorithm in 2015, both of which improved the OPnP algorithm. Specifically, the UPnP algorithm applies to PnP problems under the perspective imaging model and those under the General Camera Model. The UPnP algorithm cannot match the OPnP algorithm in the Planar Case and Quasi Singular of reference points. The optDLS algorithm reduces the number of solutions and the running time of the algorithm by increasing the constraints of the objective function in OPnP. In 2018, Wang et al. [22] proposed the SRPnP algorithm, based on the RPnP algorithm. The algorithm only requires solving a univariate polynomial set of the highest order of 7 and a univariate polynomial of the highest order of 4, and, finally, uses the Gauss–Newton method for one accurate positioning. This algorithm divides the solution of PnP into three stages, each of which witnesses the formation of an optimization problem, and utilizes the optimization value obtained in the previous stage for computation. In this way, the SRPnP algorithm is more accurate and faster than the RPnP in solution, elevating it into being one of the best algorithms for solving the PnP problem. Wang also proposed the RDLT algorithm in 2020 [23]. This algorithm, while using the linear method to solve the PnP problem, improves the DLT algorithm by considering the imaging constraint relationship between points, which outperforms the traditional linear method in stability and accuracy, rivals the existing non-linear optimization method, and even exhibits far higher solving efficiency than the non-linear optimization method. In 2020, Wang et al. [24] proposed the WIEPnP algorithm, which adopted an iterative solution by setting the weight coefficient of the spatial reference points. The algorithm is still on a par with OPnP and SRPnP as to accuracy, without significant improvement.

1.3. Problems with Existing PnP Algorithms and Solutions

The above demonstrates that, among the PnP algorithms, except for the RDLT algorithm, the linear solution method is not as efficient, accurate, and stable as the non-linear solution method. The non-linear method, for solutions, adopts Gauss–Newton method, the

Levenberg–Marquardt method for iteration, the orthogonal iteration method, and other iterative algorithms to solve the PnP problem. However, the selection of the initial value affects the accuracy, stability, and computational efficiency of the algorithms, which explains their less favorable performance compared with the non-iterative method. Among non-iterative methods, the PnP algorithm, with the best performance, introduces CGR (Cayley–Gibbs–Rodriguez) to parameterize the rotation matrix, and utilizes the parametrization of the imaging model equation to translate the vector. The Gröbner is employed to solve pose parameters, which avoids lengthy computational duration and local optimum caused by the unreasonable initial values of the iterative method. Instead, the polynomial sets formed by the PnP problem, are generally rather complex, and the application of the Gröbner technique to solve the polynomial sets requires the construction of a large elimination template matrix, which enlarges the computational burden to solve the PnP problem, and affects the real-time performance of the PnP algorithm in practical applications. In order to make up for the shortcomings of the above methods, and to improve the robustness, accuracy, efficiency, and universality of the PnP algorithm, this paper proposes the Hidden PnP, a method to solve polynomial sets based on hidden variables, which guarantees higher accuracy, stability, and efficiency in solving the PnP problem. The main contributions of this paper are as follows:

(1) To prevent singularities in the rotation matrix parametrized by the CGR, a rotation matrix, R1, is constructed, and the rotation matrix, R, parametrized by the CGR is processed using R1.
(2) In this paper, accuracy experiments and noise immunity experiments of the hidden variable method were carried out in three cases, i.e., Planar Case, Ordinary 3D and Quasi-Singular. The performance of some new PnP algorithms in recent years were compared and detailed experimental data given.
(3) In this paper, the hidden variable-based PnP algorithm, and other PnP algorithms, were applied to the physical experiments with monocular vision cameras, and reprojection experiments of the corner points of calibration plates were carried out. In addition, detailed reprojection experimental error data for each PnP algorithm are given.

In the second section, the basic mathematical description of the PnP problem is given and the mathematical solution process of the hidden variables is described in detail. In the third section, simulation experiments and results analysis are carried out on Matlab software using synthetic data. In this section, the posited solution performance of the proposed hidden PnP method is compared with our Hidden PnP method in three cases, namely the Ordinary 3D case, the Planar Case and the Quasi-Singular case, as well as in the case of noise under these three cases. In addition, the section gives experimental data comparing the operational efficiency of various algorithms, and,finally, the algorithms are applied to the physical experiment of the reprojection error. Section 4 presents the conclusions and outlook.

2. Methods

2.1. Description of the PnP Mathematical Model

The imaging model of a camera projecting n 3D reference points under a 3D world onto a 2D image plane through a lens can be represented by the model in Figure 2. The equations of the projected imaging model from the world coordinate system to camera one can be expressed in the following form:

$$\lambda_i P_i = R q_i + t, (i = 1, 2, 3, \ldots, n) \tag{1}$$

where $q_i = [X_i, Y_i, Z_i]^T$, (i = 1, 2, 3, …, n) are the coordinates of n 3D spatial reference points in the world coordinate system, $P_i = [u_i, v_i, 1]^T$, (i = 1, 2, 3, …, n) are the plane coordinates of n points in the camera coordinate system, and $t = [t_x, t_y, t_z]^T$ is the translation vector.

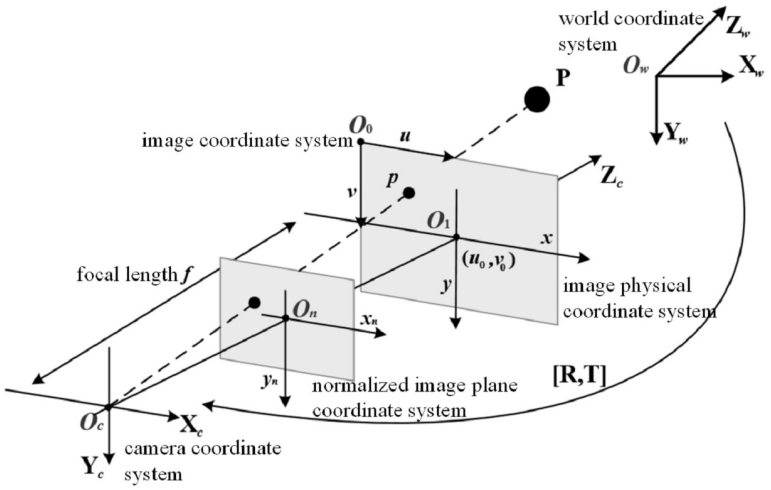


Figure 2. Coordinate system diagram of camera imaging model.

2.2. Parametric Representation of Rotation Matrices and Translation Vectors, and the Elimination of Depth Factors

Mathematically, the rotation matrix R is represented in more than one way, including direction cosine algorithm, Euler angle representation, unit quaternion representation, Cayley parameter representation (CGR), etc. This paper mainly employed the Cayley parameter representation (CGR), which is directly derived from the quaternion method. For example, the rotation matrix R, represented by the quaternion, is:

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

Among them, $(s = 1 + b^2 + c^2 + d^2)$. The rotation angle closing to 180° leads to an extremely large rotation matrix, represented by CGR, leading to a singular value in the R matrix and a large calculation error. In response, a rotation matrix R_1 is constructed in the process of code implementation, R_1 is adopted to process the rotation matrix R, parameterized by CGR, and R is restored to its original state after obtaining the final result. In this way, the singularity caused by R parameterized by CGR is effectively avoided.

Equation (1) is rewritten into matrix form:

$$\lambda_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = Rq_i + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}, (i = 1, 2, 3, \ldots, n) \tag{3}$$

Meanwhile, the U matrix in Equation (2) is set to be:

$$U = \begin{bmatrix} r_1^T \\ r_2^T \\ r_3^T \end{bmatrix} = \begin{bmatrix} 1+b^2-c^2-d^2 & 2bc-2d & 2bd+2c \\ 2bc+2d & 1-b^2+c^2-d^2 & 2cd-2b \\ 2bd-2c & 2cd+2b & 1-b^2-c^2+d^2 \end{bmatrix}$$

Multiply both sides of Equation (3) by s to obtain:

$$\hat{\lambda}_i P_i = \begin{bmatrix} r_1^T \\ r_2^T \\ r_3^T \end{bmatrix} q_i + \hat{t}, (i = 1, 2, 3, \ldots, n) \tag{4}$$

where $\hat{\lambda}_i = \lambda_i s$, $P_i = [u_i, v_i, 1]^T$, $\hat{t}_i = s[t_x, t_y, t_z]^T = [\hat{t}_1, \hat{t}_2, \hat{t}_3]^T$.

According to Equation (4), the depth factor satisfies:

$$\hat{\lambda}_i = r_3^T q_i + \hat{t_3}, (i = 1, 2, 3, \ldots, n) \tag{5}$$

Therefore, Equation (5) is substituted into Equation (4) to eliminate all depth factors:

$$\hat{t_1} - \hat{t_3} u_i = r_3^T q_i u_i - r_1^T q_i \tag{6}$$

$$\hat{t_2} - \hat{t_3} v_i = r_3^T q_i v_i - r_2^T q_i \tag{7}$$

The new variable $L = [1, b, c, d, b^2, bc, bd, c^2, cd, d^2]^T$is introduced, and Equations (6) and (7) are rewritten into matrix forms:

$$A_i \hat{t} = \hat{N}_i L \tag{8}$$

where $A_i = \begin{bmatrix} 1 & 0 & -u_i \\ 0 & 1 & -v_i \end{bmatrix}$,

$$\hat{N}_i = \begin{bmatrix} -X_i + Z_i u_i & 2Y_i u_i & -2Z_i - 2X_i u_i & 2Y_i & -X_i - Z_i u_i & -2Y_i & 2X_i u_i - 2Z_i & X_i - Z_i u_i & 2Y_i u_i & X_i + Z_i u_i \\ -Y_i + Z_i v_i & 2Z_i + 2Y_i v_i & -2X_i v_i & -2X_i & Y_i - Z_i v_i & -2X_i & 2X_i v_i & -Y_i - Z_i v_i & 2Y_i v_i - 2Z_i & Y_i + Z_i v_i \end{bmatrix}$$

Since n spatial reference points all satisfy Equation (8), it can be obtained as follows:

$$\begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ \vdots \\ A_n \end{bmatrix} \hat{t} = \begin{bmatrix} N_1 \\ N_2 \\ N_3 \\ \vdots \\ N_n \end{bmatrix} v \Leftrightarrow A\hat{t} = NL \tag{9}$$

In the case that L has been specified, the linear least square fitting is used to obtain:

$$\hat{t} = A^+ NL \tag{10}$$

where $A^+ = (A^T A)^{-1} A^T$ is the inverse matrix of Moore-Penrose matrix.

Equation (10) is substituted into Equation (8) to obtain:

$$(A_i A^+ N_i - N_i) L = 0 \Leftrightarrow \begin{bmatrix} A_1 A^+ N - N_1 \\ A_2 A^+ N - N_2 \\ A_3 A^+ N - N_3 \\ \vdots \\ A_n A^+ N - N_n \end{bmatrix} L = 0 \tag{11}$$

Let $J = \begin{bmatrix} A_1 A^+ N - N_1 \\ A_2 A^+ N - N_2 \\ A_3 A^+ N - N_3 \\ \vdots \\ A_n A^+ N - N_n \end{bmatrix}$, Equation (11) is rewritten as:

$$JL = 0 \tag{12}$$

In Equation (12), the J matrix of the following scale can be obtained through the above calculation, based on the known parameters in Equation (1):

$$J = \begin{bmatrix} j_{11} & j_{12} & j_{13} & j_{14} & j_{15} & j_{16} & j_{17} & j_{18} & j_{19} & j_{1,10} \\ j_{21} & j_{22} & j_{23} & j_{24} & j_{25} & j_{26} & j_{27} & j_{28} & j_{29} & j_{2,10} \\ j_{31} & j_{32} & j_{33} & j_{34} & j_{35} & j_{36} & j_{37} & j_{38} & j_{39} & j_{3,10} \\ & & & & \vdots & & & & & \\ j_{2n,1} & j_{2n,2} & j_{2n,3} & j_{2n,4} & j_{2n,5} & j_{2n,6} & j_{2n,7} & j_{2n,8} & j_{2n,9} & j_{2n,10} \end{bmatrix}$$

Some existing PnP methods adopt the obtained Equation (12) to construct the objective function, and take the Gauss–Newton method, Levenberg–Marquardt iteration, orthogonal iteration, and other iterative methods, to solve the pose parameters. According to Section 1, the selection of the initial value and proneness to local optimum of the iterative methods undermine their calculation accuracy and speed. In addition, some PnP methods also use the Gröbner technique to solve the polynomial sets constructed here, which overcomes the defects of the iterative methods. Instead, its operational process generally requires a huge elimination template matrix, given the complexity of PnP problems, which seriously affects the efficiency of the algorithm. Accordingly, Section 4 employs a hidden variable method to solve the pose parameters of CGR in Equation (12).

2.3. The Application of Hidden Variable Method to Solve Pose Parameters

Due to the numerous polynomial manipulations, the simplification, and the deformation involved in this section, the application of a general calculation makes the derivation process rather complicated. Therefore, the extensively used symbolic operation in matlab software is adopted in this paper to simplify the mathematical description process of hidden variables, so that readers can easily and intuitively understand the derivation process.

Among them, $J(:i)$, i = 1, 2, 3, …, 10 denotes all elements of the ith column of the J matrix. In the hidden variables, $L = [1, b, c, d, b^2, bc, bd, c^2, cd, d^2]^T$, the new variables introduced in the previous section, are adopted, where b is set as a constant, and the remaining parameters are taken as unknowns. Then, the unknowns in Equation (12) are introduced as new variables to obtain the following equation in the form of a matrix:

$$Q \begin{bmatrix} c^2 \\ d^2 \\ cd \end{bmatrix} = -T(b) \begin{bmatrix} c \\ d \\ 1 \end{bmatrix} \tag{13}$$

where Q = [J(:8), J(:10), J(:9)],

$$T(b) = \begin{bmatrix} J_{13} + J_{16}b & J_{14} + J_{17}b & J_{11} + J_{12}b + J_{15}b^2 \\ J_{23} + J_{26}b & J_{24} + J_{27}b & J_{21} + J_{22}b + J_{25}b^2 \\ J_{33} + J_{36}b & J_{34} + J_{37}b & J_{31} + J_{32}b + J_{35}b^2 \\ \vdots & \vdots & \vdots \\ J_{2n,3} + J_{2n,6}b & J_{2n,4} + J_{2n,7}b & J_{2n,1} + J_{2n,2}b + J_{2n,5}b^2 \end{bmatrix}$$

According to Equation (13), c^2, d^2, and cd is solved through the following mathematical calculation, exhibiting the following closed-form solution:

$$\begin{bmatrix} c^2 \\ d^2 \\ cd \end{bmatrix} = GT(b) \begin{bmatrix} c \\ d \\ 1 \end{bmatrix} \Leftrightarrow \begin{bmatrix} c^2 \\ d^2 \\ cd \end{bmatrix} = W(b) \begin{bmatrix} c \\ d \\ 1 \end{bmatrix} \tag{14}$$

where $G = -(Q^TQ)^{-1}Q^T$. The calculation results show that G is a $3 \times 2n$ pure numerical matrix, and $W(b)$ is a 3×3 polynomial matrix containing b. At the same time,

$$G = \begin{bmatrix} g_{11} & g_{12} & g_{13} & \cdots & g_{1,2n} \\ g_{21} & g_{22} & g_{23} & \cdots & g_{2,2n} \\ g_{31} & g_{32} & g_{33} & \cdots & g_{3,2n} \end{bmatrix}$$

$$W(b) = \begin{bmatrix} w_{12}+w_{11}b & w_{14}+w_{13}b & w_{17}+w_{16}b+w_{15}b^2 \\ w_{22}+w_{21}b & w_{24}+w_{23}b & w_{27}+w_{26}b+w_{25}b^2 \\ w_{32}+w_{31}b & w_{34}+w_{33}b & w_{37}+w_{36}b+w_{35}b^2 \end{bmatrix} = \begin{bmatrix} W_{11} & W_{12} & W_{13} \\ W_{21} & W_{22} & W_{23} \\ W_{31} & W_{32} & W_{33} \end{bmatrix}$$

where $w_{11} = G(1:)J(:6)$, $w_{21} = G(2:)J(:6)$, $w_{31} = G(3:)J(:6)$,
$w_{12} = G(1:)J(:3)$, $w_{22} = G(2:)J(:3)$, $w_{32} = G(3:)J(:3)$,
$w_{13} = G(1:)J(:7)$, $w_{23} = G(2:)J(:7)$, $w_{33} = G(3:)J(:7)$,
$w_{14} = G(1:)J(:4)$, $w_{24} = G(2:)J(:4)$, $w_{34} = G(3:)J(:4)$,
$w_{15} = G(1:)J(:5)$, $w_{25} = G(2:)J(:5)$, $w_{35} = G(3:)J(:5)$,
$w_{16} = G(1:)J(:2)$, $w_{26} = G(2:)J(:2)$, $w_{36} = G(3:)J(:2)$,
$w_{17} = G(1:)J(:1)$, $w_{27} = G(2:)J(:1)$, $w_{37} = G(3:)J(:1)$.

Therefore, Equation (14) can be written as:

$$\begin{cases} c^2 = cW_{11} + dW_{12} + W_{13} \\ d^2 = cW_{21} + dW_{22} + W_{23} \\ cd = cW_{31} + dW_{32} + W_{33} \end{cases} \tag{15}$$

By observing the relationship between the unknowns c, d, c^2, d^2, and cd in Equation (14), the following three identity constraint relations are established:

$$\begin{cases} c^2 d = (cd)c \\ (cd)d = d^2 c \\ (cd)(cd) = (c^2)(d^2) \end{cases} \tag{16}$$

Equation (15) is substituted into Equation (16) to obtain:

$$\begin{cases} (cW_{11} + dW_{12} + W_{13})d = (cW_{31} + dW_{32} + W_{33})c \\ (cW_{31} + dW_{32} + W_{33})d = (cW_{21} + dW_{22} + W_{23})c \\ (cW_{31} + dW_{32} + W_{33})^2 = (cW_{11} + dW_{12} + W_{13})(cW_{21} + dW_{22} + W_{23}) \end{cases} \tag{17}$$

Equation (17) can be expanded as follows:

$$\begin{cases} s_{11}c^2 + s_{12}cd + s_{13}c + s_{14}d^2 + s_{15}d = 0 \\ s_{21}c^2 + s_{22}cd + s_{23}c + s_{24}d^2 + s_{25}d = 0 \\ s_{31}c^2 + s_{32}cd + s_{33}c + s_{34}d^2 + s_{35}d + s_{36} = 0 \end{cases} \tag{18}$$

where $s_{11} = -W_{31}$, $s_{12} = W_{11} - W_{32}$, $s_{13} = -W_{33}$, $s_{14} = W_{12}$, $s_{15} = W_{13}$,
$s_{21} = -W_{21}$, $s_{22} = W_{31} - W_{22}$, $s_{23} = -W_{23}$, $s_{24} = W_{32}$, $s_{25} = W_{33}$,
$s_{31} = W_{31}^2 - W_{11}W_{21}$, $s_{32} = 2W_{31}W_{32} - W_{12}W_{21} - W_{11}W_{22}$,
$s_{33} = 2W_{31}W_{33} - W_{13}W_{21} - W_{11}W_{23}$, $s_{34} = W_{32}^2 - W_{12}W_{22}$,
$s_{35} = 2W_{32}W_{33} - W_{13}W_{22} - W_{12}W_{23}$, $s_{36} = W_{33}^2 - W_{13}W_{23}$

Since Equation (18) still contains c^2, d^2, and cd coefficient, Equation (15) is substituted into Equation (18) to obtain:

$$(s_{13} + s_{11}W_{11} + s_{14}W_{21} + s_{12}W_{31})c + (s_{15} + s_{11}W_{12} + s_{14}W_{22} + s_{12}W_{32})d + s_{11}W_{13} + s_{14}W_{23} + s_{12}W_{33} = 0 \tag{19}$$

$$(s_{23} + s_{21}W_{11} + s_{24}W_{21} + s_{22}W_{31})c + (s_{25} + s_{21}W_{12} + s_{24}W_{22} + s_{22}W_{32})d + s_{21}W_{13} + s_{24}W_{23} + s_{22}W_{33} = 0 \tag{20}$$

$$(s_{33} + s_{31}W_{11} + s_{34}W_{21} + s_{32}W_{31})c + (s_{35} + s_{31}W_{12} + s_{34}W_{22} + s_{32}W_{32})d + s_{36} + s_{31}W_{13} + s_{34}W_{23} + s_{32}W_{33} = 0 \tag{21}$$

At this time, Equations (19)–(21) are jointly written into a matrix form:

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} c \\ d \\ 1 \end{bmatrix} = 0 \Leftrightarrow H(b) \begin{bmatrix} c \\ d \\ 1 \end{bmatrix} = 0 \tag{22}$$

where $h_{11} = s_{13} + s_{11}W_{11} + s_{14}W_{21} + s_{12}W_{31}$,
$h_{12} = s_{15} + s_{11}W_{12} + s_{14}W_{22} + s_{12}W_{32}$,
$h_{13} = s_{11}W_{13} + s_{14}W_{23} + s_{12}W_{33}$,
$h_{21} = s_{23} + s_{21}W_{11} + s_{24}W_{21} + s_{22}W_{31}$,
$h_{22} = s_{25} + s_{21}W_{12} + s_{24}W_{22} + s_{22}W_{32}$,
$h_{23} = s_{21}W_{13} + s_{24}W_{23} + s_{22}W_{33}$,
$h_{31} = s_{33} + s_{31}W_{11} + s_{34}W_{21} + s_{32}W_{31}$,
$h_{32} = s_{35} + s_{31}W_{12} + s_{34}W_{22} + s_{32}W_{32}$,
$h_{33} = s_{36} + s_{31}W_{13} + s_{34}W_{23} + s_{32}W_{33}$.

According to Equation (22),

$$det(H(b)) = 0 \tag{23}$$

After the operation and arrangement of Equation (23) with Matlab software, a polynomial equation, with the highest degree of 8 and only the unknown b, can, finally, be obtained:

$$E_8b^8 + E_7b^7 + E_6b^6 + E_5b^5 + E_4b^4 + E_3b^3 + E_2b^2 + E_1b = 0 \tag{24}$$

where E_i (i = 0, 1, 2, 3, …, 8) is the known coefficient. The eigenvalue method is used to calculate the root of b, which has at most 8 roots. The value of c and d can be obtained by substituting b back into Equations (19)–(21), and, thus, clarifying the value of the rotation matrix R and translation vector t.

2.4. Precise Positioning Based on Gauss–Newton Iterative Method

In the previous section, the solution x is obtained with the help of Equation (24), but Equation (11) is not strictly satisfied, due to the presence of noise. As the solution, after using the hidden variable method, is very close to the optimal solution, the solution x is further accurately located thanks to the application of the Gauss–Newton method to solve the least square method. The iteration times of the Gauss–Newton method are no more than 2, given the optimal initial solution. Therefore, the pose measurement problem is taken as an optimization problem again, making the pose solution as close to the global optimal value as possible through precise positioning. The following mathematical description is given:

$$x' = argmin(x^T F^T F x)$$

The F^TF is the constructed 10×10 symmetric matrix, and the iterative equation of x is $x = x + \Delta x$, where $\Delta x = -[(F_J)^T(F_J) + \lambda I_{3\times3}]^{-1}(F_J)^T F x$, λ is the damping factor, and F_J is the Jacobian matrix of F, having expression:

$$F_J^T = \begin{bmatrix} 0 & 1 & 0 & 0 & 2b & c & d & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & b & 0 & 2c & d & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & b & 0 & c & 2d \end{bmatrix}$$

3. Experiments and Analysis

3.1. Testbed and Test Synthesis Data Generation

This section describes the Matlab simulation experiments conducted on Hidden PnP and the existing classical PnP algorithm, based on synthetic data, in terms of solving accuracy, anti-noise ability, and operational efficiency. The experimental results are compared and analyzed. The PnP algorithms involved in the comparison were LHM, EPnP+GN, RPnP, DLS, OPnP, SRPnP, RDLT, and ASPnP, etc. The simulation experiment was completed on a 4-core CPU model i5-4200H notebook with the main frequency of 2.80 GHz and a running memory of 8 G.

A virtual camera, with an image size of 640 × 480 pixels and a focal length of 800 pixels, was constructed on the Matlab platform. When generating random 3D reference points, the matrix composed of 3D reference points was set as $Q = [q_1^w, q_2^w, q_3^w, \cdots, q_n^w,]^T$, as the solving accuracy of PnP algorithm was affected by the distribution of 3D spatial points, so as to reasonably test the solving accuracy of each PnP algorithm. According to the different rank of the matrix Q^TQ, the following three situations could be selected to randomly generate 3D reference points:

(1) Ordinary 3D: rank(Q^TQ) = 3, in which 3D reference points were randomly generated within the range of [−2,2] × [−2,2] × [4,8]. In this case, the minimum eigenvalue was almost close to 0.

(2) Planar Case: rank(Q^TQ) = 2 in which 3D reference points were randomly generated within the range of [−2,2] × [−2,2] × [0,0]. In this case, all reference points were on the same plane.

(3) Quasi-Singular: rank(Q^TQ) = 3, in which 3D reference points were randomly generated within the range of [1,2] × [1,2] × [4,8], and the ratio of the minimum eigenvalue to the maximum eigenvalue was less than 0.05.

After generating the above initial 3D spatial reference points, a rotation matrix and a translation vector were first generated randomly as the truth values of the simulation experiment, which were respectively R_{ture} and T_{ture}. Secondly, the generated 3D spatial reference points were converted from the camera coordinate system to the world coordinate system. Finally, different PnP algorithms were adopted to convert the reference points in the world coordinate system, obtained in the previous step, to the image plane, and, thus, obtaining the estimated rotation matrix R_{est} and translation vector T_{est}. In order to be consistent with the definition of evaluation error in existing literature, this paper adopted the definition of error in [17]:

$$e_{rot}(degree) = \max_{k \in 1,2,3} cos^{-1}(r_{k,true}^T r_{k,est}^T) \times 180/\pi \tag{25}$$

$$e_{rot}(\%) = \frac{\| t_{est} - t_{true} \|}{\| t_{true} \|} \times 100 \tag{26}$$

The performance of PnP algorithm was certainly immune from the definitions adopted.

3.2. Comparative Simulation Test of the Calculation Accuracy of PnP Methods under Different Circumstances

In this section, the method described in the previous section was adopted to generate 4 to 20 3D spatial reference points under the cases of Ordinary 3D, Planar Case, and Quasi-Singular, respectively, which served as the experimental input data of the PnP algorithm to be tested. At the same time, zero-mean Gauss noise, with a size of δ = 2 pixels, was added to the test image. In the test, each PnP algorithm was independently tested 500 times under a fixed number of 3D reference points. Due to the large number of algorithms and the data, the following graph could only be used for macroscopic performance comparison, and readers can refer to Appendix A (Tables A1–A3) for the specific test data. Figures 3–5 show the comparison of the mean errors (a,c) and median errors (b,d) of the rotation matrix and the translation vector in the Ordinary 3D, Planar, and Quasi-Singular cases, respectively, with spatial reference points n ranging from 4 to 20, and δ = 2. For all figure groups, the four sub-figures are: a. mean error of the rotation matrix; b. median error of rotation matrix; c. mean error of translation vector; d. median error of translation vector. To avoid repetitive expressions, only the different cases are given in the title of the figure.

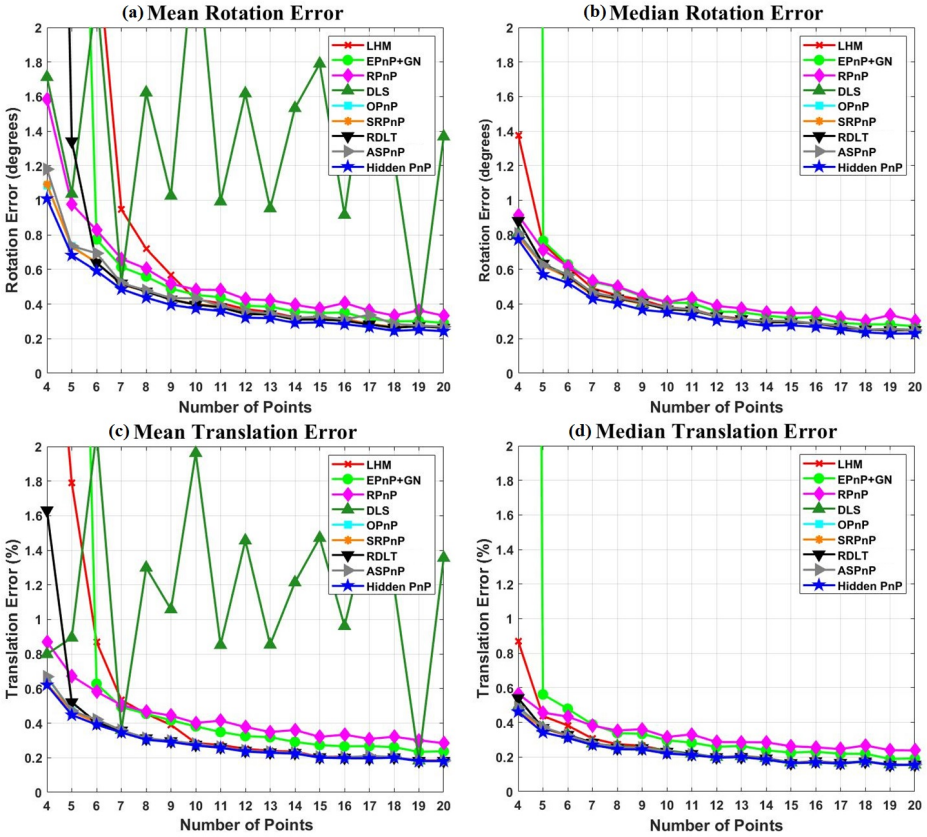


Figure 3. Error comparison in the Ordinary 3D case.

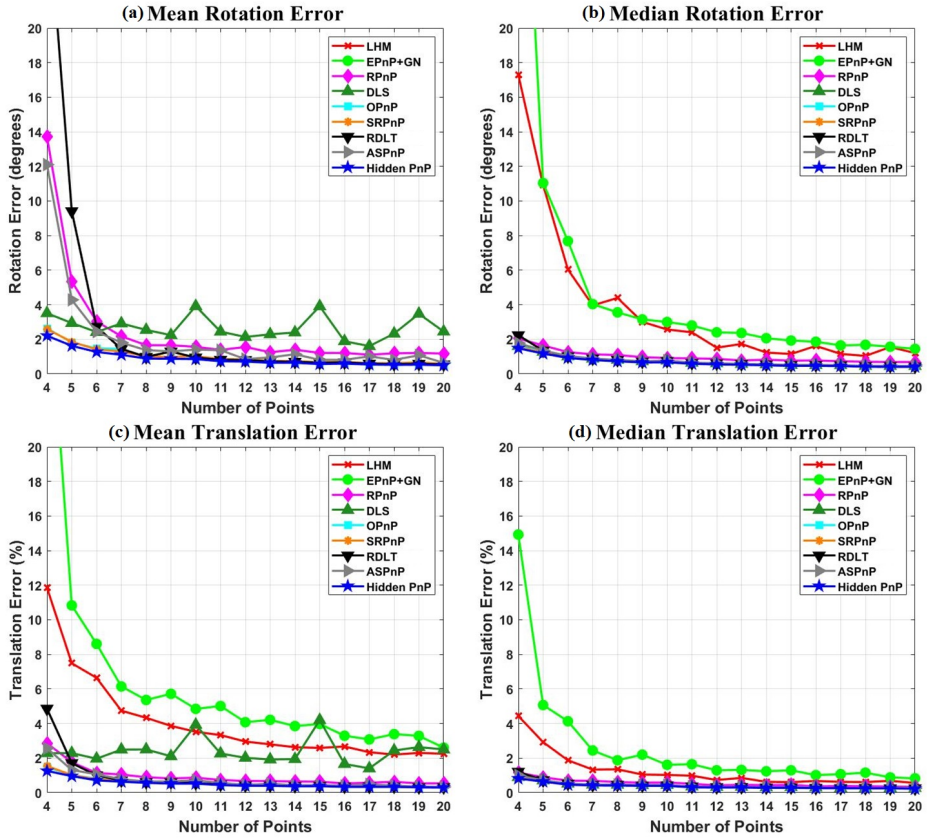


Figure 4. Error comparison in the Planar Case.

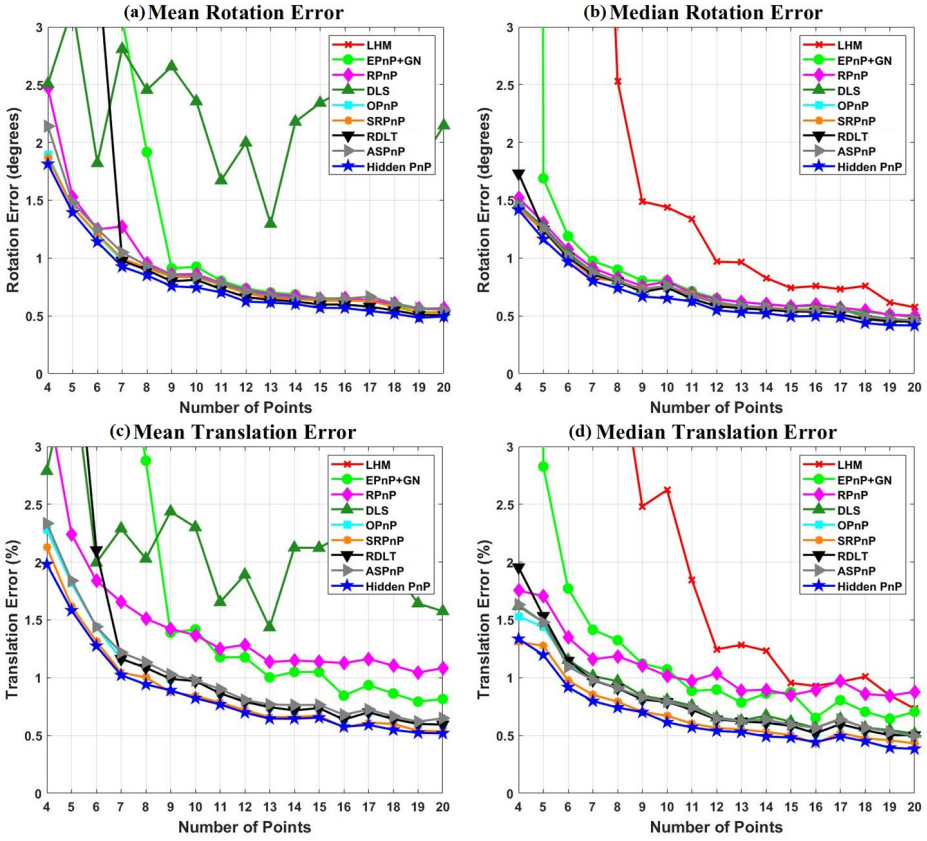


Figure 5. Error comparison in the Quasi-Singular case.

According to the above simulation tests, due to the iterative method used and the sensitivity of the initial value problem, the LHM algorithm exhibited higher error and rapid fluctuation of accuracy error in Ordinary 3D, Planar Case, and the Quasi-Singular case, and especially so in the latter, compared with other PnP algorithms in the presence of fewer spatial redundant points. The EPnP+GN algorithm, which was also a non-iterative algorithm, also produced a large error when there was noise and few spatial redundant points, which came down to the linear solution method adopted. Meanwhile, it was found that, if the Gauss–Newton method was not introduced to optimize the solutions of EPnP+GN, the performance of the EPnP algorithm alone was as unsatisfactory as that of the LHM algorithm. Interested readers could conduct experiments to ascertain the performance gap. The authors no longer tested the difference alone. In the above three cases, the DLS algorithm fluctuated greatly, which could be explained by the lack of proper treatment regarding the singularity of the CGR parameters, and, thus, the poor accuracy error and stability of its solution. The RPnP algorithm used linearization to transform non-linear equations into linear equations, and its optimization was still non-global, which led to its less favorable solution error and stability, compared with OPnP, SRPnP, RDLT, ASPnP, and Hidden PnP, etc. The OPnP, SRPnP, RDLT, and ASPnP, etc. exhibited favorable stability, and the accuracy error and stability of the four algorithms were similar.

Through a detailed comparison of the simulation data, it was easy to see that the best performance among the above nine algorithms was the Hidden PnP, which had the smallest error in solution accuracy. In addition, this algorithm provided the best stability in solving the three cases, without large fluctuations. The reason for this was that the Hidden PnP could quickly calculate the solution near the optimal solution. On this basis, the algorithm used this solution as the initial value of the Gauss–Newton method to achieve the precise location of the solution in one or two iterative steps. This process avoided the problem of unreasonable choice of initial values when using the Gaussian–Newton method directly,

and also avoided the operational instability and large computational burden caused by the need to construct a large matrix elimination template using the Gröbner basis technique. The Hidden PnP, therefore, had the high accuracy and universality sought by the PnP algorithm.

3.3. Comparative Simulation Test of Anti-Noise Performance of the PnP Method

The above simulation tests demonstrated the smaller accuracy error and better stability of the Hidden PnP, compared with other PnP algorithms, in different cases. Due to the interference of the environment and electronic equipment in practice, the collected information contained noise, so it was necessary to compare the anti-noise performance of the Hidden PnP algorithm and other PnP algorithms, which is a vital index of the comprehensive performance of a PnP algorithm. Accordingly, the following anti-noise performance simulation tests were conducted, in which the 3D spatial reference point n was set to be 10, and 10 different noise zero-mean Gaussian noises were set in Ordinary 3D, Planar Case, and Quasi-Singular case, respectively. The pixel value distribution started from δ = 0.5 pixels and increased to δ = 5 pixels by arithmetic sequence, with a tolerance of 0.5 pixels. In the above three cases, each PnP algorithm was independently tested 500 times for each fixed noise pixel value. Due to the large number of algorithms, compared in the simulation test, and the large amount of data, the following data curve could only be used as a macroscopic performance comparison display. Refer to Appendix B (Tables A4–A6) for the specific test data. Figures 6–8 shows the error comparison of zero-mean Gaussian image noise from 0.5 to 5 pixels in ordinary 3D, coplanar, and quasi-singularity cases, respectively. For all figure groups, the four sub-figures were: a. mean error of the rotation matrix; b. median error of rotation matrix; c. mean error of translation vector; d. median error of translation vector. The spatial reference point n = 10.

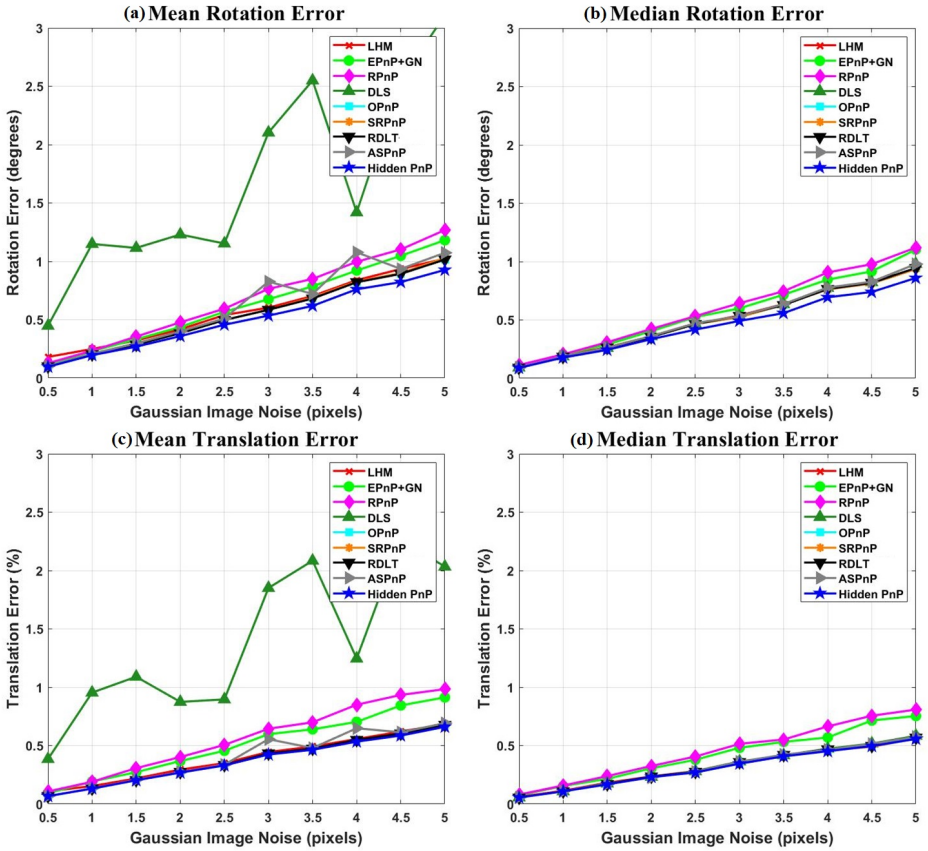


Figure 6. Error comparison in ordinary 3D case.

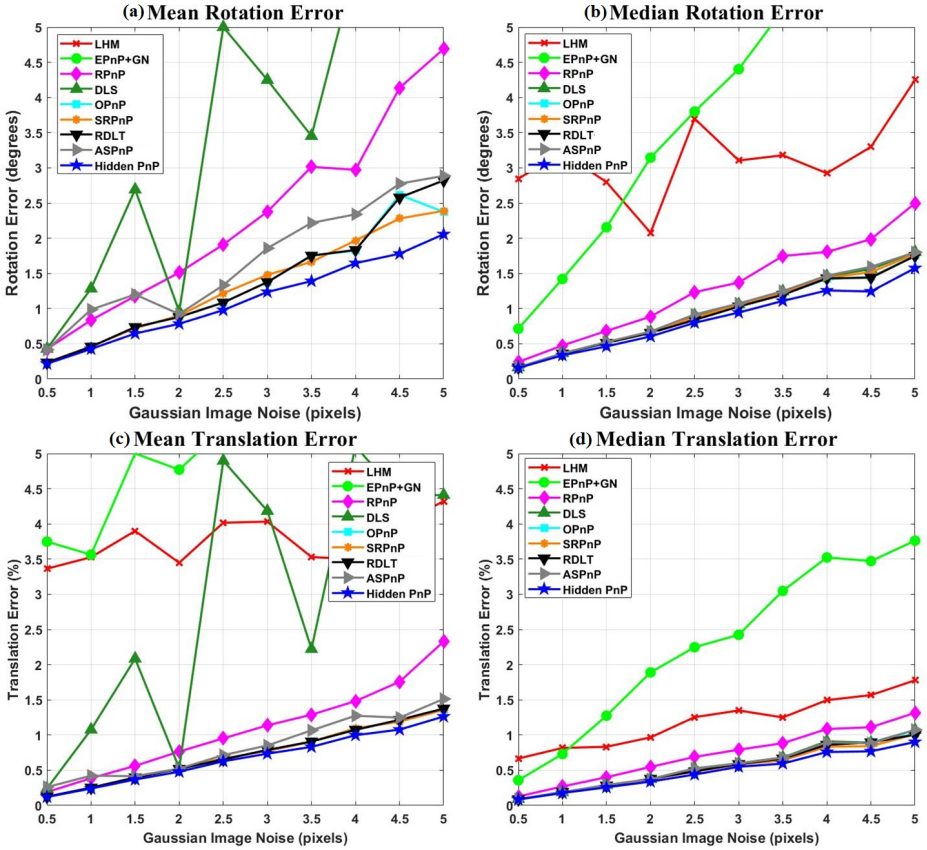


Figure 7. Error comparison in Planar case.

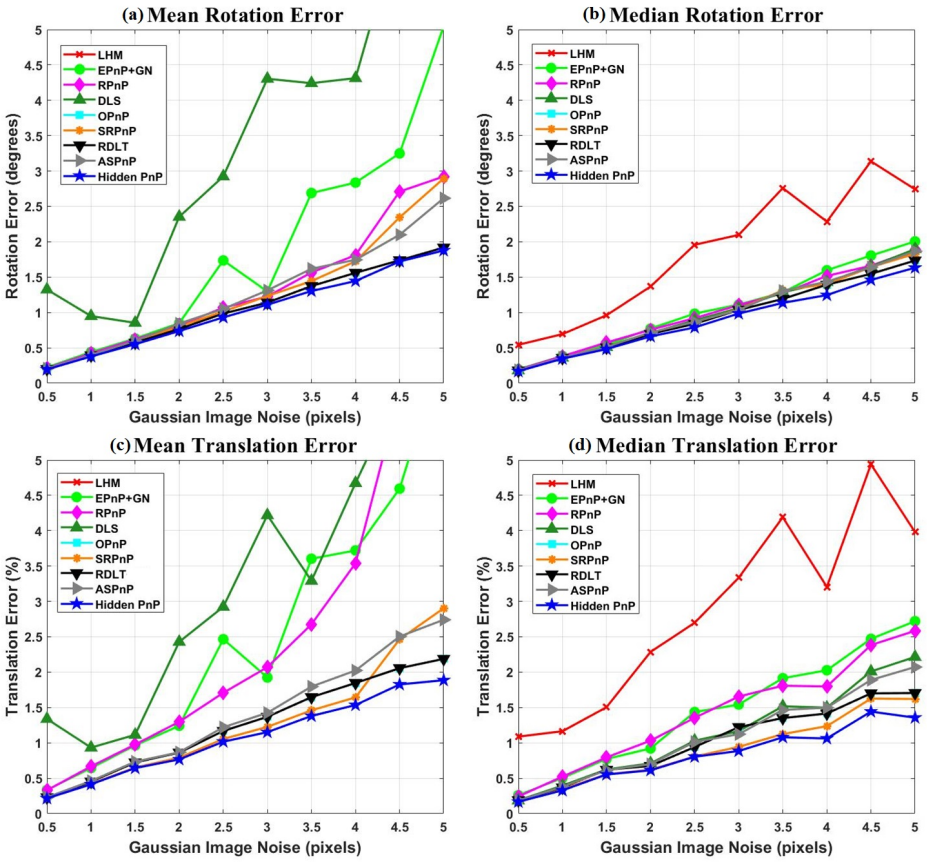


Figure 8. Error comparison in Quasi-singularity case.

The above simulation test curve and data revealed that the LHM algorithm exhibited the worst anti-noise ability in the above three cases, and the solution accuracy error of the DLS and EPnP+GN algorithms jumped with larger noise pixels. The anti-noise ability of the RPnP algorithm was better than that of the DLS and EPnP+GN algorithms, but worse than that of the OPnP, SRPnP, RDLT, and ASPnP algorithms. It was obvious that the Hidden PnP algorithm had the best anti-noise ability among the above nine algorithms.

3.4. Comparative Simulation Test of Calculation Efficiency of Different PnP Methods

The above simulation tests demonstrated that the solution accuracy error of the Hidden PnP algorithm was smaller than that of existing excellent PnP algorithms in Ordinary 3D, Planar Case, and Quasi-Singular cases. The Hidden PnP outperformed in anti-noise ability, which, however, was not enough. As mentioned in the introduction, an excellent PnP algorithm should perform not only with stability, accuracy, and universality, etc., but also with high computational efficiency. This section compares the computational efficiency of algorithms, including LHM, EPnP+GN, RPnP, DLS, OPnP, SRPnP, RDLT, and ASPnP, with Hidden PnP. In this simulation experiment, the pixel value δ of zero-mean Gaussian noise was fixed to be 2, and the number of spatial reference points n in the PnP algorithms involved increased from 4 to 500. Each PnP algorithm was independently tested 1000 times with different numbesr of spatial reference points, and the average running time calculated.

As shown in Figure 9 and Table 2, the simulation results here were the same as those mentioned in the introduction. The main factor affecting the efficiency of the LHM algorithm was the iterative solution, the duration of which accelerated with increase of spatial redundant point quantity. The DLS algorithm transformed the pose estimation problem into an unconstrained least square minimization problem through the construction of a cost function, which failed to avoid increase in its computational time with more spatial redundant points. The computation time of the ASPnP and OPnP algorithms did not increase much with the climbing spatial reference points. However, the Gröbner technique required the construction of a large elimination template matrix, which was the main burden that affected the operational efficiency of these algorithms. It is worth noting that the phased solution of the RPnP algorithm had the fastest speed among all algorithms when the spatial reference point was $n \leq 200$. Nevertheless, with the increase in the number of points, the computational time gradually increased. The SRPnP, as an improved version, also had the same problem. The RDLT algorithm adopted the traditional linear solution, considering the constraints between points, the efficiency of which was better than that of the ASPnP algorithm when the spatial reference point was $n \leq 100$. To sum up, the runtime simulation results from the various algorithms showed that, at 500 spatial reference points, the hidden variable-based PnP algorithm had a computational efficiency 1.5 to 7 times higher than those of the other algorithms that are considered excellent.

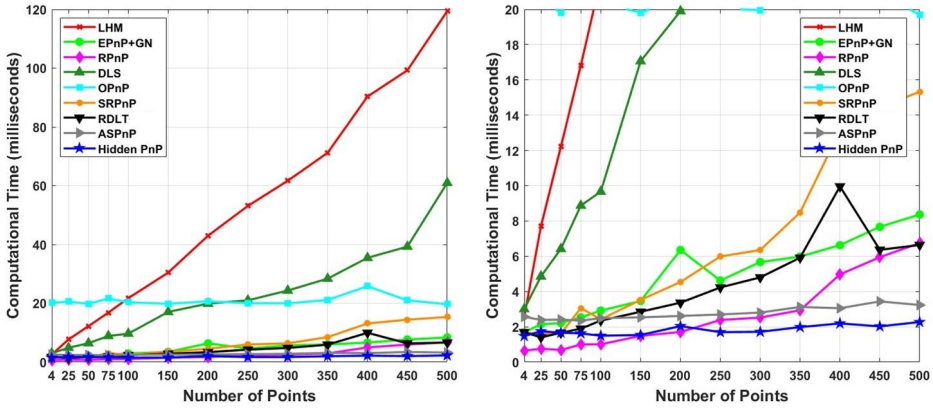


Figure 9. Comparison of the average running time of the PnP algorithm when the number of spatial reference points n ranged from 4 to 500.

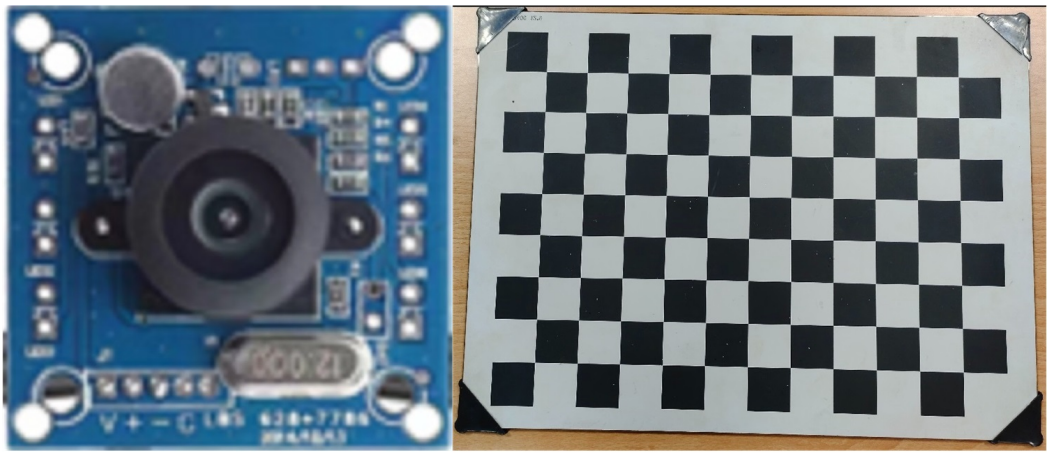
Table 2. Comparison of the average running time of the PnP algorithm when the number of spatial reference points n ranged from 4 to 500 (unit: ms).

Algorithm	4	25	50	75	100	150	200	250	300	350	400	450	500
LHM	2.5509	7.7007	12.2105	16.8193	21.7186	30.4288	42.9690	53.1203	61.6848	71.1986	90.3208	99.2493	119.3778
EPnP+GN	1.6838	2.1403	2.2151	2.5364	2.9156	3.4627	6.3518	4.6227	5.6691	5.9800	6.6296	7.6668	8.3636
RPnP	0.6458	0.7559	0.6870	0.9943	1.0058	1.4853	1.7070	2.3842	2.5221	2.9366	4.9716	5.9435	6.7481
DLS	3.0037	4.8548	6.4148	8.8721	9.6642	17.0611	19.8981	21.0115	24.3279	28.3810	35.4492	39.2732	60.9486
OPnP	20.1334	20.5736	19.7894	21.6439	20.3364	19.8135	20.6554	20.0438	19.9601	21.1331	25.7936	20.9996	19.6722
SRPnP	1.6868	1.5887	1.6163	3.0589	2.4218	3.5239	4.5388	5.9875	6.3517	8.4796	13.12230	14.4261	15.3281
RDLT	1.6937	1.39685	1.66310	1.8911	2.3511	2.8533	3.3670	4.2310	4.8015	5.9169	9.9572	6.3648	6.6381
ASPnP	2.5683	2.3886	2.4077	2.3488	2.4894	2.5342	2.6157	2.6969	2.8046	3.1248	3.0530	3.4392	3.2354
Hidden PnP	1.4739	1.7446	1.6663	1.6259	1.4931	1.5347	2.0399	1.6966	1.7153	1.9807	2.1906	2.0242	2.2700

3.5. Materials and Experimental Protocol for Physical Experiments

Based on synthetic data, the above section compared the performance of the Hidden PnP algorithm with the other eight PnP algorithms, and the Hidden PnP algorithm proved to have better accuracy, stability, anti-noise ability, and calculation speed. In this section, the authors applied the Hidden PnP algorithm and the other eight algorithms to physical objects, so as to reveal their actual performances.

The experimental materials employed included a high-resolution camera and high-precision checkerboard calibration board. In addition, the experiment adopted a Rmoncam G180 camera, featuring a distortionless angle of 120 (Figure 10a), a resolution of 1920 × 1080, and a pixel size of 3.01 µm. Zhang's camera calibration method [25] was also employed to calibrate the internal parameters of the Rmoncam G180 camera. As to the calibration board, a customized black and white checkerboard was utilized (Figure 10b), in which the size of each checkerboard was 30 mm × 30 mm, and the machining accuracy error was 0.01 mm. Table 3 depicts the internal parameters of the Rmoncam G180 camera's calibration.



(a) **(b)**

Figure 10. (**a**) Rmoncam G180 camera. (**b**) High-precision checkerboard calibration board.

Table 3. Internal parameters of Rmoncam G180 camera.

Parameter	Specific Data
Focal length	$f_x = 735.725581$, $f_y = 735.80416$
Principal point	$(CC_x, CC_y) = (598.40811, 381.51785)$
Radial distortion coefficient	$k_1 = 0.06689, k_2 = -0.08019$
Tangential distortion coefficient	$p_1 = -0.00088, p_2 = -0.00196$

Figure 11 illustrates the specific experimental process. The cyan circle '○' in Figure 12 refers to the position of all corner points within the extraction range of corner points on the calibration plate. Figure 13 reveals the 10 images of the calibration plate under different attitudes, where blue '○' stands for the sub-pixel corner point randomly extracted by various PnP algorithms, while the red '+' represents the reprojection corner point calculated through the PnP algorithms. The loss function to evaluate the performance of the PnP algorithms was constructed as:

$$\bar{\delta} = \frac{1}{100}\sum_{i=1}^{100}(u_i - \bar{u}_i)^2 + (v_i - \bar{v}_i)^2 \tag{27}$$

The following can be observed from the reprojection error test data in Table 4: the DLS algorithm had the largest error, followed by the LHM algorithm. Except for the Hidden PnP algorithm, the error accuracy of the remaining six algorithms did not differ greatly. In

general, the mean and median errors of reprojection coordinate points of the Hidden PnP algorithm were basically equal to, or even slightly superior to, the optimized algorithms, including OPnP, SRPnP, and ASPnP. Nonetheless, the solution method of the Hidden PnP performed with the best efficiency. The experiment on physical objects in this section also revealed that the Hidden PnP could match the best existing PnP method in solution accuracy, which supported the cost performance of the algorithm in solving PnP problems.

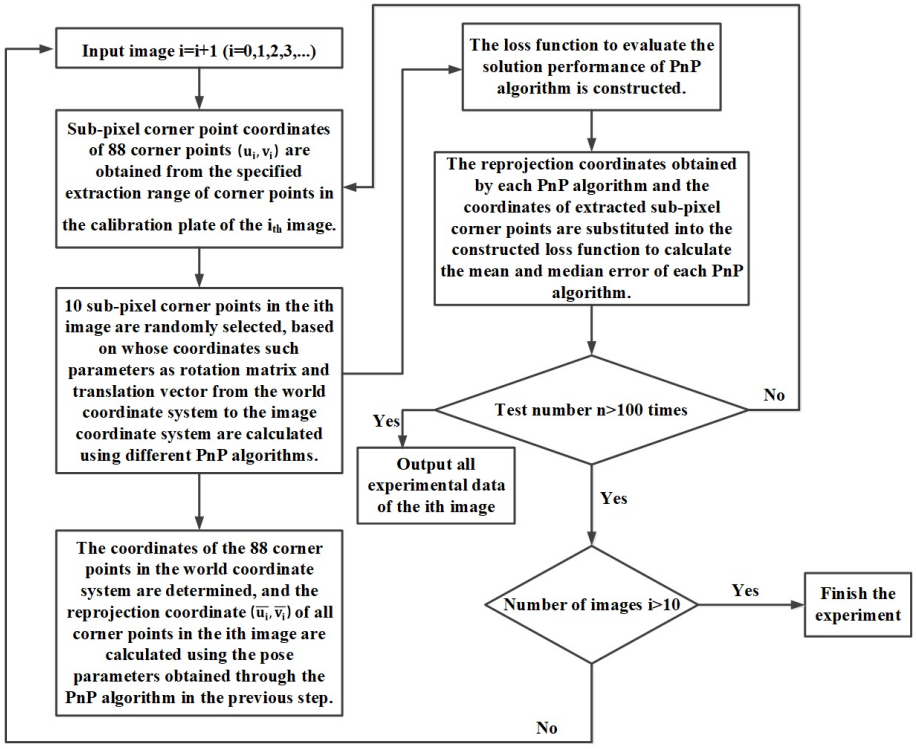


Figure 11. Flow chart of physical objects reprojection experiment.

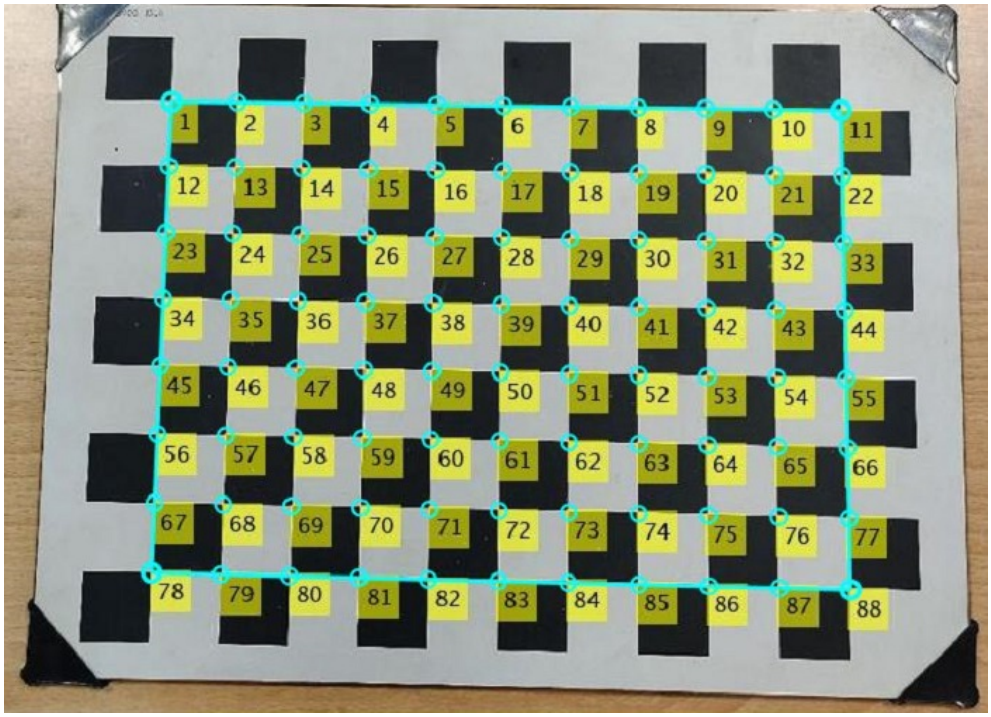


Figure 12. Extraction range of corner points on calibration board.

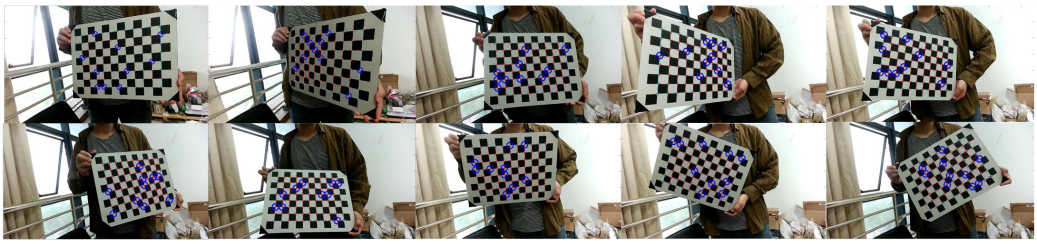


Figure 13. Ten images of the calibration plate at different distances and attitudes and the reprojection effects.

Table 4. Mean error ($mean_{err}$) and median errors (med_{err}) of reprojection coordinate points in 10 images of the calibration plate at different distances and attitudes (unit: pixel).

Algorithm		1	2	3	4	5	6	7	8	9	10
LHM	$mean_{err}$	45	136.1	25.2	134.5	53.3	29.1	51.1	34.1	30.3	22.3
	med_{err}	42.4	123.6	22.4	137.1	51.9	20.2	51.0	34.6	29.7	20.9
EPnP+GN	$mean_{err}$	14.8	10.6	10.2	19.7	8.7	19.4	6.1	9.5	18.6	15.1
	med_{err}	15.1	10.5	10.4	19.7	8.9	19.5	6.1	9.6	18.8	15.1
RPnP	$mean_{err}$	14.7	10.8	12.9	20.2	10.4	18.8	6.7	10.1	18.5	15.1
	med_{err}	14.7	10.8	10.9	20.2	9.1	18.8	6.2	10.1	18.6	15.1
DLS	$mean_{err}$	2541.7	2542.5	2606.1	1928.4	2156.4	2907.1	2470.4	2118.2	1889.5	2583.5
	med_{err}	2534.5	2551.7	2602.6	1936.5	2160.8	2913.1	2468.3	2118.7	1901.1	2584.3
OPnP	$mean_{err}$	14.7	10.5	10.4	20.1	8.7	18.9	6.6	9.9	32.5	14.9
	med_{err}	14.9	10.5	10.4	19.8	8.8	18.9	6.1	9.8	22.8	15.1
SRPnP	$mean_{err}$	14.8	10.5	10.2	19.8	8.6	18.9	5.8	9.6	18.5	14.8
	med_{err}	14.9	10.4	10.5	19.9	8.8	18.9	5.8	9.6	18.8	15.0
RDLT	$mean_{err}$	14.8	10.6	10.1	19.8	8.7	19.6	6.1	9.6	18.6	15.1
	med_{err}	14.9	10.6	10.4	19.8	9.1	19.6	6.0	9.6	18.8	15.1
ASPnP	$mean_{err}$	15.2	10.6	10.2	19.8	8.6	18.9	6.9	9.6	21.2	14.8
	med_{err}	14.9	10.4	10.4	19.9	8.8	18.9	5.8	9.6	18.9	15.0
Hidden PnP	$mean_{err}$	14.5	10.4	10.1	19.8	8.5	18.7	5.8	9.6	18.5	14.8
	med_{err}	14.8	10.4	10.4	19.7	8.8	18.9	5.8	9.6	18.7	15.0

4. Conclusions and Prospects

The PnP solution, based on a projective imaging model of 3D points, fails to cover accuracy, robustness, and efficiency simultaneously because of the diverse spatial distribution and quantity of 3D reference points, which prompted us to propose a new method for solving the PnP problem, the Hidden PnP. The Hidden PnP employs the CGR parameter to parameterize the rotation matrix. Differing from the best matrix synthesis technique, the Gröbner technique, the method reduces computational burdens, as it does not require construction of a large matrix elimination template in the polynomial solution phase. The CGR parameter matrix was solved by the hidden variable method, and the Gauss–Newton method was adopted for rapid and accurate location of a solution. The comparison test demonstrated that the Hidden PnP outperformed eight other algorithms in solution accuracy, stability, anti-noise ability, and calculation speed, in both synthetic data and real experiments, which supported its application to high precision position measurement in a common environment. Future work will, therefore, involve applying this algorithm to vision-based navigation for intelligent vehicles and UAVs with real-time positional measurement techniques.

Author Contributions: Conceptualization, R.Q., G.X., P.W. and Y.C.; Methodology, R.Q. and P.W.; Resources, G.X., Y.C. and W.D.; Software, R.Q. and P.W.; formal analysis, R.Q., G.X. and P.W.; Writing—original draft, R.Q.; Writing—review and editing, G.X., P.W. and R.Q. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported, in part, by the National Key Research and Development Plan, under Grant 2018YFB2003803, and, in part, by the National Natural Science Foundation of China, under Grants 62073161, 61905112 and U1804157.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

P3P	Perspective-3-Point
P4P	Perspective-4-Point
P5P	Perspective-5-Point
PnP	Perspective-n-Point
GN	Gauss-Newton iterative method
CGR	Cayley-Gibbs-Rodriguez
SLAM	Simultaneous Localization and Mapping
DLT	Traditional direct liner transformation
ASPnP	Accurate and Scalable Solution to the Perspective-n-Point problem
RPnP	A Robust O(n)Solution to the Perspective-n-Point problem
OptDLS	Optimal DLS Method
SRPnP	A simple, robust and fast method for the Perspective-n-Point problem
Hidden PnP	hidden variable-based PnP algorithm

Appendix A

Table A1. Error comparison in the Ordinary 3D case.

Algorithm		4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
LHM	mean_r	13.0754	4.7251	2.5100	0.9473	0.7202	0.5688	0.4262	0.4057	0.3716	0.3542	0.3234	0.3122	0.3132	0.2850	0.2636	0.2737	0.2643
	med_r	1.3771	0.7489	0.6150	0.4903	0.4502	0.4237	0.3813	0.3706	0.3314	0.3156	0.2954	0.2972	0.2882	0.2654	0.2538	0.2517	0.2489
	mean_t	3.4679	1.7890	0.8695	0.5337	0.4541	0.3898	0.2853	0.27371	0.2519	0.2394	0.2355	0.2046	0.2067	0.2002	0.2034	0.1843	0.1840
	med_t	0.8707	0.4374	0.3825	0.3066	0.2743	0.2652	0.2358	0.2189	0.2001	0.2064	0.1950	0.1676	0.1767	0.1658	0.1766	0.1564	0.1566
EPnP+GN	mean_r	76.8848	5.5589	0.7745	0.6151	0.5600	0.4898	0.4536	0.4390	0.3906	0.3853	0.3567	0.3488	0.3514	0.3139	0.3004	0.3021	0.2911
	med_r	62.5052	0.7661	0.6297	0.5285	0.4979	0.4444	0.4094	0.4063	0.3570	0.3534	0.3334	0.3192	0.3262	0.2916	0.2832	0.2826	0.2719
	mean_t	84.1045	6.1323	0.6279	0.4937	0.4539	0.4167	0.3812	0.3489	0.3252	0.3186	0.2913	0.2734	0.2659	0.2670	0.2615	0.2336	0.2373
	med_t	25.3651	0.5623	0.4791	0.3882	0.3386	0.3342	0.2973	0.2833	0.2589	0.2645	0.2390	0.2256	0.2309	0.2189	0.2193	0.1891	0.1919
RPnP	mean_r	1.5839	0.9786	0.8274	0.6625	0.6047	0.5180	0.4834	0.4818	0.4288	0.4227	0.3963	0.3732	0.4069	0.3623	0.3341	0.3638	0.3328
	med_r	0.9132	0.7124	0.6170	0.5334	0.5019	0.4501	0.4116	0.4353	0.3894	0.3754	0.3535	0.3477	0.3481	0.3203	0.3041	0.3362	0.3049
	mean_t	0.8706	0.6730	0.5824	0.5043	0.4675	0.4435	0.4016	0.4142	0.3775	0.3491	0.3600	0.3207	0.3336	0.3090	0.3214	0.3014	0.2838
	med_t	0.5658	0.4572	0.4337	0.3815	0.3532	0.3611	0.3164	0.3311	0.2869	0.2862	0.2854	0.2624	0.2562	0.2461	0.2673	0.2395	0.2384
DLS	mean_r	1.7126	1.0378	2.2025	0.5228	1.6235	1.0265	2.5497	0.9934	1.6183	0.9533	1.5333	1.7904	0.9160	1.9107	1.2365	0.2764	1.3690
	med_r	0.8089	0.6307	0.5686	0.4656	0.4399	0.4049	0.3797	0.3693	0.3294	0.3125	0.3038	0.3002	0.2883	0.2771	0.2525	0.2564	0.2514
	mean_t	0.7990	0.8944	2.0803	0.3606	1.2989	1.0579	1.9619	0.8520	1.4562	0.8537	1.2139	1.4712	0.9613	1.5552	1.1896	0.1840	1.3562
	med_t	0.4904	0.3676	0.3281	0.2778	0.2626	0.2559	0.2386	0.2192	0.2005	0.2047	0.1934	0.1686	0.1746	0.1671	0.1767	0.1578	0.1561
OPnP	mean_r	1.0844	0.7366	0.6380	0.5188	0.4724	0.4259	0.3967	0.3817	0.3430	0.3402	0.3106	0.3106	0.3038	0.2821	0.2635	0.2722	0.2628
	med_r	0.8024	0.6321	0.5611	0.4614	0.4329	0.4075	0.3691	0.3631	0.3291	0.3130	0.2956	0.2954	0.2889	0.2670	0.2532	0.2485	0.2502
	mean_t	0.6626	0.4722	0.4035	0.3571	0.3119	0.2955	0.2754	0.2614	0.2404	0.2312	0.2277	0.2030	0.2008	0.1964	0.2019	0.1814	0.1818
	med_t	0.4823	0.3676	0.3281	0.2832	0.2523	0.2513	0.2309	0.2177	0.1994	0.2006	0.1914	0.1653	0.1758	0.1661	0.1740	0.1546	0.1566
SRPnP	mean_r	1.0925	0.7302	0.6423	0.5126	0.4748	0.4264	0.4012	0.3848	0.3460	0.3400	0.3178	0.3156	0.3082	0.2865	0.2674	0.2720	0.2628
	med_r	0.7914	0.6225	0.5551	0.4503	0.4308	0.4014	0.3762	0.3633	0.3278	0.3077	0.2996	0.3005	0.2884	0.2715	0.2510	0.2536	0.2472
	mean_t	0.6262	0.4662	0.4039	0.3560	0.3123	0.2954	0.2753	0.2597	0.2402	0.2312	0.2284	0.2028	0.2004	0.1974	0.2015	0.1814	0.1818
	med_t	0.4636	0.3688	0.3203	0.2795	0.2532	0.2500	0.2313	0.2132	0.1966	0.2014	0.1893	0.1674	0.1750	0.1631	0.1750	0.1543	0.1570
RDLT	mean_r	8.1381	1.3415	0.6382	0.5188	0.4724	0.4259	0.3967	0.3817	0.3430	0.3402	0.3106	0.3106	0.3038	0.2821	0.2635	0.2722	0.2628
	med_r	0.8814	0.6333	0.5611	0.4614	0.4329	0.4075	0.3691	0.3631	0.3291	0.3130	0.2956	0.2954	0.2889	0.2670	0.2532	0.2485	0.2502
	mean_t	1.6306	0.5207	0.4037	0.3571	0.3119	0.2955	0.2754	0.2614	0.2404	0.2312	0.2277	0.2030	0.2008	0.1964	0.2019	0.1814	0.1818
	med_t	0.5407	0.3676	0.3281	0.2832	0.2523	0.2513	0.2309	0.2177	0.1994	0.2006	0.1914	0.1653	0.1758	0.1661	0.1740	0.1546	0.1566
ASPnP	mean_r	1.1801	0.7359	0.6940	0.5215	0.4792	0.4353	0.4332	0.3900	0.3592	0.3459	0.3203	0.3275	0.3121	0.3374	0.2844	0.2762	0.2698
	med_r	0.8074	0.6254	0.5683	0.4651	0.4380	0.4105	0.3790	0.3686	0.3291	0.3114	0.3033	0.3026	0.2886	0.2740	0.2532	0.2549	0.2504
	mean_t	0.6688	0.4709	0.4209	0.3591	0.3130	0.2977	0.2808	0.2620	0.2417	0.2335	0.2285	0.2044	0.2023	0.2106	0.2045	0.1817	0.1822
	med_t	0.4869	0.3739	0.3250	0.2798	0.2567	0.2540	0.2324	0.2142	0.2012	0.2029	0.1896	0.1666	0.1757	0.1648	0.1751	0.1560	0.1570
Hidden PnP	mean_r	1.0096	0.6825	0.5921	0.4868	0.4374	0.3959	0.3746	0.3603	0.3213	0.3191	0.2911	0.2940	0.2840	0.2675	0.2452	0.2520	0.2432
	med_r	0.7724	0.5711	0.5231	0.4308	0.4047	0.3669	0.3530	0.3350	0.3053	0.2922	0.2747	0.2774	0.2682	0.2532	0.2366	0.2293	0.2305
	mean_t	0.6206	0.4472	0.3912	0.3457	0.3041	0.2911	0.2708	0.2575	0.2358	0.2283	0.2241	0.2006	0.1971	0.1963	0.2000	0.1805	0.1803
	med_t	0.4614	0.3423	0.3102	0.2698	0.2452	0.2436	0.2198	0.2116	0.1963	0.1999	0.1850	0.1641	0.1685	0.1617	0.1743	0.1537	0.1546

Note: mean_r and med_r represent mean error and median error of rotation matrix, respectively. mean_t and med_t represent the mean error and median error of the translation vector, respectively.

Table A2. Error comparison in the Planar Case.

Algorithm		4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
LHM	mean_r	47.6084	41.1666	38.5179	29.1996	35.2541	35.3723	30.1953	32.5595	27.3411	32.6208	27.3603	26.6317	24.9990	30.0272	24.1386	29.0261	27.5716
	med_r	17.3057	10.9226	6.0421	3.9639	4.3974	3.0146	2.5718	2.4091	1.5180	1.7360	1.2385	1.1547	1.6215	1.1610	1.0436	1.5580	1.2107
	mean_t	11.8580	7.4881	6.6424	4.7441	4.3324	3.8645	3.5253	3.3329	2.9597	2.7985	2.6314	2.5928	2.6732	2.3425	2.2064	2.3023	2.2538
	med_t	4.4363	2.9252	1.8759	1.3380	1.3540	1.0515	1.0270	0.9867	0.7359	0.8561	0.6310	0.6124	0.6781	0.6283	0.6173	0.6853	0.5652
EPnP+GN	mean_r	72.5487	52.7761	55.4373	43.6290	43.7338	47.6463	49.1237	54.2072	46.3542	53.3214	48.8542	49.0449	47.1507	44.5502	50.4445	45.7662	39.7199
	med_r	38.6826	11.0440	7.6876	4.0413	3.5632	3.1551	3.0051	2.8045	2.4111	2.3700	2.0679	1.9320	1.8653	1.6525	1.6850	1.5760	1.4522
	mean_t	31.1458	10.8352	8.6082	6.1455	5.3650	5.7193	4.8501	5.0141	4.0794	4.2141	3.8548	3.9762	3.2950	3.0837	3.3897	3.2871	2.6048
	med_t	14.9291	5.0701	4.1344	2.4429	1.8784	2.2016	1.6147	1.6494	1.2980	1.3211	1.2384	1.2976	1.0184	1.0724	1.1601	0.8845	0.8339
RPnP	mean_r	13.7271	5.3508	3.0398	2.1776	1.6670	1.6599	1.5712	1.3900	1.5611	1.2574	1.3994	1.2204	1.2133	1.1129	1.1933	1.2165	1.1784
	med_r	2.0487	1.6477	1.2644	1.1359	1.0954	0.9717	0.9196	0.8995	0.8759	0.7788	0.8219	0.7708	0.7830	0.7336	0.7049	0.6939	0.6917
	mean_t	2.8646	1.7623	1.1446	1.0526	0.8943	0.8262	0.8689	0.7395	0.6782	0.6734	0.6490	0.6375	0.5383	0.5801	0.6324	0.5377	0.5457
	med_t	1.1586	0.8932	0.6991	0.6915	0.6136	0.5845	0.5937	0.5173	0.4245	0.4576	0.4378	0.4487	0.3731	0.3838	0.3813	0.3563	0.3402
DLS	mean_r	3.5122	2.9464	2.4260	2.9221	2.5462	2.2454	3.9204	2.4494	2.1385	2.2930	2.3993	3.9147	1.8941	1.6100	2.3338	3.4797	2.4525
	med_r	1.6132	1.2857	1.0201	0.8861	0.8174	0.7358	0.7703	0.6504	0.6189	0.5802	0.5604	0.5232	0.5141	0.5095	0.4659	0.4648	0.4623
	mean_t	2.2829	2.2897	1.9681	2.4944	2.5019	2.1091	3.9274	2.2677	2.0180	1.9241	1.9448	4.1985	1.6631	1.4068	2.4417	2.6388	2.5014
	med_t	0.9078	0.7168	0.5260	0.4702	0.4652	0.4426	0.4466	0.3633	0.3342	0.3431	0.3136	0.3265	0.2985	0.27037	0.2760	0.2716	0.2727
OPnP	mean_r	2.6390	1.8239	1.4753	1.4502	0.9841	0.9869	0.9514	0.8194	0.8034	0.7222	0.7024	0.6511	0.6700	0.5955	0.5721	0.5849	0.5535
	med_r	1.5836	1.2897	1.0140	0.8749	0.8175	0.7151	0.7563	0.6384	0.6184	0.5748	0.5521	0.5167	0.5198	0.4965	0.4534	0.4605	0.4277
	mean_t	1.4720	1.0224	0.8103	0.7082	0.6156	0.5724	0.5581	0.4732	0.4245	0.4209	0.3870	0.4075	0.3567	0.3551	0.3591	0.3239	0.3174
	med_t	0.8802	0.6960	0.5277	0.4516	0.4592	0.4256	0.4336	0.3575	0.3224	0.3310	0.3013	0.3122	0.2780	0.2678	0.2755	0.2548	0.2520
SRPnP	mean_r	2.5963	1.8562	1.4161	1.3039	1.0142	0.9866	1.0194	0.8687	0.8384	0.7547	0.7466	0.6795	0.7001	0.6216	0.6201	0.6693	0.5857
	med_r	1.5498	1.2400	1.0141	0.8707	0.8091	0.7389	0.7573	0.6399	0.6160	0.5681	0.5638	0.5175	0.5116	0.5056	0.4666	0.4644	0.4632
	mean_t	1.5448	1.0200	0.7810	0.7041	0.6038	0.5649	0.5523	0.4861	0.4243	0.4291	0.3861	0.4115	0.3560	0.3546	0.3715	0.3407	0.3244
	med_t	0.8369	0.6811	0.5081	0.4637	0.4508	0.4246	0.4108	0.3553	0.3303	0.3205	0.2974	0.3201	0.2780	0.2661	0.2684	0.2725	0.2561
RDLT	mean_r	27.8986	9.4104	2.7283	1.4517	0.9881	1.2952	0.9504	0.8201	0.8038	0.7227	0.7024	0.6497	0.6697	0.5952	0.5729	0.5849	0.5533
	med_r	2.2386	1.3494	1.0219	0.8923	0.8128	0.7142	0.7561	0.6350	0.6150	0.5753	0.5503	0.5158	0.5194	0.4974	0.4538	0.4614	0.4273
	mean_t	4.8601	1.7110	1.0066	0.7103	0.6165	0.6102	0.5581	0.4737	0.4247	0.4210	0.3870	0.4076	0.3565	0.3551	0.3592	0.3239	0.3174
	med_t	1.2106	0.7245	0.5313	0.4519	0.4603	0.4208	0.4298	0.3598	0.3203	0.3308	0.3023	0.3114	0.2785	0.2670	0.2758	0.2578	0.2511
ASPnP	mean_r	12.1086	4.2874	2.4447	1.8109	1.3761	1.3070	1.4086	1.370	0.8622	0.9546	1.1412	0.8169	0.8210	1.0147	0.8163	1.0789	0.6431
	med_r	1.7827	1.3545	1.0436	0.8918	0.8152	0.7375	0.7723	0.6578	0.6234	0.5817	0.5566	0.5267	0.5232	0.5115	0.4662	0.4700	0.4596
	mean_t	2.5704	1.3143	1.0646	0.7800	0.6510	0.6491	0.7230	0.5723	0.4506	0.5207	0.4291	0.4430	0.4074	0.4873	0.4124	0.3526	0.3315
	med_t	1.0066	0.7218	0.5315	0.4708	0.4523	0.4363	0.4371	0.3590	0.3337	0.3415	0.3095	0.3237	0.2970	0.2714	0.2753	0.2722	0.2650
Hidden PnP	mean_r	2.2123	1.6286	1.2707	1.1090	0.8888	0.8777	0.8582	0.7365	0.7161	0.6461	0.6473	0.5736	0.5986	0.5464	0.5263	0.5372	0.5011
	med_r	1.4838	1.1802	0.9272	0.8069	0.7313	0.6661	0.6751	0.5891	0.5510	0.5312	0.5015	0.4680	0.4720	.4566	0.4164	0.4103	0.4130
	mean_t	1.2603	0.9613	0.7147	0.6370	0.5701	0.5297	0.5259	0.4406	0.3996	0.4007	0.3696	0.3785	0.3381	0.3392	0.3358	0.3143	0.2988
	med_t	0.8210	0.6414	0.4654	0.4213	0.4139	0.3986	0.3955	0.3136	0.3009	0.3048	0.2779	0.2851	0.2651	0.2613	0.2545	0.2551	0.2284

Note: mean_r and med_r represent mean error and median error of rotation matrix, respectively. mean_t and med_t represent the mean error and median error of the translation vector, respectively.

Table A3. Error comparison in the Quasi-Singular case.

Algorithm		4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
LHM	mean_r	88.5363	83.4160	76.3662	64.9129	59.4713	57.7804	53.2612	52.2715	45.9908	46.9352	44.3802	34.8269	34.8904	32.9860	41.7693	29.6207	27.1142
	med_r	114.8179	113.0583	107.2753	7.5616	2.5322	1.4896	1.4404	1.3385	0.9689	0.9629	0.8284	0.7423	0.7579	0.7313	0.7587	0.6166	0.5740
	mean_t	22.4275	23.5323	23.1096	21.5192	21.9526	22.1201	22.4902	23.9994	20.6341	22.2973	21.0504	17.1425	16.9422	17.0355	21.4164	15.1585	14.5199
	med_t	13.3753	14.3777	12.4502	6.3186	3.8556	2.4819	2.6239	1.8476	1.2448	1.2844	1.2337	0.9537	0.9284	0.9641	1.0083	0.8474	0.7319
EPnP+GN	mean_r	79.9803	22.0146	5.8036	3.0681	1.9169	0.9111	0.9253	0.8037	0.7383	0.7027	0.6857	0.6468	0.6426	0.6275	0.5920	0.5585	0.5544
	med_r	109.5947	1.6905	1.1897	0.9748	0.8991	0.8050	0.8055	0.7131	0.6509	0.6197	0.5987	0.5823	0.5839	0.5351	0.5356	0.5121	0.4868
	mean_t	85.9515	25.5783	7.5526	4.2305	2.8772	1.3890	1.4177	1.1768	1.1766	1.0031	1.0493	1.0488	0.8451	0.9353	0.8632	0.7929	0.8156
	med_t	35.4233	2.8269	1.7718	1.4147	1.3241	1.1203	1.0722	0.8859	0.8969	0.7860	0.8629	0.8747	0.6522	0.8047	0.7056	0.6456	0.7045
RPnP	mean_r	2.4754	1.5292	1.2479	1.2740	0.9538	0.8581	0.8610	0.7846	0.7237	0.6890	0.6723	0.6504	0.6531	0.6340	0.6077	0.5630	0.5647
	med_r	1.5216	1.3075	1.0744	0.9148	0.8316	0.7598	0.7967	0.6982	0.6437	0.6188	0.6017	0.5832	0.5967	0.5710	0.5483	0.5105	0.5024
	mean_t	3.4242	2.2408	1.8396	1.6556	1.5115	1.4216	1.3692	1.2521	1.2847	1.1356	1.1485	1.1397	1.1262	1.1622	1.1048	1.0447	1.0828
	med_t	1.7536	1.7059	1.3482	1.1599	1.1854	1.1057	1.0153	0.9726	1.0366	0.8878	0.8960	0.8505	0.8917	0.9717	0.8575	0.8454	0.8771
DLS	mean_r	2.5075	3.1715	1.8212	2.8081	2.4542	2.6568	2.3554	1.6720	1.9985	1.2962	2.1792	2.3421	2.4598	2.8239	2.2215	1.8087	2.1475
	med_r	1.4606	1.2713	1.0437	0.8794	0.8034	0.7300	0.7588	0.6885	0.6164	0.5861	0.5736	0.5553	0.5542	0.5652	0.5016	0.4737	0.4619
	mean_t	2.7870	3.7659	1.9955	2.2902	2.0288	2.4368	2.3009	1.6533	1.8906	1.4351	2.1246	2.1229	2.2386	2.2340	1.9413	1.6419	1.5749
	med_t	1.6199	1.4788	1.1536	1.0144	0.9694	0.8427	0.8076	0.7584	0.6547	0.6283	0.6716	0.6193	0.5669	0.6438	0.5678	0.5472	0.5149
OPnP	mean_r	1.8925	1.4384	1.1989	0.9776	0.8968	0.7994	0.8120	0.7318	0.6629	0.6380	0.6243	0.5995	0.5982	0.5796	0.5481	0.5083	0.5075
	med_r	1.4476	1.2364	1.0059	0.8574	0.7972	0.7071	0.7454	0.6477	0.5837	0.5659	0.5527	0.5386	0.5349	0.5124	0.4729	0.4529	0.4495
	mean_t	2.2792	1.8178	1.4381	1.1628	1.0862	0.9894	0.9710	0.8649	0.7913	0.7451	0.7172	0.7368	0.6374	0.7012	0.6420	0.5993	0.5948
	med_t	1.5295	1.4414	1.1241	0.9778	0.9097	0.8119	0.7907	0.7238	0.6383	0.6219	0.6100	0.5837	0.5189	0.5980	0.5455	0.5032	0.5020
SRPnP	mean_r	1.8680	1.4407	1.2095	0.9957	0.9167	0.8362	0.8345	0.7667	0.6981	0.6511	0.6477	0.6310	0.6316	0.6297	0.5794	0.5305	0.5305
	med_r	1.4332	1.2343	1.0021	0.8286	0.7998	0.7363	0.7563	0.6882	0.6060	0.5668	0.5754	0.5547	0.5512	0.5604	0.4914	0.4698	0.4664
	mean_t	2.1290	1.6188	1.3144	1.0451	1.0025	0.8758	0.8434	0.7850	0.7204	0.6581	0.6624	0.6706	0.5687	0.6117	0.5999	0.5411	0.5357
	med_t	1.3096	1.2755	0.9757	0.8524	0.7920	0.7056	0.6701	0.6034	0.5649	0.5526	0.5305	0.5035	0.4311	0.5232	0.4754	0.4588	0.4310
RDLT	mean_r	20.2851	6.0432	3.6167	0.9776	0.8968	0.7994	0.8120	0.7318	0.6629	0.6380	0.6243	0.5995	0.5982	0.5796	0.5481	0.5083	0.5075
	med_r	1.7325	1.2456	1.0085	0.8574	0.7972	0.7071	0.7454	0.6477	0.5837	0.5659	0.5527	0.5386	0.5349	0.5124	0.4729	0.4529	0.4495
	mean_t	9.4143	4.4118	2.1033	1.1628	1.0862	0.9894	0.9710	0.8649	0.7913	0.7451	0.7172	0.7368	0.6374	0.7012	0.6420	0.5993	0.5948
	med_t	1.9552	1.5340	1.1465	0.9777	0.9097	0.8119	0.7907	0.7238	0.6383	0.6219	0.6100	0.5837	0.5189	0.5980	0.5455	0.5032	0.5020
ASPnP	mean_r	2.1409	1.4813	1.2524	1.0484	0.9350	0.8538	0.8421	0.7811	0.7125	0.6732	0.6523	0.6527	0.6495	0.6667	0.6085	0.5666	0.5510
	med_r	1.4528	1.2572	1.0228	0.8752	0.8030	0.7273	0.7610	0.6730	0.6129	0.5828	0.5709	0.5532	0.5449	0.5585	0.4912	0.4720	0.4606
	mean_t	2.3347	1.8393	1.4414	1.2182	1.1339	1.0286	0.9771	0.9011	0.8075	0.7681	0.7632	0.7650	0.6796	0.7252	0.6649	0.6198	0.6496
	med_t	1.6315	1.4785	1.0954	0.9817	0.9123	0.8301	0.7925	0.7412	0.6462	0.6247	0.6375	0.5944	0.5658	0.6386	0.5677	0.5323	0.4971
Hidden PnP	mean_r	1.8136	1.3940	1.1402	0.9256	0.8505	0.7569	0.7456	0.7023	0.6240	0.6154	0.6022	0.5694	0.5685	0.5423	0.5196	0.4836	0.4949
	med_r	1.4174	1.1641	0.9662	0.8011	0.7392	0.6673	0.6530	0.6256	0.5491	0.5306	0.5213	0.4951	0.4992	0.4905	0.4379	0.4194	0.4173
	mean_t	1.9815	1.5816	1.2749	1.0220	0.9419	0.8908	0.8221	0.7688	0.6978	0.6459	0.6452	0.6546	0.5755	0.5923	0.5489	0.5231	0.5183
	med_t	1.3354	1.1969	0.9179	0.7979	0.7418	0.7017	0.6140	0.5704	0.5393	0.5293	0.4917	0.4822	0.4424	0.4928	0.4494	0.3941	0.3843

Note: mean_r and med_r represent mean error and median error of rotation matrix, respectively. mean_t and med_t represent the mean error and median error of the translation vector, respectively.

Appendix B

Table A4. Error comparison in ordinary 3D case.

Algorithm		0.5	1	1.5	2	2.5	3	3.5	4	4.5	5
LHM	mean_r	0.1807	0.1807	0.3220	0.4175	0.5393	0.6030	0.7031	0.8386	0.9343	1.0227
	med_r	0.0972	0.1952	0.2646	0.3605	0.4667	0.5371	0.6364	0.7646	0.8217	0.9407
	mean_t	0.1086	0.1537	0.2190	0.2947	0.3502	0.4461	0.4919	0.5568	0.6207	0.6865
	med_t	0.0606	0.1140	0.1821	0.2359	0.2818	0.3645	0.4172	0.4595	0.5096	0.5810
EPnP+GN	mean_r	0.1141	0.2300	0.3324	0.4371	0.5675	0.6753	0.7855	0.9213	1.0472	1.1810
	med_r	0.1037	0.2047	0.2890	0.4032	0.5246	0.5990	0.7160	0.8451	0.9160	1.1029
	mean_t	0.0925	0.1924	0.2730	0.3683	0.4562	0.5988	0.6400	0.7036	0.8448	0.9137
	med_t	0.0736	0.1540	0.2159	0.3060	0.3794	0.4819	0.5323	0.5700	0.7159	0.7540
RPnP	mean_r	0.1275	0.2334	0.3549	0.4771	0.5938	0.7643	0.8491	0.9958	1.1019	1.2677
	med_r	0.1130	0.2050	0.3083	0.4229	0.5323	0.6433	0.7452	0.9071	0.9775	1.1195
	mean_t	0.1075	0.1907	0.3073	0.4024	0.5064	0.6449	0.7000	0.8506	0.9345	0.9841
	med_t	0.0813	0.1584	0.2391	0.3259	0.4066	0.5156	0.5525	0.6650	0.7564	0.8090
DLS	mean_r	0.4475	1.1487	1.1147	1.2296	1.1524	2.1032	2.5493	1.4194	2.5500	3.1087
	med_r	0.0915	0.1875	0.2599	0.3599	0.4670	0.5328	0.6348	0.7746	0.8233	0.9797
	mean_t	0.3853	0.9546	1.0905	0.8749	0.8964	1.8509	2.0846	1.2460	2.2659	2.0337
	med_t	0.0558	0.1086	0.1755	0.2320	0.2803	0.3647	0.4179	0.4737	0.5178	0.5843
OPnP	mean_r	0.1001	0.2037	0.2863	0.3836	0.4968	0.5846	0.6813	0.8209	0.8924	1.0151
	med_r	0.0905	0.1842	0.2625	0.3545	0.4642	0.5344	0.6272	0.7651	0.8177	0.9449
	mean_t	0.0677	0.1334	0.2054	0.2747	0.3345	0.4335	0.4747	0.5492	0.6042	0.6784
	med_t	0.0554	0.1084	0.1705	0.2353	0.2764	0.3592	0.4207	0.4682	0.4982	0.5691
SRPnP	mean_r	0.1002	0.2052	0.2914	0.3892	0.4973	0.5841	0.6909	0.8270	0.8988	1.0289
	med_r	0.0908	0.1855	0.2616	0.3586	0.4651	0.5233	0.6299	0.7614	0.8114	0.9360
	mean_t	0.0678	0.1331	0.2063	0.2744	0.3335	0.4323	0.4706	0.5472	0.5991	0.6751
	med_t	0.0554	0.1096	0.1698	0.2280	0.2730	0.3604	0.4165	0.4625	0.4893	0.5630
RDLT	mean_r	0.1001	0.2037	0.2863	0.3836	0.4968	0.5846	0.6813	0.8209	0.8924	1.0151
	med_r	0.0905	0.1842	0.2625	0.3545	0.4642	0.5344	0.6272	0.7651	0.8177	0.9449
	mean_t	0.0677	0.1334	10.2054	0.2747	0.3345	0.4335	0.4747	0.5492	0.6042	50.6784
	med_t	0.0554	0.1084	0.1705	0.2353	0.2764	0.3592	0.4207	0.4682	0.4982	0.5691
ASPnP	mean_r	0.1010	0.2082	0.2973	0.3996	0.5100	0.8267	0.7203	1.0774	0.9347	1.0732
	med_r	0.0913	0.1865	0.2621	0.3614	0.4719	0.5322	0.6358	0.7774	0.8271	0.9787
	mean_t	0.0679	0.1341	0.2070	0.2735	0.3373	0.5553	0.4797	0.6491	0.6145	0.6926
	med_t	0.0555	0.1091	0.1696	0.2272	0.2754	0.3629	0.4221	0.4652	0.5154	0.5753
Hidden PnP	mean_r	0.0964	0.1958	0.2689	0.3588	0.4567	0.5347	0.6182	0.7606	0.8220	0.9265
	med_r	0.0889	0.1773	0.2432	0.3353	0.4169	0.4913	0.5580	0.6941	0.7393	0.8596
	mean_t	0.0673	0.1317	0.2033	0.2689	0.3304	0.4242	0.4647	0.5345	0.5866	0.6613
	med_t	0.0555	0.1084	0.1681	0.2315	0.2703	0.3452	0.4085	0.4526	0.4962	0.5577

Table A5. Error comparison in Planar case.

Algorithm		0.5	1	1.5	2	2.5	3	3.5	4	4.5	5
LHM	mean_r	29.9013	31.7417	34.0830	30.4676	34.8833	34.1832	33.9616	34.6558	31.7799	31.9322
	med_r	2.8468	3.2152	2.7973	2.0785	3.6989	3.1051	3.1786	2.9233	3.2985	4.2504
	mean_t	3.3637	3.5275	3.9004	3.4480	4.0174	4.0321	3.5296	3.4999	4.0043	4.3222
	med_t	0.6659	0.8165	0.8331	0.9665	1.2537	1.3488	1.2499	1.4956	1.5673	1.7776
EPnP+GN	mean_r	46.8393	42.6777	51.0806	48.3854	46.9577	45.8880	52.0428	52.9036	45.1681	50.2341
	med_r	0.7138	1.4216	2.1556	3.1449	3.7999	4.4034	5.2400	6.0751	6.2405	7.4797
	mean_t	3.7485	3.5632	5.0009	4.7705	5.3529	5.6749	6.0509	6.7726	6.8051	7.6502
	med_t	0.3575	0.7303	1.2746	1.8901	2.2481	2.4254	3.0487	3.5236	3.4720	3.7619
RPnP	mean_r	0.4261	0.8353	1.1778	1.5130	1.9098	2.3771	3.0138	2.9731	4.1376	4.6917
	med_r	0.2416	0.4758	0.6824	0.8861	1.2338	1.3706	1.7477	1.8044	1.9868	2.4974
	mean_t	0.1888	0.3862	0.5620	0.7651	0.9524	1.1376	1.2889	1.4823	1.7577	2.3301
	med_t	0.1271	0.2688	0.4004	0.5474	0.6886	0.7919	0.8837	1.0852	1.1113	1.3150
DLS	mean_r	0.4276	1.2854	2.6917	0.9575	5.0002	4.2490	3.4537	5.8777	5.0284	5.3160
	med_r	0.1648	0.3662	0.5233	0.6719	0.9063	1.0596	1.2445	1.4611	1.5596	1.8049
	mean_t	0.2401	1.0773	2.0854	0.5296	4.8974	4.1841	2.2231	5.1008	4.3977	4.4116
	med_t	0.0898	0.1885	0.2912	0.3745	0.5265	0.5984	0.6815	0.9106	0.8918	1.0721
OPnP	mean_r	0.2326	0.4586	0.7364	0.8815	1.0890	1.3746	1.7562	1.8247	2.6162	2.3749
	med_r	0.1652	0.3593	0.5107	0.6574	0.8389	1.0283	1.1975	1.4204	1.4475	1.7539
	mean_t	0.1240	0.2519	0.3974	0.5193	0.6545	0.7879	0.9072	1.0739	1.2102	1.3622
	med_t	0.0873	0.1902	0.2806	0.3760	0.4851	0.5923	0.6681	0.8576	0.9000	1.0170
SRPnP	mean_r	0.2357	0.4629	0.7205	0.9008	1.2179	1.4794	1.6644	1.9671	2.2805	2.3890
	med_r	0.1658	0.3579	0.5173	0.6685	0.8743	1.0466	1.2245	1.4397	1.5172	1.7760
	mean_t	0.1258	0.2518	0.3946	0.5234	0.6682	0.7730	0.9051	1.0956	1.1898	1.3577
	med_t	0.0898	0.1874	0.2788	0.3717	0.4793	0.5806	0.6398	0.8341	0.8424	1.0107

Table A5. *Cont.*

Algorithm		0.5	1	1.5	2	2.5	3	3.5	4	4.5	5
RDLT	mean_r	0.2327	0.4588	0.7368	0.8807	1.0852	1.3764	1.7521	1.8329	2.5784	2.8207
	med_r	0.1652	0.3607	0.5124	0.6550	0.8386	1.0291	1.1959	1.4296	1.4404	1.7471
	mean_t	0.1240	0.2521	0.3974	0.5191	0.6542	0.7879	0.9071	1.0746	1.2163	1.3757
	med_t	0.0875	0.1910	0.2816	0.3782	0.4846	0.5868	0.6647	0.8616	0.8947	1.0055
ASPnP	mean_r	0.4148	0.9866	1.1999	0.9221	1.3332	1.8575	2.2183	2.3400	2.7756	2.8859
	med_r	0.1671	0.3639	0.5251	0.6741	0.9168	1.0721	1.2483	1.4691	1.5937	1.8008
	mean_t	0.2600	0.4219	0.4155	0.5228	0.7140	0.8517	1.0662	1.2743	1.2471	1.5128
	med_t	0.0899	0.1877	0.2916	0.3719	0.5261	0.5950	0.6790	0.8965	0.8823	1.0750
Hidden PnP	mean_r	0.2159	0.4265	0.6458	0.7841	0.9784	1.2362	1.3908	1.6459	1.7801	2.0570
	med_r	0.1573	0.3359	0.4634	0.6046	0.7988	0.9436	1.1092	1.2575	1.2435	1.5755
	mean_t	0.1197	0.2395	0.3678	0.4788	0.6268	0.7351	0.8315	0.9976	1.0773	1.2631
	med_t	0.0847	0.1761	0.2568	0.3386	0.4384	0.5490	0.5939	0.7586	0.7682	0.9011

Table A6. Error comparison in Quasi-singularity case.

Algorithm		0.5	1	1.5	2	2.5	3	3.5	4	4.5	5
LHM	mean_r	54.0364	54.3921	49.1851	54.7541	55.5350	53.8090	61.1037	50.8340	59.5363	48.8130
	med_r	0.5430	0.6938	0.9608	1.3668	1.9575	2.0970	2.7591	2.2852	3.1373	2.7422
	mean_t	24.0074	22.4198	20.3533	23.1433	23.9086	24.4829	24.6227	21.8371	25.1056	21.9802
	med_t	1.0878	1.1643	1.5035	2.2859	2.6978	3.3351	4.1897	3.2054	4.9388	3.9800
EPnP+GN	mean_r	0.2244	0.4413	0.6315	0.8488	1.7360	1.2971	2.6893	2.8364	3.2494	5.0486
	med_r	0.1970	0.3855	0.5353	0.7729	0.9831	1.1117	1.2841	1.5965	1.8046	2.0050
	mean_t	0.3342	0.6441	0.9640	1.2418	2.4658	1.9245	3.6031	3.7211	4.5948	6.5992
	med_t	0.2598	0.5030	0.7724	0.9210	1.4354	1.5420	1.9140	2.0268	2.4701	2.7201
RPnP	mean_r	0.2154	0.4220	0.6192	0.8204	1.0657	1.2264	1.5655	1.8108	2.7098	2.9224
	med_r	0.1904	0.3817	0.5750	0.7554	0.9199	1.1043	1.2703	1.5176	1.6604	1.8881
	mean_t	0.3290	0.6677	0.9760	1.2976	1.7074	2.0699	2.6763	3.5394	5.9400	6.7567
	med_t	00.2416	0.5224	0.7942	1.0313	1.3539	1.6545	1.8082	1.7970	2.3827	2.5840
DLS	mean_r	1.3246	0.9490	0.8534	2.3530	2.9240	4.3047	0.2423	4.3139	6.4904	7.9017
	med_r	0.1814	0.3607	0.5110	0.7116	0.8884	1.0614	1.2847	1.4287	1.6488	1.8967
	mean_t	1.3381	0.9327	1.1137	2.4278	2.9235	4.2191	3.2908	4.6732	5.9860	7.7954
	med_t	0.1857	0.3956	0.6232	0.7111	1.0305	1.1656	1.5168	1.4978	2.0095	2.2148
OPnP	mean_r	0.2009	0.3935	0.5741	0.7661	0.9848	1.1401	1.3736	1.5609	1.7385	1.9173
	med_r	0.1800	0.3576	0.5058	0.6969	0.8395	1.0358	1.1920	1.3928	1.5463	1.7345
	mean_t	0.2305	0.4519	0.7225	0.8622	1.1681	1.3680	1.6463	1.8462	2.0557	2.1876
	med_t	0.1890	0.3693	0.6191	0.6738	0.9407	1.2223	1.3506	1.4146	1.6998	1.7047
SRPnP	mean_r	0.2043	0.4002	0.5853	0.8017	1.0143	1.2321	1.4438	1.7237	2.3445	2.8959
	med_r	0.1831	0.3637	0.5166	0.7146	0.8596	1.0442	1.3145	1.3997	1.6407	1.8299
	mean_t	0.2170	0.4213	0.6510	0.7903	1.0547	1.2260	1.4596	1.6429	2.4650	2.9031
	med_t	0.1674	0.3443	0.5555	0.6205	0.8075	0.9431	1.1241	1.2359	1.6262	1.6205
RDLT	mean_r	0.2009	0.3935	0.5741	0.7661	0.9848	1.1401	1.3736	1.5609	1.7385	1.9173
	med_r	0.1800	0.3576	0.5058	0.6969	0.8395	1.0358	1.1920	1.3928	1.5463	1.7345
	mean_t	0.2305	0.4519	0.7225	0.8622	1.1681	1.3680	1.6463	1.8462	2.0557	2.1876
	med_t	0.1890	0.3693	0.6191	0.6738	0.9407	1.2223	1.3506	1.4146	1.6998	1.7047
ASPnP	mean_r	0.2086	0.4022	0.5988	0.8428	1.0503	1.3127	1.6188	1.7450	2.0989	2.6162
	med_r	0.1831	0.3627	10.5136	0.7179	0.8674	1.0355	1.3072	1.4347	1.6401	1.8674
	mean_t	0.2298	0.4605	0.7340	0.8666	1.2207	1.4245	1.7956	2.0231	2.5034	2.7405
	med_t	0.1833	0.3788	0.6139	0.6991	1.0133	1.1224	1.4662	1.4969	1.8896	2.0720
Hidden PnP	mean_r	0.1951	0.3781	0.5507	0.7379	0.9307	1.1099	1.3035	1.4450	1.7213	1.8770
	med_r	0.1693	0.3463	0.4822	0.6619	0.7891	0.9849	1.1354	1.2461	1.4599	1.6329
	mean_t	0.2150	0.4134	0.6432	0.7656	1.0164	1.1514	1.3801	1.5349	1.8260	1.8857
	med_t	0.1665	0.3278	0.5525	0.6121	0.8036	0.8853	1.0788	1.0622	1.4421	1.3561

References

1. Fischler, M.A.; Bolles, R.C. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM* **1981**, *24*, 381–395. [CrossRef]
2. Gai, S.; Jung, E.J.; Yi, B.J. Multi-group localization problem of service robots based on hybrid external localization algorithm with application to shopping mall environment. *Intell. Serv. Robot.* **2016**, *9*, 257–275. [CrossRef]
3. Hijikata, S.; Terabayashi, K.; Umeda, K. A simple indoor self-localization system using infrared LEDs. In Proceedings of the 2009 Sixth International Conference on Networked Sensing Systems (INSS), Pittsburgh, PA, USA. 17–19 June 2009; pp. 1–7. [CrossRef]
4. Daniilidis, K. Hand-Eye Calibration Using Dual Quaternions. *Int. J. Robot. Res.* **1999**, *18*, 286–298. [CrossRef]
5. Kelsey, J.; Byrne, J.; Cosgrove, M.; Seereeram, S.; Mehra, R. Vision-based relative pose estimation for autonomous rendezvous and docking. In Proceedings of the 2006 IEEE Aerospace Conference, Big Sky, MT, USA, 4–11 March 2006; p. 20 . [CrossRef]
6. Yao, N. Research of Monocular Vision Based Target Tracking and Positioning Techniques. Ph.D. Thesis, Shanghai Jiao Tong University, Shanghai, China, 2014.

7. Park, J.S.; Lee, B.J. Vision-based real-time camera match moving using a known marker. *Opt. Eng.* **2008**, *47*, 027201. [CrossRef]
8. Tsai, R.Y. An efficient and accurate camera calibration technique fro 3d machine vision. In Proceedings of the CVPR'86, Miami Beach, FL, USA, 22–26 June 1986; pp. 364–374.
9. Shahzad, M.G.; Roth, G.; Mcdonald, C. Robust 2D Tracking for Real-Time Augmented Reality. 2002. Available online: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=df4bae2e7b7000f80c2ebc01f64a8832b0d6330b (accessed on 7 January 2023).
10. Abdel-Aziz, Y.; Karara, H.; Hauck, M. Direct Linear Transformation from Comparator Coordinates into Object Space Coordinates in Close-Range Photogrammetry. *Photogramm. Eng. Remote Sens.* **2015**, *81*, 103–107. [CrossRef]
11. David, P.; Dementhon, D.; Duraiswami, R.; Samet, H. SoftPOSIT: Simultaneous pose and correspondence determination. *Int. J. Comput. Vis.* **2004**, *59*, 259–284. [CrossRef]
12. Lu, C.P.; Hager, G.; Mjolsness, E. Fast and globally convergent pose estimation from video images. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *22*, 610–622. [CrossRef]
13. Lepetit, V.; Moreno-Noguer, F.; Fua, P. Epnp: An accurate $O(n)$ solution to the pnp problem. *Int. J. Comput. Vis.* **2009**, *81*, 155–166. [CrossRef]
14. Hesch, J.A.; Roumeliotis, S.I. A Direct Least-Squares (DLS) method for PnP. In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 383–390. [CrossRef]
15. Li, S.; Xu, C.; Xie, M. A Robust $O(n)$ Solution to the Perspective-n-Point Problem. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 1444–1450. [CrossRef] [PubMed]
16. Zheng, Y.; Sugimoto, S.; Okutomi, M. ASP*n*P: An Accurate and Scalable Solution to the Perspective-n-Point Problem. *IEICE Trans. Inf. Syst.* **2013**, *E96.D*, 1525–1535. [CrossRef]
17. Zheng, Y.; Kuang, Y.; Sugimoto, S.; Åström, K.; Okutomi, M. Revisiting the PnP Problem: A Fast, General and Optimal Solution. In Proceedings of the 2013 IEEE International Conference on Computer Vision, Sydney, Australia, 1–8 December 2013; pp. 2344–2351. [CrossRef]
18. Kneip, L.; Li, H.; Seo, Y. Upnp: An optimal $O(n)$ solution to the absolute pose problem with universal applicability. In *Computer Vision—ECCV 2014, Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014*; Springer: Cham, Switerland, 2014; pp. 127–142. [CrossRef]
19. Nakano, G. Globally Optimal DLS Method for PnP Problem with Cayley parameterization. In Proceedings of the British Machine Vision Conference (BMVC), Swansea, UK, 7–10 September 2015; Xie, X., Jones, M.W., Tam, G.K.L., Eds.; BMVA Press: Malvern, UK, 2015; pp. 78.1–78.11. [CrossRef]
20. Kukelova, Z.; Bujnak, M.; Pajdla, T. Automatic generator of minimal problem solvers. In *Computer Vision—ECCV 2008, Proceedings of the European Conference on Computer Vision, Marseille, France, 12–18 October 2008*; Springer: Cham, Switzerland, 2008; pp. 302–315. [CrossRef]
21. Ene, V.; rgen Herzog, J. *Gröbner Bases in Commutative Algebra*; American Mathematical Society: Providence, RI, USA, 2011; Volume 130, p. 164.
22. Wang, P.; Xu, G.; Cheng, Y.; Yu, Q. A simple, robust and fast method for the perspective-n-point Problem. *Pattern Recognit. Lett.* **2018**, *108*, 31–37. [CrossRef]
23. Wang, P.; Zhou, X.; An, A.; He, Q.; Zhang, A. Robust and linear solving method for Perspective-n-Point problem. *Chin. J. Sci. Instrum.* **2020**, *41*, 271–280. [CrossRef]
24. Jiabao, W.; Shirong, Z.; Qingya, Z. Vision based real-time 3D displacement measurement using weighted iterative EPnP algorithm. *Chin. J. Sci. Instrum.* **2020**, *41*, 166–175.
25. Zhang, Z. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *22*, 1330–1334. [CrossRef]