Revisiting the P*n*P Problem: A Fast, General and Optimal Solution

Yinqiang Zheng[†] Yubin Kuang[‡] Shigeki Sugimoto[†] Kalle Åström[‡] Masatoshi Okutomi[†]
[†]Department of Mechanical and Control Engineering, Tokyo Institute of Technology, JAPAN
{zheng,shige}@ok.ctrl.titech.ac.jp mxo@ctrl.titech.ac.jp
[‡]Centre for Mathematical Sciences, Lund University, SWEDEN
{yubin,kalle}@maths.lth.se

Abstract

In this paper, we revisit the classical perspective-n-point (PnP) problem, and propose the first non-iterative $O(n)$ solution that is fast, generally applicable and globally optimal. Our basic idea is to formulate the PnP problem into a functional minimization problem and retrieve all its stationary points by using the Gröbner basis technique. The novelty lies in a non-unit quaternion representation to parameterize the rotation and a simple but elegant formulation of the PnP problem into an unconstrained optimization problem. Interestingly, the polynomial system arising from its first-order optimality condition assumes two-fold symmetry, a nice property that can be utilized to improve speed and numerical stability of a Gröbner basis solver. Experiment results have demonstrated that, in terms of accuracy, our proposed solution is definitely better than the state-of-the-art $O(n)$ methods, and even comparable with the reprojection error minimization method.

1. Introduction

Given n ($n \geq 3$) 3D reference points in the object framework and their corresponding 2D projections, to determine the orientation and position of a fully calibrated perspective camera is known as the perspective-n-point (PnP) problem [9]. It has widespread applications in augmented reality, incremental structure-from-motion, robot localization and so on. Considering its importance, there is no surprise that, in the past few decades, a huge amount of works have addressed this problem. Unfortunately, to the best of our knowledge, there does not exist a fast (preferably, real-time) and globally optimal solution, which is accurate and applicable to a PnP problem with any point number n ($n \geq 3$), any 3D point configuration and arbitrary camera pose.

1.1. Literature Review

The minimal P3P problem has been systematically investigated in the literature, such as [5] and the recent work [12]. In practice, a P3P solution is usually used in combination with RANSAC [4] to remove outliers. Considering that data redundancy generally contributes to improving accuracy, most of existing works on PnP focus on overconstrained cases with more than three points. To properly account for the latest progress, we would like to roughly categorize them into two groups - *the multi-stage methods* and *the direct minimization methods*.

Typically, *the multi-stage methods* first estimate the coordinates of some (or all) points in the camera framework, and transform the PnP problem into the 3D-3D absolute pose problem, for which there exist closed-form solutions [21]. There are a few works, like [11], dedicated to P4P or P5P, whose application is limited due to the restriction of point number. To relax this restriction, two linear solutions were presented in [18] and [1], with respective computational complexity $O(n^5)$ and $O(n^8)$, which are, at least in theory, applicable to general PnP with $n \geq 4$. However, they are inaccurate when n is small, due to ignoring some nonlinear constraints. On the contrary, when n is large, their speed is slow because of their high complexity.

Lepetit *et al*. [14] introduced several virtual control points to represent the 3D reference points, and successfully reduced the complexity to $O(n)$. As pointed out in [15], its accuracy is low for slightly redundant cases with $n = 4$ or $n = 5$, due to its underlying linearization scheme. To improve accuracy, Li *et al*. [15] proposed another non-iterative $O(n)$ solution, which estimates the coordinates of two special endpoints and ignores only one nonlinear constraint.

Without estimating point coordinates, the well-known direct linear transformation (DLT) method [9] is also a multi-stage method, since it first determines the projection matrix and then extracts the calibration parameters and the camera pose. In the scenario of a calibrated camera, the DLT method is quite inaccurate due to overlooking the known calibration parameters.

To sum up, all the aforementioned multi-stage methods are usually poor in accuracy, due primarily to ignoring some nonlinear correlation. This is especially true for a PnP prob-

lem with a few points, in which the accuracy loss can hardly be compensated by data redundancy. In addition, without a clearly defined objective function, these methods do not assume overall global optimality, even supposing that there exists an optimal solution at each stage.

In contrast, *the direct minimization methods* are characteristic of minimizing a properly defined error function, either in the image space or the object space, while taking into consideration all nonlinear constraints. It is widely known that minimizing the reprojection error is the best criterion, which leads to a challenging nonconvex fractional programming problem. Olsson *et al*. [17] proposed a branch-and-bound method to retrieve its global optimum. Unfortunately, it can rarely be used in practice due to its tremendous computational cost.

As a trade off, some direct minimization methods [6, 16] minimize instead certain algebraic error functions via local optimization techniques. For example, Lu *et al*. [16] developed an orthogonal iteration method to directly minimize the object space error, while Garro *et al*. [6] offered an alternating minimization method to minimize an algebraic error defined in the image space. However, these local optimization based methods suffer from the risk of getting trapped into local minimum, and provide poor results when they indeed do so. Schweighofer and Pinz [19] partially addressed the problem of multiple local minima with a planar target, but failed to provide a general solution.

The work [20] tried to avoid local minimum by relaxing the P*n*P problem into a semidefinite programme (SDP). The major drawback lies in that the relaxation is usually not tight. It is also inappropriate for real-time applications due to the dependence on an off-the-shelf SDP solver, in spite of its $O(n)$ complexity.

The above direct minimization methods [6, 16, 17, 20] share another shortage that they return only a single solution, which might not correspond to the true camera pose in case of multiple solutions.

To resolve these drawbacks, Hesch and Roumeliotis [10] developed a direct least square (DLS) method with complexity $O(n)$, in which all stationary points are retrieved by solving the polynomial system derived from its first-order optimal condition via the resultant technique. Unfortunately, they parameterized rotation by using the Cayley representation, which is degenerate in all cases of 180 degree rotations around the x-, y- and z-axis[1]. The accuracy deteriorates seriously when the camera pose approaches these singularities.

As pointed out in [15], in addition to the number of points n, the configuration of the 3D reference points plays a critical role as well. A desirable P*n*P solution should be able to handle all 3D point configurations, including the ordinary-3D, the planar and the quasi-singular (near-planar or near-linear) configuration. However, some existing methods, like [14, 20], handle the ordinary-3D and the planar configuration separately, thus tend to be inaccurate in the in-between quasi-singular case. Additionally, such works as [19] dedicate to the planar case, which are inapplicable to the other two configurations at all.

[1]On the project page, Hesch and Roumeliotis provided a remedy by solving DLS three times under different rotated 3D points. Since the computational time would be tripled, this remedy is not attractive, especially considering that DLS itself is not very fast. In addition, such a remedy harms the theoretical elegance of global optimality.

1.2. Overview of the Proposed Solution

In this paper, we propose the first non-iterative $O(n)$ solution that is fast, globally optimal and universally applicable. Our basic idea is to formulate the P*n*P problem into a minimization problem and retrieve all its stationary points by using the Gröbner basis technique. It is therefore a direct minimization method. By using a unusual non-unit quaternion representation to parameterize rotation, we formulate the P*n*P problem into an unconstrained optimization problem. The polynomial system arising from its first-order optimality condition is simpler than using the standard unit quaternion parameterization. More interestingly, this polynomial system is of odd-degree, thus assuming two-fold symmetry, a nice property that can be utilized to improve speed and numerical stability of a Gröbner basis solver. Being globally optimal, our proposed solution successfully conquers the problem of local optimality (or even divergence) that might upset a local optimization based method. It is capable of retrieving all solutions, when multiple solutions indeed exist. Unlike [10], our solution does not suffer from any degeneracy of camera pose.

Experiment results have demonstrated that, in terms of accuracy, our proposed solution is definitely better than the examined state-of-the-art methods. Actually, although our cost function is only algebraically meaningful, its accuracy is even comparable to that of the reprojection error minimization method. Theoretically speaking, the computational complexity of our solution is $O(n)$. However, we have empirically observed that its computational time keeps almost constant even with thousands of points. Therefore, the proposed solution is universally applicable to any P*n*P problem, irrespective of the 3D point configuration, the camera pose and the number of points.

2. Mathematical Formulation

Throughout this paper, matrices, vectors and scalars are denoted by using capital letters, bold lowercase letters and plain lowercase letters, respectively. One exception is that the capital letter T represents matrix or vector transpose. All vectors are column-wise in default.

2.1. Preliminaries of the PnP Problem

Given n 3D reference points $\mathbf{q}_i = \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}^T, i = 1, 2, \cdots, n$, in the object reference framework, and their corresponding projections $\mathbf{p}_i = \begin{bmatrix} u_i & v_i & 1 \end{bmatrix}^T$, the P$n$P problem aims to retrieve the rotation matrix R and the translation vector $\mathbf{t}$, accounting for camera orientation and position, respectively. Considering that the perspective camera has been calibrated, we simply assume that the projections $\mathbf{p}_i$ are measured in the normalized homogeneous image coordinate framework. The perspective imaging equation reads

$$\lambda_i \mathbf{p}_i = R\mathbf{q}_i + \mathbf{t}, i = 1, 2, \cdots, n, \tag{1}$$

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

2.2. Rotation Parameterization

A critical issue is how to parameterize the rotation matrix R, such that the orthonormal constraint $RR^T = I$ and the determinant constraint $det(R) = 1$ could be satisfied.

There are various parameterization methods for R, such as the Euler angle, rotation axis-angle, Cayley and unit quaternion parameterization. To facilitate global optimization via polynomial system solving, we advocate instead the non-unit quaternion parameterization, which is free of any trigonometric function. Specifically, letting $s = a^2 + b^2 + c^2 + d^2$, the non-unit quaternion parameterization reads

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

where a, b, c, d are the four unknown parameters. It is straightforward to verify that the parameterization in Eq.(2) satisfies $RR^T = I$ and $det(R) = 1$.

At first sight, the above parameterization is unattractive at all. First of all, one has to make sure that s is rigorously positive, i.e., a, b, c, d are not simultaneously zero, so as to avoid the singularity of dividing by zero. Secondly, Eq.(2) introduces a fractional term $\frac{1}{s}$, which is difficult to handle.

Fortunately, we have recognized that a, b, c, d in Eq.(2) assume scale and sign ambiguity, *i.e.*, $R(a, b, c, d) = R(ka, kb, kc, kd)$, for any nonzero k. It is possible to exploit this property to resolve the concerns on the non-unit quaternion representation.

2.3. The Unconstrained Minimization Problem

Since the absolute scale of a, b, c, d in Eq.(2) is arbitrary, we can fix it by using the reciprocal of the average depth, *i.e.*, $s \equiv \frac{1}{\frac{1}{n}\sum_{i=1}^n \lambda_i} = \frac{1}{\bar{\lambda}}$. Due to the chirality condition [9], the average depth $\bar{\lambda}$ is rigorously positive. The possibility of dividing by zero has thus been naturally avoided.

Now we multiply $\frac{1}{\bar{\lambda}}$ at both sides of Eq.(1) and obtain the following equation

$$\hat{\lambda}_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1^T \\ \mathbf{r}_2^T \\ \mathbf{r}_3^T \end{bmatrix} \mathbf{q}_i + \begin{bmatrix} \hat{t}_1 \\ \hat{t}_2 \\ \hat{t}_3 \end{bmatrix}, i = 1, 2, \cdots, n, \tag{3}$$

in which $\hat{\lambda}_i = \frac{\lambda_i}{\bar{\lambda}}$, $\begin{bmatrix} \hat{t}_1 & \hat{t}_2 & \hat{t}_3 \end{bmatrix}^T = \frac{1}{\bar{\lambda}}\mathbf{t}$, and

$$\begin{bmatrix} \mathbf{r}_1^T \\ \mathbf{r}_2^T \\ \mathbf{r}_3^T \end{bmatrix} = \begin{bmatrix} a^2+b^2-c^2-d^2 & 2bc-2ad & 2bd+2ac \\ 2bc+2ad & a^2-b^2+c^2-d^2 & 2cd-2ab \\ 2bd-2ac & 2cd+2ab & a^2-b^2-c^2+d^2 \end{bmatrix}. \tag{4}$$

It is straightforward to recognize that

$$\sum_{i=1}^n \hat{\lambda}_i = n\frac{\sum_{i=1}^n \lambda_i}{\sum_{i=1}^n \lambda_i} = n. \tag{5}$$

In addition, from Eq.(3), we have

$$\hat{\lambda}_i = \mathbf{r}_3^T \mathbf{q}_i + \hat{t}_3, i = 1, 2, \cdots, n. \tag{6}$$

By combining Eq.(5) and Eq.(6), we can express $\hat{t}_3$ via

$$\hat{t}_3 = 1 - \mathbf{r}_3^T(\frac{1}{n}\sum_{i=1}^n \mathbf{q}_i) = 1 - \mathbf{r}_3^T \bar{\mathbf{q}}, \tag{7}$$

where $\bar{\mathbf{q}}$ represents the centroid of the 3D points.

After plugging $\hat{\lambda}_i = 1 + \mathbf{r}_3^T(\mathbf{q}_i - \bar{\mathbf{q}}) = 1 + \mathbf{r}_3^T \tilde{\mathbf{q}}_i$ back into Eq.(3), we have the following equation

$$(1 + \mathbf{r}_3^T \tilde{\mathbf{q}}_i)\begin{bmatrix} u_i \\ v_i \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1^T \\ \mathbf{r}_2^T \end{bmatrix} \mathbf{q}_i + \begin{bmatrix} \hat{t}_1 \\ \hat{t}_2 \end{bmatrix}, i = 1, 2, \cdots, n, \tag{8}$$

where $\tilde{\mathbf{q}}_i$ denotes the i-th 3D point after centralization.

Until now, we have implicitly assumed that the projections are noise-free. Due to noise, Eq.(8) could not be completely satisfied in general. Therefore, we directly minimize the sum of the squared error as our cost function

$$\begin{aligned} \min_{a,b,c,d,\hat{t}_1,\hat{t}_2} \quad & \sum_{i=1}^n [(1 + \mathbf{r}_3^T \tilde{\mathbf{q}}_i)u_i - \mathbf{r}_1^T \mathbf{q}_i - \hat{t}_1]^2 \\ & + \sum_{i=1}^n [(1 + \mathbf{r}_3^T \tilde{\mathbf{q}}_i)v_i - \mathbf{r}_2^T \mathbf{q}_i - \hat{t}_2]^2. \end{aligned} \tag{9}$$

Although it is only an algebraic error, as will be demonstrated in the experiment section, its accuracy is very close to that of minimizing the reprojection error, *i.e.* the gold-standard in multiview geometry [9].

Before really solving Eq.(9), we can easily project out $\hat{t}_1$ and $\hat{t}_2$ in closed-form as follows

$$\begin{aligned} \hat{t}_1 &= \bar{u} + \mathbf{r}_3^T(\frac{1}{n}\sum_{i=1}^n u_i \tilde{\mathbf{q}}_i) - \mathbf{r}_1^T \bar{\mathbf{q}}, \\ \hat{t}_2 &= \bar{v} + \mathbf{r}_3^T(\frac{1}{n}\sum_{i=1}^n v_i \tilde{\mathbf{q}}_i) - \mathbf{r}_2^T \bar{\mathbf{q}}, \end{aligned} \tag{10}$$

in which $[\bar{u}, \bar{v}]^T$ is the centroid of the image projections in the normalized image coordinate system.

Now letting $\boldsymbol{\alpha} = [1, a^2, ab, ac, ad, b^2, bc, bd, c^2, cd, d^2]^T$ and plugging Eq.(10) into Eq.(9), we rewrite the cost function into the matrix form

$$\min_{a,b,c,d} \quad f(a,b,c,d) = \|M\boldsymbol{\alpha}\|_2^2 = \boldsymbol{\alpha}^T M^T M \boldsymbol{\alpha}, \tag{11}$$

where M is a 2n×11 matrix that can be constructed by using $\mathbf{p}_i$ and $\mathbf{q}_i$.

Eq.(11) is our ultimate optimization problem, which does not include any trigonometric function nor any constraint. In addition, it suffers from no degeneracy of camera pose, and is independent of the 3D point configuration.

2.4. Relation to Existing Works

In the previous section, we have used the non-unit quaternion to parameterize the rotation, and fixed its scale by using the reciprocal of the average depth. Actually, we are able to interpret some existing works in terms of how to fix the scale of Eq.(2).

The unit quaternion was used in [20]. It is nothing but to fix the scale in Eq.(2) by using the unit norm constraint $a^2 + b^2 + c^2 + d^2 = 1$. According to [20], the PnP problem can be formulated into a constrained optimization problem

$$\min_{a,b,c,d} \quad \hat{\boldsymbol{\alpha}}^T \hat{M}^T \hat{M} \hat{\boldsymbol{\alpha}}, \; s.t., a^2 + b^2 + c^2 + d^2 = 1, \tag{12}$$

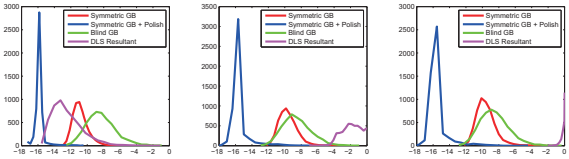
in which $\hat{M}$ is a 2n×10 data matrix and $\hat{\boldsymbol{\alpha}}$ equals $\boldsymbol{\alpha}$ after removing the first element.

Similar to our formulation, it does not suffer from any degeneracy. To find its global optimum, [20] used convex relaxation techniques, which usually provide an approximate solution, rather than the guaranteed global optimum.

The Cayley parameterization was used in [10]. Through some basic operations, one can verify that the Cayley parameterization is the same as $R(1, b, c, d)$, that is, fixing the scale of Eq.(2) by using $a = 1$. The major advantage lies in that the polynomial system in [10] is simpler to solve, since there remain only three variables. Unfortunately, this scheme is unstable in case of near-Cayley-degenerate rotations ($a \approx 0$), and inapplicable at all in case of Cayley-degenerate rotations ($a = 0$).

3. Global Optimization Method

Global optimization has attracted a lot of attention in multiview geometry, see [8] for a review. Such popular techniques as branch-and-bound and convex relaxation are usually time-consuming and only capable of retrieving one (approximate) optimal solution. Here, we prefer to solve the polynomial system of the first-order optimality condition of Eq.(11), so as to identify all stationary points.



(a) Random (b) Near degenerate (c) Degenerate

Figure 1. Numerical stability of the polynomial system solvers. The investigated solvers include the blind GB solver without utilizing symmetry (Blind GB), the GB solver using two-fold symmetry (Symmetric GB), the Symmetric GB followed by one damped Newton polishing step (Symmetric GB + Polish) and the resultant based solver used in DLS [10]. We randomly generate 50 ordinary-3D points and simulate their noise-free projections. Fully random, near-Cayley-degenerate and Cayley-degenerate rotations are used in (a), (b) and (c), respectively. The horizontal axis shows the log_{10} value of the absolute error between the ground-truth unit-norm quaternion and the estimated quaternion after normalization, while the vertical axis shows the counts over 5,000 independent runs. The instability problem of DLS in (b) and (c) is obvious.

By calculating the derivative of Eq.(11) with respect to a, b, c, d, the first-order optimality condition reads

$$\frac{\partial f}{\partial a} = 0, \frac{\partial f}{\partial b} = 0, \frac{\partial f}{\partial c} = 0, \frac{\partial f}{\partial d} = 0, \tag{13}$$

which is composed of four three-degree polynomials with respect to a, b, c, d.

3.1. A Blind Gröbner Basis Solver

Although solving multivariate polynomial systems is challenging in general, the multiview geometry community has achieved much progress by means of the Gröbner basis (GB) technique [3]. Kukelova *et al.* [13] even developed an automatic generator of GB solvers, which facilitates the solving of polynomial systems arising from geometric computer vision problems. The basic procedure is first to determine the Gröbner bases and the monomial bases of the quotient ring under the graded reverse lexicographical ordering, and then to construct the elimination template that determines which polynomials from the ideal should be added so as to build the action matrix. Finally, the solutions to the original polynomial system are extracted from the eigen-factorization of the action matrix. The readers are referred to [13] for more details on the automatic generator and to [3] for general theories.

By using the automatic generator in [13], we have found that the polynomial system in Eq.(13) has at most 81 solutions. The size of the elimination template is 575×656, while that of the action matrix is 81×81. The generated GB solver takes about 37.2 milliseconds (ms).

One might be interested in solving the polynomial system arising from the first-order optimality condition of Eq.(12), so as to retrieve the guaranteed optimal solution(s).

Due to the unit-norm constraint, a Lagrange multiplier has to be introduced, thus leading to a three-degree polynomial system with respect to five variables. The GB solver automatically generated by [13] is much more complex (*e.g.*, the elimination template is of size 1523×1603) and thus much slower. This verifies the advantages of our non-unit quaternion parameterization and our unconstrained formulation.

3.2. Utilizing Two-Fold Symmetry

By carefully investigating Eq.(11), we have further noted that the polynomial system in Eq.(13) is of odd-degree. Specifically, the polynomials in Eq.(13) include three-degree and one-degree monomials only. It therefore assumes two-fold symmetry, that is, $\mathbf{w} = \mathbf{0}$ is a trivial solution, and, if any non-zero $\mathbf{w}$ is a solution, so is -$\mathbf{w}$, where $\mathbf{w} = [a, b, c, d]^T$. Actually, instead of 81 solutions, there are at most 40 independent solutions to Eq.(13), which indicates the complexity of the PnP problem with general n.

Very recently, Ask *et al.* [2] developed general techniques to make use of p-fold symmetry ($p = 2$ in our problem) arising from some minimal problems. The basic idea is to directly eliminate the trivial all-zero solution and carefully generate new equations such that the symmetry could be preserved. By using symmetry, the size of the elimination template and that of the action matrix could be drastically reduced, which in return improves computational speed and numerical stability.

In [2], one has to solve an integer linear system to extract the symmetric solutions from the action matrix, which is very slow. When implementing the two-fold symmetry GB solver for Eq.(13), we have improved the solution extraction operation by using the problem structure. We refer the readers to our source code and the supplementary materials for the implementation details. With an elimination template of size 348×376 and an action matrix of size 40×40, our two-fold symmetry GB solver takes about 18.5 ms, about twice as fast as the blind GB solver. As shown in Fig.1, its numerical stability is also stronger than the blind version.

It is worthy of mentioning that the five-variable polynomial system from Eq.(12) does not assume full symmetry, because the introduced Lagrange multiplier is always positive. Actually, Eq.(13) is the first fully symmetric system, arising from a non-minimal problem, that we know of.

3.3. Solution Polishing and Extraction

After obtaining all stationary points of Eq.(11), we can further improve the numerical stability by polishing them via a single damped Newton step. Specifically, assuming that $\mathbf{w}$ is a stationary point, we polish $\mathbf{w}$ through the updating rule $\mathbf{w} \leftarrow \mathbf{w} - \Delta\mathbf{w}$. The increment $\Delta\mathbf{w}$ is determined by $\Delta\mathbf{w} = (\frac{\partial^2 f}{\partial^2 \mathbf{w}} + \mu I)^{-1} \frac{\partial f}{\partial \mathbf{w}}$, in which the damped factor μ is chosen such that $f(\mathbf{w} - \Delta\mathbf{w}) \leq f(\mathbf{w})$.

As shown in Fig.1, the polishing strategy can drastically improve the numerical precision, although only one damped Newton step is used. In addition, the computational cost is almost negligible, because the dimension of $M^T M$ in Eq.(11) is fixed.

After the polishing step, we only retain those real and physically feasible stationary points with positive definite Hessian, *i.e.* minima. When $n \geq 6$, the PnP problem has a unique solution in general. Therefore, we choose the stationary point with smallest objective value in Eq.(11) as the final solution.

In the slightly redundant scenarios with $4 \leq n \leq 5$, it is a little bit complicated. We have observed a few extreme cases, in which two widely different stationary points have almost the same objective value, yet the objective value of the correct stationary point is even slightly larger. Therefore, when $4 \leq n \leq 5$, we return all remaining minima to the end user, who might be able to choose the correct one by using, *e.g.*, motion coherence in a tracking scenario. In the minimal $n = 3$ case, we use the same strategy.

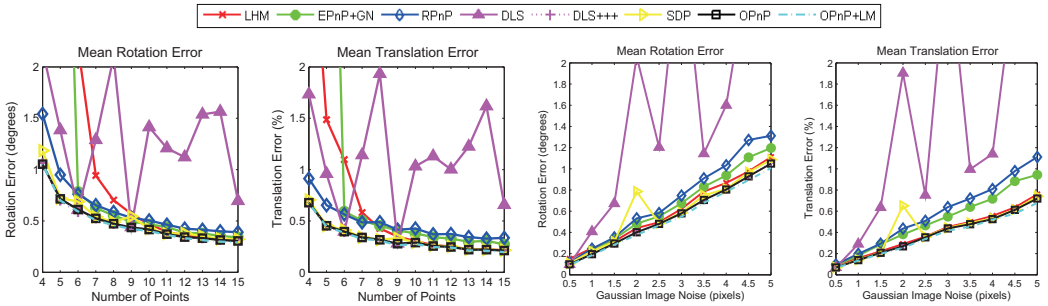
4. Experiment Results

In this section, we experimentally investigate our optimal solution to the PnP problem, referred to as **OPnP**, and compare it with the state-of-the-art solutions. For the ordinary-3D case and the quasi-singular case, we consider two multi-stage methods, including **EPnP+GN** together with a few Gauss-Newton steps [14] and **RPnP** [15], as well as three direct minimization based methods, including the direct least square solution (**DLS**) [10], the approximate optimal solution by using SDP convex relaxation (**SDP**) [20] and the popular iterative method by Lu *et al.* [16], denoted by **LHM** in short. For the planar case, we include into comparison **EPnP** without Gauss-Newton steps, **RPnP**, **DLS** and **SDP**. In addition, the iterative method in [19] specialized to the planar case is also considered, which is denoted by **SP+LHM**. The authors of **DLS** [10] provided a remedy to conquer the degeneracy of the Cayley representation by solving **DLS** three times under differently rotated 3D points. We include into comparison this remedy (**DLS+++**) as well.

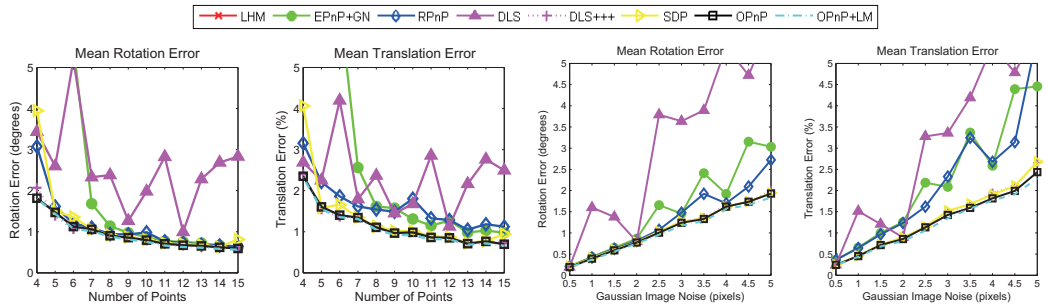
Considering that our objective function is only algebraically meaningful, it might be of great interest to compare it with the reprojection error minimization method. Unfortunately, the branch-and-bound method in [17] is very slow. In addition, it returns a single solution, which might be totally different from the ground-truth when $4 \leq n \leq 5$. Therefore, we minimize the reprojection error by using the Levenberg-Marquardt method, starting from the solution(s) from **OPnP**. We denote it by **OPnP+LM**.

We implement our **OPnP** solution in MATLAB[2], which

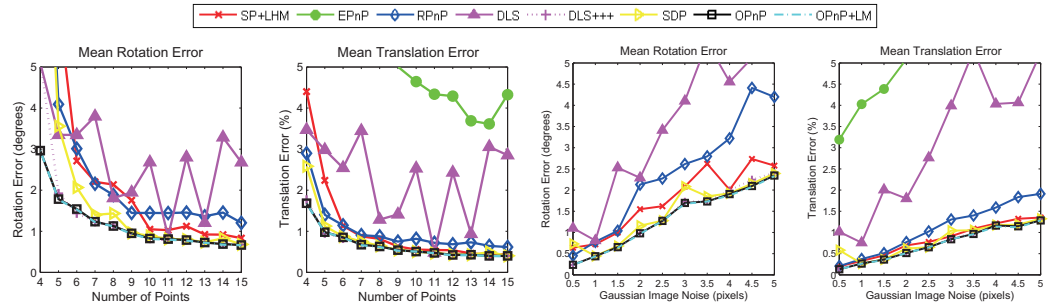
[2]Our source code and scripts to reproduce all results are available at https://sites.google.com/site/yinqiangzheng/.



(a) Ordinary-3D Point Configuration



(b) Quasi-Singular Point Configuration (Note that **LHM** lies outside the visible region)



(c) Planar Point Configuration

Figure 2. Experiment results w.r.t. varying point numbers (1st and 2nd columns, δ=2 pixels) and varying noise levels (3rd and 4th columns, n=10) in case of ordinary-3D, quasi-singular and planar point configuration, shown in (a), (b) and (c), respectively.

incorporates the symmetric GB solver and the polishing strategy. We use publicly available source codes of **EP*n*P**, **EP*n*P+GN**, **RP*n*P**, **DLS** (the fast version), **DLS+++** and **SDP**. **LHM** and **SP+LHM** are available in the toolbox provided in [15]. We run all codes in MATLAB on a notebook with 2.8GHz CPU and 4GB RAM.

4.1. Simulated Data

We assume a virtual perspective camera with image resolution of 640×480 pixels and focal length 800 pixels. The principle point lies in the image center. n 3D reference points are randomly generated in the camera framework. For the ordinary-3D case, these points are randomly distributed in the x-, y- and z-range of [-2,2]×[-2,2]×[4,8], while for the quasi-singular case, they are in the range of [1,2]×[1,2]×[4,8]. Then, we choose the ground-truth translation $\mathbf{t}_{true}$ such that the origin of the object framework coincides with the centroid of these 3D points, and rotate these 3D points by using a randomly generated ground-truth rotation matrix R_{true}. We measure the absolute error in degrees between R_{true} and the estimated rotation matrix R, which is defined as $e_{rot}(degrees) = \max_{k=1}^{3} acos(dot(\mathbf{r}_{true}^k, \mathbf{r}^k)) \times 180/\pi$, where $\mathbf{r}_{true}^k$ and $\mathbf{r}^k$ are the k-th column of R_{true} and R, and $dot(\cdot,\cdot)$ and $acos(\cdot)$ represent the dot product and arccosine operation, respectively. The translation error is measured by the relative difference between $\mathbf{t}_{true}$ and $\mathbf{t}$ defined as $e_{trans}(\%) = \|\mathbf{t}_{true} - \mathbf{t}\|/\|\mathbf{t}\| \times 100$.

4.1.1 Varying Point Numbers and Noise Levels

We first vary the point number n from 4 to 15, and add zero-mean Gaussian noise with fixed deviation δ=2 pixels onto

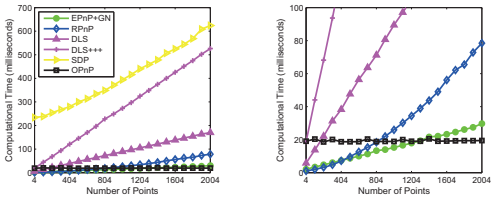


Figure 3. Running time w.r.t. varying number of points. A close-up is shown at the right.

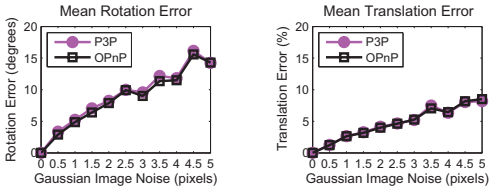


Figure 4. The mean rotation error (left) and the mean translation error (right) w.r.t. varying noise levels in the minimal n=3 case.

the image projections. At each n, 500 independent test sets are generated. We present the average rotation and translation error in the 1st and 2nd column of Fig.2.

Then, we fix the point number n to be 10 and vary the noise deviation level δ from 0.5 to 5 pixels. At each noise level, we run 500 independent trials and report the average rotation and translation error in the 3rd and 4th column of Fig.2. More results with the median errors are shown in the supplementary material.

From Fig.2, we can observe that the two representative multi-stage methods **EPnP+GN** (**EPnP** in the planar case) and **RPnP** are not accurate enough, even in the presence of redundant correspondences (*e.g.*, n=10) and moderate noise. The major reason lies in their underlying approximation schemes. However, this does not necessarily mean that any direct minimization method is accurate. For example, due to possible local optimum, the local optimization based methods **LHM** and **SP+LHM** tend to be inaccurate, especially when n is small. The convex relaxation in **SDP** might not be tight. As expected, **DLS** is inaccurate on the average, due to the singularities of the Cayley parameterization. Although the remedy in **DLS+++** is effective to avoid degeneracy, its computational time would be tripled.

Some existing methods are sensitive to the point configuration. For example, **EPnP+GN** and **LHM** are much less accurate in the quasi-singular case than in the ordinary-3D case, while **SDP** and **EPnP** deteriorate drastically in the planar case.

As shown in Fig.2, **OPnP** has definitely better accuracy than the examined state-of-the-art methods (except **DLS+++**), irrespective of the point configuration and the point number. Being an algebraic error based method, it is even comparable with the reprojection error based method **OPnP+LM**. This is understandable. As pointed out by Hartley [7], minimizing a reasonably defined algebraic error might provide accurate results, as long as all constraints are exactly enforced and data are properly normalized. Our **OPnP** solution exactly handles the challenging rotation constraint and uses centralized 3D points and image projections (Eq.(8) and Eq.(11)), which serves as data normalization in a certain sense.

4.1.2 Computational Time

The complexity of solving the polynomial system in Eq.(13) is $O(1)$. Due to the matrix multiplication of M^TM in Eq.(11), the overall complexity of **OPnP** is $O(n)$. We are interested in how it compares with existing $O(n)$ solutions, including **EPnP+GN**, **RPnP**, **DLS** and **SDP**.

We vary n from 4 to 2004. For each n, we conduct 500 independent tests and report the average running time in milliseconds (ms) in Fig.3. We can observe that **DLS** and **SDP** are not fast enough for real-time applications even with moderate n. In contrast, our **OPnP** takes about 20 ms even in the presence of thousands of points. This is due primarily to the vectorization implementation in **OPnP**, which is not so straightforward in the process of implementating **DLS**. Although **EPnP+GN** and **RPnP** are faster when n is not extremely large (*e.g.*, $n \leq 1000$), our **OPnP** solution is still very competitive, especially considering its high accuracy and general applicability.

4.1.3 Comparison with the P3P Solver

As a general solution, **OPnP** is not designed to defeat existing minimal P3P solutions, like [12]. However, if one happens to encounter a P3P problem, **OPnP** is still applicable. Here, we compare **OPnP** with the prominent P3P solution in [12]. We vary the noise level from 0 to 5 pixels and show the mean rotation and translation error over 500 independent trials in Fig.4, from which we can see that **OPnP** has the same accuracy as that of [12].

4.2. Real Images

We have also tested our **OPnP** solution on real images. We first establish tentative correspondences by matching SIFT points between the input image and the reference image. After removing outliers by RANSAC, we estimate the camera pose and augment the input image by using the projection of the model contour. As shown in Fig.5, our **OPnP** solution offers visually pleasing results.

5. Conclusion

We have revisited the PnP problem and proposed the first non-iterative $O(n)$ solution that is fast, general and globally optimal. Our contribution is to parameterize the rotation by

Figure 5. Experiment results using a 3D box (1st row) and a planar book cover (2nd row). In each row, the first image is the reference, for which we have constructed a model. The remaining four images are the input ones, which have been augmented with the projected contour by using the pose from our **OP*n*P** solution.

using non-unit quaternion and formulate the P*n*P problem into an unconstrained optimization problem. Surprisingly, the polynomial system arising from its first-order optimality condition is of odd-degree, thus assuming two-fold symmetry, a nice property that can be utilized to improve speed and numerical stability of a Gröbner basis solver. As verified by experiment results, even with a few point correspondences, our proposed solution is quite accurate, irrespective of the point configuration and the camera pose. In addition, the problem scale would not pose any difficulty, since its computational time keeps almost unchanged as the point number increases (up to a few thousands).

Acknowledgement. This work was partially supported by the Grants-in-Aid for Scientific Research (no. 25240025) from the Japan Society for the Promotion of Science, and the strategic research projects ELLIIT and eSSENCE, Swedish Foundation for Strategic Research projects ENGROSS and VINST (no. RIT08-0043).

References

[1] A. Ansar and K. Daniilidis. Linear pose estimation from points or lines. *IEEE TPAMI*, 25(5):578–589, 2003.

[2] E. Ask, Y. Kuang, and K. Astrom. Exploiting p-fold symmetries for faster polynomial equation solving. *Proc. ICPR*, pages 3232–3235, 2012.

[3] D. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry*. Springer-Verlag, 2nd edition, 2005.

[4] M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. ACM*, 24(6):381–395, 1981.

[5] X. Gao, X. Hou, J. Tang, and H. Cheng. Complete solution classification for the perspective-three-point problem. *IEEE TPAMI*, 25(8):930–943, 2003.

[6] V. Garro, F. Crosilla, and A. Fusiello. Solving the PnP problem with anisotropic orthogonal procrustes analysis. *Proc. 3DIM/3DPVT*, pages 262–269, 2012.

[7] R. Hartley. Minimizing algebraic error in geometric estimation problems. *Proc. ICCV*, pages 469–476, 1998.

[8] R. Hartley and F. Kahl. Optimal algorithms in multiview geometry. *Proc. ACCV*, pages 13–34, 2007.

[9] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge Univ. Press, 2nd edition, 2003.

[10] J. Hesch and S. Roumeliotis. A direct least-squares (DLS) method for PnP. *Proc. ICCV*, pages 383–390, 2011.

[11] Z. Hu and F. Wu. A note on the number of solutions of the non-coplanar P4P problem. *IEEE TPAMI*, 24(4):550–555, 2002.

[12] L. Kneip, D. Scaramuzza, and R. Siegwart. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. *Proc. CVPR*, pages 2969–2976, 2011.

[13] Z. Kukelova, M. Bujnak, and T. Pajdla. Automatic generator of minimal problem solvers. *Proc. ECCV*, pages 302–315, 2008.

[14] V. Lepetit, F. Moreno-Noguer, and P. Fua. Epnp: An accurate O(n) solution to the PnP problem. *IJCV*, 81(2):155–166, 2008.

[15] S. Li, C. Xu, and M. Xie. A robust O(n) solution to the perspective-n-point problem. *IEEE TPAMI*, 34(7):1444–1450, 2012.

[16] C. Lu, G. Hager, and E. Mjolsness. Fast and globally convergent pose estimation from video images. *IEEE TPAMI*, 22(6):610–622, 2000.

[17] C. Olsson, F. Kahl, and M. Oskarsson. Branch-and-Bound methods for Euclidean registration problems. *IEEE TPAMI*, 31(5):783–794, 2009.

[18] L. Quan and Z. Lan. Linear n-point camera pose determination. *IEEE TPAMI*, 21(8):774–780, 1999.

[19] G. Schweighofer and A. Pinz. Robust pose estimation from a planar target. *IEEE TPAMI*, 28(12):2024–2030, 2006.

[20] G. Schweighofer and A. Pinz. Globally optimal O(n) solution to the PnP problem for general camera models. *Proc. BMVC*, pages 1–10, 2008.

[21] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE TPAMI*, 13(4):376–380, 1991.