ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/274463118

ASPnP: An Accurate and Scalable Solution to the Perspective-n-Point Problem

Article *in* IEICE Transactions on Information and Systems · July 2013

DOI: 10.1587/transinf.E96.D.1525

CITATIONS
10

READS
224

3 authors, including:

Shigeki Sugimoto
Tokyo Institute of Technology
31 PUBLICATIONS **263** CITATIONS

SEE PROFILE

All content following this page was uploaded by Shigeki Sugimoto on 08 March 2016.

The user has requested enhancement of the downloaded file.

PAPER

ASP*n*P: An Accurate and Scalable Solution to the Perspective-*n*-Point Problem

Yinqiang ZHENG[†a)], ***Nonmember***, **Shigeki SUGIMOTO**[†b)], ***and*** **Masatoshi OKUTOMI**[†c)], ***Members***

SUMMARY We propose an accurate and scalable solution to the perspective-*n*-point problem, referred to as ASP*n*P. Our main idea is to estimate the orientation and position parameters by directly minimizing a properly defined algebraic error. By using a novel quaternion representation of the rotation, our solution is immune to any parametrization degeneracy. To obtain the global optimum, we use the Gröbner basis technique to solve the polynomial system derived from the first-order optimality condition. The main advantages of our proposed solution lie in accuracy and scalability. Extensive experiment results, with both synthetic and real data, demonstrate that our proposed solution has better accuracy than the state-of-the-art noniterative solutions. More importantly, by exploiting vectorization operations, the computational cost of our ASP*n*P solution is almost constant, independent of the number of point correspondences *n* in the wide range from 4 to 1000. In our experiment settings, the ASP*n*P solution takes about 4 milliseconds, thus best suited for real-time applications with a drastically varying number of 3D-to-2D point correspondences*.

key words: *pose estimation, perspective-n-point, Gröbner basis, global optimum*

1. Introduction

The perspective-*n*-point problem (P*n*P) aims to estimate the orientation and position, or rotation and translation, of a calibrated perspective camera by using *n* known 3D reference points and their corresponding 2D image projections. It has widespread applications in robot localization [1], hand-eye calibration [2], augmented reality [3] and so on. Considering these various application scenarios, a desirable P*n*P solution should be accurate, efficient and generally applicable, i.e., capable of handling both planar and non-planar cases with either a few or even hundreds of 3D-to-2D correspondences.

1.1 Related Works

As the minimal case, P3P ($n = 3$) has been thoroughly investigated in the literature [4], [5]. However, it has at most four possible solutions, and this multiplicity makes a P3P solver very sensitive to noise. In practice, it is usually used together with some robust estimation methods, like RANSAC [6], to remove outliers.

To improve robustness to noise, cases of four and more than four correspondences should be considered. There are some specialized algorithms [7] restricted to the slightly redundant $n = 4$ (P4P) or $n = 5$ (P5P) cases only. However, the application of these specialized P4P and P5P solutions is limited, since the number of point correspondences might differ from one frame to the other, even in a specific application scenario. As a result, a large body of existing works have tried to improve flexibility, thus applicable to the general $n \geq 4$ cases. Quan and Lan [8] proposed a linear solution by combining all the constraints from three-point subsets, whose computational complexity is $O(n^5)$. Ansar and Daniilidis [9] developed linear solutions, but with complexity $O(n^8)$, for general P*n*P with *n* points and *n* lines. Due to higher data redundancy, these methods generally work better as the number of points increases. Unfortunately, considering their high computational complexities, they become quite inefficient when the number of points is large, for example, $n \geq 100$.

The excellent work by Lepetit et al. [10] proposed to use four virtual control points to represent the 3D reference points, and successfully reduced the computational complexity to $O(n)$, which is much more appropriate for P*n*P with large *n*. Taking advantage of data redundancy, their proposed method, named as EP*n*P in short, usually offers satisfactory solutions for redundant cases with $n \geq 6$ points. As pointed out by Li et al. [11], however, its accuracy is low for slightly redundant cases with $n = 4$ or $n = 5$ points, due to ignoring some nonlinear constraints in linearization.

To improve estimation accuracy, Li et al. [11] proposed a noniterative $O(n)$ solution, named RP*n*P, which estimates the rotation axis and angle separately and removes only one nonlinear constraint. Among all existing noniterative solutions, their proposed method is the fastest and most accurate one. However, we should note that all those methods [8]–[11] mentioned above estimate the parameters in an indirect way, that is, to estimate the depth factors first and then transform the P*n*P problem into the well-known 3D-3D relative pose problem [12]. Some nonlinear constraints are also ignored so as to develop a linear solution, which surely harms accuracy. This is especially true for weakly redundant $n = 4$ and $n = 5$ cases, where the accuracy loss can hardly be compensated by data redundancy.

Ideally, the highest accuracy can be achieved by directly minimizing a properly defined error function, either

Manuscript received August 29, 2012.
Manuscript revised January 25, 2013.

[†]The authors are with the Department of Mechanical and Control Engineering, Tokyo Institute of Technology, Tokyo, 152–8550 Japan.
a) E-mail: zheng@ok.ctrl.titech.ac.jp
b) E-mail: shige@ok.ctrl.titech.ac.jp
c) E-mail: mxo@ctrl.titech.ac.jp

DOI: 10.1587/transinf.E96.D.1525

*Our MATLAB source code for ASP*n*P and the scripts to reproduce all figures in the experiment section are publicly available at https://sites.google.com/site/yinqiangzheng/

Copyright © 2013 The Institute of Electronics, Information and Communication Engineers

in the image space or the object space, while accounting for all nonlinear constraints. Olsson et al. [13] proposed to minimize the reprojection error in the image space by using branch-and-bound iterations. Their method retrieves the global optimum, irrespective of initialization. In addition, under the assumption of independently and identically distributed (i.i.d.) Gaussian noise, minimizing the reprojection error is also statistically optimal in the sense of maximum likelihood estimation (MLE). Despite these advantages, it can rarely be used in practice due to its tremendous computational cost.

As a trade off, most direct minimization based methods [14], [15] chose instead certain algebraic error functions and local optimization methods. Lu et al. [14] developed an orthogonal iteration method to directly minimize the object space error. It is one of the most widely used iterative minimization methods. However, these local optimization methods suffer from the risk of getting trapped into local minimum, and provide poor results when they indeed do so.

Some recent works tried to avoid local minimum, by using convex optimization techniques, like sum of square (SOS) relaxation [16] and semidefinite program (SDP) relaxation [17]. Their computational complexities are $O(n)$, and the size of the SDP relaxation problems is constant. Although the relaxed SDP problem can be solved in polynomial time, these methods usually take more than 100 milliseconds, making them inappropriate for real-time applications.

Quite recently, Hesch and Roumeliotis [18] developed a direct least square (DLS) method with complexity $O(n)$, in which the polynomial system derived from the first-order optimal condition of the objective function is directly solved by using matrix resultant technique. Unfortunately, they parameterized the rotation matrix by using Cayley parametrization, which degenerates in all cases of 180 degree rotations around the x-, y- and z-axis[†]. The accuracy deteriorates significantly when the camera pose approaches these singular cases. Note that these singular cases are introduced by the parametrization method, not due to the PnP problem itself.

In addition to the number of point correspondences, the configuration of the 3D reference points may also affect the estimation accuracy. Given $3 \times n$ matrix U, with one column denoting one 3D reference point, Li et al. [11] categorized the configurations into the ordinary-3D, quasi-singular and planar case. Specifically, when the rank of UU^T is 3, i.e., $rank(UU^T) = 3$, and the condition number of UU^T is moderate, the 3D point configuration is grouped as the ordinary-3D case. When $rank(UU^T) = 3$ but with a huge condition number, it is regarded as the quasi-singular case. When $rank(UU^T) = 2$, it corresponds to the well-known planar case. There are some methods [19] dedicated to the planar case, while most of the recently proposed methods [10], [11], [18] can handle planar and nonplanar cases simultaneously. However, Li et al. [11] pointed out that some existing methods, like the iterative solution by Lu et al. [14] and the noniterative EPnP solution, work poorly for the quasi-singular case, while the method [19] dedicated to the planar case is not applicable to the quasi-singular one at all.

1.2 Overview of Our Work

In this work, we propose an accurate and scalable solution to the PnP problem, referred to as ASPnP. Similar to DLS [18], we directly minimize a properly designed error function, leading to a nonlinear least square problem with respect to the rotation parameters only. In contrast to the Cayley parametrization, we use the quaternion parametrization, and properly avoid all possibilities of degeneracy caused by parametrization. To find the global optimum, we solve the polynomial system derived from the first-order optimal condition of the objective function by using standard Gröbner basis technique. Compared with the state-of-the-art indirect and noniterative methods, like RPnP [11] and EPnP [10], our direct minimization method has better accuracy for all 3D reference point configurations, including the challenging planar and quasi-singular cases. Without suffering from degeneracy in parametrization, our solution is more stable than the direct DLS method. Theoretically speaking, the computational complexity of our proposed ASPnP solution is $O(n)$, which is the same as that of the RPnP [11], EPnP [10] and DLS method [18]. However, by taking advantage of the problem structure, we can easily vectorize the ASPnP implementation. As a result, our ASPnP solution is quite efficient and scalable, taking almost constant computational time (about 4 milliseconds in our experiment settings) for the wide spectrum of point correspondences from $n = 4$ to $n = 1000$. In contrast, the computational cost of existing methods increases drastically, as the problem scale expands.

The organization of the remaining parts is as follows. In Sect. 2, we present some preliminaries of the PnP problem. Section 3 covers the formulation of our method as well as the details of solving polynomial systems. The computational complexity and vectorization techniques are shown in Sect. 4. Section 5 includes the experiment results using both synthetic and real data, and Sect. 6 gives the concluding remarks.

2. Preliminaries

2.1 Notations

Throughout this work, we denote matrices, vectors and scalars by using capital letters, bold lowercase letters and plain lowercase letters, respectively, with the exception that T is reserved for matrix or vector transpose. All vectors are column-wise.

[†]When this paper was under preparation, Hesch and Roumeliotis provided a remedy to conquer this problem by solving DLS three times under different rotated 3D points. Since the computational time would be tripled, we believe this remedy is not attractive, especially considering that DLS itself is not very fast.

2.2 The P*n*P Problem

It is assumed that we know 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 $\mathbf{p}_i = \begin{bmatrix} u_i & v_i & 1 \end{bmatrix}^T, i = 1, 2, \cdots, n$, in the homogeneous image coordinate framework. The perspective camera is assumed to follow the pinhole imaging model, and its intrinsic parameter matrix K has been calibrated. The objective is to retrieve the rotation matrix R and the translation vector $\mathbf{t}$, accounting for the camera orientation and position, respectively, such that the projective imaging equation

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

could be satisfied, in which λ_i denotes the depth factor of the i-th point.

Considering that K is known, it is convenient to use the normalized image coordinate $\tilde{\mathbf{p}}_i = \begin{bmatrix} \tilde{u}_i & \tilde{v}_i & 1 \end{bmatrix}^T = K^{-1}\mathbf{p}_i, i = 1, 2, \cdots, n$.

3. The Proposed ASP*n*P Solution

In this section, we shall present our proposed solution to the P*n*P problem.

3.1 Eliminating the Depth Factors

Since the depth factor satisfies that

$$\lambda_i = \mathbf{r}_3^T \mathbf{q}_i + t_3, \tag{2}$$

it is desirable to eliminate all depth factors as follows

$$\begin{aligned} (\mathbf{r}_3^T \mathbf{q}_i + t_3)\tilde{u}_i &= \mathbf{r}_1^T \mathbf{q}_i + t_1, \\ (\mathbf{r}_3^T \mathbf{q}_i + t_3)\tilde{v}_i &= \mathbf{r}_2^T \mathbf{q}_i + t_2, \end{aligned} \tag{3}$$

in which $\mathbf{r}_1^T$, $\mathbf{r}_2^T$, $\mathbf{r}_3^T$ are the 1st, 2nd and 3rd row of R, respectively, and $\mathbf{t} = \begin{bmatrix} t_1 & t_2 & t_3 \end{bmatrix}^T$.

Let us point out that the DLS method [18] eliminates the depth factors by solving a large linear system, which increases the complexity unnecessarily.

3.2 Rotation Parametrization

It is well known that three independent parameters are enough to represent the rotation matrix R, due to the orthonormal constraint $RR^T = I$ and the determinant constraint $det(R) = 1$.

There are various parametrization methods for R, such as the Euler angle, rotation axis-angle, Cayley and quaternion parametrization. In contrast to the Cayley parametrization [18], we choose the quaternion parametrization in its fractional form, which is free of any trigonometric function and any possibility of degeneracy. Specifically, letting $s = a^2 + b^2 + c^2 + d^2$, the quaternion parametrization in its fractional form reads

$$R = \frac{1}{s}\begin{bmatrix} a^2+b^2-c^2-d^2 & 2bc-2ad & 2bd+2ac \\ 2bc+2ad & a^2-b^2+c^2-d^2 & 2cd-2ab \\ 2bd-2ac & 2cd+2ab & a^2-b^2-c^2+d^2 \end{bmatrix}, \tag{4}$$

in which a, b, c, d are the four unknown parameters, and $s \neq 0$, i.e., a, b, c, d are not simultaneously zero. It is straightforward to verify that the parametrization in Eq. (4) satisfies $RR^T = I$ and $det(R) = 1$.

Actually, the fractional form in Eq. (4) is quite uncommon in computer vision literature. It is more conventional to enforce the unit norm constraint $a^2+b^2+c^2+d^2 = 1$ so as to eliminate the fractional term in Eq. (4). Here, we prefer the fractional form, since Eq. (3) is homogeneous. Closer inspection of Eq. (4) leads us to the important observation that a, b, c, d in Eq. (4) assume scale ambiguity and sign ambiguity, i.e., $R(a, b, c, d) = R(ka, kb, kc, kd)$, for any nonzero k. Considering that there are only three independent parameters in R, we explore the following four cases:

◇ Case-1. When $a \neq 0$, without loss of generality, we can rescale a, b, c, d implicitly, such that $a = 1$. Then, there remain only three parameters b, c, d in R

$$R = \frac{1}{s_1}\begin{bmatrix} 1+b^2-c^2-d^2 & 2bc-2d & 2bd+2c \\ 2bc+2d & 1-b^2+c^2-d^2 & 2cd-2b \\ 2bd-2c & 2cd+2b & 1-b^2-c^2+d^2 \end{bmatrix}, \tag{5}$$

in which $s_1 = 1 + b^2 + c^2 + d^2$.

Through some basic calculations, one can verify that Eq. (5) is equivalent to the Cayley parametrization used in DLS [18]. Therefore, it is straightforward to see that the Cayley parametrization is always degenerate when a is zero.

◇ Case-2. When $a = 0$, $b \neq 0$, by letting $b = 1$, it is possible to reduce the parametrization in Eq. (4) into

$$R = \frac{1}{1+c^2+d^2}\begin{bmatrix} 1-c^2-d^2 & 2c & 2d \\ 2c & -1+c^2-d^2 & 2cd \\ 2d & 2cd & -1-c^2+d^2 \end{bmatrix}. \tag{6}$$

◇ Case-3. When $a = 0$, $b = 0$, $c \neq 0$, we can let $c = 1$, and simplify Eq. (4) into

$$R = \frac{1}{1+d^2}\begin{bmatrix} -1-d^2 & 0 & 0 \\ 0 & 1-d^2 & 2d \\ 0 & 2d & -1+d^2 \end{bmatrix}. \tag{7}$$

◇ Case-4. When $a = 0$, $b = 0$, $c = 0$, $d \neq 0$, we can let $d = 1$, and now Eq. (4) becomes

$$R = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{8}$$

It is trivial to recognize that these four cases are independent, and cover all possible rotation matrix R. However, for a given P*n*P problem, before solving all these four cases,

there is no way of determining which case is the correct one. In the following, we show how to build the objective function and its first-order optimality condition for all these four cases. We are going to focus primarily on Case-1, since the remaining three cases can be handled in a similar way.

3.3 Objective Function and Minimization

As for Case-1, let us note that the fractional term in Eq. (5) would not cause any inconvenience, because of the homogeneous structure in Eq. (3). After rescaling $\mathbf{t}$ such that $\tilde{\mathbf{t}} = \begin{bmatrix}\tilde{t}_1 & \tilde{t}_2 & \tilde{t}_3\end{bmatrix}^T = s_1 \begin{bmatrix}t_1 & t_2 & t_3\end{bmatrix}^T$, we can eliminate the fractional term $\frac{1}{s_1}$ by multiplying s_1 at both sides of Eq. (3). Specifically, we denote that $\mathbf{v} = \begin{bmatrix}1 & b & c & d & b^2 & bc & bd & c^2 & cd & d^2\end{bmatrix}^T$, and transform Eq. (3) into the following matrix form

$$M_i\tilde{\mathbf{t}} = N_i\mathbf{v}, i = 1, 2, \cdots, n, \tag{9}$$

in which

$$M_i = \begin{bmatrix}-1 & 0 & \tilde{u}_i \\ 0 & -1 & \tilde{v}_i\end{bmatrix}, \tag{10}$$

and

$$N_i = \begin{bmatrix}\mathbf{n}_i^1 & \mathbf{n}_i^2\end{bmatrix}^T, \tag{11}$$

$$\begin{aligned}\mathbf{n}_i^1 &= \tilde{u}_i \begin{bmatrix}-z_i & -2y_i & 2x_i & 0 & z_i & 0 & -2x_i & z_i & -2y_i & -z_i\end{bmatrix}^T \\ &\quad + \begin{bmatrix}x_i & 0 & 2z_i & -2y_i & x_i & 2y_i & 2z_i & -x_i & 0 & -x_i\end{bmatrix}^T, \\ \mathbf{n}_i^2 &= \tilde{v}_i \begin{bmatrix}-z_i & -2y_i & 2x_i & 0 & z_i & 0 & -2x_i & z_i & -2y_i & -z_i\end{bmatrix}^T \\ &\quad + \begin{bmatrix}y_i & -2z_i & 0 & 2x_i & -y_i & 2x_i & 0 & y_i & 2z_i & -y_i\end{bmatrix}^T.\end{aligned} \tag{12}$$

Due to some measurement noise, the equality in Eq. (9) generally could not be perfectly satisfied. Therefore, we directly minimize the squared error of Eq. (9), and reach the following nonlinear least square problem

$$\min_{b,c,d,\tilde{\mathbf{t}}} \|M\tilde{\mathbf{t}} - N\mathbf{v}\|^2, \tag{13}$$

in which $M = \begin{bmatrix}M_1^T & M_2^T & \cdots & M_n^T\end{bmatrix}^T$, and $N = \begin{bmatrix}N_1^T & N_2^T & \cdots & N_n^T\end{bmatrix}^T$.

Let us note that the scaled translation $\tilde{\mathbf{t}}$ appears linearly in the objective function, and all nonlinear terms in $\mathbf{v}$ only. Therefore, we can project out $\tilde{\mathbf{t}}$ without changing the optimal solutions.

Given $\mathbf{v}$, we can easily solve $\tilde{\mathbf{t}}$ by using linear least square

$$\tilde{\mathbf{t}} = M^+N\mathbf{v}, \tag{14}$$

in which M^+ denotes the Moore-Penrose matrix inverse.

Plugging Eq. (14) back into Eq. (13), we obtain the simplified objective function

$$\begin{aligned}f(b, c, d) &= \|(MM^+ - I)N\mathbf{v}\|^2 \\ &= -\mathbf{v}^TN^T(MM^+ - I)N\mathbf{v} \\ &= -\mathbf{v}^TG\mathbf{v},\end{aligned} \tag{15}$$

in which I is the $2n\times2n$ identity matrix, $(MM^+-I)^T(MM^+-I) = -(MM^+ - I)$, and $G = N^T(MM^+ - I)N$.

Interestingly, the objective function in Eq. (15) has only three unknown variables, irrespective of the number of point correspondences n. This fact enables us to solve large scale problems with even hundreds of reference points.

To minimize f with respect to b, c, d, one can surely use some standard local optimization methods for nonlinear least square problems, like Newton and Gauss-Newton methods, which might get trapped into undesirable local optimum. In contrast, we are interested in developing a noniterative method and retrieving the global optimal solution.

It is straightforward to build the first-order optimality condition of Eq. (15), and identify all the stationary points. By calculating the derivative of f with respect to b, c, d, we get the first-order optimality condition composed of three 3-order polynomials

$$\begin{aligned}\frac{\partial f}{\partial b} &= -\mathbf{v}^TG\frac{\partial \mathbf{v}}{\partial b} = 0, \\ \frac{\partial f}{\partial c} &= -\mathbf{v}^TG\frac{\partial \mathbf{v}}{\partial c} = 0, \\ \frac{\partial f}{\partial d} &= -\mathbf{v}^TG\frac{\partial \mathbf{v}}{\partial d} = 0.\end{aligned} \tag{16}$$

To solve the above multivariate polynomial system, we adopt the standard Gröbner basis technique, the details of which shall be addressed in the following section.

Until now, we have shown how to build the objective function and its first-order optimality condition for Case-1. As for Case-2, we can apply the same technique to eliminate the fractional term in Eq. (6). Since there are only two variables c and d in R, the first-order optimality condition is composed of two 3-order polynomials with respect to c and d. We are going to adopt the Gröbner basis technique to solve this polynomial system as well.

As for Case-3, after eliminating the fractional term in Eq. (7), the objective function is much simpler, since there is only one variable d. In this case, the first-order optimality condition is a cubic univariate polynomial with respect to d only. It is well-known that there exist closed form solutions for a cubic univariate polynomial. As an alternative, one can also calculate the eigenvalue of the companion matrix corresponding to the cubic polynomial.

Resolving Case-4 is trivial, since R is constant. We can directly solve the translation vector by using linear least square, which is similar to Eq. (14).

3.4 Solving Multivariate Polynomial Systems

Solving multivariate polynomial systems has been system-

atically investigated in many minimal problems in computer vision, based primarily on the Gröbner basis technique [20]. Most noticeably, Kukelova et al. [21] introduced a novel automatic generator of polynomial system solvers, which is also applicable to the multivariate polynomial systems arising from Case-1 and Case-2.

Briefly speaking, the automatic generator first identifies the Gröbner bases in graded reverse lexicographical ordering that correspond to the original polynomial system, and then constructs the elimination template by choosing proper polynomials in the *Ideal* generated by multiplying proper monomials. Finally, the action matrix is built, the eigen factorization of which provides the solutions of the original system. In this sense, the action matrix is very similar to the companion matrix corresponding to an univariate polynomial.

As for the three-variable 3-order polynomial system arising from Case-1, there are at most 27 solutions, thus the size of the action matrix is 27×27. The elimination template matrix is of size 89×116. As for the two-variable 3-order polynomial system corresponding to Case-2, the elimination template matrix is of size 12×21, and the action matrix is of size 9×9. The eigen factorization of this action matrix gives all the possible 9 solutions of the original two-variable 3-order polynomial system. The reader can refer to our source code for the details on the monomials in the Gröbner bases and the construction process of the elimination templates, which are a little bit too verbose to present here.

In both cases, the size of the elimination matrix and that of the action matrix are independent of the number of point correspondences n, the computational cost of solving the multivariate polynomial system is therefore constant. In addition, the eigen factorization of a small matrix, e.g. 27×27 matrix for Case-1 and 9×9 matrix for Case-2, is computationally fast, which makes the development of an efficient P*n*P solution possible.

3.5 Identifying the Best Solution

After solving the multivariate polynomial systems for Case-1 and Case-2, the univariate cubic polynomial for Case-3 and the linear least square problem for Case-4, we can obtain all the real stationary points of the P*n*P problem. By using the chirality condition that all 3D reference points are in front of the camera, we can further filter out those physically infeasible solutions. Finally, we evaluate the reprojection error for each feasible solution, and choose the one with the smallest reprojection error as the optimal solution.

4. Computational Complexity and Vectorization

4.1 Computational Complexity

As shown in the previous section, there are two critical steps in our ASP*n*P solution, i.e., the construction step to build the polynomial system and the polynomial system solving step. The computational cost for the latter is constant, irrespective of the number of point correspondences n. In contrast, the construction step depends indeed on the problem scale.

Let's first consider the complexity of the construction step for Case-1. In Eq. (15), we need to construct $G = N^T(MM^+ - I)N = (N^TM)(M^TM)^{-1}(N^TM)^T - N^TN$. Considering that the size of M^TM is 3×3, the cost of inverting M^TM is constant and almost negligible. Then, it is straightforward to recognize that the complexity of this construction step is $O(n)$, due to the multiplication of N^TM, M^TM and N^TN, in which N is of size $2n\times10$ and M of size $2n\times3$. Similarly, the complexity of the construction step for Case-2, Case-3 and Case-4 would not exceed $O(n)$.

Generally speaking, the computational cost of an $O(n)$ algorithm increases moderately as the problem scale expands. However, in some application scenarios, such as realtime tracking of a well-textured object, it is possible to encounter a P*n*P problem with hundreds of 3D-to-2D correspondences. Therefore, it is quite important to improve scalability further by exploring and taking advantage of the problem structure.

4.2 Vectorization

First, by investigating the structure of Eq. (12), it is possible to vectorize the construction of N, i.e., constructing all rows of N simultaneously, instead of every two rows in an iterative way. In addition, M is well structured, specifically,

$$M = \begin{bmatrix} -1 & 0 & \tilde{u}_1 \\ 0 & -1 & \tilde{v}_1 \\ \cdots & \cdots & \cdots \\ -1 & 0 & \tilde{u}_n \\ 0 & -1 & \tilde{v}_n \end{bmatrix}, \tag{17}$$

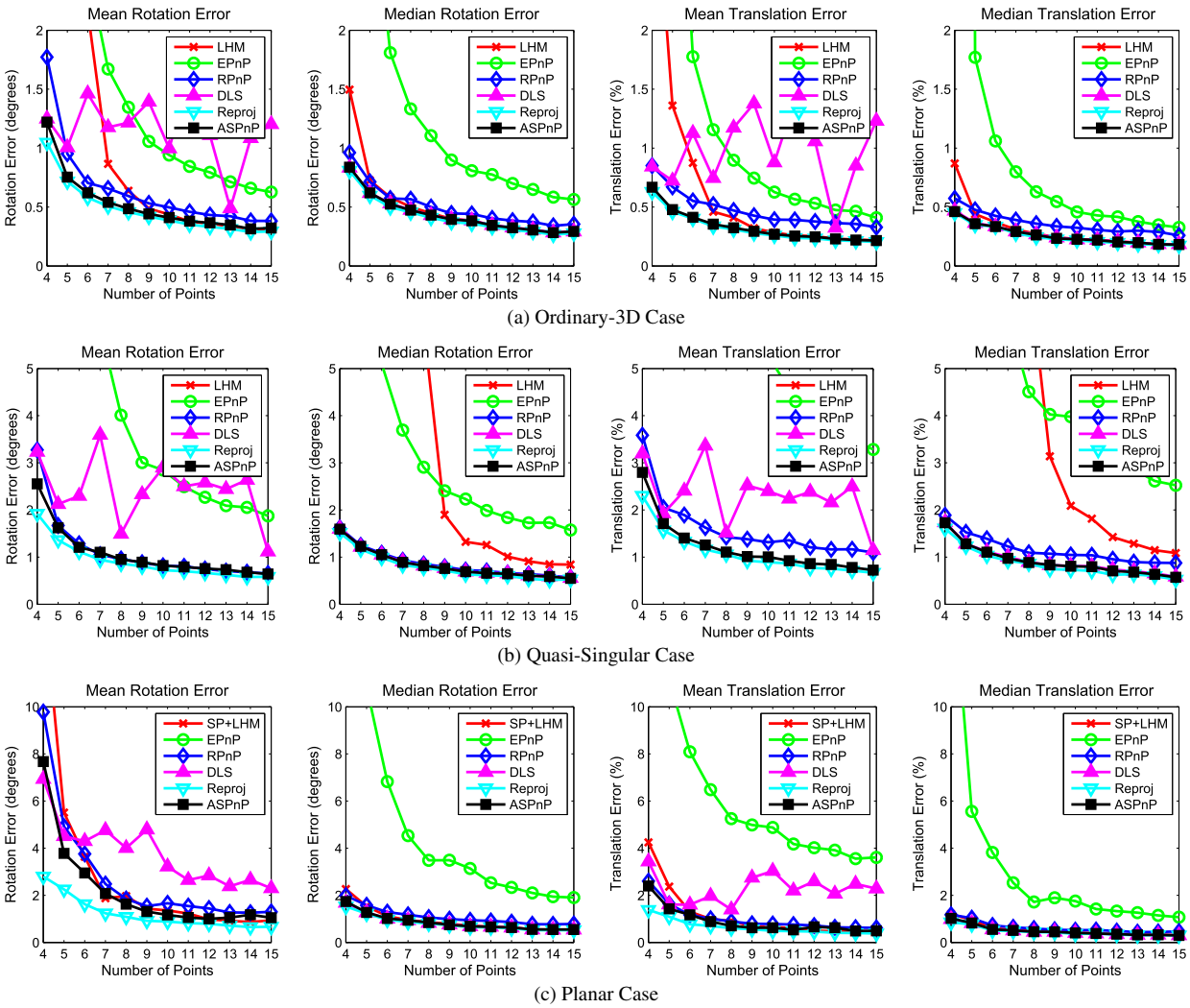
therefore, instead of using matrix multiplication blindly, M^TM is simply

$$M^TM = \begin{bmatrix} n & 0 & -\sum_{i=1}^n \tilde{u}_i \\ 0 & n & -\sum_{i=1}^n \tilde{v}_i \\ -\sum_{i=1}^n \tilde{u}_i & -\sum_{i=1}^n \tilde{v}_i & \sum_{i=1}^n (\tilde{u}_i^2 + \tilde{v}_i^2) \end{bmatrix}. \tag{18}$$

Similarly, we can compute N^TM in a vectorized way. Through these simple vectorization techniques, surprisingly, our P*n*P solver takes almost constant computational time for varying number of 3D-to-2D correspondences from $n = 4$ to $n = 1000$, a wide working range that probably covers all practical applications. It is also worthy of mentioning that existing EP*n*P and RP*n*P could not benefit from such kind of simple vectorization techniques.

5. Experimental Results

In this section, we test our accurate and scalable solution to the P*n*P problem, referred to as **ASP*n*P**, and compare it with the state-of-the-art P*n*P solutions. For the ordinary-3D case and the quasi-singular case, we consider three noniterative solutions, including **EP*n*P** [10], **RP*n*P** [11] and the direct least square solution (**DLS**) [18], as well as the popular iterative method by Lu et al. [14], denoted by **LHM**



(a) Ordinary-3D Case

(b) Quasi-Singular Case

(c) Planar Case

Fig. 1 Average and median rotation and translation errors with respect to varying number of points $n = 4, 5, \ldots, 15$, and fixed image noise level $\delta = 2$ pixels for the ordinary-3D case (a), the quasi-singular case (b) and the planar case (c).

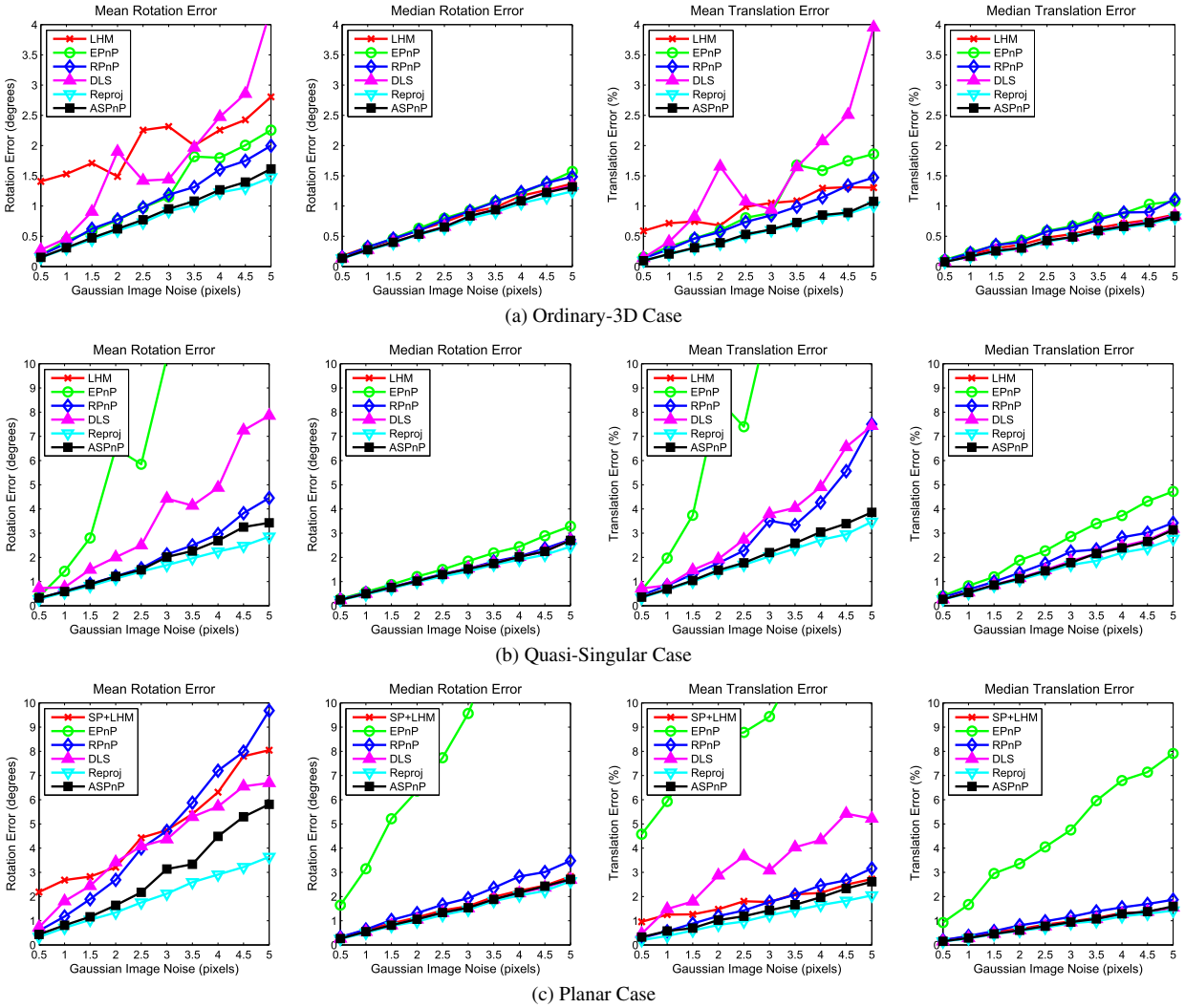
in short. For the planar case, we include into comparison the **EP*n*P**, **RP*n*P**, **DLS** solutions. In addition, the iterative method by Schweighofer and Pinz [19] dedicated to the planar case is also considered, which is denoted by **SP+LHM**, since it incorporates the iterative **LHM** method as the final minimization tool. In all synthetic experiments with ground-truth pose parameters, we also minimize the reprojection error via local optimization techniques initialized by the ground-truth. We denote it by **Reproj**.

5.1 Experiments with Synthetic Data

To make our results easily reproducible, we use similar experiment configurations as in the work of **RP*n*P** [11]. We assume a virtual perspective camera with image size 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 $[-2, 2] \times [-2, 2] \times [4, 8]$, while for the quasi-singular case, they are in the range of $[1, 2] \times [1, 2] \times [4, 8]$. 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 arc-cosine operation, respectively. The translation error is measured by the relative difference between $\mathbf{t}_{true}$ and $\mathbf{t}$ defined as $e_{trans}(\%) = \|\mathbf{t}_{true} - \mathbf{t}\|/\|\mathbf{t}\| \times 100$.

5.1.1 Varying Number of Points

We investigate the performance of competing solutions with varying number of correspondences for the ordinary-3D, quasi-singular and planar case. We vary n from 4 to 15, and add zero-mean Gaussian noise with deviation $\delta = 2$ pixels



(a) Ordinary-3D Case

(b) Quasi-Singular Case

(c) Planar Case

Fig. 2 Average and median rotation and translation errors with respect to varying image noise level $\delta = 0.5, 1, \ldots, 5$ pixels for the ordinary-3D case (a), the quasi-singular case (b) and the planar case (c). The number of point correspondences is fixed to be 6.

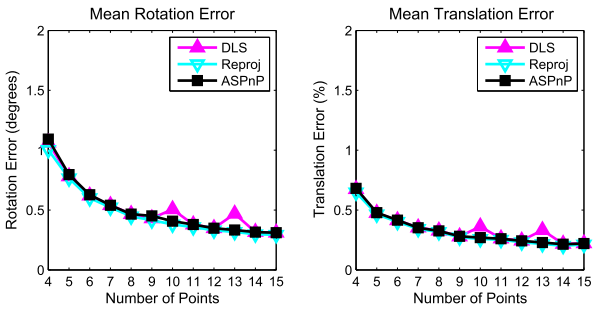
onto the image projections. At each n, 1000 independent test data sets are generated. As in Fig. 1, we present the average and the median rotation and translation error among these 1000 independent trials. It's straightforward to recognize that all solutions work better as the point number n increases, demonstrating that data redundancy actually contributes to improving accuracy.

As for the ordinary-3D case in Fig. 1 (a), when there are only $n = 4$ and $n = 5$ point correspondences, the accuracy of **EP*n*P** is very poor, due to ignoring some nonlinear constraints in the process of linearization. As an indirect method, **RP*n*P** is much better than **EP*n*P**, since it only removes one nonlinear constraint. Although taking into consideration all nonlinear constraints, the iterative **LHM** method fails to work satisfactorily, which reveals that it is highly risky of local optimum. The direct **DLS** method does not suffer from local optimum, however, it might offer poor estimation when the camera pose is close to the singularities of the Cayley parametrization. Among all the competing solutions except **Reproj**, our proposed **ASP*n*P** is the most accurate one. Even compared with **RP*n*P**, **ASP*n*P** is slightly but consistently better, especially for weakly redundant $n = 4$ and $n = 5$ cases.

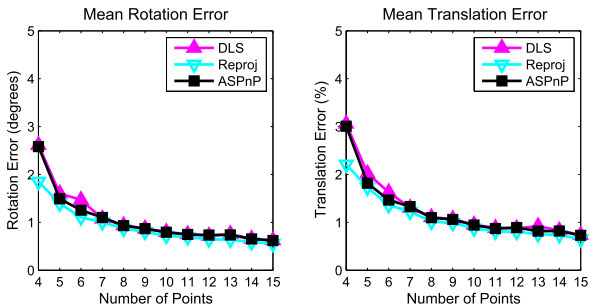
As for the quasi-singular case in Fig. 1 (b), **LHM** is the most seriously deteriorated solution, followed by **EP*n*P**. **DLS** still suffers from instability. In this case, **RP*n*P** and **ASP*n*P** are comparable in terms of the accuracy of rotation, but **ASP*n*P** is better in terms of that of translation.

Among all the solutions for the planar case in Fig. 1 (c), **EP*n*P** is the poorest. **DLS** provides some unstable results, thus leading to substantial mean rotation errors. The specialized **SP+LHM** solution works satisfactorily when $n \geq 6$. It might also converge to bad local optimal solutions, as will be shown in the experiment with real images.

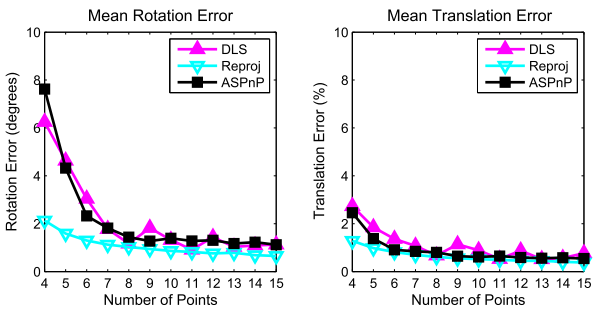
To sum up, in all sub-figures of Fig. 1, **ASP*n*P** is the closest solution to **Reproj**, thus verifying its accuracy and widespread applicability for both planar and nonplanar 3D point configurations even with a few point correspondences.



(a) Ordinary-3D Case

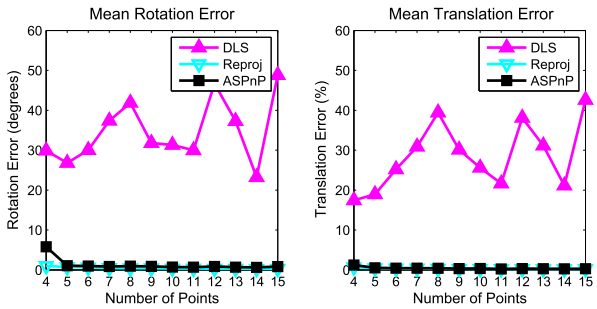


(b) Quasi-Singular Case

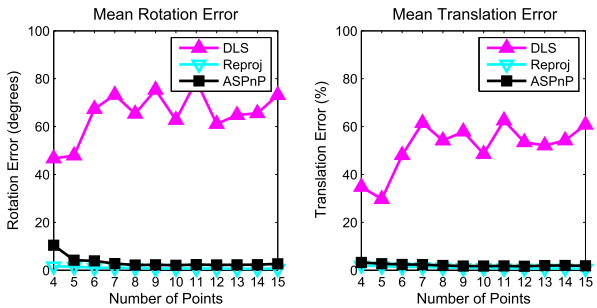


(c) Planar Case

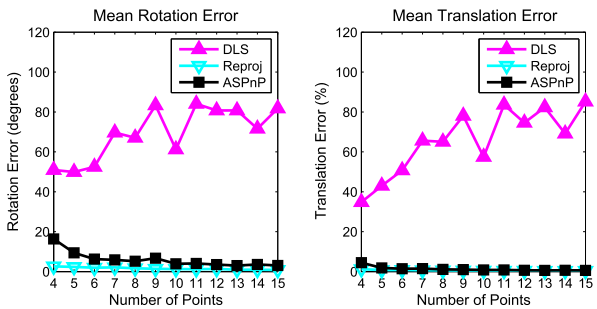
Fig. 3 Experiment results using Cayley non-degenerate random rotations.



(a) Ordinary-3D Case



(b) Quasi-Singular Case



(c) Planar Case

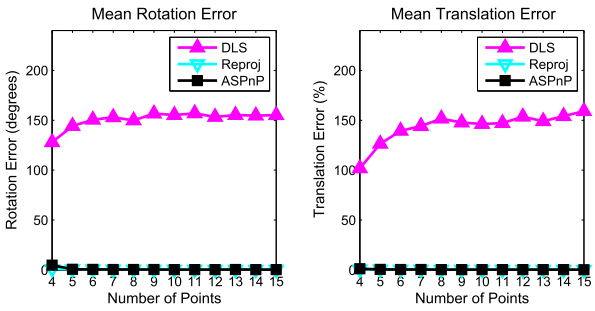
Fig. 4 Experiment results using Cayley quasi-degenerate random rotations.

5.1.2 Varying Noise Levels

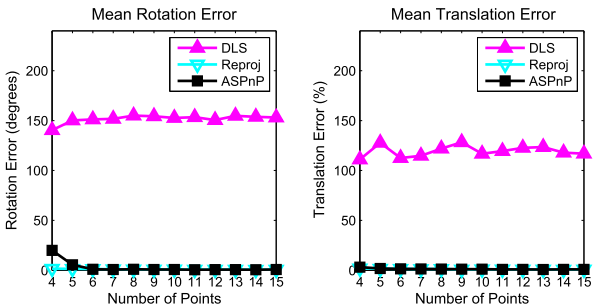
Here, we fix the number of correspondences to be 6, which is a very challenging situation, and vary the noise deviation level δ from 0.5 to 5 pixels. At each noise level, we run 1000 independent trials and report the average and the median rotation and translation error for the ordinary-3D, quasi-singular and planar case in Fig. 2 (a), Fig. 2 (b) and Fig. 2 (c), respectively. **EP*n*P** is quite inaccurate, especially in the quasi-singular and planar case. Again, we can observe that the iterative solutions, including **LHM** for the ordinary-3D and the quasi-singular case and **SP+LHM** for the planar case, suffer from poor convergence and possible local optimum. The instability of **DLS** is obvious. Both **RP*n*P** and **ASP*n*P** work properly when the noise level is low. However, **ASP*n*P** is consistently better when the noise level is higher than 2 pixels.

5.1.3 Investigating Degeneracy of Cayley Parametrization

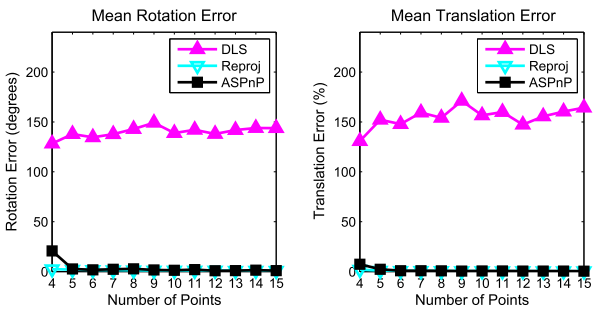
From Fig. 1 and Fig. 2, we have concluded that the overall accuracy of **DLS** is very poor. To further verify that the inaccuracy of **DLS** is caused by the degeneracy of Cayley parametrization, we repeat the aforementioned experiments by using random rotations generated under three different conditions. As mentioned in Sect. 3.2, when $a \neq 0$, we can rescale the quaternion parameter (a, b, c, d) such that $a = 1$, leading to the Cayley representation. Therefore, when $a = 0$ or $a \approx 0$, the Cayley representation is degenerate or quasi-degenerate, respectively. We thus randomly generate non-degenerate, quasi-degenerate and degenerate rotations according to $a \in [0.1, 1]$, $a \in [0, 0.1]$ and $a = 0$, respectively. Using the same experiment settings as in Sect. 5.1.1, the experiment results in the case of non-degenerate, quasi-degenerate and degenerate rotations are shown in Fig. 3, Fig. 4 and Fig. 5, respectively. For clarity, only the mean rotation and translation error curves of **DLS**, **ASP*n*P** and **Reproj** are shown. From Fig. 3, we observe that **DLS** and **ASP*n*P** are of comparable accuracy. The minor difference might be explained by the difference in their problem formulation. However, as shown in Fig. 4 and Fig. 5, the accuracy of **DLS** deteriorates drastically in the quasi-degenerate



(a) Ordinary-3D Case



(b) Quasi-Singular Case



(c) Planar Case

Fig. 5 Experiment results using Cayley degenerate rotations.

and degenerate cases, while that of **ASP*n*P** is sufficiently satisfactory.

5.1.4 Computational time

We are interested in the speed of our **ASP*n*P** solution and how it compares with existing fast noniterative solutions, including **EP*n*P**, **RP*n*P** and **DLS**. The iterative solution **LHM** is excluded from comparison, because it is known to be slow [10], [11], especially when the number of points is large. We implement our **ASP*n*P** solution in MATLAB and incorporate the vectorization techniques shown in Sect. 4.2. We use the publicly available source code of **EP*n*P**[†], **RP*n*P**[††] and the fast version of **DLS**[†††]. We run all codes in MATLAB environment on a notebook with 2.26 GHz CPU.

We vary the number of points n from 4 to 1004. For each n, we conduct 1000 independent tests and report the average running time in milliseconds (ms) for each solution. As shown in Fig. 6, the computational cost of **EP*n*P**, **RP*n*P** and **DLS** increases nontrivially as the problem scale expands. Interestingly, our **ASP*n*P** solution always takes about

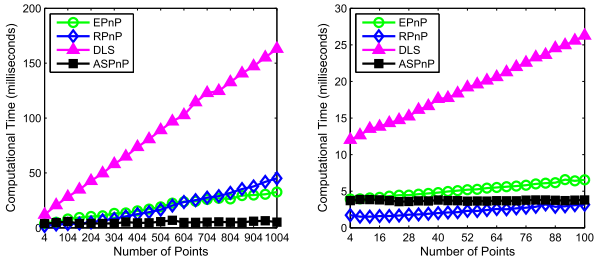


Fig. 6 Average computational time with respect to varying number of points $n = 4, 54, \ldots, 1004$ (left), and the close-up in the range of $n = 4, 8, \ldots, 100$ (right).

(a) Image Pair ♯1 with 169 inliers among 178 correspondences

(b) Image Pair ♯2 with 408 inliers among 422 correspondences

Fig. 7 Real experiments with a 3D box. For each row, the reference image is at left, while the input image right. Inliers and outliers are indicated by thin green lines and dashed red lines, respectively. The input images are overlaid with the box contours using the pose parameters from ASP*n*P.

4 ms, independent of the number of points in the whole range from $n = 4$ to $n = 1004$, thus verifying its scalability and applicability to large scale P*n*P problems. For example, when $n = 500$, our **ASP*n*P** solution is about 5 times faster than **EP*n*P** and **RP*n*P**, and about 20 times faster than **DLS**. Let's observe that when the number of points n is less than 100, **RP*n*P** is faster than **ASP*n*P**. In spite of that, we still recommend our **ASP*n*P** solution as a competitive alternative, on the basis of its superiority in accuracy for slightly redundant cases and highly noisy cases.

5.2 Experiments with Real Data

For experiments with real images, we use a calibrated perspective camera with image resolution 640×480 pixels. As shown in Fig. 7 and Fig. 8, we construct the 3D template of a box and that of a planar book cover, with 1338 and 971 SIFT [22] feature points in their corresponding refer-

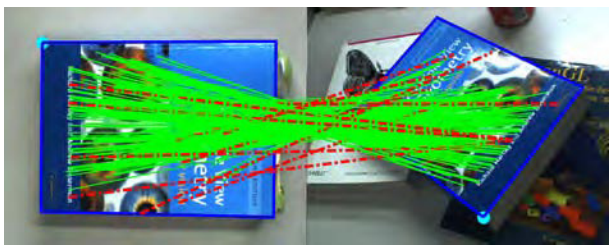
[†]http://cvlab.epfl.ch/software/EPnP/
[††]http://xuchi.weebly.com/rpnp.html
[†††]http://www-users.cs.umn.edu/~joel/index.php?page=software

Table 1 Performance statistics of competing solutions for the experiment with a box shown in Fig. 7.

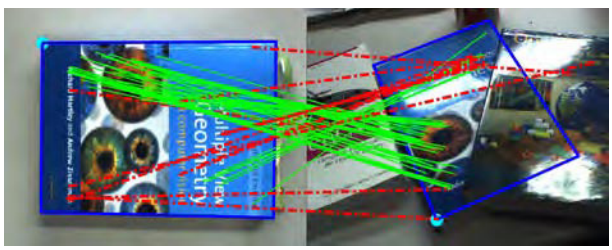
image	♯1 (n = 169)					♯2 (n = 408)				
method	EPnP	RPnP	DLS	LHM	ASPnP	EPnP	RPnP	DLS	LHM	ASPnP
e_{rot} ($deg.$)	0.849	0.190	0.108	0.101	0.108	0.315	0.095	0.027	0.227	0.026
e_{trans} (%)	0.418	0.132	0.014	0.013	0.014	0.288	0.061	0.044	0.092	0.043
time (ms)	5.2	5.3	41.3	220.9	4.6	10.2	18.1	78.6	547.4	4.7

Table 2 Performance statistics of competing solutions for the experiment with a planar book cover shown in Fig. 8.

image	♯1 (n = 213)					♯2 (n = 46)				
method	EPnP	RPnP	DLS	SP+LHM	ASPnP	EPnP	RPnP	DLS	SP+LHM	ASPnP
e_{rot} ($deg.$)	1.440	0.390	0.178	0.080	0.087	1.611	0.146	0.105	17.874	0.051
e_{trans} (%)	1.122	0.686	0.109	0.182	0.172	0.252	0.087	0.172	17.322	0.070
time (ms)	6.8	9.7	61.1	872.4	5.2	3.4	3.3	23.3	147.4	4.4



(a) Image Pair ♯1 with 213 inliers among 223 correspondences



(b) Image Pair ♯2 with 46 inliers among 58 correspondences

Fig. 8 Real experiments with a planar book cover. For each row, the reference image is at left, while the input image right. Inliers and outliers are indicated by thin green lines and dashed red lines, respectively. The input images are overlaid with the contours using the pose parameters from ASPnP.

ence images. Given an input image, we first build tentative correspondences by matching SIFT points between the reference and the input image. To remove mismatches, a P3P solution is used in combination with RANSAC [6]. All inliers are used to estimate the camera pose. Since the ground truth camera pose is not easily available, we minimize the reprojection error by using the branch-and-bound method [13] and use the estimated pose as the ground truth, with respect to which the rotation and translation error are evaluated. The branch-and-bound method takes more than two minutes in each of the experiments, thus inappropriate for real-time applications at all. The performance statistics for all competing methods are shown in Table 1 and Table 2, from which we can observe that, in all the experiments, our proposed **ASPnP** is more accurate than **EPnP** and **RPnP**. In case of proper convergence, the iterative solution **LHM** and **SP+LHM** could provide accurate results (e.g., in Fig. 7 (a) and Fig. 8 (a)), but they indeed suffer from undesirable local optimum (e.g., in Fig. 7 (b) and Fig. 8 (b)). For all the experiments in Fig. 7 and Fig. 8, **DLS** retrieves sufficiently accurate results, since the real camera pose happens to be far from the singularities of its parametrization. In spite of that, our **ASPnP** solution is much faster.

6. Conclusion

In this work, we have proposed an accurate and scalable solution to the perspective-n-point pose estimation problem. Our central idea is to estimate the orientation and position parameters by directly minimizing a properly defined algebraic error. By using Gröbner basis technique, we solve the polynomial system derived from the first-order optimality condition, and retrieve the global optimum without iterations. Experiment results have demonstrated its superiority over the state-of-the-art noniterative methods in terms of accuracy for both planar and nonplanar 3D point configurations. Our method is fast enough for real-time applications, and takes almost constant computation cost for widely varying number of point correspondences from $n = 4$ to $n = 1000$. To facilitate reproduction and further improvement, we have made the source code publicly available.

Acknowledgments

The authors would like to thank Prof. Shiqi Li and Dr. Chi Xu for allowing us to use their source code of RPnP. The authors are also grateful to the editor and the anonymous reviewers for their constructive suggestions. Yinqiang Zheng is partly supported by the Japanese Government (MEXT) Scholarship.

References

[1] S. Hijikata, K. Terabayashi, and K. Umeda, "A simple indoor self-localization system using infrared LEDs," Proc. Int'l Conf. on Networked Sensing Systems, pp.1–7, 2009.
[2] K. Daniilidis, "Hand-eye calibration using dual quaternions," Int. J. Robotics Research, vol.18, no.3, pp.286–298, 1999.
[3] J.S. Park and B.J. Lee, "Vision-based real-time camera match moving using a known marker," Optical Engineering, vol.47, no.2, p.027201, 2008.
[4] B. Haralick, C. Lee, K. Ottenberg, and M. lle, "Review and analysis of solutions of the three point perspective pose estimation problem,"

Int. J. Comput. Vis., vol.13, no.3, pp.331–356, 1994.
[5] X. Gao, X. Hou, J. Tang, and H. Cheng, "Complete solution classification for the perspective-three-point problem," IEEE Trans. Pattern Anal. Mach. Intell., vol.25, no.8, pp.930–943, 2003.
[6] M. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," Commun. ACM, vol.24, no.6, pp.381–395, 1981.
[7] B. Triggs, "Camera pose and calibration from 4 or 5 known 3d points," Proc. Int'l Conf. on Computer Vision, pp.278–284, 1999.
[8] L. Quan and Z. Lan, "Linear n-point camera pose determination," IEEE Trans. Pattern Anal. Mach. Intell., vol.21, no.8, pp.774–780, 1999.
[9] A. Ansar and K. Daniilidis, "Linear pose estimation from points or lines," IEEE Trans. Pattern Anal. Mach. Intell., vol.25, no.5, pp.578–589, 2003.
[10] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate O(n) solution to the PnP problem," Int. J. Comput. Vis., vol.81, no.2, pp.155–166, 2008.
[11] S. Li, C. Xu, and M. Xie, "A robust O(n) solution to the perspective-n-point problem," IEEE Trans. Pattern Anal. Mach. Intell., vol.34, no.7, pp.1444–1450, 2012.
[12] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," IEEE Trans. Pattern Anal. Mach. Intell., vol.13, no.4, pp.376–380, 1991.
[13] C. Olsson, F. Kahl, and M. Oskarsson, "Branch-and-Bound methods for Euclidean registration problems," IEEE Trans. Pattern Anal. Mach. Intell., vol.31, no.5, pp.783–794, 2009.
[14] C. Lu, G. Hager, and E. Mjolsness, "Fast and globally convergent pose estimation from video images," IEEE Trans. Pattern Anal. Mach. Intell., vol.22, no.6, pp.610–622, 2000.
[15] D. DeMenthon and L. Davis, "Model-based object pose in 25 lines of code," Int. J. Comput. Vis., vol.15, no.1, pp.123–141, 1995.
[16] G. Schweighofer and A. Pinz, "Globally optimal O(n) solution to the PnP problem for general camera models," Proc. British Machine Vision Conference, pp.1–10, 2008.
[17] H. Hmam and J. Kim, "Optimal non-iterative pose estimation via convex relaxation," Image Vis. Comput., vol.28, no.11, pp.1515–1523, 2010.
[18] J.A. Hesch and S.I. Roumeliotis, "A direct least-squares (DLS) method for PnP," Proc. Int'l Conf. on Computer Vision, pp.383–390, 2011.
[19] G. Schweighofer and A. Pinz, "Robust pose estimation from a planar target," IEEE Trans. Pattern Anal. Mach. Intell., vol.28, no.12, pp.2024–2030, 2006.
[20] D. Cox, J. Little, and D. O'Shea, Using Algebraic Geometry, 2nd ed., Springer-Verlag, 2005.
[21] Z. Kukelova, M. Bujnak, and T. Pajdla, "Automatic generator of minimal problem solvers," Proc. Euro. Conf. on Computer Vision, pp.302–315, 2008.
[22] D. Lowe, "Distinctive image features from scale-invariant keypoints," Int. J. Comput. Vis., vol.60, no.2, pp.91–110, 2004.

Yinqiang Zheng received his Bachelor degree from the Department of Automation, Tianjin University, Tianjin, China, in 2006, and Master degree of engineering from Shanghai Jiao Tong University, Shanghai, China, in 2009. From 2009, he has been studying toward a Doctoral degree in the Department of Mechanical and Control Engineering, Tokyo Institute of Technology, Tokyo, Japan. His research interests include image processing, computer vision and optimization methods.

Shigeki Sugimoto received a B.S. degree in controls from Tokyo Institute of Technology in 1995 and M.S. degree from the Department of Mechanical and Environmental Informatics in 1997. He became an associate researcher in the Department of Mechanical and Control Engineering, Tokyo Institute of Technology Graduate School, in 2003. His research concerns computer vision, image measurement, and ITS-related research.

Masatoshi Okutomi received a BEng degree in mathematical engineering and information physics from the University of Tokyo, Japan, in 1981, and a MEng degree in control engineering from Tokyo Institute of Technology, Japan, in 1983. He joined Canon Research Center, Canon Incorporated, Tokyo, Japan, in 1983. From 1987 to 1990, he was a visiting research scientist in the School of Computer Science at Carnegie Mellon University, Pittsburgh, Pennsylvania. In 1993, he received a PhD degree for his research on stereo vision from the Tokyo Institute of Technology. Since 1994, he has been with the Tokyo Institute of Technology, where he is currently a professor in the Department of Mechanical and Control Engineering in the Graduate School of Science and Engineering.

View publication stats