A General and Simple Method for Camera Pose and Focal Length Determination

Yinqiang Zheng[1] Shigeki Sugimoto[2] Imari Sato[1,3] Masatoshi Okutomi[2]
[1]National Institute of Informatics, JAPAN
yqzheng@nii.ac.jp imarik@nii.ac.jp
[2]Department of Mechanical and Control Engineering, Tokyo Institute of Technology, JAPAN
shige@ok.ctrl.titech.ac.jp mxo@ctrl.titech.ac.jp
[3]Department of Information Processing, Tokyo Institute of Technology, JAPAN

Abstract

In this paper, we revisit the pose determination problem of a partially calibrated camera with unknown focal length, hereafter referred to as the PnPf problem, by using n ($n \geq 4$) 3D-to-2D point correspondences. Our core contribution is to introduce the angle constraint and derive a compact bivariate polynomial equation for each point triplet. Based on this polynomial equation, we propose a truly general method for the PnPf problem, which is suited both to the minimal 4-point based RANSAC application, and also to large scale scenarios with thousands of points, irrespective of the 3D point configuration. In addition, by solving bivariate polynomial systems via the Sylvester resultant, our method is very simple and easy to implement. Its simplicity is especially obvious when one needs to develop a fast solver for the 4-point case on the basis of the characteristic polynomial technique. Experiment results have also demonstrated its superiority in accuracy and efficiency when compared with the existing state-of-the-art solutions.

1. Introduction

To determine the camera pose with respect to (w.r.t.) a known 3D model by using n 3D points and their image projections is a classical problem in computer vision and photogrammetry [8]. This problem has been extensively studied by existing literature, in the case of a fully calibrated, partially calibrated and uncalibrated camera. It reduces to the well-known perspective-n-point (PnP) problem [11, 15–17, 22] in the presence of a fully calibrated camera, while to the camera resection problem [8] in the uncalibrated case. The strict requirement of full calibration in PnP prevents its application from scenarios where a portion of the intrinsic parameters might change online. In contrast, the uncalibrated case completely ignores some reasonable assumptions on the camera parameters, such as unit aspect ratio, zero skew and centered principle point for a decent digital camera. This dilemma arouses much attention on the pose determination problem of a partially calibrated camera, with unknown focal length [1, 3, 5, 18, 21], unknown focal length and aspect ratio [6], unknown focal length and principle point [21], unknown focal length and radial distortion [2, 10, 14], and so on.

In this paper, we focus primarily on the partially calibrated case with unknown focal length, *i.e*. the PnPf problem. The other intrinsic parameters, including aspect ratio, skew, principle point and distortion, are assumed to be calibrated beforehand or known due to some prior knowledge. Although it seems to be more restricted than the other aforementioned variants, we believe that the PnPf problem itself deserves deeper investigation. This is especially true, when considering its usefulness in such important applications as augmented reality with a zooming camera, incremental structure-and-motion [9] and geolocalization [19] using Internet photo collections. Note that in [19], the authors had to discretize the focal length and apply a standard *P3P* solver with RANSAC, which is surely not elegant.

1.1. Related Works on PnP

Since some existing works for PnPf are deeply rooted in their predecessors for PnP, we would like first to review some closely related works on PnP. The minimal P3P problem has been properly resolved, leading to many prominent solvers, like those in [11, 16]. One current trend is to develop accurate solutions for the PnP problem, which should scale reasonably w.r.t. the number of points n. Several excellent noniterative $O(n)$ solutions have been proposed, including EPnP [15], RPnP [17] and OPnP [22]. Specifically, EPnP [15] tries to express all 3D points into the linear combination of 4 (3 in the planar case) virtual control points, and approximately solve the resulting multivariate polynomial system using linearization. However, EPnP is poor in accuracy when n is small (*e.g*., $4 \leq n \leq 6$), because the loss of accuracy in linearization can hardly be compensated

by data redundancy. In addition, it explicitly differentiates the planar point configuration from the nonplanar one, thus tends to be inaccurate in the intermediate near-planar case. RP*n*P [17] successfully conquers these drawbacks by dividing all points into triplets with a common edge and separating the estimation of rotation axis and angle.

1.2. Related Works on P*n*Pf

When the focal length f is unknown, the pose determination problem becomes much more complicated. For the minimal P4Pf problem[1], specialized solutions were proposed in [1] for the planar case, while in [21] for the nonplanar case. The first general method in terms of point configuration was developed by Bujnak *et al*. [3], on the basis of the ratio of distance constraint and the polynomial solving techniques via Gröbner basis (GB) and hidden variable resultant (HVR). As complained in [19], both solvers are not fast enough for RANSAC. Although the automatic GB solver generator [12] has relieved much burden, it would still be very challenging, especially for nonexperts, to improve the GB solver, say, by reducing the size of the elimination template for faster speed or choosing other monomials in the basis for higher numerical stability. Actually, Bujnak [2] tried to find the best GB solver from millions of candidates, which is undoubtedly a tough task. This P4Pf solver was later significantly accelerated in [4]. However, to implement it requires deep understanding on how to calculate the characteristic polynomial of the action matrix.

On the other hand, the HVR based solver is much simpler in implementation through polynomial eigenvalue factorization (*e.g*. polyeig in MATLAB). Again, when one needs to accelerate the solver, the characteristic polynomial of an 8×8 resultant matrix has to be computed. The quotient-free Gauss-Jordan elimination method was revitalized to this end [7], which still requires some tedium low-level operations, like polynomial multiplication and division.

The aforementioned works are dedicated to the P4Pf problem, whose underlying schemes could not be extended to the general P*n*Pf problem with n varying from 5 to several thousands. There are some endeavours toward a general P*n*Pf solver. Choi *et al*. [5] developed a branch-and-bound based method for general configuration, which is unfortunately too slow for online applications. By extending the idea of EP*n*P [15], Penate-Sanchez *et al*. [18] made one solid step toward a noniterative $O(n)$ solution for the P*n*Pf problem. Unfortunately, the proposed method does not work for the P4Pf and P5Pf problems. As for the point configuration, similar to EP*n*P, it requires to develop a specialized solution for the planar case. In addition, the exhaustive linearization technique might deteriorate the computational efficiency severely.

[1]Strictly speaking, the 4-point case is slightly overconstrained, since three and a half points in general are sufficient to determine the 6dof pose parameters and the focal length. A common practice, as in [2, 3], is to ignore one constraint and regard the 4-point case as exactly minimal.

1.3. Overview of the Proposed Method

The most common procedure in geometric computer vision is first to use a minimal solver in conjunction with RANSAC to filter out possible outliers, and then to find a better estimate using redundant inliers [8]. This motivates us to develop a general method for the P*n*Pf problem, which is suited for the minimal 4-point based RANSAC application, and can also be easily generalized to the redundant case with even thousands of points. We draw inspiration from the RP*n*P method [17] to divide all 3D points into triplets sharing two endpoints, which define the rotation axis of the rotation between the camera framework and the intermediate object framework. The rotation axis and angle can be sequentially, thus more easily, determined than estimating both of them simultaneously.

We introduce the angle constraint into the P*n*Pf problem, as an important complement to the well-known distance constraint and the ratio of distance constraint. Based on this constraint and a variable translation technique, we derive a compact bivariate polynomial equation for a point triplet, which is much simpler than those derived from the distance and the ratio constraint. Thanks to this polynomial equation, we are able to develop a truly general method for P*n*Pf, in terms both of the number of points n and of the point configuration[2]. Rather than dealing with multivariate polynomial systems as in [2, 3] for P4Pf and [18] for P*n*Pf, we only need to solve bivariate polynomials via the Sylvester resultant, which makes our method easy to implement. Especially, for P4Pf, the characteristic polynomial of the structured 6×6 resultant matrix can be directly calculated via cofactors in double precision arithmetics, in contrast to the more complicated methods in [4, 7].

To resolve the minor issues of error accumulation and randomness in rotation axis selection, we also propose to refine the solution further by directly minimizing an unconstrained cost function, at the cost of a little extra computational burden. Experiment results have demonstrated that our proposed method is more accurate and computationally efficient than existing state-of-the-art solutions.

2. Preliminaries

2.1. The P*n*Pf Problem

As shown in Fig.1, the P*n*Pf problem is to estimate the focal length f, the rotation R_w^c and translation $\mathbf{t}_w^c$ between the world framework $O_wX_wY_wZ_w$ and the camera framework $O_cX_cY_cZ_c$, by using n ($n \geq 4$) 3D points X_i^w and their image

[2]Except the inherently degenerate configurations, such as all points lying on a line and planar points viewed by a parallel-front camera.

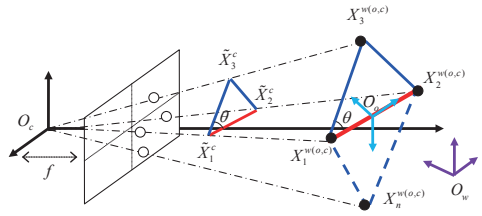


Figure 1. A diagram illustrating the PnPf problem.

projections $\mathbf{x}_i, i = 1, 2, \cdots, n$. Without loss of generality, we can assume that the camera calibration matrix K satisfies $K = diag([f, f, 1])$. The pinhole projection equation reads

$$\lambda_i \left[\mathbf{x}_i; 1\right] = K\left(R_w^c X_i^w + \mathbf{t}_w^c\right), i = 1, 2, \cdots, n, \tag{1}$$

where λ_i denotes the depth factor. To simplify notation, we use the MATLAB-style operator ';' to denote column-wise concatenation of column vectors or scalars.

When the focal length is known, the PnPf problem is simplified to the well-known PnP problem.

2.2. Procedures of RPnP

RPnP is a prominent solution to the PnP problem. Its basic procedures are as follows:

Step-1. Choose two 3D points, *e.g.* X_1^w and X_2^w in Fig.1, and build an intermediate object framework $O_oX_oY_oZ_o$, whose origin lies at the middle of X_1^w and X_2^w, while the z_o-axis is aligned with $\overrightarrow{X_1^w X_2^w}$. The x_o-axis can be chosen arbitrarily. It is possible to transform all 3D points X_i^w into X_i^o by using the known rotation R_w^o and translation $\mathbf{t}_w^o$. The remaining task is to estimate the rigid transformation R_o^c and $\mathbf{t}_o^c$ between $O_cX_cY_cZ_c$ and $O_oX_oY_oZ_o$. An important observation is that R_o^c can be decomposed as

$$R_o^c = R_1 R_2 = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} 0 & c & -s \\ 0 & s & c \\ 1 & 0 & 0 \end{bmatrix}, \tag{2}$$

where R_1 is an arbitrary rotation matrix such that its third column $[r_{13}; r_{23}; r_{33}]$ equals z_o, and R_2 denotes a rotation of φ degrees around the z_o-axis, with $c = cos\varphi$ and $s = sin\varphi$.

Step-2. Every remaining point, *e.g.* X_3^w in Fig.1, together with the two endpoints, forms a triangle, *i.e.* point triplet, for which one can build a 4-degree univariate polynomial. The key is to choose one variable properly such that it is shared by all $n-2$ univariate polynomials.

Step-3. Determine the axis z_o by minimizing the least-square error of all $n-2$ polynomials, which boils down to the solving of a 7-degree univariate polynomial.

Step-4. Estimate the angle φ and the translation $\mathbf{t}_o^c$.

Impressed by the excellent performance of RPnP in [17], we are particularly interested in the generalization of the RPnP procedures to the PnPf problem.

Since the PnPf problem concerned in this paper relies on point correspondences as well, the operations in *Step-1* can be reused completely. As for the criterion to select the rotation axis, we follow the heuristic in [17] to randomly sample n 3D point pairs and choose the one with longest distance as the two endpoints. In the following, we focus primarily on *Step-2* to find a polynomial for each point triplet, whose variables are shared by all triplets.

3. Polynomial Equation for Point Triplet

3.1. Distance and Ratio of Distance Constraints

The distance constraint has been used in [2] for P4Pf and [18] for PnPf. It is based on the simple fact that the Euclidean distance keeps unchanged under any rigid transformation. For example, the triplet $X_1^cX_2^cX_3^c$ in Fig.1 provides three quadratic distance constraints

$$\left\|X_1^c - X_2^c\right\|_2^2 = d_{12}, \left\|X_1^c - X_3^c\right\|_2^2 = d_{13}, \left\|X_2^c - X_3^c\right\|_2^2 = d_{23}, \tag{3}$$

in which d_{12}, d_{13} and d_{23}, denoting the squared length of $X_1^wX_2^w$, $X_1^wX_3^w$ and $X_2^wX_3^w$, are known in the world framework.

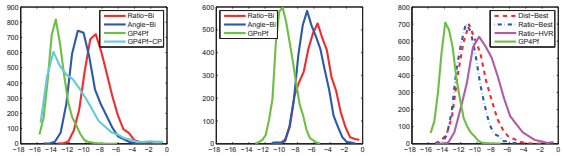
As confirmed in [2], the distance constraint would lead to much more complicated polynomials than those from the following ratio of distance constraint, thus it shall not be explored further.

The ratio of distance constraint was first utilized for the P4Pf problem in [3], and later systematically investigated in [2]. The core idea is to build another triangle (*e.g.* $\tilde{X}_1^c\tilde{X}_2^c\tilde{X}_3^c$) that is similar to the original one (*e.g.* $X_1^cX_2^cX_3^c$). When image points are noise-free, the coordinate of $\tilde{X}_i^c$ in $O_cX_cY_cZ_c$ can be expressed as $\alpha_i[\mathbf{x}_i; f]$, where α_i is the pseudo-depth.

Now, we take the triangle $\tilde{X}_1^c\tilde{X}_2^c\tilde{X}_3^c$ as an example. Since it is similar to $X_1^cX_2^cX_3^c$, the invariance of ratio of distance provides two homogeneous equations

$$\begin{aligned} \frac{\left\|\tilde{X}_1^c - \tilde{X}_3^c\right\|_2^2}{\left\|\tilde{X}_1^c - \tilde{X}_2^c\right\|_2^2} &= \frac{\left\|\alpha_1\left[\mathbf{x}_1; f\right] - \alpha_3\left[\mathbf{x}_3; f\right]\right\|_2^2}{\left\|\alpha_1\left[\mathbf{x}_1; f\right] - \alpha_2\left[\mathbf{x}_2; f\right]\right\|_2^2} = \frac{d_{13}}{d_{12}}, \\ \frac{\left\|\tilde{X}_2^c - \tilde{X}_3^c\right\|_2^2}{\left\|\tilde{X}_1^c - \tilde{X}_2^c\right\|_2^2} &= \frac{\left\|\alpha_2\left[\mathbf{x}_2; f\right] - \alpha_3\left[\mathbf{x}_3; f\right]\right\|_2^2}{\left\|\alpha_1\left[\mathbf{x}_1; f\right] - \alpha_2\left[\mathbf{x}_2; f\right]\right\|_2^2} = \frac{d_{23}}{d_{12}}. \end{aligned} \tag{4}$$

Due to the homogeneity, it is possible to fix the scale of α_i ($\alpha_i > 0$) without causing any singularity. In this paper, we let that $\alpha_1 = 2 - \alpha_2$, that is, the average pseudo-depth of $\tilde{X}_1^c$ and $\tilde{X}_2^c$ is 1. This is slightly different from the way in [2] to fix α_1 to be 1. In addition, considering that f appears in even degrees in Eq.(4), we introduce w such that $w = f^2$.



(a) 4-point (b) n-point (n=20) (c) 4-point

Figure 2. Numerical stability with noise-free synthetic data. The experiment details can be found in Sec.5.1.1.

After regarding α_2 and w as the hidden variable, we can eliminate α_3 by using the Sylvester resultant. This would lead to a bivariate polynomial with 25 monomials, whose highest degree monomial is $w^4\alpha_2^4$.

As shown in Fig.2, the numerical stability of using this bivariate polynomial, especially for PnPf, is not very strong. In the following, we try to reduce the degree by introducing the angle constraint.

3.2. Angle Constraint

Since $\tilde{X}_1^c\tilde{X}_2^c\tilde{X}_3^c$ is similar to $X_1^cX_2^cX_3^c$, the angle θ between $\overrightarrow{X_1^cX_2^c}$ and $\overrightarrow{X_1^cX_3^c}$ keeps constant. Due to the cosine law, $cos\theta$ can be calculated by $(d_{12}+d_{13}-d_{23})\,/\left(2\sqrt{d_{12}}\sqrt{d_{13}}\right)$, while in the camera framework, it can be expressed as

$$\begin{aligned} cos\theta =& \frac{(\tilde{X}_2^c-\tilde{X}_1^c)\cdot(\tilde{X}_3^c-\tilde{X}_1^c)}{\left\|\tilde{X}_2^c-\tilde{X}_1^c\right\|_2\left\|\tilde{X}_3^c-\tilde{X}_1^c\right\|_2} = \frac{(\tilde{X}_2^c-\tilde{X}_1^c)\cdot(\tilde{X}_3^c-\tilde{X}_1^c)}{\sqrt{d_{13}/d_{12}}\left\|\tilde{X}_2^c-\tilde{X}_1^c\right\|_2^2} \\ =& \frac{(\alpha_2\left[\mathbf{x}_2;f\right]-\alpha_1\left[\mathbf{x}_1;f\right])\cdot(\alpha_3\left[\mathbf{x}_3;f\right]-\alpha_1\left[\mathbf{x}_1;f\right])}{\sqrt{d_{13}/d_{12}}\left\|\alpha_2\left[\mathbf{x}_2;f\right]-\alpha_1\left[\mathbf{x}_1;f\right]\right\|_2^2}, \end{aligned} \tag{5}$$

in which the '$\cdot$' operator denotes the dot-product of two vectors, and the first ratio constraint in Eq.(4) has been incorporated.

It is critical to note that α_3 appears linearly in Eq.(5). After expressing α_3 as a function of α_2 and w, we can plug the expression into the second ratio constraint in Eq.(4). This would lead to a bivariate polynomial with 20 monomials, whose highest degree monomial has been successfully reduced to $w^3\alpha_2^4$, in contrast to $w^4\alpha_2^4$ from the ratio of distance constraint only.

Actually, the angle constraint in Eq.(5) is widely known, and has been used for the P3P problem in a different mathematical form [16].

3.3. Translating Variable

Unfortunately, the 20 coefficients of the bivariate polynomial from the angle constraint are correlated, which would weaken numerical stability, as shown in Fig.2. We have recognized that, by translating the variable α_2 to α_2+1, the number of monomials can be further reduced to 14, although the highest degree monomial keeps to be $w^3\alpha_2^4$.

Specifically, on the basis of the angle constraint together with variable translation, every remainging point $X_i^c, i = 3,4,\cdots,n$, in conjunction with the two endpoints X_1^c and X_2^c, leads to a compact bivariate polynomial $P_i(w,\alpha_2)$ with 14 monomials $w^3\alpha_2^4$, $w^2\alpha_2^4$, $w\alpha_2^4$, α_2^4, $w^2\alpha_2^3$, $w\alpha_2^3$, α_2^3, $w^2\alpha_2^2$, $w\alpha_2^2$, α_2^2, $w\alpha_2$, α_2, w, 1. The polynomial coefficients can be easily calculated by using the cosine values of angles, the distance between points and the image projections.

Before proceeding to solving the PnPf problem, let us point out that the polynomials $P_i(w,\alpha_2), i = 3,4,\cdots,n$, assume some trivial (physically infeasible) solutions.

Proposition 1. *The following four trivial solutions (TS) are compatible with all $n-2$ polynomials $P_i(w,\alpha_2)$: (TS-1). $\alpha_2 = -1$, $w = -\|\boldsymbol{x}_1\|_2^2$; (TS-2). $\alpha_2 = 1$, $w = -\|\boldsymbol{x}_2\|_2^2$; (TS-3) and (TS-4). a conjugate complex pair $w = 0$, $\alpha_2 = \left(-b \pm \sqrt{b^2-4ac}\right)/(2a) - 1$, where $a = \|\boldsymbol{x}_1\|_2^2 + \|\boldsymbol{x}_2\|_2^2 + 2\boldsymbol{x}_1\cdot\boldsymbol{x}_2$, $b = -4(\|\boldsymbol{x}_1\|_2^2 + \boldsymbol{x}_1\cdot\boldsymbol{x}_2)$, $c = 4\|\boldsymbol{x}_1\|_2^2$.*

The proof is very straightforward by plugging the values of α_2 or w back into the ratio of distance constraint in Eq.(4). Note that similar trivial solutions exist in the ratio based GB solvers in [2]. The existence of such trivial solutions and their multiplicity could largely explain why the size of action matrix varies among the solvers in Table 6.1 of [2].

In the following, we show how to solve the PnPf problem by using our compact bivariate polynomial.

4. Proposed Method for PnPf

4.1. Estimating Focal Length and Rotation Axis

Considering that the focal length $f = \sqrt{w}$ and the rotation axis $z_o = \alpha_2\left[\mathbf{x}_2;f\right] - \alpha_1\left[\mathbf{x}_1;f\right]$, we only need to estimate w and α_2.

4.1.1 Minimal 4-Point Case

In the presence of 4 points, two triplets lead to a bivariate polynomial system

$$P_3(w,\alpha_2) = 0, P_4(w,\alpha_2) = 0. \tag{6}$$

By treating α_2 as the hidden variable and using the Sylvester resultant, Eq.(6) can be solved via the polynomial eigenvalue factorization technique [13]

$$M(\alpha_2)\mathbf{v} = \textstyle\sum_{k=0}^{4}\alpha_2^k M_k\mathbf{v} = \mathbf{0}, \tag{7}$$

in which M_k can be constructed by using the coefficients of the two polynomials. $\mathbf{v} = [w^5; w^4; \cdots; w; 1]$ is the eigenvector, from which we can extract w.

As suggested in [4,7], we can solve Eq.(7) via the characteristic polynomial technique as well. Specifically, we first calculate the symbolic determinant of $M(\alpha_2)$ and then solve α_2 via the Sturm-sequence method in a prescribed interval ([-1, 1] in our method). Given α_2, w can be determined in closed form using either polynomial in Eq.(6).

The challenge lies in how to calculate the determinant of $M(\alpha_2)$ accurately and efficiently. Fortunately, $M(\alpha_2)$

is a 6×6 well-structured matrix with 12 zero entries. This enables us to directly calculate its determinant via cofactors in double precision arithmetics, rather than resorting to the more complicated techniques in [4, 7]. As shown in Fig.2(a), the stability is sufficiently close to that of the polynomial eigenvalue technique.

We have found that Eq.(6) usually has 16 solutions, among which 6 solutions, including TS-1 (multiplicity 2), TS-2 (multiplicity 2), TS-3 and TS-4, are trivial.

4.1.2 Overconstrained n-Point Case

In the presence of n ($n \geq 5$) points, we can build $n-2$ bivariate polynomials in all. Due to image noise, these polynomials are not simultaneously feasible, except for those trivial solutions in Proposition 1.

Inspired by [17], we directly minimize the sum of squared error defined as $F(w, \alpha_2) = \sum_{i=3}^{n} [P_i(w, \alpha_2)]^2$. To retrieve its global minimum, we can solve the following bivariate polynomial system derived from its first order optimality condition

$$\partial F(w, \alpha_2)/\partial w = 0, \partial F(w, \alpha_2)/\partial \alpha_2 = 0. \tag{8}$$

Similarly, Eq.(8) can be solved by polynomial eigenvalue factorization. Although the maximum degree of α_2 is 8 here, the numerical stability is sufficiently strong, as shown in Fig.2(b), thanks to the derived compact bivariate polynomial.

4.2. Estimating Rotation Angle and Translation

After the focal length and rotation axis are determined, R_1 in Eq.(2) and K in Eq.(1) are known. Based on Eq.(1), the projection equation using 3D points in the intermediate framework can be rewritten as

$$\lambda_i \left[\mathbf{x}_i; 1\right] = K \left(R_1 R_2 X_i^o + \mathbf{t}_o^c\right), i = 1, 2, \cdots, n. \tag{9}$$

Due to image noise, the equation could not be completely satisfied. We can minimize the sum of squared error

$$\min_{\lambda_i, c, s, \mathbf{t}_o^c} \quad \sum_{i=1}^{n} \left\| \lambda_i \left[\mathbf{x}_i; 1\right] - K \left(R_1 R_2 X_i^o + \mathbf{t}_o^c\right) \right\|_2^2. \tag{10}$$

After projecting out the linear variables λ_i and $\mathbf{t}_o^c$, Eq.(10) can be simplified in a standard quadratic minimization problem with a unit-norm constraint

$$\min_{c,s} \quad [c; s]^T G [c; s] - \mathbf{b}^T [c; s], s.t., \; c^2 + s^2 = 1, \tag{11}$$

in which G is a known 2×2 data matrix, while $\mathbf{b}$ a 2D column vector. This problem can be easily solved via quadratic eigenvalue factorization [20]. It can be further written into a quartic polynomial, for which closed-from solutions exist. Note that the solution for Eq.(10) in RPnP is not optimal.

4.3. Iterative Refinement

Until now, all the focal length and pose parameters have been determined. To conquer the minor problems of error accumulation in sequential determination of rotation axis and angle, as well as the randomness in rotation axis selection, we propose to refine the estimated results further.

Inspired by [22], we parameterize the rotation R_o^c using the fractional quaternion and fix the scale of the quaternion to be the reciprocal of the average depth. To minimize the sum of squared error leads to a unconstrained minimization problem with five variables (see the supplementary material for the details). We use the typical Gauss-Newton method to solve the optimization problem. Since the initialization is accurate enough, the maximum number of iteration is set to be 10 throughout the following experiment section.

5. Experiment Results

In this section, we investigate the performance of our proposed general method for the PnPf problem, referred to as **GPnPf** (**GP4Pf** when n=4) without refinement and **GPnPf+GN** (**GP4Pf+GN** when n=4) with Gauss-Newton refinement. When $n > 4$, we compare our method with the-state-of-the-art solution in [18], denoted by **UPnPf** or **UPnPf+GN** when the refinement step in [18] is used. We also include into comparison the direct linear transformation (**DLT**) for non-planar 3D points and **HOMO** for planar points, which extract the focal length and pose parameters from the projection matrix and the homography, respectively. Note that **DLT** assumes an uncalibrated camera, while **HOMO** uses partial prior calibration information. As reference, we also consider the calibrated **RPnP** method by using the ground-truth focal length.

In the 4-point case, we compare our method with the best distance-based GB solver (**Dist-Best**) with a 561×581 elimination template [2], the best ratio-based GB solver (**Ratio-Best**) with a 139×153 elimination template [2] and the ratio-based HVR solver (**Ratio-HVR**) [3].

We implement our method in MATLAB and use the MATLAB codes of **UPnPf**, **UPnPf+GN**, **Dist-Best**, **Ratio-Best** and **Ratio-HVR**, which are provided by their respective authors. However, the characteristic polynomial technique of **GP4Pf** is implemented in C and interfaced via a MATLAB-MEX. We run all codes on a notebook with 2.8GHz CPU and 4GB RAM.

5.1. Synthetic Data

We synthesize a virtual perspective camera with zero-distortion, zero-skew and unit aspect ratio. The principle point lies at the image center. The image resolution is 800×600 pixels. We randomly vary the focal length from 200 to 2000 pixels. n 3D reference points are randomly generated in the camera framework. For the nonplanar 3D

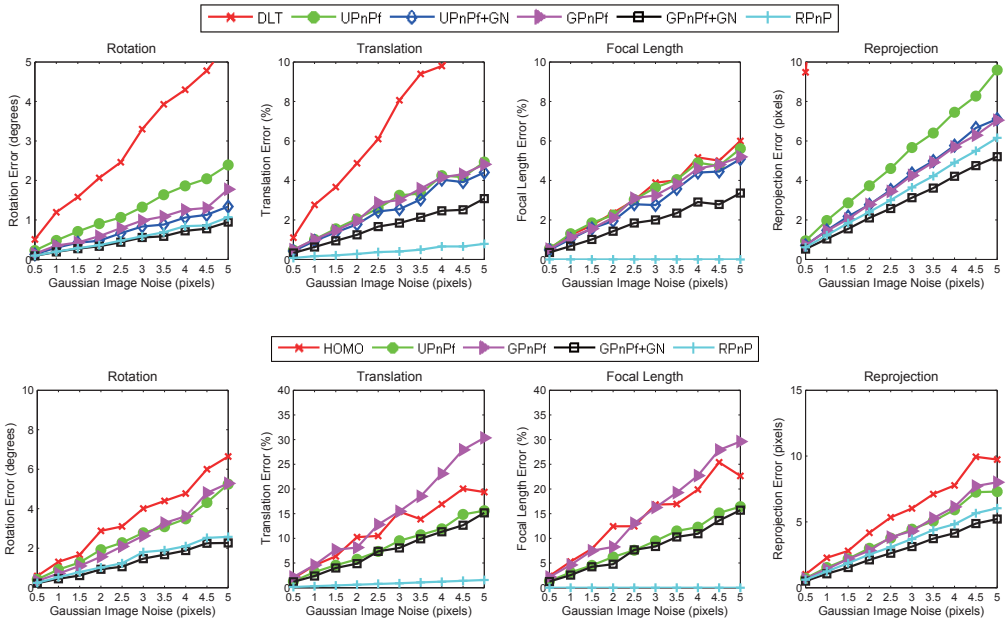


Figure 3. Experiment results w.r.t. varying noise levels (n=8 points) in case of nonplanar 3D (1st row) and planar (2nd row) point configuration. The median rotation, translation, focal length and reprojection error are shown in the 1st, 2nd, 3rd and 4th column, respectively.

case, these points are randomly distributed in the box of [-2,2]×[-2,2]×[4,8], while for the near-planar case, they are in [-2,2]×[1,2]×[4,8]. Then, we randomly synthesize the ground-truth rotation and translation, and transform those 3D points into the world framework. Unless explicitly stated otherwise, we measure the absolute error (in degrees) of rotation, the relative error (%) of translation and focal length, between the estimated entities and their corresponding ground truth. The root mean square reprojection error (in pixels) is also measured.

5.1.1 Numerical Stability

First, we experimentally study how the bivariate polynomial from the angle constraint with variable translation (in **GP4Pf** and **GPnPf**) improves numerical stability over the ones from the angle constraint alone (**Angle-Bi**) and the ratio constraint (**Ratio-Bi**). At each trial, we synthesize 4 or n (n=20) image points without any noise, and measure the log_{10} value of the relative error between the estimated focal length f and the ground truth f_{gt}, *i.e.* $|f - f_{gt}|/f_{gt}$. The error histograms over 4000 trials are shown in Fig.2(a) and Fig.2(b) for the 4-point and n-point (n=20) case. In the 4-point case, we also compare the stability of **GP4Pf** with that of **Dist-Best**, **Ratio-Best** and **Ratio-HVR** in Fig.2(c).

From Fig.2, we can see that our compact bivariate polynomial contributes significantly to improving numerical stability. Actually, **GP4Pf** is clearly superior in stability over the state-of-the-art 4-point solutions.

We have also compared the characteristic polynomial based implementation of **GP4Pf**, denoted by **GP4Pf-CP** in Fig.2(a). Its stability is very strong overall, except a few inaccurate estimates, which are probably caused by error accumulation in the three-stage procedure to calculate f.

5.1.2 Accuracy w.r.t. Varying Noise

Now, we add zero-mean Gaussian noise onto the image points. We first vary the noise deviation level δ from 0.5 to 5 pixels, while keeping n to be 8. At each noise level, we run 500 independent tests and report the median rotation, translation, focal length and reprojection error in Fig.3. Note that in the planar case, we use the specialized version of **UPnPf**, in which the Gauss-Newton step plays no role.

Based on Fig.3, in terms of the rotation and reprojection error, **GPnPf** is better than **UPnPf** in the nonplanar 3D case (1st row), but slightly worse in the planar case (2nd row). Our **GPnPf+GN** has the highest accuracy, irrespective of the error criterion and the point configuration. Its rotation accuracy is even comparable to that of the RPnP method.

5.1.3 Accuracy w.r.t. Varying Number of Points

Then, we fix δ=2 pixels but vary n from 4 to 15. At each n, 500 independent tests are conducted. To illustrate the error distribution, we present the rotation error of **UPnPf**, **UPnPf+GN**, **GPnPf** and **GPnPf+GN** in Fig.4 by using the MATLAB boxplot function.

From Fig.4, we can observe that **UPnPf** is inaccurate when $6 \leq n \leq 7$, and inapplicable at all when $4 \leq n \leq 5$. Our **GPnPf** is general in terms of the number of points, and works reasonably well even in the near-planar case (2nd

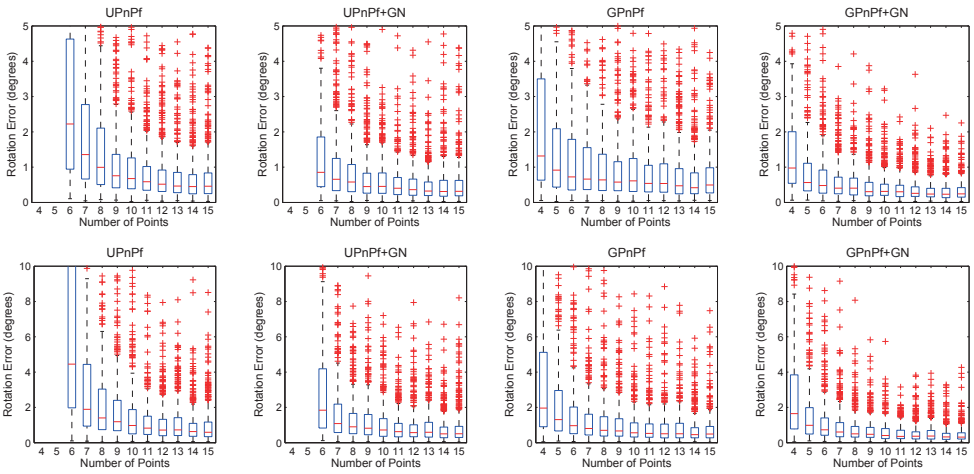


Figure 4. Experiment results w.r.t. varying number of points (δ=2 pixels) in case of nonplanar 3D (1st row) and near-planar (2nd row) point configuration. For each method, the distribution of rotation error over 500 trials is shown by using the MATLAB boxplot function. The values between 25% and 75% percentiles are shown as a box with horizontal line at the median. The red crosses indicate data beyond 1.5 times the inter-percentile range.

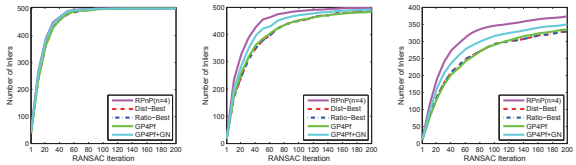


Figure 5. Results of competing 4-point solutions in combination with RANSAC on synthetic data with 50% outliers and noise levels of 0.1 pixels (left), 0.5 pixels (middle) and 1 pixel (right).

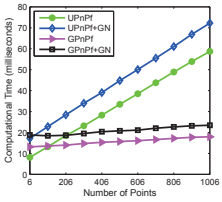


Figure 6. Running time w.r.t varying number of points.

Method	Time
Dist-Best	100.5 ms
Ratio-Best	2.1 ms
Ratio-HVR	7.6 ms
GP4Pf	1.3 ms
GP4Pf-CP	26.8 μs

Table 1. Running time of competing 4-point solutions.

row). Again, **GP*n*Pf+GN** is the best one among all compared methods. It is therefore recommended to refine the solutions whenever possible.

5.1.4 Effectiveness with RANSAC

Here, we explore the effectiveness of competing 4-point solutions when used with RANSAC. At each trial, we randomly generate 1000 3D points and project them on the image. 50% of the image points are severely corrupted to simulate outliers, while the remaining points disturbed by 0.1, 0.5 and 1 pixel Gaussian noises. Then, we randomly sample 500 4-point subsets and use the same 4-point subset for all solutions. We count the maximum number of inliers found so far by each solution. The inlier threshold is 2 pixels. The average counts over 200 trials in the first 200 samplings are shown in Fig.5, from which we can observe that **GP4Pf** is as effective as **Dist-Best** and **Ratio-Best**, while **GP4Pf+GN** can be slightly better. **Ratio-HVR** is omitted since it overlaps **Dist-Best** and **Ratio-Best**.

5.1.5 Computational Time

It is straightforward to verify that the complexity of **GP*n*Pf** and **GP*n*Pf+GN** is $O(n)$. Here, we compare its speed with that of **UP*n*Pf** and **UP*n*Pf+GN** when varying n from 6 to 1006. For each n, we conduct 500 tests and report the average running time in milliseconds (ms) in Fig.6.

In the minimal 4-point case, considering that those peripheral operations, like calculating the polynomial coefficients, Gauss-Newton refinement (in **GP4Pf+GN**) and 3D-3D registration (in **Dist-Best** and **Ratio-Best**), can be made very fast by implementing them in C, we simply show the time of solving the polynomial system, as in Table 1. We observe that **GP4Pf** is faster than the other compared solutions. Its simple characteristic polynomial based implementation in C takes only 26.8 microseconds (μs), which compares favorably with the fast solver reported in [4].

5.2. Real Images

We have also tested our proposed method using real images captured by a zooming camera with 2144×1424 resolution. The principle point is assumed to be at the image center. After establishing tentative correspondences between the input image and the reference image, we first use **GP4Pf** in RANSAC to remove potential outliers. Then, all inliers are used to estimate the focal length and pose parameters by **GP*n*Pf+GN**. As shown in Fig.7, our proposed method works very well in real scenarios and provides visually satisfying results.

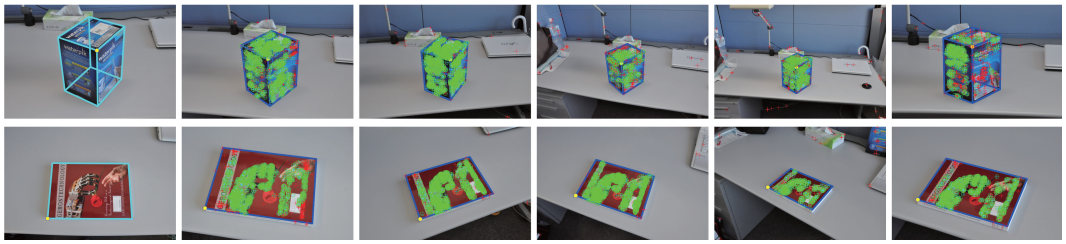


Figure 7. Experiments using a 3D box (1st row) and a planar book cover (2nd row). In each row, the first image shows the template, while the remaining five images are the input ones. For each input image, inliers (green circle) have been differentiated from outliers (red cross) by using RANSAC together with **GP4Pf**. The input image is augmented by the projected contour using the results of **GP*n*Pf+GN**.

6. Conclusion

We have proposed a general method for the pose determination problem of a partially calibrated camera with unknown focal length. In the minimal 4-point case, the proposed method is numerically stable and effective when used in RANSAC loop to remove outliers. It can also be generalized to handle large-scale problems due to its $O(n)$ complexity, and compares favorably with the state-of-the-art methods in speed and accuracy. Our method can handle any nondegenerate configuration in a unified way. In spite of its generality, the proposed method is very simple to implement, without requiring expertise in the Gröbner basis technique nor low-level polynomial operations. All these advantages are attributed to the compact bivariate polynomial equation for a point triplet that we have derived from the angle constraint[3].

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.

References

[1] M. Abidi and T. Chandra. A new efficient and direct solution for pose estimation using quadrangular targets: algorithm and evaluation. *TPAMI*, 17(5):534–538, 1995. 1, 2

[2] M. Bujnak. Algebraic solutions to absolute pose problems. *PhD Thesis, Czech Technical University*, 2012. 1, 2, 3, 4, 5

[3] M. Bujnak, Z. Kukelova, and T. Pajdla. A general solution to the p4p problem for camera with unknown focal length. *Proc. CVPR*, pages 1–8, 2008. 1, 2, 3, 5

[4] M. Bujnak, Z. Kukelova, and T. Pajdla. Making minimal solvers fast. *Proc. CVPR*, pages 1506–1513, 2012. 2, 4, 5, 7

[5] K. Choi, S. Lee, and Y. Seo. A branch-and-bound algorithm for globally optimal camera pose and focal length. *Image and Vistion Computing*, 28(9):1369–1376, 2010. 1, 2

[6] Y. Guo. A novel solution to the P4P problem for an uncalibrated camera. *Journal of Mathematical Imaging and Vision*, 45(2):186–198, 2013. 1

[7] R. Hartley and H. Li. An efficient hidden variable approach to minimal-case camera motion estimation. *TPAMI*, 34(12):2303–2314, 2012. 2, 4, 5

[8] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge Univ. Press, 2nd edition, 2003. 1, 2

[9] A. Irschara1, C. Zach, J.-M. Frahm, and H. Bischof. From structure-from-motion point clouds to fast location recognition. *Proc. CVPR*, pages 2599–2606, 2009. 1

[10] K. Josephson and M. Byrod. Pose estimation with radial distortion and unknown focal length. *Proc. CVPR*, pages 2419–2426, 2009. 1

[11] 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. 1

[12] Z. Kukelova, M. Bujnak, and T. Pajdla. Automatic generator of minimal problem solvers. *Proc. ECCV*, pages 302–315, 2008. 2

[13] Z. Kukelova, M. Bujnak, and T. Pajdla. Polynomial eigenvalue solutions to minimal problems in computer vision. *TPAMI*, 34(7):1381–1393, 2012. 4

[14] Z. Kukelova, M. Bujnak, and T. Pajdla. Real-time solution to the absolute pose problem with unknown radial distortion and focal length. *Proc. ICCV*, pages 2816–2823, 2013. 1

[15] V. Lepetit, F. Moreno-Noguer, and P. Fua. EPnP: An accurate O(n) solution to the PnP problem. *IJCV*, 81(2):155–166, 2008. 1, 2

[16] S. Li and C. Xu. A stable direct solution of perspective-three-point problem. *IJPRAI*, 25(5):627–642, 2011. 1, 4

[17] S. Li, C. Xu, and M. Xie. A robust O(n) solution to the perspective-n-point problem. *TPAMI*, 34(7):1444–1450, 2012. 1, 2, 3, 5

[18] A. Penate-Sanchez, J. Andrade-Cetto, and F. Moreno-Noguer. Exhaustive linearization for robust camera pose and focal length estimation. *TPAMI*, 35(10):2387–2400, 2013. 1, 2, 3, 5

[19] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring image collections in 3D. *ACM TOG (SIGGRAPH 2006)*, pages 835–846, 2006. 1, 2

[20] F. Tisseur and K. Meerbergen. The quadratic eigenvalue problem. *SIAM Review*, 43(2):235–286, 2001. 5

[21] B. Triggs. Camera pose and calibration from 4 or 5 known 3d points. *Proc. ICCV*, pages 278–284, 1999. 1, 2

[22] Y. Zheng, Y. Kuang, S. Sugimoto, K. Astrom, and M. Okutomi. Revisting the PnP problem: A fast, general and optimal solution. *Proc. ICCV*, pages 2344–2351, 2013. 1, 5

[3] Our MATLAB source codes are publicly available at https://sites.google.com/site/yinqiangzheng/.