Revisiting the P3P Problem

Yaqing Ding[1] Jian Yang[2] Viktor Larsson[1] Carl Olsson[1] Kalle Åström[1]
[1]Lund University [2]Nanjing University of Science and Technology

Abstract

One of the classical multi-view geometry problems is the so called P3P problem, where the absolute pose of a calibrated camera is determined from three 2D-to-3D correspondences. Since these solvers form a critical component of many vision systems (e.g. in localization and Structure-from-Motion), there have been significant effort in developing faster and more stable algorithms. While the current state-of-the-art solvers are both extremely fast and stable, there still exist configurations where they break down.

In this paper we algebraically formulate the problem as finding the intersection of two conics. With this formulation we are able to analytically characterize the real roots of the polynomial system and employ a tailored solution strategy for each problem instance. The result is a fast and stable solver, that is able to correctly solve cases where competing methods might fail. Our experimental evaluation shows that we outperform the current state-of-the-art methods both in terms of speed and success rate.

1. Introduction and Related Work

Registering a new image to a given 3D model is a critical step in many computer vision pipelines, e.g. visual positioning and localization [22], augmented reality [23] or autonomous mapping and navigation [18]. In addition, it has been combined with deep learning to perform learning and geometric optimization end-to-end [4]. The problem is generally solved by establishing a sparse set of 2D-3D point correspondences between the image and the model using feature-based matching [17]. To deal with the potential mismatches, robust estimators based on hypothesis-and-test frameworks such as RANSAC [7] are employed. These methods work by generating multiple candidate models from randomly selected minimal subsets of the data (to reduce the risk of outlier contamination). In the context of the absolute pose problem, i.e. estimating the position and orientation of a camera given a set of 2D-3D point correspondences, the minimal is called *Perspective-Three-Point* (P3P). As the name suggests, the problem is minimal with three point correspondences in the calibrated setting, and has up to four real solutions.

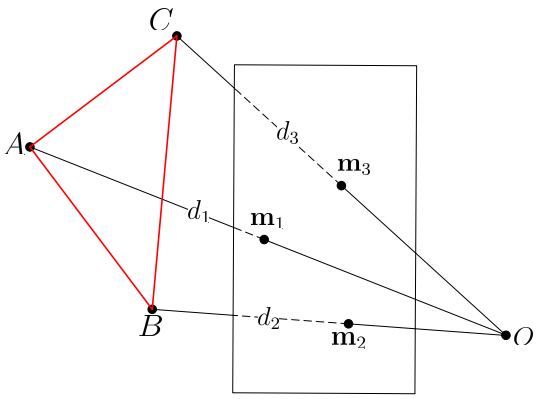

Figure 1. The perspective-three-point problem

The P3P problem has a long history, predating the field of computer vision by a large margin. The geometric problem itself (though not in the context of cameras) was considered as early as 1773 by Lagrange [16] (see [24] for details). In his work Lagrange showed that it had at most four solutions and could be reduced to a quartic polynomial. Almost a century later, in 1841, Grunert [10] revisited the problem and provided a direct solution method. In the early 20th century the problem was also studied in the photogrammetry community, though the main focus was on refinement-based methods instead of solving the problem from scratch (see Haralick [12] for details). Finstenvalder and Scheufele [6] first show that the P3P problem only required to find a root of a cubic polynomial and the roots of two quadratic polynomials. The problem later resurfaced in the computer vision community in the seminal RANSAC paper from Fischler and Bolles [7]. Due to the success of RANSAC-based estimators the problem has since received significant attention.

Based on the degree of the final univariate polynomial, the P3P solutions can be mainly divided into two categories: solving a quartic equation and solving a cubic equation. Most of the modern papers focus on converting the P3P problem into solving a quartic equation. Gao *et al*. [8] used Wu-Ritt's zero decomposition algorithm [27] to give a first

complete analytical solution to the P3P problem. Kneip [15] proposed a direct method for solving the P3P-problem as a computation of the absolute camera position and orientation, which avoids doing the eigenvalue decomposition or singular value decomposition. Ke *et al*. [14] proposed an approach which directly determine the camera's attitude by employing the corresponding geometric constraints. Banno [1] and [19] proposed direct P3P methods estimating the distance in the intermediate coordinate system so that the rotation matrix can be formulated as a linear representation of the distances.

Unlike the quartic equation based methods, the cubic equation based formulation has not been given much attention in the P3P problem literature. Since the work of [6], the cubic formulation has also been used in the work by Grafarend *et al*. [9]. They seeked to reduce (3) to homogenous form and then they use the same technique as [6]. Haralick *et al*. [12] reviewed the major cubic based solutions to the P3P problem and discussed the numerical accuracy. Recently, Persson and Nordberg [21] showed more details on finding the rotation and translation and proposed an efficient algorithm using a single root of a cubic. To the best of our knowledge, the solver by Persson and Nordberg [21] has better numerical accuracy and is faster than previous work.

In this paper we again revisit the P3P problem. We focus on the solution strategy that is based on intersecting two conics, which was also used in recent work [21]. The relative position of two ellipses has been studied in several papers [5, 26]. However, none of them considered the computation of the intersection points. By contrast, We provide a fast and stable solver based on the characterization of the possible solution configurations. Experimentally we show that these extreme cases are the reason for failures and instabilities in previous methods. Finally, leveraging our new understanding we design a novel P3P algorithm that explicitly handles the dangerous cases. The result is a stable P3P solver that as an added benefit is faster than previous approaches.

2. Problem Statement

Consider three 3D points $\mathbf{X}_i \in \mathbb{R}^3, i \in \{1,2,3\}$ in the world coordinate and their corresponding normalized image points $\mathbf{m}_i \in \mathbb{R}^3, |\mathbf{m}_i| = 1, i \in \{1,2,3\}$. The two set of the points are related by the rigid transformation

$$d_i \mathbf{m}_i = \mathbf{R}\mathbf{X}_i + \mathbf{t}, \tag{1}$$

where $d_i \in \mathbb{R}^+$ are some positive numbers. To eliminate the rotation and translation parameters, we first have (taking the notation from Figure 1)

$$\begin{aligned} |AB| &= |\mathbf{X}_1 - \mathbf{X}_2|, \\ |AC| &= |\mathbf{X}_1 - \mathbf{X}_3|, \\ |BC| &= |\mathbf{X}_2 - \mathbf{X}_3|. \end{aligned} \tag{2}$$

Based on the law of cosines from the triangles, and normalize the image points as unit vectors, we have the following constraints

$$\begin{aligned} d_1^2 + d_2^2 - 2d_1 d_2 \mathbf{m}_1^\top \mathbf{m}_2 &= |AB|^2, \\ d_1^2 + d_3^2 - 2d_1 d_3 \mathbf{m}_1^\top \mathbf{m}_3 &= |AC|^2, \\ d_2^2 + d_3^2 - 2d_2 d_3 \mathbf{m}_2^\top \mathbf{m}_3 &= |BC|^2, \end{aligned} \tag{3}$$

where $d_1 = |OA|,\ d_2 = |OB|,\ d_3 = |OC|$. Our aim is to finding the solutions to $\{d_1, d_2, d_3\}$ and recover the rotation and translation. We can assume $d_3 \neq 0$ since otherwise the 3D point coincides with the camera center. We then perform the following change of variables, $x = d_1/d_3$, $y = d_2/d_3$, and divide the first two equations in (3) by the last equation in (3) to eliminate d_3. Then we have the following two quadratic equations in two unknowns x and y,

$$x^2 + (1-a)y^2 - 2m_{12}xy + 2am_{23}y - a = 0, \tag{4}$$

$$x^2 - by^2 - 2m_{13}x + 2bm_{23}y + 1 - b = 0, \tag{5}$$

where

$$\begin{aligned} &a = |AB|^2/|BC|^2,\ b = |AC|^2/|BC|^2, \\ &m_{12} = \mathbf{m}_1^\top \mathbf{m}_2,\ m_{13} = \mathbf{m}_1^\top \mathbf{m}_3,\ m_{23} = \mathbf{m}_2^\top \mathbf{m}_3. \end{aligned} \tag{6}$$

Now the P3P problem is reduced to find the real solutions of the above two quadratic equations. In other words, we need to find the real intersections of two conics.

3. Our Approach

We now present our approach for P3P. We follow a similar strategy as Persson and Nordberg [21] where the main idea is to formulate the problem as intersecting two conics. The two quadratic equation (4) and (5) can be written as the following matrix representations

$$[1, x, y]\mathbf{C}_1[1, x, y]^\top = 0, \tag{7}$$

$$[1, x, y]\mathbf{C}_2[1, x, y]^\top = 0, \tag{8}$$

where $\mathbf{C}_1, \mathbf{C}_2$ are 3×3 matrices. In order to find the intersections, we first construct a 3×3 matrix

$$\mathbf{C} = \mathbf{C}_1 + \sigma \mathbf{C}_2. \tag{9}$$

The intersections are found by building a degenerate conic which also intersects the true solutions. We can calculate the parameter σ such that $\mathbf{C}$ is a degenerate conic. Degenerate conics are characterized by the following proposition.

Proposition 1 (**Degenerate conics** [13])**.**
If the matrix $\mathbf{C}$ *is not of full rank, then the conic is termed* ***degenerate****. Degenerate point conics are either two lines (rank 2) or a repeated line (rank 1), and can be written as*

$$\mathbf{C} = \mathbf{p}\mathbf{q}^\top + \mathbf{q}\mathbf{p}^\top, \tag{10}$$

where $\mathbf{p}, \mathbf{q} \in \mathbb{R}^3$. Once we have constructed the degenerate conic, it can be factorized into (at most) two lines ($\mathbf{p}$ and $\mathbf{q}$), which we can then easily intersect with the original two conics.

3.1. Finding the Degenerate Conic

By Prop. 1, the conic should be rank-deficient, i.e.

$$\begin{aligned} f(\sigma) &= \det(\mathbf{C}) \\ &= \det(\mathbf{C}_1 + \sigma\mathbf{C}_2) = 0, \end{aligned} \tag{11}$$

which results in a cubic equation in σ. Solving (11) we can find the values of σ and obtain the matrix $\mathbf{C}$. Note that any solution to the original equations (belonging to both conics $\mathbf{C}_1$ and $\mathbf{C}_2$) will lie on the degenerate conic $\mathbf{C}$ as well.

For each solution σ to (11), we can obtain a degenerate conic $\mathbf{C}$. As shown in (10), the degenerate conic is composed of two lines $\mathbf{p}$ and $\mathbf{q}$. Our goal is now to factorize the conic and recover the two lines.

3.2. Extracting the Lines Directly

We first show a direct method to find the lines. Assuming we have obtained a degenerate conic $\mathbf{C}$, which can be written as

$$\mathbf{C} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{12} & c_{22} & c_{23} \\ c_{13} & c_{23} & c_{33} \end{bmatrix}. \tag{12}$$

Since $\mathbf{C} = \mathbf{p}\mathbf{q}^\top + \mathbf{q}\mathbf{p}^\top$, let $\mathbf{p} = [p_1, p_2, p_3]^\top$, $\mathbf{q} = [q_1, q_2, q_3]^\top$, the matrix $\mathbf{C}$ can also be written as

$$\mathbf{C} = \begin{bmatrix} 2p_1q_1 & p_1q_2 + p_2q_1 & p_1q_3 + p_3q_1 \\ p_1q_2 + p_2q_1 & 2p_2q_2 & p_2q_3 + p_3q_2 \\ p_1q_3 + p_3q_1 & p_2q_3 + p_3q_2 & 2p_3q_3 \end{bmatrix}. \tag{13}$$

Suppose $p_1q_1 \neq 0$, let $\tilde{p}_2 = p_2/p_1$, $\tilde{q}_2 = q_2/q_1$, $\tilde{p}_3 = p_3/p_1$, $\tilde{q}_3 = q_3/q_1$. We have

$$\tilde{p}_2 + \tilde{q}_2 = 2c_{12}/c_{11}, \tag{14}$$

$$\tilde{p}_2\tilde{q}_2 = c_{22}/c_{11}, \tag{15}$$

$$\tilde{p}_3 + \tilde{q}_3 = 2c_{13}/c_{11}, \tag{16}$$

$$\tilde{p}_2\tilde{q}_3 + \tilde{p}_3\tilde{q}_2 = 2c_{23}/c_{11}, \tag{17}$$

Based on (14) and (15) we may obtain the solution to $\{\tilde{p}_2, \tilde{q}_2\}$. Substituting this solution into (17) and combining (16) we can compute the solution to $\{\tilde{p}_3, \tilde{q}_3\}$. In this case, we can obtain a pair of lines $[1, \tilde{p}_2, \tilde{p}_3]$ and $[1, \tilde{q}_2, \tilde{q}_3]$. To avoid the case $p_1q_1 = 0$, we can first find the absolute maximum of the diagonal elements of $\mathbf{C}$, then the pair of lines can be computed more stably.

3.3. Finding the intersection of two lines

On the other hand, we can first recover the intersection point $\mathbf{v} = \mathbf{p} \times \mathbf{q}$ which can then be used to extract the lines from $\mathbf{C}$. For the finding intersection point $\mathbf{v}$, we present two different methods.

Method 1: Null Space. From (10) it is clear that $\mathbf{v}$ lies in the nullspace of $\mathbf{C}$. Taking any nullvector $\mathbf{n}$ of $\mathbf{C}$, we know that $\mathbf{v} = \alpha\mathbf{n}$. We must now find $\alpha \in \mathbb{R}$ such that the scale of $\mathbf{v}$ is consistent with $\mathbf{C}$ (and $\mathbf{p}, \mathbf{q}$). Since $\mathbf{v} = \mathbf{p} \times \mathbf{q}$, we have

$$\mathbf{v} = [p_2q_3 - p_3q_2,\ p_3q_1 - p_1q_3,\ p_1q_2 - p_2q_1] \tag{18}$$

Combining (12), (12) and (18), we can derive the relationship between the norm of $\mathbf{v}$ and the elements of the matrix $\mathbf{C}$

$$\|\mathbf{v}\|^2 = c_{12}^2 + c_{13}^2 + c_{23}^2 - c_{11}c_{22} - c_{11}c_{33} - c_{22}c_{33}. \tag{19}$$

Hence, rescaling $\mathbf{n}$ appropriately the intersection point $\mathbf{v}$ with in the correct scale is obtained.

Method 2: Adjoint matrix: On the other hand, the adjoint matrix of $\mathbf{C}$ should have the following formulation

$$-\mathbf{C}^* = \mathbf{v}\mathbf{v}^\top, \tag{20}$$

Proof. Eq. (20) can simply be proved by using (13) to formulate $-\mathbf{C}^*$

$$\begin{bmatrix} (p_2q_3 - p_3q_2)^2 & (p_3q_1 - p_1q_3)(p_2q_3 - p_3q_2) & (p_1q_2 - p_2q_1)(p_2q_3 - p_3q_2) \\ (p_3q_1 - p_1q_3)(p_2q_3 - p_3q_2) & (p_1q_3 - p_3q_1)^2 & (p_1q_2 - p_2q_1)(p_3q_1 - p_1q_3) \\ (p_1q_2 - p_2q_1)(p_2q_3 - p_3q_2) & (p_1q_2 - p_2q_1)(p_3q_1 - p_1q_3) & (p_1q_2 - p_2q_1)^2 \end{bmatrix}$$

whose entries are identical to the entries of $\mathbf{v}\mathbf{v}^\top$. Given a matrix $\mathbf{C}$, we can first obtain $-\mathbf{C}^*$. Then, to avoid zero elements, we can find the maximum of its diagonal elements and the corresponding column. The intersection point $\mathbf{v}$ can be extracted from the column divided by the squared root of the max diagonal element.

Recovering the lines: Once we obtain the intersection point $\mathbf{v}$, the skew-symmetric matrix of $\mathbf{v}$ is given by

$$[\mathbf{v}]_\times = \mathbf{p}\mathbf{q}^\top - \mathbf{q}\mathbf{p}^\top, \tag{21}$$

which can be proved by expanding the terms on both sides of the above formulation. Then we define a new matrix

$$\mathbf{D} = \mathbf{C} + [\mathbf{v}]_\times. \tag{22}$$

Combining (22) and (10) we have $\mathbf{D} = 2\mathbf{p}\mathbf{q}^\top$. The pair of lines $\{\mathbf{p}, \mathbf{q}\}$ can be found from one row and the corresponding column of $\mathbf{D}$.

Rank-1 case: If the degenerate conic $\mathbf{C}$ includes a pair of repeated lines, the matrix $\mathbf{C}$ will be rank-1. In this case, the repeated lines can be recovered directly from one row or column.

3.4. Extracting the Solutions

The lines recovered from the degenerate conic pass through the original solutions to (7)-(8), thus to recover

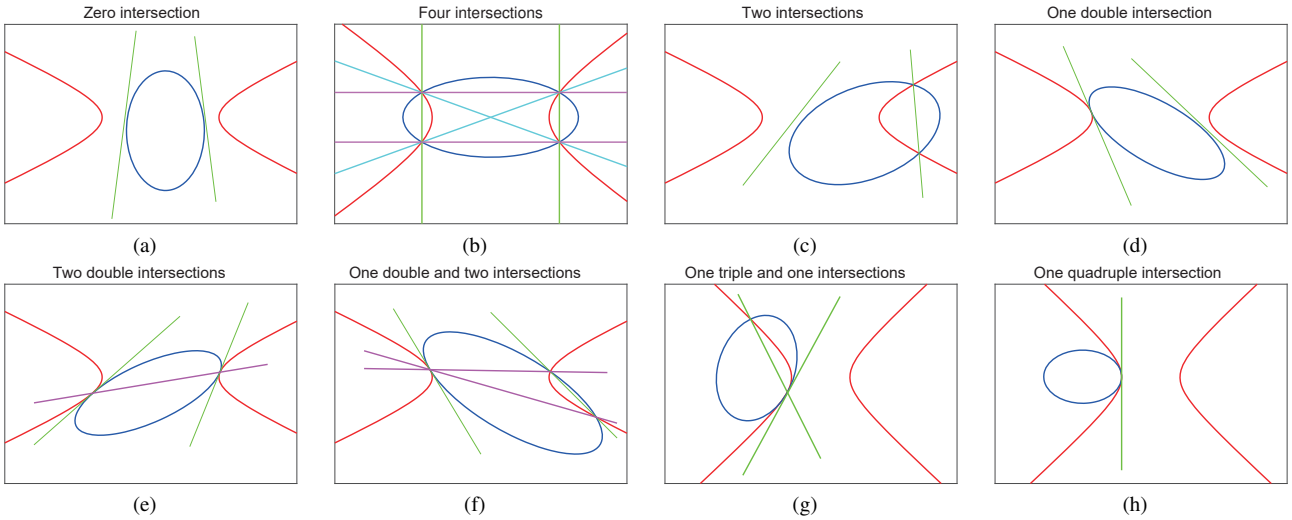

Figure 2. Illustration of eight possible cases for the relative position of a hyperbola and an ellipse. The pair of lines with different colors corresponds to different cubic roots. (a)-(c): general cases that the two conics have zero, four and two real intersections, respectively. (d)-(f): critical cases that the two conics are tangent to each other, with one double, two double, and one double and two intersections, respectively. (g)-(h): the two conics are osculating (the two conics have the same curvature at the tangent point) to each other with one triple and one intersections and one quadruple intersection, respectively.

them we simply need to compute the intersection between the lines and either of the two conics. Assuming the equation of the first line in the pair is

$$p_1 + p_2 x + p_3 y = 0 \tag{23}$$

and substituting (23) into (5) gives a quadratic equation in x, for which there are up to two solutions. Note that, we are only interested in the positive real solutions. Using (23) we can obtain the corresponding y. Since $x = d_1/d_3$, we have $d_1 = x d_3$. Substituting the formulation into (3) we obtain a quadratic equation in d_3

$$(x^2 - 2m_{13}x + 1)d_3^2 = |AC|^2. \tag{24}$$

Note that $m_{13} < 1$, hence the solution to d_3 is given by $d_3 = |AC|/\sqrt{x^2 - 2m_{13}x + 1}$. In this case, we can obtain the values of d_1, d_2, d_3. There will be four possible solutions to d_i since there is a pair of lines. Once d_i are known, we can perform Gauss-Newton optimization on the sum of squares of (3) to refine the solutions. Similar refinements have been used in several P3P algorithms [14, 19, 21].

Finding rotation and translation: For each $\{d_1, d_2, d_3\}$, we first use (1) to eliminate the translation and obtain two set of equations

$$\begin{aligned} d_1\mathbf{m}_1 - d_2\mathbf{m}_2 &= \mathbf{R}(\mathbf{X}_1 - \mathbf{X}_2), \\ d_3\mathbf{m}_3 - d_1\mathbf{m}_1 &= \mathbf{R}(\mathbf{X}_3 - \mathbf{X}_1). \end{aligned} \tag{25}$$

To find another non-coplanar vector correspondence, we can use the normal of the plane defined by the three 3D points and image points as in [21] (See Figure 3). The normal vector also satisfies

$$\mathbf{n}_b = \mathbf{R}\mathbf{n}_a. \tag{26}$$

with

$$\begin{aligned} \mathbf{n}_b &= (d_1\mathbf{m}_1 - d_2\mathbf{m}_2) \times (d_3\mathbf{m}_3 - d_1\mathbf{m}_1), \\ \mathbf{n}_a &= (\mathbf{X}_1 - \mathbf{X}_2) \times (\mathbf{X}_3 - \mathbf{X}_1). \end{aligned} \tag{27}$$

Combining (25) and (26), we have

$$\begin{aligned} \mathbf{R} &= \mathbf{B}\mathbf{A}^{-1}, \\ \mathbf{B} &= [d_1\mathbf{m}_1 - d_2\mathbf{m}_2, d_3\mathbf{m}_3 - d_1\mathbf{m}_1, \mathbf{n}_b], \\ \mathbf{A} &= [\mathbf{X}_1 - \mathbf{X}_2, \mathbf{X}_3 - \mathbf{X}_1, \mathbf{n}_a]. \end{aligned} \tag{28}$$

After the rotation is recovered, the translation can be found using (1).

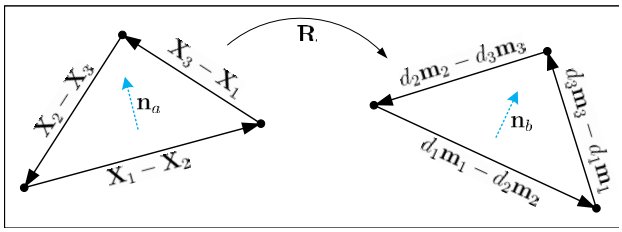

Figure 3. Rotation from vector correspondences.

3.5. Analysis of the Possible Solution Configurations

Above we presented a general scheme for solving the P3P problem. The algorithm has two main steps

- Solving for σ to build the degenerate conic (9).
- Factorizing the degenerate conic C into (up to) two lines that we substitute into the equations (4)-(5).

Case	Roots of the cubic		Number of lines		Intersections of each pair of lines	
					real	imaginary
a	simple real	3	simple real	1 pair	0	4
	imaginary	0	imaginary	2 pairs	0	4
b	simple real	3	simple real	3 pairs	4	0
	imaginary	0	imaginary	0	-	-
c	simple real	1	simple real	1 pair	2	2
	imaginary	2	imaginary	2 pairs	2	2
d	simple real	1	simple real	1 pair	1D	2
	double real	1	imaginary	1 pair (twice)	1D	2
e	simple real	1	simple real	1 pair	2D	0
	double real	1	repeated real	1 pair (twice)	2D	0
f	simple real	1	simple real	1 pair	1D + 2	0
	double real	1	simple real	1 pair (twice)	1D + 2	0
g	Triple root	1	simple real	1 pair (thrice)	1T + 1	0
h	Triple root	1	repeated real	1 pair (thrice)	1Q	0

Table 1. The relationship among the roots of the cubic equation, the number of the lines from the degenerate conic and the intersections of the two conics. 1D, 1T and 1Q denote one double, one triple and one quadruple intersection, respectively.

Next, inspired by [5, 26], we analyze the configurations of real solutions that are possible, and then leverage this to build a robust algorithm.

The general solution to the cubic equation in (11) may have four cases: three real roots, one real and two complex roots, one real and one double root, one real triple root. These four possibilities correspond to different cases of the lines and the intersections. In our case, the first conic (4) is indefinite, and the second one (5) is a hyperbola ($b > 0$). Without loss of generality, we assume the first one is an ellipse, and show the possible relative positions of the two conics in Figure 2. To briefly present our method, we use the Figure 2b as an example. The two conics have four real intersections, and the characteristic cubic equation has three real roots. Each real root corresponds to a pair of real lines. Each pair of real lines intersects any of the two conics at the four real intersections (Figure 2b). Similar situations also exist in other cases. The relationship among the roots of the cubic equation, the number of the lines and the intersections is shown in Table 1. Assuming the cubic equation (11) can be written as

$$\sigma^3 + \kappa_2\sigma^2 + \kappa_1\sigma + \kappa_0 = 0. \tag{29}$$

The change of variable $\sigma = \gamma - \frac{\kappa_2}{3}$ gives a depressed cubic [3] in γ which has no term in γ^2

$$\gamma^3 + \alpha\gamma + \beta = 0, \tag{30}$$

with

$$\begin{aligned} \alpha &= (3\kappa_1 - \kappa_2^2)/3, \\ \beta &= (2\kappa_2^3 - 9\kappa_2\kappa_1 + 27\kappa_0)/27. \end{aligned} \tag{31}$$

The discriminant of (30) is

$$\Delta = -(4\alpha^3 + 27\beta^2). \tag{32}$$

- If $\Delta > 0$, (30) has three distinct real roots which corresponds to case (a) and (b). For case (a), since all the real roots don't correspond to any real intersections, we can pick any of the three roots. The root can be found using the trigonometric solution [28]. This root may correspond to one pair of real lines or no real lines. If the picked real root gives one pair of real lines, we can skip the second line after verifying there are no real intersections for the first line. For case (b), any of the three roots will result in one pair of real lines and four real intersections.

- If $\Delta < 0$, (30) has one real root and two complex conjugate roots which corresponds to case (c). The real root can be found using Cardano's formula [3], and we can obtain one pair of real lines. If the first line in the pair has real intersections with the conic, we can skip the second line. Otherwise, we need to check the second line if there are no real intersections for the first line.

- If $\Delta = 0$, and $\alpha \neq 0$, then (30) has a simple real root $\gamma_1 = \frac{3\beta}{\alpha}$ and a double real root $\gamma_{2,3} = -\frac{3\beta}{2\alpha}$ which corresponds

to case (d), (e) and (f). For case (d), we can see that the double real root results in imaginary lines. Hence, we can use the simple real root for this situation.

- If $\Delta = 0$, and $\alpha = 0$, then $\alpha = \beta = 0$. In this instance, Eq. (30) has a triple root $\gamma_{1,2,3} = 0$ which corresponds to case (g) and (h). The real root of the cubic gives a pair of real lines, and the real intersections can be easily found.

Based on the above analysis and Table 1 we can notice that the real intersections can be recovered using any pair of the real lines, and any real root of the cubic equation corresponds to a pair of real lines except for case (d). We only need to avoid using the double real root when $\Delta = 0$, and $\alpha \neq 0$. On the other hand, if $\Delta = 0$, there must be double intersections between the pair of lines and the conic. To avoid duplicate solutions, we need to check the intersections between the line and the conic. We summarize the complete procedure as Algorithm 1.

Algorithm 1:

Input: 3D points $\mathbf{X}_i$, image points $\mathbf{m}_i$, $i = 1, 2, 3$
Output: Rotation $\mathbf{R}$, translation $\mathbf{t}$

```
Normalize the image points m_i = m_i/|m_i|
Compute a, b, m_12, m_13, m_23 based on (6)
Construct matrix C_1, C_2, and obtain the cubic equation in σ using (11)
Compute the discriminant using (32)
if Δ > 0 then
    Compute one real root of γ using the trigonometric solution
    Compute the matrix C
    Compute the intersection point v of the lines using (20)
    Extract the lines using (22)
    Compute the intersection between the first line and one of the conic using (23)(5)
    if Has real solutions then
        Do the second line
    else if No real solutions then
        Skip the second line
else if Δ < 0 then
    Compute the real root of γ using Cardano's formula
    ··· // The same as Δ > 0
    Do the first line
    if Has real solutions then
        Skip the second line
    else if No real solutions then
        Do the second line
else if Δ = 0, α ≠ 0 then
    Choose γ = 3β/α, compute the intersections using both lines
else if Δ = 0, α = 0 then
    γ = 0, compute the intersections using both lines
Compute d_3 using (24)
Gauss-Newton root polishing
Compute R, t using (28)
```

4. Experimental Results

In this section, we compare the numerical stability and efficiency of the proposed method with the state-of-the-art methods. All the solvers are implemented in C++ and the experiments are run on a desktop computer with an Intel Core i7-9700 3.0GHz CPU. We compare with the solvers whose C++ implementations are publicly available: the P3P solver by Ke *et al*. [14][1] and the solver by Kneip *et al*. [15][2] which result in solving a quartic equation. We also compare with the solver by Nakano [19]. Unfortunately, only the MATLAB implementation of this solver is available online[3], so we re-implement this solver in C++ using the Eigen libarary [11]. In addition, we compare with the state-of-the-art P3P solver which solves a cubic equation [20,21][4]. Note that, we have proposed several methods to extract the lines. In practice, we find that Method 2 is more stable. To save space, we only show the results based on Method 2, and more results are shown in the supplementary material. The experiments are tested on synthetic data with ground truth. We omit the experiments with noisy data and real data, since the noisy case can be considered as noise free data with different unknown ground truth. The results by the solvers are only affected by the numerical stability (+/- some numerical instabilities).

To make a fair comparison, the synthetic data used in this paper is generated based on [21], which strains the solvers and can find the failure cases. In short, the unit quaternion is drawn from a isotropic Gaussian distribution and then converted into the ground truth rotation matrix $\mathbf{R}_{gt}$. The ground truth translation $\mathbf{t}_{gt}$ is generated from the standard normal distribution. The normalized image points are generated by a uniform sampling of 2D coordinates in the range [-1,1], and then the corresponding 3D point is calculated by $\mathbf{X}_i = \mathbf{R}_{gt}^{\top}(d_i\mathbf{m}_i - \mathbf{t}_{gt})$ with a uniform random positive depth $d \in [0.1, 10]$. The rotation and translation errors are defined by $\xi_R = \|\mathbf{R}_{gt} - \mathbf{R}_{est}\|_{L1}$ and $\xi_t = \|\mathbf{t}_{gt} - \mathbf{t}_{est}\|_{L1}$, respectively. The cases where three points are collinear are removed since such cases doesn't give enough constraints to find the motion parameters.

[1] https://opencv.org/
[2] https://www.laurentkneip.com/software/
[3] https://github.com/g9nkn/p3p_problem
[4] https://github.com/midjji/lambdatwist-p3p

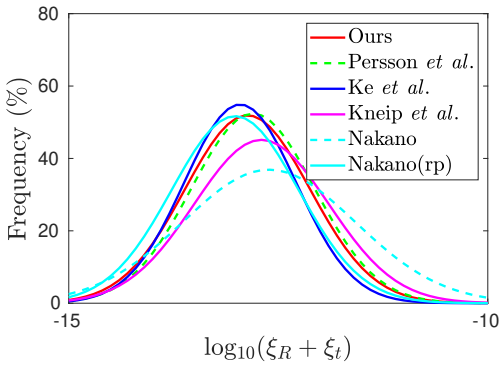

Figure 4. Gaussian kernel smoothed histograms of the sum of the rotation and translation errors for 100,000 runs on noise-free data. We measure the L1-norm of the difference between the estimation and the ground truth.

Method	Mean	Median	Max
Ours	**3.5e-12**	1.4e-13	**2.3e-8**
Persson *et al.* [21]	4.2e-12	1.6e-13	4.3e-8
Ke *et al.* [14]	3.5e-7	1.1e-13	0.011
Kneip *et al.* [15]	2.5e-6	2.5e-13	0.070
Nakano [19]	4.4e-7	3.0e-13	0.002
Nakano(rp) [19]	1.7e-9	**9.4e-14**	1.8e-5

Table 2. The mean, median and max values of the errors. The best results are marked bold.

4.1. Numerical Stability

In Figure 4 we show the Gaussian kernel smoothed histograms of the sum of the rotation and translation errors under noise-free data with 100,000 runs. We can see that all the methods are numerically stable. The original MATLAB implementation by Nakano [19] includes an optional root polishing by Gauss-Newton method, so we evaluate both the two cases (with and without root polishing, Nakano(rp) denotes the solver with root polishing). It seems that the solver with root polishing by Nakano [19] gives more distributions on small errors. In addition, the mean, median and max errors are shown in Table 2. We can find that the proposed solver achieves the best performance for mean and max errors, while the solver by Nakano *et al.* [19] with root polishing has the best performance for median error. Note that, the quartic based solvers contain failure cases which have been removed for this test.

4.2. Solution Discussion

In Table 3 we show the solution comparison with the current state-of-the-art. We generated 10^7 scenes in this experiment to strain the solvers and show more possible failure cases. Valid solutions represent the total number of solutions returned by the solver. Unique solutions represent the number of correct solutions (with valid rotation matrix) by removing the duplicates. If two solutions from one trial satisfy $\xi_R + \xi_t < 10^{-5}$, then there are defined as duplicates. For each trial, if there is at least one unique solution, then we say that there is good solution for this trial. The condition to define a returned solution as ground truth is given by $\xi_R + \xi_t < 10^{-6}$. We can see that our solver outperforms the existing methods. For almost all the trials, we can find good solutions and ground truth without duplicates or incorrect solutions. The quartic equation based solvers by Ke *et al.* [14] and Kneip *et al.* [15] have many duplicates and incorrect solutions, the reason is that those solvers use all the four roots (omit the imaginary part) from the quartic equation to find possible estimations. It can improve the possibility to find the ground truth, but it will reduce the efficiency since each hypothesis needs to be evaluated within RANSAC in practice. The solver by Nakano [19] uses a threshold (10^{-8}) in the imaginary part to eliminate unnecessary complex roots, but the results are not as good as ours. The duplicates and incorrect solutions of Ke *et al.* and Kneip *et al.* can also be reduced by using such threshold to remove incorrect roots of the quartic equation.

4.3. Running Times

In Table 4 we show the mean, median, min and max running times in nanosecond of the proposed and the state-of-the-art solvers. The timings, averaged over 10^7 trials with 100 times each, are reported. We can see that the cubic based solvers are more efficient than the quartic based solvers. The proposed solver is about 15.4% faster than the current state-of-the-art solver. The speedup is mainly due to two reasons: (i) based on the analysis of the relative position and the discriminant, we can quickly choose the suitable root of the cubic. We can avoid unnecessary computation according to the sign of the discriminant. (ii) The method to recover the lines from a degenerate conic is more efficient than [21]. It seems that our C++ implementation of [19] doesn't provide good performance, since the results of MATLAB implementation in [19] showed that the solver is slower than [21] but slightly faster than [14]. We think the reason is that C++ and MATLAB used different matrix computation libraries.

4.4. Analysis of Failure Cases

We have found that the discriminant of the no solution cases by Persson and Nordberg [21] is very close to zero. It's understandable, since zero discriminant corresponds to the critical cases (d)-(h) in Figure 2. Due to the float point round-off error and numerical instability, it may be difficult to recover the motion parameters from such cases. We also find that most of the failure cases are case (d) and (f), where

Method	Ours	Persson *et al*. [21]	Ke *et al*. [14]	Kneip *et al*. [15]	Nakano [19]	Nakano(rp) [19]
Valid solutions	16825700	16825700	17389005	24159054	16823126	16826586
Unique solutions	16825700	16825686	16850758	16827917	16815718	16826042
Duplicates	0	0	163038	3038	0	0
Good solution	10000000	9999989	9999622	9999663	9996957	9999249
No solution	0	11	378	337	3043	751
Ground truth	9999993	9999978	9997345	9991078	9985342	9998727
Incorrect	0	2	375209	7328099	7408	544

Table 3. Solution comparison with the current state-of-the-art solvers

Timing (ns)	Ours	Persson *et al*. [21]	Ke *et al*. [14]	Kneip *et al*. [15]	Nakano [19]	Nakano(rp) [19]
Mean	**225.8**	260.6	387.1	667.2	591.3	702.0
Median	**225.7**	260.7	387.1	667.3	591.0	702.1
Min	**224.0**	258.2	384.4	664.1	588.1	699.4
Max	**231.6**	263.7	393.5	670.7	611.8	705.5
Speed up	**1.154**	1.0	0.6732	0.3906	0.4407	0.3712

Table 4. Running times comparison averaged over 10^7 trials with 100 times each.

case (e), (g) and (h) rarely happen. A detailed analysis of how such cases occur is interesting, although characterization of these is future work.

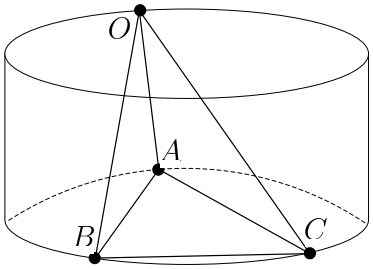

Figure 5. The danger cylinder is defined as a circular cylinder circumscribing points A, B, C with axis normal to the plane ABC.

4.5. Relationship to the Danger Cylinder

The three 3D points A, B, C defines a cylinder with the generatrix parallel to the normal of the plane ABC. This cylinder is known as the *danger cylinder* in the literature. It has been shown in [25] that the solution of the P3P problem is unstable if the optical center O lies on the surface of this danger cylinder. The situation is shown in the following figure. We find that the danger cylinder will result in $\Delta = 0$ which corresponds to the critical cases in Figure 2. This property has been found by generating synthetic data that satisfies the danger cylinder. Due to the lack of space we included the details for the danger cylinder in the supplementary material.

4.6. Limitations

Our approach is based on studying the real intersections of two conics. However, P3P problem is special since the solutions should be positive real numbers. There might be more constraints on the cubic equation. Unfortunately, we have not found a good way to do this. Note that, Eq. (4) and (5) can be solved using Homotopy continuation [2], which is good at solving large-scale squared system. For P3P problem, we found that it is not as efficient as current solvers which are essentially closed-form.

5. Conclusion

In this paper we have revisited the P3P problem and in particular investigated the solution strategy that is based on intersecting two conics, similar to what was used by Persson and Nordberg [21]. By analyzing the set of possible solutions and explicitly enumerating the corner cases, we are able to design a P3P algorithm that is fast and stable. In experiments, we show that the solver is both more robust and faster compared to previous methods. The source code is available at https://github.com/yaqding/P3P

Acknowledgments. This work was supported by the National Authorities, under GA 876019, the strategic research projects ELLIIT, the Swedish Foundation for Strategic Research project, Semantic Mapping and Visual Navigation for Smart Robots (grant no. RIT15-0038), the Swedish Research Council (grant no. 2018-05375), and by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. In addition, this work was funded in part by Jiangsu Funding Program for Excellent Postdoctoral Talent, and China Postdoctoral Science Foundation (No: 2022M711636).

References

[1] Atsuhiko Banno. A p3p problem solver representing all parameters as a linear combination. *Image and Vision Computing*, 2018. 2

[2] Paul Breiding and Sascha Timme. HomotopyContinuation.jl: A Package for Homotopy Continuation in Julia. In *International Congress on Mathematical Software*, 2018. 8

[3] Girolamo Cardano, T Richard Witmer, and Oystein Ore. *The rules of algebra: Ars Magna*, volume 685. Courier Corporation, 2007. 5

[4] Bo Chen, Alvaro Parra, Jiewei Cao, Nan Li, and Tat-Jun Chin. End-to-end learnable geometric vision by backpropagating pnp optimization. In *Computer Vision and Pattern Recognition (CVPR)*, 2020. 1

[5] Fernando Etayo, Laureano Gonzalez-Vega, and Natalia del Rio. A new approach to characterizing the relative position of two ellipses depending on one parameter. *Computer aided geometric design*, 2006. 2, 5

[6] S Finstenvalder and W Scheufele. Das rückwärtseinschneiden im raum. *Zum 75-Geburtstage Verlag Herbert Wichmann*, pages 86–100, 1937. 1, 2

[7] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 1

[8] Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng. Complete solution classification for the perspective-three-point problem. *IEEE transactions on pattern analysis and machine intelligence*, 2003. 1

[9] EW Grafarend, P Lohse, and B Schaffrin. Dreidimensionaler rückwärtsschnitt. *Zeitschrift für Vermessungswesen*, 114(2):61–67, 1989. 2

[10] Johann August Grunert. Das pothenotische problem in erweiterter gestalt nebst bber seine anwendungen in der geodasie. *Grunerts Archiv fur Mathematik und Physik*, pages 238–248, 1841. 1

[11] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010. 6

[12] Robert M Haralick, Chung-nan Lee, Kars Ottenburg, and Michael Nölle. Analysis and solutions of the three point perspective pose estimation problem. In *Computer Vision and Pattern Recognition (CVPR)*, 1991. 1, 2

[13] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 2

[14] Tong Ke and Stergios I Roumeliotis. An efficient algebraic solution to the perspective-three-point problem. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7225–7233, 2017. 2, 4, 6, 7, 8

[15] Laurent Kneip, Davide Scaramuzza, and Roland Siegwart. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *CVPR 2011*, pages 2969–2976. IEEE, 2011. 2, 6, 7, 8

[16] Joseph Louis Lagrange. *Solutions analytiques de quelques problemes sur les pyramides triangulaires*. Académie royale des sciences et belles lettres, 1773. 1

[17] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 1

[18] Raul Mur-Artal and Juan D Tardós. ORB-SLAM2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics*, 33(5):1255–1262, 2017. 1

[19] Gaku Nakano. A simple direct solution to the perspective-three-point problem. In *BMVC*, page 26, 2019. 2, 4, 6, 7, 8

[20] Mikael Persson. *Visual Odometryin Principle and Practice*. PhD thesis, Linköping University Electronic Press, 2022. 6

[21] Mikael Persson and Klas Nordberg. Lambda twist: An accurate fast robust perspective three point (p3p) solver. In *Proceedings of the European conference on computer vision (ECCV)*, pages 318–332, 2018. 2, 4, 6, 7, 8

[22] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *CVPR*, 2019. 1

[23] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Efficient & effective prioritized matching for large-scale image-based localization. *IEEE transactions on pattern analysis and machine intelligence*, 39(9):1744–1756, 2016. 1

[24] Peter Sturm. A historical survey of geometric computer vision. In *International Conference on Computer Analysis of Images and Patterns*, pages 1–8. Springer, 2011. 1

[25] EH Thompson. Space resection: Failure cases. *The Photogrammetric Record*, 5(27):201–207, 1966. 8

[26] Wenping Wang, Jiaye Wang, and Myung-Soo Kim. An algebraic condition for the separation of two ellipsoids. *Computer aided geometric design*, 2001. 2, 5

[27] Wu Wen-Tsun. Basic principles of mechanical theorem proving in elementary geometries. *Journal of automated Reasoning*, 1986. 1

[28] Dan Zwillinger. *CRC standard mathematical tables and formulas*. chapman and hall/CRC, 2018. 5