An Efficient Algebraic Solution to the Perspective-Three-Point Problem

Tong Ke
University of Minnesota
Minneapolis, USA
kexxx069@cs.umn.edu

Stergios I. Roumeliotis
University of Minnesota
Minneapolis, USA
stergios@cs.umn.edu *

Abstract

In this work, we present an algebraic solution to the classical perspective-3-point (P3P) problem for determining the position and attitude of a camera from observations of three known reference points. In contrast to previous approaches, we first directly determine the camera's attitude by employing the corresponding geometric constraints to formulate a system of trigonometric equations. This is then efficiently solved, following an algebraic approach, to determine the unknown rotation matrix and subsequently the camera's position. As compared to recent alternatives, our method avoids computing unnecessary (and potentially numerically unstable) intermediate results, and thus achieves higher numerical accuracy and robustness at a lower computational cost. These benefits are validated through extensive Monte-Carlo simulations for both nominal and close-to-singular geometric configurations.

1. Introduction

The Perspective-n-Point (PnP) is the problem of determining the 3D position and orientation (pose) of a camera from observations of known point features. The PnP is typically formulated and solved linearly by employing lifting (*e.g.*, [1]), or as a nonlinear least-squares problem minimized iteratively (*e.g.*, [9]) or directly (*e.g.*, [11]). The minimal case of the PnP (for n=3) is often used in practice, in conjunction with RANSAC, for removing outliers [5].

The first solution to the P3P problem was given by Grunert [8] in 1841. Since then, several methods have been introduced, some of which [8, 4, 19, 5, 17, 7] were reviewed and compared, in terms of numerical accuracy, by Haralick *et al*. [10]. Common to these algorithms is that they employ the law of cosines to formulate a system of three quadratic equations in the features' distances from the camera. They differ, however, in the elimination process followed for arriving at a univariate polynomial. Later on, Quan and Lan [22] and more recently Gao *et al*. [6] employed the same formulation but instead used the Sylvester resultant [3] and Wu-Ritz's zero-decomposition method [24], respectively, to solve the resulting system of equations, and, in the case of [6], to determine the number of real solutions. Regardless of the approach followed, once the feature's distances have been computed, finding the camera's orientation, expressed as a unit quaternion [12] or a rotation matrix [13], often requires computing the eigenvectors of a 4×4 matrix (*e.g.*, [22]) or performing singular value decomposition (SVD) of a 3×3 matrix (*e.g.*, [6]), respectively, both of which are time-consuming. Furthermore, numerical error propagation from the computed distances to the rotation matrix significantly reduces the accuracy of the computed pose estimates.

To the best of our knowledge, the first method[1] that does not employ the law of cosines in its P3P problem formulation is that of Kneip *et al*. [15], and later on that of Masselli and Zell [18]. Specifically, [15] and [18] follow a geometric approach for avoiding computing the features' distances and instead directly solve for the camera's pose. In both cases, however, several intermediate terms (*e.g.*, tangents and cotangents of certain angles) need to be computed, which negatively affect the speed and numerical precision of the resulting algorithms.

Similar to [15] and [18], our proposed approach does not require first computing the features' distances. Differently though, in our derivation, we first eliminate the camera's position and the features' distances to result into a system of three equations involving *only the camera's orientation*. Then, we follow an algebraic process for successively eliminating two of the unknown 3-dof and arriving into a quartic polynomial. Our algorithm (summarized in Alg. 1) requires fewer operations and involves simpler and numerically more stable expressions, as compared to either [15] or [18], and thus performs better in terms of efficiency, ac-

*This work was supported by the National Science Foundation (IIS-1328722).

[1] Nister and Stewenius [20] earlier also followed a geometric approach for solving the *generalized* P3P resulting into an octic univariate polynomial whose odd monomials vanish for the case of the central P3P.

curacy, and robustness. Specifically, the main advantages of our approach are:

- Our algorithm's implementation takes about 33% of the time required by the current state of the art [15]. [2]
- Our method achieves better accuracy than [15, 18] under nominal conditions. Moreover, we are able to further improve the numerical precision by applying root polishing to the solutions of the quartic polynomial while remaining significantly faster than [15, 18].
- Our algorithm is more robust than [15, 18] when considering close-to-singular configurations (the three points are almost collinear or very close to each other).

The remaining of this paper is structured as follows. Section 2 presents the definition of the P3P problem, as well as our derivations for estimating first the orientation and then the position of the camera. In Section 3, we assess the performance of our approach against [15] , [18], and [6] in simulation for both nominal and singular configurations. Finally, we conclude our work in Section 4.

2. Problem Formulation and Solution

2.1. Problem Definition

Given the positions, ${}^G\mathbf{p}_i$, of three known features f_i, $i = 1, 2, 3$, with respect to a reference frame $\{G\}$, and the corresponding unit-vector, bearing measurements, ${}^C\mathbf{b}_i$, $i = 1, 2, 3$, our objective is to estimate the position, ${}^G\mathbf{p}_C$, and orientation, *i.e.*, the rotation matrix ${}^G_C\mathbf{C}$, of the camera $\{C\}$.

2.2. Solving for the orientation

From the geometry of the problem (see Fig. 1), we have (for $i = 1, 2, 3$):

$$ {}^G\mathbf{p}_i = {}^G\mathbf{p}_C + d_i {}^G_C\mathbf{C} {}^C\mathbf{b}_i \tag{1} $$

where $d_i \triangleq \|{}^G\mathbf{p}_C - {}^G\mathbf{p}_i\|$ is the distance between the camera and the feature f_i.

In order to eliminate the unknown camera position, ${}^G\mathbf{p}_C$, and feature distance, d_i, $i = 1, 2, 3$, we subtract pairwise the three equations corresponding to (1) for $(i, j) = (1, 2)$, $(1, 3)$, and $(2, 3)$, and project them on the vector

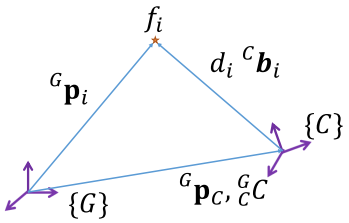

Figure 1. The camera $\{C\}$, whose position, ${}^G\mathbf{p}_C$, and orientation, ${}^G_C\mathbf{C}$, we seek to determine, observes unit-vector bearing measurement ${}^C\mathbf{b}_i$ of a feature f_i, whose position, ${}^G\mathbf{p}_i$, is known.

${}^G_C\mathbf{C}({}^C\mathbf{b}_i \times {}^C\mathbf{b}_j)$ to yield the following system of 3 equations in the unknown rotation ${}^G_C\mathbf{C}$:

$$ ({}^G\mathbf{p}_1 - {}^G\mathbf{p}_2)^T {}^G_C\mathbf{C}({}^C\mathbf{b}_1 \times {}^C\mathbf{b}_2) = 0 \tag{2} $$
$$ ({}^G\mathbf{p}_1 - {}^G\mathbf{p}_3)^T {}^G_C\mathbf{C}({}^C\mathbf{b}_1 \times {}^C\mathbf{b}_3) = 0 \tag{3} $$
$$ ({}^G\mathbf{p}_2 - {}^G\mathbf{p}_3)^T {}^G_C\mathbf{C}({}^C\mathbf{b}_2 \times {}^C\mathbf{b}_3) = 0 \tag{4} $$

Next, and in order to compute one of the 3 unknown degrees of rotational freedom, we introduce the following factorization of ${}^G_C\mathbf{C}$:

$$ {}^G_C\mathbf{C} = \mathbf{C}(\mathbf{k}_1, \theta_1)\mathbf{C}(\mathbf{k}_2, \theta_2)\mathbf{C}(\mathbf{k}_3, \theta_3) \tag{5} $$

where[3]

$$ \mathbf{k}_1 \triangleq \frac{{}^G\mathbf{p}_1 - {}^G\mathbf{p}_2}{\|{}^G\mathbf{p}_1 - {}^G\mathbf{p}_2\|},\ \mathbf{k}_3 \triangleq \frac{{}^C\mathbf{b}_1 \times {}^C\mathbf{b}_2}{\|{}^C\mathbf{b}_1 \times {}^C\mathbf{b}_2\|},\ \mathbf{k}_2 \triangleq \frac{\mathbf{k}_1 \times \mathbf{k}_3}{\|\mathbf{k}_1 \times \mathbf{k}_3\|} \tag{6} $$

Substituting (5) in (2), yields a scalar equation θ_2:

$$ \mathbf{k}_1^T \mathbf{C}(\mathbf{k}_2, \theta_2)\mathbf{k}_3 = 0 \tag{7} $$

which we solve by employing Rodrigues' rotation formula [16]:[4]

$$ \mathbf{C}(\mathbf{k}_2, \theta_2) = \cos\theta_2 \mathbf{I} - \sin\theta_2 \lfloor \mathbf{k}_2 \rfloor + (1 - \cos\theta_2)\mathbf{k}_2\mathbf{k}_2^T \tag{8} $$

to get

$$ \theta_2 = \arccos(\mathbf{k}_1^T \mathbf{k}_3) \pm \frac{\pi}{2} \tag{9} $$

Note that we only need to consider one of these two solutions [in our case, we select $\theta_2 = \arccos(\mathbf{k}_1^T\mathbf{k}_3) - \frac{\pi}{2}$; see Fig. 2], since the other one will result in the same ${}^G_C\mathbf{C}$ (see

[2] Although Masselli and Zell [18] claim that their algorithm runs faster than Kneip *et al.*'s [15], our results (see Section 3) show the opposite to be true (by a small margin). The reason we arrive at a different conclusion is that our simulation randomly generates a new geometric configuration for each run, while Masselli employs only one configuration during their entire simulation, in which they save time due to caching.

[3] $\mathbf{C}(\mathbf{k}, \theta)$ denotes the rotation matrix describing the rotation about the unit vector, $\mathbf{k}$, by an angle θ. Note that in the ensuing derivations, all rotation angles are defined using the left-hand rule.

[4] $\lfloor \mathbf{k} \rfloor$ denotes the 3×3 skew-symmetric matrix corresponding to $\mathbf{k}$ such that $\lfloor \mathbf{k} \rfloor \mathbf{a} = \mathbf{k} \times \mathbf{a}$, $\forall\, \mathbf{k}, \mathbf{a} \in \mathbb{R}^3$. Note also that if $\mathbf{k}$ is a unit vector, then $\lfloor \mathbf{k} \rfloor^2 = \mathbf{k}\mathbf{k}^T - \mathbf{I}$, while for two vectors $\mathbf{a}, \mathbf{b}$, $\lfloor \mathbf{a} \rfloor \lfloor \mathbf{b} \rfloor = \mathbf{b}\mathbf{a}^T - (\mathbf{a}^T\mathbf{b})\mathbf{I}$. Lastly, it is easy to show that $\lfloor \lfloor \mathbf{a} \rfloor \mathbf{b} \rfloor = \mathbf{b}\mathbf{a}^T - \mathbf{a}\mathbf{b}^T$.

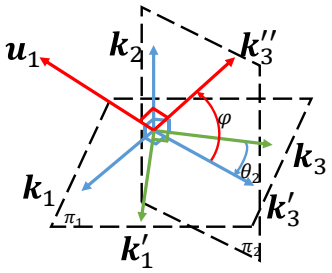

Figure 2. Geometric relation between unit vectors $\mathbf{k}_1$, $\mathbf{k}_2$, $\mathbf{k}_3$, $\mathbf{k}_3'$, $\mathbf{k}_3''$, and $\mathbf{u}_1$. Note that $\mathbf{k}_1$, $\mathbf{k}_3$, and $\mathbf{k}_3'$ belong to a plane π_1 whose normal is $\mathbf{k}_2$. Also, $\mathbf{k}_2$, $\mathbf{k}_3'$, and $\mathbf{k}_3''$ lie on a plane, π_2, normal to π_1.

Appendix 5.2 for a formal proof).

In what follows, we describe the process for eliminating θ_3 from (3) and (4), and eventually arriving into a quartic polynomial involving a trigonometric function of θ_1. To do so, we once again substitute in (3) and (4) the factorization of ${}^G_C\mathbf{C}$ defined in (5) to get (for $i = 1, 2$):

$$\mathbf{u}_i^T \mathbf{C}(\mathbf{k}_1, \theta_1)\mathbf{C}(\mathbf{k}_2, \theta_2)\mathbf{C}(\mathbf{k}_3, \theta_3)\mathbf{v}_i = 0 \tag{10}$$

where

$$\mathbf{u}_i \triangleq {}^G\mathbf{p}_i - {}^G\mathbf{p}_3, \;\; \mathbf{v}_i \triangleq {}^C\mathbf{b}_i \times {}^C\mathbf{b}_3, \;\; i = 1, 2, \tag{11}$$

and employ the following property of rotation matrices

$$\mathbf{C}(\mathbf{k}_1, \theta_1)\mathbf{C}(\mathbf{k}_2, \theta_2)\mathbf{C}^T(\mathbf{k}_1, \theta_1) = \mathbf{C}(\mathbf{C}(\mathbf{k}_1, \theta_1)\mathbf{k}_2, \theta_2)$$

to rewrite (10) in a simpler form as

$$\begin{aligned} &\mathbf{u}_i^T \mathbf{C}(\mathbf{k}_1, \theta_1)\mathbf{C}(\mathbf{C}(\mathbf{k}_2, \theta_2)\mathbf{k}_3, \theta_3)\mathbf{C}(\mathbf{k}_2, \theta_2)\mathbf{v}_i = 0 \\ &\qquad \Rightarrow \mathbf{u}_i^T \mathbf{C}(\mathbf{k}_1, \theta_1)\mathbf{C}(\mathbf{k}_3', \theta_3)\mathbf{v}_i' = 0 \end{aligned} \tag{12}$$

where

$$\begin{aligned} \mathbf{v}_i' &\triangleq \mathbf{C}(\mathbf{k}_2, \theta_2)\mathbf{v}_i, \; i = 1, 2 \\ \mathbf{k}_3' &\triangleq \mathbf{C}(\mathbf{k}_2, \theta_2)\mathbf{k}_3 = \mathbf{k}_2 \times \mathbf{k}_1 \end{aligned} \tag{13}$$

The last equality in (13) is geometrically depicted in Fig. 2 and algebraically derived in Appendix 5.1. Analogously, it is straightforward to show that

$$\mathbf{k}_1' \triangleq \mathbf{C}^T(\mathbf{k}_2, \theta_2)\mathbf{k}_1 = -\mathbf{k}_2 \times \mathbf{k}_3$$

Next, by employing Rodrigues' rotation formula [see (8)], for expressing the product of a rotation matrix and a vector as a linear function of the unknown $\begin{bmatrix}\cos\theta & \sin\theta\end{bmatrix}^T$, *i.e.*,

$$\begin{aligned} \mathbf{C}(\mathbf{k}, \theta)\mathbf{v} &= (-\cos\theta\lfloor\mathbf{k}\rfloor^2 - \sin\theta\lfloor\mathbf{k}\rfloor + \mathbf{k}\mathbf{k}^T)\mathbf{v} \\ &= \begin{bmatrix}-\lfloor\mathbf{k}\rfloor^2\mathbf{v} & -\lfloor\mathbf{k}\rfloor\mathbf{v}\end{bmatrix}\begin{bmatrix}\cos\theta \\ \sin\theta\end{bmatrix} + (\mathbf{k}^T\mathbf{v})\mathbf{k} \end{aligned} \tag{14}$$

in (12) yields (for $i = 1, 2$):

$$\begin{aligned} &\left(\begin{bmatrix}-\lfloor\mathbf{k}_1\rfloor^2\mathbf{u}_i & \lfloor\mathbf{k}_1\rfloor\mathbf{u}_i\end{bmatrix}\begin{bmatrix}\cos\theta_1 \\ \sin\theta_1\end{bmatrix} + (\mathbf{k}_1^T\mathbf{u}_i)\mathbf{k}_1\right)^T \\ &\cdot\left(\begin{bmatrix}-\lfloor\mathbf{k}_3'\rfloor^2\mathbf{v}_i' & -\lfloor\mathbf{k}_3'\rfloor\mathbf{v}_i'\end{bmatrix}\begin{bmatrix}\cos\theta_3 \\ \sin\theta_3\end{bmatrix} + ({\mathbf{k}_3'}^T\mathbf{v}_i')\mathbf{k}_3'\right) = 0 \end{aligned} \tag{15}$$

Expanding (15) and rearranging terms, yields (for $i = 1, 2$)

$$\begin{aligned} &\begin{bmatrix}\cos\theta_1 \\ \sin\theta_1\end{bmatrix}^T \begin{bmatrix}\mathbf{u}_i^T\lfloor\mathbf{k}_1\rfloor^2\lfloor\mathbf{k}_3'\rfloor^2\mathbf{v}_i' & \mathbf{u}_i^T\lfloor\mathbf{k}_1\rfloor^2\lfloor\mathbf{k}_3'\rfloor\mathbf{v}_i' \\ \mathbf{u}_i^T\lfloor\mathbf{k}_1\rfloor\lfloor\mathbf{k}_3'\rfloor^2\mathbf{v}_i' & \mathbf{u}_i^T\lfloor\mathbf{k}_1\rfloor\lfloor\mathbf{k}_3'\rfloor\mathbf{v}_i'\end{bmatrix}\begin{bmatrix}\cos\theta_3 \\ \sin\theta_3\end{bmatrix} \\ &+(\mathbf{k}_1^T\mathbf{u}_i)\begin{bmatrix}-\mathbf{k}_1^T\lfloor\mathbf{k}_3'\rfloor^2\mathbf{v}_i' & -\mathbf{k}_1^T\lfloor\mathbf{k}_3'\rfloor\mathbf{v}_i'\end{bmatrix}\begin{bmatrix}\cos\theta_3 \\ \sin\theta_3\end{bmatrix} \\ &=({\mathbf{k}_3'}^T\mathbf{v}_i')\begin{bmatrix}\mathbf{u}_i^T\lfloor\mathbf{k}_1\rfloor\lfloor\mathbf{k}_3'\rfloor\mathbf{k}_1 & \mathbf{u}_i^T\lfloor\mathbf{k}_1\rfloor\mathbf{k}_3'\end{bmatrix}\begin{bmatrix}\cos\theta_1 \\ \sin\theta_1\end{bmatrix} \end{aligned} \tag{16}$$

Notice that the term $\mathbf{u}_i^T\lfloor\mathbf{k}_1\rfloor\lfloor\mathbf{k}_3'\rfloor$ appears three times in (16), and

$$\begin{aligned} \mathbf{u}_1^T\lfloor\mathbf{k}_1\rfloor\lfloor\mathbf{k}_3'\rfloor &= \mathbf{u}_1^T\mathbf{k}_3'\mathbf{k}_1^T \\ &= ({}^G\mathbf{p}_1 - {}^G\mathbf{p}_3)^T\lfloor\mathbf{k}_1\rfloor\lfloor\mathbf{k}_3'\rfloor \\ &= ({}^G\mathbf{p}_1 - {}^G\mathbf{p}_2 + {}^G\mathbf{p}_2 - {}^G\mathbf{p}_3)^T\lfloor\mathbf{k}_1\rfloor\lfloor\mathbf{k}_3'\rfloor \\ &= ({}^G\mathbf{p}_2 - {}^G\mathbf{p}_3)^T\lfloor\mathbf{k}_1\rfloor\lfloor\mathbf{k}_3'\rfloor \\ &= \mathbf{u}_2^T\mathbf{k}_3'\mathbf{k}_1^T = \mathbf{u}_2^T\lfloor\mathbf{k}_1\rfloor\lfloor\mathbf{k}_3'\rfloor \end{aligned} \tag{17}$$

This motivates us to rewrite (12) as (for $i = 1, 2$):

$$\begin{aligned} 0 &= \mathbf{u}_i^T\mathbf{C}(\mathbf{k}_1, \theta_1)\mathbf{C}(\mathbf{k}_3', \theta_3)\mathbf{v}_i' \\ &= \mathbf{u}_i^T\mathbf{C}(\mathbf{k}_1, \theta_1)\mathbf{C}(\mathbf{k}_1, -\phi)\mathbf{C}(\mathbf{k}_1, \phi)\mathbf{C}(\mathbf{k}_3', \theta_3)\mathbf{v}_i' \\ &= \mathbf{u}_i^T\mathbf{C}(\mathbf{k}_1, \theta_1 - \phi)\mathbf{C}(\mathbf{C}(\mathbf{k}_1, \phi)\mathbf{k}_3', \theta_3)\mathbf{C}(\mathbf{k}_1, \phi)\mathbf{v}_i' \\ &= \mathbf{u}_i^T\mathbf{C}(\mathbf{k}_1, \theta_1')\mathbf{C}(\mathbf{k}_3'', \theta_3)\mathbf{v}_i'' \end{aligned} \tag{18}$$

where

$$\theta_1' \triangleq \theta_1 - \phi, \;\; \mathbf{v}_i'' \triangleq \mathbf{C}(\mathbf{k}_1, \phi)\mathbf{v}_i', \;\; \mathbf{k}_3'' \triangleq \mathbf{C}(\mathbf{k}_1, \phi)\mathbf{k}_3' \tag{19}$$

To simplify the equation analogous to (16) that will result from (18) [instead of (16)], we seek to find a ϕ (not necessarily unique) such that $\mathbf{u}_1^T\mathbf{k}_3'' = 0$, and hence, $\mathbf{u}_i^T\lfloor\mathbf{k}_1\rfloor\lfloor\mathbf{k}_3''\rfloor = 0$ [see (17)], *i.e.*,

$$\begin{aligned} 0 &= \mathbf{u}_1^T\mathbf{k}_3'' \\ &= \mathbf{u}_1^T\mathbf{C}(\mathbf{k}_1, \phi)\mathbf{k}_3' \\ &= \mathbf{u}_1^T(\cos\phi\mathbf{I} - \sin\phi\lfloor\mathbf{k}_1\rfloor + (1 - \cos\phi)\mathbf{k}_1\mathbf{k}_1^T)\mathbf{k}_3' \\ &= \cos\phi\mathbf{u}_1^T\mathbf{k}_3' - \sin\phi\mathbf{u}_1^T\lfloor\mathbf{k}_1\rfloor\mathbf{k}_3' \\ &= \cos\phi\mathbf{u}_1^T\mathbf{k}_3' - \sin\phi\mathbf{u}_1^T\mathbf{k}_2 \end{aligned} \tag{20}$$

$$\Rightarrow \begin{bmatrix}\cos\phi & \sin\phi\end{bmatrix} = \frac{1}{\delta}\begin{bmatrix}\mathbf{u}_1^T\mathbf{k}_2 & \mathbf{u}_1^T\mathbf{k}_3'\end{bmatrix} \tag{21}$$

where

$$\delta \triangleq \sqrt{(\mathbf{u}_1^T\mathbf{k}_3')^2 + (\mathbf{u}_1^T\mathbf{k}_2)^2} = \|\mathbf{u}_1 \times \mathbf{k}_1\| \tag{22}$$

and thus [from (19) using (8)]

$$\begin{aligned}\mathbf{k}_3'' &= \cos\phi\mathbf{k}_3' - \sin\phi\lfloor\mathbf{k}_1\rfloor\mathbf{k}_3' + (1-\cos\phi)\mathbf{k}_1\mathbf{k}_1^T\mathbf{k}_3' \\ &= (\mathbf{k}_3'\mathbf{k}_2^T\mathbf{u}_1 - \mathbf{k}_2{\mathbf{k}_3'}^T\mathbf{u}_1)/\delta = \mathbf{u}_1 \times (\mathbf{k}_3' \times \mathbf{k}_2)/\delta \\ &= \frac{\mathbf{u}_1 \times \mathbf{k}_1}{\|\mathbf{u}_1 \times \mathbf{k}_1\|}\end{aligned} \tag{23}$$

Now, we can expand (18) using (14) to get an equation analogous to (16):

$$\begin{aligned}&\begin{bmatrix}\cos\theta_1' \\ \sin\theta_1'\end{bmatrix}^T \begin{bmatrix}\mathbf{u}_i^T\lfloor\mathbf{k}_1\rfloor^2\lfloor\mathbf{k}_3''\rfloor^2\mathbf{v}_i'' & \mathbf{u}_i^T\lfloor\mathbf{k}_1\rfloor^2\lfloor\mathbf{k}_3''\rfloor\mathbf{v}_i'' \\ \mathbf{u}_i^T\lfloor\mathbf{k}_1\rfloor\lfloor\mathbf{k}_3''\rfloor^2\mathbf{v}_i'' & \mathbf{u}_i^T\lfloor\mathbf{k}_1\rfloor\lfloor\mathbf{k}_3''\rfloor\mathbf{v}_i''\end{bmatrix}\begin{bmatrix}\cos\theta_3 \\ \sin\theta_3\end{bmatrix} \\ &+ (\mathbf{k}_1^T\mathbf{u}_i)\begin{bmatrix}-\mathbf{k}_1^T\lfloor\mathbf{k}_3''\rfloor^2\mathbf{v}_i'' & -\mathbf{k}_1^T\lfloor\mathbf{k}_3''\rfloor\mathbf{v}_i''\end{bmatrix}\begin{bmatrix}\cos\theta_3 \\ \sin\theta_3\end{bmatrix} = \\ &(\mathbf{k}_3''^T\mathbf{v}_i'')\begin{bmatrix}\mathbf{u}_i^T\lfloor\mathbf{k}_1\rfloor\lfloor\mathbf{k}_3''\rfloor\mathbf{k}_1 & \mathbf{u}_i^T\lfloor\mathbf{k}_1\rfloor\mathbf{k}_3''\end{bmatrix}\begin{bmatrix}\cos\theta_1' \\ \sin\theta_1'\end{bmatrix}\end{aligned} \tag{24}$$

Substituting $\mathbf{u}_i^T\lfloor\mathbf{k}_1\rfloor\lfloor\mathbf{k}_3''\rfloor = 0$ [see (17)] in (24) and renaming terms, yields (for $i = 1, 2$):

$$\begin{aligned}&\begin{bmatrix}\cos\theta_1' \\ \sin\theta_1'\end{bmatrix}^T \begin{bmatrix}\bar{f}_{i1} & \bar{f}_{i2} \\ 0 & 0\end{bmatrix}\begin{bmatrix}\cos\theta_3 \\ \sin\theta_3\end{bmatrix} + \begin{bmatrix}\bar{f}_{i4} & \bar{f}_{i5}\end{bmatrix}\begin{bmatrix}\cos\theta_3 \\ \sin\theta_3\end{bmatrix} \\ &= \begin{bmatrix}0 & \bar{f}_{i3}\end{bmatrix}\begin{bmatrix}\cos\theta_1' \\ \sin\theta_1'\end{bmatrix} \\ &\Rightarrow \begin{bmatrix}\bar{f}_{i1}\cos\theta_1' + \bar{f}_{i4} & \bar{f}_{i2}\cos\theta_1' + \bar{f}_{i5}\end{bmatrix}\begin{bmatrix}\cos\theta_3 \\ \sin\theta_3\end{bmatrix} = \bar{f}_{i3}\sin\theta_1'\end{aligned} \tag{25}$$

where[5]

$$\begin{aligned}\bar{f}_{i1} &\triangleq \mathbf{u}_i^T\lfloor\mathbf{k}_1\rfloor^2\lfloor\mathbf{k}_3''\rfloor^2\mathbf{v}_i'' = \delta\mathbf{v}_i^T\mathbf{k}_2 \\ \bar{f}_{i2} &\triangleq \mathbf{u}_i^T\lfloor\mathbf{k}_1\rfloor^2\lfloor\mathbf{k}_3''\rfloor\mathbf{v}_i'' = \delta\mathbf{v}_i^T\mathbf{k}_1' \\ \bar{f}_{i3} &\triangleq (\mathbf{k}_3''^T\mathbf{v}_i'')\mathbf{u}_i^T\lfloor\mathbf{k}_1\rfloor\mathbf{k}_3'' = \delta\mathbf{v}_i^T\mathbf{k}_3 \\ \bar{f}_{i4} &\triangleq -(\mathbf{u}_i^T\mathbf{k}_1)\mathbf{k}_1^T\lfloor\mathbf{k}_3''\rfloor^2\mathbf{v}_i'' = (\mathbf{u}_i^T\mathbf{k}_1)(\mathbf{v}_i^T\mathbf{k}_1') \\ \bar{f}_{i5} &\triangleq -(\mathbf{u}_i^T\mathbf{k}_1)\mathbf{k}_1^T\lfloor\mathbf{k}_3''\rfloor\mathbf{v}_i'' = -(\mathbf{u}_i^T\mathbf{k}_1)(\mathbf{v}_i^T\mathbf{k}_2)\end{aligned}$$

For $i = 1, 2$, (25) results into the following system:

$$\begin{bmatrix}\bar{f}_{11}\cos\theta_1' + \bar{f}_{14} & \bar{f}_{12}\cos\theta_1' + \bar{f}_{15} \\ \bar{f}_{21}\cos\theta_1' + \bar{f}_{24} & \bar{f}_{22}\cos\theta_1' + \bar{f}_{25}\end{bmatrix}\begin{bmatrix}\cos\theta_3 \\ \sin\theta_3\end{bmatrix} = \begin{bmatrix}\bar{f}_{13} \\ \bar{f}_{23}\end{bmatrix}\sin\theta_1' \tag{26}$$

Note that since $\bar{f}_{11}\bar{f}_{14} + \bar{f}_{12}\bar{f}_{15} = 0$, we can further simplify (26) by introducing θ_3', where

$$\begin{bmatrix}\cos\theta_3' \\ \sin\theta_3'\end{bmatrix} \triangleq \begin{bmatrix}\frac{\bar{f}_{11}\cos\theta_3 + \bar{f}_{12}\sin\theta_3}{\sqrt{\bar{f}_{11}^2 + \bar{f}_{12}^2}} & -\frac{\bar{f}_{14}\cos\theta_3 + \bar{f}_{15}\sin\theta_3}{\sqrt{\bar{f}_{14}^2 + \bar{f}_{15}^2}}\end{bmatrix}^T \tag{27}$$

[5]The simplified expressions for the terms shown after the second equality, require lengthy derivations which we omit due to space limitations.

Replacing θ_3 by θ_3' in (26), we have

$$\begin{bmatrix}f_{11}\cos\theta_1' & f_{15} \\ f_{21}\cos\theta_1' + f_{24} & f_{22}\cos\theta_1' + f_{25}\end{bmatrix}\begin{bmatrix}\cos\theta_3' \\ \sin\theta_3'\end{bmatrix} = \begin{bmatrix}f_{13} \\ f_{23}\end{bmatrix}\sin\theta_1' \tag{28}$$

where

$$f_{11} \triangleq \delta\mathbf{k}_3^{T\,C}\mathbf{b}_3 \tag{29}$$
$$f_{21} \triangleq \delta({}^C\mathbf{b}_1^{T\,C}\mathbf{b}_2)(\mathbf{k}_3^{T\,C}\mathbf{b}_3) \tag{30}$$
$$f_{22} \triangleq \delta(\mathbf{k}_3^{T\,C}\mathbf{b}_3)\|{}^C\mathbf{b}_1 \times {}^C\mathbf{b}_2\| \tag{31}$$
$$f_{13} \triangleq \bar{f}_{13} = \delta\mathbf{v}_1^T\mathbf{k}_3 \tag{32}$$
$$f_{23} \triangleq \bar{f}_{23} = \delta\mathbf{v}_2^T\mathbf{k}_3 \tag{33}$$
$$f_{24} \triangleq (\mathbf{u}_2^T\mathbf{k}_1)(\mathbf{k}_3^{T\,C}\mathbf{b}_3)\|{}^C\mathbf{b}_1 \times {}^C\mathbf{b}_2\| \tag{34}$$
$$f_{15} \triangleq -(\mathbf{u}_1^T\mathbf{k}_1)(\mathbf{k}_3^{T\,C}\mathbf{b}_3) \tag{35}$$
$$f_{25} \triangleq -(\mathbf{u}_2^T\mathbf{k}_1)({}^C\mathbf{b}_1^{T\,C}\mathbf{b}_2)(\mathbf{k}_3^{T\,C}\mathbf{b}_3) \tag{36}$$

From (28), we have

$$\begin{aligned}\begin{bmatrix}\cos\theta_3' \\ \sin\theta_3'\end{bmatrix} &= \det\left(\begin{bmatrix}f_{11}\cos\theta_1' & f_{15} \\ f_{21}\cos\theta_1' + f_{24} & f_{22}\cos\theta_1' + f_{25}\end{bmatrix}\right)^{-1} \\ &\cdot \begin{bmatrix}f_{22}\cos\theta_1' + f_{25} & -f_{15} \\ -(f_{21}\cos\theta_1' + f_{24}) & f_{11}\cos\theta_1'\end{bmatrix}\begin{bmatrix}f_{13} \\ f_{23}\end{bmatrix}\sin\theta_1'\end{aligned} \tag{37}$$

Computing the norm of both sides of (37), results in

$$\begin{aligned}&\left\|\begin{bmatrix}f_{22}\cos\theta_1' + f_{25} & -f_{15} \\ -(f_{21}\cos\theta_1' + f_{24}) & f_{11}\cos\theta_1'\end{bmatrix}\begin{bmatrix}f_{13} \\ f_{23}\end{bmatrix}\right\|^2 (1 - \cos^2\theta_1') \\ &= \det\left(\begin{bmatrix}f_{11}\cos\theta_1' & f_{15} \\ f_{21}\cos\theta_1' + f_{24} & f_{22}\cos\theta_1' + f_{25}\end{bmatrix}\right)^2\end{aligned}$$

which is a 4th-order polynomial in $\cos\theta_1'$ that can be compactly written as:

$$\sum_{j=0}^{4}\alpha_j\cos^j\theta_1' = 0 \tag{38}$$

with

$$\alpha_4 \triangleq g_5^2 + g_1^2 + g_3^2 \tag{39}$$
$$\alpha_3 \triangleq 2(g_5g_6 + g_1g_2 + g_3g_4) \tag{40}$$
$$\alpha_2 \triangleq g_6^2 + 2g_5g_7 + g_2^2 + g_4^2 - g_1^2 - g_3^2 \tag{41}$$
$$\alpha_1 \triangleq 2(g_6g_7 - g_1g_2 - g_3g_4) \tag{42}$$
$$\alpha_0 \triangleq g_7^2 - g_2^2 - g_4^2 \tag{43}$$
$$\tag{44}$$

where

$$g_1 \triangleq f_{13}f_{22} \tag{45}$$
$$g_2 \triangleq f_{13}f_{25} - f_{15}f_{23} \tag{46}$$
$$g_3 \triangleq f_{11}f_{23} - f_{13}f_{21} \tag{47}$$
$$g_4 \triangleq -f_{13}f_{24} \tag{48}$$
$$g_5 \triangleq f_{11}f_{22} \tag{49}$$
$$g_6 \triangleq f_{11}f_{25} - f_{15}f_{21} \tag{50}$$
$$g_7 \triangleq -f_{15}f_{24} \tag{51}$$

We compute the roots of (38) in closed form to find $\cos\theta_1'$. Similarly to [15] and [18], we employ Ferrari's method [2] to attain the resolvent cubic of (38), which is subsequently solved by Cardano's formula [2]. Once the (up to) four real solutions of (38) have been determined, an *optional* step is to apply root polishing following Newton's method, which improves accuracy for minimal increase in the processing cost (see Section 3.2). Regardless, for each solution of $\cos\theta_1'$, we will have two possible solutions for $\sin\theta_1'$, *i.e.*,

$$\sin\theta_1' = \pm\sqrt{1-\cos^2\theta_1'} \tag{52}$$

which, in general, will result in two different solutions for ${}^C_G\mathbf{C}$. Note though that only one of them is valid if we use the fact that $d_i > 0$ (see Appendix 5.3).

Next, for each pair of $(\cos\theta_1', \sin\theta_1')$, we compute $\cos\theta_3'$ and $\sin\theta_3'$ from (37), which can be written as

$$\begin{bmatrix}\cos\theta_3' \\ \sin\theta_3'\end{bmatrix} = \frac{\sin\theta_1'}{g_5\cos^2\theta_1' + g_6\cos\theta_1' + g_7}\begin{bmatrix}g_1\cos\theta_1' + g_2 \\ g_3\cos\theta_1' + g_4\end{bmatrix} \tag{53}$$

Lastly, instead of first computing θ_1 from (19) and θ_3 from (27) to find ${}^C_G\mathbf{C}$ using (5), we hereafter describe a faster method for recovering ${}^C_G\mathbf{C}$. Specifically, from (5), (12) and (18), we have

$$\begin{aligned}{}^C_G\mathbf{C} &= \mathbf{C}(\mathbf{k}_1,\theta_1)\mathbf{C}(\mathbf{k}_2,\theta_2)\mathbf{C}(\mathbf{k}_3,\theta_3) \\ &= \mathbf{C}(\mathbf{k}_1,\theta_1)\mathbf{C}(\mathbf{k}_3',\theta_3)\mathbf{C}(\mathbf{k}_2,\theta_2) \\ &= \mathbf{C}(\mathbf{k}_1,\theta_1')\mathbf{C}(\mathbf{k}_3'',\theta_3)\mathbf{C}(\mathbf{k}_1,\phi)\mathbf{C}(\mathbf{k}_2,\theta_2)\end{aligned} \tag{54}$$

Since $\mathbf{k}_1$ is perpendicular to $\mathbf{k}_3''$, we can construct a rotation matrix $\bar{\mathbf{C}}$ such that

$$\bar{\mathbf{C}} = \begin{bmatrix}\mathbf{k}_1 & \mathbf{k}_3'' & \mathbf{k}_1\times\mathbf{k}_3''\end{bmatrix}$$

and hence

$$\mathbf{k}_1 = \bar{\mathbf{C}}\mathbf{e}_1,\ \mathbf{k}_3'' = \bar{\mathbf{C}}\mathbf{e}_2 \tag{55}$$

where

$$\begin{bmatrix}\mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3\end{bmatrix} \triangleq \mathbf{I}_3$$

Substituting (55) in (54), we have

$$\begin{aligned}{}^C_G\mathbf{C} &= \bar{\mathbf{C}}\mathbf{C}(\mathbf{e}_1,\theta_1')\mathbf{C}(\mathbf{e}_2,\theta_3)\bar{\mathbf{C}}^T\mathbf{C}(\mathbf{k}_1,\phi)\mathbf{C}(\mathbf{k}_2,\theta_2) \\ &= \bar{\mathbf{C}}\mathbf{C}(\mathbf{e}_1,\theta_1')\mathbf{C}(\mathbf{e}_2,\theta_3)\mathbf{C}(\mathbf{e}_2,\theta_3'-\theta_3)\bar{\bar{\mathbf{C}}} \\ &= \bar{\mathbf{C}}\mathbf{C}(\mathbf{e}_1,\theta_1')\mathbf{C}(\mathbf{e}_2,\theta_3')\bar{\bar{\mathbf{C}}}\end{aligned} \tag{56}$$

where

$$\begin{aligned}\bar{\bar{\mathbf{C}}} &\triangleq \mathbf{C}(\mathbf{e}_2,\theta_3-\theta_3')\bar{\mathbf{C}}^T\mathbf{C}(\mathbf{k}_1,\phi)\mathbf{C}(\mathbf{k}_2,\theta_2) \\ &= \mathbf{C}(\mathbf{e}_2,\theta_3-\theta_3')\begin{bmatrix}\mathbf{k}_1' & \mathbf{k}_3 & \mathbf{k}_1'\times\mathbf{k}_3\end{bmatrix}^T \\ &\overset{(27)}{=} \begin{bmatrix}{}^C\mathbf{b}_1 & \mathbf{k}_3 & {}^C\mathbf{b}_1\times\mathbf{k}_3\end{bmatrix}^T\end{aligned}$$

The advantages of (56) are: (i) The matrix product $\mathbf{C}(\mathbf{e}_1,\theta_1')\mathbf{C}(\mathbf{e}_2,\theta_3')$ can be computed analytically; (ii) $\bar{\mathbf{C}}, \bar{\bar{\mathbf{C}}}$ are invariant to the (up to) four possible solutions and thus, we only need to construct them once.

2.3. Solving for the position

Substituting in (1) the expression for d_3 from (63) and rearranging terms, yields

$${}^G\mathbf{p}_C = {}^G\mathbf{p}_3 - \frac{\delta\sin\theta_1'}{\mathbf{k}_3^{T\,C}\mathbf{b}_3}{}^G_C\mathbf{C}\,{}^C\mathbf{b}_3 \tag{57}$$

Note that we only use (1) for $i=3$ to compute ${}^G\mathbf{p}_C$ from ${}^G_C\mathbf{C}$. Alternatively, if we care more for accuracy than speed, we can find the position using a least-squares approach based on (1) (see [14] for details). Lastly, the proposed P3P solution is summarized in Alg. 1.

Algorithm 1: Solving for the camera's pose

Input: ${}^G\mathbf{p}_i,\ i=1,2,3$ the features' positions; ${}^C\mathbf{b}_i,\ i=1,2,3$ bearing measurements
Output: ${}^G\mathbf{p}_C$, the position of the camera; ${}^C_G\mathbf{C}$, the orientation of the camera

Compute $\mathbf{k}_1, \mathbf{k}_3$ using (6)
Compute $\mathbf{u}_i$ and $\mathbf{v}_i$ using (11), $i=1,2$
Compute δ and $\mathbf{k}_3''$ using (22) and (23)
Compute the f_{ij}'s using (29)-(36)
Compute α_i, $i=0,1,2,3,4$ using (39)-(51)
Solve (38) to get n ($n=2$ or 4) real solutions for $\cos\theta_1'$, denoted as $\cos\theta_1'^{(i)}$, $i=1...n$
for $i=1:n$ **do**
  $\sin\theta_1'^{(i)} \leftarrow \text{sign}(\mathbf{k}_3^{T\,C}\mathbf{b}_3)\sqrt{1-\cos^2\theta_1'^{(i)}}$
  Compute $\cos\theta_3'^{(i)}$ and $\sin\theta_3'^{(i)}$ using (53)
  Compute ${}^C_G\mathbf{C}^{(i)}$ using (56)
  Compute ${}^G\mathbf{p}_C^{(i)}$ using (57)
end

	position	orientation
Gao's method	6.36E-05	1.31E-04
Kneip's method	1.18E-05	1.02E-05
Masselli's method	1.84E-08	4.89E-10
Proposed method	**1.66E-10**	**5.30E-12**
Proposed method+Root polishing	**5.07E-11**	**1.53E-13**

Table 1. Nominal case: Pose mean errors.

3. Simulation results

Our algorithm is implemented[6] in C++ using the same linear algebra library, TooN [23], as [15] We employ simulation data to test our code and compare it to the solutions of [15] and [18]. For each single P3P problem, we randomly generate three 3D landmarks, which are uniformly distributed in a $0.4 \times 0.3 \times 0.4$ cuboid centered around the origin. The position of the camera is ${}^{G}\mathbf{p}_{C} = \mathbf{e}_3$, and its orientation is ${}^{C}_{G}\mathbf{C} = \mathbf{C}(\mathbf{e}_1, \pi)$.

3.1. Numerical accuracy

We generate simulation data without adding any noise or rounding error to the bearing measurements, and run all three algorithms on 50,000 randomly-generated configurations to assess their numerical accuracy. Note that the position error is computed as the norm of the difference between the estimate and the ground truth of ${}^{G}\mathbf{p}_{C}$. As for the orientation error, we compute the rotation matrix that transforms the estimated ${}^{G}_{C}\mathbf{C}$ to the true one, convert it to the equivalent axis-angle representation, and use the absolute value of the angle as the error. Since there are multiple solutions to a P3P problem, we compute the errors for all of them and pick the smallest one (*i.e.*, the root closest to the true solution).

The distributions and the means of the position and orientation errors are depicted in Fig.s 3 - 4 and Table 1. As evident, we get similar results to those presented in [18] for Kneip *et al.*'s [15] and Masselli and Zell's methods [18], while our approach outperforms both of them by two orders of magnitude in terms of accuracy. This can be attributed to the fact that our algorithm requires fewer operations and thus exhibits lower numerical-error propagation. For completeness, in Table 1, we also list the pose errors for OpenCV's [21] implementation of Gao's method [6].

Lastly, and as shown in the results of Table 1, we can further improve the numerical precision by applying root polishing. Typically, two iterations of Newton's method [25] lead to significantly better results, especially for the orientation, while taking only 0.01 μs per iteration, or about 5% of the total processing time.

3.2. Processing cost

We use a test program that solves 100,000 randomly generated P3P problems and calculates the total execution time

[6]Our code is available as supplemental material and in OpenCV [21].

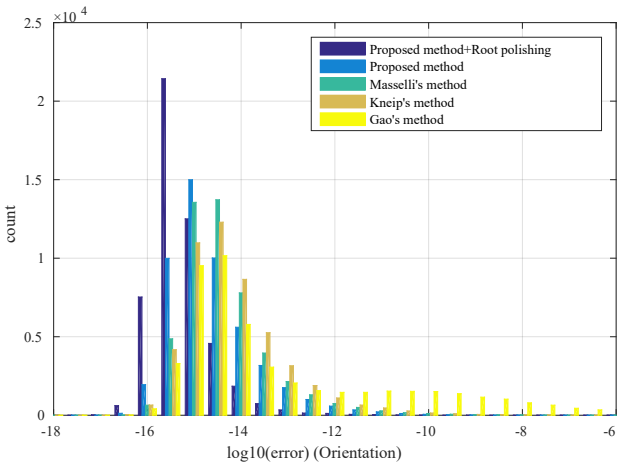

Figure 3. Nominal case: Histogram of orientation errors.

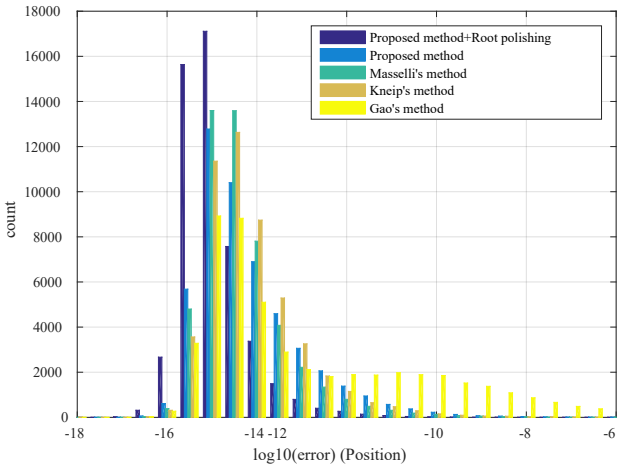

Figure 4. Nominal case: Histogram of position errors.

to evaluate the computational cost of the three algorithms considered. We run it on a 2.0 GHz×4 Core laptop and the results show that our code takes 0.43 μs on average (0.41 μs without root polishing) while [15], [18] and [6] take 1.3 μs, 1.5 μs, and 3.1 μs respectively. This corresponds to a 3× speed up (or 33% of the time of [15]). Note also, in contrast to what is reported in [18], Masselli's method is actually slower than Kneip's. As mentioned earlier, Masselli's results in [18] are based on 1,000 runs of the same features' configuration, and take advantage of data caching to outperform Kneip.

3.3. Robustness

There are two typical singular cases that lead to infinite solutions in the P3P problem:

- Singular case 1: The 3 landmarks are collinear.
- Singular case 2: Any two of the 3 bearing measurements coincide.

In practice, it is almost impossible for these conditions to hold exactly, but we may still have numerical issues when the geometric configuration is close to these cases. To test the robustness of the three algorithms considered, we generate simulation data corresponding to small perturbations (uniformly distributed within $[-0.05\ \ 0.05]$) of the features' positions when in singular configurations. The errors are defined as in Section 3.1, while we compute the medians[7] of them to assess the robustness of the three methods. For fairness, we do not apply root polishing to our code here. According to the results shown in Fig.s 5 - 8 and Tables 2 - 3, our method achieves the best accuracy in these two close-to-singular cases. The reason is that we do not compute any quantities that may suffer from numerical issues, such as cotangent and tangent in [15] and [18], respectively.

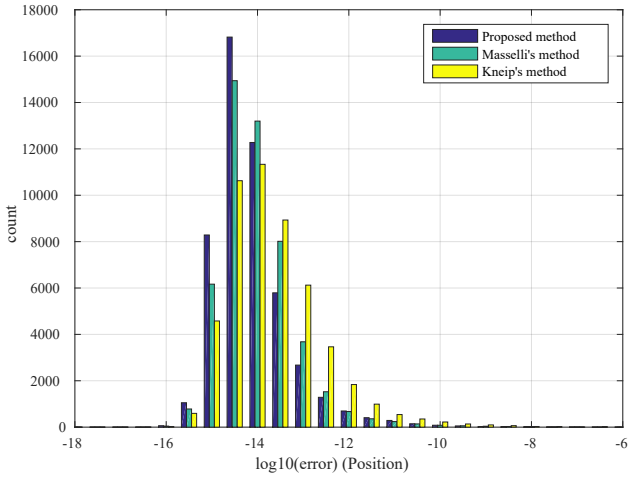

Figure 5. Singular case 1: Histogram of position errors.

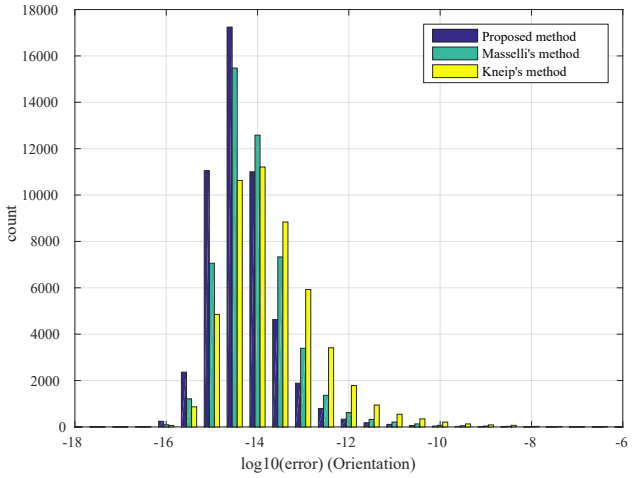

Figure 6. Singular case 1: Histogram of orientation errors.

[7]In close-to-singular cases, a few instances of large errors (10^{-1}) may dominate and skew the mean. Instead, the median is a robust measure of the central tendency, and provides a better performance metric.

	position	orientation
Kneip's method	1.42E-14	1.34E-14
Masselli's method	7.13E-15	6.15E-15
Proposed method	**5.16E-15**	**3.73E-15**

Table 2. Singular case 1: Pose median errors.

	position	orientation
Kneip's method	8.10E-14	8.85E-14
Masselli's method	7.24E-14	6.07E-14
Proposed method	**6.73E-14**	**1.75E-14**

Table 3. Singular case 2: Pose median errors.

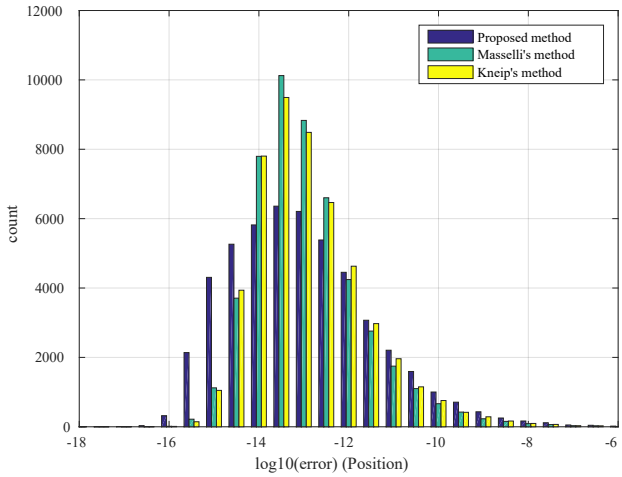

Figure 7. Singular case 2: Histogram of position errors.

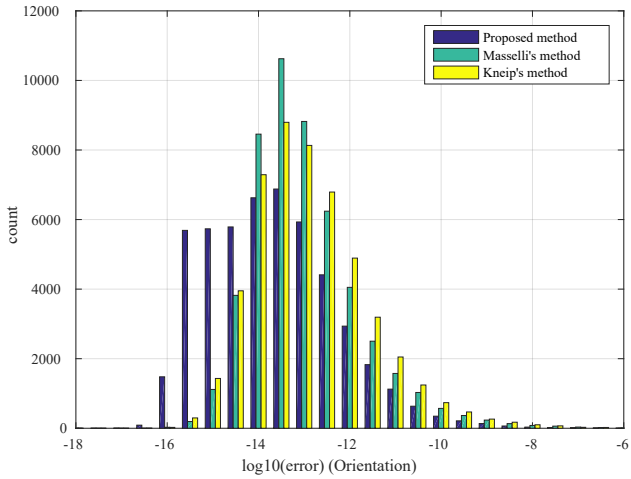

Figure 8. Singular case 2: Histogram of orientation errors.

4. Conclusion and Future Work

In this paper, we have introduced an algebraic approach for computing the solutions of the P3P problem in closed form. Similarly to [15] and [18], our algorithm does *not* solve for the distances first, and hence reduces numerical-error propagation. Differently though, it does not involve

numerically-unstable functions (*e.g.*, tangent, or cotangent) and has simpler expressions than the two recent alternative methods [15, 18], and thus it outperforms them in terms of speed, accuracy, and robustness to close-to-singular cases.

As part of our ongoing work, we are currently extending our approach to also address the case of the generalized (non-central camera) P3P [20].

5. Appendix

5.1. Proof of $\mathbf{k}'_3 = \mathbf{k}_2 \times \mathbf{k}_1$

First, note that $\mathbf{k}_2 \times \mathbf{k}_1$ is a unit vector since $\mathbf{k}_2$ is perpendicular to $\mathbf{k}_1$. Also, from (13) and (7) we have

$$\mathbf{k}_1^T \mathbf{k}'_3 = \mathbf{k}_1^T \mathbf{C}(\mathbf{k}_2, \theta_2)\mathbf{k}_3 = 0 \tag{58}$$

Then, we can prove $\mathbf{k}'_3 = \mathbf{k}_2 \times \mathbf{k}_1$ by showing that their inner product is equal to 1:

$$\begin{aligned}(\mathbf{k}_2 \times \mathbf{k}_1)^T \mathbf{k}'_3 &= \mathbf{k}_1^T (\mathbf{k}'_3 \times \mathbf{k}_2) \\ &\overset{(6)}{=} \frac{\mathbf{k}_1^T (\mathbf{k}'_3 \times (\mathbf{k}_1 \times \mathbf{k}_3))}{\|\mathbf{k}_1 \times \mathbf{k}_3\|} \\ &\overset{(9)}{=} \frac{\mathbf{k}_1^T (\mathbf{k}_1 (\mathbf{k}_3^T \mathbf{k}'_3) - \mathbf{k}_3 (\mathbf{k}_1^T \mathbf{k}'_3))}{\cos\theta_2} \\ &= \frac{\mathbf{k}_3^T \mathbf{C}(\mathbf{k}_2, \theta_2)\mathbf{k}_3}{\cos\theta_2} = 1\end{aligned}$$

5.2. Equivalence between the two solutions of θ_2

When solving for θ_2 [see (9)], we have two possible solutions $\theta_2^{(1)} = \arccos(\mathbf{k}_1^T \mathbf{k}_3) - \frac{\pi}{2}$ and $\theta_2^{(2)} = \theta_2^{(1)} + \pi$. Next, we will prove that using $\theta_2^{(2)}$ to find ${}^G_C\mathbf{C}$ is equivalent to using $\theta_2^{(1)}$. First, note that (see Fig. 2)

$$\begin{aligned}\mathbf{C}(\mathbf{k}_2, \theta_2^{(1)} + \frac{\pi}{2})\mathbf{k}_3 &= \mathbf{C}(\mathbf{k}_2, \frac{\pi}{2})\mathbf{k}'_3 = -\mathbf{k}_2 \times \mathbf{k}'_3 \\ &= -\mathbf{k}_2 \times (\mathbf{k}_2 \times \mathbf{k}_1) = \mathbf{k}_1\end{aligned} \tag{59}$$

Then, we can write $\mathbf{C}(\mathbf{k}_2, \theta_2^{(2)})$ as

$$\begin{aligned}&\mathbf{C}(\mathbf{k}_2, \theta_2^{(2)}) \\ =&\mathbf{C}(\mathbf{k}_2, \theta_2^{(1)} + \frac{\pi}{2})\mathbf{C}(\mathbf{k}_2, \frac{\pi}{2}) \\ =&\mathbf{C}(\mathbf{k}_2, \theta_2^{(1)} + \frac{\pi}{2})\mathbf{C}(\mathbf{k}_3, \pi)\mathbf{C}(\mathbf{k}_2, -\frac{\pi}{2})\mathbf{C}(\mathbf{k}_3, -\pi) \\ =&\mathbf{C}(\mathbf{C}(\mathbf{k}_2, \theta_2^{(1)} + \frac{\pi}{2})\mathbf{k}_3, \pi)\mathbf{C}(\mathbf{k}_2, \theta_2^{(1)} + \frac{\pi}{2})\mathbf{C}(\mathbf{k}_2, -\frac{\pi}{2})\mathbf{C}(\mathbf{k}_3, \pi) \\ \overset{(59)}{=}&\mathbf{C}(\mathbf{k}_1, \pi)\mathbf{C}(\mathbf{k}_2, \theta_2^{(1)})\mathbf{C}(\mathbf{k}_3, \pi)\end{aligned} \tag{60}$$

If we use $\theta_2^{(2)}$ to find ${}^G_C\mathbf{C}$,

$$\begin{aligned}{}^G_C\mathbf{C} &= \mathbf{C}(\mathbf{k}_1, \theta_1^{(2)})\mathbf{C}(\mathbf{k}_2, \theta_2^{(2)})\mathbf{C}(\mathbf{k}_3, \theta_3^{(2)}) \\ &\overset{(60)}{=} \mathbf{C}(\mathbf{k}_1, \theta_1^{(2)})\mathbf{C}(\mathbf{k}_1, \pi)\mathbf{C}(\mathbf{k}_2, \theta_2^{(1)})\mathbf{C}(\mathbf{k}_3, \pi)\mathbf{C}(\mathbf{k}_3, \theta_3^{(2)}) \\ &= \mathbf{C}(\mathbf{k}_1, \theta_1^{(2)} + \pi)\mathbf{C}(\mathbf{k}_2, \theta_2^{(1)})\mathbf{C}(\mathbf{k}_3, \theta_3^{(2)} + \pi)\end{aligned} \tag{61}$$

Note that ${}^G_C\mathbf{C}$ in (61) is of the same form as that in (5), so any solutions of ${}^G_C\mathbf{C}$ computed using $\theta_2^{(2)}$ will be found by using $\theta_2^{(1)}$. Thus, we do not need to consider any other solutions for θ_1 and θ_3 beyond the ones found for ${}^G_C\mathbf{C}$.

5.3. Determining the sign of $\sin\theta'_1$

From (52), we have two solutions for $\sin\theta'_1$, and thus for θ'_1, with $\theta_1^{\prime(2)} = -\theta'_1$. This will also result into two solutions for θ'_3 [see (53)] and, hence, two solutions for θ_3: θ_3 and $\theta_3^{(2)} = \theta_3 + \pi$. Considering these two options, we get two distinct solutions for ${}^G_C\mathbf{C}$ [see (54)]:

$$\begin{aligned}\mathbf{C}_1 &\triangleq \mathbf{C}(\mathbf{k}_1, \theta'_1)\mathbf{C}(\mathbf{k}''_3, \theta_3)\mathbf{C}(\mathbf{k}_1, \phi)\mathbf{C}(\mathbf{k}_2, \theta_2) \\ \mathbf{C}_2 &\triangleq \mathbf{C}(\mathbf{k}_1, -\theta'_1)\mathbf{C}(\mathbf{k}''_3, \theta_3 + \pi)\mathbf{C}(\mathbf{k}_1, \phi)\mathbf{C}(\mathbf{k}_2, \theta_2)\end{aligned}$$

Then, notice that

$$\begin{aligned}\mathbf{C}_2\mathbf{C}_1^T &= \mathbf{C}(\mathbf{k}_1, -\theta'_1)\mathbf{C}(\mathbf{k}''_3, \pi)\mathbf{C}(\mathbf{k}_1, -\theta'_1) \\ &= \mathbf{C}(\mathbf{k}''_3, \pi)\mathbf{C}(\mathbf{C}^T(\mathbf{k}''_3, \pi)\mathbf{k}_1, -\theta'_1)\mathbf{C}(\mathbf{k}_1, -\theta'_1) \\ &= \mathbf{C}(\mathbf{k}''_3, \pi)\mathbf{C}(-\mathbf{k}_1, -\theta'_1)\mathbf{C}(\mathbf{k}_1, -\theta'_1) \\ &= \mathbf{C}(\mathbf{k}''_3, \pi)\end{aligned}$$

If $\mathbf{C}_1 = \mathbf{C}_2$, this would require

$$\mathbf{C}(\mathbf{k}''_3, \pi) = \mathbf{C}_2\mathbf{C}_1^T = \mathbf{I}$$

which cannot be true, hence $\mathbf{C}_1$ and $\mathbf{C}_2$ cannot be equal. Thus, there are always two different solutions of ${}^G_C\mathbf{C}$.

If, however, we use the fact that d_i $(i = 1, 2, 3)$ is positive, we can determine the sign of $\sin\theta'_1$, and choose the valid one among the two solutions of ${}^G_C\mathbf{C}$. Subtracting (1) pairwise for $(i = 3)$ from $(i = 1)$, we have

$$\begin{aligned}{}^G\mathbf{p}_1 - {}^G\mathbf{p}_3 =& d_1 {}^G_C\mathbf{C}\,{}^C\mathbf{b}_1 - d_3 {}^G_C\mathbf{C}\,{}^C\mathbf{b}_3 \\ \Rightarrow {}^G\mathbf{p}_1 - {}^G\mathbf{p}_3 =& \mathbf{C}(\mathbf{k}_1, \theta'_1)\mathbf{C}(\mathbf{k}''_3, \theta_3)\mathbf{C}(\mathbf{k}_1, \phi) \\ &\cdot\mathbf{C}(\mathbf{k}_2, \theta_2)(d_1 {}^C\mathbf{b}_1 - d_3 {}^C\mathbf{b}_3)\end{aligned} \tag{62}$$

Multiplying both sides of (62) with $\mathbf{k}_3^{\prime\prime T}\mathbf{C}(\mathbf{k}_1, -\theta'_1)$ from the left, yields

$$\begin{aligned}&\mathbf{k}_3^{\prime\prime T}\mathbf{C}(\mathbf{k}_1, -\theta'_1)({}^G\mathbf{p}_1 - {}^G\mathbf{p}_3) \\ =&\mathbf{k}_3^{\prime\prime T}\mathbf{C}(\mathbf{k}_1, \phi)\mathbf{C}(\mathbf{k}_2, \theta_2)(d_1 {}^C\mathbf{b}_1 - d_3 {}^C\mathbf{b}_3) \\ \Rightarrow&\mathbf{k}_3^{\prime\prime T}(\cos\theta'_1\mathbf{I} + \sin\theta'_1\lfloor\mathbf{k}_1\rfloor + (1 - \cos\theta'_1)\mathbf{k}_1\mathbf{k}_1^T)\mathbf{u}_1 \\ =&\mathbf{k}_3^{\prime T}\mathbf{C}(\mathbf{k}_2, \theta_2)(d_1 {}^C\mathbf{b}_1 - d_3 {}^C\mathbf{b}_3) \\ \overset{(23)}{\Rightarrow}&\sin\theta'_1\mathbf{k}_3^{\prime\prime T}\lfloor\mathbf{k}_1\rfloor\mathbf{u}_1 = \mathbf{k}_3^T(d_1 {}^C\mathbf{b}_1 - d_3 {}^C\mathbf{b}_3) \\ \Rightarrow&-\sin\theta'_1\mathbf{u}_1^T\lfloor\mathbf{k}_1\rfloor\mathbf{k}''_3 = -d_3\mathbf{k}_3^{T C}\mathbf{b}_3 \\ \Rightarrow&\delta\sin\theta'_1 = d_3(\mathbf{k}_3^{T C}\mathbf{b}_3)\end{aligned} \tag{63}$$

Using the fact that $d_3 > 0$ and $\delta > 0$, we select the sign of $\sin\theta'_1$ to be the same as that of $\mathbf{k}_3^{T C}\mathbf{b}_3$.

References

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

[2] G. Cardano, T. R. Witmer, and O. Ore. *The Rules of Algebra: Ars Magna*, volume 685. Courier Corporation, 2007.

[3] D. A. Cox, J. Little, and D. O'Shea. *Using algebraic geometry*, volume 185. Springer Science & Business Media, 2006.

[4] S. Finsterwalder and W. Scheufele. *Das rückwärtseinschneiden im raum*. Verlag d. Bayer. Akad. d. Wiss., 1903.

[5] M. A. Fischler and R. 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.

[6] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng. Complete solution classification for the perspective-three-point problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):930–943, 2003.

[7] E. W. Grafarend, P. Lohse, and B. Schaffrin. Dreidimensionaler rückwärtsschnitt teil i: Die projektiven gleichungen. *Zeitschrift für Vermessungswesen*, pages 1–37, 1989.

[8] J. A. Grunert. Das pothenotische problem in erweiterter gestalt nebst über seine anwendungen in der geodäsie. *Grunerts archiv für mathematik und physik*, 1:238–248, 1841.

[9] R. M. Haralick, H. Joo, C.-N. Lee, X. Zhuang, V. G. Vaidya, and M. B. Kim. Pose estimation from corresponding point data. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(6):1426–1446, 1989.

[10] R. M. Haralick, D. Lee, K. Ottenburg, and M. Nolle. Analysis and solutions of the three point perspective pose estimation problem. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 592–598, Lahaina, HI, June 3–6 1991.

[11] J. A. Hesch and S. I. Roumeliotis. A direct least-squares (DLS) method for PnP. In *Proc. of the 13th International Conference on Computer Vision*, pages 383–390, Barcelona, Spain, Nov. 6–13 2011.

[12] B. K. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, 1987.

[13] B. K. Horn, H. M. Hilden, and S. Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *Journal of the Optical Society of America A*, 5(7):1127–1135, 1988.

[14] T. Ke and S. I. Roumeliotis. An efficient algebraic solution to the perspective-three-point problem. Available: `https://arxiv.org/abs/1701.08237`.

[15] 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. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2969–2976, Colorado Springs, CO, June 21–25 2011.

[16] D. Koks. A roundabout route to geometric algebra. *Explorations in Mathematical Physics: The Concepts behind an Elegant Language*, pages 147–184, 2006.

[17] S. Linnainmaa, D. Harwood, and L. S. Davis. Pose determination of a three-dimensional object using triangle pairs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5):634–647, 1988.

[18] A. Masselli and A. Zell. A new geometric approach for faster solving the perspective-three-point problem. In *Proc. of the IEEE International Conference on Pattern Recognition*, pages 2119–2124, Stockholm, Sweden, Aug. 24–28 2014.

[19] E. Merritt. Explicit three-point resection in space. *Photogrammetric Engineering*, 15(4):649–655, 1949.

[20] D. Nistér and H. Stewénius. A minimal solution to the generalised 3-point pose problem. *Journal of Mathematical Imaging and Vision*, 27(1):67–79, 2007.

[21] OpenCV. Open Source Computer Vision Library. Available: `http://opencv.org/`. Accessed Apr. 10, 2017.

[22] L. Quan and Z. Lan. Linear n-point camera pose determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):774–780, 1999.

[23] TooN. C++ Linear Algebra Library. Available: `https://www.edwardrosten.com/cvd/toon/html-user/`. Accessed Apr. 10, 2017.

[24] W. Wen-Tsun. Basic principles of mechanical theorem proving in elementary geometries. *Journal of Automated Reasoning*, 2(3):221–252, 1986.

[25] T. J. Ypma. Historical development of the Newton-Raphson method. *SIAM review*, 37(4):531–551, 1995.