A Minimal Closed-Form Solution for Multi-perspective Pose Estimation using Points and Lines

Pedro Miraldo[1(✉)], Tiago Dias[1], and Srikumar Ramalingam[2]

[1] Instituto Superior Técnico, Lisboa, Portugal
{pedro.miraldo,tiagojdias,}@tecnico.ulisboa.pt
[2] School of Computing, University of Utah, Salt Lake City, USA
srikumar@cs.utah.edu

Abstract. We propose a minimal solution for pose estimation using both points and lines for a multi-perspective camera. In this paper, we treat the multi-perspective camera as a collection of rigidly attached perspective cameras. These type of imaging devices are useful for several computer vision applications that require a large coverage such as surveillance, self-driving cars, and motion-capture studios. While prior methods have considered the cases using solely points or lines, the hybrid case involving both points and lines has not been solved for multi-perspective cameras. We present the solutions for two cases. In the first case, we are given 2D to 3D correspondences for two points and one line. In the later case, we are given 2D to 3D correspondences for one point and two lines. We show that the solution for the case of two points and one line can be formulated as a fourth degree equation. This is interesting because we can get a closed-form solution and thereby achieve high computational efficiency. The later case involving two lines and one point can be mapped to an eighth degree equation. We show simulations and real experiments to demonstrate the advantages and benefits over existing methods.

Keywords: Multi-perspective camera · Pose estimation · Points Lines

1 Introduction

Pose estimation is a fundamental problem that is used in a wide variety of applications such as image-based localization (complementary to global positioning units that are prone to suffer from multi-path effects), augmented/virtual reality, surround-view and birds-eye view synthesis from a car-mounted multi-camera system, and telemanipulation of robotic arms. The basic idea of pose estimation

The original version of this chapter was revised: An error in Eq. 5 was corrected. The correction to this chapter is available at https://doi.org/10.1007/978-3-030-01270-0_51

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-01270-0_29) contains supplementary material, which is available to authorized users.

V. Ferrari et al. (Eds.): ECCV 2018, LNCS 11220, pp. 490–507, 2018.
https://doi.org/10.1007/978-3-030-01270-0_29

is to recover the camera position and orientation with respect to some known 3D object in the world. We typically associate a world coordinate frame to the 3D object and pose estimation denotes the computation of the rigid transformation between world frame and the camera coordinate frame. The problem setting for pose estimation is generally the same. We are given the correspondences between 2D features (points or lines) and 3D features for a calibrated camera, and the goal is to compute the rigid transformation between the camera and the world.

It is relatively easier to develop non-minimal solutions, which utilize more than the minimum number of correspondences, for pose estimation problems. However, non-practitioners of multi-view geometry algorithms may ponder over the following questions. Is it really necessary to develop a complex algorithm to use the minimum number of features? What is the need for hybrid algorithms [1,2] that utilize both point and line features? In practice with noisy data with outliers, non-minimal solvers produce inferior results compared to minimal solvers. For example, in a challenging pose estimation scenario involving crowded roads and dynamic obstacles, we face the problem of having a large number of incorrect feature correspondences. Having the flexibility of using point or line correspondences improves the robustness of the algorithms. While there has already been several solvers, three main factors can be used to distinguish one solver from another: minimal/non-minimal, features, and camera system. For example, we could think about a pose estimation algorithm for a central camera system that is minimal and uses only point correspondences. Table 1 lists several minimal solvers for different types of camera systems and features.

Table 1. List of minimal pose problems, for perspective, multi-perspective, and general camera models, using both points and/or lines.

Minimal problem	#Points/Lines	#Solutions	Closed-form	Papers
Perspective with points	3/0	4	Yes	[3–7]
Perspective with lines	0/3	8	No	[8,9]
Perspective with points and lines	2/1	4	Yes	[1]
Perspective with points and lines	1/2	8	No	[1]
Multi-perspective with points	3/0	8	No	[10]
Multi-perspective with lines	0/3	8	No	[11]
Multi-perspective with points and lines	2/1	4	Yes	Ours
Multi-perspective with points and lines	1/2	8	No	Ours
General camera with points	3/0	8	No	[12–14]

Minimal solvers for central cameras: The minimal camera pose solver using 3 point correspondences gives up to four solutions and can be computed in closed-form [3–7]. On the other hand, using 3 line correspondences, we get 8 solutions [8, 9], requiring the use of slower iterative methods. In [1], two mixed scenarios were considered: (1) 2 points and 1 line yielding 4 solutions in closed-form; and (2) 1 point and 2 lines yielding 8 solutions, requiring the use of iterative methods. Non-minimal solvers using both points and lines have also been studied [15,16].

Minimal solvers for generalized cameras: The general camera model [17–20] is represented by the individual association between unconstrained 3D projection rays and pixels, i.e. when projection rays may not intersect at a single 3D point in the world. This problem was addressed for the pose estimation using three 3D points and their 2D correspondences [12–14]. On the other hand, no solutions were yet proposed for the case of using 3D straight lines and their images, neither the case of using the combination of points and lines. There are non-minimal solvers using both points and lines [21,22].

Minimal solvers for multi-perspective cameras: We refer to multi-perspective camera as a system that models multiple perspective cameras that are rigidly attached with respect to each other. Examples include stereo cameras, multi-camera system mounted on a car for surround-view capture, etc. While multi-perspective camera systems are non-central and can be treated as generalized cameras, they are not completely unconstrained. In both perspective and multi-perspective systems, 3D lines project as 2D lines and lead to interpretation "planes". The minimal solvers for multi-perspective cameras have been addressed independently for points [10] and lines [11]. In this paper, we propose a novel solution for pose estimation for multi-perspective cameras using both points and lines. We are not aware of any prior work that solves this problem. Both 3D point and line correspondences provide two degrees of freedom (as shown in [3–7,10,12–14] for points and [8,9,11] for lines). Since we have 6 DOF, to compute the camera pose we need at least three lines and/or points[1].

The pose estimation has also been studied under other settings [26–38]. The main contributions of this paper are summarized below:

- We present two minimal pose estimation solvers for a multi-perspective camera system given 2D and 3D correspondences (Sect. 3 and Fig. 1):
  1. Using 2 points and 1 line, we get 4 solutions in closed-form; and
  2. Using 1 point and 2 lines, we get 8 solutions.
- The proposed solvers using both points and lines produce comparable or superior results to the ones that employ solely points or lines (Sect. 4);
- While most prior methods require iterative solutions (See Table 1), 2 points and 1 line correspondences yield an efficient closed-form solver (very useful for real-world applications such as self-driving); and
- We demonstrate a standalone SLAM system using the proposed solvers for large-scale reconstruction (Sect. 4).

2 Problem Statement

Our goal is to solve the minimal pose estimation for multi-perspective cameras using both points and lines. First, we present the general pose problem using points or lines (Sect. 2.1) and, then we define the minimal case studied in this paper (Sect. 2.2).

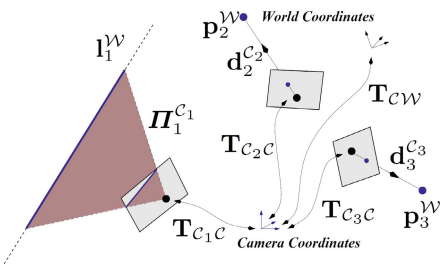
[1] In other cases, such as catadioptric cameras 3D lines project as curves in images which, in theory, it is possible to get the 3D line parameters from a single image [23–25], providing therefore more degrees of freedom to solve the pose problem.

2.1 Camera Pose Using Points or Lines

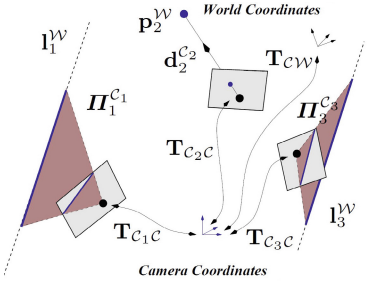
To distinguish between features in the world and the camera coordinate system we use $\mathcal{W}$ and $\mathcal{C}$, respectively. The camera pose is given by the estimation of the rotation matrix $\mathbf{R}_{\mathcal{CW}} \in \mathcal{SO}(3)$ and the translation vector $\mathbf{t}_{\mathcal{CW}} \in \mathbb{R}^3$ that define the rigid transformation between the camera and world coordinate systems:

$$\mathbf{T}_{\mathcal{CW}} \in \mathbb{R}^{4\times 4} = \begin{bmatrix} \mathbf{R}_{\mathcal{CW}} & \mathbf{t}_{\mathcal{CW}} \\ \mathbf{0}_{1,3} & 1 \end{bmatrix}. \tag{1}$$

A multi-perspective camera is seen as a collection of individual perspective cameras rigidly mounted with respect to each other. We use $\mathcal{C}_i$ to denote the features in the i^{th} perspective camera. The transformations between the perspective cameras and the global camera coordinate system are known, i.e. $\mathbf{T}_{\mathcal{C}_i\mathcal{C}}$ is known, for all i. Next, we define the pose for the multi-perspective system.



(a) Two points and one line



(b) One point and two lines.

Fig. 1. Illustration of the two minimal problems solved in this paper. We estimate the transformation parameters $\mathbf{T}_{\mathcal{CW}}$ in two different settings: the case where we have two 3D points, one 3D line, and their respective images (a); and the case where we have two 3D lines, one 3D point and their respective images (b).

(1) Camera Pose using 3D Points: For a set of 3D points $\mathbf{p}_j^{\mathcal{W}}$ and their respective images, since the camera parameters are known, the pose for a multi-perspective camera is given by $\mathbf{T}_{\mathcal{CW}}$, for a $\{\mathbf{p}_j^{\mathcal{W}} \mapsto \mathbf{d}_j^{\mathcal{C}_i}\}$ for $j = 1, \ldots, N$, where $\mathbf{d}_j^{\mathcal{C}_i} \in \mathbb{R}^3$ is the inverse projection direction, given by the image of $\mathbf{p}_j^{\mathcal{W}}$ seen in the camera $\mathcal{C}_i$ [39,40]. Formally, the pose is given by the $\mathbf{T}_{\mathcal{CW}}$, such that

$$\mathbf{T}_{\mathcal{CW}} \begin{bmatrix} \delta_j \ \mathbf{R}_{\mathcal{C}_i\mathcal{C}} \ \mathbf{d}_j^{\mathcal{C}_i} + \mathbf{c}_i^{\mathcal{C}} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_j^{\mathcal{W}} \\ 1 \end{bmatrix} \quad \text{for all } j = 1, \ldots, N, \tag{2}$$

where δ_j is an unknown depth of $\mathbf{p}_j^{\mathcal{C}_i}$, w.r.t. the camera center $\mathbf{c}_i^{\mathcal{C}} \in \mathbb{R}^3$.

(2) Camera Pose using 3D Lines: To represent 3D straight lines in the world we use *Plücker* coordinates [41], i.e. $\mathbf{l}_j^{\mathcal{W}} \dot{\sim} (\bar{\mathbf{l}}_j^{\mathcal{W}}, \tilde{\mathbf{l}}_j^{\mathcal{W}})$ where $\bar{\mathbf{l}}_j^{\mathcal{W}}, \tilde{\mathbf{l}}_j^{\mathcal{W}} \in \mathbb{R}^3$ are the line's direction and moment, respectively. Since the camera parameters are

known, their respective images can be represented by an interpretation plane $\mathbf{\Pi}_j^{\mathcal{C}_i} \in \mathbb{R}^4 = (\bar{\boldsymbol{\pi}}_j^{\mathcal{C}_i}, \check{\pi}_j^{\mathcal{C}_i})$ [39,40], where $\bar{\boldsymbol{\pi}}_j^{\mathcal{C}_i}$ is the normal vector to the plane and $\check{\pi}_j^{\mathcal{C}_i}$ is its distance to the origin of the respective coordinate system, which in this case is equal to zero (the interpretation plane passes through the center of the camera $\mathcal{C}_i$). Under the correct pose the 3D line lies on the interpretation plane formed by the corresponding 2D line and the camera center. Thus the required pose using lines is given by $\mathbf{T}_{\mathcal{C}\mathcal{W}}$ for a $\{\mathbf{l}_j^{\mathcal{W}} \mapsto \mathbf{\Pi}_j^{\mathcal{C}_i}\}$, such that

$$\underbrace{\begin{bmatrix} \hat{\bar{\mathbf{l}}}_j^{\mathcal{W}} & \bar{\mathbf{l}}_j^{\mathcal{W}} \\ \bar{\mathbf{l}}_j^{\mathcal{W}\,T} & 0 \end{bmatrix}}_{\mathbf{L}_j^{\mathcal{W}} \in \mathbb{R}^{4\times 4}} \mathbf{T}_{\mathcal{C}\mathcal{W}}^{-T}\, \mathbf{T}_{\mathcal{C}_i\mathcal{C}}^{-T}\, \mathbf{\Pi}_j^{\mathcal{C}_i} = \mathbf{0}, \quad \text{for all } j = 1, \ldots, N, \tag{3}$$

where: $\mathbf{L}_j^{\mathcal{W}}$ is the *Plücker* matrix of the line $\mathbf{l}_j^{\mathcal{W}}$ [41]; the hat represents skew-symmetric matrix that linearizes the external product, such that $\mathbf{a} \times \mathbf{b} = \hat{\mathbf{a}}\mathbf{b}$; and N is the number of correspondences between 3D lines in the world and their respective interpretation planes.

2.2 Minimal Pose Using Points and Lines

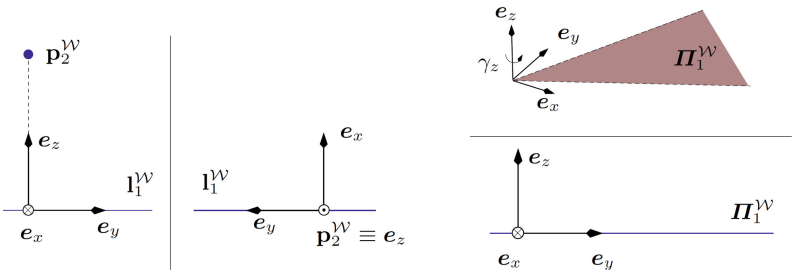
Similar to the cases of using only points or lines, the minimal pose is computed by having three of these features in the world, and their respective images. This means that, for the minimal pose addressed in this paper, and according to Sect. 1, there are two cases:

- The estimation of $\mathbf{T}_{\mathcal{C}\mathcal{W}}$, knowing: $\mathbf{l}_1^{\mathcal{W}} \mapsto \mathbf{\Pi}_1^{\mathcal{C}_1}$; $\mathbf{p}_2^{\mathcal{W}} \mapsto \mathbf{d}_2^{\mathcal{C}_2}$; $\mathbf{p}_3^{\mathcal{W}} \mapsto \mathbf{d}_3^{\mathcal{C}_3}$; and $\mathbf{T}_{\mathcal{C}_i\mathcal{C}}$ for $i = 1, 2, 3$. A graphical representation of this problem is shown in Fig. 1(a); and
- The estimation of $\mathbf{T}_{\mathcal{C}\mathcal{W}}$, knowing: $\mathbf{l}_1^{\mathcal{W}} \mapsto \mathbf{\Pi}_1^{\mathcal{C}_1}$; $\mathbf{p}_2^{\mathcal{W}} \mapsto \mathbf{d}_2^{\mathcal{C}_2}$; and $\mathbf{l}_3^{\mathcal{W}} \mapsto \mathbf{\Pi}_3^{\mathcal{C}_3}$; and $\mathbf{T}_{\mathcal{C}_i\mathcal{C}}$ for $i = 1, 2, 3$. This problem is depicted in Fig. 1(b).

We show the solutions to these minimal problems in the next section.

3 Solution to the Minimal Pose Problem Using Points and Lines

As a first step, we transform the world and camera coordinate systems using predefined transformations to the data (Sect. 3.1). Note that we can first compute the pose in this new coordinate systems, and then recover the real pose in the original coordinate frames by using the inverse of the predefined transformations. The use of such predefined transformations can greatly simplify the underlying polynomial equations and enable us to develop low-degree polynomial solutions.



(a) World coordinate syste (b) Camera coordinate system

Fig. 2. Depiction of the selected world and camera coordinate systems. (a) shows the considered world coordinate system, while (b) presents the selected camera coordinate system. We note that while the world coordinate system is uniquely defined, the camera coordinate system can be defined up to a z–axis rotation.

3.1 Select the World and Camera Coordinate Systems

Let us consider initial transformations such that the data in the world coordinate system verify the following specifications:

- Centered on the $\mathbf{l}_1^{\mathcal{W}}$;
- With the y–axis aligned with the line's direction; and
- Such that $\mathbf{p}_2^{\mathcal{W}} = \begin{bmatrix} 0 & 0 & * \end{bmatrix}^T$, where $*$ takes any value in $\mathbb{R}$.

A graphical representation of these specifications is shown in Fig. 2(a). Regarding the camera coordinate system, we aim at having the following specifications:

- Centered in the $\mathcal{C}_1$; and
- With the z–axis aligned with the interpretation plane normal.

A graphical representation of the camera's coordinate system is shown in Fig. 2(b) . The predefined transformation can be computed easily and the details are shown in the supplementary material. In the following subsections we present our solutions to the minimal case.

3.2 Solution Using Two 3D Points and a 3D Straight Line

Here, we present a closed-form solution using 2 points and 1 line. The minimal pose is computed using the coplanarity constraint on the 3D line and its associated interpretation plane (3), and using collinearity constraint associated with the point correspondences (2).

From the selected coordinate systems (Sect. 3.1), the rotation between the camera and world coordinate systems is given by

$$\mathbf{R}_{\mathcal{CW}} = \begin{bmatrix} c\theta & 0 & -s\theta \\ 0 & 1 & 0 \\ s\theta & 0 & c\theta \end{bmatrix} \begin{bmatrix} c\alpha & s\alpha & 0 \\ -s\alpha & c\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4}$$

with unknowns $c\theta$ & $s\theta^2$ and $c\alpha$ & $s\alpha$. One can notice that, as a result of the predefined transformations, we have reduced one degree of freedom on the rotation matrix. In addition, one has

$$\mathbf{R}_{\mathcal{CW}}^T \mathbf{t}_{\mathcal{CW}} = \begin{bmatrix} * \\ * \\ 0 \end{bmatrix}, \text{where } * \text{ can take any value in } \mathbb{R}. \tag{5}$$

Now, let us consider the effect of collinearity associated with the first point correspondence. From (2), we have

$$\mathbf{T}_{\mathcal{CW}} \left(\delta_2 \mathbf{d}_2^{\mathcal{C}} + \mathbf{c}_2^{\mathcal{C}} \right) = \mathbf{p}_2^{\mathcal{W}}, \tag{6}$$

where $\mathbf{d}_2^{\mathcal{C}} = \mathbf{R}_{\mathcal{C}_2\mathcal{C}} \mathbf{d}_2^{\mathcal{C}_2}$. Then, this can be solved as a function of the unknowns t_1, t_2, t_3, resulting in

$$t_1 = \kappa_1^3[c\theta, s\theta, c\alpha, s\alpha, \delta_2]; \tag{7}$$

$$t_2 = \kappa_2^2[c\alpha, s\alpha, \delta_2]; \text{and} \tag{8}$$

$$t_3 = \kappa_3^3[c\theta, s\theta, c\alpha, s\alpha, \delta_2], \tag{9}$$

where $\kappa_i^j[.]$ denotes the i^{th} polynomial equation, with degree j. The analytic representation of all coefficients is sent in the supplementary material.

Next, we take into account the effects of the second point:

$$\mathbf{T}_{\mathcal{CW}} \left(\delta_3 \mathbf{d}_3^{\mathcal{C}} + \mathbf{c}_3^{\mathcal{C}} \right) = \mathbf{p}_3^{\mathcal{W}}. \tag{10}$$

Replacing the unknowns t_1, t_2, and t_3 in (10) by the results of (7)–(9), we get three constraints on the unknowns θ, α, δ_2, and δ_3, such that

$$\kappa_4^3[c\theta, s\theta, c\alpha, s\alpha, \delta_2, \delta_3] = 0; \tag{11}$$

$$\kappa_5^2[c\alpha, s\alpha, \delta_2, \delta_3] = 0; \text{ and} \tag{12}$$

$$\kappa_6^3[c\theta, s\theta, c\alpha, s\alpha, \delta_2, \delta_3] = 0. \tag{13}$$

In addition, considering the third row of the constraint defined in (5) and replacing t_1, t_2, and t_3 in this equation by the results of (7)–(9), we obtain the following constraint

$$\kappa_7^3[c\theta, s\theta, \delta_2] = 0. \tag{14}$$

Now, solving (11)–(13), and (14) as a function of $c\theta$, $s\theta$, $c\alpha$, and $s\alpha$, we get

$$c\theta = \frac{\kappa_8^1[\delta_2]}{\kappa_9^2[\delta_2, \delta_3]}, \; s\theta = \frac{\kappa_{10}^1[\delta_2, \delta_3]}{\kappa_{11}^2[\delta_2, \delta_3]}, \; c\alpha = \frac{\kappa_{12}^2[\delta_2, \delta_3]}{\kappa_{13}^2[\delta_2, \delta_3]}, \text{ and } s\alpha = \frac{\kappa_{14}^2[\delta_2, \delta_3]}{\kappa_{15}^2[\delta_2, \delta_3]}. \tag{15}$$

which we replace in the trigonometric relations $c\theta^2 + s\theta^2 - 1 = 0$ and $c\alpha^2 + s\alpha^2 - 1 = 0$, getting two constraints of the form

$$\frac{\kappa_{16}^2[\delta_2, \delta_3]}{\kappa_{17}^2[\delta_2, \delta_3]} = 0 \quad \text{and} \quad \frac{\kappa_{18}^2[\delta_2, \delta_3]}{\kappa_{19}^2[\delta_2, \delta_3]} = 0. \tag{16}$$

[2] To simplify, we denote $\cos(\theta)$ as $c\theta$ and $\sin(\theta)$ as $s\theta$, respectively.

However, solving the above equations as a function of the unknowns δ_2 and δ_3 is the same as solving

$$\kappa_{16}^2[\delta_2, \delta_3] = \kappa_{18}^2[\delta_2, \delta_3] = 0, \tag{17}$$

that corresponds to the estimation of the intersection points between two quadratic curves which, according to *Bézout's* theorem [42], has four solutions. There are many generic solvers in the literature to compute these solutions (such as [43–47]). However, since we are dealing with very simple polynomial equations, we derive our own fourth degree polynomial. From (17), solving one polynomial as a function of δ_2 and replacing these results in the other (the square root is removed using simple algebraic manipulations), we get

$$\kappa_{19}^4[\delta_2] = 0 \quad \text{and} \tag{18}$$

$$\delta_3 = \frac{\kappa_{20}^1[\delta_2] \pm \sqrt{\kappa_{21}^2[\delta_2]}}{\kappa_{22}^1[\delta_2]}. \tag{19}$$

Details on these derivations are provided in the supplementary material. Finally, to compute the pose, one has to solve (18), which can be computed in a closed-form (using Ferrari's formula), getting up to four real solutions for δ_2. Then, by back-substituting δ_2 in (19) we get the respective solutions for δ_3 (notice that from the two possible solutions for δ_3 one will be trivially ignored, since (17) can have only up to four solutions). The pair $\{\delta_2, \delta_3\}$ is afterwards used in (15) to compute the respective $\{c\theta, s\theta, c\alpha, s\alpha\}$, and then in (7)–(9) to estimate $\{t_1, t_2, t_3\}$.

3.3 Solution Using two 3D Straight Lines and a 3D Point

This subsection presents the solution to the multi-perspective pose problem using 2 lines and 1 point. As before, we consider the predefined transformations to the input data defined in Sect. 3.1, which already includes the coplanarity constraint associated with the first 3D line. Under these assumptions, we start by considering the collinearity constraint associated with the 3D point and its respective image (2) and, then, use the coplanarity constraint of the second 3D line and its respective interpretation plane (3).

We start by using the same steps of Sect. 3.2, i.e. we get the translation parameters as a function of $c\theta$, $s\theta$, $c\alpha$, $s\alpha$, and δ_2, which are given by (7)–(9). Then, we replace the translation parameters in the third row of (5) by the results of (7)–(9), which gives (14). Afterwards, we solve (14) and the trigonometric constraint $c\theta^2 + s\theta^2 - 1 = 0$, as a function of $c\theta$ and $s\theta$, resulting in

$$c\theta = \kappa_{23}^1[\delta_2] \quad \text{and} \quad s\theta = \pm\sqrt{\kappa_{24}^2[\delta_2]}. \tag{20}$$

Now, we consider the constraints associated with the second line which, since $\mathbf{T}_{\mathcal{C}_3\mathcal{C}}$ is known, is given by

$$\mathbf{L}_3^{\mathcal{W}} \, \mathbf{T}_{\mathcal{C}\mathcal{W}}^{-T} \, \mathbf{\Pi}_3^{\mathcal{C}} = \mathbf{0}, \tag{21}$$

where $\mathbf{\Pi}_3^{\mathcal{C}} = \mathbf{T}_{\mathcal{C}_3\mathcal{C}}^{-T}\,\mathbf{\Pi}_3^{\mathcal{C}_3}$. Replacing the translation parameters in the above equations by the results of (7)–(9), and $c\theta$ by the outcome of (20) (notice that, for now we keep the unknown $s\theta$), we get four polynomial equations with degree two, as a function of variables δ_2, $s\alpha$, $c\alpha$, and $s\theta$. Solving two of them as a function of $c\alpha$ and $s\alpha$, we get

$$c\alpha = \frac{\kappa_{25}^2[s\theta,\delta_2]}{\kappa_{26}^1[s\theta,\delta_2]} \quad \text{and} \quad s\alpha = \frac{\kappa_{27}^2[s\theta,\delta_2]}{\kappa_{28}^1[s\theta,\delta_2]}. \tag{22}$$

Now, replacing these results into the trigonometric relation $c\alpha^2 + s\alpha^2 - 1 = 0$, we get a constraint of the form

$$\frac{\kappa_{29}^4[s\theta,\delta_2]}{\kappa_{30}^2[s\theta,\delta_2]} = 0 \quad \Rightarrow \quad \kappa_{29}^4[s\theta,\delta_2] = 0. \tag{23}$$

Notice that, from (20), the expression that defines $s\theta$, as a function of δ_2, has a square root of a polynomial equation. Then, starting from (23), we simplify the problem by: (1) taking the terms with $s\theta$ to the right side of the equation:

$$\kappa_{29}^4[s\theta,\delta_2] = 0 \Rightarrow s\theta^4 + \kappa_{31}^2[\delta_2]s\theta^2 + \kappa_{32}^4[\delta_2] = -\left(\kappa_{33}^1[\delta_2]s\theta^2 + \kappa_{34}^3[\delta_2]\right)s\theta; \tag{24}$$

(2) squaring both sides & moving all the terms to the left side of the equation:

$$s\theta^8 + \kappa_{35}^2[\delta_2]s\theta^6 + \kappa_{36}^6[\delta_2]s\theta^4 + \kappa_{37}^6[\delta_2]s\theta^2 + \kappa_{38}^8[\delta_2] = 0; \tag{25}$$

and, finally, (3) replacing $s\theta$ using (20) (notice that the square root and the $\pm$ signal is removed), we get

$$\kappa_{39}^8[\delta_2] = 0, \tag{26}$$

which as up to eight real solutions. To get the pose: (1) we compute δ_2 from the real roots of (26); (2) for each δ_2, we get $\{c\theta, s\theta\}$ from (20); (3) we compute $\{c\alpha, s\alpha\}$ from (22); and (4) by back-substituting all these unknowns, we get $\{t_1, t_2, t_3\}$ from (7)–(9), obtaining the estimation of the camera pose.

4 Experimental Results

In these experiments, we consider the methods proposed in Sect. 3 and existing multi-perspective algorithms to solve the pose using three points [10] or three lines [11]. All algorithms were implemented in `MATLAB` and are available in the author's website.

We start by using synthetic data (Sect. 4.1): (1) we evaluate the number of real solutions and analyze their computational complexity; and (2) we test each method with noise. Next, we show results using real data: (1) we evaluate the minimal solutions in a RANSAC framework (Sect. 4.2); and (2) we use each method in a 3D path reconstruction using a real multi-perspective camera (Sect. 4.3).

4.1 Results with Synthetic Data

To get the data, we randomly define the ground truth camera pose, $\mathbf{T}_{GT}$. Three perspective cameras were generated (randomly distributed in the environment) in which their position w.r.t the camera coordinate system is assumed to be known, $\mathbf{T}_{\mathcal{C}_i\mathcal{C}}$. Then, for each camera $\mathcal{C}_i$, we define a feature in the world, and their projection into the image:

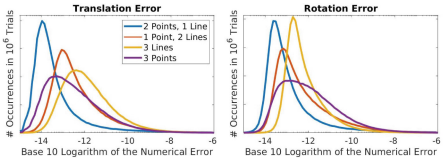
$\mathbf{p}_i^{\mathcal{W}} \mapsto \mathbf{d}_i^{\mathcal{C}_i}$**:** Points in the world $\mathbf{p}_i^{\mathcal{W}}$ are projected into the image $\mathbf{u}_i^{\mathcal{I}_i}$, by using a predefined calibration matrix. We had noise in the image pixels, and get the corresponding 3D inverse projection direction $\mathbf{d}_i^{\mathcal{C}_i}$.

$\mathbf{l}_i^{\mathcal{W}} \mapsto \mathbf{\Pi}_i^{\mathcal{C}_i}$**:** 3D points defining the edges of the 3D line $\mathbf{l}_i^{\mathcal{W}}$ are projected into the image, $\{\mathbf{u}_{1,i}^{\mathcal{I}_i}, \mathbf{u}_{2,i}^{\mathcal{I}_i}\}$. To each image point of the edge, we add noise (as we did in the previous point) and compute the respective inverse projection directions $\{\mathbf{d}_{1,i}^{\mathcal{C}_i}, \mathbf{d}_{2,i}^{\mathcal{C}_i}\}$. The interpretation plane is given by $\mathbf{\Pi}_i^{\mathcal{C}_i} = \begin{bmatrix}\mathbf{d}_{1,i}^{\mathcal{C}_i} \times \mathbf{d}_{2,i}^{\mathcal{C}_i} & 0\end{bmatrix}$.

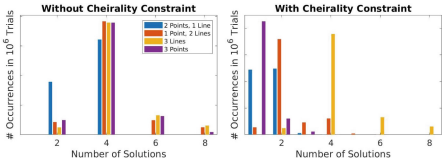
After getting the data, we apply the known transformations $\mathbf{T}_{\mathcal{C}_i\mathcal{C}}$ to obtain the corresponding features in the global camera coordinate system, such that

$\mathbf{p}_i^{\mathcal{W}} \mapsto \mathbf{c}_i^{\mathcal{C}} + \delta_i \mathbf{d}_i^{\mathcal{C}}$ in which $\mathbf{c}_i^{\mathcal{C}}$ is the perspective camera center and $\mathbf{d}_i^{\mathcal{C}} = \mathbf{R}_{\mathcal{C}_i\mathcal{C}}\mathbf{d}_i^{\mathcal{C}_i}$.

$\mathbf{l}_i^{\mathcal{W}} \mapsto \mathbf{\Pi}_i^{\mathcal{C}}$ in which $\mathbf{\Pi}_i^{\mathcal{C}} = \mathbf{T}_{\mathcal{C}_i\mathcal{C}}^{-T}\mathbf{\Pi}_i^{\mathcal{C}_i}$.



(a) Numerical Results.



(b) Number of Occurrences.

Algorithm	2 Points & 1 Line	1 Point & 2 Lines	3 Points [10]	3 Lines [11]
Total	$151.3s$	$516.6s$	$512.2s$	$1481.7s$
Median	$135\mu s$	$439\mu s$	$480\mu s$	$1161\mu s$

(c) The total and median of the computation time for solving the camera poses.

Fig. 3. Results obtained with numerical errors for the pose estimation, using the methods proposed in this paper (`2 Points, 1 Line` and `1 Point, 2 Lines`) and existing solutions for points (`3 Points`) and lines (`3 Lines`).

For the evaluation, we start by running an experiment in which we consider 10^6 random trials, without adding noise in the image pixels. We use both methods presented in this paper, as well as the algorithms presented in [10,11]. For each trial/method, in which $\mathbf{R}_{\mathcal{CW}}$ and $\mathbf{t}_{\mathcal{CW}}$ are the estimated rotation and translation parameters, we: (1) compute the relative rotation that caused the

deviation of the estimated rotation w.r.t. the ground-truth $\Delta\mathbf{R} = \mathbf{R}_{\mathcal{CW}}\mathbf{R}_{GT}^{T}$, which can be represented by an axis-angle rotation, and the error is set as the respective angle in degrees; and (2) set $\|\mathbf{t}_{\mathcal{CW}} - \mathbf{t}_{GT}\|$ as the translation error[3]. Fig. 3(a) shows that the methods are very similar in terms of numerical evaluation.

Cheirality Constraint: All of the methods evaluated in this section produce multiple solutions for the pose. Our methods of Sects. 3.2 and 3.3 give up to four and eight solutions respectively. We discard imaginary solutions and the ones that are not physically realizable. The so-called *cheirality* constraint [40] restricts points behind the camera (this is only possible to check in the cases in which we use point correspondences). We obtain the result for the 10^6 trials with and without the *cheirality* constraint. Figure 3(b) shows that the number of valid solutions (with the *cheirality* constraint) is lower for the algorithms that use more points.

Computation Time: To conclude these tests, we present the evaluation of the computation time, required for each algorithm to compute all the 10^6 trials. In theory, the method presented in Sect. 3.2 is the fastest, since it is computed in closed-form. On the other hand, both our method presented in Sect. 3.3 and the case of three points [10] require the computation of the roots of an eighth degree polynomial equation, which requires iterative techniques. Moreover, the case of three lines [11] not only requires the computation of an eighth degree polynomial equation, but also the computation of the null-space of a 3×9 matrix, that also slows down the execution time. Results shown in Table 3(c) validate the above assumptions. Note that these timing analysis are done using `Matlab`, and porting the code to `C++` would produce further speedup.

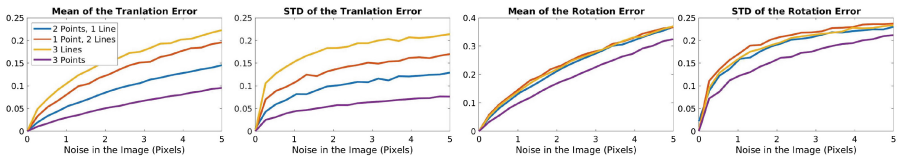


Fig. 4. Comparative results for the methods using different type of features, as a function of the noise in the image pixels. `2 Points, 1 Line` and `1 Point, 2 Lines` show the results for the methods presented in this paper, while `3 Points` and lines `3 Lines` are techniques proposed in [10,11].

Next, we evaluate the robustness of the methods in terms of image noise. For that purpose, we consider the same data-set generated, but we add noise in the image varying from 0 to 5 pixels. For each level of noise, we get 10^3 random trials, compute the pose for all the four algorithms (notice the data required for each of the algorithms is different), and extract the average and standard

[3] For the cases in which the algorithms return multiple solutions, it was considered the cases with the smallest error using these metrics.

deviation for all the 10^3 trials for each level of noise. The results shown in Fig. 4 indicate that the algorithms that use more points are more robust to the noise.

4.2 Evaluating Minimal Solutions in a RANSAC Framework

For these experiments, using real data, we evaluate the results of the minimal solutions in a RANSAC framework [48,49]. Since we are using points and lines correspondences, one needs to define two metrics for the re-projection errors: (1) For points, we use the geometric distance between the known pixels and the re-projection of 3D points using the estimated camera pose; and (2) For lines, we use the result presented in [50] which, for a ground truth line $\mathbf{l}_{GT}$ and a re-projected line $\mathbf{l}$ (both in the image), is given by

$$d_L(\mathbf{l}_{GT}, \mathbf{l})^2 = \left(d_P(\mathbf{u}_1, \mathbf{l})^2 + d_P(\mathbf{u}_2, \mathbf{l})^2\right) \exp(2\angle(\mathbf{l}_{GT}, \mathbf{l})), \tag{27}$$

where $d_P(.)$ denotes the geometric distance between a point and line in the image, and $\mathbf{u}_1$ & $\mathbf{u}_2$ are the end points of $\mathbf{l}_{GT}$.

Then, we use a data-set from the ETH3D Benchmark [51]. The data-set gives us the calibration and poses from a set of cameras, and the 3D points and their correspondences in the images. To extract the 3D lines and their correspondences in the images, we use the camera calibration & pose parameters from the data-set and the `Line3D++` algorithm [50].

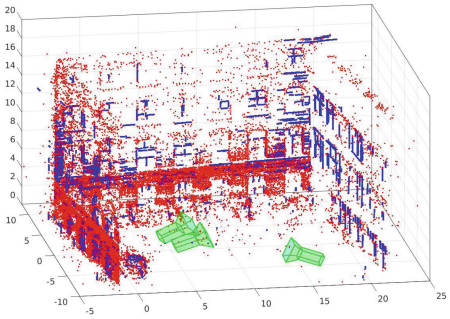
Examples of features in the image and its respective coordinates in the world are shown in Fig. 5 (a, b), respectively. We run two experiments with these data, using both our methods and [10,11], under a RANSAC framework. We start by defining a threshold for points and lines[4]. To fairly select these thresholds, we run [10,11] (that use solely points or lines respectively), and calibrate the values to ensure similar results in terms of errors as a function of the required number of inliers. We use these line and point thresholds in our techniques. Then, we vary the required number of inliers (a percentage of the all points and lines in the image), and for each we run the methods 10^4 times. In Fig. 5(c) we show the results for the errors (using the metrics presented in Sect. 4.1 for the translation and rotation errors), as a function of the percentage of inliers.

To conclude these experiments, we do some tests varying the threshold, for a fixed number of required inliers (in this case we consider 40 percent of the data), in a RANSAC framework. To vary the threshold, we start from the values indicated in the previous paragraph, and vary as a function of the percentage of the corresponding values. The results are shown in Fig. 5(d), for thresholds ranging from 50 to 150 percent of the original threshold.

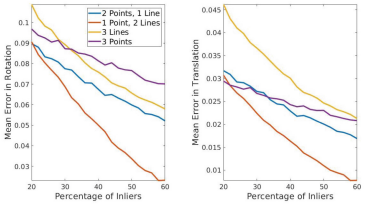
Positives: As it can be seen from the results of Fig. 5(c) and (d), for the threshold values previously defined, both methods using three points and three lines have similar results, and, when comparing to the results of our solutions (using 2 points & 1 line and 1 point & 2 lines) one can see that the errors on the rotation and translation parameters are in general significantly lower.

[4] Notice these thresholds must be different because of the differences between the metrics presented above.

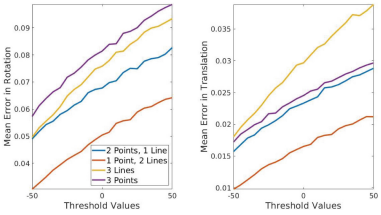
(a) Examples of images and 2D Data used in this experimental results (blue lines and red points).



(b) 3D Data set used in this experiments (blue lines and red points), and the estimated camera positions (green).

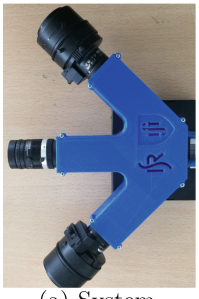


(c) Minimal solution in a RANSAC framework: varying the required number of inliers to stop the cycle.



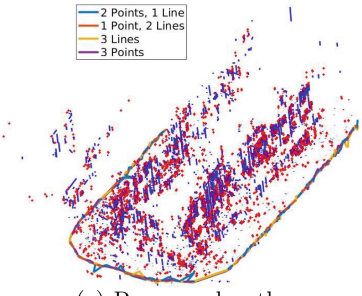
(d) Minimal solution in a RANSAC framework: varying the thresholds to stop the cycle.

Fig. 5. Evaluation of the proposed techniques and existing methods. As the evaluation criteria, we consider the required number of inliers and threshold to stop the RANSAC cycle. The errors in terms of rotation and translation parameters are afterwards computed, and compared between all the methods. (a) and (b) show three views and the 2D-3D data (points and lines) used in these experiments. (c) and (d) show the proposed evaluation.



(a) System.

(b) Sample images.



(c) Recovered path.

Fig. 6. Results of our methods in the path estimation, using a RANSAC framework. At the left, we show the used imaging device (three cameras with angles of 45 degrees between them). In the middle, we show two columns representing two sequences acquired at the same instance by our camera system. At the right, we show a reconstructed path obtained using all methods evaluated.

4.3 Path Reconstruction Using a Multi-perspective System

Using the presented methods, we demonstrate a 3D reconstruction pipeline for a multi-perspective camera. For that purpose, we use a real imaging device (see Fig. 6(a)), and acquired several images from an outdoor environment. We extract the correspondences between world and image features as follows:

- We get camera poses and correspondences between 3D points and image pixels using the `VisualSFM` framework [52,53]; and
- To get the line correspondences, we use the `Line3D++` algorithm, [50]. This method requires as input the camera positions, in which we use the poses given by the `VisualSFM` application.

Then, we calibrate each camera individually, using the `Matlab` calibration toolbox. The transformations parameters $\mathbf{T}_{\mathcal{C}_i\mathcal{C}}$ are given by the system's CAD model. Then, we run both methods proposed in this paper and existing solutions, using the RANSAC framework with a 30% of required inliers and thresholds used in the previous subsection. The data-set, including images, 3D-2D correspondences (for both lines and points) and camera system calibration are available in the author's website, as well as a video with the reconstructed paths for this experiment. A total of 606 images were taken from a path of around 200 meters (examples of these pictures are shown in Fig. 6(b)). An average of 130 lines and 50 points per image were used, within a total of 5814 3D lines and 2230 3D points in the world.

Figure 6(c) shows the results of the path reconstruction using various solvers, and they produce similar results.

5 Discussion

We present 2 minimal solvers for a multi-perspective camera: (a) using 2 points and 1 line yielding 4 solutions, and (b) using 2 lines and 1 point yielding 8 solutions. While the latter case requires iterative methods, the former can be solved efficiently in closed form. To the best of our knowledge, there is no prior work on using hybrid features for a multi-camera system. Note that existing solutions (i.e. using only points or lines) require the use of iterative techniques.

We show comparison with other minimal solvers, and we perform similar or superior to the ones that solely use points or lines. While the difference in performance among different minimal solvers can only be marginal, it is more important to note that these hybrid solvers can be beneficial and robust in noisy, dynamic, and challenging on-road scenarios where it is difficult to even get a few good correspondences. We also demonstrate a real experiment to recover the path of an outdoor sequence using a 3-camera system.

Our method can be seen as a generalization of existing pose solvers for central cameras that uses points and lines correspondences. If we set $\mathbf{T}_{\mathcal{C}_1\mathcal{C}} = \mathbf{T}_{\mathcal{C}_2\mathcal{C}} = \mathbf{T}_{\mathcal{C}_3\mathcal{C}}$, our method solves the problem of minimal problem for perspective cameras as the current state-of-the-art method.

Acknowledgments. P. Miraldo and T. Dias are with the Institute for Systems and Robotics (ISR/IST), LARSyS, Instituto Superior Técnico, University of Lisboa, Portugal. This work was partially supported by the Portuguese projects [UID/EEA/50009/2013] & [PTDC/EEI-SII/4698/2014] and grant [SFRH/BPD/111495/2015]. We thank the reviewers and ACs for valuable feedback.

References

1. Ramalingam, S., Bouaziz, S., Sturm, P.: Pose estimation using both points and lines for geo-localization. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 4716–4723 (2011)
2. Kuang, Y., Astrom, K.: Pose estimation with unknown focal length using points, directions and lines. In: IEEE International Conference on Computer Vision (ICCV), pp. 529–536 (2013)
3. Haralick, R., Lee, C.N., Ottenberg, K., Nolle, M.: Review and analysis of solutions of the three point perspective pose estimation problem. Int. J. Comput. Vis. (IJCV) **13**(3), 331–356 (1994)
4. Gao, X.S., Hou, X.R., Tang, J., Cheng, H.F.: Complete solution classification for the perspective-three-point problem. IEEE Trans. Pattern Anal. Mach. Intell. (T-PAMI) **25**(8), 930–943 (2003)
5. Kneip, L., Scaramuzza, D., Siegwart, R.: A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In: IEEE Conference on Computer Vision and Patter Recognition (CVPR), pp. 2969–2976 (2011)
6. Ke, T., Roumeliotis, S.I.: An efficient algebraic solution to the perspective-three-point problem. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4618–4626 (2017)
7. Wang, P., Xu, G., Wang, Z., Cheng, Y.: An efficient solution to the perspective-three-point pose problem. Comput. Vis. Image Underst. (CVIU) **166**, 81–87 (2018)
8. Dhome, M., Richetin, M., Lapreste, J.T., Rives, G.: Determination of the attitude of 3D objects from a single perspective view. IEEE Trans. Pattern Anal. Mach. Intell. (T-PAMI) **11**(12), 1265–1278 (1989)
9. Chen, H.H.: Pose determination from line-to-plane correspondences: existence condition and closed-form solutions. In: IEEE International Conference on Computer Vision (ICCV), pp. 374–378 (1990)
10. Lee, G.H., Lin, B., Pollefeys, M., Fraundorfer, F.: Minimal solutions for the multi-camera pose estimation problem. Int. J. Robot. Res. (IJRR) **34**(7), 837–848 (2015)
11. Lee, G.H.: A minimal solution for non-perspective pose estimation from line correspondences. In: European Conference on Computer Vision (ECCV), pp. 170–185 (2016)
12. Chen, C.S., Chang, W.Y.: On pose recovery for generalized visual sensors. IEEE Trans. Pattern Anal. Mach. Intell. **26**(7), 848–861 (2004)
13. Nister, D.: A minimal solution to the generalised 3-point pose problem. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1, pp. 560–567 (2004)
14. Miraldo, P., Araujo, H.: A simple and robust solution to the minimal general pose estimation. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 2119–2125 (2014)
15. Ansar, A., Daniilidis, K.: Linear pose estimation using points or lines. IEEE Trans. Pattern Anal. Mach. Intell. (T-PAMI) **25**(5), 578–589 (2003)

16. Vakhitov, A., Funke, J., Moreno-Noguer, F.: Accurate and linear time pose estimation from points and lines. In: European Conference on Computer Vision (ECCV), pp. 583–599 (2016)
17. Grossberg, M.D., Nayar, S.K.: A general imaging model and a method for finding its parameters. IEEE Int. Conf. Comput. Vis. (ICCV) **2**, 108–115 (2001)
18. Sturm, P., Ramalingam, S.: A generic concept for camera calibration. In: European Conference on Computer Vision (ECCV), pp. 1–13 (2004)
19. Miraldo, P., Araujo, H., Queiro, J.: Point-based calibration using a parametric representation of general imaging models. In: IEEE International Conference Computer Vision (ICCV), pp. 2304–2311 (2011)
20. Sturm, P., Ramalingam, S., Tardif, J.P., Gasparini, S., Barreto, J.: Camera models and fundamental concepts used in geometric computer vision. Found. Trends Comput. Graph. Comput. Vis. **2**(1–2), 1–183 (2011)
21. Schweighofer, G., Pinz, A.: Globally optimal O(n) solution to the pnp problem for general camera models. In: British Machine Vision Conference (BMVC), pp. 1–8 (2008)
22. Miraldo, P., Araujo, H., Gonalves, N.: Pose estimation for general cameras using lines. IEEE Trans. Cybern. **45**(10), 2156–2164
23. Caglioti, V., Gasparini, S.: On the localization of straight lines in 3D space from single 2D images. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1, pp. 1129–1134 (2005)
24. Swaminathan, R., Wu, A., Dong, H.: Depth from distortions. In: Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras (OMNIVIS) (2008)
25. Bermudez-Cameo, J., Barreto, J.P., Lopez-Nicolas, G., Guerrero, J.J.: Minimal solution for computing pairs of lines in non-central cameras. In: Asian Conference on Computer Vision (ACCV), pp. 585–597 (2014)
26. Camposeco, F., Cohen, A., Pollefeys, M., Sattler, T.: Hybrid pose estimation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 136–144 (2018)
27. Bujnak, M., Kukelova, Z., Pajdla, T.: New efficient solution to the absolute pose problem for camera with unknown focal length and radial distortion. In: Asian Conference on Computer Vision (ACCV), pp. 11–24 (2011)
28. Micusik, B., Wildenauer, H.: Minimal solution for uncalibrated absolute pose problem with a known vanishing point. In: IEEE International Conference on 3D Vision (3DV), pp. 143–150 (2013)
29. Kukelova, Z., Bujnak, M., Pajdla, T.: Real-time solution to the absolute pose problem with unknown radial distortion and focal length. In: IEEE International Conference on Computer Vision (ICCV), pp. 2816–2823 (2013)
30. Kuang, Y., Solem, J.E., Kahl, F., Astrom, K.: Minimal solvers for relative pose with a single unknown radial distortion. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 33–40 (2014)
31. Wu, C.: P3.5P: pose estimation with unknown focal length. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2440–2448 (2015)
32. Kukelova, Z., Bujnak, M., Pajdla, T.: Closed-form solutions to minimal absolute pose problems with known vertical direction. In: Asian Conference on Computer Vision (ACCV), pp. 216–229 (2010)
33. Sweeney, C., Flynn, J., Nuernberger, B., Turk, M., Hollerer, T.: Efficient computation of absolute pose for gravity-aware augmented reality. In: IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp. 19–24 (2015)

34. Camposeco, F., Sattler, T., Pollefeys, M.: Minimal solvers for generalized pose and scale estimation from two rays and one point. In: European Conference on Computer Vision (ECCV), pp. 202–218 (2016)
35. Sweeney, C., Fragoso, V., Höllerer, T., Turk, M.: gDLS: a scalable solution to the generalized pose and scale problem. In: European Conference on Computer Vision (ECCV), pp. 16–31 (2014)
36. Ventura, J., Arth, C., Reitmayr, G., Schmalstieg, D.: A minimal solution to the generalized pose-and-scale problem. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 422–429 (2014)
37. Haner, S., Astrom, K.: Absolute pose for cameras under flat refractive interfaces. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1428–1436 (2015)
38. Albl, C., Kukelova, Z., Pajdla, T.: R6P - rolling shutter absolute pose problem. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2292–2300 (2015)
39. Hartley, R., Zisserman, A.: Multiple View in Geometry in Computer Vision, 2nd edn. Cambridge University Press Cambridge (2004)
40. Ma, Y., Soatto, S., Kosecka, J., Sastry, S.S.: An Invitation to 3-D Vision: From Images to Geometric Models, 1st edn. Springer, New York (2003)
41. Pottmann, H., Wallner, J.: Computational Line Geometry, 1st edn. Springer, Berlin (2001)
42. Cox, D.A., Little, J., O'Shea, D.: Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra, 4th edn. Springer International Publishing, Berlin (2015)
43. Stewénius, H.: Gröbner Basis Methods for Minimal Problems in Computer Vision. Ph.D. thesis, Centre for Mathematical Sciences, Lund University, Box 118, SE-221 00 Lund, Sweden (2005)
44. Kukelova, Z., Bujnak, M., Pajdla, T.: Automatic generator of minimal problem solvers. In: European Conference on Computer Vision (ECCV), pp. 302–315 (2008)
45. Kukelova, Z., Bujnak, M., Pajdla, T.: Polynomial eigenvalue solutions to minimal problems in computer vision. IEEE Trans. Pattern Anal. Mach. Intell. (T-PAMI) **7**(34), 1381–1393 (2012)
46. Larsson, V., Astrom, K.: Uncovering symmetries in polynomial systems. In: European Conferene on Computer Vision (ECCV), pp. 252–267 (2016)
47. Larsson, V., Astrom, K., Oskarsson, M.: Polynomial solvers for saturated ideals. In: IEEE International Conference on Computer Vision (ICCV), pp. 2307–2316 (2017)
48. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM **24**(6), 381–395 (1981)
49. Nister, D.: Preemptive ransac for live structure and motion estimation. IEEE Int. Conf. Comput. Vis. (ICCV) **1**, 199–206 (2003)
50. Hofer, M., Maurer, M., Bischof, H.: Efficient 3D scene abstraction using line segments. Comput. Vis. Image Underst. **157**, 167–178 (2017)
51. Schöps, T., et al.: ETH3D Benchmark. https://www.eth3d.net/ (2018). Accessed 24 July 2018

52. Wu, C., Agarwal, S., Curless, B., Seitz, S.M.: Multicore bundle adjustment. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3057–3064 (2011)
53. Wu, C., Agarwal, S., Curless, B., Seitz, S.M.: Towards linear-time incremental structure from motion. In: IEEE International Conference on 3D Vision (3DV), pp. 127–134 (2013)