Camera and inertial sensor fusion for the PnP problem: algorithms and experimental results

Luigi D'Alfonso¹ · Emanuele Garone³ · Pietro Muraca² · Paolo Pugliese²

Received: 14 November 2019 / Revised: 14 January 2021 / Accepted: 24 May 2021
© The Author(s) 2021, corrected publication 2021

Abstract

In this work, we face the problem of estimating the relative position and orientation of a camera and an object, when they are both equipped with inertial measurement units (IMUs), and the object exhibits a set of n landmark points with known coordinates (the so-called *Pose estimation* or PnP Problem). We present two algorithms that, fusing the information provided by the camera and the IMUs, solve the PnP problem with good accuracy. These algorithms only use the measurements given by IMUs' inclinometers, as the magnetometers usually give inaccurate estimates of the Earth magnetic vector. The effectiveness of the proposed methods is assessed by numerical simulations and experimental tests. The results of the tests are compared with the most recent methods proposed in the literature.

Keywords Pose estimation · PnP Problem · Heterogeneous sensors fusion

1 Introduction

The pose estimation problem, or perspective- n -point (PnP) problem, was first introduced in [1] as:

Given a set of 'landmarks' (control points), whose locations are known in some coordinate frame, determine the location (relative to the coordinate frame of the landmarks) of that point in space from which an image of the landmarks was obtained.

In other words, the problem consists in obtaining the relative position and orientation of an object (for example, a drone,

or a robotic hand) with respect to a camera, using an image provided by the camera and the knowledge of a set of n landmarks on the object. Figure 1, where landmarks are displayed in blue, shows a typical physical configuration. This problem is relevant in computer vision [2,3], augmented reality [4], photogrammetry [5], and robotics [6–8].

It is worth noticing that the terms 'landmarks', 'landmark points' and 'control points' are used interchangeably in the literature, and the set of those points is often referred to as a 'feature' on the object. Because of that, the landmarks are often referred to as a 'feature points'.

Although the PnP problem dates back to the '80s, some recent papers witness the perduring interest in this topic and its applications. In [9], a PnP module is used in a vision-based SLAM algorithm to recover the camera pose after a tracking failure, and the $EPnP$ method presented in [10] is used to solve that problem. In [11], the problem of an unmanned aerial vehicle (UAV) landing on a runway is faced, and it is shown that a PnP algorithm using landmark points on the runway outperforms the landing procedure based on GPS only. There, the PnP algorithm presented in [12] is used to compute the solution. In [13], a PnP framework is used to solve the perspective- n -lines (PnL) problem, which is a generalization of the former. The similitude between the two problems, signately with the approach of [10], gives rise to many PnL algorithms "inspired by their PnP counterparts".

✉ Paolo Pugliese
pugliese@unical.it

Luigi D'Alfonso
l.dalfonso@gipstech.com

Emanuele Garone
egarone@ulb.ac.be

Pietro Muraca
muraca@dimes.unical.it

¹ GiPStech, P.za Vermicelli (Arcavacata), 87036 Rende, CS, Italy

² DIMES, Università della Calabria, via P. Bucci, 42/C, 87030 Rende, CS, Italy

³ Université Libre de Bruxelles, avenue F.D. Roosevelt, 50, Brussels, Belgium

Also, the consumer electronic industry nowadays makes wide use of PnP modules. As an example, *Isaac*, NVIDIA's open platform for intelligent robots, has a PnP module for navigation and manipulation [14].

1.1 Standard PnP problem

Owing to its relevance in the applications, the PnP problem has been deeply investigated. It has been shown that the two points' feature (P2P) leads to infinitely many solutions for the position and the orientation and that a three points one (P3P) leads to a finite number of solutions [1]. In [15], it is shown that the P3P has at most four solutions, and conditions for the unicity of the solution are provided. For $n = 4$, a unique solution is obtained if the points are coplanar and at most two of them lie on the same line [16].

Because of the unicity of its solution, the P4P problem has been thoroughly investigated. Paper [1] has introduced the RANSAC algorithm, which faces the P4P by solving the P3P problem for each groups of three points out of the four and then taking the intersection of their solutions.

PnP problems with $n > 4$ have also been considered, to obtain a solution less sensitive to measurement noise. This has been done, e.g., in [10,18], using linear algebra techniques, and in [12] using an iterative optimization method.

1.2 PnP problems and IMUs

Standard PnP methods only use the pictures taken by the camera; as of today, however, many 'intelligent objects' (e.g., mobile robots and drones) and cameras are equipped with inertial measurement units (IMUs), which are able to measure the Earth's gravity and magnetic vectors. This information can be used to improve the accuracy of the relative pose estimation. For instance, in [19,20] an extended Kalman filter is used to fuse the data coming from the camera and the IMUs and to obtain the relative pose, whilst in [21] the orientation provided by the IMUs is used to compute the translation vector between the object and the camera.

In this paper, we show how the measurements taken from the IMUs and the camera can be fused to solve the PnP problem, without using the compass data from the IMUs, which is usually noisy, and due to the high resolution of IMUs' accelerometers, we can obtain an accurate solution.

An earlier work along this approach is [22], but there the authors assume to know the coordinates of two landmarks in the absolute reference frame; then, using the roll and pitch provided by the IMU on the camera, they solve the P2P.

The present paper extends the preliminary results given in [23,24]; as in [22], we assume that both the camera and the object are equipped with IMUs, but no hypothesis on their coordinates is made. First, we face the P2P problem using inclinometers only: we prove that it admits two solutions and

propose a geometrical test to mitigate the ambiguity between them. Also, we show that singular configurations and ambiguities can be avoided using three non-collinear landmarks (P3P). Then, we face the general PnP, formulated as the optimization of a least square index.

The performance of the proposed algorithm will be analyzed by both numerical simulations and experimental tests, and it will be contrasted against the most recent methods from the literature.

1.3 Problem statement

Consider an object in the field of view of a camera and define two reference frames: one for the camera, denoted by O_{xyz} , whose origin is in the camera focus, and one for the object, denoted by O'_{uvw} .

Moreover, let $R = [r_1^T; r_2^T; r_3^T]$ and $t = [t_x, t_y, t_z]^T$ be the rotation matrix and the translation vector from the object reference frame to the camera reference frame, respectively (r_i^T denotes the i -th row of R).

Assume that the object is provided with n landmarks P_i , $i = 1, \dots, n$, known a priori in the object reference frame; the coordinates π_i of the landmark P_i in the camera reference frame O_{xyz} are

$$\pi_i = [\pi_{x,i}, \pi_{y,i}, \pi_{z,i}]^T = R P_i + t. \quad (1)$$

Both the object and the camera are equipped with IMUs, which are able to measure the gravity and the magnetic unit vectors

$$g_o = [g_u, g_v, g_w]^T, \quad m_o = [m_u, m_v, m_w]^T,$$

in the object reference frame O'_{uvw} , and

$$g_c = [g_x, g_y, g_z]^T, \quad m_c = [m_x, m_y, m_z]^T,$$

in the camera reference frame O_{xyz} , respectively; the couples (g_c, m_c) and (g_o, m_o) denote the same vectors in the different coordinate frames, i.e., $g_c = R g_o$ and $m_c = R m_o$.

We model the camera as the pinhole [25], defined by the distance-per-pixel dpx and the focal length f . The bi-dimensional image reference frame is denoted by $O_{x_c y_c}^c$, the image center is $C_I = (x_{C_I}, y_{C_I})$ and the image plane is $z = f$. Accordingly, each 3-D feature point P_i is projected into the 2-D pixel

$$P_i^* = \left[\frac{f}{dpx} \frac{\pi_{x,i}}{\pi_{z,i}} + x_{C_I}, \frac{f}{dpx} \frac{\pi_{y,i}}{\pi_{z,i}} + y_{C_I} \right], \quad (2)$$

which is the intersection between the image plane and the line connecting P_i and the camera focus: see Fig. 1.

Fig. 1 Camera, image and object reference frames

Our goal is to solve the PnP using data g_o and g_c given by the IMUs, the landmarks' coordinates in the object reference frame and their projections on the image plane, thus obtaining the transformation matrix R_t from O'_{uvw} to O_{xyz} :

$$R_t = \begin{bmatrix} R & t \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

2 The method

If the whole information from IMUs was used, the rotation matrix R would be known immediately; however, we wish to use only the information coming from IMUs' accelerometers (g_c and g_o), neglecting m_c and m_o for the noise reason discussed before. To obtain the matrix R_t in this setting, we will find a parametrization of R w.r.t. the neglected data.

Let a reference frame O^h be given. If the rotation matrix R_c^h from this reference frame to the camera's one, and the rotation matrix R_o^h from this frame to the object's one were known, then the overall rotation matrix, independently of O^h , would be:

$$R = R_c^o = R_c^h R_o^h. \quad (3)$$

To obtain the rotation matrix from one frame to another, we must represent the basis of the first frame in the second's one. Thus, let us define an intermediate, *artificial* reference frame O^h , denoted as North-East-Down (NED), where the Down vector is the gravity and the North vector is arbitrary. Using the O^h frame, the rotation matrices R_c^h and R_o^h will be found and R obtained by Eq. (3).

Fig. 2 Unit vectors g_o and $\hat{m}_o$

The first column of R_c^h will be the vector g_c . As for the second column, an artificial North vector $\hat{m}_c$ in the camera reference frame is taken, orthogonal to g_c :

$$\hat{m}_c = [g_z, 0, -g_x]^T / (g_x^2 + g_z^2)^{1/2}. \quad (4)$$

The third column of R_c^h will be $\hat{n}_c = g_c \times \hat{m}_c$, i.e.,

$$\hat{n}_c = [-g_x g_y, g_x^2 + g_z^2, -g_y g_z]^T / (g_x^2 + g_z^2)^{1/2}, \quad (5)$$

and thus the rotation matrix from O^h to O_{xyz} will be

$$R_c^h = [g_c \quad \hat{m}_c \quad \hat{n}_c]. \quad (6)$$

Now we build the rotation matrix $R_o^h = [R_o^h]^T$ from O^h to the object's reference frame. The first column of R_o^h will be g_o ; the second column will contain $\hat{m}_o$, the artificial North in the object's reference frame.

The coordinates of $\hat{m}_o$ in O'_{uvw} are unknown, but it has to lie on the intersection between the plane orthogonal to g_o and a unit sphere: see Fig. 2. This gives a parametrization:

$$\hat{m}_o = \hat{m}_o(\alpha) = \hat{m}_1 \sin \alpha + \hat{m}_2 \cos \alpha, \quad (7)$$

where $\{\hat{m}_1, \hat{m}_2\}$ is an orthonormal basis of the plane orthogonal to g_o and α is the unknown angle characterizing the artificial North vector.

For the sake of simplicity, $\hat{m}_1$ is chosen as

$$\hat{m}_1 = [g_w, 0, -g_u]^T / (g_u^2 + g_w^2)^{1/2},$$

and thus $\hat{m}_2$ is given by $\hat{m}_2 = g_o \times \hat{m}_1$, which implies

$$\hat{m}_2 = [-g_u g_v, g_u^2 + g_w^2, -g_v g_w]^T / (g_u^2 + g_w^2)^{1/2}.$$

The third vector is $\hat{n}_o(\alpha) = g_o \times \hat{m}_o(\alpha)$:

$$\hat{n}_o(\alpha) = \begin{bmatrix} -(g_w \cos \alpha + g_u g_v \sin \alpha) \\ (g_u^2 + g_w^2) \sin \alpha \\ g_u \cos \alpha - g_v g_w \sin \alpha \end{bmatrix} / (g_u^2 + g_w^2)^{1/2}, \quad (8)$$

and the rotation matrix from the O^h reference frame to O'_{uvw} is

$$R_o^h(\alpha) = [g_o \hat{m}_o(\alpha) \hat{n}_o(\alpha)]. \quad (9)$$

Finally, given $R_o^h(\alpha)$ and R_c^h , we have

$$R = R(\alpha) = R_c^h [R_o^h(\alpha)]^T, \quad (10)$$

and the whole transformation matrix will be:

$$R_t(\alpha) = \begin{bmatrix} R(\alpha) & t \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (11)$$

2.1 P2P and P3P using accelerometers only

As for the P2P, let $P_i = (P_{u,i}, P_{v,i}, P_{w,i})$, $i = 1, 2$, the (known) coordinates of the two landmarks in O'_{uvw} . From Eq. (2), each point is projected on the image plane in the pixel P_i^* . Defining $\tilde{P}_i = (P_i^* - C_t) dp_x / f$, by Eqs. (1), (2), the two pixels are

$$\tilde{P}_i = (\tilde{P}_{x,i}, \tilde{P}_{y,i}) = \left(\frac{r_1^T P_i + t_x}{r_3^T P_i + t_z}, \frac{r_2^T P_i + t_y}{r_3^T P_i + t_z} \right). \quad (12)$$

Using $R(\alpha)$ in Eq. (12), a system in the four unknowns (t_x, t_y, t_z, α) is obtained, which is solved by first computing α and then getting $R(\alpha)$ by Eq. (10). Then, a unique translation t is obtained: see Sect. 2.3 in the sequel or Ref. [21].

To compute α , under the assumption that either $\Delta_x = \tilde{P}_{x,1} - \tilde{P}_{x,2} \neq 0$ or $\Delta_y = \tilde{P}_{y,1} - \tilde{P}_{y,2} \neq 0$, the following equation can be obtained:

$$a \sin \alpha + b \cos \alpha + c = 0 \quad (13)$$

where

$$\begin{aligned} a &= a(P_1, P_2, \tilde{P}_{x,1}, \tilde{P}_{y,1}, \tilde{P}_{x,2}, \tilde{P}_{y,2}, g_c, g_o), \\ b &= b(P_1, P_2, \tilde{P}_{x,1}, \tilde{P}_{y,1}, \tilde{P}_{x,2}, \tilde{P}_{y,2}, g_c, g_o), \\ c &= c(P_1, P_2, \tilde{P}_{x,1}, \tilde{P}_{y,1}, \tilde{P}_{x,2}, \tilde{P}_{y,2}, g_c, g_o), \end{aligned}$$

are quantities that can be computed from the measured data (see [23] for the complete expression of a , b , and c). At this point, using

$$a \sin \alpha + b \cos \alpha = M \cos(\alpha - \beta),$$

Fig. 3 P2P with $\Delta_x = 0$, $\Delta_y = 0$: line vector not aligned with gravity

where

$$M = \sqrt{a^2 + b^2}, \quad \beta = \arctan(a/b),$$

we obtain:

$$\alpha = \arccos(-c/M) + \beta. \quad (14)$$

Equation (14) has two solutions, denoted by α_1 and α_2 , from which we can compute two couples: (R_1, t_1) and (R_2, t_2) .

The case $\Delta_x = 0$ and $\Delta_y = 0$ happens when the two landmarks and O_{xyz} lie on the same line. The vector defining that line can be found if it is not aligned with the gravity. Depending on its verse, there are two possible values for α , but, if we can distinguish pixels $\tilde{P}_1$ and $\tilde{P}_2$ (e.g., by colored landmarks), we get the correct one. If the line and gravity vectors are aligned, then the information on the line vector is the same provided by the IMUs, and we have infinitely many solutions for the rotation matrix.

The case $\Delta_x = 0$ and $\Delta_y = 0$ means that the two pixels $\tilde{P}_1$ and $\tilde{P}_2$ coincide: as the camera sees a single point, the distance between the reference frames cannot be obtained. We thus have the quite intuitive result:

Theorem 1 Using the image provided by the camera and the gravity vectors given by the IMUs, the P2P returns:

- two solutions for R and t if $\Delta_x \neq 0$ or $\Delta_y \neq 0$;
- a unique solution for R and infinitely many solutions for t when $\tilde{P}_1 = \tilde{P}_2$, the line between the two landmarks and O_{xyz} is not aligned with g , and the closest point is recognized (see Fig. 3);
- infinitely many solutions for R and t when $\tilde{P}_1 = \tilde{P}_2$ and the line between the two landmarks and O_{xyz} is aligned with g .

Algorithm 1: P2P Solution Feasibility Test.

```

for  $i = 1$  to 2 do
  compute  $\pi_1 = R_i P_1 + t_i$  and  $\pi_2 = R_i P_2 + t_i$ 
  check if Eq. (15) or Eq. (16) is satisfied to ensure that both
     $P_1$  and  $P_2$  are in the camera view
  if one of the points is not in the view, then discard  $(R_i, t_i)$ 
end

```

From this analysis, it results that the P2P with accelerometers has two solutions whenever the camera sees two distinct points. However, each time one of the transformation matrices maps a feature point out of the camera view, one of them can be discarded.

The camera view is the conic combination of vectors $\{v_1, v_2, v_3, v_4\}$ going from the camera focus to the image corners. A point P will be in the camera view if and only if it can be expressed as a conic combination of either $\{v_1, v_2, v_3\}$ or $\{v_1, v_3, v_4\}$, or that either

$$\exists [k_1, k_2, k_3] \geq 0 : [v_1 \ v_2 \ v_3] [k_1, k_2, k_3]^T = P$$

or

$$\exists [k_1, k_3, k_4] \geq 0 : [v_1 \ v_3 \ v_4] [k_1, k_3, k_4]^T = P.$$

Being $\{v_1, v_2, v_3\}$ and $\{v_1, v_3, v_4\}$ two sets of three independent vectors of $\mathbb{R}^3$, such conditions are equivalent to

$$[v_1 \ v_2 \ v_3]^{-1} P \geq 0 \quad (15)$$

or

$$[v_1 \ v_3 \ v_4]^{-1} P \geq 0. \quad (16)$$

This fact has been used to define Algorithm 1, to try to discard one of the couples (R, t) . To evaluate how effective is this way to discard impossible configuration, we have performed ten thousand numerical simulations using that algorithm, with random configurations of the object and the camera. The results showed that Algorithm 1 is able to discard the wrong solution in 47% of the tests.

2.2 A heuristic for the P3P

Another way to mitigate the ambiguities in the solutions of the P2P problem is to use a third feature point P_3 , not collinear with P_1 and P_2 . This also ensures that it is not possible to see a unique pixel in the camera image. Assume, without loss of generality, that points P_1 and P_2 are not aligned with the camera focus. The proposed method is reported in Algorithm 2.

Although this is only a heuristic approach, in *all* the ten thousand simulations used to test Algorithm 1 it always

identified the correct solution. We remark that, to increase robustness w.r.t. pixel noise, it is advisable to perform the P2P on the couple of points with the largest distance in the image plane.

2.3 Computing the translation vector for the P2P

Once matrix R has been estimated, the translation vector t must be computed; to this sake, the method in reference [21] can be used. This method solves Eq. (12) to obtain t_x, t_y, t_z :

$$\begin{aligned} t_x &= A_z \tilde{P}_{x,1} - r_1^T P_1 \\ t_y &= A_z \tilde{P}_{y,1} - r_2^T P_1, \\ t_z &= A_z - r_3^T P_1 \end{aligned} \quad (17)$$

where A_z is computed as follows:

$$\begin{aligned} \text{if } \tilde{P}_{x,2} \neq \tilde{P}_{x,1} \text{ then } A_z &= \frac{(r_1^T - \tilde{P}_{x,2} r_3^T)(P_1 - P_2)}{\tilde{P}_{x,1} - \tilde{P}_{x,2}}, \\ \text{otherwise } A_z &= \frac{(r_2^T - \tilde{P}_{y,2} r_3^T)(P_1 - P_2)}{\tilde{P}_{y,1} - \tilde{P}_{y,2}}. \end{aligned}$$

This solution, however, can be sensitive to pixel noise, as we will show in the sequel. An alternative way to compute t is the minimization of the *reprojection error*

$$\mathbb{E}(t) = \|\tilde{P}_1(t) - \tilde{P}_1\|^2 + \|\tilde{P}_2(t) - \tilde{P}_2\|^2, \quad (18)$$

where $\tilde{P}_1(t)$ and $\tilde{P}_2(t)$ are the pixels where points P_1 and P_2 are projected using the rotation matrix previously computed, the unknown translation vector t , and Eq. (2).

The reprojection error is widely used in the literature to test the performance of PnP solutions (see [10,21,22]), but the resulting optimization problem must be solved numerically. To overcome this difficulty, we propose a different least square index. To this end, we write Eqs. (12) in the explicit form $Q t = s$, where

$$Q = \begin{bmatrix} -1 & 0 & \tilde{P}_{x,1} \\ 0 & -1 & \tilde{P}_{y,1} \\ -1 & 0 & \tilde{P}_{x,2} \\ 0 & -1 & \tilde{P}_{y,2} \end{bmatrix}; \quad s = \begin{bmatrix} r_1^T P_1 - r_3^T P_1 \tilde{P}_{x,1} \\ r_2^T P_1 - r_3^T P_1 \tilde{P}_{y,1} \\ r_1^T P_2 - r_3^T P_2 \tilde{P}_{x,2} \\ r_2^T P_2 - r_3^T P_2 \tilde{P}_{y,2} \end{bmatrix} s,$$

Algorithm 2: P3P using inclinometers.

- 1 **solve** the P2P problem with points P_1 and P_2 using Eq. (13);
 - 2 **compute** couples (R_1, t_1) and (R_2, t_2) , and pixels $\tilde{P}_3^1$ and $\tilde{P}_3^2$, using Eq. (2);
 - 3 **if** $\|\tilde{P}_3^1 - \tilde{P}_3^2\| < \|\tilde{P}_3^2 - \tilde{P}_3\|$, then (R_1, t_1) is taken as the solution, otherwise it is (R_2, t_2) .
-

Fig. 4 Values of index ε_t in some of the performed tests

and we solve for t minimizing an index alternative to (18):

$$\mathbb{E}_2(t) = \|Qt - s\|^2, \quad (19)$$

which defines the least mean square (LMS) solution of $Qt = s$; this is given by the pseudo-inverse [26]:

$$t^* = Q^\dagger s. \quad (20)$$

Owing to the nature of the LMS method, the solution t^* is expected to be less sensitive to pixel noise than the one obtained using Eq. (17). This fact has been verified by ten thousand numerical tests, simulating a feature with two points in $P_1 = [0, 0, 0]$ m and $P_2 = [0.1, 0.1, 0]$ m, and a zero mean Gaussian noise with standard deviation of five pixels on the components of $\tilde{P}_1$ and $\tilde{P}_2$. In each test, the matrix R_t has been taken randomly. Comparison has been made through the absolute error

$$\varepsilon_t = \|t - t_s\|, \quad (21)$$

where t_s is the solution from either Eq. (17) or Eq. (20).

As expected (see Fig. 4), the errors obtained using the pseudo-inverse are much smaller than the ones obtained by Eq. (17): an average error of 0.029 m results from Eq. (20), versus an average error of 0.24 m from Eq. (17).

Remark Indexes $\mathbb{E}$ and $\mathbb{E}_2$ satisfy $\mathbb{E} = \|M\|^2 \mathbb{E}_2$, where

$$M = \text{diag} \left\{ \frac{1}{r_3^T P_1 + t_z}, \frac{1}{r_3^T P_1 + t_z}, \frac{1}{r_3^T P_2 + t_z}, \frac{1}{r_3^T P_2 + t_z} \right\}.$$

For this reason, under the assumption $t_z \gg r_3^T (P_1 - P_2)$, the optimal $\mathbb{E}_2$ can be taken as an estimate of the optimal $\mathbb{E}$.

Figure 5 shows, for some of the ten thousand tests, the values of ε_t computed using both vector t^* by Eq. (20) and vector t by the (computationally expensive) numerical optimization of $\mathbb{E}$. No estimation advantage is evident in using

Fig. 5 Values of index ε_t when minimizing $\mathbb{E}$ and $\mathbb{E}_2$

$\mathbb{E}$; conversely, $\mathbb{E}_2$ outperforms $\mathbb{E}$ in terms of computational time, with an average CPU time of 0.002 s against 0.556 s.

2.4 Computing the translation vector for the PnP

In this section, we consider again the case where the rotation matrix is assumed known, and we generalize the results presented in the previous subsection to the case $n \geq 3$, assuming that at least three feature points are not collinear. As before, the problem of estimating the vector t will be formulated as a set of linear equations. Given the n correspondences between landmarks and pixels, the matrix Q and the vector s defined before now contain $2n$ rows (two for each correspondence):

$$\begin{aligned} Q_{i,1} &= [-1, 0, \tilde{P}_{x,i}], & s_{i,1} &= r_1^T P_i - r_3^T P_i \tilde{P}_{x,i} \\ Q_{i,2} &= [0, -1, \tilde{P}_{y,i}], & s_{i,2} &= r_2^T P_i - r_3^T P_i \tilde{P}_{y,i}, \end{aligned}$$

the matrix Q and the vector s now will have the form:

$$Q = \begin{bmatrix} Q_{1,1} \\ Q_{1,2} \\ \vdots \\ Q_{n,1} \\ Q_{n,2} \end{bmatrix}, \quad s = \begin{bmatrix} s_{1,1} \\ s_{1,2} \\ \vdots \\ s_{n,1} \\ s_{n,2} \end{bmatrix}, \quad (22)$$

and again the LMS solution will be given by $t^* = Q^\dagger s$.

If matrix R has not yet been estimated, both Q and s depend on the unknown parameter α , i.e., $Q = Q(\alpha)$ and $s = s(\alpha)$; as a consequence, also $t = t(\alpha)$. This will be exploited in the next section to solve the general PnP Problem.

2.5 The general PnP using accelerometers only

To solve the PnP problem in our setting, we first have to find the angle α and then to substitute it in the rotation matrix $R(\alpha)$ and in the translation vector $t = t(\alpha)$, given the knowledge

of the n feature points P_i in the object reference frame and the related pixels $\tilde{P}_i$.

Also, in this case a possible solution is to define the reprojection error

$$J(\alpha, t) = \sum_{i=1}^n \|\tilde{P}_i(\alpha, t) - \tilde{P}_i\|^2, \quad (23)$$

$\tilde{P}_i(\alpha, t)$ being the pixel where feature point P_i is mapped by the transformation matrix $R_t(\alpha)$, and to solve the problem

$$\{\alpha^*, t^*\} = \arg \min_{\alpha, t} J(\alpha, t). \quad (24)$$

Once the optimal pair (α^*, t^*) has been estimated, the related transformation matrix $R_t(\alpha^*)$ will be the solution of the PnP problem.

Again, the alternative LMS cost function is proposed:

$$J_2(\alpha, t) = \|Q(\alpha)t - s(\alpha)\|^2, \quad (25)$$

where $Q(\alpha)$ and $s(\alpha)$ are given by Eq. (22) with $R = R(\alpha)$. The same considerations about the robustness of the LMS solution w.r.t. pixel noise made for the P2P case apply.

The LMS problem associated with the PnP becomes

$$\{\alpha^*, t^*\} = \arg \min_{\alpha, t} J_2(\alpha, t), \quad (26)$$

and this four-variable problem is reformulated as a single variable one: by Eq. (20), translation vector t depends on α only. Thus, $t^* = t(\alpha^*)$ and Problem (26) translates to:

$$\alpha^* = \arg \min_{\alpha} J_2(\alpha, t(\alpha)). \quad (27)$$

The advantage of this formulation is that Problem (27) is a single variable, unconstrained optimization problem, which can be solved very efficiently, e.g., using Nelder–Mead's method [27]. A good initial guess for the solution can be obtained by solving a P3P problem on three arbitrary feature points.

In Fig. 6, the shape of function $J_2(\alpha, t(\alpha))$ is shown, with $\alpha \in [-\pi, \pi]$, for some of the experimental tests. In all cases, the function J_2 looks very much like a sinusoid, and the two slightly different minima always lie one in $[-\pi, 0)$ and the other one in $[0, \pi)$.

3 Experimental results

To assess the performance of the method, we have performed numerical simulations and experiments on the real ground. In particular, the eight algorithms reported in [28]: *DLS* [29], *ASPnP* [30], *OPnP* [31], *RPnP* [32], *PPnP* [33],

Fig. 6 Shape of function $J_2(\alpha, t(\alpha))$ from some real data

EPnP [10], *LHM* [12] and finally *EPPnP* [28], all implemented in the *REPPnP* Toolbox [35], have been compared to the solution obtained solving Problem (27), denoted by α -*PnP*, solved using MATLAB function `fminsearch`, which uses Nelder–Mead's method. In the experiments using real data, the recent method from [34], denoted by *SRPnP*, has been added to the previous eight.

The results have been contrasted using the two indexes suggested in [28,31]:

– *rotation error*:

$$\epsilon_{rot} = 180 \max_{k=1,2,3} \arccos(c_k^T c_k^*) / \pi,$$

where c_k and c_k^* are the k -th columns of R and of $R(\alpha^*)$;

– *relative translation error*:

$$\epsilon_t(\%) = 100 \|t - t^*\| / \|t\|,$$

where t and t^* are the real t and $t(\alpha^*)$, respectively.

3.1 Numerical simulations

In the numerical simulations, feature points have been taken as $P_1 = (0, 0, 0)$, $P_2 = (0.1, 0.1, 0)$, $P_3 = (0.1, 0, 0)$, and the remaining ones taken randomly in the box $B_f = [-0.2, 0.2]^3$.

For each test, random relative rotation and translation between the camera and the object have been generated, with roll, pitch and yaw angles in the range $(-\pi, \pi]$, and translations in $B_t = [-0.5, 0.5] \times [-0.5, 0.5] \times [0.5, 2.5]$.

A camera with ratio $f/dpx = 800$ pixels, a resolution of $[640 \times 480]$ pixels and center $C_I = (320, 240)$ has been simulated, and the data have been corrupted by a zero-mean Gaussian noise with variances

Fig. 7 Mean rotation error (degrees) versus number of feature points: simulative results

Fig. 8 Mean translation error (percent) versus number of feature points: simulative results

- $\Sigma_g = \sigma_g I_3$, $\sigma_g = 0.001$ m, for $\hat{g}_c$ and $\hat{g}_o$;
- $\sigma_p = 4$ pixels in the image.

The performance of the various algorithms has been assessed using n -point features, with $n \in \{10, 30, 50, 70, 90\}$, in 200 random configurations for each value of n . Figures 7 and 8 show the average values of ϵ_{rot} and ϵ_t (%) versus n , and Fig. 9 shows the mean CPU time. The time performance has been estimated using MATLAB `tic` and `toc` functions.

It is evident that the α -PnP algorithm has the best result in terms of accuracy, although some others are close, in particular for a small number of feature points. Even more evident is the α -PnP CPU time performance: it is up to two orders of magnitude faster than all the others. An interesting fact to observe is also that, after about 30 feature points, adding more points to the feature does not improve noticeably the accuracy of the solution.

3.2 Real experiments—static

In the experimental setting, we have used the following hardware:

Fig. 9 Mean CPU time (milliseconds) versus number of feature points: simulative results

Fig. 10 The four-square feature used in the real experiments

- a Logitech C310 HD camera; the parameters of the camera have been estimated using the CC Toolbox [37];
- two ArduIMU v3 IMUs [38];
- the four-square feature shown in Figure 10. Each corner of the squares can be used as a feature point.

To mitigate the systematic errors due to the experimental setup, all the tests have been performed according to the following procedure:

1. the camera and the object have been set in an unknown configuration, and an estimate $\hat{R}'_t$ of the transformation matrix R'_t has been obtained by the various algorithms;
2. the object has been rotated and translated with a given matrix R''_t , and an estimate $\hat{R}''_t$ of such matrix has been

Fig. 11 Mean rotation error (degrees) versus number of feature points: real experiment

obtained. To enforce accurate placement, the rotation has

been taken by repeated bisections of an initial $\pi/2$ angle;

3. an estimate $\hat{R}_t$ of the displacement matrix between the two configurations has been computed as $\hat{R}_t = [\hat{R}_t']^{-1} \hat{R}_t''$.

The performance of the eight algorithms reported in [28] and the α -PnP has been compared in a set of fifty experiments, according to the above procedure. Each picture has been used to run the various algorithms using $n = \{4, 8, 12, 16\}$.

The average orientation and translation errors computed over R_t and $\hat{R}_t$ are shown in Figs. 11 and 12.¹ We see that even in the tests using real data the α -PnP algorithm exhibits a sound performance (the best one in estimating the translation and the third best in estimating the rotation), with a computational cost, see Fig. 13, at least twenty times smaller than the two best ones in estimating the rotation.

3.3 Real experiments—dynamic

To further investigate the performance of the proposed technique, a second experiment with real data has been carried out. The four-square feature shown in Fig. 10 and one IMU have been placed on the mobile robot Khepera III [39], and the robot has been programmed to perform, in 40 sec, a rectangular trajectory of 1.5×1 m with trapezoidal speed.

The camera, equipped with the second IMU, has been placed on the top of the motion plane at a distance of 3 m; sixteen equally time-spaced pictures have been taken during the journey; and the rotation and translation errors have been computed and averaged on those pictures.

The overall results with $n = \{4, 8, 12, 16\}$ feature points are reported in Figs. 14 to 16. Also in this case there is not a clear winner between the various algorithms in terms

Fig. 12 Mean translation error (percent) versus number of feature points: real experiment

Fig. 13 Mean CPU time (milliseconds) versus number of feature points: real experiment

Fig. 14 Mean rotation error (degrees) versus number of feature points: dynamic experiment

of estimation performance. (The *LHM* wins on the rotation estimation and the *DLS* wins on the translation estimation.) However, the proposed method performs computationally much better than the competitors also in this experiment, as it is 100 times faster of the *DLS* and 20 times faster of the *LHM*. In some sense, it guarantees a good ‘balanced performance’ between translation and rotation estimation, and it obtains the estimates at a very high speed.

¹ There, the results from the *PPnP* method have not been reported to increase readability, as they were much worse than all the others.

Fig. 15 Mean translation error (percent) versus number of feature points: dynamic experiment

Fig. 16 Mean CPU time (milliseconds) versus number of feature points: dynamic experiment

4 Discussion

The simulative and experimental results have shown that fusing the measurements given by a camera and two IMUs allows to obtain a fast and accurate estimate of the relative position and orientation between two objects.

In both the simulations and the experimental tests, the proposed α -PnP algorithm outperforms most of the methods listed in [28] in terms of estimation error. Only the LHM, and occasionally the recent SRPnP, have given a better estimation of the rotation, at the price of a much higher computational effort (about 30 times or more).

The CPU time required by the α -PnP is from 20 to 100 times less than that of all the others methods, making that method suitable for real-time applications. Also, the accuracy of the estimates it gives is practically independent of the number of feature points used.

We also emphasize that, to the authors' knowledge, the results presented here are the first ones where the experimental results are evaluated in terms of rotation and translation errors. Usually, when dealing with experimental data, only the reprojection index is used to test the performance. How-

ever, such index is not a robust evaluation method, owing to the strong nonlinearities and ambiguities embedded in the pinhole reprojection model.

On considering the low cost of IMUs and the standard linear algebra techniques used, we believe that the proposed method is a useful tool to face the pose estimation problem.

Acknowledgements Authors thank Dr. Luis Ferraz of Universitat Pompeu Fabra, Barcelona, for the kind help in using his REPPnP Toolbox. They also express their gratitude to the anonymous referees for their welcome suggestions.

Funding Open access funding provided by Università della Calabria within the CRUI-CARE Agreement.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. 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**, 381–395 (1981)
2. Hartley, R., Zisserman, A.: Multiple view geometry in computer vision, 2nd edn. Cambridge University Press, Cambridge (2003)
3. Forsyth, D., Ponce, J.: Computer vision: a modern approach, 2nd edn. Prentice Hall, London (2012)
4. Marchand, E., Uchiyama, H., Spindler, F.: Pose estimation for augmented reality: a hands-on survey. *IEEE Trans. Vis. Comput. Graph* **22**(12), 2633–2651 (2016)
5. McGlone, J., Mikhail, E., Bethel, J., Manual of photogrammetry, 5th Ed., American Society for Photogrammetry and Remote Sensing (2004)
6. Chen, G., Xu, D., Yang, P.: High precision pose measurement for humanoid robot based on PnP and OI algorithms. In: *Proc. IEEE ROBIO*, pp. 620–624 (2010)
7. Wang, Q., Zhang, X., Xu, D.: Human behavior imitation for a robot to play table tennis. In: *Proc. CCDC*, pp. 1482–1487 (2012)
8. Wang, Q., Zhang, X., Xu, D.: On pose recovery for generalized visual sensors. *IEEE Trans. Pattern Anal. Mach. Intell.* **26**, 848–861 (2004)
9. Feng, Y., Wu, Y., Fan, L.: Real-time SLAM relocalization with online learning of binary feature indexing. *Mach. Vis. Appl.* **28**, 953–963 (2017)
10. Lepetit, V., Moreno-Noguer, F., Fua, P.: EPnP: an accurate $O(n)$ solution to the PnP problem. *Int. J. Comput. Vis.* **81**, 155–166 (2009)
11. Ruchanurucks, M., Rakprayoon, P., Kongkaew, S.: Automatic landing assist system using IMU+PnP for robust positioning of fixed-wing UAVs. *J. Intell. Robot. Syst.* **90**, 189–199 (2018)

12. Lu, C.P., Hager, G.D., Mjolsness, E.: Fast and globally convergent pose estimation from video images. *IEEE Trans. Pattern Anal. Mach. Intell.* **22**, 610–622 (2000)
13. Xu, C., Zhang, L., Cheng, L., Koch, R.: Pose estimation from line correspondences: a complete analysis and a series of solutions. *IEEE Trans. Pattern Anal. Mach. Intell.* **39**(6), 1209–1222 (2017)
14. <https://docs.nvidia.com/isaac/isaac/packages/...pnp/doc/pnp.html>
15. Gao, X., Hou, X., Tang, J., Cheng, H.: Complete solution classification for the perspective-three-point problem. *IEEE Trans. Pattern Anal. Mach. Intell.* **25**, 930–943 (2003)
16. Abidi, M.A., Chandra, T.: A new efficient and direct solution for pose estimation using quadrangular targets; algorithm and evaluation. *IEEE Trans. Pattern Anal. Mach. Intell.* **17**, 534–538 (1995)
17. Haralick, R.M., Lee, D., Ottenburg, K., Nölle, M.: Review and analysis of solutions of the three point perspective pose estimation problem. *Int. J. Comput. Vision* **13**, 331–356 (1994)
18. Quan, L., Lan, Z.: Linear N -point camera pose determination. *IEEE Trans. Pattern Anal. Mach. Intell.* **21**, 774–780 (1999)
19. Mirzaei, F.M., Roumeliotis, S.I.: A Kalman filter-based algorithm for IMU-camera calibration: observability analysis and performance evaluation. *IEEE Trans. Robot.* **24**, 1143–1156 (2008)
20. Oskiper, T., Samarasekera, S., Kumar, R.: Tightly-coupled robust vision aided inertial navigation algorithm for augmented reality using monocular camera and IMU. In: *Proc. IEEE ISMAR*, pp. 255–256 (2011)
21. Merckel, L., Nishida, T.: Evaluation of a method to solve the perspective-two-point problem using a three-axis orientation sensor. In: *Proc. IEEE CIT*, pp. 862–867 (2008)
22. Kukulova, Z., Bujnak, M., Pajdla, T.: Closed-form solutions to the minimal absolute pose problems with known vertical direction. *Proc. ACCV*, pp. 216–229 (2010)
23. D'Alfonso, L., Garone, E., Muraca, P., Pugliese, P.: P3P and P2P Problems with known camera and object vertical directions. In: *Proc. 21st IEEE MED*, pp. 444–451 (2013)
24. D'Alfonso, L., Garone, E., Muraca, P., Pugliese, P.: On the use of IMUs in the PnP problem. In: *Proc. IEEE ICRA*, pp. 914–919 (2014)
25. Corke, P.: *Robotics, Vision and Control. Fundamental Algorithms in MATLAB*. Springer (2011)
26. Golub, G.H., Van Loan, C.F.: *Matrix Computations*, 3rd edn. Johns Hopkins, Baltimore (1996)
27. Lagarias, J.C., Reeds, J.A., Wright, M.H., Wright, P.E.: Convergence properties of the Nelder-Mead simplex method in low dimensions. *SIAM J. Optim.* **9**, 112–147 (1998)
28. Ferraz, L., Binefa, X., Moreno-Noguer, F.: Very fast solution to the PnP problem with algebraic outlier rejection. In: *Proc. IEEE CVPR*, pp. 501–508 (2014)
29. Hesck, J.A., Roumeliotis, S.I.: A direct least-squares (DLS) method for PnP. In: *Proc. IEEE ICCV*, pp. 383–390 (2011)
30. Zheng, Y., Sugimoto, S., Okutomi, M.: ASPnP: An accurate and scalable solution to the perspective- n -point problem. *IEICE Trans.* **96**(D(7)), 1525–1535 (2013)
31. Zheng, Y., Kuang, Y., Sugimoto, S., Aström, K., Okutomi, M.: Revisiting the PnP problem: a fast, general and optimal solution. In: *Proc. IEEE ICCV*, pp. 2344–2351 (2013)
32. Li, S., Xu, C., Xie, M.: A robust $O(n)$ solution to the perspective- n -point problem. *IEEE Trans. Pattern Anal. Mach. Intell.* **34**, 1444–1450 (2012)
33. Garro, V., Crosilla, F., Fusiello, A.: Solving the PnP problem with anisotropic orthogonal procrustes analysis. In: *Proc. 3DIMPVT*, pp. 262–269 (2012)
34. Wang, P., Xu, G., Cheng, Y., Yu, Q.: A simple, robust and fast method for the perspective- n -point problem. *Pattern Recogn. Lett.* **108**, 31–37 (2018)
35. <http://www.cmtech.upf.edu/user/62>
36. <http://www.mathworks.com>
37. http://www.vision.caltech.edu/bouguetj/calib_doc
38. <http://code.google.com/p/ardu-imu/wiki/HomePage>
39. <http://www.k-team.com/>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Luigi D'Alfonso was born in Italy in 1985. He received the Ph.D. degree in Systems and Computer Engineering from the University of Calabria, Italy, in 2014. From 2012, he was with the Control Design and System Analysis Department at the Université Libre de Bruxelles, Belgium, as a Visiting Scholar. Since 2015, he is with GiPStech s.r.l. as a researcher. His current research interests include mobile robot control, localization/mapping/SLAM for single-agent and multi-agent systems, perspective- n -point problem using cameras and inertial measurements units.

Emanuele Garone received the masters degree and the Ph.D. degree in systems engineering from the University of Calabria, Italy, in 2005 and 2009, respectively. Since 2010, he has been an Associate Professor with the Control Design and System Analysis Department, Université Libre de Bruxelles, Brussels, Belgium. His interests include constrained control, reference governors, networked control systems, and aerial robotics.

Pietro Muraca is an Associate Professor of Automatic Control at DIMES, University of Calabria, since 2001. Currently, his main research field is on mobile robots. He is the author or co-author of over 70 articles.

Paolo Pugliese was born in Cosenza, Italy, in 1964. He received the Laurea Degree in Electrical Engineering from University of Calabria, Italy, where he is currently an Associate Professor of Automatic Control. His research interests include filtering, optimization for control and polynomial approximation.