Journal of Control, Automation and Electrical Systems (2023) 34:720–730
https://doi.org/10.1007/s40313-023-01006-4

Data-driven Covariance Tuning of the Extended Kalman Filter for Visual-based Pose Estimation of the Stewart Platform

Aurélio T. Salton[1] · Guilherme A. Pimentel[2] · José V. Melo[3] · Rafael S. Castro[3] · Juliano Benfica[3]

Received: 5 April 2021 / Revised: 26 December 2022 / Accepted: 11 April 2023 / Published online: 10 May 2023

Abstract
This paper explores the quaternion representation in order to devise an extended Kalman filter approach for pose estimation: inertial measurements are fused with visual data so as to estimate the position and orientation of a six degrees-of-freedom rigid body. The filter equations are described along with a data-driven tuning method that selects the model covariance matrix based on experimental results. Finally, the proposed algorithm is applied to a six degrees-of-freedom Stewart platform, a representative system of a large class of industrial manipulators that could benefit from the proposed solution.

Keywords Extended Kalman filter (EKF) · Quartenion · Sensor fusion · State estimation · Inertial measurement units (IMU) · Computer vision

1 Introduction

The six degrees-of-freedom (6-DOF) rigid body model describes a large class of systems in different research domains. Examples go from a wide range of robotic manipulators (Sciavicco & Siciliano, 2012) to various autonomous vehicles (Tayebi & McGilvray, 2006). Most applications that use these systems require some form of estimation of the system states, leading to an intrinsic difficulty since no single sensor is able to measure the pose of a 6-DOF rigid body. While inertial measurement units (IMUs) have high bandwidth, they only provide the orientation and position of the rigid body via integration of the gyroscope and accelerometer, which leads to problems related to drift and error growth. The typical approach to work around this problem is to couple these sensors with others that can compensate the drifting effect but provide a low bandwidth response on their own. One such example is the sensor fusion of a camera and an IMU via filtering methods, of which the Kalman Filter is the most popular and widely applied (Kalman, 1960).

Several works have explored the sole use of computer vision in order to achieve state estimation (Li et al., 2020; Yang et al., 2020; Colonnier et al., 2021). Despite promising results, the sole reliance on vision sensing has intrinsic drawbacks related to the already mentioned low bandwidth, high computational cost, and, more fundamentally, to problems related to occlusion. Consequently, investigations of a camera and IMU fusion for pose estimation are a recurring topic of research where different nonlinear filtering techniques have been explored. In particular, particle filters are used by Zhang and Ye (2020) and—in association with dual quaternions—by Sveier and Egeland (2021). They provide good estimations, particularly when subject to non-Gaussian noise, but are computationally expensive and tedious to tune. Another EKF-based solution is presented in Tong et al. (2018) but solely concerned with the orientation of the

A. T. Salton acknowledges the support from CNPq Brazil under Grant 306214/2018-0.

✉ Aurélio T. Salton
aurelio.salton@ufrgs.br

Guilherme A. Pimentel
guilherme.araujopimentel@umons.ac.be

José V. Melo
jose.melo@acad.pucrs.br

Rafael S. Castro
rafael.castro@pucrs.br

Juliano Benfica
juliano.benfica@pucrs.br

[1] School of Engineering, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil

[2] Systems, Estimation, Control and Optimization (SECO) Group, University of Mons, Mons, Belgium

[3] Group of Automation and Control of Systems (GACS), School of Technology, Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, Brazil

body. Zhang et al. (2020) proposes an unscented Kalman filter approach but does not rely on quaternions, making use of trigonometric functions instead. Other works present less flexibility because they are based on the use of maps (Ratz et al., 2020), do not mention how to tune the covariance matrices of the EKF (Araguás et al., 2015), or tend to specialize in the estimation of some variables instead of offering a complete pose estimation (Nützi et al., 2011). Approaches solely based on image information are commonly referred to as the perspective-n-point (PnP) problem by the Computer Vision Community, resulting in a vast body of literature such as He et al. (2020) and Schönberger et al. (2018). While these methods do not include the added robustness of IMU integration, they may be incorporated in the current approach by replacing the image-based computation of the pose estimate.

This paper focuses on the derivation of the EKF equations for the quaternion-based 6-DOF rigid body model to estimate its full pose, i.e., orientation and position. Since the main difficulty in implementing the EKF is choosing the covariance matrix, a novel data-driven tuning method is proposed. Given an experiment where the "ground-truth" is available, this method iterates the best covariance matrix Q through an optimization problem that minimizes the estimation error. As a result, an initial value for the Q is systematically available, avoiding the tedious trial-and-error approach commonly used. In order to apply the proposed method, a quaternion representation is used to describe the dynamic equations of the system attitude. As a result, the filter design and implementation are significantly simplified since this representation eliminates costly trigonometric functions. An implementation of the resulting algorithm is performed in a 6-DOF Stewart platform, a particular application representing a large class of industrial manipulators that could benefit from the proposed approach.

This paper is organized as follows. Section 2 introduces the problem at hand and describes the quaternion notation as it is used in the development of the fusion algorithm. While the mathematical models used in the filter are derived in Sect. 3, the algorithm itself is described in Sect. 4 and subsequently implemented in the Stewart platform in Sect. 5. Conclusions are drawn in Sect. 6.

Notation: Uppercase characters denote matrices, and vectors are bold lower case. The $n \times n$ zero and identity matrices are given by 0_n and I_n. A^T is the transpose of A. $R_{bo}(\boldsymbol{q})$ is the rotation matrix from frame $\mathcal{F}_o$ to frame $\mathcal{F}_b$ given by the quaternion vector $\boldsymbol{q}$. Unless clearly stated, all vectors are referred to the origin frame $\mathcal{F}_o$.

2 Problem Formulation

This paper uses quaternions to represent rotations in space. In terms of the axis-angle rotation, quaternions can be parameterized as follows:

$$\boldsymbol{q} = \begin{bmatrix} \cos\left(\frac{\psi}{2}\right) \\ \boldsymbol{r}\sin\left(\frac{\psi}{2}\right) \end{bmatrix} = \begin{bmatrix} \eta \\ \boldsymbol{\epsilon} \end{bmatrix}, \quad \boldsymbol{q} \in \mathbb{R}^4, \tag{1}$$

where $\eta \in \mathbb{R}$ and $\boldsymbol{r}, \boldsymbol{\epsilon} \in \mathbb{R}^3$. The unit vector $\boldsymbol{r}$ describes the direction around which the rotation ψ is performed. By (1), it is clear that the following normalization equation is satisfied

$$\eta^2 + \boldsymbol{\epsilon}^\mathsf{T}\boldsymbol{\epsilon} = 1. \tag{2}$$

From $\boldsymbol{q} \in \mathbb{R}^4$, a corresponding rotation matrix readily follows. Assuming $\boldsymbol{q}$ is the orientation of a body reference frame $\mathcal{F}_\mathrm{b}$ with respect to an origin reference frame $\mathcal{F}_\mathrm{o}$, the rotation matrix from $\mathcal{F}_\mathrm{o}$ to $\mathcal{F}_\mathrm{b}$ is given by,

$$R_{\mathrm{bo}}(\boldsymbol{q}) = I_3 - 2\eta S(\boldsymbol{\epsilon}) + 2S(\boldsymbol{\epsilon})^2, \ R_{\mathrm{bo}}(\boldsymbol{q}) \in \mathbb{R}^{3\times 3}, \tag{3}$$

where the operator $S(\cdot)$ is used to represent the vector cross product in matrix form,

$$S(\boldsymbol{z}) = \begin{bmatrix} 0 & -z_3 & z_2 \\ z_3 & 0 & -z_1 \\ -z_2 & z_1 & 0 \end{bmatrix}. \tag{4}$$

Based on the quaternion representation, the dynamic equations of a rigid rotating body are given by the following expressions (Markley & Crassidis, 2014),

$$\begin{aligned} \dot{\boldsymbol{q}}(t) &= \frac{1}{2}\begin{bmatrix} 0 & -\boldsymbol{\omega}(t)^\mathsf{T} \\ \boldsymbol{\omega}(t) & -S(\boldsymbol{\omega}(t)) \end{bmatrix}\boldsymbol{q}(t) \\ J\dot{\boldsymbol{\omega}}(t) &= -S(\boldsymbol{\omega}(t))J\boldsymbol{\omega}(t) + \boldsymbol{\tau}(t) \end{aligned} \tag{5}$$

where $J \in \mathbb{R}^{3\times 3}$ is the inertia matrix and $\boldsymbol{\tau}(t) \in \mathbb{R}^3$ represents the torques applied to the body whose angular velocity is $\boldsymbol{\omega}(t) \in \mathbb{R}^3$.

Furthermore, the equations of motion of the translation dynamics are given by,

$$\begin{aligned} \dot{\boldsymbol{p}}(t) &= \boldsymbol{v}(t) \\ M\dot{\boldsymbol{v}}(t) &= \boldsymbol{F}(t) - \boldsymbol{g} \end{aligned} \tag{6}$$

where $\boldsymbol{p}(t), \ \boldsymbol{v}(t) \in \mathbb{R}^3$ are the position and velocity of the body with inertia matrix $M \in \mathbb{R}^{3\times 3}$ subject to a force $\boldsymbol{F}(t) \in \mathbb{R}^3$ and the gravitational field $\boldsymbol{g} = [0 \ \ 0 \ \ g]^\mathsf{T}$. Here, all variables are descriptions of the body frame with respect to the origin frame.

We consider that three rate-gyros and three accelerometers (i.e., an IMU), and a monocular camera are available to estimate the body's pose. Therefore, the following assumptions are made concerning these sensors:

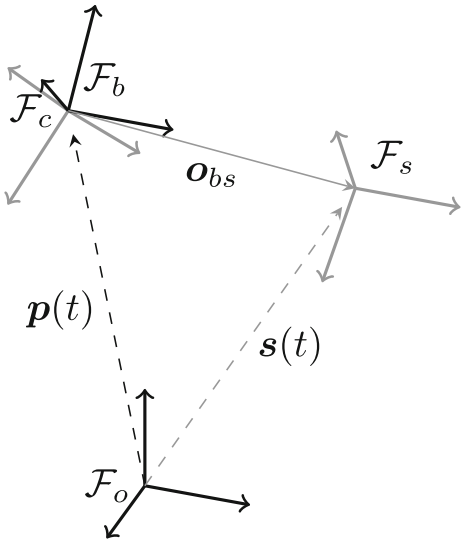

Fig. 1 The problem being addressed is that of estimating the body reference frame $\mathcal{F}_\mathrm{b}$ with respect to the origin $\mathcal{F}_\mathrm{o}$ given measurements from the IMU sensor ($\mathcal{F}_\mathrm{s}$) and the camera ($\mathcal{F}_\mathrm{c}$)

A1 Reference frames: both the IMU sensor and the camera are rigidly attached to the body and their respective reference frames $\mathcal{F}_\mathrm{s}$ and $\mathcal{F}_\mathrm{c}$ are known: with respect to $\mathcal{F}_\mathrm{b}$, the IMU sensor $\mathcal{F}_\mathrm{s}$ has displacement $\boldsymbol{o}_\mathrm{bs}$ and orientation R_sb, and the camera $\mathcal{F}_\mathrm{c}$ is coincident with orientation R_cb (see Fig. 1).

A2 Features: the environment has a set of features whose positions with respect to $\mathcal{F}_\mathrm{o}$ are known.

A3 Image Processing: the camera sensor provides the coordinates (pixels) of the incidence of the features on the image, i.e., image processing is not addressed in this paper.

Note that there is no requirement that the features remain in the camera's field of view. In their absence, it is expected that the EKF may continue estimating the pose based solely on the IMU. Furthermore, we assume no knowledge whatsoever is provided concerning the inertial parameters J and M, which may be time-varying. Likewise, no knowledge is assumed for the system inputs $\boldsymbol{F}$ and $\boldsymbol{\tau}$. Given the above, the problem to be addressed in this paper is defined as follows.

Problem definition: Given the IMU and the camera measurements, estimate the position $\boldsymbol{p}(t)$ and orientation $\boldsymbol{q}(t)$ of the 6-DOF rigid body subject to assumptions A1, A2, and A3.

3 Models for the EKF

The extended Kalman filter (EKF) relies on the system model to make predictions, and on the sensor models and data to make corrections. While these models are described in the current section, their role in the EKF algorithm is detailed in the next one.

3.1 System Model

The sensors and the algorithm will run in discrete time at constant sample rates. We assume the IMU provides data with sample time T, at a higher frequency than the camera, which provides data at a sample time $T_\mathrm{c} = \kappa T$ for a positive integer κ. Therefore, the algorithm itself will run at a frequency T and accommodate the camera data only when available. Thus, the system equations are discretized with sample time T leading to:

$$\begin{aligned}
\boldsymbol{q}_{k+1} &= \boldsymbol{q}_k + \frac{T}{2}\begin{bmatrix} 0 & -\boldsymbol{\omega}_k^\mathsf{T} \\ \boldsymbol{\omega}_k & -S(\boldsymbol{\omega}_k) \end{bmatrix}\boldsymbol{q}_k \\
\boldsymbol{\omega}_{k+1} &= \boldsymbol{\omega}_k + T\boldsymbol{\alpha}_k \\
\boldsymbol{p}_{k+1} &= \boldsymbol{p}_k + T\boldsymbol{v}_k \\
\boldsymbol{v}_{k+1} &= \boldsymbol{v}_k + T\boldsymbol{a}_k
\end{aligned} \tag{7}$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{a}$ represent the unknown angular and linear accelerations of the body.

Remark 1 In order to accommodate the lack of knowledge of the acceleration variables $\boldsymbol{\alpha}$ and $\boldsymbol{a}$ (both the inertia matrices J and M and the inputs $\boldsymbol{F}$ and $\boldsymbol{\tau}$ are unknown), their dynamics will be modeled as random walks during the Kalman Filter implementation, as detailed next section.

3.2 Gyroscope Model

The gyroscope sensor provides a vector of angular velocities $\boldsymbol{\omega}^{\boldsymbol{g}} \in \mathbb{R}^3$ with equal magnitude of that of the body $\boldsymbol{\omega}$ but with different orientation. Given R_sb, the constant rotation matrix from $\mathcal{F}_\mathrm{b}$ to $\mathcal{F}_\mathrm{s}$, the gyroscope model is,

$$\boldsymbol{\omega}_k^g = R_\mathrm{sb}\boldsymbol{\omega}_k + \boldsymbol{b}_k^g + \boldsymbol{\delta}_k^g, \tag{8}$$

where $\boldsymbol{b}^g \in \mathbb{R}^3$ is the time-varying sensor bias, and $\boldsymbol{\delta}^g \in \mathbb{R}^3$ is the measurement noise sampled from a normal distribution with zero mean and covariance $R_\mathrm{g} \in \mathbb{R}^{3\times 3}$, i.e.,

$$\mathcal{P}\big(\boldsymbol{\delta}_k^g\big) \sim \mathcal{N}\big(0, R_\mathrm{g}\big). \tag{9}$$

Given a set of measurements, an estimate for R_g may be determined experimentally. Notice that it is due to the inclusion of the bias in Eq. (8) that we may assume that the noise $\boldsymbol{\delta}_k^g$ has zero mean. However, since the bias vector $\boldsymbol{b}_k^g$ is unknown and time-varying, it will be necessary to provide an online estimate of this parameter in order to use Eq. (8).

3.3 Accelerometer Model

Refer to Fig. 1 in order to derive the accelerometer model. The vector $\boldsymbol{s}(t)$ that describes the position of the IMU sensor with respect to the origin is given by,

$$\boldsymbol{s}(t) = \boldsymbol{p}(t) + R_{\text{bo}}\big(\boldsymbol{q}(t)\big)\boldsymbol{o}_{\text{bs}}, \tag{10}$$

and its linear velocity is,

$$\dot{\boldsymbol{s}}(t) = \dot{\boldsymbol{p}}(t) + R_{\text{bo}}\big(\boldsymbol{q}(t)\big)S\big(\boldsymbol{\omega}(t)\big)\boldsymbol{o}_{\text{bs}}. \tag{11}$$

By computing the time derivative of the above, recalling that $\boldsymbol{o}_{\text{bs}}$ is a constant, one finds the linear acceleration of the IMU with respect to $\mathcal{F}_{\text{o}}$,

$$\ddot{\boldsymbol{s}}(t) = \boldsymbol{a}(t) + R_{\text{bo}}(\boldsymbol{q}(t))\mathcal{H}(\dot{\boldsymbol{\omega}}(t), \boldsymbol{\omega}(t))\boldsymbol{o}_{\text{bs}}, \tag{12}$$

where $\mathcal{H}(\dot{\boldsymbol{\omega}}(t), \boldsymbol{\omega}(t)) = S(\boldsymbol{\omega}(t))^2 + S(\dot{\boldsymbol{\omega}}(t))$.

Finally, one must take into account the orientation of the sensor, the gravitational field $\boldsymbol{g}$, and the bias and noise inherent to the measurements. The final (discrete time) model of the accelerometer is given by,

$$\boldsymbol{a}_k^{acc} = R_{\text{sb}}\left(R_{\text{bo}}(\boldsymbol{q}_k)(\boldsymbol{a}_k - \boldsymbol{g}) + \mathcal{H}(\boldsymbol{\alpha}_k, \boldsymbol{\omega}_k)\boldsymbol{o}_{\text{bs}}\right) + \boldsymbol{b}_k^a + \boldsymbol{\delta}_k^a. \tag{13}$$

As with the gyroscope, the rotation matrix product $R_{\text{sb}}R_{\text{bo}}(\boldsymbol{q}_k)$ describes the orientation of the sensor with respect to the origin. Furthermore, $\boldsymbol{b}_k^a \in \mathbb{R}^3$ is the time-varying accelerometer bias, and $\boldsymbol{\delta}_k^a \in \mathbb{R}^3$ is the measurement noise sampled from a normal distribution with zero mean and covariance $R_{\text{acc}} \in \mathbb{R}^{3\times3}$, i.e.,

$$\mathcal{P}(\boldsymbol{\delta}_k^a) \sim \mathcal{N}(0, R_{\text{acc}}). \tag{14}$$

Given a set of measurements, an estimate for R_{acc} may also be determined experimentally. Also, an online estimate of the accelerometer bias will have to be produced.

3.4 Pinhole Camera Model

The pinhole camera measurements use n_ξ known features (landmarks) whose positions with respect to the global reference frame are given by $\boldsymbol{\xi}^i \in \mathbb{R}^3, i = 1, \ldots n_\xi$, and whose time-varying projections on the image plane (in pixels) are denoted by $\boldsymbol{m}_k^i \in \mathbb{R}^2$. Based on Mariottini and Prattichizzo (2005), the image formation of the points $\boldsymbol{\xi}^i$ on a camera with orientation $\boldsymbol{q}_k$ and position $\boldsymbol{p}_k$ is

$$\lambda\tilde{\boldsymbol{m}}_k^i = K_{\text{c}}R_{\text{cb}}R_{\text{bo}}\big(\boldsymbol{q}_k\big)\left(\boldsymbol{\xi}^i - \boldsymbol{p}_k\right), \quad \forall\ \lambda > 0. \tag{15}$$

Here, $\tilde{\boldsymbol{m}}^i$ is the feature projection in the camera frame [pixels] ($\tilde{\boldsymbol{m}}^i = [\boldsymbol{m}_i\ 1]^\mathsf{T}$), and $\boldsymbol{\xi}^i$ and $\boldsymbol{p}_k$ are, respectively, the feature and camera positions with respect to $\mathcal{F}_{\text{o}}$. The scaling factor $\lambda > 0$ accounts for the fact that any point in a line passing through the focal point of a pinhole camera will be projected on the same pixel $\boldsymbol{m}^i$. Furthermore, K_{c} is a matrix containing the intrinsic parameters of the camera, represented as follows:

$$K_c = \begin{bmatrix} fs_x & fs_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{16}$$

The entries of the matrix K_c have the following geometric interpretation: o_x and o_y are the x- and y-coordinates of the principal point, measured in pixels; fs_x and fs_y are the size of unit length in horizontal and vertical pixels, and fs_θ is the skew of the pixel. These parameters are usually obtained experimentally via some optimization procedure (Ma et al., 2004).

Since $S(\tilde{\boldsymbol{m}}^i)\tilde{\boldsymbol{m}}^i = 0$, the unknown parameter λ may be eliminated by left multiplying the matrix $S(\tilde{\boldsymbol{m}}^i)$ to equation (15). By doing so, and adding the noise inherent to the measurements, the following relation is obtained,

$$0_{2\times1} = \mathbb{C}\,S\big(\tilde{\boldsymbol{m}}_k^i\big)K_{\text{c}}R_{\text{cb}}R_{\text{bo}}\big(\boldsymbol{q}_k\big)\big(\boldsymbol{\xi}^i - \boldsymbol{p}_k\big) + \boldsymbol{\delta}_k^{\text{ci}}, \tag{17}$$

where $\mathbb{C} = [I_2\ 0_{2\times1}]$ is an auxiliary matrix that eliminates the last line of Eq. (15), and $\boldsymbol{\delta}_k^{ci} \in \mathbb{R}^2$ is the measurement noise associated with the camera. Once again, the noise is assumed sampled from a normal distribution with zero mean and covariance $R_{\text{c}} \in \mathbb{R}^{2\times2}$, i.e.,

$$\mathcal{P}\big(\boldsymbol{\delta}_k^{ci}\big) \sim \mathcal{N}\big(0, R_{\text{c}}\big),\ i = 1, \ldots n_\xi. \tag{18}$$

Equation (17) will be used to incorporate camera measurements during the filter correction step.

4 EKF-based Sensor Fusion

4.1 Prediction Step

The state vector $\boldsymbol{x}_k$ includes all necessary variables that must be estimated in order to accommodate the model and the different sensors. Besides the variables of interest, namely the position and orientation of the camera, the linear and angular velocities and accelerations, along with the sensor biases, must be included in $\boldsymbol{x}_k$, leading to the following definition:

$$\boldsymbol{x}_k^\mathsf{T} := \left[\hat{\boldsymbol{q}}_k^\mathsf{T}\ \hat{\boldsymbol{\omega}}_k^\mathsf{T}\ \hat{\boldsymbol{\alpha}}_k^\mathsf{T}\ \hat{\boldsymbol{p}}_k^\mathsf{T}\ \hat{\boldsymbol{v}}_k^\mathsf{T}\ \hat{\boldsymbol{a}}_k^\mathsf{T}\ \hat{\boldsymbol{b}}_k^{g\mathsf{T}}\ \hat{\boldsymbol{b}}_k^{a\mathsf{T}}\right] \in \mathbb{R}^{25}. \tag{19}$$

Given the above state vector, the state transition equation incorporates Eq. (7) and is fully described by,

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k) + \boldsymbol{\delta}_k^x, \tag{20}$$

where

$$\boldsymbol{f}(\boldsymbol{x}_k) = \boldsymbol{x}_k + T \begin{bmatrix} \mathbb{F}(\hat{\boldsymbol{\omega}}_k)\hat{\boldsymbol{q}}_k \\ \hat{\boldsymbol{\alpha}}_k \\ 0_{3\times 1} \\ \hat{\boldsymbol{v}}_k \\ \hat{\boldsymbol{a}}_k \\ 0_{9\times 1} \end{bmatrix}, \tag{21}$$

and

$$\mathbb{F}(\hat{\boldsymbol{\omega}}_k) = \frac{1}{2}\begin{bmatrix} 0 & -\hat{\boldsymbol{\omega}}_k^{\mathsf{T}} \\ \hat{\boldsymbol{\omega}}_k & -S(\hat{\boldsymbol{\omega}}_k) \end{bmatrix}. \tag{22}$$

The random variable $\boldsymbol{\delta}_k^x \in \mathbb{R}^{25}$ is a zero-mean Gaussian random vector that models the uncertainty introduced by the state transition equation. The block-diagonal covariance matrix governs this uncertainty $Q \in \mathbb{R}^{25\times 25}$, which is a tuning parameter While the sensor covariance matrices may be computed by a set of measurements, Q is notoriously harder to be determined and is usually the result of a tiresome trial-and-error approach. Section 5.2, however, will show a systematic way of computing this matrix via an optimization algorithm that explores an external sensor during the tunning process.

It must be mentioned that Eq. (21) describes the system accelerations and sensor biases as constants. Obviously, this is a coarse approximation of the real dynamics of these variables and is only reasonable when their rate of change is slow when compared to the filter dynamics. Nevertheless, the uncertainty related to these estimates is incorporated in matrix Q.

Finally, the EKF filter requires a linearization of the state transition Eq. (21), which is given by the Jacobian matrix $F_k \in \mathbb{R}^{25\times 25}$,

$$F_k := \frac{\partial \boldsymbol{f}(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_k} = I_{25} + T\ \mathrm{diag}\{\Phi(\boldsymbol{x}_k), \Psi, 0_{6\times 6}\}, \tag{23}$$

where

$$\Phi(\boldsymbol{x}_k) = \begin{bmatrix} \mathbb{F}(\hat{\boldsymbol{\omega}}_k) & \mathbb{G}(\hat{\boldsymbol{q}}_k) & 0_{4\times 3} \\ 0_{3\times 4} & 0_3 & I_3 \\ 0_{3\times 4} & 0_3 & 0_3 \end{bmatrix}, \quad \Psi = \begin{bmatrix} 0_3 & I_3 & 0_3 \\ 0_3 & 0_3 & I_3 \\ 0_3 & 0_3 & 0_3 \end{bmatrix}, \tag{24}$$

and,

$$\mathbb{G}(\hat{\boldsymbol{q}}_k) = \frac{1}{2}\begin{bmatrix} -\hat{\boldsymbol{\epsilon}}_k^{\mathsf{T}} \\ \hat{\eta}_k I + S(\hat{\boldsymbol{\epsilon}}_k) \end{bmatrix}, \tag{25}$$

where it is recalled that $\hat{\boldsymbol{q}}_k^{\mathsf{T}} = [\hat{\eta}_k \hat{\boldsymbol{\epsilon}}_k^{\mathsf{T}}]$.

4.2 Correction Step

A sensor vector $\boldsymbol{z}_k$ with variable dimension is used in the correction step of the filter, and it is defined in the following form:

$$\boldsymbol{z}_k = \begin{bmatrix} \boldsymbol{\omega}_k^g \\ \boldsymbol{a}_k^{acc} \\ 0_{2i\times 1} \end{bmatrix} \in \mathbb{R}^{n_h}, \tag{26}$$

where $i = 0, \ldots n_\xi$ denotes the number of features that are visible to the camera at instant k. Note that the dimension $n_h = 6 + 2i$ of the sensor vector varies according to this number of features in the camera field of view. For most of the time, however, $n_h = 6$ since in between the arrival of camera sensor data the filter will rely on the IMU alone.

The nonlinear time-varying output equation that describes how $\boldsymbol{z}_k$ is reconstructed from $\boldsymbol{x}_k$, i.e.,

$$\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{\delta}_k^z, \tag{27}$$

is built from Eqs. (8), (13) and (17):

$$\boldsymbol{h}(\boldsymbol{x}_k) = \begin{bmatrix} R_{sb}\hat{\boldsymbol{\omega}}_k + \hat{\boldsymbol{b}}_k^g \\ R_{sb}\left(R_{bo}(\hat{\boldsymbol{q}}_k)(\hat{\boldsymbol{a}}_k - \boldsymbol{g}) + \mathcal{H}(\hat{\boldsymbol{\alpha}}_k, \hat{\boldsymbol{\omega}}_k)\boldsymbol{o}_{bs}\right) + \hat{\boldsymbol{b}}_k^a \\ \mathbb{C}\, S(\tilde{\boldsymbol{m}}_k^1) K_c R_{cb} R_{bo}(\hat{\boldsymbol{q}}_k)(\boldsymbol{\xi}^1 - \hat{\boldsymbol{p}}_k) \\ \vdots \\ \mathbb{C}\, S(\tilde{\boldsymbol{m}}_k^{n_\xi}) K_c R_{cb} R_{bo}(\hat{\boldsymbol{q}}_k)(\boldsymbol{\xi}^{n_\xi} - \hat{\boldsymbol{p}}_k) \end{bmatrix}, \tag{28}$$

$$\boldsymbol{\delta}_k^z = \begin{bmatrix} \boldsymbol{\delta}_k^{g\mathsf{T}} & \boldsymbol{\delta}_k^{a\mathsf{T}} & \boldsymbol{\delta}_k^{ci\mathsf{T}} \end{bmatrix}^{\mathsf{T}}. \tag{29}$$

In here, terms $\boldsymbol{h}(\boldsymbol{x}_k)$, $\boldsymbol{\delta}_k^z \in \mathbb{R}^{n_h}$ have the same varying dimension of $\boldsymbol{z}_k$.

The covariance matrix R_k for all sensor measurement noises $\boldsymbol{\delta}_k^z$ is composed by the covariance matrices of each individual sensor. It is defined in the following block diagonal form,

$$R_k = \mathrm{diag}\{R_{acc}, R_g, R_c, \ldots R_c\}, \in \mathbb{R}^{n_h\times n_h}. \tag{30}$$

As with $\boldsymbol{z}_k$, the dimension of R_k varies according to the number of features i in the camera field of view.

In order to compute the Kalman gain, the correction step needs the Jacobian matrix $H_k \in \mathbb{R}^{n_h\times 25}$ given by the partial derivatives of $\boldsymbol{h}(\boldsymbol{x}_k)$ with respect to $\boldsymbol{x}_k$,

$$H_k := \frac{\partial \boldsymbol{h}(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_k} = \begin{bmatrix} H_k^{\hat{\boldsymbol{q}}_k} & H_k^{\hat{\boldsymbol{\omega}}_k,\hat{\boldsymbol{\alpha}}_k} & H_k^{\hat{\boldsymbol{p}}_k} & H_k^{\hat{\boldsymbol{v}}_k,\hat{\boldsymbol{a}}_k} & H_k^{\hat{\boldsymbol{b}}_k} \end{bmatrix}. \quad (31)$$

The result of these computations is provided below:

$$H^{\hat{\boldsymbol{q}}_k} := \begin{bmatrix} 0_{3\times 4} \\ R_{sb}\mathbb{J}(\hat{\boldsymbol{q}}_k, \hat{\boldsymbol{a}}_k - \boldsymbol{g}) \\ \mathbb{C}\, S(\tilde{\boldsymbol{m}}_k^1) K_c R_{cb} \mathbb{J}(\hat{\boldsymbol{q}}_k, \boldsymbol{\xi}^1 - \hat{\boldsymbol{p}}_k) \\ \vdots \\ \mathbb{C}\, S(\tilde{\boldsymbol{m}}_k^i) K_c R_{cb} \mathbb{J}(\hat{\boldsymbol{q}}_k, \boldsymbol{\xi}^i - \hat{\boldsymbol{p}}_k) \end{bmatrix}, \quad (32)$$

$$H_k^{\hat{\boldsymbol{\omega}}_k,\hat{\boldsymbol{\alpha}}_k} := \begin{bmatrix} R_{sb} & 0_3 \\ R_{sb}R_{bo}(\hat{\boldsymbol{q}}_k)\mathbb{W}(\hat{\boldsymbol{\omega}}_k, \boldsymbol{o}_{bs}) & -R_{sb}R_{bo}(\hat{\boldsymbol{q}}_k)S(\boldsymbol{o}_{bs}) \\ 0_{2\times 3} & 0_{2\times 3} \\ \vdots & \vdots \\ 0_{2\times 3} & 0_{2\times 3} \end{bmatrix}, \quad (33)$$

$$H_k^{\hat{\boldsymbol{p}}_k} := \begin{bmatrix} 0_3 \\ 0_3 \\ -\mathbb{C}\, S(\tilde{\boldsymbol{m}}_k^1) K_c R_{cb} R_{bo}(\hat{\boldsymbol{q}}_k) \\ \vdots \\ -\mathbb{C}\, S(\tilde{\boldsymbol{m}}_k^{n_\xi}) K_c R_{cb} R_{bo}(\hat{\boldsymbol{q}}_k) \end{bmatrix}, \quad (34)$$

$$H_k^{\hat{\boldsymbol{v}}_k,\hat{\boldsymbol{a}}_k} := \begin{bmatrix} 0_3 & 0_3 \\ 0_3 & R_{sb}R_{bo}(\hat{\boldsymbol{q}}_k) \\ 0_{2\times 3} & 0_{2\times 3} \\ \vdots & \vdots \\ 0_{2\times 3} & 0_{2\times 3} \end{bmatrix}, \quad (35)$$

where $H_k^{\hat{\boldsymbol{b}}_k} = [I_6\ 0_{6\times 2n_\xi}]^\mathsf{T}$. The auxiliary functions $\mathbb{J}(\boldsymbol{q}, \boldsymbol{u})$ and $\mathbb{W}(\boldsymbol{y}, \boldsymbol{u})\ \forall \boldsymbol{y}, \boldsymbol{u} \in \mathbb{R}^3$ are defined by

$$\mathbb{J}(\boldsymbol{q}, \boldsymbol{u}) := \frac{\partial R_{bo}(\boldsymbol{q})\boldsymbol{u}}{\partial \boldsymbol{q}} = \frac{\partial (I_3 - 2\eta S(\boldsymbol{\epsilon}) + 2S^2(\boldsymbol{\epsilon}))\boldsymbol{u}}{\partial \begin{bmatrix} \eta \\ \boldsymbol{\epsilon} \end{bmatrix}}$$
$$= 2[-S(\boldsymbol{\epsilon})\boldsymbol{u}\ \ S(\boldsymbol{u})\eta + \mathbb{W}(\boldsymbol{\epsilon}, \boldsymbol{u})] \quad (36)$$

and

$$\mathbb{W}(\boldsymbol{y}, \boldsymbol{u}) := \frac{\partial S^2(\boldsymbol{y})\boldsymbol{u}}{\partial \boldsymbol{y}} = \boldsymbol{y}\boldsymbol{u}^\mathsf{T} - 2\boldsymbol{u}\boldsymbol{y}^\mathsf{T} + \boldsymbol{y}^\mathsf{T}\boldsymbol{u} I_3. \quad (37)$$

Given the definitions presented so far, the extended Kalman filter algorithm used for sensor fusion is described in Algorithm 1.[1] The algorithm follows the very traditional and well-known structure of an extended Kalman filter with two main peculiarities: the correction step starts (in line 7) by adjusting the dimension of the associated matrices according to the available visualdata, this will result in a Kalman gain K_k of variable dimension, as computed in line 9; furthermore, the estimated quaternion after the prediction and correction steps must be normalized[2] in order to satisfy the property stated in Eq. (2). While this simple normalization is sufficient for the current study, other methods are shown in Markley (2003).

Algorithm 1 EKF-based sensor fusion algorithm

1: **function** EKF($\boldsymbol{\omega}_k^g$, $\boldsymbol{a}_k^{acc}$, $\boldsymbol{m}_k^i$, R_k, Q)
2: **persistent** $\boldsymbol{x}_k$, P_k

Prediction Step:
3: Compute F_k from $\boldsymbol{x}_k$ as in (23);
4: $\boldsymbol{x}_k^- = \boldsymbol{f}(\boldsymbol{x}_k)$; % using (21)
5: $P_k^- = F_k P_k F_k^\mathsf{T} + Q$;
6: Normalize the quaternion in $\boldsymbol{x}_k^-$;

Correction Step:
7: Determine the dimension n_h based on available features;
8: Compute H_k from $\boldsymbol{x}_k^-$ as in (31);
9: $K_k = P_k^- H_k^\mathsf{T}(H_k P_k^- H_k^\mathsf{T} + R_k)^{-1}$;
10: $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k^- + K_k(z_k - \boldsymbol{h}(\boldsymbol{x}_k^-))$; % using (26) and (28)
11: $P_{k+1} = (I - K_k H_k)P_k^-$;
12: Normalize the quaternion in $\boldsymbol{x}_{k+1}$;
13: **end function**

The computational burden of this algorithm is centered in line 6 where the Kalman gain is computed from an inverse matrix. Since the size of this matrix will vary according to the number of features available for correction, the velocity of the algorithm may be compromised. If that is the case, an upper bound regarding the number of features being used may be imposed. Nevertheless, with up-to-date computational power, the extended Kalman filter may be embedded in inexpensive processors even for the fusion of a large number of sensors.

5 Implementation via Data-driven Tuning

This section will apply the proposed algorithm to a six degrees of freedom Stewart platform. The goal is to compute the position and orientation of the center of the platform given data from an IMU and a camera. Conversely, it is also possible to compute these variables from encoders positioned at the platform motors, a method that requires the solution of a nonlinear optimization problem not trivially implementable online. Nevertheless, this second method will be detailed and plays the role of *ground truth* for comparison purposes.

[1] In Algorithm 1, $\boldsymbol{x}_k^-$ and $\boldsymbol{P}_k^-$ denote *a priori* estimates, i.e., prior to the application of the correction step.

[2] The normalization of a quaternion in $\boldsymbol{x}$ is computed as $\boldsymbol{q} \leftarrow \boldsymbol{q}/\|\boldsymbol{q}\|$, where $\|\cdot\|$ stands for the Euclidian norm and $\boldsymbol{q}$ represents the quaternion component of $\boldsymbol{x}$ as in (19).

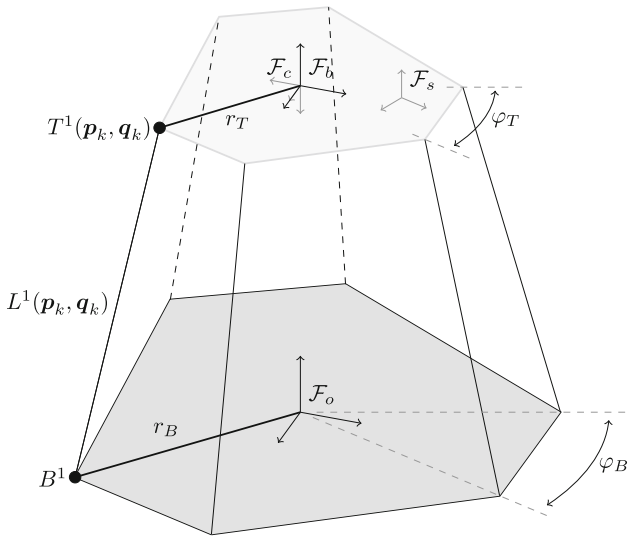

Fig. 2 The Stewart platform with its coordinate systems. The top hexagon has position $\boldsymbol{p}_k$ and orientation $\boldsymbol{q}_k$ due to the variable lengths $L^i(\boldsymbol{p}_k, \boldsymbol{q}_k)$

5.1 Estimating the Stewart Pose from Encoder Measurements

The Stewart platform is a parallel mechanism whose kinematics is determined by the relation between the pose of the top platform and its linear actuators lengths. Therefore, the forward kinematics problem—that is, the computation of the platform pose from actuators lengths—is framed as a geometric problem resulting in high degree polynomials with multiple solutions, which must be computed numerically (Cardona, 2015).

In particular, consider the Stewart platform represented in Fig. 2 and take note of two reference frames: the origin (or inertial) reference frame $\mathcal{F}_o$ centered at the base, and the time-varying body reference frame $\mathcal{F}_b$, with position $\boldsymbol{p}(t)$ and orientation $\boldsymbol{q}(t)$ attached to the center of the top hexagon. The manipulator itself has $i = 1, \ldots, 6$ linear actuators, which concede to the platform its six degrees of freedom. These actuators are attached to joints geometrically defined by B^i and T^i, positioned at the vertices of hexagons inscribed in circles with radius r_B and r_T, respectively. Consequently, the joints are arranged in pairs 120° apart from each other. As shown in the figure, angles φ_B and φ_T define the length of each edge of the base and top hexagons such that the hexagon vertices may be expressed as:

$$B^i = \begin{bmatrix} r_B \cos(\lambda_i) \\ r_B \sin(\lambda_i) \\ 0 \end{bmatrix}, \quad {}^bT^i = \begin{bmatrix} r_T \cos(\upsilon_i) \\ r_T \sin(\upsilon_i) \\ 0 \end{bmatrix}, \tag{38}$$

where the superscript b in ${}^bT^i$ states that these vectors are referred to the body coordinate frame $\mathcal{F}_b$, and

$$\lambda_i = \begin{cases} \frac{i\pi}{3} - \frac{\varphi_B}{2}, & i = (1, 3, 5), \\ \lambda_{i-1} - \varphi_B, & i = (2, 4, 6), \end{cases} \tag{39}$$

with similar definition for υ_i. Therefore, parameters r_B, r_T, φ_B and φ_T define the geometric structure of any given Stewart platform.

The variable length of the ith actuator computed from the geometric parameters of the platform, denoted by $\hat{L}^i(\boldsymbol{p}_k, \boldsymbol{q}_k)$, is given by the Euclidean distance between its joints. In the origin reference frame, the base joints are constant, but the top joints are time-varying, hence,

$$\begin{aligned} \hat{L}^i(\boldsymbol{p}_k, \boldsymbol{q}_k) &= ||T^i(\boldsymbol{p}_k, \boldsymbol{q}_k) - B^i|| \\ &= \sqrt{\left(T^i(\boldsymbol{p}_k, \boldsymbol{q}_k) - B^i\right)^\mathsf{T}\left(T^i(\boldsymbol{p}_k, \boldsymbol{q}_k) - B^i\right)}, \end{aligned} \tag{40}$$

with $T^i(\boldsymbol{p}_k, \boldsymbol{q}_k) = \boldsymbol{p}_k + R_{ob}(\boldsymbol{q}_k)\, {}^bT^i$.

The numerical method consists in finding $\boldsymbol{\theta}_k := [\hat{\boldsymbol{p}}_k{}' \ \hat{\boldsymbol{q}}_k{}']$ that minimize the error between the actuator position L^i given by the encoders and the actuator position $\hat{L}^i(\hat{\boldsymbol{p}}_k, \hat{\boldsymbol{q}}_k)$ given by equation (40). The optimization problem is defined as

$$\mathbf{P1}: \quad \boldsymbol{\theta}_k = \arg\min_{\boldsymbol{\theta}_k} \sum_{i=1}^{6} \left(L^i - \hat{L}^i(\boldsymbol{\theta}_k)\right)^2.$$

The values of the body pose are retrieved by $\boldsymbol{\theta}_k = [\hat{\boldsymbol{p}}_k^\mathsf{T} \ \hat{\boldsymbol{q}}_k^\mathsf{T}]^\mathsf{T}$. Since this is a nonlinear optimization problem, and nonconvex, at every iteration the algorithm takes the solution $\boldsymbol{\theta}_{k-1}$ as the starting point for the computation of $\boldsymbol{\theta}_k$. An estimate is provided as the starting point when the solution of $\boldsymbol{\theta}_0$ is sought. In addition, this approach will reduce the minimization problem number of possible solutions, as the estimate values in two consecutive iterations are expected to be close to each other. The positiveness of $\boldsymbol{\theta}_k$ is guaranteed using the function $log(\boldsymbol{\theta}_k)$ in the optimization problem.

Since this optimization problem requires several minutes of computation to solve a trajectory of a few seconds, it is not practical to be used in real-time but may provide a solution to which the EKF can be compared. Furthermore, it also provides a form of tuning the covariance matrices of the EKF, as detailed next section.

5.2 Selection of the Covariance Matrices

This section will show how to compute the EKF covariance matrices from experimental data, assuming that the model noise is stationary. One experiment is performed where data

from the accelerometer, gyroscope, camera and encoders are gathered. The covariance matrices will be computed such that the estimation error is minimized for this particular trajectory. Later on, results comparing the filter performance to different trajectories will be presented.

The sensor covariance matrix, as defined in (30), is a diagonal matrix containing the covariance information of the IMU and the camera. Since the uncertainty related to these sensors is primarily given by noise, it is straightforward to compute the entries for this matrix based on experimental data collected by the sensors.

The difficult task is related to the tuning of the covariance matrix $Q \in \mathbb{R}^{25\times 25}$ related to model uncertainty. In order to do so, we collect the data of the encoders and compute the trajectory by means of the optimization problem detailed in the previous section. Not only the position and orientation vectors are computed, but also their respective velocities and accelerations, that is, the whole state $\boldsymbol{x}_k^{opt}$, $k = 1, 2 \ldots n_k$. With that information in hand, we can set up another optimization problem that uses the parameter $\boldsymbol{\delta}_k^x$ and Eq. (21) in order to estimate Q. The problem consists in the minimization of the distance between the trajectory $\boldsymbol{x}_k$, $k = 1, 2 \ldots n_k$ estimated by **Algorithm 1**, and the trajectory $\boldsymbol{x}_k^{opt}$, $k = 1, 2 \ldots n_k$ computed by the optimization problem **P1**. The argument to this optimization problem is the covariance matrix Q parameterized in the following form:

$$\begin{aligned} Q := \operatorname{diag} \big\{ & q_{\boldsymbol{q}} I_4, q_{\boldsymbol{\omega}} I_3, q_{\boldsymbol{\alpha}} I_3, \ldots \\ & \ldots q_{\boldsymbol{p}} I_3, q_{\boldsymbol{v}} I_3, q_{\boldsymbol{a}} I_3, q_{\boldsymbol{b}^g} I_3, q_{\boldsymbol{b}^a} I_3 \big\}, \end{aligned} \tag{41}$$

such that the matrix Q is constrained to a diagonal format and only eight scalars need to be estimated by the optimization algorithm.

Given the above, we define the following optimization problem:

$$\mathbf{P2}: \; Q = \arg\min_{Q} \sum_{k=1}^{n_k} \left(\boldsymbol{x}_k - \boldsymbol{x}_k^{opt} \right)^{\mathsf{T}} \Omega^{-1} \left(\boldsymbol{x}_k - \boldsymbol{x}_k^{opt} \right),$$

where n_k is the trajectory length and Ω is a scaling diagonal matrix defined such that its entries are the square of the maximum associated error.

Remark 2 As **P1**, **P2** is also nonconvex. Nevertheless, our true aim with these optimization problems is to provide a systematic way to search for adequate (not necessarily optimal) values to the covariance matrix Q, instead of employing a crude trial-and-error manual procedure.

5.3 Experimental Results

It is now possible to apply the proposed EKF algorithm along with optimization problems **P1** and **P2** to the six degrees of freedom Stewart platform depicted in Fig. 3. The platform is described by $r_{\mathrm{B}} = 0.35$ m, $r_{\mathrm{T}} = 0.25$ m, $\varphi_{\mathrm{B}} = \frac{\pi}{6}$ and $\varphi_T = \frac{\pi}{2}$, and is equipped with an LSM6DS3H IMU from STMicroelectronics, running at 104 Hz, and a Logitech C270 downward pointing web camera running at 20.8 Hz, i.e., one-fifth the IMU frequency. Four colored markers (features) are placed in known locations on the platform base, as depicted in Fig. 3. The camera focal was considered as the origin of body frame and correspondent IMU offset with respect to the point is according to $o_{\mathrm{bs}} = [-0.0026 \;\; -0.0005 \;\; -0.0140]^{\mathsf{T}}$ m.

Fig. 3 The experimental setup of the Stewart platform

Three datasets are collected from three different trajectories: DataSet 01 is used to compute the covariance matrix Q from problem **P2**, DataSet 02 shows the filter performance while tracking a trajectory where no features are lost, and DataSet 03 shows the performance of the filter when subject to feature loss. All plots depict the trajectory reconstructed from the encoders via optimization problem **P1** (dashed lines), the trajectories estimated by the proposed algorithm (full lines) and the associated error (bottom plots). Both **P1** and **P2** were solved using a *Nelder–Mead* algorithm, as implemented in the `fminsearch` in *Matlab*. In order to improve visualization, orientation plots are depicted in terms of Euler angles: roll $\alpha(t)$, pitch $\beta(t)$ and yaw $\gamma(t)$.

Figure 4 depicts the trajectory from DataSet 01 from which the matrix Q was tuned. This trajectory was designed to excite all states in order to aid the optimization problem. Furthermore, given the fact that the sensors biases had insignificant variations, their covariances were manually set

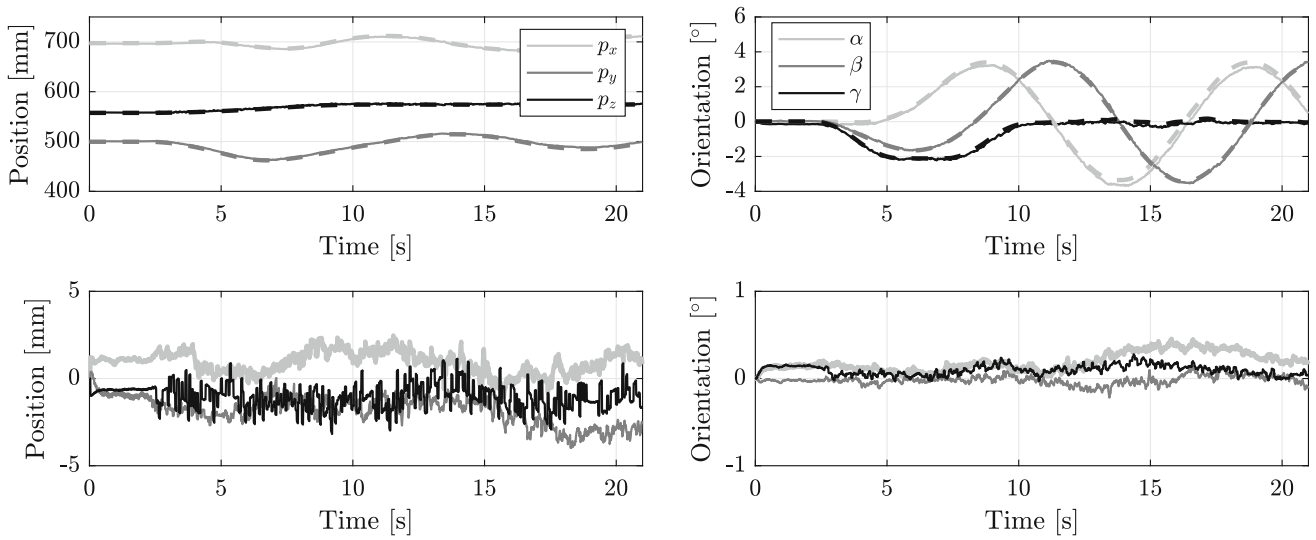

Fig. 4 Optimization results for tuning the model covariance matrix Q. Top plot: real (dashed line) and estimated (full lines) trajectories. Bottom plot: associated errors

Table 1 Covariances associated with the filter

Description	Value	Description	Value
r_q	1.0869×10^{-11}	r_p	$2.5911 \times 10^{+4}$
r_ω	$2.8780 \times 10^{+4}$	r_v	$6.0022 \times 10^{+4}$
r_α	1.1824×10^{-17}	r_a	1.4907×10^{-13}

to approximately zero in order to reduce the optimization variables to be tuned. The resulting model covariance matrix Q is described in Table 1, and the sensor covariances, also obtained experimentally, are given by,

$$\begin{aligned} R_{\text{acc}} &= 10^{-2} \times \text{diag}\{0.337,\ 0.294,\ 1.249\}, \\ R_{\text{g}} &= 10^{-3} \times \text{diag}\{0.1423,\ 0.1378,\ 0.1630\}, \\ R_{\text{c}} &= 10^{-3} \times I_2. \end{aligned} \tag{42}$$

After this tuning step, it is possible to run the EKF to different trajectories in order to validate its performance. The resulting estimation applied to DataSet 02 is depicted in Fig. 5, where one sees the filter tracking the platform movements as expected. A thorough analysis is given in Table 2 where the mean, standard deviation σ and maximum values of the errors are depicted. It is clear from this table that the

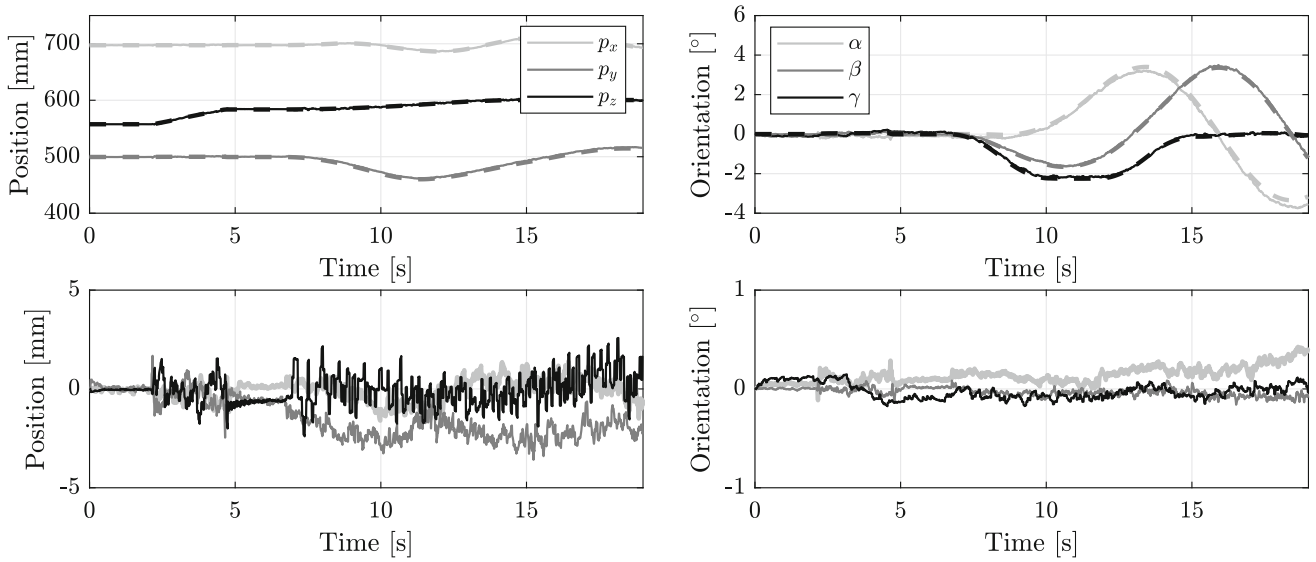

Fig. 5 Validation of the filter. Top plots: real (dashed line) and estimated (full lines) trajectories. Bottom plot: associated errors

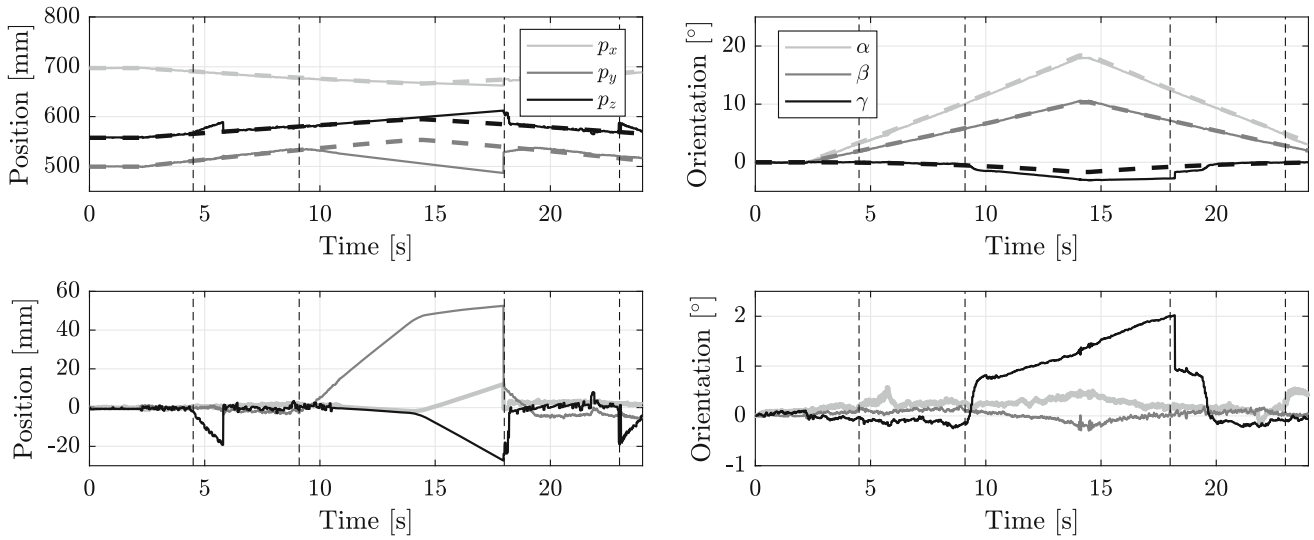

Fig. 6 Filter performance in the presence of feature loss. The first two vertical dashed lines indicate the time instants when features were lost, the next two indicate when they were recovered. Top plots: real (dashed line) and estimated (full lines) trajectories. Bottom plot: associated errors

Table 2 Statistical analysis of the errors

Description	Error position (mm)		Euler angle errors (°)	
	DataSet 01 (Tuning)	DataSet 02 (Validation)	DataSet 01 (Tuning)	DataSet 02 (Validation)
Mean x	1.3171	−0.0708	0.1025	0.0457
Mean y	−0.6491	−0.4659	0.0064	−0.0331
Mean z	−1.2413	−0.3072	0.1096	−0.0218
σ_x	0.5641	0.5000	0.0866	0.0560
σ_y	0.6383	0.7060	0.0718	0.0417
σ_z	0.7613	0.7434	0.0585	0.0749
Max($\|x\|$)	2.5661	1.5661	0.3045	0.2592
Max($\|y\|$)	2.5194	2.4274	0.1964	0.1715
Max($\|z\|$)	3.3437	2.5601	0.2832	0.1686

filter was able to track the platform with maximum errors below 2.6 mm and 0.26°.

Feature loss was imposed in the third trajectory employing an exaggerated inclination in the roll and pitch axes. The platform was tilted until all features were lost, and then it returned to the original position recovering visual information. Feature loss causes the filter to rely solely on its inertial sensors and has damaging consequences to the filter performance: the platform's orientation is estimated from the gravitational field and the integration of the gyroscope; the position is estimated from the double integration of the accelerometer. As expected, error drift is seen in this case, particularly in the position vector and yaw orientation—recall that roll and pitch may be estimated from the gravitational field. Figure 6 shows the estimation drifting from the correct values as soon as the features are lost: which occur two features at a time in the first two vertical dashed lines in the plots. The following dashed lines indicate the time instants when the features were recovered. It is important to emphasize that the filter converges to the correct pose estimation as soon as the features are available.

6 Conclusions

This paper has devised a quaternion-based extended Kalman filter algorithm for the pose estimation of a 6-DOF rigid body. Inertial measurements were fused with visual information such that a high bandwidth low-cost solution was available. A data-driven tuning approach for the model covariance matrix Q was proposed and the resulting algorithm was implemented in a 6-DOF Stewart platform. Experimental results have shown the efficacy of the sensor fusion algorithm when subject to low-cost sensors. The filter performance was also tested against feature loss, indicating that a reasonable estimation is possible provided camera occlusion occurs for

short periods of time. Our proposal for future works is to investigate more efficient ways to solve the proposed tuning optimization problems and, possibly, to also include an online tuning process that may take into account a time-varying term Q, thus dealing with nonstationary model noise (Silva et al., 2018; Yuen et al., 2013).

References

Araguás, G., Paz, C., Gaydou, D., & Paina, G. P. (2015). Quaternion-based orientation estimation fusing a camera and inertial sensors for a hovering UAV. *Journal of Intelligent and Robotic Systems, 77*, 37–53.

Cardona, M. (2015). A new approach for the forward kinematics of general Stewart–Gough platforms. In *Proceedings of the 2015 IEEE thirty fifth central American and panama convention (CONCAPAN XXXv)*.

Colonnier, F., Vedova, L. D., & Orchard, G. (2021). ESPEE: Event-based sensor pose estimation using an extended Kalman filter. *Sensors, 21*(23), 7840.

He, Y., Sun, W., Huang, H., Liu, J., Fan, H., & Sun, J. (2020). PVN3D: A deep point-wise 3D keypoints voting network for 6DoF pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering, 82*, 35–45.

Li, S., Li, D., Zhang, C., Wan, J., & Xie, M. (2020). RGB-D image processing algorithm for target recognition and pose estimation of visual servo system. *Sensors, 20*(2), 430.

Ma, Y., Soatto, S., Košecká, J., & Sastry, S. S. (2004). *An invitation to 3-D vision: From imagem to geometric models*. Springer.

Mariottini, G. L., & Prattichizzo, D. (2005). EGT for multiple view geometry and visual servoing: Robotics vision with pinhole and panoramic cameras. *IEEE Robotics & Automation Magazine, 12*(4), 26–39.

Markley, F. L. (2003). Attitude error representations for Kalman filtering. *Journal of Guidance, Control, and Dynamics, 26*(2), 311–317.

Markley, F. L., & Crassidis, J. L. (2014). *Fundamentals of spacecraft attitude determination and control*. Springer.

Nützi, G., Weiss, S., Scaramuzza, D., & Siegwart, R. (2011). Fusion of IMU and vision for absolute scale estimation in monocular SLAM. *Journal of Intelligent & Robotic Systems, 61*(1–4), 287–299.

Ratz, S., Dymczyk, M., Siegwart, R., & Dubé, R. (2020). Oneshot global localization: Instant lidar-visual pose estimation. In *2020 IEEE international conference on robotics and automation (ICRA)* (pp. 5415–5421).

Schönberger, J. L., Pollefeys, M., Geiger, A., & Sattler, T. (2018). Semantic visual localization. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*.

Sciavicco, L., & Siciliano, B. (2012). *Modelling and control of robot manipulators*. Springer.

Silva, J. G., Filho, J. O. D. A. L., & Fortaleza, E. L. F. (2018). Adaptive extended Kalman filter using exponencial moving average. In *Proceedings of the 9th IFAC symposium on robust control design* (Vol. 51, pp. 208–211).

Sveier, A., & Egeland, O. (2021). Dual quaternion particle filtering for pose estimation. *IEEE Transactions on Control Systems Technology, 29*(5), 2012–2025.

Tayebi, A., & McGilvray, S. (2006). Attitude stabilization of a VTOL quadrotor aircraft. *IEEE Transactions on Control Systems Technology, 14*(3), 562–571.

Tong, X., Li, Z., Han, G., Liu, N., Su, Y., Ning, J., & Yang, F. (2018). Adaptive EKF based on HMM recognizer for attitude estimation using MEMS MARG sensors. *IEEE Sensors Journal, 18*(8), 3299–3310.

Yang, C., Liu, Y., & Zell, A. (2020). RCPNet: Deep-learning based relative camera pose estimation for UAVs. In *2020 International conference on unmanned aircraft systems (ICUAS)* (pp. 1085–1092).

Yuen, K. V., Liang, P. F., & Kuok, S. C. (2013). Online estimation of noise parameters for Kalman filter. *Structural Engineering and Mechanics, 47*(3), 361–381.

Zhang, H., & Ye, C. (2020). A visual positioning system for indoor blind navigation. In *2020 IEEE international conference on robotics and automation (ICRA)* (pp. 9079–9085).

Zhang, Y., Bian, C., & Gao, J. (2020). An unscented Kalman filter-based visual pose estimation method for underwater vehicles. In *2020 3rd international conference on unmanned systems (ICUS)* (pp. 663–667).